Title

MULTI-PRED: A Software Module for Predictive Modeling of Coupled Multi-Physics Systems - User Manual

Authors
Cacuci, Dan G
Fang, Ruixian
Badea, Madalina C

Publication Date
2023-12-12

Peer reviewed

MULTI-PRED: A Software Module for Predictive Modeling of Coupled Multi-Physics Systems

MULTI-PRED User's Manual

February 28, 2018

Dan G. Cacuci, Ruixian Fang, Madalina C. Badea

Center for Nuclear Science and Energy
University of South Carolina
541 Main Street, Columbia, SC 29208, USA
cacuci@cec.sc.edu
fang@cec.sc.edu
badea@cec.sc.edu

Abstract

This User's Manual describes the code module MULTI-PRED, written in FORTRAN which implements the methodology for "predictive modeling of coupled multi-physics systems (PM-CMPS)" formulated by Cacuci (2014). This methodology fully takes into account the coupling terms between the systems but requires only the computational resources that would be needed to perform predictive modeling on each system separately. The PM-CMPS methodology uses the maximum entropy principle to construct an optimal approximation of the unknown a priori distribution based on a priori known mean values and uncertainties characterizing the experimental and computational parameters and results of interest responses, called for the multi-physics models under consideration. This "maximum entropy" a priori distribution is combined, using Bayes' theorem, with the "likelihood" provided by the multi-physics simulation models to obtain a formal posterior distribution. Subsequently, the posterior distribution thus obtained is evaluated using the saddle-point method to obtain analytical expressions for the optimally predicted values for the multi-physics models parameters and responses along with corresponding reduced uncertainties. Noteworthy, the predictive modeling methodology for the coupled systems is constructed such that the systems can be considered sequentially rather than simultaneously, while preserving exactly the same results as if the systems were treated simultaneously. Consequently, very large coupled systems, which could perhaps exceed available computational resources if treated simultaneously, can be treated with the PM-CMPS methodology presented in this work sequentially and without any loss of generality or information, requiring just the resources that would be needed if the systems were treated sequentially. Three illustrative demonstration problems are also provided. The first problem presents the application of the PM-CMPS methodology to a simple particle diffusion problem which admits a closed-form analytical solution which facilitates a rapid understanding of this methodology and its predicted results. The second demonstration problem presents the application of the PM-CMPS methodology to the problem of inverse prediction, from detector responses in the presence of counting uncertainties, of the thickness of a homogeneous slab of material containing uniformly distributed gamma-emitting sources, for optically thin and thick slabs. This problem highlights the essential role played by the relative uncertainties (or, conversely, accuracies) of measured and computed responses. The third demonstration problem presents the application of the PM-CMPS methodology to the F-area cooling towers at the Savannah River National Lab. This problem demonstrates that the PM-CMPS

methodology reduces the predicted response uncertainties not only at locations where measurements are available, but also at locations where measurements are not available.

1 INTRODUCTION

Results of measurements inevitably reflect the influence of experimental errors, imperfect instruments, and imperfectly known calibration standards. Around any reported experimental value, therefore, there always exists a range of values that may also be plausibly representative of the true but unknown value of the measured quantity. On the other hand, computations are also imperfect, since they are afflicted by errors stemming from numerical procedures, uncertain model parameters, boundary and initial conditions, and/or imperfectly known physical processes or problem geometry. Therefore, nominal values for experimentally measured or computed quantities are insufficient, by themselves, for applications. The quantitative uncertainties accompanying the measurements and computations are also needed, along with the respective nominal values. Extracting "best estimate" values for model parameters and predicted results (responses), together with "best estimate" uncertainties for these parameters and responses requires the combination of experimental and computational data and their uncertainties. This combination process often requires reasoning from incomplete, error-afflicted, and occasionally discrepant information.

The discrepancies between experimental and computational results provide the basic motivation for performing quantitative model verification, validation, qualification and predictive estimation. Loosely speaking, "code verification" means "are you solving the mathematical model correctly?" "Code validation" means "does the model represent reality?" "Code qualification" means certifying that a proposed simulation/design methodology/system satisfies all performance and safety specifications. Model validation addresses issues of (a) assessing model accuracy when several system response quantities have been measured and compared and (b) comparing system response quantities from multiple realizations of the experiment with computational results that are characterized by probability distributions. Model validation and qualification require selected benchmarking, including sensitivity and uncertainty analyses.

Predictive modeling commences with the identification and characterization of uncertainties from all steps in the sequence of modeling and simulation processes that leads to a computational model prediction. This includes: (a) data error or uncertainty (input data such as cross sections, model parameters such as reaction-rate coefficients, initial conditions, boundary conditions, and forcing functions such as external loading), (b) numerical discretization error, and (c) uncertainty in (e.g., lack of knowledge of) the processes being modeled. The result of the predictive modeling analysis is a probabilistic description of possible future outcomes based on all recognized errors and uncertainties.

Predictive modeling combines/assimilates computational and experimental information using response sensitivities to perform model calibration, model extrapolation, and estimation of the validation domain. Model calibration addresses the integration of experimental data for the purpose of updating the data of the computer model. Important components include the estimation of discrepancies in the data, and of the biases between model predictions and experimental data. The state-of-the-art of model calibration is fairly well developed, but current methods are still hampered in practice by the significant computational effort required. Reducing the computational effort is paramount, and methods based on adjoint models show great promise in his regard. Model extrapolation addresses the prediction uncertainty in new environments or conditions of interest, including both untested parts of the parameter space and higher levels of system complexity in the validation hierarchy. Extrapolation of models and the resulting increase of uncertainty are poorly understood, particularly the estimation of uncertainty that results from nonlinear coupling of two or more physical phenomena that were not coupled in the existing validation database. The quantification of the validation domain underlying the models of interest requires estimation of contours of constant uncertainty in the high-dimensional space that characterizes the application of interest. In practice, this involves the identification of areas where the predictive estimation of uncertainty meets specified requirements for the performance, reliability, or safety of the system of interest.

Cacuci and Ionescu-Bujor (2010a) have recently published a comprehensive methodology for predicting best-estimate values for model responses and parameters (following the assimilation experimental data and simultaneous calibration of model parameters and responses), along with reduced predicted uncertainties, for large-scale nonlinear time-dependent systems. This predictive modeling methodology generalizes and significantly extends the "data adjustment" methods customarily used in nuclear engineering, as well as those underlying the so-called 4D-VAR data assimilation procedures in the geophysical sciences (see, e.g., Lahoz et al, 2010, and Cacuci et al., 2013), and also provides a quantitative indicator, constructed from sensitivity and covariance matrices, for determining the consistency (agreement or disagreement) among the a priori computational and experimental data (parameters and responses). This consistency indicator measures (in the corresponding metric) the deviations between the experimental and nominally computed responses. Note that this consistency indicator can be evaluated directly from the originally given data (i.e., given parameters and responses, together with their original uncertainties), once the response sensitivities have been computed by either the forward or the adjoint sensitivity analysis procedure, as developed by Cacuci (1981a, 1981b, 2003; see also: Cacuci et al, 1980). When the numerical value of this consistency indicator is close to unity (per degrees of freedom), the respective data is considered to be consistent "within the respective error norms" (usually under quadratic loss). However, when the numerical value of this consistency indicator differs considerably from unity, which usually occurs when the distance between the mean values of two (sets of) measurements or two (sets of) computations of the same quantity are larger than the sum of the two accompanying standard deviations, the respective (measured of computed) data points are considered to be inconsistent or discrepant. This means that there is a nonzero probability that two non-discrepant (i.e. belonging to the same distribution) measurements that are separated by more than 2 standard deviations (thus giving the appearance of being discrepant!) could actually occur in practice. Recall that for a Gaussian sampling distribution, the probability that two equally precise measurements would be separated by more than two standard deviations is 15.7%. However, this probability is rather small; therefore it is much more likely that apparently discrepant data actually indicate the presence of unrecognized errors. Methods for treating unrecognized errors have been developed by Cacuci and Ionescu-Bujor (2010b), by applying the maximum entropy principle under quadratic loss to the discrepant data. Once the inconsistent data, if any, is discarded, the predictive modeling methodology by Cacuci and Ionescu-Bujor (2010a) predicts best-estimate values for parameters and predicted responses, as well as best-estimate reduced uncertainties (i.e., "smaller" values for the variance-covariance matrices) for the predicted best-estimate parameters and responses.

The predictive modeling methodology of Cacuci and Ionescu-Bujor (2010a) has been successfully applied by M.C. Badea et al (2012), and by Cacuci and Arslan (2014) to calibrate time-dependent model parameters and boundary conditions for a large-scale LWR core thermal-hydraulics simulations models codes using the BFBT international benchmark measurements. Furthermore, Arslan and Cacuci (2014) have also applied the predictive modeling methodology by Cacuci and

Ionescu-Bujor (2010a) to calibrate selected parameters in commercial CFD codes for predictive modeling of liquid-sodium experiments.

The predictive modeling methodology of Cacuci and Ionescu-Bujor (2010a) has been generalized from a single multi-physics system to two or more coupled multi-physics systems by Cacuci (2014). Noteworthy, the mathematical methodology underlying this "predictive modeling of coupled multi-physics systems (PM-CMPS)" is constructed such that the systems can be treated sequentially rather than simultaneously, while preserving exactly the same results as if the systems had been treated simultaneously. Consequently, very large coupled systems, which could perhaps exceed available computational resources if treated simultaneously, can be treated with the PM-CMPS methodology sequentially, without any loss of generality or information, requiring just the resources that would be needed if the systems were treated simultaneously. This new PM-CMPS methodology is presented in Chapter 2. We use the maximum entropy principle to construct an optimal approximation of the unknown a priori distribution for the a priori known mean values and uncertainties characterizing the parameters and responses for both multi-physics models. This approximate a priori distribution is subsequently combined using Bayes' theorem with the "likelihood" provided by the multi-physics computational models. Finally, the posterior distribution is evaluated using the saddle-point method to obtain analytical expressions for the optimally predicted values for the parameters and responses of both multi-physics models, along with corresponding reduced uncertainties. Chapter 3 discusses the significance and new possible applications of the new methodology, while Chapter 4 offers a summary and conclusions.

2 PREDICTIVE MODELING OF COUPLED MULTI-PHYSICS SYSTEMS (PM-CMPS)

2.1 Introduction

This Chapter presents the mathematical formalism underlying the *Predictive Modeling of Coupled Multi-Physics Systems* PM-CMPS methodology conceived by Cacuci (2014). The general mathematical framework of the PM-CMPS methodology is presented in the following sequence: Subsection 2.2.1 models the a priori information for two multi-physics models; Subsection 2.2.2 presents the application of the Maximum Entropy Principle to construct an optimal approximation of the unknown a priori distribution from the a priori known mean values and uncertainties characterizing the parameters and responses for both multi-physics models. This approximate a priori distribution is subsequently combined using Bayes' theorem with the "likelihood" provided by the multi-physics computational models, as presented in Subsection 2.2.3. This Subsection also presents the application of the saddle-point method on the posterior distribution to obtain analytical expressions for the optimally predicted values for the parameters and responses of both multi-physics models, along with corresponding reduced uncertainties. Section 2.3 presents several important particular cases of the PM-CMPS methodology, which are often encountered in practice.

2.2 Mathematical Framework

2.2.1 A Priori Information for Two Multi-Physics Models

Consider a multi-physics model, henceforth called "Model A" comprising N_α system (model) parameters α_n. Model A is used to compute results, henceforth called responses, which can also be measured experimentally. Consider now a second physical system, henceforth called "Model B," comprising N_β system (model) parameters β_m , and which is also used to compute responses that can be measured experimentally. Model A and Model B are considered to be coupled. In reactor analysis and design, for example, Model A may comprise the neutron transport and depletion equations which are coupled to Model B which computes the thermal-hydraulics conservation (mass, momentum, energy) equations.

Consider next that there are N_r experimentally measured responses r_i associated mostly, but not necessarily exclusively, with Model A. Furthermore, consider also that there are N_q experimentally measured responses q_j associated mostly, but not necessarily exclusively, with Model B. For example, measurement of reaction rates and power (or flux) distributions could be considered to be responses of type r_i, while measurements of flow rates and temperature distributions could be considered responses of type q_j. In the same spirit, cross sections can be considered to be model parameters of type α_n, while heat transfer correlations can be considered model parameters of type β_m . Parameters modeling the geometry of the system (e.g., rod and assembly dimensions, core dimensions), for example, could be considered to belong to either type of model parameters (i.e., either α_n or β_m), since they affect both the neutron transport equation and the thermal-hydraulics conservation equations.

In practice, the values of the parameters α_n and β_m are determined experimentally. Therefore, these parameters cannot be known exactly, but can be considered to behave stochastically, obeying some probability distribution function which is seldom known. Such stochastic quantities will be called *variates* in this work; thus, the parameters α_n and β_m, as well as the measured responses r_i and q_j are variates. To simplify the mathematical derivations to follow in this section, the model parameters α_n will be considered to constitute the components of the (column) vector $\boldsymbol{\alpha}$, defined as

$$\boldsymbol{\alpha} \triangleq \left(\alpha_1, ..., \alpha_{N_\alpha}\right) \text{ ,} \tag{2.1}$$

while the model parameters β_m will be considered to constitute the components of the (column) vector $\boldsymbol{\beta}$ defined as

$$\boldsymbol{\beta} \triangleq \left(\beta_1, ..., \beta_{N_\beta}\right). \tag{2.2}$$

By convention, all of the vectors considered in this work (e.g., $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$) are column vectors. A dagger ($\dagger$) will be used to denote "transposition;" thus the quantities $\boldsymbol{\alpha}^\dagger$ and $\boldsymbol{\beta}^\dagger$ are row vectors; Similarly, the N_r experimentally measured responses r_i will be considered to be components of the column vector

$$\mathbf{r} \triangleq \left(r_1, ..., r_{N_r}\right), \tag{2.3}$$

while the N_q experimentally measured responses q_j will be considered to be components of the column vector

$$\mathbf{q} \triangleq \left(q_1, ..., q_{N_q}\right). \tag{2.4}$$

Most generally, the parameters α_n and β_m, as well as the responses r_i and q_j can be considered to obey some a priori probability distribution function $P(\boldsymbol{\alpha},\boldsymbol{\beta},\mathbf{r},\mathbf{q})$. For large-scale systems, as customarily encountered in practice, the probability distribution $P(\boldsymbol{\alpha},\boldsymbol{\beta},\mathbf{r},\mathbf{q})$ cannot possibly be known. The information usually available in practice comprises the mean values of the model parameters and responses together with the corresponding uncertainties (standard deviations and, occasionally, correlations) about the respective mean values. For notational simplicity, angular brackets, $\langle f \rangle$, will be used to denote the integral of the quantity $f(\boldsymbol{\alpha},\boldsymbol{\beta},\mathbf{r},\mathbf{q})$ over the joint probability distribution $P(\boldsymbol{\alpha},\boldsymbol{\beta},\mathbf{r},\mathbf{q})$, i.e.,

$$\langle f \rangle \triangleq \int f(\boldsymbol{\alpha},\boldsymbol{\beta},\mathbf{r},\mathbf{q}) P(\boldsymbol{\alpha},\boldsymbol{\beta},\mathbf{r},\mathbf{q}) \, d\boldsymbol{\alpha} \, d\boldsymbol{\beta} \, d\mathbf{r} \, d\mathbf{q}. \tag{2.5}$$

Using the above convention, the mean values of the model parameters α_n will be denoted using the superscript "zero", i.e., as $\alpha_n^0 \triangleq \langle \alpha_n \rangle$; these mean values are considered to constitute the components of the vector $\boldsymbol{\alpha}^0$ defined as

$$\boldsymbol{\alpha}^0 \triangleq \left(\alpha_1^0, ..., \alpha_{N_\alpha}^0\right). \tag{2.6}$$

Similarly, the mean values of the parameters β_n are considered to be known, and will be denoted as $\beta_n^0 \triangleq \langle \beta_n \rangle$. These mean values are considered to be the components of the vector $\boldsymbol{\beta}^0$ defined as

$$\boldsymbol{\beta}^0 \triangleq \left(\beta_1^0, ..., \beta_{N_\beta}^0\right). \tag{2.7}$$

The parameters' second-order central moments, namely the standard deviations and correlations, are also considered to be known. For the parameters α_n, the second-order central moments are the components of covariance matrices $\mathbf{C}_{\alpha\alpha}^{(N_\alpha \times N_\alpha)}$ defined as

$$\mathbf{C}_{\alpha\alpha}^{(N_\alpha \times N_\alpha)} \triangleq \left[\operatorname{cov}\left(\alpha_i,\alpha_j\right)\right]_{N_\alpha \times N_\alpha} \triangleq \left\langle \left(\alpha_i - \alpha_i^0\right)\left(\alpha_j - \alpha_j^0\right)\right\rangle_{N_\alpha \times N_\alpha}; \; i,j = 1,\ldots,N_\alpha, \tag{2.8}$$

while the second-order central moments (i.e., the standard deviations and correlations) for the parameters β_m form covariance matrices $\mathbf{C}_{\beta\beta}^{(N_\beta \times N_\beta)}$ defined as

$$\mathbf{C}_{\beta\beta}^{(N_\beta \times N_\beta)} \triangleq \left[\operatorname{cov}\left(\beta_i,\beta_j\right)\right]_{N_\beta \times N_\beta} \triangleq \left\langle \left(\beta_i - \beta_i^0\right)\left(\beta_j - \beta_j^0\right)\right\rangle_{N_\beta \times N_\beta}; \;\; i,j = 1,\ldots,N_\beta . \tag{2.9}$$

In general, the components of the vectors $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ may be correlated. The correlations among the parameters $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are quantified by correlation matrices $\mathbf{C}_{\alpha\beta}^{(N_\alpha \times N_\beta)}$ defined as

$$\mathbf{C}_{\alpha\beta}^{(N_\alpha \times N_\beta)} \triangleq \left\langle \left(\boldsymbol{\alpha} - \boldsymbol{\alpha}^0\right)\left(\boldsymbol{\beta} - \boldsymbol{\beta}^0\right)^\dagger \right\rangle \triangleq \left[\mathbf{C}_{\beta\alpha}^{(N_\beta \times N_\alpha)}\right]^\dagger . \tag{2.10}$$

The experimentally measured responses are also considered to be characterized by known mean measured values and measured variances and covariances. Thus, for the N_r experimentally measured responses r_i, the mean measured values will be denoted as r_i^m, and will be considered to constitute the components of the vector $\mathbf{r}^m$ defined as

$$\mathbf{r}^m \triangleq \left(r_1^m,\ldots,r_{N_r}^m\right), \quad r_i^m \triangleq \left\langle r_i \right\rangle, \; i = 1,\ldots,N_r, \tag{2.11}$$

while the corresponding measured covariance matrix, denoted as $\mathbf{C}_{rr}^{(N_r \times N_r)}$, is defined as

$$\mathbf{C}_{rr}^{(N_r \times N_r)} \triangleq \left\langle \left(r_i - r_i^m\right)\left(r_j - r_j^m\right)\right\rangle_{N_r \times N_r}, \;\; i,j = 1,\ldots,N_r. \tag{2.12}$$

Similarly, the N_q experimentally measured responses q_j are characterized by mean measured values, denoted as q_j^m, and constituting the components of the vector $\mathbf{q}^m$ defined as

$$\mathbf{q}^m \triangleq \left(q_1^m,\ldots,q_{N_q}^m\right), \;\; q_j^m \triangleq \left\langle q_j \right\rangle, \; j = 1,\ldots,N_q, \tag{2.13}$$

and by the measured covariance matrix $\mathbf{C}_{qq}^{(N_q \times N_q)}$ defined as

$$\mathbf{C}_{qq}^{(N_q \times N_q)} \triangleq \left\langle \left(q_i - q_i^m\right)\left(q_j - q_j^m\right)\right\rangle_{N_q \times N_q}, \quad i,j = 1,\ldots,N_q. \tag{2.14}$$

Furthermore, the responses $\mathbf{r}$ and $\mathbf{q}$ may also be correlated; such correlations would be quantified by correlation matrices defined as

$$\mathbf{C}_{rq}^{(N_r \times N_q)} \triangleq \left\langle (\mathbf{r}-\mathbf{r}^m)(\mathbf{q}-\mathbf{q}^m)^\dagger \right\rangle \triangleq \left[\mathbf{C}_{qr}^{(N_q \times N_r)} \right]^\dagger . \tag{2.15}$$

In the most general case, correlations my also exist among all parameters and responses. Such correlations would be quantified through matrices defined as follows:

$$\mathbf{C}_{\alpha r}^{(N_\alpha \times N_r)} \triangleq \left\langle (\boldsymbol{\alpha}-\boldsymbol{\alpha}^0)(\mathbf{r}-\mathbf{r}^m)^\dagger \right\rangle \triangleq \left[\mathbf{C}_{r\alpha}^{(N_r \times N_\alpha)} \right]^\dagger , \tag{2.16}$$

$$\mathbf{C}_{\alpha q}^{(N_\alpha \times N_q)} \triangleq \left\langle (\boldsymbol{\alpha}-\boldsymbol{\alpha}^0)(\mathbf{q}-\mathbf{q}^m)^\dagger \right\rangle \triangleq \left[\mathbf{C}_{q\alpha}^{(N_q \times N_\alpha)} \right]^\dagger , \tag{2.17}$$

$$\mathbf{C}_{\beta r}^{(N_\beta \times N_r)} \triangleq \left\langle (\boldsymbol{\beta}-\boldsymbol{\beta}^0)(\mathbf{r}-\mathbf{r}^m)^\dagger \right\rangle \triangleq \left[\mathbf{C}_{r\beta}^{(N_r \times N_\beta)} \right]^\dagger , \tag{2.18}$$

$$\mathbf{C}_{\beta q}^{(N_\beta \times N_q)} \triangleq \left\langle (\boldsymbol{\beta}-\boldsymbol{\beta}^0)(\mathbf{q}-\mathbf{q}^m)^\dagger \right\rangle \triangleq \left[\mathbf{C}_{q\beta}^{(N_q \times N_\beta)} \right]^\dagger . \tag{2.19}$$

2.2.2 Construction of the A Priori Distribution Function $p(\boldsymbol{\alpha},\boldsymbol{\beta},\mathbf{r},\mathbf{q})$ as the Maximum Entropy Principle Approximation of the True but Unknown A Priori Distribution Function $P(\boldsymbol{\alpha},\boldsymbol{\beta},\mathbf{r},\mathbf{q})$

The quantities defined in Eqs. (2.1) through (2.19) constitute the prior information regarding the uncertain parameters and measured responses in the two-model multi-physics system considered in the previous section. This prior information prescribes the means (i.e., the first-order moments) and covariances (i.e., the second-order moments) of an otherwise unknown distribution function $p(\boldsymbol{\alpha},\boldsymbol{\beta},\mathbf{r},\mathbf{q})$. Mathematically, these means and covariances are functionals of $p(\boldsymbol{\alpha},\boldsymbol{\beta},\mathbf{r},\mathbf{q})$, having the generic form

$$\langle F_k \rangle \triangleq \int p(\mathbf{x}) F_k(\mathbf{x}) d\mathbf{x}, \quad \mathbf{x} \triangleq (\boldsymbol{\alpha},\boldsymbol{\beta},\mathbf{r},\mathbf{q}), \quad d\mathbf{x} \triangleq d\boldsymbol{\alpha}\, d\boldsymbol{\beta}\, d\mathbf{r}\, d\mathbf{q}, \quad k = 1,2,\ldots,K, \tag{2.20}$$

with $F_k(\mathbf{x})$ representing, in turn, the quantities: $(\alpha_n-\alpha_n^0)$, $(\beta_n-\beta_n^0)$, $(r_n-r_n^m)$, $(q_n-q_n^m)$, $(\alpha_i-\alpha_i^0)(\alpha_j-\alpha_j^0)$, $(\beta_i-\beta_i^0)(\beta_j-\beta_j^0)$, $(r_i-r_i^m)(r_j-r_j^m)$, $(q_i-q_i^m)(q_j-q_j^m)$, $(\alpha_i-\alpha_i^0)(\beta_j-\beta_j^0)$, $(\alpha_i-\alpha_i^0)(r_j-r_j^m)$, $(\alpha_i-\alpha_i^0)(q_j-q_j^m)$, $(\beta_i-\beta_i^0)(r_j-r_j^m)$, $(\beta_i-\beta_i^0)(q_j-q_j^m)$, and $(r_i-r_i^m)(q_j-q_j^m)$.

The total number of first- and second-order moments is

$$\begin{aligned}K \triangleq\, & N_\alpha+N_\beta+N_r+N_q+N_\alpha^2+N_\beta^2+N_r^2+N_q^2+(N_\alpha\times N_\beta)+(N_\alpha\times N_r)+(N_\alpha\times N_q)\\ &+(N_\beta\times N_r)+(N_\beta\times N_q)+(N_r\times N_q).\end{aligned} \tag{2.21}$$

An optimal way to approximate the true but unknown probability distribution function $P(\mathbf{x})$ using the information given in Eq. (2.20) is to apply the *maximum entropy formalism*. The maximum entropy formalism enables the determination of an approximate probability distribution function, denoted here as $p(\mathbf{x})$, which approximates the exact but unknown distribution $P(\mathbf{x})$ by maximizing over $p(\mathbf{x})$ the Shannon information entropy, defined as

$$S \triangleq -\int d\mathbf{x}\, p(\mathbf{x}) \ln \frac{p(\mathbf{x})}{m(\mathbf{x})}, \tag{2.22}$$

where $m(\mathbf{x})$ is a prior density that ensures form invariance under change of variable, while satisfying the constraints given in Eq.(2.20). This maximum entropy principle insures that the approximate distribution function $p(\mathbf{x})$ maximizes the optimal compatibility with the available information, namely the constraints given in Eq.(2.20), while simultaneously ensuring minimal spurious information content.

Maximizing the information entropy S over $p(\mathbf{x})$ subject to the constraints expressed by Eq.(2.20) constitutes a variational problem that can be solved by using the method of Lagrange multipliers to obtain a member of the exponential family, namely

$$p(\mathbf{x}) = \frac{1}{Z} m(\mathbf{x}) \exp\left[-\sum_k \lambda_k F_k(\mathbf{x})\right], \tag{2.23}$$

where the quantities λ_k are the Lagrange multipliers. The normalization constant Z in Eq (2.23). is defined as

$$Z \equiv \int d\mathbf{x}\, m(\mathbf{x}) \exp\left[-\sum_k \lambda_k F_k(\mathbf{x})\right]. \tag{2.24}$$

The Lagrange multipliers λ_k must be found directly from the constraints [i.e., using Eqs. (2.20). and (2.23) or from the equivalent equations

$$\langle F_k \rangle = -\frac{\partial}{\partial \lambda_k} \ln Z, \qquad k = 1, 2, \ldots, K, \tag{2.25}$$

which are more convenient if Z can be expressed as an analytic function of the Lagrange parameters.

In the case of discrete distributions, if only the alternatives can be enumerated but the macroscopic data $\langle F_k \rangle$ are not known, then $m(\mathbf{x}) = 1$, and the maximum entropy algorithm described in the foregoing yields the uniform distribution, as would be required by the principle of insufficient reason. Therefore, the maximum entropy principle can be considered as a far-reaching generalization of the principle of insufficient reason, ranging from discrete alternatives with no other information given, to cases with given global or macroscopic information, and also encompassing continuous distributions. Physicists will recognize the maximum entropy algorithm described above as the essence of the Gibbs-formalism for statistical mechanics, where Z is the partition function (or sum over states), carrying all information about the possible states of the system, from which the expected macroscopic parameters can be obtained by differentiation with respect to the Lagrange multipliers. If only the possible energies of a system and the average energy (i.e., the temperature) are given, one finds Gibbs' canonical ensemble, with probabilities proportional to the Boltzmann factors $\exp(-\lambda E_j)$, the Lagrange multiplier λ being essentially the inverse temperature. If, in addition, the average particle number is given, one finds the grand-canonical ensemble, with a second Lagrange multiplier equal to the chemical potential, etc.

Performing the (lengthy but straightforward) computations indicated in Eq (2.25) solving the resulting system of equation for the Lagrange multipliers λ_k, and replacing the resulting expressions in Eq. (2.23) leads to the following expression for $p(\mathbf{x})$:

$$p\left(\mathbf{x}\,|\,\langle\mathbf{x}\rangle,\mathbf{C}\right)d\mathbf{x}=\frac{\exp\left[-\frac{1}{2}\left(\mathbf{x}-\langle\mathbf{x}\rangle\right)^{\dagger}\mathbf{C}^{-1}\left(\mathbf{x}-\langle\mathbf{x}\rangle\right)\right]d\mathbf{x},}{\sqrt{\det\left(2\pi\mathbf{C}\right)}},\ -\infty<x_j<\infty, \tag{2.26}$$

where the dagger ($\dagger$) denotes transposition (Hermitian conjugation of real vectors and matrices), and the matrix $\mathbf{C}$ is defined as

$$\mathbf{C}\triangleq\begin{pmatrix}\mathbf{C}_{\alpha\alpha} & \mathbf{C}_{\alpha\beta} & \mathbf{C}_{\alpha r} & \mathbf{C}_{\alpha q}\\ \mathbf{C}_{\beta\alpha} & \mathbf{C}_{\beta\beta} & \mathbf{C}_{\beta r} & \mathbf{C}_{\beta q}\\ \mathbf{C}_{r\alpha} & \mathbf{C}_{r\beta} & \mathbf{C}_{rr} & \mathbf{C}_{rq}\\ \mathbf{C}_{q\alpha} & \mathbf{C}_{q\beta} & \mathbf{C}_{qr} & \mathbf{C}_{qq}\end{pmatrix},\ with\ \mathbf{x}\triangleq\begin{pmatrix}\boldsymbol{\alpha}\\ \boldsymbol{\beta}\\ \mathbf{r}\\ \mathbf{q}\end{pmatrix},\ \langle\mathbf{x}\rangle\triangleq\begin{pmatrix}\boldsymbol{\alpha}^0\\ \boldsymbol{\beta}^0\\ \mathbf{r}^m\\ \mathbf{q}^m\end{pmatrix}. \tag{2.27}$$

Thus, the foregoing considerations show that, when only mean values and covariances are known, the maximum entropy algorithm yields the Gaussian probability distribution shown in Eq.(2.27) as the most objective probability distribution consistent with the available information. Although all of the above results are valid for $-\infty<x_j<\infty$, these results can also be used for $0<x_j<\infty$ after introduction of a logarithmic scale (which leads to lognormal distributions on the original scale).

Gaussian distributions are often considered appropriate only if many independent random deviations act together so that the central limit theorem is applicable. At other times, Gaussian distributions are invoked for mere convenience, with accompanying warnings about consequences if the true distribution is not Gaussian. The maximum entropy principle cannot eliminate these consequences, but it reassures the data user who is given only mean values and their (co)variances that the corresponding Gaussian is the best choice for all further inferences, whatever the unknown true distribution may happen to be. In contrast to the central limit theorem, the maximum entropy principle is also valid for correlated data.

2.2.3 Construction of the A Posteriori Predicted Mean Values and Covariances for the Given Models (Likelihood Function) and Maximum Entropy Prior Distribution

Consider next that the coupled Models A and B are used to compute the $\left(N_r + N_q\right)$ experimentally measured responses. These computed responses will be considered to be the components of two vectors, denoted as $\mathbf{r}^c\left(\boldsymbol{\alpha},\boldsymbol{\beta}\right) \triangleq \left(r_1^c,...,r_{N_r}^c\right)$ and $\mathbf{q}^c\left(\boldsymbol{\alpha},\boldsymbol{\beta}\right) \triangleq \left(q_1^c,...,q_{N_q}^c\right)$, respectively, where the superscript "*c*" indicates "computed." In principle, the computed responses may depend on some or all of the components of $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$. Consequently, $\mathbf{r}^c\left(\boldsymbol{\alpha},\boldsymbol{\beta}\right)$ and $\mathbf{q}^c\left(\boldsymbol{\alpha},\boldsymbol{\beta}\right)$ are also variates, characterized by probability distribution functions, which cannot, in general, be obtained in explicitly closed forms.

The next step is to combine the experimental and computational information in order to obtain the posterior distribution of $\mathbf{x} \triangleq \left(\boldsymbol{\alpha},\boldsymbol{\beta},\mathbf{r},\mathbf{q}\right)$. This combination is rigorously performed by using Bayes' theorem, in which the (maximum entropy) prior is the Gaussian distribution computed in Eq. (2.26), while the likelihood is provided by the computational models $\mathbf{r}^c\left(\boldsymbol{\alpha},\boldsymbol{\beta}\right)$ and $\mathbf{q}^c\left(\boldsymbol{\alpha},\boldsymbol{\beta}\right)$. When the numerical and/or modeling errors are not explicitly taken into account, but are considered to be amenable to treatment via uncertain model parameters that are included among the components of $\boldsymbol{\alpha}$, the computational models are considered to be "hard constraints" of the form

$$\mathbf{r} = \mathbf{r}^c\left(\boldsymbol{\alpha},\boldsymbol{\beta}\right), \;\; \mathbf{q} = \mathbf{q}^c\left(\boldsymbol{\alpha},\boldsymbol{\beta}\right) \; . \tag{2.28}$$

Needless to say the posterior distribution, which consists of the prior given in Eq (2.26) together with the likelihood expressed by Eq.(2.28), cannot be computed exactly. Nevertheless, the main contribution to the posterior distribution, and, in particular, the main contributions to the posterior distribution's means and covariances, can be obtained by applying the saddle-point method to evaluate the Gaussian prior in Eq.(2.26) subject to the constraints expressed by Eq.(2.28). As is well known, the saddle-point is the point where the gradient of exponent of the Gaussian prior in Eq.(2.26) vanishes subject to the constraints in Eq.(2.28). The method of Lagrange multipliers can be used to determine this saddle-point, by setting to zero the (partial) gradients with respect to $\boldsymbol{\alpha},\boldsymbol{\beta},\mathbf{r},\mathbf{q}$ of the following functional:

$$P(\boldsymbol{\alpha},\boldsymbol{\beta},\mathbf{r},\mathbf{q}) \triangleq -\frac{1}{2}\left(\mathbf{x}-\langle\mathbf{x}\rangle\right)^{\dagger}\mathbf{C}^{-1}\left(\mathbf{x}-\langle\mathbf{x}\rangle\right)+\boldsymbol{\lambda}_r^{\dagger}\left[\mathbf{r}-\mathbf{r}^c(\boldsymbol{\alpha},\boldsymbol{\beta})\right]+\boldsymbol{\lambda}_q^{\dagger}\left[\mathbf{q}-\mathbf{q}^c(\boldsymbol{\alpha},\boldsymbol{\beta})\right], \quad (2.29)$$

where $\boldsymbol{\lambda}_r$ and $\boldsymbol{\lambda}_q$ are vectors of (yet undetermined) Lagrange multipliers of sizes N_r and N_q, respectively. Thus, the saddle point of $P(\boldsymbol{\alpha},\boldsymbol{\beta},\mathbf{r},\mathbf{q})$ is attained at $\mathbf{x}^{pred} \triangleq \left(\boldsymbol{\alpha}^{pred},\boldsymbol{\beta}^{pred},\mathbf{r}^{pred},\mathbf{q}^{pred}\right)$ where the following conditions are simultaneously fulfilled:

$$\nabla_{\lambda_r} P = \mathbf{r}-\mathbf{r}^c(\boldsymbol{\alpha},\boldsymbol{\beta}) = \mathbf{0}; \quad \nabla_{\lambda_q} P = \mathbf{q}-\mathbf{q}^c(\boldsymbol{\alpha},\boldsymbol{\beta}) = \mathbf{0}; \quad (2.30)$$

$$\nabla_{\alpha} P = \mathbf{0};\ \nabla_{\beta} P = \mathbf{0};\ \nabla_{r} P = \mathbf{0};\ \nabla_{q} P = \mathbf{0}. \quad (2.31)$$

The conditions expressed in Eq.(2.30) simply ensure that the saddle-point will satisfy the constraints imposed by the numerical simulation Models A and B. On the other hand, the conditions imposed in Eq.(2.31) can be written in block-matrix form as

$$\begin{pmatrix} \boldsymbol{\alpha}^{pred}-\boldsymbol{\alpha}^{0} \\ \boldsymbol{\beta}^{pred}-\boldsymbol{\beta}^{0} \\ \mathbf{r}^{pred}-\mathbf{r}^{m} \\ \mathbf{q}^{pred}-\mathbf{q}^{m} \end{pmatrix} = \begin{pmatrix} \mathbf{C}_{\alpha\alpha} & \mathbf{C}_{\alpha\beta} & \mathbf{C}_{\alpha r} & \mathbf{C}_{\alpha q} \\ \mathbf{C}_{\alpha\beta}^{\dagger} & \mathbf{C}_{\beta\beta} & \mathbf{C}_{\beta r} & \mathbf{C}_{\beta q} \\ \mathbf{C}_{\alpha r}^{\dagger} & \mathbf{C}_{\beta r}^{\dagger} & \mathbf{C}_{rr} & \mathbf{C}_{rq} \\ \mathbf{C}_{\alpha q}^{\dagger} & \mathbf{C}_{\beta q}^{\dagger} & \mathbf{C}_{rq}^{\dagger} & \mathbf{C}_{qq} \end{pmatrix} \begin{pmatrix} -\mathbf{S}_{r\alpha}^{\dagger}\boldsymbol{\lambda}_r - \mathbf{S}_{q\alpha}^{\dagger}\boldsymbol{\lambda}_q \\ -\mathbf{S}_{r\beta}^{\dagger}\boldsymbol{\lambda}_r - \mathbf{S}_{q\beta}^{\dagger}\boldsymbol{\lambda}_q \\ \boldsymbol{\lambda}_r \\ \boldsymbol{\lambda}_q \end{pmatrix} \quad (2.32)$$

where the matrices $\mathbf{S}_{r\alpha}(\boldsymbol{\alpha}^0,\boldsymbol{\beta}^0)$, $\mathbf{S}_{r\beta}(\boldsymbol{\alpha}^0,\boldsymbol{\beta}^0)$, $\mathbf{S}_{q\alpha}(\boldsymbol{\alpha}^0,\boldsymbol{\beta}^0)$, and $\mathbf{S}_{q\beta}(\boldsymbol{\alpha}^0,\boldsymbol{\beta}^0)$ comprise first-order response-derivatives with respect to the model parameters, computed at the nominal parameter values $(\boldsymbol{\alpha}^0,\boldsymbol{\beta}^0)$, and are defined as follows:

$$\mathbf{S}_{r\alpha}^{N_r\times N_\alpha} \equiv \begin{bmatrix} \frac{\partial r_1}{\partial \alpha_1} & \cdots & \frac{\partial r_1}{\partial \alpha_{N_\alpha}} \\ \vdots & \ddots & \vdots \\ \frac{\partial r_{N_r}}{\partial \alpha_1} & \cdots & \frac{\partial r_{N_r}}{\partial \alpha_{N_\alpha}} \end{bmatrix}, \quad \mathbf{S}_{r\beta}^{N_r\times N_\beta} \equiv \begin{bmatrix} \frac{\partial r_1}{\partial \beta_1} & \cdots & \frac{\partial r_1}{\partial \beta_{N_\beta}} \\ \vdots & \ddots & \vdots \\ \frac{\partial r_{N_r}}{\partial \beta_1} & \cdots & \frac{\partial r_{N_r}}{\partial \beta_{N_\beta}} \end{bmatrix}, \quad (2.33)$$

$$\mathbf{S}_{q\alpha}^{N_q\times N_\alpha}\equiv\begin{bmatrix}\frac{\partial q_1}{\partial\alpha_1} & \cdots & \frac{\partial q_1}{\partial\alpha_{N_\alpha}}\\ \vdots & \ddots & \vdots\\ \frac{\partial q_{N_q}}{\partial\alpha_1} & \cdots & \frac{\partial q_{N_q}}{\partial\alpha_{N_\alpha}}\end{bmatrix},\quad \mathbf{S}_{q\beta}^{N_q\times N_\beta}\equiv\begin{bmatrix}\frac{\partial q_1}{\partial\beta_1} & \cdots & \frac{\partial q_1}{\partial\beta_{N_\beta}}\\ \vdots & \ddots & \vdots\\ \frac{\partial q_{N_q}}{\partial\beta_1} & \cdots & \frac{\partial q_{N_q}}{\partial\beta_{N_\beta}}\end{bmatrix}. \tag{2.34}$$

When written in component form, Eq.(2.32) yields the following relations:

$$\left(\boldsymbol{\alpha}^{pred}-\boldsymbol{\alpha}^0\right)=-\mathbf{C}_{\alpha\alpha}\left(\mathbf{S}_{r\alpha}^\dagger\boldsymbol{\lambda}_r+\mathbf{S}_{q\alpha}^\dagger\boldsymbol{\lambda}_q\right)-\mathbf{C}_{\alpha\beta}\left(\mathbf{S}_{r\beta}^\dagger\boldsymbol{\lambda}_r+\mathbf{S}_{q\beta}^\dagger\boldsymbol{\lambda}_q\right)+\mathbf{C}_{\alpha r}\boldsymbol{\lambda}_r+\mathbf{C}_{\alpha q}\boldsymbol{\lambda}_q \tag{2.35}$$

$$\left(\boldsymbol{\beta}^{pred}-\boldsymbol{\beta}^0\right)=-\mathbf{C}_{\alpha\beta}^\dagger\left(\mathbf{S}_{r\alpha}^\dagger\boldsymbol{\lambda}_r+\mathbf{S}_{q\alpha}^\dagger\boldsymbol{\lambda}_q\right)-\mathbf{C}_{\beta\beta}\left(\mathbf{S}_{r\beta}^\dagger\boldsymbol{\lambda}_r+\mathbf{S}_{q\beta}^\dagger\boldsymbol{\lambda}_q\right)+\mathbf{C}_{\beta r}\boldsymbol{\lambda}_r+\mathbf{C}_{\beta q}\boldsymbol{\lambda}_q \tag{2.36}$$

$$\left(\mathbf{r}^{pred}-\mathbf{r}^0\right)=-\mathbf{C}_{\alpha r}^\dagger\left(\mathbf{S}_{r\alpha}^\dagger\boldsymbol{\lambda}_r+\mathbf{S}_{q\alpha}^\dagger\boldsymbol{\lambda}_q\right)-\mathbf{C}_{\beta r}^\dagger\left(\mathbf{S}_{r\beta}^\dagger\boldsymbol{\lambda}_r+\mathbf{S}_{q\beta}^\dagger\boldsymbol{\lambda}_q\right)+\mathbf{C}_{rr}\boldsymbol{\lambda}_r+\mathbf{C}_{rq}\boldsymbol{\lambda}_q \tag{2.37}$$

$$\left(\mathbf{q}^{pred}-\mathbf{q}^0\right)=-\mathbf{C}_{\alpha q}^\dagger\left(\mathbf{S}_{r\alpha}^\dagger\boldsymbol{\lambda}_r+\mathbf{S}_{q\alpha}^\dagger\boldsymbol{\lambda}_q\right)-\mathbf{C}_{\beta q}^\dagger\left(\mathbf{S}_{r\beta}^\dagger\boldsymbol{\lambda}_r+\mathbf{S}_{q\beta}^\dagger\boldsymbol{\lambda}_q\right)+\mathbf{C}_{rq}^\dagger\boldsymbol{\lambda}_r+\mathbf{C}_{qq}\boldsymbol{\lambda}_q \tag{2.38}$$

Note that no approximations have been introduced thus far, so that Eqs. (2.35) through (2.38) are exact for the a priori information considered to be known (i.e., known means and covariance matrices for the parameters and measured responses). On the other hand, these equations cannot be used to compute the optimally predicted mean values for the parameters and responses, since the Lagrange multipliers $\boldsymbol{\lambda}_r$ and $\boldsymbol{\lambda}_q$ are still undetermined. Two additional relations are needed to determine these Lagrange multipliers. These relations are obtained by considering the model responses as explicit functions of the model parameters.

To first-order in the parameter variations the model responses $\mathbf{r}$ (for Model A) and $\mathbf{q}$ (for Model B) would be linear functions of the parameter variations of the form

$$\mathbf{r}=\mathbf{r}^c\left(\boldsymbol{\alpha}^0,\boldsymbol{\beta}^0\right)+\mathbf{S}_{r\alpha}\left(\boldsymbol{\alpha}-\boldsymbol{\alpha}^0\right)+\mathbf{S}_{r\beta}\left(\boldsymbol{\beta}-\boldsymbol{\beta}^0\right)+\textit{higher order terms}, \tag{2.39}$$

$$\mathbf{q}=\mathbf{q}^c\left(\boldsymbol{\alpha}^0,\boldsymbol{\beta}^0\right)+\mathbf{S}_{q\alpha}\left(\boldsymbol{\alpha}-\boldsymbol{\alpha}^0\right)+\mathbf{S}_{q\beta}\left(\boldsymbol{\beta}-\boldsymbol{\beta}^0\right)+\textit{higher order terms}. \tag{2.40}$$

In particular, for the predicted parameter values $\boldsymbol{\alpha}^{pred}$ and $\boldsymbol{\beta}^{pred}$, the responses predicted by the linearized models would be given the following expressions:

$$\mathbf{r}^{pred} = \mathbf{r}^{c}\left(\boldsymbol{\alpha}^{0},\boldsymbol{\beta}^{0}\right) + \mathbf{S}_{r\alpha}\left(\boldsymbol{\alpha}^{pred} - \boldsymbol{\alpha}^{0}\right) + \mathbf{S}_{r\beta}\left(\boldsymbol{\beta}^{pred} - \boldsymbol{\beta}^{0}\right) + \textit{higher order terms}, \tag{2.41}$$

$$\mathbf{q}^{pred} = \mathbf{q}^{c}\left(\boldsymbol{\alpha}^{0},\boldsymbol{\beta}^{0}\right) + \mathbf{S}_{q\alpha}\left(\boldsymbol{\alpha}^{pred} - \boldsymbol{\alpha}^{0}\right) + \mathbf{S}_{q\beta}\left(\boldsymbol{\beta}^{pred} - \boldsymbol{\beta}^{0}\right) + \textit{higher order terms}. \tag{2.42}$$

The following intermediate steps are now performed in order to eliminate the Lagrange multipliers: (i) replace $\mathbf{r}^{pred}$ and $\mathbf{q}^{pred}$ from Eqs.(2.41) and (2.42) into Eqs.(2.35) through (2.38) to obtain a system of four equations for the four unknowns $\left(\boldsymbol{\alpha}^{pred},\boldsymbol{\beta}^{pred},\boldsymbol{\lambda}_{r},\boldsymbol{\lambda}_{q}\right)$; (ii) from this system, eliminate the quantities $\left(\boldsymbol{\alpha}^{pred} - \boldsymbol{\alpha}^{0}\right)$ and $\left(\boldsymbol{\beta}^{pred} - \boldsymbol{\beta}^{0}\right)$; and (iii) re-arrange the resulting equations to obtain the following coupled equations for the Lagrange multipliers:

$$\begin{bmatrix} \mathbf{D}_{rr} & \mathbf{D}_{rq} \\ \mathbf{D}_{qr} & \mathbf{D}_{qq} \end{bmatrix} \begin{bmatrix} \boldsymbol{\lambda}_{r} \\ \boldsymbol{\lambda}_{q} \end{bmatrix} = \begin{bmatrix} \mathbf{r}^{d}\left(\boldsymbol{\alpha}^{0},\boldsymbol{\beta}^{0}\right) \\ \mathbf{q}^{d}\left(\boldsymbol{\alpha}^{0},\boldsymbol{\beta}^{0}\right) \end{bmatrix}, \tag{2.43}$$

where the block-matrix of known quantities on the left-side, and the block-vector of known quantities on the right-side of the above equations are defined as follows:

$$\begin{aligned} \mathbf{D}_{rr} \triangleq\ & \mathbf{S}_{r\alpha}\left(\mathbf{C}_{\alpha\alpha}\mathbf{S}_{r\alpha}^{\dagger} + \mathbf{C}_{\alpha\beta}\mathbf{S}_{r\beta}^{\dagger} - \mathbf{C}_{\alpha r}\right) + \mathbf{S}_{r\beta}\left(\mathbf{C}_{\alpha\beta}^{\dagger}\mathbf{S}_{r\alpha}^{\dagger} + \mathbf{C}_{\beta\beta}\mathbf{S}_{r\beta}^{\dagger} - \mathbf{C}_{\beta r}\right) \\ & -\mathbf{C}_{\alpha r}^{\dagger}\mathbf{S}_{r\alpha}^{\dagger} - \mathbf{C}_{\beta r}^{\dagger}\mathbf{S}_{r\beta}^{\dagger} + \mathbf{C}_{rr}, \end{aligned} \tag{2.44}$$

$$\begin{aligned} \mathbf{D}_{rq} \triangleq\ & \mathbf{S}_{r\alpha}\left(\mathbf{C}_{\alpha\alpha}\mathbf{S}_{q\alpha}^{\dagger} + \mathbf{C}_{\alpha\beta}\mathbf{S}_{q\beta}^{\dagger} - \mathbf{C}_{\alpha q}\right) + \mathbf{S}_{r\beta}\left(\mathbf{C}_{\alpha\beta}^{\dagger}\mathbf{S}_{q\alpha}^{\dagger} + \mathbf{C}_{\beta\beta}\mathbf{S}_{q\beta}^{\dagger} - \mathbf{C}_{\beta q}\right) \\ & -\mathbf{C}_{\alpha r}^{\dagger}\mathbf{S}_{q\alpha}^{\dagger} - \mathbf{C}_{\beta r}^{\dagger}\mathbf{S}_{q\beta}^{\dagger} + \mathbf{C}_{rq}, \end{aligned} \tag{2.45}$$

$$\begin{aligned} \mathbf{D}_{qr} \triangleq\ & \mathbf{S}_{q\alpha}\left(\mathbf{C}_{\alpha\alpha}\mathbf{S}_{r\alpha}^{\dagger} + \mathbf{C}_{\alpha\beta}\mathbf{S}_{r\beta}^{\dagger} - \mathbf{C}_{\alpha r}\right) + \mathbf{S}_{q\beta}\left(\mathbf{C}_{\alpha\beta}^{\dagger}\mathbf{S}_{r\alpha}^{\dagger} + \mathbf{C}_{\beta\beta}\mathbf{S}_{r\beta}^{\dagger} - \mathbf{C}_{\beta r}\right) \\ & -\mathbf{C}_{\alpha q}^{\dagger}\mathbf{S}_{r\alpha}^{\dagger} - \mathbf{C}_{\beta q}^{\dagger}\mathbf{S}_{r\beta}^{\dagger} + \mathbf{C}_{rq}^{\dagger} = \mathbf{D}_{rq}^{\dagger}, \end{aligned} \tag{2.46}$$

$$\begin{aligned} \mathbf{D}_{qq} \triangleq\ & \mathbf{S}_{q\alpha}\left(\mathbf{C}_{\alpha\alpha}\mathbf{S}_{q\alpha}^{\dagger} + \mathbf{C}_{\alpha\beta}\mathbf{S}_{q\beta}^{\dagger} - \mathbf{C}_{\alpha q}\right) + \mathbf{S}_{q\beta}\left(\mathbf{C}_{\alpha\beta}^{\dagger}\mathbf{S}_{q\alpha}^{\dagger} + \mathbf{C}_{\beta\beta}\mathbf{S}_{q\beta}^{\dagger} - \mathbf{C}_{\beta q}\right) \\ & -\mathbf{C}_{\alpha q}^{\dagger}\mathbf{S}_{q\alpha}^{\dagger} - \mathbf{C}_{\beta q}^{\dagger}\mathbf{S}_{\beta q}^{\dagger} + \mathbf{C}_{qq}, \end{aligned} \tag{2.47}$$

$$\mathbf{r}^{d}\left(\boldsymbol{\alpha}^{0},\boldsymbol{\beta}^{0}\right) \triangleq \mathbf{r}^{c}\left(\boldsymbol{\alpha}^{0},\boldsymbol{\beta}^{0}\right) - \mathbf{r}^{m}; \qquad \mathbf{q}^{d}\left(\boldsymbol{\alpha}^{0},\boldsymbol{\beta}^{0}\right) \triangleq \mathbf{q}^{c}\left(\boldsymbol{\alpha}^{0},\boldsymbol{\beta}^{0}\right) - \mathbf{q}^{m}. \tag{2.48}$$

Note that the vectors $\mathbf{r}^{d}\left(\boldsymbol{\alpha}^{0},\boldsymbol{\beta}^{0}\right)$ and $\mathbf{q}^{d}\left(\boldsymbol{\alpha}^{0},\boldsymbol{\beta}^{0}\right)$ measure the differences ("deviations") between the computed and measured responses. Note also that the matrices defined in Eqs. (2.44) through

(2.47) have the following dimensions: $\dim \mathbf{D}_{rr} = (N_r \times N_r)$; $\dim \mathbf{D}_{rq} = (N_r \times N_q)$; $\dim \mathbf{D}_{qr} = \mathbf{D}_{rq}^{\dagger} = (N_q \times N_r)$; and $\dim \mathbf{D}_{qq} = (N_q \times N_q)$, and have the following physical meanings:

(i) The matrix $\mathbf{D}_{rr}$ is actually the covariance matrix of the vector of response "deviations" for Model A, i.e.,

$$\mathbf{D}_{rr} = \left\langle \mathbf{r}^d\left(\boldsymbol{\alpha}^0,\boldsymbol{\beta}^0\right)\left[\mathbf{r}^d\left(\boldsymbol{\alpha}^0,\boldsymbol{\beta}^0\right)\right]^{\dagger}\right\rangle; \tag{2.49}$$

(ii) The matrix $\mathbf{D}_{qq}$ is actually the covariance matrix of the vector of response "deviations" for Model B, i.e.,

$$\mathbf{D}_{qq} = \left\langle \mathbf{q}^d\left(\boldsymbol{\alpha}^0,\boldsymbol{\beta}^0\right)\left[\mathbf{q}^d\left(\boldsymbol{\alpha}^0,\boldsymbol{\beta}^0\right)\right]^{\dagger}\right\rangle; \tag{2.50}$$

(iii) The matrix $\mathbf{D}_{rq} = \mathbf{D}_{rq}^{\dagger}$ is actually the correlation matrix between the vector of response "deviations" for Model A and Model B, i.e.,

$$\mathbf{D}_{rq} = \left\langle \mathbf{q}^d\left(\boldsymbol{\alpha}^0,\boldsymbol{\beta}^0\right)\left[\mathbf{r}^d\left(\boldsymbol{\alpha}^0,\boldsymbol{\beta}^0\right)\right]^{\dagger}\right\rangle; \quad \mathbf{D}_{qr} = \left\langle \mathbf{r}^d\left(\boldsymbol{\alpha}^0,\boldsymbol{\beta}^0\right)\left[\mathbf{q}^d\left(\boldsymbol{\alpha}^0,\boldsymbol{\beta}^0\right)\right]^{\dagger}\right\rangle. \tag{2.51}$$

The Lagrange multipliers $\boldsymbol{\lambda}_r$ and $\boldsymbol{\lambda}_q$ are obtained by solving Eq.(2.41) , which requires the inverse of the matrix

$$\mathbf{D} \triangleq \begin{bmatrix} \mathbf{D}_{rr} & \mathbf{D}_{rq} \\ \mathbf{D}_{rq}^{\dagger} & \mathbf{D}_{qq} \end{bmatrix} \tag{2.52}$$

The matrix defined in Eq.(2.52) can be inverted by partitioning it to obtain

$$\mathbf{D}^{-1} \triangleq \begin{bmatrix} \mathbf{D}_{11} & \mathbf{D}_{12} \\ \mathbf{D}_{12}^{\dagger} & \mathbf{D}_{22} \end{bmatrix}, \tag{2.53}$$

Where

$$\mathbf{D}_{11} \triangleq \mathbf{D}_{rr}^{-1} + \mathbf{D}_{rr}^{-1}\mathbf{D}_{rq}\mathbf{D}_{22}\mathbf{D}_{rq}^{\dagger}\mathbf{D}_{rr}^{-1}, \tag{2.54}$$

$$\mathbf{D}_{12} \triangleq -\mathbf{D}_{rr}^{-1}\mathbf{D}_{rq}\mathbf{D}_{22}, \tag{2.55}$$

$$\mathbf{D}_{12}^{\dagger} \triangleq -\mathbf{D}_{22}\mathbf{D}_{rq}^{\dagger}\mathbf{D}_{rr}^{-1}, \tag{2.56}$$

$$\mathbf{D}_{22} \triangleq \left(\mathbf{D}_{qq} - \mathbf{D}_{rq}^{\dagger}\mathbf{D}_{rr}^{-1}\mathbf{D}_{rq}\right)^{-1}. \tag{2.57}$$

After obtaining the expressions of $\boldsymbol{\lambda}_r$ and $\boldsymbol{\lambda}_q$ by solving Eq.(2.43), they are replaced in Eqs.(2.35) through (2.38) to obtain the following expressions for the optimally predicted values of model parameters and responses:

$$\boldsymbol{\alpha}^{pred} = \boldsymbol{\alpha}^0 - \left[\mathbf{X}_\alpha\mathbf{D}_{11} + \mathbf{Y}_\alpha\mathbf{D}_{12}^{\dagger}\right]\mathbf{r}^d\left(\boldsymbol{\alpha}^0,\boldsymbol{\beta}^0\right) - \left[\mathbf{X}_\alpha\mathbf{D}_{12} + \mathbf{Y}_\alpha\mathbf{D}_{22}\right]\mathbf{q}^d\left(\boldsymbol{\alpha}^0,\boldsymbol{\beta}^0\right), \tag{2.58}$$

$$\boldsymbol{\beta}^{pred} = \boldsymbol{\beta}^0 - \left[\mathbf{X}_\beta\mathbf{D}_{11} + \mathbf{Y}_\beta\mathbf{D}_{12}^{\dagger}\right]\mathbf{r}^d\left(\boldsymbol{\alpha}^0,\boldsymbol{\beta}^0\right) - \left[\mathbf{X}_\beta\mathbf{D}_{12} + \mathbf{Y}_\beta\mathbf{D}_{22}\right]\mathbf{q}^d\left(\boldsymbol{\alpha}^0,\boldsymbol{\beta}^0\right), \tag{2.59}$$

$$\mathbf{r}^{pred} = \mathbf{r}^m - \left[\mathbf{X}_r\mathbf{D}_{11} + \mathbf{Y}_r\mathbf{D}_{12}^{\dagger}\right]\mathbf{r}^d\left(\boldsymbol{\alpha}^0,\boldsymbol{\beta}^0\right) - \left[\mathbf{X}_r\mathbf{D}_{12} + \mathbf{Y}_r\mathbf{D}_{22}\right]\mathbf{q}^d\left(\boldsymbol{\alpha}^0,\boldsymbol{\beta}^0\right), \tag{2.60}$$

$$\mathbf{q}^{pred} = \mathbf{q}^m - \left[\mathbf{X}_q\mathbf{D}_{11} + \mathbf{Y}_q\mathbf{D}_{12}^{\dagger}\right]\mathbf{r}^d\left(\boldsymbol{\alpha}^0,\boldsymbol{\beta}^0\right) - \left[\mathbf{X}_q\mathbf{D}_{12} + \mathbf{Y}_q\mathbf{D}_{22}\right]\mathbf{q}^d\left(\boldsymbol{\alpha}^0,\boldsymbol{\beta}^0\right), \tag{2.61}$$

where

$$\mathbf{X}_\alpha \triangleq \mathbf{C}_{\alpha\alpha}\mathbf{S}_{r\alpha}^{\dagger} + \mathbf{C}_{\alpha\beta}\mathbf{S}_{r\beta}^{\dagger} - \mathbf{C}_{\alpha r}, \tag{2.62}$$

$$\mathbf{Y}_\alpha \triangleq \mathbf{C}_{\alpha\alpha}\mathbf{S}_{q\alpha}^{\dagger} + \mathbf{C}_{\alpha\beta}\mathbf{S}_{q\beta}^{\dagger} - \mathbf{C}_{\alpha q}, \tag{2.63}$$

$$\mathbf{X}_\beta \triangleq \mathbf{C}_{\alpha\beta}^{\dagger}\mathbf{S}_{r\alpha}^{\dagger} + \mathbf{C}_{\beta\beta}\mathbf{S}_{r\beta}^{\dagger} - \mathbf{C}_{\beta r}, \tag{2.64}$$

$$\mathbf{Y}_\beta \triangleq \mathbf{C}_{\beta\alpha}\mathbf{S}_{q\alpha}^{\dagger} + \mathbf{C}_{\beta\beta}\mathbf{S}_{q\beta}^{\dagger} - \mathbf{C}_{\beta q}, \tag{2.65}$$

$$\mathbf{X}_r \triangleq \mathbf{C}_{\alpha r}^{\dagger}\mathbf{S}_{r\alpha}^{\dagger} + \mathbf{C}_{\beta r}^{\dagger}\mathbf{S}_{r\beta}^{\dagger} - \mathbf{C}_{rr}, \tag{2.66}$$

$$\mathbf{Y}_r \triangleq \mathbf{C}_{\alpha r}^{\dagger}\mathbf{S}_{q\alpha}^{\dagger} + \mathbf{C}_{\beta r}^{\dagger}\mathbf{S}_{q\beta}^{\dagger} - \mathbf{C}_{rq}, \tag{2.67}$$

$$\mathbf{X}_q \triangleq \mathbf{C}_{\alpha q}^{\dagger}\mathbf{S}_{r\alpha}^{\dagger} + \mathbf{C}_{\beta q}^{\dagger}\mathbf{S}_{r\beta}^{\dagger} - \mathbf{C}_{rq}^{\dagger}, \tag{2.68}$$

$$\mathbf{Y}_q \triangleq \mathbf{C}_{\alpha q}^{\dagger}\mathbf{S}_{q\alpha}^{\dagger} + \mathbf{C}_{\beta q}^{\dagger}\mathbf{S}_{q\beta}^{\dagger} - \mathbf{C}_{qq}. \tag{2.69}$$

The predicted optimal covariance matrix $\mathbf{C}_{\alpha\alpha}^{pred}$ for the parameters $\boldsymbol{\alpha}$ of Model A is obtained as:

$$\mathbf{C}_{\alpha\alpha}^{pred} \triangleq \left\langle \left(\boldsymbol{\alpha} - \boldsymbol{\alpha}^{pred}\right)\left(\boldsymbol{\alpha} - \boldsymbol{\alpha}^{pred}\right)^{\dagger} \right\rangle$$
$$= \mathbf{C}_{\alpha\alpha} - \left[\mathbf{X}_{\alpha}\left(\mathbf{D}_{11}\mathbf{X}_{\alpha}^{\dagger} + \mathbf{D}_{12}\mathbf{Y}_{\alpha}^{\dagger}\right) + \mathbf{Y}_{\alpha}\left(\mathbf{D}_{21}\mathbf{X}_{\alpha}^{\dagger} + \mathbf{D}_{22}\mathbf{Y}_{\alpha}^{\dagger}\right)\right]; \tag{2.70}$$

The predicted covariance matrix $\mathbf{C}_{rr}^{pred}$ for the responses $\mathbf{r}$ of Model A is obtained as:

$$\mathbf{C}_{rr}^{pred} \triangleq \left\langle \left(\mathbf{r} - \mathbf{r}^{pred}\right)\left(\mathbf{r} - \mathbf{r}^{pred}\right)^{\dagger} \right\rangle$$
$$= \mathbf{C}_{rr} - \left[\mathbf{X}_{r}\left(\mathbf{D}_{11}\mathbf{X}_{r}^{\dagger} + \mathbf{D}_{12}\mathbf{Y}_{r}^{\dagger}\right) + \mathbf{Y}_{r}\left(\mathbf{D}_{21}\mathbf{X}_{r}^{\dagger} + \mathbf{D}_{22}\mathbf{Y}_{r}^{\dagger}\right)\right]; \tag{2.71}$$

The predicted correlation matrix $\mathbf{C}_{\alpha r}^{pred}$ for the parameters $\boldsymbol{\alpha}$ and $\mathbf{r}$ responses of Model A is obtained as:

$$\mathbf{C}_{\alpha r}^{pred} \triangleq \left\langle \left(\boldsymbol{\alpha} - \boldsymbol{\alpha}^{pred}\right)\left(\mathbf{r} - \mathbf{r}^{pred}\right)^{\dagger} \right\rangle$$
$$= \mathbf{C}_{\alpha r} - \left[\mathbf{X}_{\alpha}\left(\mathbf{D}_{11}\mathbf{X}_{r}^{\dagger} + \mathbf{D}_{12}\mathbf{Y}_{r}^{\dagger}\right) + \mathbf{Y}_{\alpha}\left(\mathbf{D}_{21}\mathbf{X}_{r}^{\dagger} + \mathbf{D}_{22}\mathbf{Y}_{r}^{\dagger}\right)\right]; \tag{2.72}$$

The predicted covariance matrix $\mathbf{C}_{\beta\beta}^{pred}$ for the parameters $\boldsymbol{\beta}$ of Model B is obtained as:

$$\mathbf{C}_{\beta\beta}^{pred} \triangleq \left\langle \left(\boldsymbol{\beta} - \boldsymbol{\beta}^{pred}\right)\left(\boldsymbol{\beta} - \boldsymbol{\beta}^{pred}\right)^{\dagger} \right\rangle$$
$$= \mathbf{C}_{\beta\beta} - \left[\mathbf{X}_{\beta}\left(\mathbf{D}_{11}\mathbf{X}_{\beta}^{\dagger} + \mathbf{D}_{12}\mathbf{Y}_{\beta}^{\dagger}\right) + \mathbf{Y}_{\beta}\left(\mathbf{D}_{21}\mathbf{X}_{\beta}^{\dagger} + \mathbf{D}_{22}\mathbf{Y}_{\beta}^{\dagger}\right)\right]; \tag{2.73}$$

The predicted covariance matrix $\mathbf{C}_{qq}^{pred}$ for the responses $\mathbf{q}$ of Model B is obtained as:

$$\mathbf{C}_{qq}^{pred} \triangleq \left\langle \left(\mathbf{q} - \mathbf{q}^{pred}\right)\left(\mathbf{q} - \mathbf{q}^{pred}\right)^{\dagger} \right\rangle$$
$$= \mathbf{C}_{qq} - \left[\mathbf{X}_{q}\left(\mathbf{D}_{11}\mathbf{X}_{q}^{\dagger} + \mathbf{D}_{12}\mathbf{Y}_{q}^{\dagger}\right) + \mathbf{Y}_{q}\left(\mathbf{D}_{21}\mathbf{X}_{q}^{\dagger} + \mathbf{D}_{22}\mathbf{Y}_{q}^{\dagger}\right)\right]; \tag{2.74}$$

The predicted correlation matrix $\mathbf{C}_{\beta q}^{pred}$ for the parameters $\boldsymbol{\beta}$ and the responses $\mathbf{q}$ of Model B is obtained as:

$$\mathbf{C}_{\beta q}^{opt} \triangleq \left\langle \left(\boldsymbol{\beta} - \boldsymbol{\beta}^{pred}\right)\left(\mathbf{q} - \mathbf{q}^{pred}\right)^{\dagger} \right\rangle$$
$$= \mathbf{C}_{\beta q} - \left[\mathbf{X}_{\beta}\left(\mathbf{D}_{11}\mathbf{X}_{q}^{\dagger} + \mathbf{D}_{12}\mathbf{Y}_{q}^{\dagger}\right) + \mathbf{Y}_{\beta}\left(\mathbf{D}_{21}\mathbf{X}_{q}^{\dagger} + \mathbf{D}_{22}\mathbf{Y}_{q}^{\dagger}\right)\right]; \tag{2.75}$$

The predicted correlation matrix $\mathbf{C}_{\alpha\beta}^{pred}$ for the parameters $\boldsymbol{\alpha}$ of Model A and the parameters $\boldsymbol{\beta}$ of Model B is obtained as:

$$\begin{aligned}\mathbf{C}_{\alpha\beta}^{pred} &\triangleq \left\langle \left(\boldsymbol{\alpha}-\boldsymbol{\alpha}^{pred}\right)\left(\boldsymbol{\beta}-\boldsymbol{\beta}^{pred}\right)^{\dagger}\right\rangle \\ &= \mathbf{C}_{\alpha\beta} - \left[\mathbf{X}_{\alpha}\left(\mathbf{D}_{11}\mathbf{X}_{\beta}^{\dagger}+\mathbf{D}_{12}\mathbf{Y}_{\beta}^{\dagger}\right)+\mathbf{Y}_{\alpha}\left(\mathbf{D}_{21}\mathbf{X}_{\beta}^{\dagger}+\mathbf{D}_{22}\mathbf{Y}_{\beta}^{\dagger}\right)\right];\end{aligned} \tag{2.76}$$

The predicted correlation matrix $\mathbf{C}_{\alpha q}^{pred}$ for the parameters $\boldsymbol{\alpha}$ of Model A and the responses $\mathbf{q}$ of Model B is obtained as:

$$\begin{aligned}\mathbf{C}_{\alpha q}^{pred} &\triangleq \left\langle \left(\boldsymbol{\alpha}-\boldsymbol{\alpha}^{pred}\right)\left(\mathbf{q}-\mathbf{q}^{pred}\right)^{\dagger}\right\rangle \\ &= \mathbf{C}_{\alpha q} - \left[\mathbf{X}_{\alpha}\left(\mathbf{D}_{11}\mathbf{X}_{q}^{\dagger}+\mathbf{D}_{12}\mathbf{Y}_{q}^{\dagger}\right)+\mathbf{Y}_{\alpha}\left(\mathbf{D}_{21}\mathbf{X}_{q}^{\dagger}+\mathbf{D}_{22}\mathbf{Y}_{q}^{\dagger}\right)\right];\end{aligned} \tag{2.77}$$

The predicted correlation matrix $\mathbf{C}_{\beta r}^{pred}$ for the parameters $\boldsymbol{\beta}$ of Model B and the responses $\mathbf{r}$ of Model A is obtained as:

$$\begin{aligned}\mathbf{C}_{\beta r}^{pred} &\triangleq \left\langle \left(\boldsymbol{\beta}-\boldsymbol{\beta}^{pred}\right)\left(\mathbf{r}-\mathbf{r}^{pred}\right)^{\dagger}\right\rangle \\ &= \mathbf{C}_{\beta r} - \left[\mathbf{X}_{\beta}\left(\mathbf{D}_{11}\mathbf{X}_{r}^{\dagger}+\mathbf{D}_{12}\mathbf{Y}_{r}^{\dagger}\right)+\mathbf{Y}_{\beta}\left(\mathbf{D}_{21}\mathbf{X}_{r}^{\dagger}+\mathbf{D}_{22}\mathbf{Y}_{r}^{\dagger}\right)\right];\end{aligned} \tag{2.78}$$

The predicted correlation matrix $\mathbf{C}_{rq}^{pred}$ for the responses $\mathbf{r}$ of Model A and the responses $\mathbf{q}$ of Model B is obtained as:

$$\begin{aligned}\mathbf{C}_{rq}^{pred} &\triangleq \left\langle \left(\mathbf{r}-\mathbf{r}^{pred}\right)\left(\mathbf{q}-\mathbf{q}^{pred}\right)^{\dagger}\right\rangle \\ &= \mathbf{C}_{rq} - \left[\mathbf{X}_{r}\left(\mathbf{D}_{11}\mathbf{X}_{q}^{\dagger}+\mathbf{D}_{12}\mathbf{Y}_{q}^{\dagger}\right)+\mathbf{Y}_{r}\left(\mathbf{D}_{21}\mathbf{X}_{q}^{\dagger}+\mathbf{D}_{22}\mathbf{Y}_{q}^{\dagger}\right)\right].\end{aligned} \tag{2.79}$$

The covariance matrices of the computed responses arising from the uncertainties in the model parameters can be computed from Eqs.(2.39) and (2.40), respectively, to obtain:

$$\begin{aligned}\mathbf{C}_{rr}^{comp} &\triangleq \left\langle \left[\mathbf{r}-\mathbf{r}^{c}\left(\boldsymbol{\alpha}^{0},\boldsymbol{\beta}^{0}\right)\right]\left[\mathbf{r}-\mathbf{r}^{c}\left(\boldsymbol{\alpha}^{0},\boldsymbol{\beta}^{0}\right)\right]^{\dagger}\right\rangle \\ &= \mathbf{S}_{r\alpha}\mathbf{C}_{\alpha\alpha}\mathbf{S}_{r\alpha}^{\dagger} + 2\mathbf{S}_{r\alpha}\mathbf{C}_{\alpha\beta}\mathbf{S}_{r\beta}^{\dagger} + \mathbf{S}_{r\beta}\mathbf{C}_{\beta\beta}\mathbf{S}_{r\beta}^{\dagger},\end{aligned} \tag{2.80}$$

$$\mathbf{C}_{qq}^{comp} \triangleq \left\langle \left[\mathbf{q} - \mathbf{q}^c\left(\boldsymbol{\alpha}^0, \boldsymbol{\beta}^0\right) \right] \left[\mathbf{q} - \mathbf{q}^c\left(\boldsymbol{\alpha}^0, \boldsymbol{\beta}^0\right) \right]^\dagger \right\rangle$$
$$= \mathbf{S}_{q\alpha}\mathbf{C}_{\alpha\alpha}\mathbf{S}_{q\alpha}^\dagger + 2\mathbf{S}_{q\alpha}\mathbf{C}_{\alpha\beta}\mathbf{S}_{q\beta}^\dagger + \mathbf{S}_{q\beta}\mathbf{C}_{\beta\beta}\mathbf{S}_{q\beta}^\dagger, \tag{2.81}$$

$$\mathbf{C}_{rq}^{comp} \triangleq \left\langle \left[\mathbf{r} - \mathbf{r}^c\left(\boldsymbol{\alpha}^0, \boldsymbol{\beta}^0\right) \right] \left[\mathbf{q} - \mathbf{q}^c\left(\boldsymbol{\alpha}^0, \boldsymbol{\beta}^0\right) \right]^\dagger \right\rangle$$
$$= \mathbf{S}_{r\alpha}\mathbf{C}_{\alpha\alpha}\mathbf{S}_{q\alpha}^\dagger + \mathbf{S}_{r\alpha}\mathbf{C}_{\alpha\beta}\mathbf{S}_{q\beta}^\dagger + \mathbf{S}_{r\beta}\mathbf{C}_{\alpha\beta}^\dagger\mathbf{S}_{q\alpha}^\dagger + \mathbf{S}_{r\beta}\mathbf{C}_{\beta\beta}\mathbf{S}_{q\beta}^\dagger. \tag{2.82}$$

2.2.4 Construction of the A Posteriori Predicted Consistency Metrics for Model Validation

At the saddle-point $\left(\boldsymbol{\alpha}^{pred}, \boldsymbol{\beta}^{pred}, \mathbf{r}^{pred}, \mathbf{q}^{pred}\right)$, the functional $P\left(\boldsymbol{\alpha}, \boldsymbol{\beta}, \mathbf{r}, \mathbf{q}\right)$ defined in Eq.(2.29), and the first-order computational model equations become

$$P^{\min} = \begin{pmatrix} \boldsymbol{\alpha}^{pred} - \boldsymbol{\alpha}^0 \\ \boldsymbol{\beta}^{pred} - \boldsymbol{\beta}^0 \\ \mathbf{r}^{pred} - \mathbf{r}^m \\ \mathbf{q}^{pred} - \mathbf{q}^m \end{pmatrix}^\dagger \mathbf{C}^{-1} \begin{pmatrix} \boldsymbol{\alpha}^{pred} - \boldsymbol{\alpha}^0 \\ \boldsymbol{\beta}^{pred} - \boldsymbol{\beta}^0 \\ \mathbf{r}^{pred} - \mathbf{r}^m \\ \mathbf{q}^{pred} - \mathbf{q}^m \end{pmatrix}, \tag{2.83}$$

$$\mathbf{r}^{pred} = \mathbf{r}^c\left(\boldsymbol{\alpha}^0, \boldsymbol{\beta}^0\right) + \mathbf{S}_{r\alpha}\left(\boldsymbol{\alpha}^{pred} - \boldsymbol{\alpha}^0\right) + \mathbf{S}_{r\beta}\left(\boldsymbol{\beta}^{pred} - \boldsymbol{\beta}^0\right) = \mathbf{r}^c\left(\boldsymbol{\alpha}^{pred}, \boldsymbol{\beta}^{pred}\right), \tag{2.84}$$

$$\mathbf{q}^{pred} = \mathbf{q}^c\left(\boldsymbol{\alpha}^0, \boldsymbol{\beta}^0\right) + \mathbf{S}_{q\alpha}\left(\boldsymbol{\alpha}^{pred} - \boldsymbol{\alpha}^0\right) + \mathbf{S}_{q\beta}\left(\boldsymbol{\beta}^{pred} - \boldsymbol{\beta}^0\right) = \mathbf{q}^c\left(\boldsymbol{\alpha}^{opt}, \boldsymbol{\beta}^{opt}\right). \tag{2.85}$$

The values $\left(\boldsymbol{\alpha}^{pred}, \boldsymbol{\beta}^{pred}, \mathbf{r}^{pred}, \mathbf{q}^{pred}\right)$ can be eliminated from the expression of by using Eqs. (2.84) and (2.85) together with Eq. (2.32) to obtain

$$P^{\min} \triangleq V = \left[\left(\mathbf{r}^d\right)^\dagger, \left(\mathbf{q}^d\right)^\dagger \right] \begin{bmatrix} \mathbf{D}_{11} & \mathbf{D}_{12} \\ \mathbf{D}_{12}^\dagger & \mathbf{D}_{22} \end{bmatrix} \begin{bmatrix} \mathbf{r}^d\left(\boldsymbol{\alpha}^0, \boldsymbol{\beta}^0\right) \\ \mathbf{q}^d\left(\boldsymbol{\alpha}^0, \boldsymbol{\beta}^0\right) \end{bmatrix}. \tag{2.86}$$

Note that the quadratic form on the rightmost-side of Eq.(2.86) is distributed according to a χ^2 distribution with $\left(N_r + N_q\right)$ degrees of freedom. The "validation metric" V can be evaluated directly from the originally given data (i.e., from given parameters and responses, together with their original uncertainties), once the response sensitivities have been computed by either forward or adjoint methods (see, e.g., Cacuci 1981a, 1981b, 2003). Recall that the χ^2 (chi-square) distribution with n degrees of freedom of the continuous variable x, $0 \le x < \infty$, is defined as

$$P\left(x < \chi^2 < x + dx\right)dx = \frac{1}{2^{n/2}\Gamma\left(n/2\right)} x^{n/2-1} e^{-x/2} dx, \; x > 0, \; \left(n = 1, 2, \ldots\right). \tag{2.87}$$

The χ^2-distribution is a measure of the deviation of a "true distribution" (in this case – the distribution of experimental responses) from the hypothetic one (in this case – a Gaussian). Recall that the mean and variance of x are $\langle x \rangle = n$ and $\operatorname{var}\left(x\right) = 2n$. The value of χ^2 is computed using Eq.(2.86) to obtain

$$V \triangleq \chi^2 = \left(\mathbf{r}^c - \mathbf{r}^m\right)^\dagger \mathbf{D}_{11}\left(\mathbf{r}^c - \mathbf{r}^m\right) + 2\left(\mathbf{r}^c - \mathbf{r}^m\right)^\dagger \mathbf{D}_{12}\left(\mathbf{q}^c - \mathbf{q}^m\right) + \left(\mathbf{q}^c - \mathbf{q}^m\right)^\dagger \mathbf{D}_{22}\left(\mathbf{q}^c - \mathbf{q}^m\right). \tag{2.88}$$

The value of $V = \chi^2$ computed using Eq. (2.88) provides a very valuable quantitative indicator for investigating the agreement between the computed and experimental responses, measuring essentially the consistency of the experimental responses with the model parameters. The value of V can be used as a validation metric for measuring the consistency between the computed and experimentally measured responses.

2.3 Discussion and Particular Cases

The derivations in the previous section were carried out in the response-space because in large-scale practical problems, the number of measured responses is smaller than the number of model parameters. The only matrix inversion required in the response space is the computation of $\mathbf{D}^{-1}$ in Eq.(2.53) which is of size $\left(N_r + N_q\right)^2$. If this matrix is too large to be inverted directly, as has been assumed in this work, its inversion can be performed by partitioning it as shown in Eqs (2.54) through (2.57) . The inversion of $\mathbf{D}$ by partitioning requires only the inversion of the matrix $\mathbf{D}_{rr}$ of size N_r, and the inversion of the matrix $\left(\mathbf{D}_{qq} - \mathbf{D}_{rq}^\dagger \mathbf{D}_{rr}^{-1} \mathbf{D}_{rq}\right)$, which is of size N_q.

The PM-CMPS methodology can also be used if one starts with the data assimilation and model calibration for one of the Models (either Model A or Model B), and subsequently couples the second model to the first one. Without the PM-CMPS methodology, when the second Model (e.g., Model B) is coupled to the first one (e.g., Model A), both models would have to be calibrated anew, simultaneously, and the work performed initially for calibrating Model A alone would become useless. Using the PM-CMPS methodology, however, the work initially performed for calibrating Model A would not become useless, but would simply be augmented by the specific additional terms arising from Model B, thus performing predictive modeling of coupled multi-physics systems in a sequential and more efficient way.

It is also important to note that the explicit separation, in Eqs.(2.85) through (2.88), of contributions from Model A and Model B to the overall validation metric V enables the explicit evaluation of adding or subtracting measured responses. Large contributions to V indicate that the respective responses may be inconsistent or discrepant, and such discrepancies warrant further investigations. It often happens in practice that, after one has already performed a model calibration, e.g., using Model A (involving N_α model parameters α_n and N_r experimentally measured responses r_i), additional measurements may become available and/or additional parameters (which were not considered in the initial data assimilation/model calibration/predictive modeling procedure) may need to be taken into account (e.g., model parameters for which quantified uncertainties became available only after the initial data assimilation/model calibration/predictive modeling procedure was already performed), all for the same Model A. The predictive modeling methodology presented in Chapter 2 can also be used as a most efficient procedure for systematically adding or subtracting responses and/or parameters for performing a subsequent data assimilation/model calibration/predictive modeling procedure on the same model. In this interpretation/usage of the predictive modeling methodology presented in Section 2.2, Model B is considered to be identical to Model A (i.e., Model B and Model A represent the same physical phenomena, described by identical mathematical equations). In this context, “efficient” means “without wasting the information already obtained in previous predictive modeling computations involving a different (higher or lower) number of responses and/or model parameters.” As will be shown in the next Sub-section, the mathematical methodology for performing data assimilation/model calibration/predictive modeling by adding and/or subtracting measurements (responses) and/or model parameters to the same model-without needing to discard previous predictive modeling computations-actually amounts to considering particular cases of the general PM-CMPS methodology presented in Section 2.2.

2.3.1 Predictive modeling for a Single Multi-Physics Model

In the case of applying the PM-CMPS methodology for the predictive modeling of a single multi-physics model (e.g., Model A, involving N_α model parameters α_n and N_r experimentally measured responses r_i), Eq.(2.44) through (2.47) take on the following simplified forms:

$$\mathbf{D}_{rq} = \mathbf{0},\ \mathbf{D}_{qr} = \mathbf{0},\ \mathbf{D}_{qq} = \mathbf{0},\ \mathbf{D}_{rr} = \mathbf{S}_{r\alpha}\mathbf{C}_{\alpha\alpha}\mathbf{S}_{r\alpha}^{\dagger} - \mathbf{S}_{r\alpha}\mathbf{C}_{\alpha r} - \mathbf{C}_{\alpha r}^{\dagger}\mathbf{S}_{r\alpha}^{\dagger} + \mathbf{C}_{rr}. \tag{2.89}$$

$$\mathbf{X}_\alpha \triangleq \mathbf{C}_{\alpha\alpha}\mathbf{S}_{r\alpha}^{\dagger} - \mathbf{C}_{\alpha r},\ \ \mathbf{Y}_\alpha \triangleq 0,\ \ \mathbf{X}_r \triangleq \mathbf{C}_{\alpha r}^{\dagger}\mathbf{S}_{r\alpha}^{\dagger} - \mathbf{C}_{rr},\ \ \mathbf{Y}_r = \mathbf{0}. \tag{2.90}$$

Furthermore, the predictive modeling equations (2.58) through (2.79) reduce to the final results presented originally by Cacuci and Ionescu-Bujor (2010a), namely:

$$\boldsymbol{\alpha}^{pred} = \boldsymbol{\alpha}^0 - \left(\mathbf{C}_{\alpha\alpha}\mathbf{S}_{r\alpha}^{\dagger} - \mathbf{C}_{\alpha r}\right)\left[\mathbf{D}_{rr}\right]^{-1}\mathbf{r}^d\left(\boldsymbol{\alpha}^0\right), \tag{2.91}$$

$$\mathbf{r}^{pred} = \mathbf{r}^m - \left(\mathbf{C}_{\alpha r}^{\dagger}\mathbf{S}_{r\alpha}^{\dagger} - \mathbf{C}_{rr}\right)\left[\mathbf{D}_{rr}\right]^{-1}\mathbf{r}^d\left(\boldsymbol{\alpha}^0\right), \tag{2.92}$$

$$\mathbf{C}_{\alpha\alpha}^{pred} = \mathbf{C}_{\alpha\alpha} - \left(\mathbf{C}_{\alpha\alpha}\mathbf{S}_{r\alpha}^{\dagger} - \mathbf{C}_{\alpha r}\right)\left[\mathbf{D}_{rr}\right]^{-1}\left(\mathbf{C}_{\alpha\alpha}\mathbf{S}_{r\alpha}^{\dagger} - \mathbf{C}_{\alpha r}\right)^{\dagger}, \tag{2.93}$$

$$\mathbf{C}_{rr}^{pred} = \mathbf{C}_{rr} - \left(\mathbf{C}_{\alpha r}^{\dagger}\mathbf{S}_{r\alpha}^{\dagger} - \mathbf{C}_{rr}\right)\left[\mathbf{D}_{rr}\right]^{-1}\left(\mathbf{C}_{\alpha r}^{\dagger}\mathbf{S}_{r\alpha}^{\dagger} - \mathbf{C}_{rr}\right)^{\dagger}, \tag{2.94}$$

$$\mathbf{C}_{\alpha r}^{pred} = \mathbf{C}_{\alpha r} - \left(\mathbf{C}_{\alpha\alpha}\mathbf{S}_{r\alpha}^{\dagger} - \mathbf{C}_{\alpha r}\right)\left[\mathbf{D}_{rr}\right]^{-1}\left(\mathbf{C}_{\alpha r}^{\dagger}\mathbf{S}_{r\alpha}^{\dagger} - \mathbf{C}_{rr}\right)^{\dagger}. \tag{2.95}$$

Note that if the model is perfect (i.e., $\mathbf{C}_{\alpha\alpha} = \mathbf{0}$ and $\mathbf{C}_{\alpha r} = \mathbf{0}$), then Eqs.(2.91) through (2.95) would yield $\boldsymbol{\alpha}^{pred} = \boldsymbol{\alpha}^0$ and $\mathbf{r}^{pred} = \mathbf{r}^c\left(\boldsymbol{\alpha}^0, \boldsymbol{\beta}^0\right)$, predicted "perfectly," without any accompanying uncertainties (i.e., $\mathbf{C}_{rr}^{pred} = \mathbf{0}$, $\mathbf{C}_{\alpha\alpha}^{pred} = \mathbf{0}$, $\mathbf{C}_{\alpha r}^{pred} = \mathbf{0}$). In other words, for a perfect model, the PM-CMPS methodology predicts values for the responses and the parameters that coincide with the model's values (assumed to be perfect), and the experimental measurements would have no effect on the predictions (as would be expected, since imperfect measurements could not possibly improve the “perfect” model’s predictions).

On the other hand, if the measurements were perfect, (i.e., $\mathbf{C}_{rr}=\mathbf{0}$ and $\mathbf{C}_{\alpha r}=\mathbf{0}$), but the model were imperfect, then Eqs. (2.91) through (2.95) would yield $\boldsymbol{\alpha}^{pred}=\boldsymbol{\alpha}^{0}-\mathbf{C}_{\alpha\alpha}\mathbf{S}_{r\alpha}^{\dagger}\left[\mathbf{S}_{r\alpha}\mathbf{C}_{\alpha\alpha}\mathbf{S}_{r\alpha}^{\dagger}\right]^{-1}\mathbf{r}^{d}\left(\boldsymbol{\alpha}^{0}\right)$, $\mathbf{C}_{\alpha\alpha}^{pred}=\mathbf{C}_{\alpha\alpha}-\mathbf{C}_{\alpha\alpha}\mathbf{S}_{r\alpha}^{\dagger}\left[\mathbf{S}_{r\alpha}\mathbf{C}_{\alpha\alpha}\mathbf{S}_{r\alpha}^{\dagger}\right]^{-1}\mathbf{S}_{r\alpha}\mathbf{C}_{\alpha\alpha}$, $\mathbf{r}^{pred}=\mathbf{r}^{m}$, $\mathbf{C}_{rr}^{pred}=\mathbf{0}$, $\mathbf{C}_{\alpha r}^{pred}=\mathbf{0}$. In other words, in the case of perfect measurements, the PM-CMPS predicted values for the responses would coincide with the measured values (assumed to be perfect), but the model’s uncertain parameters would be calibrated by taking the measurements into account to yield improved nominal values and reduced parameters uncertainties.

2.3.2 Predictive modeling for Model A with $\boldsymbol{\beta}$ additional parameters, but no additional responses

In this case, Eq. (2.44) through (2.47) become

$$\mathbf{D}_{rq}=\mathbf{0},\ \mathbf{D}_{qr}=\mathbf{0},\ \mathbf{D}_{qq}=\mathbf{0}, \tag{2.96}$$

$$\begin{aligned}\mathbf{D}_{rr}=\mathbf{S}_{r\alpha}\left(\mathbf{C}_{\alpha\alpha}\mathbf{S}_{r\alpha}^{\dagger}+\mathbf{C}_{\alpha\beta}\mathbf{S}_{r\beta}^{\dagger}-\mathbf{C}_{\alpha r}\right)+\mathbf{S}_{r\beta}\left(\mathbf{C}_{\alpha\beta}^{\dagger}\mathbf{S}_{r\alpha}^{\dagger}+\mathbf{C}_{\beta\beta}\mathbf{S}_{r\beta}^{\dagger}-\mathbf{C}_{\beta r}\right)\\ -\mathbf{C}_{\alpha r}^{\dagger}\mathbf{S}_{r\alpha}^{\dagger}-\mathbf{C}_{\beta r}^{\dagger}\mathbf{S}_{r\beta}^{\dagger}+\mathbf{C}_{rr}.\end{aligned} \tag{2.97}$$

$$\mathbf{X}_{\alpha}\triangleq\mathbf{C}_{\alpha\alpha}\mathbf{S}_{r\alpha}^{\dagger}+\mathbf{C}_{\alpha\beta}\mathbf{S}_{r\beta}^{\dagger}-\mathbf{C}_{\alpha r}, \tag{2.98}$$

$$\mathbf{X}_{\beta}\triangleq\mathbf{C}_{\alpha\beta}^{\dagger}\mathbf{S}_{r\alpha}^{\dagger}+\mathbf{C}_{\beta\beta}\mathbf{S}_{r\beta}^{\dagger}-\mathbf{C}_{\beta r}, \tag{2.99}$$

$$\mathbf{X}_{r}\triangleq\mathbf{C}_{\alpha r}^{\dagger}\mathbf{S}_{r\alpha}^{\dagger}+\mathbf{C}_{\beta r}^{\dagger}\mathbf{S}_{r\beta}^{\dagger}-\mathbf{C}_{rr}, \tag{2.100}$$

$$\mathbf{X}_{q}\triangleq\mathbf{0},\ \mathbf{Y}_{\alpha}\triangleq\mathbf{0},\ \mathbf{Y}_{r}\triangleq\mathbf{0},\ \mathbf{Y}_{\beta}\triangleq\mathbf{0},\ \mathbf{Y}_{q}\triangleq\mathbf{0}, \tag{2.101}$$

$$\mathbf{D}_{11}\triangleq\mathbf{D}_{rr}^{-1},\ \mathbf{D}_{12}=\mathbf{0},\ \mathbf{D}_{12}^{\dagger}=\mathbf{0},\ \mathbf{D}_{12}^{\dagger}=\mathbf{0},\ \mathbf{D}_{22}=\mathbf{0}, \tag{2.102}$$

$$\boldsymbol{\alpha}^{pred}=\boldsymbol{\alpha}^{0}-\mathbf{X}_{\alpha}\mathbf{D}_{11}\mathbf{r}^{d}\left(\boldsymbol{\alpha}^{0},\boldsymbol{\beta}^{0}\right), \tag{2.103}$$

$$\boldsymbol{\beta}^{pred}=\boldsymbol{\beta}^{0}-\mathbf{X}_{\beta}\mathbf{D}_{11}\mathbf{r}^{d}\left(\boldsymbol{\alpha}^{0},\boldsymbol{\beta}^{0}\right), \tag{2.104}$$

$$\mathbf{r}^{pred} = \mathbf{r}^m - \mathbf{X}_r \mathbf{D}_{11} \mathbf{r}^d \left(\boldsymbol{\alpha}^0, \boldsymbol{\beta}^0 \right), \tag{2.105}$$

$$\mathbf{C}_{\alpha\alpha}^{pred} = \mathbf{C}_{\alpha\alpha} - \mathbf{X}_\alpha \mathbf{D}_{11} \mathbf{X}_\alpha^\dagger, \tag{2.106}$$

$$\mathbf{C}_{rr}^{pred} = \mathbf{C}_{rr} - \mathbf{X}_r \mathbf{D}_{11} \mathbf{X}_r^\dagger, \tag{2.107}$$

$$\mathbf{C}_{\alpha r}^{pred} = \mathbf{C}_{\alpha r} - \mathbf{X}_\alpha \mathbf{D}_{11} \mathbf{X}_r^\dagger, \tag{2.108}$$

$$\mathbf{C}_{\beta\beta}^{opt} = \mathbf{C}_{\beta\beta} - \mathbf{X}_\beta \mathbf{D}_{11} \mathbf{X}_\beta^\dagger, \tag{2.109}$$

$$\mathbf{C}_{\alpha\beta}^{pred} = \mathbf{C}_{\alpha\beta} - \mathbf{X}_\alpha \mathbf{D}_{11} \mathbf{X}_\beta^\dagger, \tag{2.110}$$

$$\mathbf{C}_{\beta r}^{pred} = \mathbf{C}_{\beta r} - \mathbf{X}_\beta \mathbf{D}_{11} \mathbf{X}_r^\dagger. \tag{2.111}$$

As the above expressions clearly demonstrate, the predictive modeling formulation in the "response space" (as has been developed in Chapter 2) allows the consideration of additional parameters for a model without increasing the size of the matrix $\mathbf{D}_{rr}$ to be inverted.

2.3.3 Predictive modeling for Model A with q additional responses, but no additional parameters

In this case, Eq.(2.44) through (2.47) become

$$\mathbf{D}_{rr} = \mathbf{S}_{r\alpha} \mathbf{C}_{\alpha\alpha} \mathbf{S}_{r\alpha}^\dagger - \mathbf{S}_{r\alpha} \mathbf{C}_{\alpha r} - \mathbf{C}_{\alpha r}^\dagger \mathbf{S}_{r\alpha}^\dagger + \mathbf{C}_{rr}, \; Dim\left(\mathbf{D}_{rr}\right) = \left(N_r \times N_r\right), \tag{2.112}$$

$$\mathbf{D}_{rq} = \mathbf{S}_{r\alpha} \mathbf{C}_{\alpha\alpha} \mathbf{S}_{q\alpha}^\dagger - \mathbf{S}_{r\alpha} \mathbf{C}_{\alpha q} - \mathbf{C}_{\alpha r}^\dagger \mathbf{S}_{q\alpha}^\dagger + \mathbf{C}_{rq}, \; Dim\left(\mathbf{D}_{rq}\right) = \left(N_r \times N_q\right), \tag{2.113}$$

$$\mathbf{D}_{qr} = \mathbf{S}_{q\alpha} \mathbf{C}_{\alpha\alpha} \mathbf{S}_{r\alpha}^\dagger - \mathbf{C}_{\alpha q}^\dagger \mathbf{S}_{r\alpha}^\dagger - \mathbf{S}_{q\alpha} \mathbf{C}_{\alpha r} + \mathbf{C}_{rq}^\dagger, \; Dim\left(\mathbf{D}_{qr}\right) = \left(N_q \times N_r\right), \tag{2.114}$$

$$\mathbf{D}_{qq} = \mathbf{S}_{q\alpha} \mathbf{C}_{\alpha\alpha} \mathbf{S}_{q\alpha}^\dagger - \mathbf{S}_{q\alpha} \mathbf{C}_{\alpha q} - \mathbf{C}_{\alpha q}^\dagger \mathbf{S}_{q\alpha}^\dagger + \mathbf{C}_{qq}, \; Dim\left(\mathbf{D}_{qq}\right) = \left(N_q \times N_q\right). \tag{2.115}$$

$$\mathbf{X}_\alpha \triangleq \mathbf{C}_{\alpha\alpha} \mathbf{S}_{r\alpha}^\dagger - \mathbf{C}_{\alpha r}, \tag{2.116}$$

$$\mathbf{Y}_\alpha \triangleq \mathbf{C}_{\alpha\alpha} \mathbf{S}_{q\alpha}^\dagger - \mathbf{C}_{\alpha q}, \tag{2.117}$$

$$\mathbf{X}_\beta \triangleq \mathbf{0}, \ \ \mathbf{Y}_\beta \triangleq \mathbf{0}, \tag{2.118}$$

$$\mathbf{X}_r \triangleq \mathbf{C}_{\alpha r}^\dagger \mathbf{S}_{r\alpha}^\dagger - \mathbf{C}_{rr}, \tag{2.119}$$

$$\mathbf{Y}_r \triangleq \mathbf{C}_{\alpha r}^\dagger \mathbf{S}_{q\alpha}^\dagger - \mathbf{C}_{rq}^\dagger \tag{2.120}$$

$$\mathbf{X}_q \triangleq \mathbf{C}_{\alpha q}^\dagger \mathbf{S}_{r\alpha}^\dagger - \mathbf{C}_{rq}^\dagger, \tag{2.121}$$

$$\mathbf{Y}_q \triangleq \mathbf{C}_{\alpha q}^\dagger \mathbf{S}_{q\alpha}^\dagger - \mathbf{C}_{qq}, \tag{2.122}$$

$$\boldsymbol{\alpha}^{pred} = \boldsymbol{\alpha}^0 - \left[\mathbf{X}_\alpha \mathbf{D}_{11} + \mathbf{Y}_\alpha \mathbf{D}_{12}^\dagger\right]\mathbf{r}^d\left(\boldsymbol{\alpha}^0, \boldsymbol{\beta}^0\right) - \left[\mathbf{X}_\alpha \mathbf{D}_{12} + \mathbf{Y}_\alpha \mathbf{D}_{22}\right]\mathbf{q}^d\left(\boldsymbol{\alpha}^0, \boldsymbol{\beta}^0\right), \tag{2.123}$$

$$\mathbf{r}^{pred} = \mathbf{r}^m - \left[\mathbf{X}_r \mathbf{D}_{11} + \mathbf{Y}_r \mathbf{D}_{12}^\dagger\right]\mathbf{r}^d\left(\boldsymbol{\alpha}^0, \boldsymbol{\beta}^0\right) - \left[\mathbf{X}_r \mathbf{D}_{12} + \mathbf{Y}_r \mathbf{D}_{22}\right]\mathbf{q}^d\left(\boldsymbol{\alpha}^0, \boldsymbol{\beta}^0\right), \tag{2.124}$$

$$\mathbf{q}^{pred} = \mathbf{q}^m - \left[\mathbf{X}_q \mathbf{D}_{11} + \mathbf{Y}_q \mathbf{D}_{12}^\dagger\right]\mathbf{r}^d\left(\boldsymbol{\alpha}^0, \boldsymbol{\beta}^0\right) - \left[\mathbf{X}_q \mathbf{D}_{12} + \mathbf{Y}_q \mathbf{D}_{22}\right]\mathbf{q}^d\left(\boldsymbol{\alpha}^0, \boldsymbol{\beta}^0\right), \tag{2.125}$$

$$\mathbf{C}_{\alpha\alpha}^{pred} = \mathbf{C}_{\alpha\alpha} - \left[\mathbf{X}_\alpha\left(\mathbf{D}_{11}\mathbf{X}_\alpha^\dagger + \mathbf{D}_{12}\mathbf{Y}_\alpha^\dagger\right) + \mathbf{Y}_\alpha\left(\mathbf{D}_{21}\mathbf{X}_\alpha^\dagger + \mathbf{D}_{22}\mathbf{Y}_\alpha^\dagger\right)\right], \tag{2.126}$$

$$\mathbf{C}_{rr}^{pred} = \mathbf{C}_{rr} - \left[\mathbf{X}_r\left(\mathbf{D}_{11}\mathbf{X}_r^\dagger + \mathbf{D}_{12}\mathbf{Y}_r^\dagger\right) + \mathbf{Y}_r\left(\mathbf{D}_{21}\mathbf{X}_r^\dagger + \mathbf{D}_{22}\mathbf{Y}_r^\dagger\right)\right], \tag{2.127}$$

$$\mathbf{C}_{\alpha r}^{pred} = \mathbf{C}_{\alpha r} - \left[\mathbf{X}_\alpha\left(\mathbf{D}_{11}\mathbf{X}_r^\dagger + \mathbf{D}_{12}\mathbf{Y}_r^\dagger\right) + \mathbf{Y}_\alpha\left(\mathbf{D}_{21}\mathbf{X}_r^\dagger + \mathbf{D}_{22}\mathbf{Y}_r^\dagger\right)\right], \tag{2.128}$$

$$\mathbf{C}_{qq}^{pred} = \mathbf{C}_{qq} - \left[\mathbf{X}_q\left(\mathbf{D}_{11}\mathbf{X}_q^\dagger + \mathbf{D}_{12}\mathbf{Y}_q^\dagger\right) + \mathbf{Y}_q\left(\mathbf{D}_{21}\mathbf{X}_q^\dagger + \mathbf{D}_{22}\mathbf{Y}_q^\dagger\right)\right], \tag{2.129}$$

$$\mathbf{C}_{\alpha q}^{pred} = \mathbf{C}_{\alpha q} - \left[\mathbf{X}_\alpha\left(\mathbf{D}_{11}\mathbf{X}_q^\dagger + \mathbf{D}_{12}\mathbf{Y}_q^\dagger\right) + \mathbf{Y}_\alpha\left(\mathbf{D}_{21}\mathbf{X}_q^\dagger + \mathbf{D}_{22}\mathbf{Y}_q^\dagger\right)\right], \tag{2.130}$$

$$\mathbf{C}_{rq}^{pred} = \mathbf{C}_{rq} - \left[\mathbf{X}_r\left(\mathbf{D}_{11}\mathbf{X}_q^\dagger + \mathbf{D}_{12}\mathbf{Y}_q^\dagger\right) + \mathbf{Y}_r\left(\mathbf{D}_{21}\mathbf{X}_q^\dagger + \mathbf{D}_{22}\mathbf{Y}_q^\dagger\right)\right], \tag{2.131}$$

$$\mathbf{C}_{\alpha\beta}^{pred} = \mathbf{0}, \ \mathbf{C}_{\beta\beta}^{opt} = \mathbf{0}, \ \mathbf{C}_{\beta r}^{pred} = \mathbf{0}, \ \mathbf{C}_{\beta q}^{opt} = \mathbf{0}. \tag{2.132}$$

Note also that (to first-order in response sensitivities) the covariance matrices of the computed responses arising from the uncertainties in the model parameters become:

$$\mathbf{C}_{rr}^{comp} \triangleq \left\langle \left[\mathbf{r} - \mathbf{r}^c\left(\boldsymbol{\alpha}^0, \boldsymbol{\beta}^0\right)\right]\left[\mathbf{r} - \mathbf{r}^c\left(\boldsymbol{\alpha}^0, \boldsymbol{\beta}^0\right)\right]^\dagger \right\rangle = \mathbf{S}_{r\alpha}\mathbf{C}_{\alpha\alpha}\mathbf{S}_{r\alpha}^\dagger, \tag{2.133}$$

$$\mathbf{C}_{qq}^{comp} \triangleq \left\langle \left[\mathbf{q} - \mathbf{q}^{c}\left(\boldsymbol{\alpha}^{0}, \boldsymbol{\beta}^{0}\right) \right] \left[\mathbf{q} - \mathbf{q}^{c}\left(\boldsymbol{\alpha}^{0}, \boldsymbol{\beta}^{0}\right) \right]^{\dagger} \right\rangle = \mathbf{S}_{q\alpha} \mathbf{C}_{\alpha\alpha} \mathbf{S}_{q\alpha}^{\dagger}, \tag{2.134}$$

$$\mathbf{C}_{rq}^{comp} \triangleq \left\langle \left[\mathbf{r} - \mathbf{r}^{c}\left(\boldsymbol{\alpha}^{0}, \boldsymbol{\beta}^{0}\right) \right] \left[\mathbf{q} - \mathbf{q}^{c}\left(\boldsymbol{\alpha}^{0}, \boldsymbol{\beta}^{0}\right) \right]^{\dagger} \right\rangle = \mathbf{S}_{r\alpha} \mathbf{C}_{\alpha\alpha} \mathbf{S}_{q\alpha}^{\dagger}. \tag{2.135}$$

3 PREDICTIVE MODELING OF A SIMPLE NEUTRON DIFFUSION MODEL

The results presented in this Chapter are based on the work by Cacuci (2014). Consider the diffusion of monoenergetic neutrons due to distributed sources of strength S $\text{neutrons}/\text{cm}^3 \cdot \text{s}$ within a slab of material of extrapolated thickness $2a$. The linear neutron diffusion equation that models mathematically this problem is

$$D\frac{d^2\varphi}{dx^2} - \Sigma_a \varphi + S = 0\,, \quad x \in (-a, a)\,, \tag{3.1}$$

where $\varphi(x)$ is the neutron flux, D is the diffusion coefficient, Σ_a is the macroscopic absorption cross section, and S is the distributed source term. Note that, in view of the problem's symmetry, the origin $x = 0$ has been conveniently chosen at the middle (center) of the slab. The boundary conditions for Eq.(3.1) are that the neutron flux must vanish at the extrapolated distance, i.e.,

$$\varphi(\pm a) = 0\,. \tag{3.2}$$

A typical response R for the neutron diffusion problem modeled by Eqs. (2.1) and (2.2) would be the reading of a detector placed within the slab, for example, at a distance b from the slab's midline at $x = 0$. Such a response is given by the reaction rate

$$R(\boldsymbol{e}) \triangleq \Sigma_d \varphi(b)\,, \tag{3.3}$$

where Σ_d represents the detector's equivalent reaction cross section. The system parameters for this problem are thus the positive constants Σ_a, D, S, and Σ_d, which will be considered to be the components of the vector $\boldsymbol{\alpha}$ of system parameters, defined as

$$\boldsymbol{\alpha} \triangleq \left(\Sigma_a,\, D,\, S,\, \Sigma_d\right). \tag{3.4}$$

Consider that the components of $\boldsymbol{\alpha} \triangleq \left(\Sigma_a,\, D,\, S,\, \Sigma_d\right)$ are imprecisely (e.g., experimentally) determined quantities, with mean nominal values $\boldsymbol{\alpha}^0 \triangleq \left(\Sigma_a^0,\, D^0,\, S^0,\, \Sigma_d^0\right)$ and standard deviations $\boldsymbol{h}_\alpha \triangleq \left(\delta\Sigma_a,\, \delta D,\, \delta S,\, \delta\Sigma_d\right)$, respectively. The vector $\boldsymbol{e}(x)$ appearing in the functional dependence of R in Eq.(3.3) denotes the concatenation of $\varphi(x)$ with $\boldsymbol{\alpha}$, defined as

$$\boldsymbol{e} \triangleq \left(\varphi,\, \boldsymbol{\alpha}\right). \tag{3.5}$$

The nominal value $\varphi^0(x)$ of the flux is determined by solving Eqs.(3.1) and (3.2) for the nominal parameter values $\boldsymbol{\alpha}^0 = \left(\Sigma_a^0,\, D^0,\, S^0,\, \Sigma_d^0\right)$, to obtain

$$\varphi^0(x) = \frac{S^0}{\Sigma_a^0}\left(1 - \frac{\cosh xk}{\cosh ak}\right), \quad k \equiv \sqrt{\Sigma_a^0 / D^0}\,, \tag{3.6}$$

where $k \triangleq \sqrt{\Sigma_a^0 / D^0}$ is the nominal value of the reciprocal diffusion length for our illustrative example. Inserting Eq.(3.6) together with the nominal value Σ_d^0 into Eq.(3.3) gives the nominal value of the response:

$$R\left(\boldsymbol{e}^0\right) = \frac{S^0 \Sigma_d^0}{\Sigma_a^0}\left(1 - \frac{\cosh bk}{\cosh ak}\right), \quad \boldsymbol{e}^0 \triangleq \left(\varphi^0, \boldsymbol{\alpha}^0\right). \tag{3.7}$$

Note that even though Eq.(3.1) is linear in φ, the solution $\varphi(x)$ depends nonlinearly on $\boldsymbol{\alpha}$, as evidenced by Eq.(3.6). The same is true of the response $R(\boldsymbol{e})$. Even though $R(\boldsymbol{e})$ is linear separately in φ and in $\boldsymbol{\alpha}$, as shown in Eq.(3.3), R is not simultaneously linear in φ and $\boldsymbol{\alpha}$, which leads to a nonlinear dependence of $R(\boldsymbol{e})$ on $\boldsymbol{\alpha}$. This fact is confirmed by the explicit expression of $R(\boldsymbol{e})$ given in Eq.(3.7).

The sensitivities of the system response to the system parameters have been computed efficiently using the Adjoint Sensitivity Analysis Methodolgy in the work of Cacuci (2014), and are reproduced below:

$$\frac{\partial R}{\partial S} = \frac{\Sigma_d^0}{\Sigma_a^0}\left(1 - \frac{\cosh bk}{\cosh ak}\right), \tag{3.8}$$

$$\frac{\partial R}{\partial \Sigma_d} = \frac{S^0}{\Sigma_a^0}\left(1 - \frac{\cosh bk}{\cosh ak}\right), \tag{3.9}$$

$$\frac{\partial R}{\partial \Sigma_a} = -\frac{S^0\Sigma_d^0}{\left(\Sigma_a^0\right)^2}\left(1 - \frac{\cosh bk}{\cosh ak}\right) + \frac{1}{2\sqrt{D^0\Sigma_a^0}}\frac{S^0\Sigma_d^0}{\Sigma_a^0}\frac{a\sinh ak\cosh bk - b\sinh bk\cosh ak}{\left(\cosh ak\right)^2}, \tag{3.10}$$

$$\frac{\partial R}{\partial D} = -\frac{1}{2}\sqrt{\frac{\Sigma_a^0}{D^0}}\frac{S^0\Sigma_d^0}{D^0\Sigma_a^0}\frac{a\sinh ak\cosh bk - b\sinh bk\cosh ak}{\left(\cosh ak\right)^2}. \tag{3.11}$$

To illustrate with numerical values the application of these formulas, consider that the slab of extrapolated thickness a consists of water with material properties having the following nominal values: $\Sigma_a^0 = 0.0197\,cm^{-1}$, $D^0 = 0.16\ cm$, containing distributed neutron sources emitting nominally $S^0 = 10^7\,neutrons \cdot cm^{-3} \cdot s^{-1}$. For the sake of argument, consider that all of these parameters are uncorrelated and have the following relative standard deviations: $\Delta\Sigma_a^0 / \Sigma_a^0 = 5\%$, $\Delta D^0 / D^0 = 5\%$, $\Delta S^0 / S^0 = 15\%$.

Furthermore, consider that measurements are performed with an infinitely thin detector immersed at different locations, $x = b$, in the water slab, having an indium-like nominal detector cross section $\Sigma_d^0 = 7.438\,cm^{-1}$, uncorrelated to the other parameters, with a standard deviation $\Delta\Sigma_d^0 / \Sigma_d^0 = 10\%$. Collecting this information (and omitting, for simplicity, the respective units), it follows that the covariance matrix for the model parameters is

$$\mathbf{C}_\alpha = \begin{pmatrix} \left(9.85\times10^{-4}\right)^2 & 0 & 0 & 0 \\ 0 & \left(8.0\times10^{-3}\right)^2 & 0 & 0 \\ 0 & 0 & \left(1.5\times10^{6}\right)^2 & 0 \\ 0 & 0 & 0 & \left(7.44\times10^{-1}\right)^2 \end{pmatrix}. \tag{3.12}$$

To illustrate the effects of several consistent measurements, and also to test that symmetric measurements (with respect to the vertical plane through the origin) do preserve the solution's symmetry, we consider four consistent ($\chi^2 = 1.21$) measurements, taken at the symmetric locations $10\,cm, -10\,cm, \; -40\,cm, 40\,cm$, and having the following values and relative standard deviations (abbreviated as "*rsd*"):

$$r_1^m \triangleq r\left(meas.at\;\; 10\,cm\right) = 3.40\times 10^9\, n\cdot cm^{-3}\cdot \sec^{-1};\; rsd\left(r_1^m\right) = 5\%; \tag{3.13}$$

$$r_2^m \triangleq r\left(meas.at\;\; -10\,cm\right) = 3.59\times 10^9\, n\cdot cm^{-3}\cdot \sec^{-1};\; rsd\left(r_2^m\right) = 6\%; \tag{3.14}$$

$$r_3^m \triangleq r\left(meas.at\;\; -40\,cm\right) = 3.77\times 10^9\, n\cdot cm^{-3}\cdot \sec^{-1};\; rsd\left(r_3^m\right) = 5\%; \tag{3.15}$$

$$r_4^m \triangleq r\left(meas.at\;\; 40\,cm\right) = 3.74\times 10^9\, n\cdot cm^{-3}\cdot \sec^{-1};\; rsd\left(r_4^m\right) = 5\%; \tag{3.16}$$

Thus, the covariance matrix of the measured responses is

$$\mathbf{C}_m = \begin{pmatrix} \left(1.7\times 10^8\right)^2 & 0 & 0 & 0 \\ 0 & \left(2.15\times 10^8\right)^2 & 0 & 0 \\ 0 & 0 & \left(1.89\times 10^8\right)^2 & 0 \\ 0 & 0 & 0 & \left(1.87\times 10^8\right)^2 \end{pmatrix} \tag{3.17}$$

The nominal values of the computed responses at the above locations are as follows:

$$r_1\left(comp.\ at\;\ 10\,cm\right) = 3.77\times 10^9\, n\cdot cm^{-3}\cdot \sec^{-1}; \tag{3.18}$$

$$r_2\left(comp.\ at\;\ -10\,cm\right) = 3.77\times 10^9\, n\cdot cm^{-3}\cdot \sec^{-1}; \tag{3.19}$$

$$r_3\left(comp.\ \ at\;\ -40\,cm\right) = 3.66\times 10^9\, n\cdot cm^{-3}\cdot \sec^{-1}; \tag{3.20}$$

$$r_4\left(comp.\ \ at\;\ 40\,cm\right) = 3.66\times 10^9\, n\cdot cm^{-3}\cdot \sec^{-1}; \tag{3.21}$$

As expected, the above computed responses confirm the problem's symmetry. The matrices S and S_{rel}, with $\Delta\alpha_j \triangleq std.\,dev.\left(\alpha_j\right)$, containing the nominal values of the absolute and relative sensitivities, respectively, are:

$$\mathbf{S} \triangleq \left(\frac{\partial R_i}{\partial \alpha_j}\right) = \begin{pmatrix} -1.92\times10^{11} & -1.33\times10^{5} & 3.78\times10^{2} & 5.08\times10^{8} \\ -1.92\times10^{11} & -1.33\times10^{5} & 3.78\times10^{2} & 5.08\times10^{8} \\ -1.76\times10^{11} & -1.24\times10^{9} & 3.66\times10^{2} & 4.92\times10^{8} \\ -1.76\times10^{11} & -1.24\times10^{9} & 3.66\times10^{2} & 4.92\times10^{8} \end{pmatrix}, \tag{3.22}$$

$$\mathbf{S}_{rel} \triangleq \left(\frac{\partial R_i}{\partial \alpha_j}\frac{\Delta\alpha_j}{R_i}\right) = \begin{pmatrix} -0.99999 & -5.41\times10^{-6} & 1.00 & 1.00 \\ -0.99999 & -5.64\times10^{-6} & 1.00 & 1.00 \\ -9.46\times10^{-1} & -5.64\times10^{-2} & 1.00 & 1.00 \\ -9.46\times10^{-1} & -5.41\times10^{-2} & 1.00 & 1.00 \end{pmatrix}. \tag{3.23}$$

Using the above sensitivities together with the parameter covariance matrix given in Eq.(3.12) yields the following value for the covariance matrix of the computed responses:

$$\mathbf{C}_{rc} = \mathbf{S}\mathbf{C}_{\alpha}\mathbf{S}^{\dagger} = \begin{pmatrix} 4.99\times10^{17} & 4.99\times10^{17} & 4.82\times10^{17} & 4.82\times10^{17} \\ 4.99\times10^{17} & 4.99\times10^{17} & 4.82\times10^{17} & 4.82\times10^{17} \\ 4.82\times10^{17} & 4.82\times10^{17} & 4.66\times10^{17} & 4.66\times10^{17} \\ 4.82\times10^{17} & 4.82\times10^{17} & 4.66\times10^{17} & 4.66\times10^{17} \end{pmatrix} \tag{3.24}$$

Note that the particular values (essentially either unity or zero) of the components of the sensitivity matrix lead to a fully correlated covariance matrix for the four computed responses.
Applying the PM-CMPS to the above information leads to the following optimal best-estimate parameter values, relative standard deviations (abbreviated as "rsd"), and covariance matrix:

$$\Sigma_a^{be} = 0.0198\,cm^{-1},\;\; rsd\left(\Sigma_a^{be}\right) = 4.79\%; \tag{3.25}$$

$$D^{be} = 0.1591\;cm,\;\; rsd\left(D^{be}\right) = 5.00\%; \tag{3.26}$$

$$S^{be} = 9.85\times10^{6}\,n\cdot cm^{-3}\cdot s^{-1},\;\; rsd\left(S^{be}\right) = 9.21\%; \tag{3.27}$$

$$\Sigma_d^{be} = 7.388\,cm^{-1},\;\;\; rsd\left(\Sigma_d^{be}\right) = 8.53\%; \tag{3.28}$$

$$
\mathbf{C}_{\alpha}^{be} = \begin{pmatrix} 9.50\times10^{-4} & 0 & 0 & 0 \\ 0 & 7.99\times10^{-3} & 0 & 0 \\ 0 & 0 & 9.08\times10^{5} & 0 \\ 0 & 0 & 0 & 6.30\times10^{-1} \end{pmatrix}
\times \begin{pmatrix} 1.0 & -8.89\times10^{-4} & 3.51\times10^{-1} & 1.67\times10^{-1} \\ -8.89\times10^{-4} & 1.0 & 1.02\times10^{-2} & 4.84\times10^{-3} \\ 3.51\times10^{-1} & 1.02\times10^{-2} & 1.0 & -8.24\times10^{-1} \\ 1.67\times10^{-1} & 4.84\times10^{-3} & -8.24\times10^{-1} & 1.0 \end{pmatrix}
\times \begin{pmatrix} 9.50\times10^{-4} & 0 & 0 & 0 \\ 0 & 7.99\times10^{-3} & 0 & 0 \\ 0 & 0 & 9.08\times10^{5} & 0 \\ 0 & 0 & 0 & 6.30\times10^{-1} \end{pmatrix}, \tag{3.29}
$$

Furthermore, the best estimate response values, relative standard deviations (abbreviated as "*rsd*"), and covariance matrix are as follows:

$$
at\ (10\,cm):\ r_1^{be} = 3.66\times10^{9}\, n\cdot cm^{-3}\cdot \sec^{-1};\ rsd\left(r_1^{be}\right) = 2.59\%; \tag{3.30}
$$

$$
at\ (-10\,cm):\ r_2^{be} = 3.66\times10^{9}\, n\cdot cm^{-3}\cdot \sec^{-1};\ rsd\left(r_2^{be}\right) = 2.59\%; \tag{3.31}
$$

$$
at\ (-40\,cm):\ r_3^{be} = 3.56\times10^{9}\, n\cdot cm^{-3}\cdot \sec^{-1};\ rsd\left(r_3^{be}\right) = 2.58\%; \tag{3.32}
$$

$$
at\ (40\,cm):\ r_4^{be} = 3.56\times10^{9}\, n\cdot cm^{-3}\cdot \sec^{-1};\ rsd\left(r_4^{be}\right) = 2.58\%; \tag{3.33}
$$

$$
\mathbf{C}_{r}^{be} = \begin{pmatrix} 9.04\times10^{15} & 9.04\times10^{15} & 8.64\times10^{15} & 8.64\times10^{15} \\ 9.04\times10^{15} & 9.04\times10^{15} & 8.64\times10^{15} & 8.64\times10^{15} \\ 8.64\times10^{15} & 8.64\times10^{15} & 8.45\times10^{15} & 8.45\times10^{15} \\ 8.64\times10^{15} & 8.64\times10^{15} & 8.45\times10^{15} & 8.45\times10^{15} \end{pmatrix} \tag{3.34}
$$

The best-estimate predicted response-parameter correlation matrix is:

$$\mathbf{C}_{r\alpha}^{be} = \begin{pmatrix} -7.81\times10^{3} & 3.89\times10^{4} & 1.38\times10^{13} & 4.57\times10^{6} \\ -7.81\times10^{3} & 3.89\times10^{4} & 1.38\times10^{13} & 4.57\times10^{6} \\ 1.50\times10^{3} & -4.13\times10^{4} & 1.64\times10^{13} & 5.41\times10^{6} \\ 1.50\times10^{3} & -4.13\times10^{4} & 1.64\times10^{13} & 5.41\times10^{6} \end{pmatrix}. \tag{3.35}$$

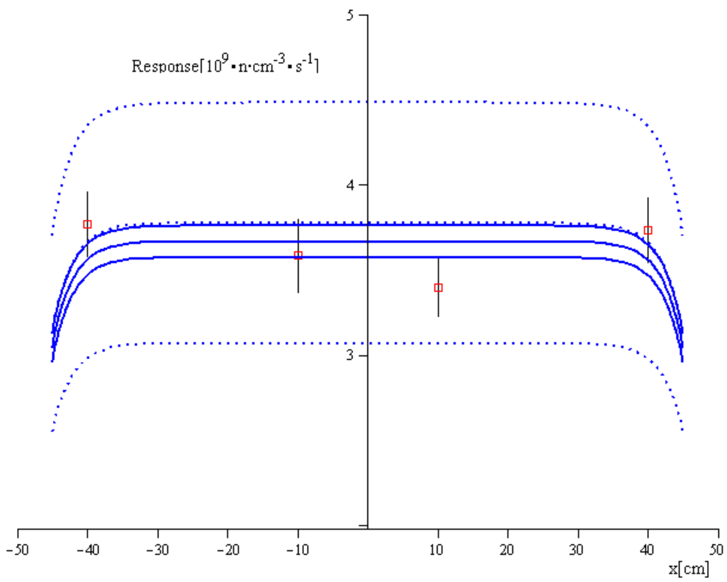

Figure 3.1: Four precise consistent precise measurements ($\chi^2 = 1.21$)

Figure 3.1 shows the spatial variation of the original nominal computed values and standard deviations (depicted using solid lines) together with the best estimate response values and corresponding standard deviations (depicted using broken lines). The value of $\chi^2 = 1.21$ indicates a very good consistency among the four measurements.

4 INVERSE PREDICTIVE MODELING OF RADIATION TRANSPORT THROUGH OPTICALLY THICK MEDIA IN THE PRESENCE OF COUNTING UNCERTAINTIES

Abstract

This Chapter is based on the work by Cacuci (2017), and illustrates the application of the PM-CMPS methodology to the problem of *inverse prediction*, from detector responses in the presence of counting uncertainties, of the thickness of a homogeneous slab of material containing uniformly distributed gamma-emitting sources, for optically thin and thick slabs. For optically thin slabs, this Section shows that both the traditional chi-square-minimization method and the PM-CMPS methodology predict the slab's thickness accurately. However, the PM-CMPS methodology is considerably more efficient computationally, and a single application of the PM-CMPS methodology predicts the thin slab's thickness at least as precisely as the traditional chi-square-minimization method, even though the measurements used in the PM-CMPS methodology were ten times less accurate than the ones used for the traditional chi-square-minimization method. For optically thick slabs, the results obtained in this work show that: (i) the traditional inverse-problem methods based on the minimization of chi-square-type functionals fail to predict the slab's thickness; (ii) the PM-CMPS methodology under-predicts the slab's actual physical thickness when imprecise experimental results are assimilated, even though the predicted responses agrees within the imposed error criterion with the experimental results; (iii) the PM-CMPS methodology correctly predicts the slab's actual physical thickness when precise experimental results are assimilated, while also predicting the physically correct response within the selected precision criterion; and (iv) the PM-CMPS methodology is computational vastly more efficient while yielding significantly more accurate results than the traditional chi-square-minimization methodology.

4.1 Transport of Uncollided Photons through a Slab

Consider a one-dimensional slab of homogeneous material extending from $z=0$ to $z=a\left[cm\right]$, placed in air and characterized by a total interaction coefficient $\mu\left[cm^{-1}\right]$. The slab contains a uniformly distributed source of strength $Q\left[photons / cm^3 \sec \right]$ emitting isotropically monoenergetic photons within the slab. It is assumed that there is no scattering into the energy lines. Under these conditions, the angular flux of photons within the slab is described by the Boltzmann transport equation without scattering and with "vacuum" incoming boundary condition, i.e.,

$$\omega \frac{d\psi(z,\omega)}{dz} + \mu \psi(z,\omega) = \frac{Q}{2}, \quad 0 < z \le a, \ \omega > 0, \tag{4.1}$$

$$\psi(0,\omega) = 0. \tag{4.2}$$

where $\psi(z,\omega)$ denotes the neutron angular flux at position z and direction $\omega \triangleq \cos\theta$, where θ denotes the angle between the photon's direction and the z-axis. The solution of Eqs.(4.1) and (4.2) can be readily obtained as

$$\psi(z,\omega) = \frac{Q}{2\mu}\left[1 - \exp(\mu z/\omega) \right]. \tag{4.3}$$

Consider further that the leakage flux of uncollided photons is measured by an "infinite plane" detector placed in air at some location $z > a$ external to the slab. The detector's response function, denoted as $\Sigma_d \left[cm^{-1} \right]$, is considered to be a perfectly well-known constant. If the detection process were a perfectly deterministic process, rather than a stochastic one, it would follow from Eq.(4.3) that the "exact detector response", denoted as $r(\mu a)$, would be given by the expression

$$r(\mu a) \triangleq \Sigma_d \int_0^1 \psi(z,\omega)\, d\omega = \frac{Q\Sigma_d}{2\mu}\left[1 - E_2(\mu a) \right], \tag{4.4}$$

where the exponential-integral function is defined as

$$E_n(x) = \int_0^1 u^{n-2} e^{-x/u} du, \quad n = 0,1,2,... \tag{4.5}$$

4.2 Determination of Slab Thickness from Detector Response in the Absence of Uncertainties

Since the focus of this work is the determination of the slab's optical thickness from detector measurements, the quantities Σ_d, μ, and Q will be considered to be perfectly well known. Without loss of generality, these quantities can be normalized to unity, i.e.: $Q = 1\left[photons / cm^3 \sec \right], \quad \Sigma_d = 1\left[cm^{-1} \right], \quad \mu = 1\left[cm^{-1} \right]$. If the detector were perfect and if its response $r\left(\mu a\right)$ were the consequence of an exactly-known deterministic counting process, Eq. (4) could be "inverted" to obtain the slab's optical thickness $\left(\mu a\right)$ by solving deterministically the following nonlinear equation:

$$E_2\left(x\right) = 1 - \frac{2\mu r\left(x\right)}{Q\Sigma_d} \triangleq C, \;\; x \triangleq \mu a . \tag{4.6}$$

When $r\left(x\right)$ is known, the right-side of Eq.(4.6) is a known constant, denoted as *C*. Since the function $E_1\left(x\right)$ is everywhere positive, i.e., $E_1\left(x\right) > 0, \;\; for \;\; 0 < x < \infty$, it follows that

$$\frac{dE_2\left(x\right)}{dx} = -E_1\left(x\right) < 0, \;\; 0 < x < \infty . \tag{4.7}$$

The result in Eq.(4.7) indicates that $E_2\left(x\right)$ is a monotonically decreasing function of x as $x \geq 0$ increases, and the "amount of decrease" increases as x increases. In other words, the value of $E_2\left(x\right)$ decreases monotonically, at an increasingly slower rate, as x increases. Since $E_2\left(0\right) = 1$ and $E_2\left(x\right) \xrightarrow{x \to \infty} 0$, it follows that $E_2\left(x\right)$ will take on at most once each value in the interval $1 \geq E_2\left(x\right) = C > 0$ as x increases monotonically in the interval $0 \leq x < \infty$. Hence, despite the fact that the axis $x = 0$ is asymptotically tangent to $E_2\left(x\right)$ in the limit when $x \to \infty$, Eq.(4.6) admits just a *single real-valued root*. Consequently, for each value of $r\left(\mu a\right)$, which determines the value of *C*, there corresponds a single, well-defined, slab optical thickness $\mu a = x$. In other words, *Eq.*(4.6) *does not admit degenerate roots*, in the sense that more than one distinct value of the slab's optical thickness $(\mu a = x)$ might correspond to the same value $r(\mu a)$. The fact that Eq.(4.6) admits a single real-valued root is also underscored by recalling the asymptotic expansions for $E_2(x)$, i.e.,:

$$E_2(x) \sim \frac{e^{-x}}{x+2}\left[1+\frac{2}{(x+2)^2}+\frac{2(2-2x)}{(x+2)^4}+\frac{2\left(6x^2-16x+4\right)}{(x+2)^6}+\ldots\right] \triangleq A(x), \quad x \triangleq \mu a > 1, \tag{4.8}$$

$$E_2(x) \sim 1 + x\left[\log(x) - 0.422784\right] - \frac{x^2}{2} + \frac{x^3}{12} - \frac{x^4}{72} + \ldots \triangleq B(x), \; x \triangleq \mu a < 1. \tag{4.9}$$

The asymptotic expansion in Eq.(4.8) can be used to compute the real-valued root of Eq.(4.6) for *C<0.8*; (ii) both asymptotic expansions given in Eqs.(4.8) and (4.9) can be used to compute the real-valued root of Eq.(4.6) when *0.2<C<0.8*; (iii) the asymptotic expansion in Eq.(4.9) can be used to compute the real-valued root of Eq.(4.6) when *C>0.2*. The left- and right-sides of the equations

$$A(x) = C, \quad B(x) = C, \tag{4.10}$$

where $A(x)$ and $B(x)$ are defined in Eqs.(4.8) and (4.9), respectively, are plotted in Figure 4.1, below. The intersection of the horizontal line with the decreasing curve depicting the function $E_2(x)$ provides the location of the real root of Eq.(4.6). It is also evident from Eqs.(4.6), (4.8) and (4.9) that in the limit of infinitely thin or infinitely thick slabs, respectively, the corresponding "readings" by perfect detectors would be

$$r(0) = 0, \quad r(\infty) = \frac{Q\Sigma_d}{2\mu}. \tag{4.11}$$

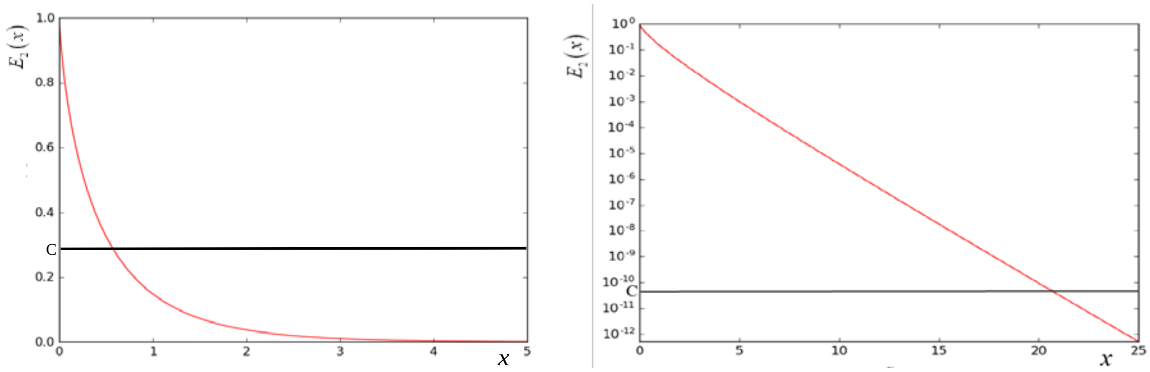

Figure 4.1. Location of the (unique) real root of Eq. (6): Left: linear-linear scale; Right: log-linear scale.

4.3 Traditional Chi-Square Minimization Method for Determining the Slab's Thickness from Detector Responses in the Presence of Counting Uncertainties

It is reasonable to expect that the slab's unknown optical thickness should be obtainable from detector measurements, since the detector measurements implicitly "know" the exact thickness of the slab, which is reflected in the respective number of photons reaching the detector. Also, in the limit of infinite experimental precision and accuracy, the detector response must indicate the exact thickness of the slab, as shown in the previous section. As is well known, the process of detecting photons (as well as other particles) can be described by a Poisson distribution. When a sufficiently large number of events are counted, as is usually the case with photon detection, the respective Poisson distribution can be approximated well by a normal (Gaussian) distribution. For this paradigm example, it suffices to consider that the k^{th}-experimentally-measured response, which will be denoted as $r_{\exp}^{(k)}$, is obtained as a random event drawn from a normal distribution having the mean equal to the exact response, $r(\mu a)$, and the standard deviation equal to $\beta r(\mu a)$, where β is the relative standard deviation (in %), so that

$$r_{\exp}^{(k)} = random\ normal\left[r,\ \beta r\right],\quad k = 1,...,K. \tag{4.12}$$

The current state-of-the-art methods for solving "inverse problems" such as determining the optical dimension of a uniform homogeneous medium from K uncertain photon measurements external to the medium rely on minimizing a user-defined chi-square-type functional of the following form:

$$\chi^2 \triangleq \sum_{k=1}^{K} \left[\frac{r_{\text{model}} - r_{\text{exp}}^{(k)}}{std.dev\left(r_{\text{exp}}\right)} \right]^2 , \tag{4.13}$$

where, for the slab considered in this Section,

$$r_{\text{model}} \triangleq \frac{Q\Sigma_d}{2\mu}\left[1 - E_2\left(\mu a_{\text{model}}^{(k)}\right)\right] \tag{4.14}$$

Since only counting uncertainties in the detector response will be considered in this illustrative example, the quantities Σ_d, μ, and Q will be considered, as in the previous Section, to be perfectly well known and be normalized to unity, i.e., $Q = 1\left[photons / cm^3 \sec\right]$, $\Sigma_d = 1\left[cm^{-1}\right]$, $\mu = 1\left[cm^{-1}\right]$. A direct attempt to determine the slab's optical thickness would be by plotting the difference

$$\delta \triangleq \left(r_{\text{model}} - r_{\text{exp}}\right), \tag{4.15}$$

between a random realization of a detector response, r_{exp}, and the "model response", r_{model}, defined in Eq.(4.14). While studying the behavior of Eq.(4.13), Mattingly (2015) has plotted the behavior of the quantity $\delta \triangleq \left(r_{\text{model}} - r_{\text{exp}}\right)$ as a function of μa_{model}, for various actual slab thicknesses μa. The results in Figure 4.2 were obtained using software based on Mattingly's program to plot the quantity $\delta \triangleq \left(r_{\text{model}} - r_{\text{exp}}\right)$ for four values of the actual optical thickness μa (namely: $\mu a = 0.1$, $\mu a = 1.0$, $\mu a = 3.0$ and $\mu a = 10.0$) and by considering that the corresponding detector response, r_{exp}, is distributed normally with a mean equal to (the exact) r_{model}, and having a relative standard deviation of 1% [i.e., $std.dev\left(r_{\text{exp}}\right) = (0.01)r_{\text{exp}}$]. As the plots in Figure 4.2 indicate, for measurements having a relative standard deviation of 1% (i.e., fairly accurate measurements), the "zero-crossings" of the respective differences $\delta \triangleq \left(r_{\text{model}} - r_{\text{exp}}\right)$ are clearly identifed for optically thin slabs, as exemplified by the graphs for $\mu a = 0.1$ and $\mu a = 1$. These zeros also correctly correspond to the values $\mu a_{\text{model}} = 0.1$ and $\mu a_{\text{model}} = 1$, respectively. On the other hand, for measurements having a relative standard deviation of 1%, the plots corresponding to $\mu a = 3.0$ and $\mu a = 10.0$ in Figure 4.2 indicate that the "zero-crossings" of the corresponding differences $\delta \triangleq (r_{\text{model}} - r_{\text{exp}})$ can no longer be identified beyond about three mean free paths (i.e., $\mu a > 3$); the respective "zero-crossings" appear to be multiple-valued, perhaps even degenerate.

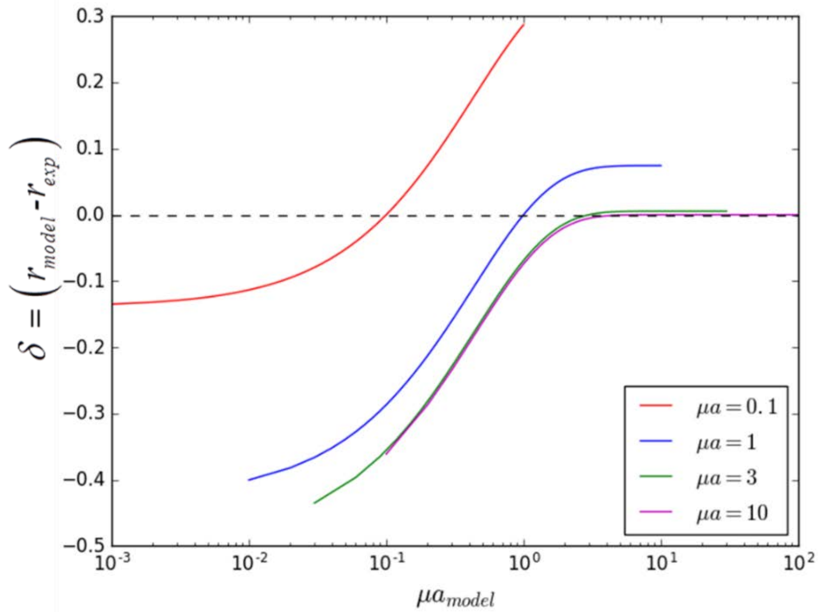

Figure 4.2: Variation of the difference between the computed detector response, r_{model}, and a measured (normally distributed, with a relative standard deviation of 1%) detector response, r_{exp}, as a function of the model's optical thickness (μa_{model}).

Applying various minimization procedures, the value $(\mu a)_{\min}$ which yields the minimum value, $\chi^2_{\min}$, of χ^2 is considered to be the slab's optical thickness. Mattingly (2015) has plotted the quantity $\delta^2 \triangleq \left(r_{\text{model}} - r_{\text{exp}}^{(k)}\right)^2$ as a function of the model's optical thickness (μa_{model}), for various values of the actual optical thickness, μa, and by considering (as before) that the corresponding detector response, r_{exp}, is distributed normally with a mean equal to (the exact) r_{model}. Using software based on Mattingly's program (2015), Figures 4.3 through 4.6 present plots of $\delta^2 \triangleq \left(r_{\text{model}} - r_{\text{exp}}^{(k)}\right)^2$ for the same values (namely: $\mu a = 0.1$, $\mu a = 1.0$, $\mu a = 3.0$ and $\mu a = 10.0$) of the actual optical thickness, μa, as considered in Figure 4.2, for ten measurements $r_{\text{exp}}^{(k)}$, *k=1,...,10*, which are considered to be distributed normally with a mean equal to (the exact response) r_{model} and a relative standard deviation of 1% [i.e., $std.dev\left(r_{\text{exp}}\right) = (0.01) r_{\text{exp}}$].

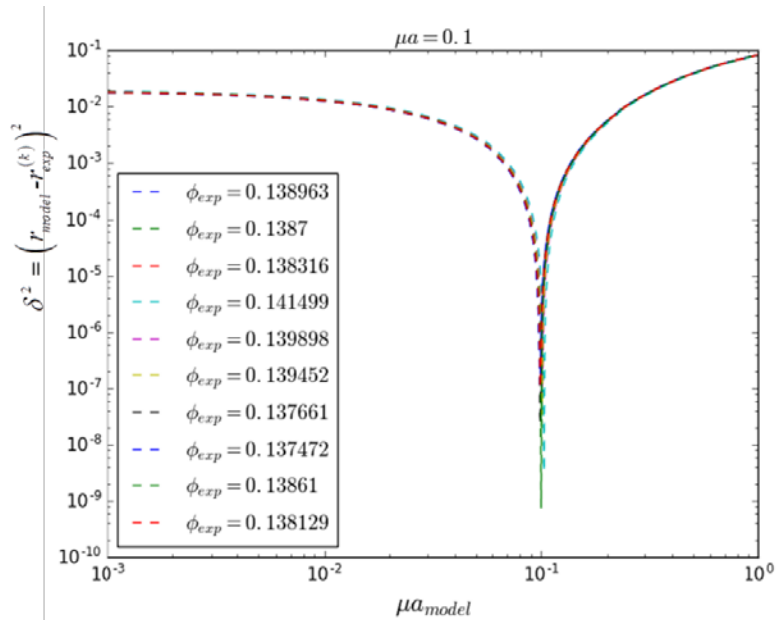

Figure 4.3: Variation of $\delta^2 \triangleq \left(r_{\text{model}} - r_{\text{exp}}^{(k)}\right)^2$ as a function of the model's optical thickness $\left(\mu a_{\text{model}}\right)$ for a slab of actual optical thickness $\mu a = 0.1$, for measurements with a relative standard deviation of 1%.

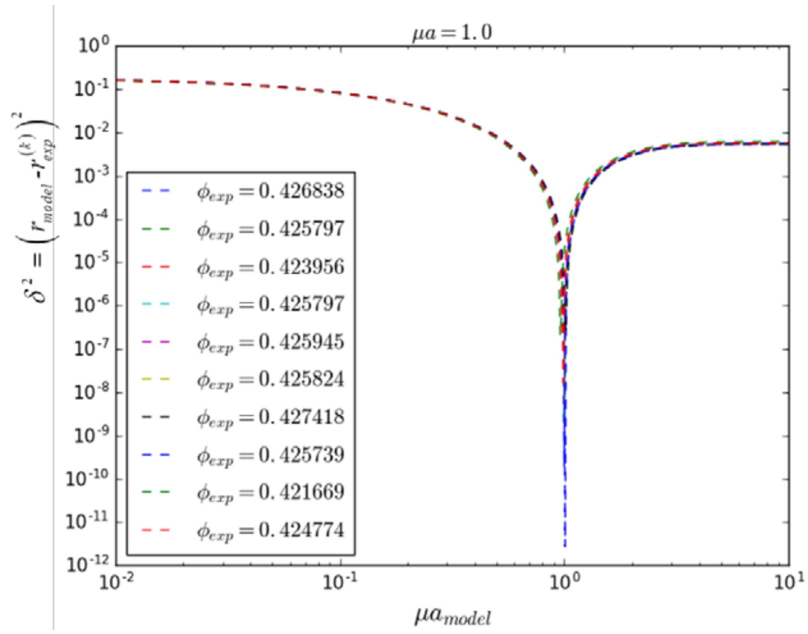

Figure 4.4: Variation of $\delta^2 \triangleq \left(r_{\text{model}} - r_{\text{exp}}^{(k)}\right)^2$ as a function of the model's optical thickness $\left(\mu a_{\text{model}}\right)$ for a slab of actual optical thickness $\mu a = 1.0$, for measurements with a relative standard deviation of 1%.

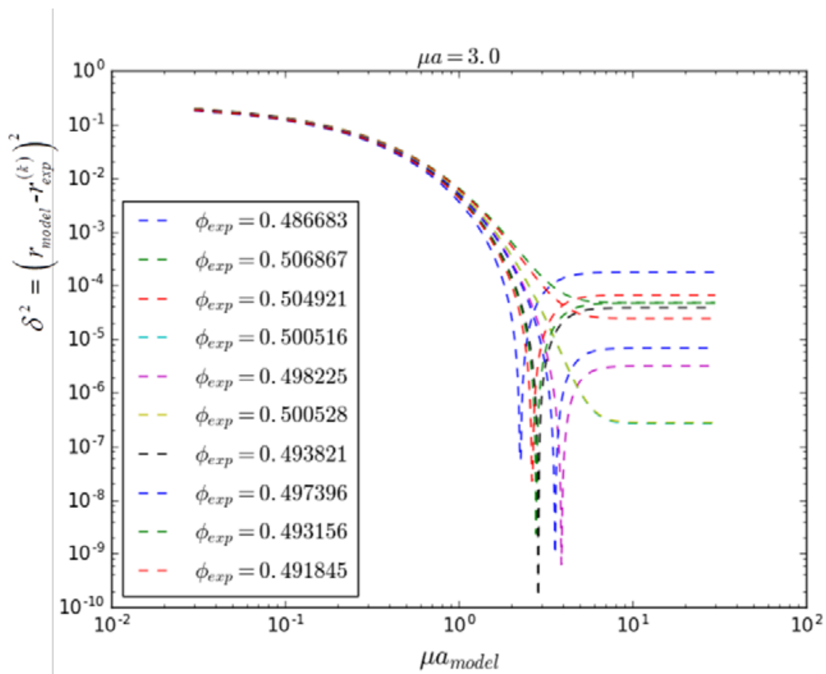

Figure 4.5: Variation of $\delta^2 \triangleq \left(r_{\text{model}} - r_{\text{exp}}^{(k)}\right)^2$ as a function of the model's optical thickness $\left(\mu a_{\text{model}}\right)$ for a slab of actual optical thickness $\mu a = 3.0$, for measurements with a relative standard deviation of 1%.

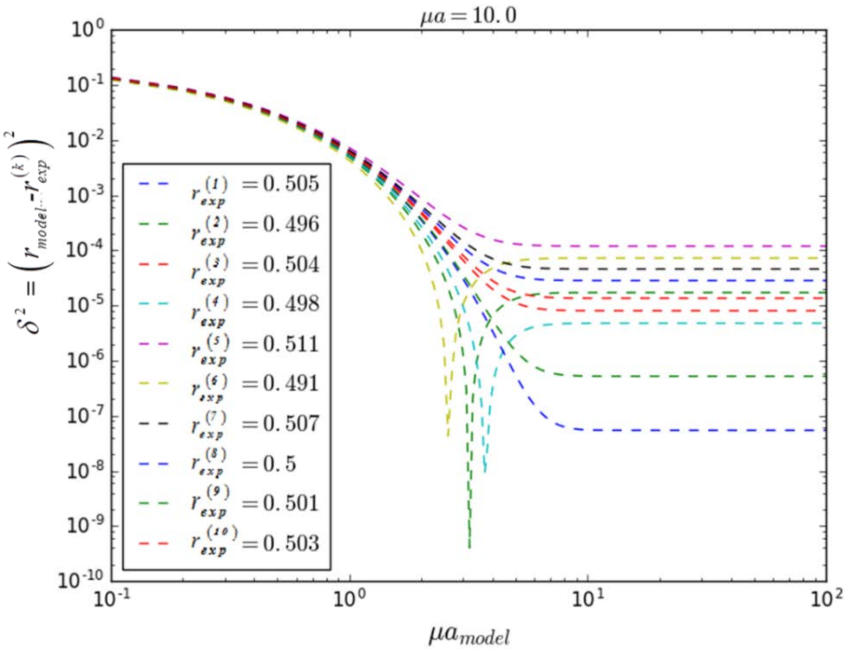

Figure 4.6: Variation of $\delta^2 \triangleq \left(r_{\text{model}} - r_{\text{exp}}^{(k)}\right)^2$ as a function of the model's optical thickness $\left(\mu a_{\text{model}}\right)$ for a slab of actual optical thickness $\mu a = 10.0$, for measurements with a relative standard deviation of 1%.

Figures 4.3 and 4.4 indicate that the minimum of the quantity $\delta^2 \triangleq \left(r_{\text{model}} - r_{\text{exp}}^{(k)}\right)^2$ appears to be uniquely corresponding to the actual value of the slab's thickness, irrespective of the precise value of the measurements. In other words, for slabs that are optically thin, the minimum of the quantity $\delta^2 \triangleq \left(r_{\text{model}} - r_{\text{exp}}^{(k)}\right)^2$ is unique, insensitive to the precision of the respective measurements, and identifies the slab's actual optical thickness correctly and accurately.

A very different situation becomes evident in Figure 4.5 for a slab of optical thickness $\mu a = 3.0$: depending on the value of the respective measurement, the corresponding quantity $\delta^2 \triangleq \left(r_{\text{model}} - r_{\text{exp}}^{(k)}\right)^2$ displays a minimum at various locations within the interval $1.0 < \mu a < 4.0$, or may display no minimum at all. The various minima depicted in Figure 4.5 either under-predict or over-predict, in an apparent random fashion, the actual optical slab thickness of $\mu a = 3.0$. Similar conclusions can be drawn from the results depicted in Figure 4.6, for a (thick) slab of optical thickness $\mu a = 10.0$. The results in Figure 4.6 indicate that, depending on the value of the respective measurement, the corresponding quantity $\delta^2 \triangleq \left(r_{\text{model}} - r_{\text{exp}}^{(k)}\right)^2$ displays a minimum at various locations within the interval $1.0 < \mu a < 4.0$, or may display no minimum at all. In this case, however, there are no over-predictions of the slab's correct thickness: all of the minima under-predict, in an apparent random fashion, the actual optical slab thickness $\mu a = 10.0$.

Figures 4.3 and 4.4 have indicated that for optically thin slabs, the precision of measurements does *not* affect the location of the unique minimum of the quantity $\delta^2 \triangleq \left(r_{\text{model}} - r_{\text{exp}}^{(k)}\right)^2$, and the actual thickness of the respective slab is determined sufficiently accurately (for practical purposes) by the unique location of this minimum. As indicated by the results depicted in Figures 4.5 and 4.6, however, the precision of the measurements decisively affects the results for optically thick slabs. It would be intuitively expected that more precise measurements would yield results "more tightly grouped" around a "better defined" minimum, and hence lead to more accurate predictions of the actual thickness for optically thick slabs. This intuitive expectation is supported by the typical results presented in Figures 4.7 and 4.8 for a thick slab of actual optical thickness $\mu a = 10.0$. The results Figure 4.7 correspond to measurements following a normal distribution with a mean equal to (the exact response) r_{model} and a relative standard deviation of 10%. The results presented in Figure 4.8 are deliberately taken for extremely (unrealistically?) precise measurements assumed to be normally distributed with a mean equal to (the exact response) r_{model} and having a relative standard deviation of 0.001%, to underscore the essential role played by the measurements'' precision.

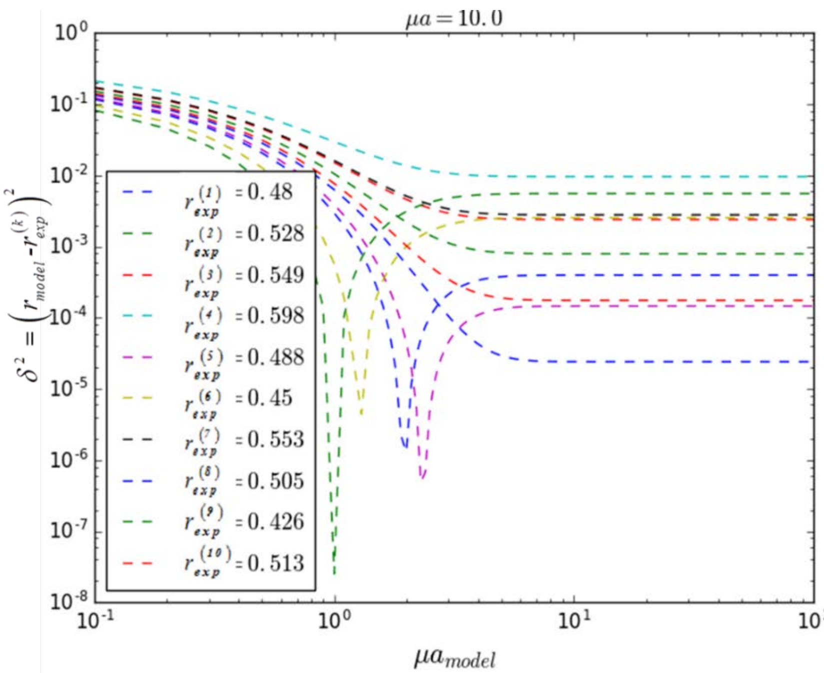

Figure 4.7: Variation of $\delta^2 \triangleq \left(r_{\text{model}} - r_{\text{exp}}^{(k)}\right)^2$ as a function of the model's optical thickness $\left(\mu a_{\text{model}}\right)$ for a slab of actual optical thickness $\mu a = 10.0$, for measurements with a relative standard deviation of 10%.

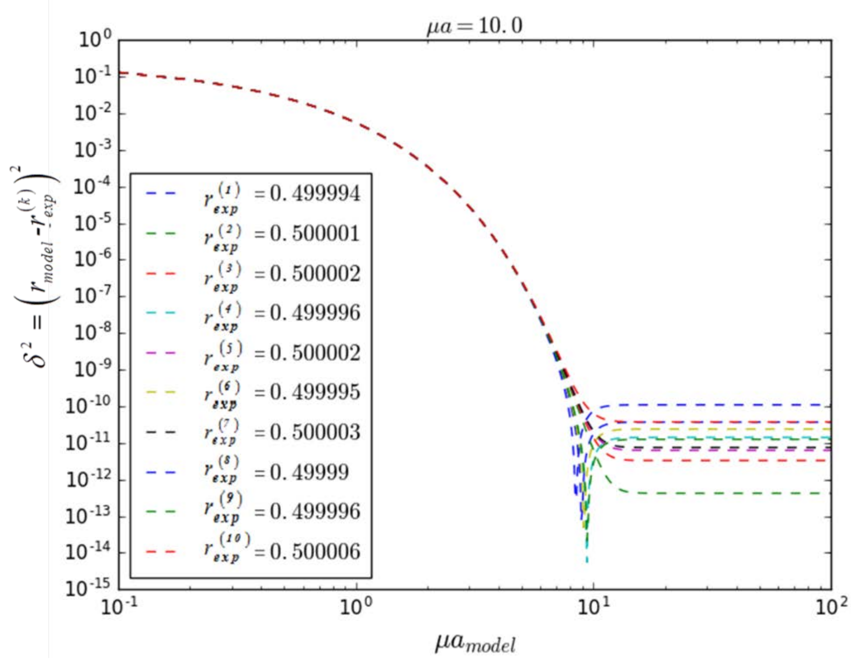

Figure 4.8: Variation of $\delta^2 \triangleq \left(r_{\text{model}} - r_{\text{exp}}^{(k)}\right)^2$ as a function of the model's optical thickness $\left(\mu a_{\text{model}}\right)$ for a slab of actual optical thickness $\mu a = 10.0$, for extremely precise measurements with a relative standard deviation of 0.001%.

Comparing the results depicted in Figure 4.7 with those depicted in Figure 4.6 shows that the quantity $\delta^2 \triangleq \left(r_{\text{model}} - r_{\exp}^{(k)}\right)^2$ corresponding to the less precise measurements (relative standard deviation of 10% for Figure 4.6) displays either no minima, or minima that depend sensitively on the individual measurements, just as displayed by the results for the more precise measurements (relative standard deviation of 1%) presented in Figure 4.7. Furthermore, the minima displayed by the less precise measurements (in Figure 4.7) fall within the interval $1.0 < \mu a < 3.0$, thus being even less indicative of the correct slab thickness than the indication provided by the more precise measurements (in Figure 4.6). This conclusion is further strengthened by comparing all of the results presented in Figures 4.6, 4.7 and 4.8 for a slab of optical thickness $\mu a = 10.0$, namely that the quantity $\delta^2 \triangleq \left(r_{\text{model}} - r_{\exp}^{(k)}\right)^2$ may display no minimum for some measurements, and when it does display a minimum, they respective minimum depends sensitively on the respective measurements. Furthermore, the more accurate the measurements (i.e., the smaller the respective standard deviations), the tighter together grouped are the measurement values; hence, the minima of the squared-differences δ^2 corresponding to the respective measurements are "grouped" more tightly together, and the respective "group of minima" is closer to the correct slab thickness.

Since, as shown in Figures 4.5 through 4.8, some of the summands in Eq.(4.13) may not admit any real-valued minimum while those summands that do have minima which do not coincide with one another, it is not surprising that a numerical algorithm for minimizing the χ^2-functional may yield some minimum value that has no physical meaning, in that the actual physical slab thickness would differ from the value $(\mu a)_{\min}$. On the other hand, in the absence of counting uncertainties, the detector's response yields a unique slab thickness, as demonstrated in Section 4.2. If the measurements are inaccurate, then any minimization of the expression in Eq.(4.13) will lead to erroneous *physical* results, in that the result delivered by any minimization procedure will not be physically correct. Furthermore, for equally precise measurements, the larger the optical thickness of the slab, the more unphysical will likely be the result of the minimization procedure. Altogether, therefore, the results presented in this Section indicate that the reason for the failure of the current state-of-the-art methods to predict accurately the actual thickness of optically thicker slabs stems not from the numerical method used to minimize the χ^2- functional, but stems from the very *formulation* of the χ^2-functional, which makes this functional to be extremely sensitive to the random value of each measurement. In the next Section, it will be shown that Cacuci's PM-CMPS methodology (2014), which incorporates considerably more features of the model than the methods based on minimizing a user-defined χ^2-functional, alleviates the shortcomings of the latter methods while yielding results that are physically accurate up to machine precision.

4.4 Applying the PM-CMPS Methodology for the Inverse Determination of Slab Thickness in the Presence of Counting Uncertainties

For the paradigm system consisting of the slab and detector considered in this Section, "Model B" reduces to a point (i.e., the point detector). Consequently, the PM-CMPS methodology reduces to the inverse predictive modeling of a single multi-physics model ("Model A," involving N_α model parameters α_n and N_r experimentally measured responses r_i), which is governed by Eqs.(2.91) through (2.95). For easy reference, those equations a reproduced below:

$$\boldsymbol{\alpha}^{pred} = \boldsymbol{\alpha}^0 - \left(\mathbf{C}_{\alpha\alpha}\mathbf{S}_{r\alpha}^{\dagger} - \mathbf{C}_{\alpha r}\right)\left[\mathbf{D}_{rr}\right]^{-1}\mathbf{r}^d\left(\boldsymbol{\alpha}^0\right), \tag{4.16}$$

$$\mathbf{r}^{pred} = \mathbf{r}^m - \left(\mathbf{C}_{\alpha r}^{\dagger}\mathbf{S}_{r\alpha}^{\dagger} - \mathbf{C}_{rr}\right)\left[\mathbf{D}_{rr}\right]^{-1}\mathbf{r}^d\left(\boldsymbol{\alpha}^0\right), \tag{4.17}$$

$$\mathbf{C}_{\alpha\alpha}^{pred} = \mathbf{C}_{\alpha\alpha} - \left(\mathbf{C}_{\alpha\alpha}\mathbf{S}_{r\alpha}^{\dagger} - \mathbf{C}_{\alpha r}\right)\left[\mathbf{D}_{rr}\right]^{-1}\left(\mathbf{C}_{\alpha\alpha}\mathbf{S}_{r\alpha}^{\dagger} - \mathbf{C}_{\alpha r}\right)^{\dagger}, \tag{4.18}$$

$$\mathbf{C}_{rr}^{pred} = \mathbf{C}_{rr} - \left(\mathbf{C}_{\alpha r}^{\dagger}\mathbf{S}_{r\alpha}^{\dagger} - \mathbf{C}_{rr}\right)\left[\mathbf{D}_{rr}\right]^{-1}\left(\mathbf{C}_{\alpha r}^{\dagger}\mathbf{S}_{r\alpha}^{\dagger} - \mathbf{C}_{rr}\right)^{\dagger}, \tag{4.19}$$

$$\mathbf{C}_{\alpha r}^{pred} = \mathbf{C}_{\alpha r} - \left(\mathbf{C}_{\alpha\alpha}\mathbf{S}_{r\alpha}^{\dagger} - \mathbf{C}_{\alpha r}\right)\left[\mathbf{D}_{rr}\right]^{-1}\left(\mathbf{C}_{\alpha r}^{\dagger}\mathbf{S}_{r\alpha}^{\dagger} - \mathbf{C}_{rr}\right)^{\dagger}. \tag{4.20}$$

where

$$\mathbf{D}_{rr} \triangleq \mathbf{C}_{rc} - \mathbf{S}_{r\alpha}\mathbf{C}_{\alpha r} - \mathbf{C}_{\alpha r}^{\dagger}\mathbf{S}_{r\alpha}^{\dagger} + \mathbf{C}_{rr}, \tag{4.21}$$

and where the "computed response covariance matrix", $\mathbf{C}_{rc}$, is defined as

$$\mathbf{C}_{rc} \triangleq \mathbf{S}_{r\alpha}\mathbf{C}_{\alpha\alpha}\mathbf{S}_{r\alpha}^{\dagger} . \tag{4.22}$$

The validation metric (or "consistency indicator") takes on the following expression

$$V \triangleq \chi^2 = \left(\mathbf{r}^c - \mathbf{r}^m\right)^{\dagger} \mathbf{D}_{rr}^{-1} \left(\mathbf{r}^c - \mathbf{r}^m\right). \tag{4.23}$$

When Eqs. (4.16) through (4.22) are employed for forward predictive modeling, all of the quantities on the right sides of these equations are known, and the best-estimate predicted quantities are those on the left-side of the respective equations. Note that the detector measures, albeit statistically, the exact response, which implicitly comprises the information about the exact optical thickness of the medium under investigation. Each measured response represents a "point" or "element" sampled from the counting statistical distribution characterizing the detected particles (photons). For simplicity and without loss of generality, the counting statistics are considered to be Gaussian, so that each measured detector response, $r_m^{(k)}$, has the value $r_m^{(k)} = random\ normal\left[r^{exact},\ sd\left(r_m^{(exact)}\right)\right],\ \ k = 1,...,K_n$, where K_n denotes the total number of experiments performed in the "batch *n*". On the other hand, when Eqs. (4.16) through (4.22) are employed for inverse predictive modeling, the set of parameters $\boldsymbol{\alpha}^0$ are unknown, and the first set of measurements is used to estimate these parameter values. Subsequent measurements are assimilated to improve the predictions of both the response and parameter values, until the predicted response and/or parameter values satisfy some a priori imposed accuracy criteria. The detailed inverse predictive algorithm is as follows:

1. Perform the *initial set* of measurements, $r_{\exp}^{(k)}$, by drawing random results from the normal distribution $r_m^{(k)} = random\ normal\left[r^{exact},\ \beta r^{exact}\right],\ \ k = 1,...,K_0$.
2. Compute the initial "sample average": $r_{m,ave}^{(K_0)} = \dfrac{1}{K_0}\sum_{k=1}^{K_0} r_m^{(k)}$.
3. Compute the initial "measurement variance": $C_{rr}^{(K_0)} = \dfrac{1}{K_0 - 1}\sum_{k=1}^{K_0}\left[r_m^{(k)} - r_{m,ave}^{(K_0)}\right]^2$.
4. Compute the initial "sample standard deviation" $SD_m^{(K_0)} = \sqrt{C_{rr}^{(K_0)}}$.

5. Compute the initial estimated parameter value $\alpha^{(1)}$ by using the model, i.e., by solving the nonlinear equation $E_2\left[\alpha^{(1)}\right]=1-\frac{2\mu r_{m,ave}^{(K_0)}}{Q\Sigma_d}$.
6. Compute the initial sensitivities of the response to the uncertain (unknown) model parameter. In general, this computation is performed by using the *adjoint sensitivity analysis methodology*. For the paradigm problem under consideration, the only uncertain model parameter is the medium's optical thickness, so the detector response's sensitivity is readily obtained as: $S^{(1)}=\frac{Q\Sigma_d}{2\mu}E_1\left[\alpha^{(1)}\right]$.
7. Define the "initial parameter standard deviation": $sd\left(\alpha^{(1)}\right)=\gamma\alpha^{(1)}$ and the variance $C_{\alpha\alpha}^{(1)}=\left[\gamma\alpha^{(1)}\right]^2$. The effects of this "initial parameter standard deviation" can be assessed by considering various values for γ. In this study, however, the fixed value $\gamma=10^{-1}$ has been used throughout.
8. Use Eq.(4.22) to compute the initial "computed response covariance": $C_{rc}^{(1)}=S^{(1)\dagger}C_{\alpha}^{(1)}S^{(1)}$.
9. Assuming, in the absence of information to the contrary, that the measured responses are uncorrelated to the model parameters (in this case: the slab's optical thickness), use Eq.(4.21) to compute the following initial value $D_{rr}^{(1)}$.
10. Use Eq.(4.20) to compute the initial "parameter response covariance": $C_{\alpha r}^{(1)}=C_{\alpha\alpha}^{(1)}S^{(1)\dagger}\left[D_{rr}^{(1)}\right]^{-1}C_{rr}^{(K_0)}$.
11. Since the initial parameter value was computed by solving the inverse problem using the "average measurement", set the initial computed response value to be the same as the initial measurement: $r_{comp}^{(1)}=r_{m,ave}^{(K_0)}$.
12. Commence performing experiments to be used for the "inverse predictive modeling" of the slab's optical thickness: perform $n=1,...,N$ *sets* of measurements, $r_m^{(k)}$, $k=1,...,K_n$, , by sampling from the normal distribution $r_m^{(k)}=random\ normal\left[r^{exact},\ sd\left(r_m^{(exact)}\right)\right]$.
13. For each set of experiments, K_n, compute the following quantities:

a. the “sample average”: $r_{m,ave}^{(K_n)} = \frac{1}{K_n}\sum_{k=1}^{K_n} r_m^{(k)}$;

b. the “measurement variance”: $C_{rr}^{(K_n)} = \frac{1}{K_1 - 1}\sum_{k=1}^{K_1}\left[r_m^{(k)} - r_{m,ave}^{(K_n)} \right]^2$;

c. the “sample standard deviation” $SD_m^{(K_n)} = \sqrt{C_{rr}^{(K_n)}}$;

d. the measured response, $r_{meas}^{(n)} \equiv r_{m,ave}^{(K_n)}$, and its covariance $C_{meas}^{(n)} \equiv C_m^{(K_n)}$.

14. Use Eq.(4.17) to compute the new “predicted response” values:

$$r_{pred}^{(n+1)} = r_{meas}^{(n)} + \left[C_{meas}^{(n)} - C_{\alpha r}^{(n)\dagger} S^{(n)\dagger} \right]\left[D_{rr}^{(n)} \right]^{-1}\left[r_{comp}^{(n)} - r_{meas}^{(n)} \right];$$

15. Use Eq.(4.16) to compute the new “predicted parameter” values:

$$\alpha_{pred}^{(n+1)} = \alpha^{(n)} + \left[C_{\alpha r}^{(n)} - C_{\alpha}^{(n)} S^{(n)\dagger} \right]\left[D_{rr}^{(n)} \right]^{-1}\left[r_{comp}^{(n)} - r_{meas}^{(n)} \right];$$

16. Use Eq.(4.18) to compute the new “predicted parameter covariances:

$$C_{\alpha}^{(n+1)} = C_{\alpha\alpha}^{(n)} - \left[C_{\alpha\alpha}^{(n)} S^{(n)\dagger} - C_{\alpha r}^{(n)} \right]\left[D_{rr}^{(n)} \right]^{-1}\left[C_{\alpha\alpha}^{(n)} S^{(n)\dagger} - C_{\alpha r}^{(n)} \right]^{\dagger};$$

17. Use Eq.(4.19) to compute the new “predicted response covariances”:

$$C_{r,pred}^{(n+1)} = C_{meas}^{(n)} - \left[C_{\alpha r}^{(n)\dagger} S^{(n)\dagger} - C_{meas}^{(n)} \right]\left[D_{r}^{(n)} \right]^{-1}\left[C_{\alpha r}^{(n)\dagger} S^{(n)\dagger} - C_{meas}^{(n)} \right]^{\dagger} \text{with } C_{\alpha r}^{(n)} \neq 0$$

18. Use Eq.(4.20) to compute the new “predicted response-parameter covariances”:

$$C_{\alpha r}^{(n+1)} = C_{\alpha r}^{(n)} - \left[C_{\alpha\alpha}^{(n)} S^{(n)\dagger} - C_{\alpha r}^{(n)} \right]\left[D_{rr}^{(n)} \right]^{-1}\left[C_{\alpha r}^{(n)\dagger} S^{(n)\dagger} - C_{meas}^{(n)} \right]^{\dagger}$$

19. Use Eq.(4.23) to compute the predicted “consistency indicator” (or “validation metric”):

$$(CI)^{n+1} = \left[r_{comp}^{(n)} - r_{meas}^{(n)} \right]^{\dagger}\left[D_{rr}^{(n)} \right]^{-1}\left[r_{comp}^{(n)} - r_{meas}^{(n)} \right]$$

20. Optionally: to quantify the possible effects of nonlinearities, perform the new $(n+1)^{th}$ computation with the “calibrated model parameters”:

$$r_{comp}^{(n+1)} = \frac{Q\Sigma_d}{2\mu}\left[1 - E_2\left(\alpha_{pred}^{(n+1)} \right) \right];$$

$$S^{(n+1)} = \frac{Q\Sigma_d}{2\mu} E_1\left(\alpha_{pred}^{(n+1)} \right);$$

$$C_{rc}^{(n+1)} = S^{(n+1)\dagger} C_{\alpha}^{(n+1)} S^{(n+1)};$$

$$\alpha^{(n+1)} \equiv \alpha_{pred}^{(n+1)}$$

Note: the recomputed matrix $C_{rc}^{(n+1)}$ may differ from $C_{r,pred}^{(n+1)}$ because of model nonlinearities; the later matrix is used as the current best-estimate for the covariance matrix of the experimental measurements, to compute the matrix below.

21. Prepare for the next batch of experiments by using computing the quantity

$$D_{rr}^{(n+1)} = C_{rr}^{(n+1)} - S^{(n+1)} C_{\alpha r}^{(n+1)} - C_{\alpha r}^{(n+1)\dagger} S^{(n+1)\dagger} + C_{r,pred}^{(n+1)};$$

22. *Stop when* $\left| \dfrac{r_{comp}^{(n+1)} - r_{pred}^{(n+1)}}{r_{comp}^{(n+1)}} \right| < \varepsilon$. Recall that the experimentally measured detector results reflect the physics of the situations in that the experimental results represent random realizations of a distribution that has the exact response, r_{exact}, as its mean. Thus, the detector results embody (i.e., "know") the exact slab thickness, even though this thickness is unknown to the experimentalist who is attempting to determine it from the model and the experimental results, using the PM-CMPS methodology described in the previous Section. Since the successively predicted responses contain *directly* the effects of all of the measured responses (which reflect the actual physics of the problem) while the successively computed responses contain indirectly the effects of the successively predicted slab thicknesses, the convergence stopping criterion for the PM-CMPS iterations *is imposed on the convergence between the predicted and computed responses*, rather than on the convergence of the computationally predicted slab optical thickness. It is logical to strive towards attaining agreement between computational results and experimental measurements as directly as possible, whenever possible.

For demonstration purposes, the distribution of response measurements is considered to be the normal distribution with mean equal to r_{exact} and with relative standard deviation β, the value of which will be varied to study its influence on the accuracy of the prediction of the unknown optical thickness of the slab under consideration. Simulated experimental results drawn from a normal distribution with a relative standard deviation of 10% $(\beta = 10^{-1})$ will be considered to be "imprecise;" the experimental results drawn from a normal distribution with a relative standard deviation of 0.1% $(\beta = 10^{-3})$ will be considered as being "precise" and the experimental results drawn from a normal distribution with a relative standard deviation of 0.001% $\left(\beta = 10^{-5}\right)$ will be considered as being "very precise."

4.4.1 Prediction of Optically Very Thin Slab (Exact Optical Thickness=0.1)

a) Imprecise measurements $\left(\beta = 10^{-1}\right)$

The exact detector response stemming from a slab of optical thickness $\mu a = 0.1$ is r_{exact} = *1.387275x10*$^{-1}$ *photons/cm*^{2}*sec*, as shown in the last row of Table 4.1. Consider a set $K_1 = 100$ of rather imprecise measurements, characterized by a relative standard deviation $\beta = 10\%$, drawn from a random normal distribution with the mean taken to be the exact response, r_{exact}. The results predicted by the PM-CMPS methodology are: (i) the "predicted response value"; (ii) the "predicted response standard deviation"; (iii) the "predicted slab thickness (parameter)"; and (iv) the "predicted standard deviation of the slab thickness". These results are shown in columns 2 through 5 of Table 4.1. It is seen that the first ($n=1$) set of imprecise measurements predicts the exact response within a standard deviation of *0.01 photons/cm*^{2}*sec*, and the exact optical slab thickness within a standard deviation of $8.89x10^{-3}$. Assimilating the second ($n=2$) set of 100 measurements, which are just as imprecise as the first set, nevertheless improves even further the prediction of the exact response and slab thickness while reducing even further the respective standard deviations. This reduction in the predicted standard deviations accompanying the predicted response and parameter (slab thickness), respectively, is a consequence of the properties of the PM-CMPS methodology.

Table 4.1: Results predicted by PM-CMPS methodology for a slab of exact thickness $\mu a = 0.1$ after successively assimilating 2 batches of 100 imprecise experiments ($\beta = 10^{-1}$)

$\mu a = 0.1$; $\beta = 10^{-1}$; $\varepsilon = 10^{-3}$; $K_n = 100$; Measured response = *Normal* (r_{exact}, $\beta\, r_{exact}$)					
n	Experimental Response Mean Value	Predicted Response	Predicted Response SD	Predicted Parameter	Predicted Parameter SD
1	1.405016x10^{-1}	1.395441x10^{-1}	1.002145x10^{-2}	9.98860x10^{-2}	8.896069x10^{-3}
2	1.401943x10^{-1}	1.389812x10^{-1}	7.129884x10^{-3}	1.00278x10^{-1}	7.818043x10^{-4}
		Exact Response	Exact Response SD	Exact Parameter	
		1.387275x10^{-1}	1.387275x10^{-2}	0.1	

b) Very precise measurements $\left(\beta = 10^{-5}\right)$

Consider a set $K_1 = 100$ of very precise measurements (relative standard deviation $\beta = 10^{-5}$) drawn from the same random normal distribution, i.e., with the distribution's mean taken to be r_{exact} = *1.387275x10^-1 photons/cm^2sec*. Using these very precise measurements, the PM-CMPS methodology predicts the exact response value within a standard deviation of $1.3x10^{-6}$ and the slab thickness within a standard deviation of $2x10^{-6}$, respectively, as shown in Table 4.2. These results clearly indicate the important consequences of precise measurements, which enable the PM-CMPS methodology to produce considerably more precise predictions than when less precise experiments are assimilated.

Table 4.2: Results predicted by PM-CMPS methodology for a slab of exact thickness $\mu a = 0.1$ after assimilating one batch of 100 very precise experiments ($\beta = 10^{-5}$)

$\mu a = 0.1$; $\beta = 10^{-5}$; $\varepsilon = 10^{-8}$; $K_n = 100$; Measured response = *Normal* (r_{exact} , β r_{exact})					
n	Experimental Response Mean Value	Predicted Response	Predicted Response SD	Predicted Parameter	Predicted Parameter SD
1	1.387274×10^{-1}	1.387277×10^{-1}	1.303206×10^{-6}	1.00002×10^{-1}	2.003542×10^{-6}
		Exact Response	Exact Response SD	Exact Parameter	
		1.387275×10^{-1}	1.387275×10^{-2}	0.1	

The results presented in Table 4.2 indicate that a single application of the *PM-CMPS* methodology using very precise measurements predicts the slab thickness within 6 significant digits. The response is also predicted within 6 significant digits. The measurements' precision is the most important factor that affects the accuracy of the prediction of the slab's thickness using the PM-CMPS methodology.

4.4.2 Prediction of Optically Thin Slab (Exact Optical Thickness =1.0)

a) Measurements with 10% relative standard deviation $\left(\beta = 10^{-1}\right)$

Consider a set $K_1 = 100$ of rather imprecise measurements (relative standard deviation $\beta = 10^{-1}$) drawn from the random normal distribution with the mean taken to be the exact response (r_{exact} = *4.257522x10^-1 photons/cm^2sec*). The results predicted by the PM-CMPS methodology are presented in columns 2 through 5 of Table 4.3. It is seen that the first (*n=1*) set of imprecise measurements predicts the exact response within a standard deviation of *2.85x10⁻²*, and the exact optical slab thickness is predicted within a standard deviation of *9.65x10⁻²*. As expected from the properties of the PM-CMPS methodology, the assimilation of the second (*n=2*) set of 100 measurements further improves the prediction of the exact response and slab thickness and reduces further the respective standard deviations, even though the second set of experiments is just as imprecise as the first set.

Table 4.3: Results predicted by PM-CMPS methodology for a slab of exact thickness $\mu a = 1$ after assimilating two batches of 100 experiments with $\beta = 10^{-1}$.

$\mu a = 1$; $\beta = 10^{-1}$; $\varepsilon = 10^{-3}$; $K_n = 100$; Measured response = *Normal* (r_{exact} , β r_{exact})					
n	Experimental Response Mean Value	Predicted Response	Predicted Response SD	Predicted Parameter	Predicted Parameter SD
1	4.311969×10^{-1}	4.276684×10^{-1}	2.854158×10^{-2}	9.867036×10^{-1}	9.655633×10^{-2}
2	4.302537×10^{-1}	4.245931×10^{-1}	1.054078×10^{-2}	9.895185×10^{-1}	9.397102×10^{-2}
		Exact Response	Exact Response SD	Exact Parameter	
		4.257522×10^{-1}	4.257522×10^{-2}	1.00	

b) *Measurements with 0.001% relative standard deviation* $\left(\beta = 10^{-5}\right)$

Consider a set $K_1 = 100$ of precise measurements (relative standard deviation $\beta = 10^{-5}$) drawn from the same random normal distribution, with r_{exact} = *4.257522x10⁻¹ photons/cm²sec* as the distribution's mean. As shown in Table 4.4, using these precise measurements, the PM-CMPS methodology predicts the response within 7 significant digits. These results indicate, as before, the important consequences of precise measurements, which enable the PM-CMPS to produce considerably more precise predictions than when less precise experiments are assimilated.

Table 4.4: Results predicted by PM-CMPS methodology for a slab of exact thickness $\mu a = 1$ after assimilating one batch of 100 experiments with $\beta = 10^{-5}$.

$\mu a = 1$; $\beta = 10^{-5}$; $\varepsilon = 10^{-8}$; $K_n = 100$; Measured response = *Normal* (r_{exact} , β r_{exact})					
n	Experimental Response Mean Value	Predicted Response	Predicted Response SD	Predicted Parameter	Predicted Parameter SD
1	4.257528×10^{-1}	4.257528×10^{-1}	3.999515×10^{-6}	1.000005	5.106484×10^{-6}
		Exact Response	Exact Response SD	Exact Parameter	
		4.257522×10^{-1}	4.257522×10^{-6}	1.00	

The results presented in Table 4.4 indicate that a single application of the PM-CMPS methodology using very precise measurements predicts the slab thickness within 6 significant digits. The response is also predicted within 6 significant digits. Once again, the measurements' precision is the most important factor that affects the accuracy of the prediction of the slab's thickness using the PM-CMPS methodology.

4.4.3 Prediction of Optically Thick Slab (Exact Optical Thickness=3.0)

a) Measurements with 10% relative standard deviation $\left(\beta = 10^{-1}\right)$

Consider a set $K_1 = 100$ of rather imprecise measurements (relative standard deviation $\beta = 10^{-1}$) drawn from the random normal distribution with the mean taken to be the exact response (r_{exact} = $4.94679x10^{-1}$ *photons/cm²sec*). The results predicted by the PM-CMPS methodology are shown in columns 2 through 5 of Table 4.5. It is seen that the first (*n=1*) set of imprecise measurements predicts the exact response within a standard deviation of $3.25x10^{-2}$, and the exact optical slab thickness is predicted within a standard deviation of *0.273*. Assimilating the second (*n=2*) set of 100 measurements, which are just as imprecise as the first set, improves only slightly the prediction of the exact response and of the slab thickness.

Table 4.5: Results predicted by PM-CMPS methodology for a slab of exact thickness $\mu a = 3$ after assimilating batches of 100 experiments with $\beta = 10^{-1}$.

$\mu a = 3$; $\beta = 10^{-1}$; $\varepsilon = 10^{-3}$; $K_n = 100$; Measured response = *Normal* (r_{exact} , $\beta\, r_{exact}$)					
n	Experimental Response Mean Value	Predicted Response	Predicted Response SD	Predicted Parameter	Predicted Parameter SD
1	$5.010051x10^{-1}$	$4.967511x10^{-1}$	$3.255519x10^{-2}$	2.739635	$2.736269x10^{-1}$
2	$4.999093x10^{-1}$	$4.926791x10^{-1}$	$2.490237x10^{-3}$	2.741372	$2.733279\ x10^{-1}$
		Exact Response	Exact Response SD	Exact Parameter	
		$4.94679x10^{-1}$	$4.94679x10^{-2}$	3.00	

b) Measurements with 0.001% relative standard deviation $(\beta = 10^{-5})$

Consider a set $K_1 = 100$ of precise measurements (relative standard deviation $\beta = 10^{-5}$) drawn from the same random normal distribution, with r_{exact} = *4.94679x10*$^{-1}$ *photons/cm*^{2}*sec* as the distribution's mean. Using these precise measurements, the PM-CMPS methodology predicts the response within a standard deviation of *4.65x10*$^{-6}$ *photons/cm*^{2}*sec*, and predicts the slab thickness within six significant digits, respectively, as shown in Table 4.6. As before, these results again indicate that precise measurements enable the PM-CMPS to produce considerably more precise predictions than when less precise experiments are assimilated.

Table 4.6: Results predicted by PM-CMPS methodology for a slab of exact thickness $\mu a = 3$ after assimilating batches of 100 experiments with $\beta = 10^{-5}$.

$\mu a = 3$; $\beta = 10^{-5}$; $\varepsilon = 10^{-8}$; $K_n = 100$; Measured response = *Normal* (r_{exact}, $\beta\, r_{exact}$)					
n	Experimental Response Mean Value	Predicted Response	Predicted Response SD	Predicted Parameter	Predicted Parameter SD
1	4.946797x10^{-1}	4.946797x10^{-1}	4.647000x10^{-6}	3.000097	9.973716 x10^{-5}
		Exact Response	Exact Response SD	Exact Parameter	
		4.94679x10^{-1}	4.94679x10^{-6}	3.00	

4.4.4 Prediction of Optically Very Thick Slab (Exact Optical Thickness=7.0)

a) Measurements with 10% relative standard deviation $(\beta = 10^{-1})$

Consider a set $K_1 = 100$ of rather imprecise measurements (relative standard deviation $\beta = 10^{-1}$) drawn from the random normal distribution with the mean taken to be the exact response (r_{exact} = *4.999482x10*$^{-1}$ *photons/cm*^{2}*sec*). The results predicted by the PM-CMPS methodology are shown in columns 2 through 5 of Table 4.7. It is seen that the first (*n=1*) set of imprecise measurements predicts the exact response within a standard deviation of *9.41x10*$^{-4}$, but the exact optical slab thickness is severely under-predicted. Assimilating the second (*n=2*) set of 100 measurements, which are just as imprecise as the first set, improves significantly the prediction of the exact response, but improves just marginally the prediction of the slab thickness. Additional imprecise experiments would not improve significantly the prediction of the slab thickness.

Table 4.7: Results predicted by PM-CMPS methodology for a slab of exact thickness $\mu a = 7$ after assimilating batches of 100 experiments with $\beta = 10^{-1}$.

$\mu a = 7$; $\beta = 10^{-1}$; $\varepsilon = 10^{-3}$; $K_n = 100$; Measured response = *Normal* (r_{exact} , β r_{exact})				
n	Experimental Response Mean Value	Predicted Response	Predicted Response SD	Predicted Parameter
1	5.063417×10^{-1}	5.020369×10^{-1}	3.288029×10^{-2}	3.770365
2	5.052342×10^{-1}	4.979026×10^{-1}	9.418037×10^{-4}	3.771262
		Exact Response	Exact Response SD	Exact Parameter
		4.999482×10^{-1}	4.999482×10^{-6}	7.00

b) Measurements with 0.001% relative standard deviation $\left(\beta = 10^{-5}\right)$

Consider a set $K_1 = 100$ of precise measurements (relative standard deviation $\beta = 10^{-5}$) drawn from the same random normal distribution, with r_{exact} = *4.999482x10*$^{-1}$ *photons/cm*^{2}*sec photons/cm*^{2}*se* as the distribution's mean. It is seen from the results presented in Table 4.8 that the first (*n=1*) set of precise measurements predicts the exact response within a standard deviation of *4.66x10*$^{-6}$. In addition, the PM-CMPS methodology predicts the slab's thickness within a standard deviation of *0.112*. The second (*n=2*) set of precise measurements further improve the predicted values of both the response and the slab's thickness. As before, these results again indicate that precise measurements enable the PM-CMPS methodology to produce considerably more precise predictions than when less precise experiments are assimilated.

Table 4.8: Results predicted by PM-CMPS methodology for a slab of exact thickness $\mu a = 7$ after assimilating batches of 100 experiments with $\beta = 10^{-5}$.

$\mu a = 7$; $\beta = 10^{-5}$; $\varepsilon = 10^{-8}$; $K_n = 100$; Measured response = *Normal* (r_{exact} , β r_{exact})					
n	Experimental Response Mean Value	Predicted Response	Predicted Response SD	Predicted Parameter	Predicted Parameter SD
1	4.99948×10^{-1}	4.999489×10^{-1}	4.665733×10^{-6}	7.010649	1.11982×10^{-2}
2	4.9994×10^{-1}	4.999488×10^{-1}	4.117474×10^{-6}	7.009803	7.217122×10^{-3}
		Exact Response	Exact Response SD	Exact Parameter	
		4.999482×10^{-1}	4.999482×10^{-6}	7.00	

The results presented in Tables 4.7 and 4.8 for the slab having the exact optical thickness $\mu a = 7$ reinforce the conclusions drawn from Tables 4.5 and 4.6 for the slab having the exact optical thickness $\mu a = 3$, namely that: (i) the *PM-CMPS* methodology under-predicts the slab's actual physical thickness when imprecise experimental results are assimilated, even though the predicted responses agrees within the imposed error criterion with the experimental results; and (ii) the PM-CMPS methodology correctly predicts the slab's actual physical thickness when precise experimental results are assimilated, while also predicting the physically correct response within the selected precision criterion.

4.4.5 Prediction of Extremely Thick Slab (Exact Optical Thickness=10.0)

a) Measurements with 10% relative standard deviation $\left(\beta = 10^{-1}\right)$

Table 4.9 presents results predicted by the PM-CMPS methodology when sets comprising increasingly more experiments, all having relative standard deviations of 10%, are being assimilated. After assimilating a set of $K_n = 5$ experiments, the PM-CMPS methodology predicts the correct value of the response with 2 digits of accuracy, but the slab's thickness is under-predicted by a factor of 5. Increasing the numbers of similarly imprecise measurements from $K_n = 5$ experiments to $K_n = 100$ experiments per set does not appreciably increase the precision of the predicted response, but increases the accuracy of the predicted value of the slab thickness by a factor of about two, although the exact value remains severely under-predicted, due to the relatively large standard deviation ($\beta = 10^{-1}$) considered for the experimental responses.

Table 4.9: Results predicted by PM-CMPS methodology for a slab of exact thickness $\mu a = 10$ after assimilating batches of experiments with $\beta = 10^{-1}$

$\mu a = 10$; $\beta = 10^{-1}$; $\varepsilon = 10^{-3}$; $K_n = 5$;				
n	Experimental Response Mean Value	Predicted Response Value	Predicted Response SD	Predicted ParameterValue
1	4.959541 x10^{-1}	4.920234 x10^{-1}	1.644221 x10^{-2}	2.022075
$\mu a = 10$; $\beta = 10^{-1}$; $\varepsilon = 10^{-3}$; $K_n = 10$;				
1	5.079625 x10^{-1}	4.960152 x10^{-1}	2.033668 x10^{-2}	2.406645
$\mu a = 10$; $\beta = 10^{-1}$; $\varepsilon = 10^{-3}$; $K_n = 50$				
1	4.993500 x10^{-1}	4.977449 x10^{-1}	3.236887 x10^{-2}	3.355578
$\mu a = 10$; $\beta = 10^{-1}$; $\varepsilon = 10^{-3}$; $K_n = 100$;				
1	5.063922x10^{-1}	5.020869x10^{-1}	3.288343x10^{-2}	3.790445
2	5.052846x10^{-1}	4.979521x10^{-1}	9.238754x10^{-4}	3.791330
		Exact Response Value	Exact Response SD	Exact Parameter Value
		4.999981x10^{-1}	4.999981x10^{-2}	10.0

b) Measurements with 1% relative standard deviation $\left(\beta = 10^{-2}\right)$

Table 4.10 presents results predicted by the PM-CMPS methodology when sets comprising increasingly more experiments, all having relative standard deviations of 1%, are being assimilated.

Table 4.10: Results predicted by PM-CMPS methodology for a slab of exact thickness $\mu a = 10$ after assimilating batches of experiments with $\beta = 10^{-2}$.

$\mu a = 10$; $\beta = 10^{-2}$; $\varepsilon = 10^{-5}$; $K_n = 5$;				
n	Experimental Response Mean Value	Predicted Response Value	Predicted Response SD	Predicted ParameterValue
1	4.995937x10^{-1}	4.992142x10^{-1}	1.654840 x10^{-3}	3.910644
$\mu a = 10$; $\beta = 10^{-2}$; $\varepsilon = 10^{-5}$; $K_n = 10$;				
1	5.007945x10^{-1}	4.996136x10^{-1}	2.052874x10^{-3}	4.322908
$\mu a = 10$; $\beta = 10^{-2}$; $\varepsilon = 10^{-5}$; $K_n = 50$				
1	4.999333x10^{-1}	4.997729x10^{-1}	3.238153x10^{-3}	5.315490
$\mu a = 10$; $\beta = 10^{-2}$; $\varepsilon = 10^{-5}$; $K_n = 100$;				
1	5.006375x10^{-1}	5.020869x10^{-1}	3.288731x10^{-3}	5.769938
2	5.005267x10^{-1}	4.997939x10^{-1}	1.352363x10^{-4}	5.771909
		Exact Response Value	Exact Response SD	Exact Parameter Value
		4.999981x10^{-1}	4.999981x10^{-3}	10.0

After assimilating a set of $K_n = 5$ such experiments, the results presented in Table 4.10 indicate that the PM-CMPS methodology predicts the correct value of the response with 3 digits of accuracy, but the slab's thickness is under-predicted by a factor of 2.5. Increasing the numbers of similar measurements from $K_n = 5$ experiments to $K_n = 100$ experiments per set does not increase significantly the precision of the predicted response, but increases the accuracy of the predicted value of the slab thickness, although the exact value remains under-predicted by about 40%, which is the prediction limit for the experimental responses drawn from a normal distribution with a relative standard deviation of 1%.

c) Measurements with 0.1% relative standard deviation $\left(\beta = 10^{-3}\right)$

Table 4.11 presents results predicted by the PM-CMPS methodology when sets comprising increasingly more experiments, all having relative standard deviations of 0.1%, are being assimilated.

Table 4.11: Results predicted by PM-CMPS methodology for a slab of exact thickness $\mu a = 10$ after assimilating batches of experiments with $\beta = 10^{-3}$.

$\mu a = 10$; $\beta = 10^{-3}$; $\varepsilon = 10^{-5}$; $K_n = 5$;				
n	Experimental Response Mean Value	Predicted Response Value	Predicted Response SD	Predicted ParameterValue
1	4.999576x10^{-1}	4.999218x10^{-1}	1.67085 x10^{-4}	5.929063
$\mu a = 10$; $\beta = 10^{-3}$; $\varepsilon = 10^{-5}$; $K_n = 10$;				
1	5.000777 x10^{-1}	4.999618 x10^{-1}	2.082969 x10^{-4}	6.345481
$\mu a = 10$; $\beta = 10^{-3}$; $\varepsilon = 10^{-5}$; $K_n = 50$				
1	4.999916x10^{-1}	4.999756x10^{-1}	3.240331x10^{-4}	7.316728
$\mu a = 10$; $\beta = 10^{-3}$; $\varepsilon = 10^{-5}$; $K_n = 100$;				
1	5.000620x10^{-1}	5.000190 x10^{-1}	3.289465 x10^{-4}	7.756322
2	5.000509x10^{-1}	4.999778 x10^{-1}	1.913978 x10^{-5}	7.760068
		Exact Response Value	Exact Response SD	Exact Parameter Value
		4.999981x10^{-1}	4.999981x10^{-4}	10.0

After assimilating a set of $K_n = 5$ such experiments, the results presented in Table 4.11 indicate that the PM-CMPS methodology predicts the correct value of the response with 4 digits of accuracy, but the slab's thickness is under-predicted by 40%. Increasing the numbers of measurements having the same standard deviation from $K_n = 5$ experiments to $K_n = 100$ experiments per set does not increase significantly the precision of the predicted response, but increases the accuracy of the predicted value of the slab thickness, although the exact value remains under-predicted by about 20%, which is the prediction limit for the experimental responses drawn from a normal distribution with a relative standard deviation of 0.1%.

d) Measurements with 0.01% relative standard deviation $\left(\beta = 10^{-4}\right)$

Table 4.12 presents results predicted by the PM-CMPS methodology when sets comprising increasingly more experiments, all having relative standard deviations of 0.01%, are being assimilated. After assimilating a set of $K_n = 5$ such experiments, the results presented in Table 4.12 indicate that the PM-CMPS methodology predicts the correct value of the response with 5 digits of accuracy, but the slab's thickness is under-predicted by 20%. Increasing the numbers of measurements from $K_n = 5$ experiments to $K_n = 100$ experiments per set increases the accuracy of the predicted value of the slab thickness, although the exact value remains under-predicted by about 7%, which is the prediction limit for the experimental responses drawn from a normal distribution with a relative standard deviation of 0.01%.

Table 4.12: Results predicted by PM-CMPS methodology for a slab of exact thickness $\mu a = 10$ after assimilating batches of experiments with $\beta = 10^{-4}$.

$\mu a = 10$; $\beta = 10^{-4}$; $\varepsilon = 10^{-5}$; $K_n = 5$;				
n	Experimental Response Mean Value	Predicted Response Value	Predicted Response SD	Predicted ParameterValue
1	4.999940x10^{-1}	4.999908x10^{-1}	1.697100x10^{-5}	7.959722
$\mu a = 10$; $\beta = 10^{-4}$; $\varepsilon = 10^{-5}$; $K_n = 10$;				
1	5.000060x10^{-1}	4.999949x10^{-1}	2.146285x10^{-5}	8.334934
$\mu a = 10$; $\beta = 10^{-4}$; $\varepsilon = 10^{-5}$; $K_n = 50$				
1	4.999974x10^{-1}	4.999958x10^{-1}	3.250068x10^{-5}	9.056938
$\mu a = 10$; $\beta = 10^{-4}$; $\varepsilon = 10^{-5}$; $K_n = 100$;				
1	5.000045x10^{-1}	5.000002x10^{-1}	3.294413x10^{-5}	9.338401
		Exact Response Value	Exact Response SD	Exact Parameter Value
		4.999981x10^{-1}	4.999981x10^{-5}	10.0

e) Measurements with 0.001% relative standard deviation $(\beta = 10^{-5})$

Table 4.13 presents results predicted by the PM-CMPS methodology when sets comprising increasingly more experiments, all having relative standard deviations of 0.001%, are being assimilated. After assimilating a set of $K_n = 5$ such experiments, the results presented in Table 4.13 indicate that the PM-CMPS methodology predicts the correct value of the response with 5 digits of accuracy, while the slab's thickness is under-predicted by 5%. Increasing the numbers of measurements from $K_n = 5$ experiments to $K_n = 100$ experiments per set enables the PM-CMPS methodology to predict practically the exact value of the response, and also enables the prediction of the slab thickness within a (negative) difference of 0.02 (2%) of the exact value.

Table 4.13: Results predicted by PM-CMPS methodology for a slab of exact thickness $\mu a = 10$ after assimilating batches of experiments with $\beta = 10^{-5}$.

$\mu a = 10$; $\beta = 10^{-5}$; $\varepsilon = 10^{-8}$; $K_n = 5$;					
n	Experimental Response Mean Value	Predicted Response	Predicted Response SD	Predicted Parameter	Predicted Parameter SD
1	4.999977x10^{-1}	4.999975x10^{-1}	1.796069x10^{-6}	9.563645	6.465910x10^{-1}
$\mu a = 10$; $\beta = 10^{-5}$; $\varepsilon = 10^{-8}$; $K_n = 10$;					
1	4.999989x10^{-1}	4.999981x10^{-1}	2.566375x10^{-6}	9.786590	7.628210x10^{-1}
$\mu a = 10$; $\beta = 10^{-5}$; $\varepsilon = 10^{-8}$; $K_n = 50$;					
1	4.999980x10^{-1}	4.999979x10^{-1}	3.486978x10^{-6}	9.861132	9.236508x10^{-1}
2	4.999986x10^{-1}	4.999981x10^{-1}	1.680819x10^{-6}	9.987424	7.737391x10^{-1}
$\mu a = 10$; $\beta = 10^{-5}$; $\varepsilon = 10^{-8}$; $K_n = 100$;					
1	4.999987x10^{-1}	4.999983x10^{-1}	3.466796x10^{-6}	9.945714	9.359559x10^{-1}
2	4.999986x10^{-1}	4.999981x10^{-1}	1.927486x10^{-6}	9.983171	8.739583x10^{-1}
		Exact Response	Exact Response SD	Exact Parameter	
		4.999981x10^{-1}	4.999981x10^{-6}	10.0	

f) Discussion

The results presented in Tables 4.9 through 4.13 for the slab having the exact optical thickness $\mu a = 10$ reinforce the conclusions previously drawn from the analysis of the slabs of exact optical thickness $\mu a = 3$ and $\mu a = 7$, respectively, namely that: (i) the PM-CMPS methodology under-predicts the slab's actual physical thickness when imprecise experimental results are assimilated, even though the predicted responses agrees within the imposed error criterion with the experimental results; (ii) the PM-CMPS methodology correctly predicts the slab's actual physical thickness when precise experimental results are assimilated, while also predicting the physically correct response within the selected precision criterion.

4.4.6 Prediction Limit for Single-Precision Computations: Slab of Exact Optical Thickness=10.0

For single precision computations, the limits of prediction accuracy when applying the PM-CMPS methodology are illustrated by the results presented in Table 4.14 for a slab of exact optical thickness $\mu a = 15$. Assimilating 169 extremely precise experiments, distributed normally with a relative standard deviation $\beta = 10^{-7}$ around the exact response value, the PM-CMPS methodology predicts the exact response value with 10 significant digits and the exact thickness within 0.2%. This is a remarkable achievement for such a "deep penetration" paradigm problem, in which exponentially fewer gamma rays originating deeply within the slab escape to its surface.

Table 4.14: Prediction limit for single-precision computations using the PM-CMPS methodology

$\mu a = 15$; $\beta = 10^{-7}$; $\varepsilon = 10^{-9}$; $K_n = 169$;					
n	Experimental Response Mean Value	Predicted Response	Predicted Response SD	Predicted Parameter Value
1	5.000000×10^{-1}	5.000000×10^{-1}	3.498662×10^{-8}	15.41315
		Exact Response	Exact Response SD	Exact Parameter
		$4.999999909 \times 10^{-1}$	$4.999999909 \times 10^{-8}$	15.0

The results in this Section indicate that for optically thin slabs, both the traditional chi-square-minimization method and the PM-CMPS methodology predict the slab's thickness accurately. For optically thick slabs, the results obtained in this work have led to following conclusions: (i) the traditional inverse-problem methods based on the minimization of chi-square-type functionals fail to predict the slab's thickness; (ii) the PM-CMPS methodology under-predicts the slab's actual physical thickness when imprecise experimental results are assimilated, even though the predicted responses agrees within the imposed error criterion with the experimental results; (iii) the PM-CMPS methodology correctly predicts the slab's actual physical thickness when precise experimental results are assimilated, while also predicting the physically correct response within the selected precision criterion. For single precision computations, the limits of prediction accuracy when applying the PM-CMPS methodology were illustrated by assimilating 169 extremely precise experiments, distributed normally with a relative standard deviation $\beta = 10^{-7}$ around the exact response value, and showing that the PM-CMPS methodology predicts the exact response value with 10 significant digits and the exact thickness within 0.2%, --a remarkable achievement for such a “deep penetration” paradigm problem. Most of the results obtained in this work correspond to realistic measured standard deviations, obtainable routinely in gamma-ray measurements. The “very precise” measurements were used for illustrative purposes, to highlight the fact that the accuracy of the results predicted by using the PM-CMPS methodology in the “inverse predictive” mode is limited by the precision of the measurements, not by the PM-CMPS methodology or by its underlying computational algorithm.

5 PREDICTIVE MODELING APPLICATION TO SAVANNAH RIVER NATIONAL LABORATORY’S F-AREA COOLING TOWERS

Abstract:

This Chapter illustrates the application of the PM-CMPS methodology to the SRNL F-AREA cooling towers model and actually measured data to obtain predicted optimal nominal values for the model responses and parameters, along with reduced predicted standard deviations for the predicted model parameters and responses. The results presented in this chapter demonstrate that the PM-CMPS methodology reduces the predicted standard deviations to values that are smaller than either the computed or the experimentally measured ones, even for responses (e.g., the outlet water flow rate) for which no measurements are available. These improvements stem from the global characteristics of the PM-CMPS methodology, which combines all of the available information simultaneously in phase-space, as opposed to combining it sequentially, as in current data assimilation procedures.

5.1 Introduction

A mechanical draft cooling tower (MDCT) discharges waste heat from an industrial process into the atmosphere. Using a numerical simulation model of the cooling tower together with measurements of outlet air relative humidity, outlet air and water temperatures enables the quantification of the rate of thermal energy dissipation removed from the respective process. In addition to computing the temperature drop of the cooling water as it passes through the tower, a MDCT model that derives heat dissipation rates from thermal imagery needs to convert the remotely measured cooling tower throat or area-weighted temperature to a cooling water inlet temperature. Therefore, a MDCT model comprises two main components, namely: (i) an inner model which computes the amount of cooling undergone by the water as it passes through the tower as a function of inlet cooling water temperature and ambient weather conditions (air temperature and humidity); and (ii) an outer model which uses a remotely measured throat or area-weighted temperature and adjusts the inlet water temperature to match the target temperature of interest. The MDCT model produces an estimate of the rate at which energy is being discharged to the atmosphere by evaporation and sensible heat transfer. The sensible heat transfer is estimated using the computed change in air or water enthalpy as it passes through the MDCT. If the MDCT fans are on, a prescribed mass flow rate of air and water is used. If the MDCT fans are off, an additional mechanical energy equation is iteratively solved to determine the mass flow rate of air. The flow regime in the fill section of a cooling tower, which can be cross-flow or counter-flow, determines the type of the respective cooling tower.

This Section illustrates the application of the PM-CMPS methodology to the MDCT model developed by Aleman and Sebastian (2015) and extended by Cacuci and Fang (2016) for computing the steady-state thermal performance of the F-AREA cooling towers at the Savannah River National Laboratory. The MDCT model is presented in Section 5.2. Using as inputs the temperature and mass flow rate of the incoming water together with the temperature and humidity ratio of the incoming ambient air, this model computes the temperature and mass flow rate of the effluent water, as well as the temperature and water vapor content of the exhaust air. The air mass flow rate is specified when the cooling tower operates in the mechanical draft mode. When the fan is turned-off, the cooling tower operates in the natural draft/wind-aided mode, in which case the air mass flow rate is calculated using the numerical model.

During the period from April, 2004 through August, 2004, a total of 8079 measured benchmark data sets for the F-area cooling towers (fan-on case) were recorded every fifteen minutes at SRNL. These measured quantities provide the basis for choosing the state functions underlying the mathematical modeling of the cooling tower. Section 5.3 presents the results for the sensitivity analysis of responses of interest, using the *cooling tower adjoint sensitivity model* which was developed by applying the general *adjoint sensitivity analysis methodology* (ASAM) *for nonlinear systems*, which was originally developed by Cacuci (1981). The response sensitivities are needed for (i) ranking the parameters in the order of their importance for contributing to response uncertainties; (ii) propagating the uncertainties (variances and covariances) in the model parameters to quantify the uncertainties (variances and covariances) in the model responses; (iii) performing predictive modeling, which includes assimilation of experimental measurements and calibration of model parameters to produce optimally predicted nominal values for both model parameters and responses, with reduced predicted uncertainties. in Section 5.4 presents the results of applying the PM-CMPS methodology to reduce the uncertainties in the predicted results. At the locations where measurements of outlet air relative humidity, outlet air temperature, and outlet water temperature were available, the PM-CMPS methodology is shown to reduce the predicted standard deviations of predicted responses to values that are smaller than either the computed or the experimentally measured responses. Section 5.4 also shows that the PM-CMPS methodology reduces the predicted uncertainties for responses (such as the distributions of the air and water temperatures, and the air humidity inside the fill section of the cooling tower) for which no direct measurements are available.

5.2 Mathematical Model of the Counter-Flow Cooling Tower

The counter-flow cooling tower is schematically presented in Figure 5.1, which indicates that forced air flow enters the tower through the “rain section” above the water basin, flows upward through the fill section and the drift eliminator, and exits at the tower’s top through an exhaust that encloses a fan. Hot water enters above the fill section and is sprayed onto the top of the fill section to create a uniform, downward falling, film flow through the fill’s numerous meandering vertical passages. Film fills are designed to maximize the water free surface area and the residence time inside of the fill section. Heat and mass transfer occurs at the falling film’s free surface between the water film and the upward air flow. The drift eliminator above the spray zone removes entrained water droplets from the upward flowing air. Below the fill section, the water droplets fall into a collection basin, placed at the bottom of the cooling tower. The heat and mass transfer processes occur overwhelmingly in the fill section. Modeling the heat and mass transfer processes between falling water film and rising air in the cooling tower’s fill section is accomplished solving the following balance equations: (A) liquid continuity; (B) liquid energy balance; (C) water vapor continuity; (D) air/water vapor energy balance. The assumptions used in deriving these equations are as follows:

1. the air and/or water temperatures are uniform throughout each stream at any cross section;
2. the cooling tower has uniform cross-sectional area;
3. the heat and mass transfer occur solely in the direction normal to flows;

4. the heat and mass transfer through tower walls to the environment is negligible;
5. the heat transfer from the cooling tower fan and motor assembly to the air is negligible;
6. the air and water vapor mix as ideal gasses;
7. the flow between flat plates is unsaturated through the fill section.

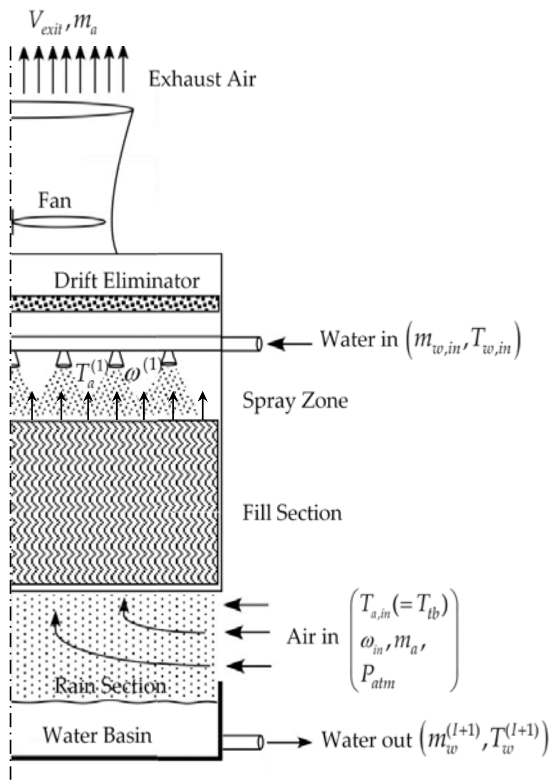

Figure 5.1. Flow through a counter-flow cooling tower.

The fill section is modeled by discretizing it in vertically stacked control volumes as depicted in Figure 5.2. In mechanical draft mode, the mass flow rate of dry air is specified. With the fan off and hot water flowing through the cooling tower, air will continue to flow through the tower due to buoyancy. Wind pressure at the air inlet into the cooling tower will also enhance air flow through the tower. The air flow rate is determined from the overall mechanical energy equation for the dry air flow. The heat and mass transfer between the falling water film and the rising air in a typical control volume of the cooling tower's fill section is presented in Figure 5.3.

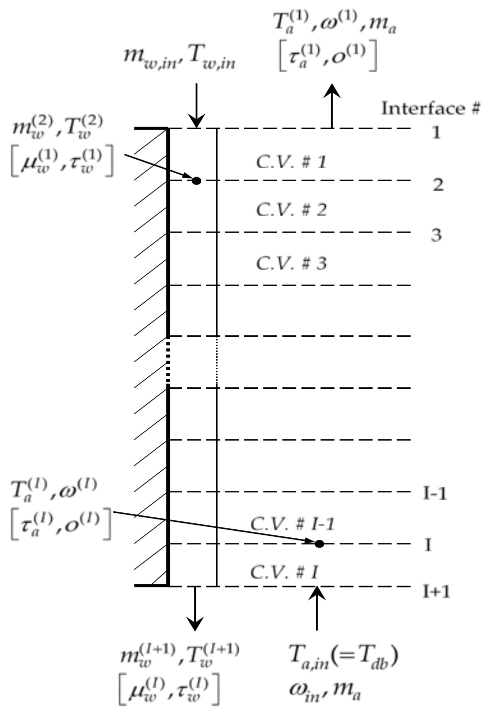

Figure 5.2. Control volumes $(i = 1,.., I)$ comprising the counter-flow cooling tower, together with the symbols denoting the forward state functions $\left(m_w^{(i)}, T_w^{(i)}, T_a^{(i)}, \omega^{(i)},\ i = 1,.., I\right)$ and the adjoint state functions $\left(\mu_w^{(i)}, \tau_w^{(i)}, \tau_a^{(i)}, o^{(i)};\ i = 1,.., I\right)$, respectively.

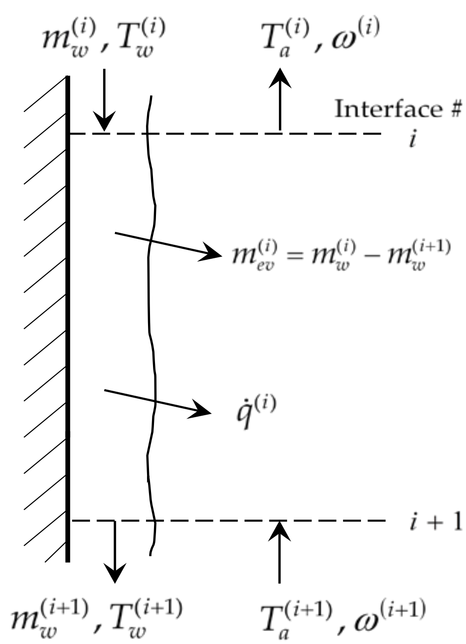

Figure 5.3. Heat and mass transfer between falling water film and rising air in a typical control volume of the cooling tower's fill section.

The state functions underlying the cooling tower model (cf., Figures 5.1 through 5.3) are as follows:

1. the water mass flow rates, denoted as $m_w^{(i)}\ (i = 2,...,50)$, at the exit of each control volume, *i*, along the height of the fill section of the cooling tower;
2. the water temperatures, denoted as $T_w^{(i)}\ (i = 2,...,50)$, at the exit of each control volume, *i*, along the height of the fill section of the cooling tower;
3. the air temperatures, denoted as $T_a^{(i)}\ (i = 1,...,49)$, at the exit of each control volume, *i*, along the height of the fill section of the cooling tower; and
4. the humidity ratios, denoted as $\omega^{(i)}\ (i = 1,...,49)$, at the exit of each control volume, *i*, along the height of the fill section of the cooling tower.

It is convenient to consider the above state functions to be components of the following (column) vectors:

$$\mathbf{m}_w \triangleq \left[m_w^{(2)},...,m_w^{(I+1)}\right]^\dagger, \mathbf{T}_w \triangleq \left[T_w^{(2)},...,T_w^{(I+1)}\right]^\dagger, \mathbf{T}_a \triangleq \left[T_a^{(1)},...,T_a^{(I)}\right]^\dagger, \boldsymbol{\omega} \triangleq \left[\omega^{(1)},...,\omega^{(I)}\right]^\dagger, \qquad (5.1)$$

The governing conservation equations within the total of *I=49* control volumes represented in Figure 5.2 are as follows:

<u>A. Liquid continuity equations:</u>

(i) Control Volume i=1:

$$N_1^{(1)}\left(\mathbf{m}_w, \mathbf{T}_w, \mathbf{T}_a, \boldsymbol{\omega}; \boldsymbol{\alpha}\right) \triangleq$$
$$m_w^{(2)} - m_{w,in} + \frac{M(m_a, \boldsymbol{\alpha})}{\overline{R}}\left[\frac{P_{vs}^{(2)}(T_w^{(2)}, \boldsymbol{\alpha})}{T_w^{(2)}} - \frac{\omega^{(1)} P_{atm}}{T_a^{(1)}(0.622 + \omega^{(1)})}\right] = 0; \qquad (5.2)$$

(ii) Control Volumes i=2,..., I-1:

$$N_1^{(i)}\left(\mathbf{m}_w, \mathbf{T}_w, \mathbf{T}_a, \boldsymbol{\omega}; \boldsymbol{\alpha}\right) \triangleq$$

$$m_w^{(i+1)} - m_w^{(i)} + \frac{M(m_a, \boldsymbol{\alpha})}{\overline{R}}\left[\frac{P_{vs}^{(i+1)}(T_w^{(i+1)}, \boldsymbol{\alpha})}{T_w^{(i+1)}} - \frac{\omega^{(i)} P_{atm}}{T_a^{(i)}(0.622 + \omega^{(i)})}\right] = 0; \tag{5.3}$$

(iii) Control Volume i=I:

$$N_1^{(I)}\left(\mathbf{m}_w, \mathbf{T}_w, \mathbf{T}_a, \boldsymbol{\omega}; \boldsymbol{\alpha}\right) \triangleq$$

$$m_w^{(I+1)} - m_w^{(I)} + \frac{M(m_a, \boldsymbol{\alpha})}{\overline{R}}\left[\frac{P_{vs}^{(I+1)}(T_w^{(I+1)}, \boldsymbol{\alpha})}{T_w^{(I+1)}} - \frac{\omega^{(I)} P_{atm}}{T_a^{(I)}(0.622 + \omega^{(I)})}\right] = 0; \tag{5.4}$$

<u>B. Liquid energy balance equations:</u>

(i) Control Volume i=1:

$$\begin{aligned} N_2^{(1)}\left(\mathbf{m}_w, \mathbf{T}_w, \mathbf{T}_a, \boldsymbol{\omega}; \boldsymbol{\alpha}\right) &\triangleq m_{w,in} h_f(T_{w,in}, \boldsymbol{\alpha}) - (T_w^{(2)} - T_a^{(1)}) H(m_a, \boldsymbol{\alpha}) \\ &- m_w^{(2)} h_f^{(2)}(T_w^{(2)}, \boldsymbol{\alpha}) - (m_{w,in} - m_w^{(2)}) h_{g,w}^{(2)}(T_w^{(2)}, \boldsymbol{\alpha}) = 0; \end{aligned} \tag{5.5}$$

(ii) Control Volumes i=2,..., I-1:

$$\begin{aligned} N_2^{(i)}\left(\mathbf{m}_w, \mathbf{T}_w, \mathbf{T}_a, \boldsymbol{\omega}; \boldsymbol{\alpha}\right) &\triangleq m_w^{(i)} h_f^{(i)}(T_w^{(i)}, \boldsymbol{\alpha}) - (T_w^{(i+1)} - T_a^{(i)}) H(m_a, \boldsymbol{\alpha}) \\ &- m_w^{(i+1)} h_f^{(i+1)}(T_w^{(i+1)}, \boldsymbol{\alpha}) - (m_w^{(i)} - m_w^{(i+1)}) h_{g,w}^{(i+1)}(T_w^{(i+1)}, \boldsymbol{\alpha}) = 0; \end{aligned} \tag{5.6}$$

(iii) Control Volume i=I:

$$\begin{aligned} N_2^{(I)}\left(\mathbf{m}_w, \mathbf{T}_w, \mathbf{T}_a, \boldsymbol{\omega}; \boldsymbol{\alpha}\right) &\triangleq m_w^{(I)} h_f^{(I)}(T_w^{(I)}, \boldsymbol{\alpha}) - (T_w^{(I+1)} - T_a^{(I)}) H(m_a, \boldsymbol{\alpha}) \\ &- m_w^{(I+1)} h_f^{(I+1)}(T_w^{(I+1)}, \boldsymbol{\alpha}) - (m_w^{(I)} - m_w^{(I+1)}) h_{g,w}^{(I+1)}(T_w^{(I+1)}, \boldsymbol{\alpha}) = 0; \end{aligned} \tag{5.7}$$

<u>C. Water vapor continuity equations:</u>

(i) Control Volume i=1:

$$N_3^{(1)}\left(\mathbf{m}_w, \mathbf{T}_w, \mathbf{T}_a, \boldsymbol{\omega}; \boldsymbol{\alpha}\right) \triangleq \omega^{(2)} - \omega^{(1)} + \frac{m_{w.in} - m_w^{(2)}}{|m_a|} = 0; \tag{5.8}$$

(ii) Control Volumes i=2,..., I-1:

$$N_3^{(i)}\left(\mathbf{m}_w,\mathbf{T}_w,\mathbf{T}_a,\boldsymbol{\omega};\boldsymbol{\alpha}\right)\triangleq\omega^{(i+1)}-\omega^{(i)}+\frac{m_w^{(i)}-m_w^{(i+1)}}{|m_a|}=0; \qquad (5.9)$$

(iii) Control Volume i=I:

$$N_3^{(I)}\left(\mathbf{m}_w,\mathbf{T}_w,\mathbf{T}_a,\boldsymbol{\omega};\boldsymbol{\alpha}\right)\triangleq\omega_{in}-\omega^{(I)}+\frac{m_w^{(I)}-m_w^{(I+1)}}{|m_a|}=0; \qquad (5.10)$$

<u>D. The air/water vapor energy balance equations:</u>

(i) Control Volume i=1:

$$\begin{aligned}N_4^{(1)}\left(\mathbf{m}_w,\mathbf{T}_w,\mathbf{T}_a,\boldsymbol{\omega};\boldsymbol{\alpha}\right)&\triangleq(T_a^{(2)}-T_a^{(1)})C_p^{(1)}(\frac{T_a^{(1)}+273.15}{2},\boldsymbol{\alpha})-\omega^{(1)}h_{g,a}^{(1)}(T_a^{(1)},\boldsymbol{\alpha})\\&+\frac{(T_w^{(2)}-T_a^{(1)})H(m_a,\boldsymbol{\alpha})}{|m_a|}+\frac{(m_{w,in}-m_w^{(2)})h_{g,w}^{(2)}(T_w^{(2)},\boldsymbol{\alpha})}{|m_a|}+\omega^{(2)}h_{g,a}^{(2)}(T_a^{(2)},\boldsymbol{\alpha})=0;\end{aligned} \qquad (5.11)$$

(ii) Control Volumes i=2,..., I-1:

$$\begin{aligned}N_4^{(i)}\left(\mathbf{m}_w,\mathbf{T}_w,\mathbf{T}_a,\boldsymbol{\omega};\boldsymbol{\alpha}\right)&\triangleq(T_a^{(i+1)}-T_a^{(i)})C_p^{(i)}(\frac{T_a^{(i)}+273.15}{2},\boldsymbol{\alpha})\\&-\omega^{(i)}h_{g,a}^{(i)}(T_a^{(i)},\boldsymbol{\alpha})+\frac{(T_w^{(i+1)}-T_a^{(i)})H(m_a,\boldsymbol{\alpha})}{|m_a|}\\&+\frac{(m_w^{(i)}-m_w^{(i+1)})h_{g,w}^{(i+1)}(T_w^{(i+1)},\boldsymbol{\alpha})}{|m_a|}+\omega^{(i+1)}h_{g,a}^{(i+1)}(T_a^{(i+1)},\boldsymbol{\alpha})=0;\end{aligned} \qquad (5.12)$$

(iii) Control Volume i=I:

$$\begin{aligned}N_4^{(I)}\left(\mathbf{m}_w,\mathbf{T}_w,\mathbf{T}_a,\boldsymbol{\omega};\boldsymbol{\alpha}\right)&\triangleq(T_{a,in}-T_a^{(I)})C_p{}^{(I)}(\frac{T_a^{(I)}+273.15}{2},\boldsymbol{\alpha})\\&-\omega^{(I)}h_{g,a}^{(I)}(T_a^{(I)},\boldsymbol{\alpha})+\frac{(T_w^{(I+1)}-T_a^{(I)})H(m_a,\boldsymbol{\alpha})}{|m_a|}+\\&\frac{(m_w^{(I)}-m_w^{(I+1)})h_{g,w}^{(I+1)}(T_w^{(I+1)},\boldsymbol{\alpha})}{|m_a|}+\omega_{in}h_{g,a}(T_{a,in},\boldsymbol{\alpha})=0.\end{aligned} \qquad \underline{(5.13)}$$

The components of the vector $\boldsymbol{\alpha}$, which appears in Eqs. (5.2) through (5.13) comprise the model parameters, which are generically denoted as α_i, i.e.,

$$\boldsymbol{\alpha} \triangleq \left(\alpha_1,...,\alpha_{N_\alpha}\right), \tag{5.14}$$

where N_α denotes the total number of model parameters. These model parameters are experimentally derived quantities, and their complete distributions parameters are not known; however, we have determined the first four moments (means, variance/covariance, skewness, and kurtosis) of each of these parameter distributions, as detailed in Section 5.4. Equations (5.2) through (5.13) are solved by Newton's method together with the GMRES linear iterative solver for sparse matrices (Saad, Y. and Schultz, M.H. 1986) provided in the NSPCG package (Oppe et al, 1988). This GMRES solver approximates the exact solution-vector of a linear system by using the Arnoldi iteration to find the approximate solution-vector by minimizing the norm of the residual vector over a Krylov subspace. The specific computational steps are as follows:

(a) Write Eqs.(5.2) through (5.13) in vector form as

$$\mathbf{N}\left(\mathbf{u}\right) = \mathbf{0}, \tag{5.15}$$

where the following definitions are used:

$$\mathbf{N} \triangleq \left(N_1^{(1)},..,N_1^{(I)},...,N_4^{(1)},...,N_4^{(I)}\right)^\dagger, \ \ \mathbf{u} \triangleq \left(\mathbf{m}_w, \mathbf{T}_w, \mathbf{T}_a, \boldsymbol{\omega}\right)^\dagger; \tag{5.16}$$

(b) Set the initial guess, $\mathbf{u}_0$, to be the inlet boundary conditions;

(c) Start outer iteration loop: Steps *d* through *g*, below, constitute the outer iteration loop; for $n = 0,1,2,...$, iterate over the following steps until convergence:

(d) Start inner iteration loop: for $m = 1,2,...,$ use the iterative GMRES linear solver with the Modified Incomplete Cholesky (MIC) preconditioner, with restarts, to solve, until convergence, the following system to compute the vector $\delta\mathbf{u}$:

$$\mathbf{J}\left(\mathbf{u}_n\right)\delta\mathbf{u} = -\mathbf{N}\left(\mathbf{u}_n\right), \tag{5.17}$$

where n is the current outer loop iteration number, and the Jacobian matrix of derivatives of Eqs. (5.3) through (5.13) with respect to the state functions is following the block-matrix:

$$\mathbf{J}(\mathbf{u}_n) \triangleq \begin{pmatrix} \mathbf{A}_1 & \mathbf{B}_1 & \mathbf{C}_1 & \mathbf{D}_1 \\ \mathbf{A}_2 & \mathbf{B}_2 & \mathbf{C}_2 & \mathbf{D}_2 \\ \mathbf{A}_3 & \mathbf{B}_3 & \mathbf{C}_3 & \mathbf{D}_3 \\ \mathbf{A}_4 & \mathbf{B}_4 & \mathbf{C}_4 & \mathbf{D}_4 \end{pmatrix}. \tag{5.18}$$

The components of the matrices appearing in Eq.(5.18) are defined as follows:

$$a_\ell^{i,j} \triangleq \frac{\partial N_\ell^{(i)}}{\partial m_w^{(j+1)}};\ \ell = 1,2,3,4;\ i = 1,\ldots,I;\ \ j = 1,\ldots,I; \tag{5.19}$$

$$b_\ell^{i,j} \triangleq \frac{\partial N_\ell^{(i)}}{\partial T_w^{(j+1)}};\ \ell = 1,2,3,4;\ i = 1,\ldots,I;\ \ j = 1,\ldots,I; \tag{5.20}$$

$$c_\ell^{i,j} \triangleq \frac{\partial N_\ell^{(i)}}{\partial T_a^{(j)}};\ \ell = 1,2,3,4; i = 1,\ldots,I;\ \ j = 1,\ldots,I; \tag{5.21}$$

$$d_\ell^{i,j} \triangleq \frac{\partial N_\ell^{(i)}}{\partial \omega^{(j)}};\ \ell = 1,2,3,4;\ i = 1,\ldots,I;\ \ j = 1,\ldots,I; \tag{5.22}$$

Computing the derivatives of the "liquid continuity equations" with respect to $m_w^{(j)}$ yields:

$$\mathbf{A}_1 \triangleq \left(a_1^{i,j}\right)_{I\times I} = \begin{pmatrix} 1 & 0 & . & 0 & 0 \\ -1 & 1 & . & 0 & 0 \\ . & . & . & . & . \\ 0 & 0 & . & 1 & 0 \\ 0 & 0 & . & -1 & 1 \end{pmatrix}. \tag{5.23}$$

Computing the derivatives of the "liquid continuity equations" with respect to $T_w^{(j)}$ yields:

$$\mathbf{B}_1 \triangleq \left(b_1^{i,j}\right)_{I\times I} = \begin{pmatrix} b_1^{1,1} & 0 & . & 0 & 0 \\ 0 & b_1^{2,2} & . & 0 & 0 \\ . & . & . & . & . \\ 0 & 0 & . & b_1^{I-1,I-1} & 0 \\ 0 & 0 & . & 0 & b_1^{I,I} \end{pmatrix}, \tag{5.24}$$

where

$$b_1^{i,i} \triangleq -\frac{M(m_a,\boldsymbol{\alpha})}{\overline{R}} \frac{P_{vs}^{(i+1)}(T_w^{(i+1)},\boldsymbol{\alpha})}{[T_w^{(i+1)}]^2} \left\{ \frac{a_1}{T_w^{(i+1)}} + 1 \right\}. \tag{5.25}$$

Computing the derivatives of the "liquid continuity equations" with respect to $T_a^{(j)}$ yields:

$$\mathbf{C}_1 \triangleq \left(c_1^{i,j}\right)_{I\times I} = \begin{pmatrix} c_1^{1,1} & 0 & . & 0 & 0 \\ 0 & c_1^{2,2} & . & 0 & 0 \\ . & . & . & . & . \\ 0 & 0 & . & c_1^{I-1,I-1} & 0 \\ 0 & 0 & . & 0 & c_1^{I,I} \end{pmatrix}, \tag{5.26}$$

where

$$c_1^{i,i} \triangleq \frac{M(m_a,\boldsymbol{\alpha})}{\overline{R}} \frac{\omega^{(i)} P_{atm}}{\left[T_a^{(i)}\right]^2 \left(0.622 + \omega^{(i)}\right)}. \tag{5.27}$$

Computing the derivatives of the "liquid continuity equations" with respect to $\omega^{(j)}$ yields:

$$\mathbf{D}_1 \triangleq \left(d_1^{i,j}\right)_{I\times I} = \begin{pmatrix} d_1^{1,1} & 0 & . & 0 & 0 \\ 0 & d_1^{2,2} & . & 0 & 0 \\ . & . & . & . & . \\ 0 & 0 & . & d_1^{I-1,I-1} & 0 \\ 0 & 0 & . & 0 & d_1^{I,I} \end{pmatrix}, \tag{5.28}$$

Where

$$d_1^{i,i} = \frac{M(m_a, \boldsymbol{\alpha})}{\overline{R}} \frac{P_{atm}}{\left[0.622 + \omega^{(i)}\right] T_a^{(i)}} \left\{ \frac{\omega^{(i)}}{\left[0.622 + \omega^{(i)}\right]} - 1 \right\}. \tag{5.29}$$

Computing the derivatives of the liquid energy balance equations with respect to $m_w^{(j)}$ yields:

$$\mathbf{A}_2 \triangleq \left(a_2^{i,j}\right)_{I \times I} = \begin{pmatrix} a_2^{1,1} & 0 & . & 0 & 0 \\ a_2^{2,1} & a_2^{2,2} & . & 0 & 0 \\ . & . & . & . & . \\ 0 & 0 & . & a_2^{I-1,I-1} & 0 \\ 0 & 0 & . & a_2^{I,I-1} & a_2^{I,I} \end{pmatrix}, \tag{5.30}$$

Where

$$a_2^{i,i-1} \triangleq h_f^{(i)}(T_w^{(i)}, \boldsymbol{\alpha}) - h_g^{(i+1)}(T_w^{(i+1)}, \boldsymbol{\alpha}), \quad i = 2, \ldots, I; \;\; j = i-1; \tag{5.31}$$

$$a_2^{i,i} \triangleq h_g^{(i+1)}(T_w^{(i+1)}, \boldsymbol{\alpha}) - h_f^{(i+1)}(T_w^{(i+1)}, \boldsymbol{\alpha}), \quad i = 1, \ldots, I; \;\; j = i. \tag{5.32}$$

Computing the derivatives of the liquid energy balance equations with respect to $T_w^{(j)}$ yields:

$$\mathbf{B}_2 \triangleq \left(b_2^{i,j}\right)_{I \times I} = \begin{pmatrix} b_2^{1,1} & 0 & . & 0 & 0 \\ b_2^{2,1} & b_2^{2,2} & . & 0 & 0 \\ . & . & . & . & . \\ 0 & 0 & . & b_2^{I-1,I-1} & 0 \\ 0 & 0 & . & b_2^{I,I-1} & b_2^{I,I} \end{pmatrix}, \tag{5.33}$$

Where

$$b_2^{i,i-1} \triangleq m_w^{(i)} \frac{\partial h_f^{(i)}}{\partial T_w^{(i)}}; \;\; i = 2, \ldots, I; \;\; j = i-1; \tag{5.34}$$

$$b_2^{i,i} \triangleq -m_w^{(i+1)} \frac{\partial h_f^{(i+1)}}{\partial T_w^{(i+1)}} - \left(m_w^{(i)} - m_w^{(i+1)}\right) \frac{\partial h_{g,w}^{(i+1)}}{\partial T_w^{(i+1)}} - H(m_a, \alpha); \quad i = 1, \ldots, I; \;\; j = i. \tag{5.35}$$

Computing the derivatives of the liquid energy balance equations with respect to $T_a^{(j)}$ yields:

$$\mathbf{C}_2 \triangleq \left(c_2^{i,j}\right)_{I\times I} = \begin{pmatrix} c_2^{1,1} & 0 & . & 0 & 0 \\ 0 & c_2^{2,2} & . & 0 & 0 \\ . & . & . & . & . \\ 0 & 0 & . & c_2^{I-1,I-1} & 0 \\ 0 & 0 & . & 0 & c_2^{I,I} \end{pmatrix}, \tag{5.36}$$

where

$$c_2^{i,i} \triangleq H(m_a, \boldsymbol{\alpha}); \quad i = 1,\ldots,I; \;\; j = i. \tag{5.37}$$

Computing the derivatives of the liquid energy balance equations with respect to $\omega^{(j)}$ yields:

$$\mathbf{D}_2 \triangleq \left[d_2^{i,j}\right]_{I\times I} = \mathbf{0}. \tag{5.38}$$

Computing the derivatives of the water vapor continuity equations with respect to $m_w^{(j)}$ yields:

$$\mathbf{A}_3 \triangleq \left(a_3^{i,j}\right)_{I\times I} = \frac{1}{m_a}\begin{pmatrix} -1 & 0 & . & 0 & 0 \\ 1 & -1 & . & 0 & 0 \\ . & . & . & . & . \\ 0 & 0 & . & -1 & 0 \\ 0 & 0 & . & 1 & -1 \end{pmatrix}, \tag{5.39}$$

Computing the derivatives of the water vapor continuity equations with respect to $T_w^{(j)}$ yields:

$$\mathbf{B}_3 \triangleq \left[b_3^{i,j}\right]_{I\times I} = \mathbf{0}. \tag{5.40}$$

Computing the derivatives of the water vapor continuity equations with respect to $T_a^{(j)}$ yields:

$$\mathbf{C}_3 \triangleq \left[c_3^{i,j}\right]_{I\times I} = \mathbf{0}. \tag{5.41}$$

Computing the derivatives of the water vapor continuity equations with respect to $\omega^{(j)}$ yields:

$$\mathbf{D}_3 \triangleq \left(d_3^{i,j}\right)_{I\times I} = \begin{pmatrix} -1 & 1 & . & 0 & 0 \\ 0 & -1 & . & 0 & 0 \\ . & . & . & . & . \\ 0 & 0 & . & -1 & 1 \\ 0 & 0 & . & 0 & -1 \end{pmatrix}. \tag{5.42}$$

Computing the derivatives of the air/water vapor energy balance equations with respect to $m_w^{(j)}$ yields:

$$\mathbf{A}_4 \triangleq \left(a_4^{i,j}\right)_{I\times I} = \begin{pmatrix} a_4^{1,1} & 0 & . & 0 & 0 \\ a_4^{2,1} & a_4^{2,2} & . & 0 & 0 \\ . & . & . & . & . \\ 0 & 0 & . & a_4^{I-1,I-1} & 0 \\ 0 & 0 & . & a_4^{I,I-1} & a_4^{I,I} \end{pmatrix}, \tag{5.43}$$

where

$$a_4^{i,i-1} \triangleq \frac{h_{g,w}^{(i+1)}(T_w^{(i+1)},\boldsymbol{\alpha})}{m_a};\; i=2,...,I;\;\; j=i-1; \tag{5.44}$$

$$a_4^{i,i} \triangleq -\frac{h_{g,w}^{(i+1)}(T_w^{(i+1)},\boldsymbol{\alpha})}{m_a};\; i=1,...,I;\;\; j=i. \tag{5.45}$$

Computing the derivatives of the air/water vapor energy balance equations with respect $T_w^{(j)}$ yields:

$$\mathbf{B}_4 \triangleq \left(b_4^{i,j}\right)_{I\times I} = \begin{pmatrix} b_4^{1,1} & 0 & . & 0 & 0 \\ 0 & b_4^{2,2} & . & 0 & 0 \\ . & . & . & . & . \\ 0 & 0 & . & b_4^{I-1,I-1} & 0 \\ 0 & 0 & . & 0 & b_4^{I,I} \end{pmatrix}, \tag{5.46}$$

where

$$b_4^{i,i} \triangleq \frac{1}{m_a}\left[\left(m_w^{(i)} - m_w^{(i+1)}\right)\frac{\partial h_{g,w}^{(i+1)}}{\partial T_w^{(i+1)}} + H(m_a,\boldsymbol{\alpha})\right];\quad i=1,...,I;\;\; j=i.. \tag{5.47}$$

Computing the derivatives of the air/water vapor energy balance equations with respect to $T_a^{(j)}$ yields:

$$\mathbf{C}_4 \triangleq \left(c_4^{i,j}\right)_{I\times I} = \begin{pmatrix} c_4^{1,1} & c_4^{1,2} & . & 0 & 0 \\ 0 & c_4^{2,2} & . & 0 & 0 \\ . & . & . & . & . \\ 0 & 0 & . & c_4^{I-1,I-1} & c_4^{I-1,I} \\ 0 & 0 & . & 0 & c_4^{I,I} \end{pmatrix}, \tag{5.48}$$

where

$$c_4^{i,i} \triangleq \left(T_a^{(i+1)} - T_a^{(i)}\right)\frac{\partial C_p^{(i)}}{\partial T_a^{(i)}} - C_p^{(i)}\left(\frac{T_a^{(i)} + 273.15}{2}, \boldsymbol{\alpha}\right) - \omega^{(i)}\frac{\partial h_{g,a}^{(i)}}{\partial T_a^{(i)}} - \frac{H(m_a, \boldsymbol{\alpha})}{m_a}; \quad i = 1,\dots,I; \quad j = i; \tag{5.49}$$

$$c_4^{i,i+1} \triangleq C_p^{(i)}\left(\frac{T_a^{(i)} + 273.15}{2}, \boldsymbol{\alpha}\right) + \omega^{(i+1)}\frac{\partial h_{g,a}^{(i+1)}}{\partial T_a^{(i+1)}}; \quad i = 1,\dots,I-1; \quad j = i+1. \tag{5.50}$$

Computing the derivatives of the air/water vapor energy balance equations with respect to $\omega^{(j)}$ yields:

$$\mathbf{D}_4 \triangleq \left(d_4^{i,j}\right)_{I\times I} = \begin{pmatrix} d_4^{1,1} & d_4^{1,2} & . & 0 & 0 \\ 0 & d_4^{2,2} & . & 0 & 0 \\ . & . & . & . & . \\ 0 & 0 & . & d_4^{I-1,I-1} & d_4^{I-1,I} \\ 0 & 0 & . & 0 & d_4^{I,I} \end{pmatrix}, \tag{5.51}$$

where

$$d_4^{i,i} \triangleq -h_{g,a}^{(i)}(T_a^{(i)}, \boldsymbol{\alpha}); \quad i = 1,\dots,I; \quad j = i; \tag{5.52}$$

$$d_4^{i,i+1} \triangleq h_{g,a}^{(i+1)}(T_a^{(i+1)}, \boldsymbol{\alpha}); \quad i = 1,\dots,I-1; \quad j = i+1. \tag{5.53}$$

In view of Eqs. (5.19) through (5.53) , the Jacobian represented by Eq. (5.18) is a non-symmetric sparse matrix of order 196 by 196, with 14 nonzero diagonals. The non-symmetric diagonal storage format is used to store the respective 14 nonzero diagonals, so that the "condensed" Jacobian matrix has dimensions 196 by 14. Since the Jacobian is highly non-symmetric, the cost of the iterations of the GMRES solver grows as $O(m^2)$, where m is the iteration number within the

GMRES solver. To reduce this computational cost, the GMRES solver is configured to run with the restart feature. The optimized value for the restart frequency is 10 for this specific application. The MIC preconditioner can speed up the convergence of the GMRES solver using the parameters OMEGA and LVFILL in the modified incomplete factorization methods for the MIC preconditioner; for this application the following values were found to be optimal: OMEGA = 0.000000001 and LVFILL = 1. The Jacobian is not updated inside the sparse GMRES solver. The default convergence of GMRES is tested with the following criterion ,

$$\left[\frac{\left\langle \tilde{\mathbf{z}}^{(m)}, \tilde{\mathbf{z}}^{(m)} \right\rangle}{\left\langle \delta\mathbf{u}^{(m)}, \delta\mathbf{u}^{(m)} \right\rangle}\right]^{\frac{1}{2}} < \zeta \tag{5.54}$$

where $\tilde{\mathbf{z}}^{(m)}$ denotes the pseudo-residual at m^{th}-iteration of the GMRES solver, $\delta\mathbf{u}^{(m)}$ is the solution of Eq. (5.17) at m^{th}-iteration, and ζ denotes the stopping test value for the GMRES solver.

(e) Set the next step:

$$\mathbf{u}_{n+1} = \mathbf{u}_n + \delta\mathbf{u}, \tag{5.55}$$

where n is the current outer loop iteration number, and update the Jacobian.

(f) test for convergence of the outer loop until the error in the solution is less than a specified maximum value. For solving Eqs. (5.2) through (5.13), the following error criterion has been used:

$$error = \max\left(\frac{\left|\delta m_w^{(i)}\right|}{m_w^{(i)}}, \frac{\left|\delta T_w^{(i)}\right|}{T_w^{(i)}}, \frac{\left|\delta T_a^{(i)}\right|}{T_a^{(i)}}, \frac{\left|\delta \omega^{(i)}\right|}{\omega^{(i)}}\right) < 10^{-6} \tag{5.56}$$

(g) Set $n = n+1$, thus closing the outer iteration loop, and go to step (d).

The solution strategy described above in steps (a) through (g), cf. Eqs.(5.15) through (5.56) for solving Eqs. (5.2) through (5.13) converged successfully for all the 8079 benchmark data sets, which will be described in Section 5.4. For each of these benchmark data sets, the outer loop iterations described above (i.e., steps c through g) converge in 4 iterations; for each outer loop iteration, the GMRES solver used for solving Eq. (5.17) converges in 12 iterations. The "zero-to-zero" verification of the solution's accuracy using Eqs. (5.2) through (5.13) gives an error of the order of 10^{-7}.

The responses that correspond to the measurements to be described in Section 5.4, below, are as follows:

(a) the vector $\mathbf{m}_w \triangleq \left[m_w^{(2)},...,m_w^{(I+1)} \right]^\dagger$ of water mass flow rates at the exit of each control volume i, $(i=1,...,49)$;

(b) the vector $\mathbf{T}_w \triangleq \left[T_w^{(2)},...,T_w^{(I+1)} \right]^\dagger$ of water temperatures at the exit of each control volume i, $(i=1,...,49)$;

(c) the vector $\mathbf{T}_a \triangleq \left[T_a^{(1)},...,T_a^{(I)} \right]^\dagger$ of air temperatures at the exit of each control volume i, $(i=1,...,49)$;

(d) the vector $\mathbf{RH} \triangleq \left[RH^{(1)},...,RH^{(I)} \right]^\dagger$, having as components the air relative humidity at the exit of each control volume i, $(i=1,...,49)$.

While the water mass flow rates, the water temperatures, and the air temperatures are obtained directly as the solutions of Eqs.(5.2) through (5.13), the air relative humidity, $RH^{(i)}$, is computed for each control volume using the expression :

$$RH^{(i)} = \frac{P_v\left(\omega^{(i)},\boldsymbol{\alpha}\right)}{P_{vs}\left(T_a^{(i)},\boldsymbol{\alpha}\right)}\times 100 = \frac{\left(\dfrac{\omega^{(i)}P_{atm}}{\omega^{(i)}+0.622}\right)}{\left(e^{a_0+\frac{a_1}{T_a^{(i)}}}\right)}\times 100 \tag{5.57}$$

The bar plots, showing the respective values of the water mass flow rates, the water temperatures, the air temperatures, and the air relative humidity, at the exit of each control volume, are presented in Figures 5.4 through 5.7, below.

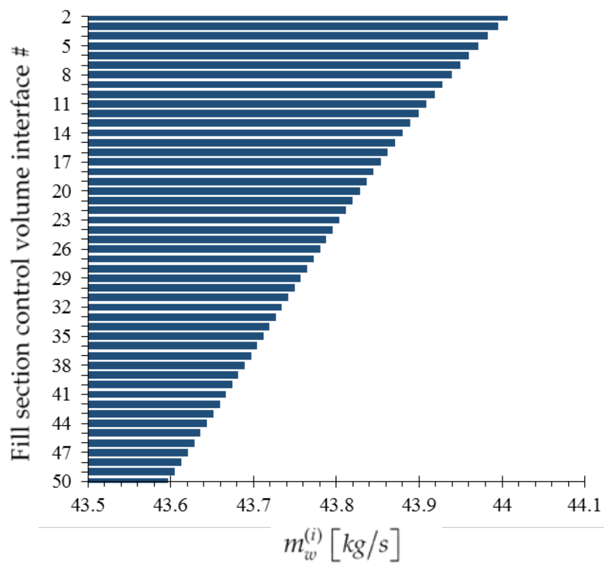

Figure 5.4. Bar plot of the water mass flow rates $m_w^{(i)}$, $(i = 2,...,50)$, at the exit of each control volume along the height of the fill section of the cooling tower.

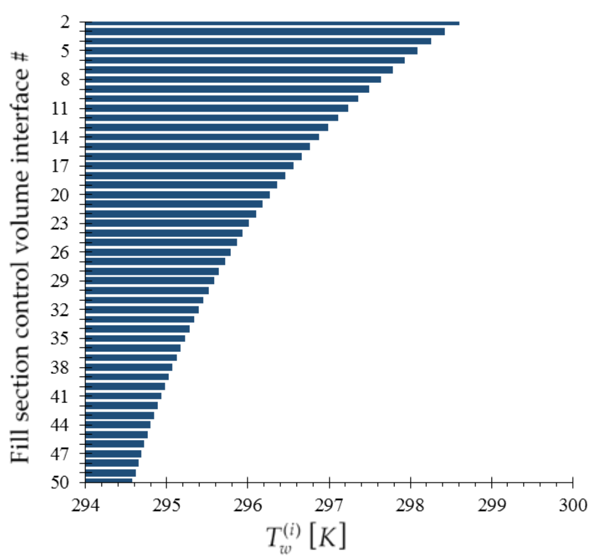

Figure 5.5. Bar plot of the water temperatures $T_w^{(i)}$, $(i = 2,...,50)$, at the exit of each control volume along the height of the fill section of the cooling tower.

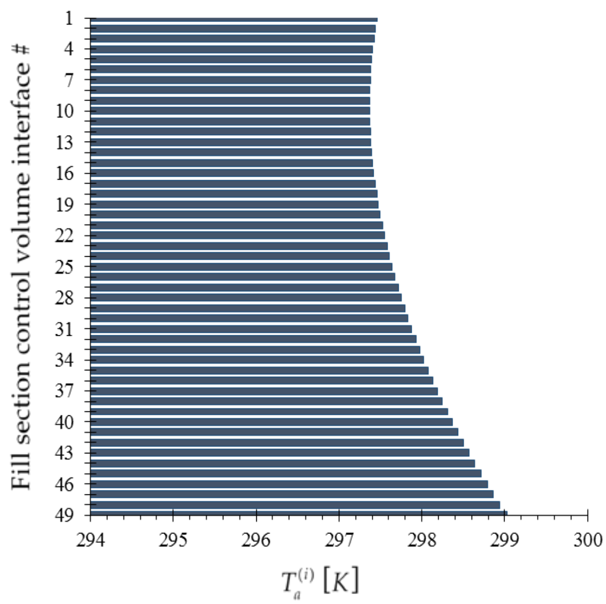

Figure 5.6. Bar plot of the air temperatures $T_a^{(i)}$, $(i = 1,...,49)$, at the exit of each control volume along the height of the fill section of the cooling tower.

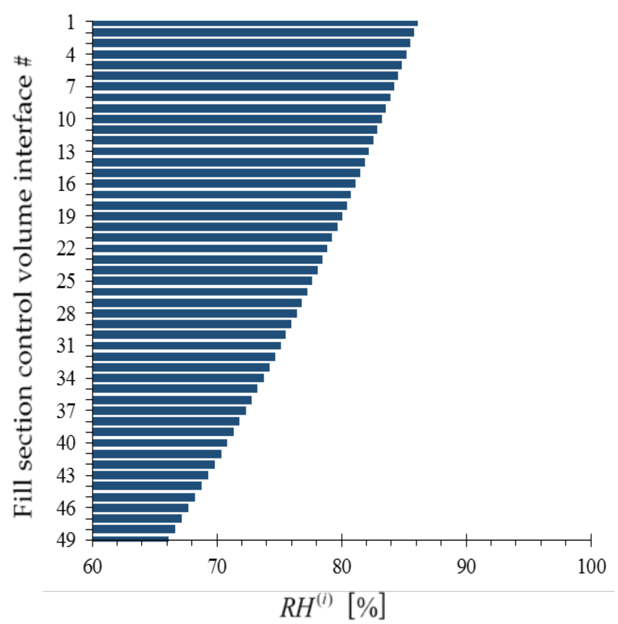

Figure 5.7. Bar plot of the air relative humidity $RH^{(i)}$, $(i = 1,...,49)$, at the exit of each control volume along the height of the fill section of the cooling tower.

5.3 Adjoint Sensitivity Analysis of Cooling Tower Model

All of the responses of interest in this section, e.g., the experimentally measured and/or computed responses discussed in the previous Sections, can be generally represented in the functional form $R\left(\mathbf{m}_w, \mathbf{T}_w, \mathbf{T}_a, \boldsymbol{\omega}; \boldsymbol{\alpha}\right)$, where R is a known functional of the model's state functions and parameters. As generally shown by Cacuci (1981), the sensitivity of such a response to arbitrary variations in the model's parameters $\delta\boldsymbol{\alpha} \triangleq \left(\delta\alpha_1, ..., \delta\alpha_{N_\alpha}\right)$ and state functions $\delta\mathbf{m}_w, \delta\mathbf{T}_w, \delta\mathbf{T}_a, \delta\boldsymbol{\omega}$ is provided by the response's Gateaux (G-) differential $DR\left(\mathbf{m}_w^0, \mathbf{T}_w^0, \mathbf{T}_a^0, \boldsymbol{\omega}^0; \boldsymbol{\alpha}^0; \delta\mathbf{m}_w, \delta\mathbf{T}_w, \delta\mathbf{T}_a, \delta\boldsymbol{\omega}; \delta\boldsymbol{\alpha}\right)$, which is defined as follows:

$$\begin{aligned} & DR\left(\mathbf{m}_w^0, \mathbf{T}_w^0, \mathbf{T}_a^0, \boldsymbol{\omega}^0; \boldsymbol{\alpha}^0; \delta\mathbf{m}_w, \delta\mathbf{T}_w, \delta\mathbf{T}_a, \delta\boldsymbol{\omega}; \delta\boldsymbol{\alpha}\right) \triangleq \\ & \quad \frac{d}{d\varepsilon}\left[R\left(\mathbf{m}_w^0 + \varepsilon\delta\mathbf{m}_w, \mathbf{T}_w^0 + \varepsilon\delta\mathbf{T}_w, \mathbf{T}_a^0 + \varepsilon\delta\mathbf{T}_a, \boldsymbol{\omega}^0 + \varepsilon\delta\boldsymbol{\omega}; \boldsymbol{\alpha}^0 + \varepsilon\delta\boldsymbol{\alpha}\right)\right]_{\varepsilon=0} \\ & \quad = DR_{direct} + DR_{indirect}, \end{aligned} \tag{5.58}$$

where the "direct effect" term, DR_{direct}, and the "indirect effect" term, $DR_{indirect}$, are defined, respectively, as follows:

$$DR_{direct} \equiv \sum_{i=1}^{N_\alpha}\left(\frac{\partial R}{\partial \alpha_i}\delta\alpha_i\right), \tag{5.59}$$

$$\begin{aligned} DR_{indirect} & \triangleq \sum_{i=1}^{I}\left(\frac{\partial R}{\partial m_w^{(i+1)}}\delta m_w^{(i+1)} + \frac{\partial R}{\partial T_w^{(i+1)}}\delta T_w^{(i+1)} + \frac{\partial R}{\partial T_a^{(i)}}\delta T_a^{(i)} + \frac{\partial R}{\partial \omega^{(i)}}\delta\omega^{(i)}\right) \\ & = \mathbf{R}_1 \cdot \delta\mathbf{m}_w + \mathbf{R}_2 \cdot \delta\mathbf{T}_w + \mathbf{R}_3 \cdot \delta\mathbf{T}_a + \mathbf{R}_4 \cdot \delta\boldsymbol{\omega}. \end{aligned} \tag{5.60}$$

The components of the vectors $\mathbf{R}_\ell \equiv \left(r_\ell^{(1)}, ..., r_\ell^{(I)}\right)$, $\ell = 1,2,3,4$, which appear in Eq.(5.60) are defined as follows:

$$r_1^{(i)} \triangleq \frac{\partial R}{\partial m_w^{(i+1)}}; \quad r_2^{(i)} \triangleq \frac{\partial R}{\partial T_w^{(i+1)}}; \quad r_3^{(i)} \triangleq \frac{\partial R}{\partial T_a^{(i)}}; \quad r_4^{(i)} \triangleq \frac{\partial R}{\partial \omega^{(i)}}; \quad i = 1, ..., I. \tag{5.61}$$

Since the model parameters are related to the model's state functions via Eqs. (5.2) through (5.13) , it follows that variations in the model parameter will induce variations in the state variables, which can be computed by solving the G-differentiated model equations, namely:

$$\frac{d}{d\varepsilon}\left[\mathbf{N}\left(\mathbf{u}^0+\varepsilon\delta\mathbf{u};\boldsymbol{\alpha}^0+\varepsilon\delta\boldsymbol{\alpha}\right)\right]_{\varepsilon=0}=\mathbf{0} \tag{5.62}$$

Performing the above G-differentiation on Eqs. (5.2) through (5.13) yields the following forward sensitivity system:

$$\begin{pmatrix} \mathbf{A}_1 & \mathbf{B}_1 & \mathbf{C}_1 & \mathbf{D}_1 \\ \mathbf{A}_2 & \mathbf{B}_2 & \mathbf{C}_2 & \mathbf{D}_2 \\ \mathbf{A}_3 & \mathbf{B}_3 & \mathbf{C}_3 & \mathbf{D}_3 \\ \mathbf{A}_4 & \mathbf{B}_4 & \mathbf{C}_4 & \mathbf{D}_4 \end{pmatrix}\begin{pmatrix} \delta\mathbf{m}_w \\ \delta\mathbf{T}_w \\ \delta\mathbf{T}_a \\ \delta\boldsymbol{\omega} \end{pmatrix}=\begin{pmatrix} \mathbf{Q}_1 \\ \mathbf{Q}_2 \\ \mathbf{Q}_3 \\ \mathbf{Q}_4 \end{pmatrix} \tag{5.63}$$

where the components of the vectors $\mathbf{Q}_\ell \triangleq \left(q_\ell^{(1)},...,q_\ell^{(I)}\right),\ \ell=1,2,3,4,$ are defined as follows:

$$q_\ell^{(i)} \equiv \sum_{j=1}^{N_\alpha}\left(\frac{\partial N_\ell^{(i)}}{\partial\alpha_j}\delta\alpha_j\right);\ i=1,...,I;\ \ell=1,2,3,4, \tag{5.64}$$

and where the matrices $\mathbf{A}_\ell\ ,\mathbf{B}_\ell,\mathbf{C}_\ell,\mathbf{D}_\ell,\ \ \ell=1,2,3,4,$ have been defined in Section 5.2.
The system represented by Eq. (5.63) is called the *forward sensitivity system*, which can be solved, in principle, to compute the variations in the state functions for every variation in the model parameters. In turn, the solution of Eq. (5.63) can be used in Eq. (5.60) to compute the "indirect effect" term, $DR_{indirect}$. However, since there are many parameter variations to consider, solving Eq. (5.63) repeatedly to compute $DR_{indirect}$ becomes computationally impracticable. The need for solving Eq. (5.63) repeatedly to compute $DR_{indirect}$ can be circumvented by applying the Adjoint Sensitivity Analysis Methodology (Cacuci, 1981), which proceeds by forming the inner-product of Eq. (5.63) with a yet unspecified vector of the form $\left[\boldsymbol{\mu}_w,\boldsymbol{\tau}_w,\boldsymbol{\tau}_a,\mathbf{o}\right]^\dagger$, having the same structure as the vector $\mathbf{u}\triangleq\left(\mathbf{m}_w,\mathbf{T}_w,\ \mathbf{T}_a,\boldsymbol{\omega}\right)^\dagger$, transposing the resulting scalar equation and subsequently using Eq. (5.60). By requiring the vector $\left[\boldsymbol{\mu}_w,\boldsymbol{\tau}_w,\boldsymbol{\tau}_a,\mathbf{o}\right]^\dagger$ to satisfy the following adjoint sensitivity system:

$$\begin{pmatrix} \mathbf{A}_1^\dagger & \mathbf{A}_2^\dagger & \mathbf{A}_3^\dagger & \mathbf{A}_4^\dagger \\ \mathbf{B}_1^\dagger & \mathbf{B}_2^\dagger & \mathbf{B}_3^\dagger & \mathbf{B}_4^\dagger \\ \mathbf{C}_1^\dagger & \mathbf{C}_2^\dagger & \mathbf{C}_3^\dagger & \mathbf{C}_4^\dagger \\ \mathbf{D}_1^\dagger & \mathbf{D}_2^\dagger & \mathbf{D}_3^\dagger & \mathbf{D}_4^\dagger \end{pmatrix} \begin{pmatrix} \boldsymbol{\mu}_w \\ \boldsymbol{\tau}_w \\ \boldsymbol{\tau}_a \\ \mathbf{o} \end{pmatrix} = \begin{pmatrix} \mathbf{R}_1 \\ \mathbf{R}_2 \\ \mathbf{R}_3 \\ \mathbf{R}_4 \end{pmatrix}, \tag{5.65}$$

the "indirect effect" term can be expressed in the following form

$$DR_{indirect} = \boldsymbol{\mu}_w \cdot \mathbf{Q}_1 + \boldsymbol{\tau}_w \cdot \mathbf{Q}_2 + \boldsymbol{\tau}_a \cdot \mathbf{Q}_3 + \mathbf{o} \cdot \mathbf{Q}_4 . \tag{5.66}$$

The system represented by Eq. (5.65) is called the *adjoint sensitivity system*, which –notably– is independent of parameter variations. Therefore, the adjoint sensitivity system needs to be solved only once, to compute the adjoint functions $\left[\boldsymbol{\mu}_w, \boldsymbol{\tau}_w, \boldsymbol{\tau}_a, \mathbf{o}\right]^\dagger$. In turn, the adjoint functions are used to compute $DR_{indirect}$, efficiently and exactly, using Eq. (5.66). The units of the adjoint functions are determined from Eq. (5.66) through dimensional analysis:

$$\left[\mu_w^{(i)}\right] = \frac{[R]}{[N_1]}; \quad \left[\tau_w^{(i)}\right] = \frac{[R]}{[N_2]}; \quad \left[\tau_a^{(i)}\right] = \frac{[R]}{[N_3]}; \quad \left[o^{(i)}\right] = \frac{[R]}{[N_4]} \tag{5.67}$$

where "[R]" denotes the unit of the response R, and where the units for the respective equations are as follows:

$$[N_1] = \frac{kg}{s}; \quad [N_2] = \frac{J}{s}; \quad [N_3] = [-]; \quad [N_4] = \frac{J}{kg} . \tag{5.68}$$

Table 5.1, below, lists the units of the adjoint functions for four responses: $R \triangleq T_a^{(1)}$, $R \triangleq T_w^{(50)}$, $R \triangleq RH^{(1)}$ and $R \triangleq m_w^{(50)}$, respectively, in which, $T_a^{(1)}$ denotes exit air temperature; $T_w^{(50)}$ denotes exit water temperature; $RH^{(1)}$ denotes exit air relative humidity; and $m_w^{(50)}$ denotes exit water mass flow rate.

Table 5.1. Units of the adjoint functions for different responses.

Responses	$\left[\mu_w^{(i)}\right]$	$\left[\tau_w^{(i)}\right]$	$\left[\tau_a^{(i)}\right]$	$\left[o^{(i)}\right]$
$R \triangleq T_a^{(1)}$	$K/(kg/s)$	$K/(J/s)$	K	$K/(J/kg)$
$R \triangleq T_w^{(50)}$	$K/(kg/s)$	$K/(J/s)$	K	$K/(J/kg)$
$R \triangleq RH^{(1)}$	$(kg/s)^{-1}$	$(J/s)^{-1}$	−	$(J/kg)^{-1}$
$R \triangleq m_w^{(50)}$	−	$(J/kg)^{-1}$	kg/s	$(kg/s)/(J/kg)$

Note that the adjoint sensitivity system represented by Eq. (5.65) is linear in the adjoint state functions, so it can be solved by using numerical methods appropriate for large-scale sparse linear systems. In particular, we solved it by using NSPCG, (Oppe et al.1988); 12 to 18 iterations sufficed for solving the adjoint system within convergence criterion of $\zeta = 10^{-12}$. Bar plots of the adjoint functions corresponding to the four measured responses of interest, namely: (i) the exit air temperature $R \triangleq T_a^{(1)}$; (ii) the outlet (exit) water temperature $R \triangleq T_w^{(50)}$; (iii) the exit air humidity ratio $R \triangleq RH^{(1)}$; and (iv) the outlet (exit) water mass flow rate $R \triangleq m_w^{(50)}$, are presented by Cacuci and Fang (2016).

The model responses of interest in this work are the following quantities: (i) the outlet air temperature, $T_a^{(1)}$; (ii) the outlet water temperature, $T_w^{(50)}$; (iii) the outlet water flow rate, $m_w^{(50)}$; and (iv) the outlet air relative humidity, $RH^{(1)}$. The analytical expressions of these sensitivities are presented by Cacuci and Fang (2016), and their respective numerical values and rankings, in descending order, are reproduced in Tables 5.2 through 5.5, below. Note that the relative sensitivity, $RS(\alpha_i)$, of a response $R(\alpha_i)$ to a parameter α_i is defined as $RS(\alpha_i) \triangleq \left[dR(\alpha_i)/d\alpha_i\right]\left[\alpha_i/R(\alpha_i)\right]$. Thus, the relative sensitivities are unit less numbers that are very useful in ranking the sensitivities to highlight their relative importance for the respective response. For example, a relative sensitivity of 1.00 indicates that a change of 1% in the respective parameter will induce a 1% change in a response that is linear in the respective sensitivity. The higher the relative sensitivity, the more important the respective parameter to the respective response.

The numerical results and ranking of the relative sensitivities of the air outlet temperature with respect to all of the model's parameters are provided, in descending order of their respective magnitudes, in Table 5.2, below, along with their respective relative standard deviations.

Table 5.2. Ranked relative sensitivities of the outlet air temperature, $T_a^{(1)}$.

Rank #	**Parameter** (α_i)	**Nominal Value**	**Rel. Sens.** $RS(\alpha_i)$	**Rel. std. dev. (%)**
1	Inlet air temperature, $T_{a,in}$	299.11 K	0.4858	1.39
2	Air temperature (dry bulb) , T_{db}	299.11 K	0.4829	1.39
3	Inlet water temperature, $T_{w,in}$	298.79 K	0.2756	0.57
4	Dew point temperature , T_{dp}	292.05 K	0.1834	0.81
5	P_{vs}(T) parameter, a_0	25.5943	-0.0945	0.04
6	P_{vs}(T) parameter, a_1	5229.89	0.0618	0.08
7	Inlet air humidity ratio, ω_{in}	0.0138	0.0100	14.93
8	Fan shroud inner diameter, D_{fan}	4.1 m	-0.0056	1.00
9	Water enthalpy h_f(T) parameter, a_{1f}	4186.51	0.0050	0.04
10	Wetted fraction of fill surface area, w_{tsa}	1.0	-0.0049	0.00
11	Nusselt number, Nu	14.94	-0.0049	34.0
12	Fill section surface area, A_{surf}	14221 m^2	-0.0049	25.0
13	Dynamic viscosity of air at T=300K, μ	1.983E-5 kg/(m s)	0.0045	4.88
14	Nu parameter, $a_{1,Nu}$	0.0031498	-0.0045	31.75
15	Reynolds number, Re_d	4428	-0.0045	15.17
16	Fill section flow area, A_{fill}	67.29 m^2	0.0045	10.0
17	C_{pa}(T) parameter, $a_{0,cpa}$	1030.5	0.0032	0.03
18	Inlet water mass flow rate, $m_{w,in}$	44.02 kg/s	0.0031	5.0
19	h_g(T) parameter, a_{0g}	2005744	-0.0030	0.05

20	D_{av}(T) parameter, $a_{1,dav}$	2.65322	0.0028	0.11
21	Exit air speed at the shroud, V_{exit}	10.0 m/s	-0.0028	10.0
22	Inlet air mass flow rate, m_a	155.07 kg/s	-0.0028	10.26
23	Heat transfer coefficient multiplier, f_{ht}	1.0	-0.0026	50.0
24	Thermal conductivity of air at T=300K, k_{air}	0.02624 W/(m K)	-0.0026	6.04
25	Mass transfer coefficient multiplier, f_{mt}	1.0	-0.0022	50.0
26	Sherwood number, Sh	14.13	-0.0022	34.25
27	D_{av}(T) parameter, $a_{2,dav}$	-6.1681E-3	-0.0019	0.37
28	h_f(T) parameter, a_{0f}	1143423	-0.0017	0.05
29	D_{av}(T) parameter, $a_{0,dav}$	7.06085E-9	-0.0015	0
30	Atmospheric pressure, P_{atm}	100586 Pa	-0.0013	0.40
31	Kinematic viscosity of air at 300 K, ν	1.568E-5 m^2/s	-0.00074	12.09
32	Prandlt number of air at T=80 C, Pr	0.708	0.00074	0.71
33	Schmidt number, Sc	0.60	-0.00074	12.41
34	h_g(T) parameter, a_{1g}	1815.437	-0.00074	0.19
35	D_{av}(T) parameter, $a_{3,dav}$	6.55265E-6	0.00063	0.58
36	Nu parameter, $a_{2,Nu}$	0.9902987	-0.00032	33.02
37	Fill section equivalent diameter, D_h	0.0381 m	0.00032	1.0
38	C_{pa}(T) parameter, $a_{1,cpa}$	-0.19975	-0.00018	1.0
39	C_{pa} (T) parameter, $a_{2,cpa}$	3.9734E-4	0.00010	0.84
40	Sum of loss coefficients above fill, k_{sum}	10.0	0.000	50.0
41	Fill section frictional loss multiplier, f	4.0	0.000	50.0
42	Nu parameter, $a_{0,Nu}$	8.235	0.000	25.0
43	Nu parameter, $a_{3,Nu}$	0.023	0.000	38.26

44	Cooling tower deck width in x-dir, W_{dkx}	8.5 m	0.000	1.0
45	Cooling tower deck width in y-dir, W_{dky}	8.5 m	0.000	1.0
46	Cooling tower deck height above ground, Δz_{dk}	10.0 m	0.000	1.0
47	Fan shroud height, Δz_{fan}	3.0 m	0.000	1.0
48	Fill section height, Δz_{fill}	2.013 m	0.000	1.0
49	Rain section height, Δz_{rain}	1.633 m	0.000	1.0
50	Basin section height, Δz_{bs}	1.168 m	0.000	1.0
51	Drift eliminator thickness, Δz_{de}	0.1524 m	0.000	1.0
52	Wind speed, V_w	1.80 m/s	0.000	51.1

As the results in Table 5.2 indicate, the first 5 parameters (i.e., $T_{a,in}$, T_{db}, $T_{w,in}$, T_{dp}, a_0) have relative sensitivities between ca. 10% and 50%, and are therefore the most important for the air outlet temperature response, $T_a^{(1)}$. The two largest sensitivities have values of 48%, which means that a 1% change in $T_{a,in}$ or T_{db} would induce a 0.48% change in $T_a^{(1)}$. The next two parameters (i.e., a_1 and ω_{in}) have relative sensitivities between 1% and 6%, and are therefore somewhat important. Parameters #8 through #16 (i.e.,. D_{fan}, a_{1f}, w_{tsa}, Nu, A_{surf}, μ, $a_{1,Nu}$, Re_d, A_{fill}) have relative sensitivities of the order of 0.5%. The remaining 36 parameters are relatively unimportant for this response, having relative sensitivities smaller than 1% of the largest relative sensitivity (with respect to $T_{a,in}$) for this response. Positive sensitivities imply that a positive change in the respective parameter would cause an increase in the response, while negative sensitivities imply that a positive change in the respective parameter would cause a decrease in the response.

The results and ranking of the relative sensitivities of the outlet water temperature with respect to the most important 12 parameters for this response are listed in Table 5.3. The largest sensitivity of $T_w^{(50)}$ is to the parameter T_{dp}, and has the value of 0.548; this means that a 1% increase in T_{db} would induce a 0.548% increase in $T_w^{(50)}$ The sensitivities to the remaining 40 model parameters have not been listed since they are smaller than 1% of the largest sensitivity (with respect to T_{dp}) for this response.

Table 5.3. Most important relative sensitivities of the outlet water temperature, $T_w^{(50)}$.

Rank #	**Parameter** (α_i)	**Nominal value**	**Rel. Sens.** $RS(\alpha_i)$	**Rel. std. dev.(%)**
1	Dew point temperature , T_{dp}	292.05 K	0.5482	0.81
2	Inlet air temperature, $T_{a,in}$	299.11 K	0.2318	1.39
3	Air temperature (dry bulb) , T_{db}	299.11 K	0.2244	1.39
4	P_{vs}(T) parameters, a_0	25.5943	-0.1949	0.04
5	P_{vs}(T) parameters, a_1	-5229.89	0.1282	0.08
6	Inlet water temperature, $T_{w,in}$	298.79 K	0.1066	0.57
7	Inlet air humidity ratio, ω_{in}	0.0138	0.0299	14.93
8	Fan shroud inner diameter, D_{fan}	4.1 m	-0.0085	1.00
9	Water enthalpy hf(T) parameter, a_{1f}	4186.51	0.0082	0.04
10	$D_{av}(T_{db})$ parameter, $a_{1,dav}$	2.653	0.0071	0.11
11	Enthalpy h_g(T) parameter, a_{0g}	2005744	-0.0062	0.05
12	Sherwood number, Sh	14.13	-0.0056	34.25

The results and ranking of the relative sensitivities of the outlet water mass flow rate with respect to the most important 10 parameters for this response are listed in Table 5.4. This response is most sensitive to $m_{w,in}$ (a 1% increase in this parameter would cause a 1.01% increase in the response) and the second largest sensitivity is to the parameter $T_{w,in}$ (a 1% increase in this parameter would cause a 0.447% decrease in the response). The sensitivities to the remaining 42 model parameters have not been listed since they are smaller than 1% of the largest sensitivity (namely, with respect to $m_{w,in}$) for this response.

Table 5.4. Most important relative sensitivities of the outlet water mass flow rate, $m_w^{(50)}$.

Rank #	**Parameter** (α_i)	**Nominal value**	**Rel. Sens.** $RS(\alpha_i)$	**Rel. std. dev. (%)**
1	Inlet water mass flow rate, $m_{w,in}$	44.02 kg/s	1.0060	5.00
2	Inlet water temperature, $T_{w,in}$	298.79 K	-0.4474	0.57
3	Dew point temperature , T_{dp}	292.05 K	0.3560	0.81
4	Pvs(T) parameters, a_0	25.5943	-0.1416	0.04
5	Air temperature (dry bulb) , T_{db}	299.11 K	-0.1184	1.39
6	Inlet air temperature, $T_{a,in}$	299.11 K	-0.1134	1.39
7	Pvs(T) parameters, a_1	5229.89	0.0930	0.08
8	Inlet air humidity ratio, ω_{in}	0.0138	0.0195	14.93
9	Fan shroud inner diameter, D_{fan}	4.1 m	-0.0117	1.00
10	Inlet air mass flow rate, m_a	155.07 kg/s	-0.0058	10.26

The results and ranking of the relative sensitivities of the outlet air relative humidity with respect to the most important 20 parameters for this response are listed in Table 5.5. The first three sensitivities of this response are quite large (relative sensitivities larger than unity are customarily considered to be very significant). In particular, an increase of 1% in $T_{a,in}$ or T_{db} would cause a decrease in the response of 6.66% or 6.525%, respectively. On the other hand, an increase of 1% in T_{dp} would cause an increase of 5.75% in the response. The sensitivities to the remaining 32 model parameters have not been listed since they are smaller than 1% of the largest sensitivity (with respect to $T_{a,in}$) for this response.

Table 5.5. Most important relative sensitivities of the outlet air relative humidity, $RH^{(1)}$.

Rank #	**Parameter** (α_i)	**Nominal value**	**Rel. Sens.** $RS(\alpha_i)$	**Rel. std. dev. (%)**
1	Inlet air temperature, $T_{a,in}$	299.11 K	-6.660	1.39
2	Air temperature (dry bulb) , T_{db}	299.11 K	-6.525	1.39
3	Dew point temperature , T_{dp}	292.05 K	5.750	0.81
4	Inlet water temperature, $T_{w,in}$	298.79 K	0.747	0.57
5	Inlet air humidity ratio, ω_{in}	0.0138	0.3141	14.93
6	P_{vs}(T) parameters, a_0	25.5943	-0.3123	0.04
7	Wetted fraction of fill surface area, w_{tsa}	1.0	0.1487	0.00
8	Fill section surface area, A_{surf}	14221 m^2	0.1487	25.0
9	Nusselt number, Nu	14.94	0.1487	34.0
10	Dynamic viscosity of air at T=300 K, μ	1.983E-5 kg/(m s)	-0.1388	4.88
11	Nu parameters, $a_{1,Nu}$	0.0031498	0.1388	31.75
12	Fill section flow area, A_{fill}	67.29 m^2	-0.1388	10.0
13	Reynold's number, Re	4428	0.1388	15.17
14	$D_{av}(T_{db})$ parameter, $a_{1,dav}$	2.65322	-0.1297	0.11
15	Mass transfer coefficient multiplier, f_{mt}	1.0	0.1023	50.0
16	Sherwood number, Sh	14.13	0.1023	34.25
17	Atmosphere pressure, P_{atm}	100586 Pa	0.0992	0.40
18	$D_{av}(T_{db})$ parameter, $a_{2,dav}$	-6.1681E-3	0.0902	0.37
19	$D_{av}(T_{db})$ parameter, $a_{0,dav}$	7.06085E-9	0.0682	0.00
20	P_{vs}(T) parameters, a_1	-5229.89	0.0681	0.08

Overall, the outlet air relative humidity, $RH^{(1)}$, displays the largest sensitivities, so this response is the most sensitive to parameter variations. The other responses, namely the outlet air temperature, the outlet water temperature, and the outlet water mass flow rate display sensitivities of comparable magnitudes.

5.4 Predictive Modeling: Optimal Best-Estimate Results with Reduced Predicted Uncertainties

A total of 7668 measured data sets fall into the "unsaturated" case presented in this illustrative example. The measured outlet (exit) air relative humidity, RH^{meas}, was obtained using Hobo humidity sensors. The accuracy of these sensors is depicted in Figure 5.7, which indicates the following tolerances (standard deviations): ±2.5% for relative humidity from 10 to 90%; between ±2.5% and ±3.5% for relative humidity from 90% to 95%; and ±3.5% ~ ±4.0% from 95 to 100%. However, when exposed to relative humidity above 95%, the maximum sensor error may temporally increase by an additional 1%, so that the error can reach values between ±4.5% to ±5.0% for relative humidity from 95 to 100%.

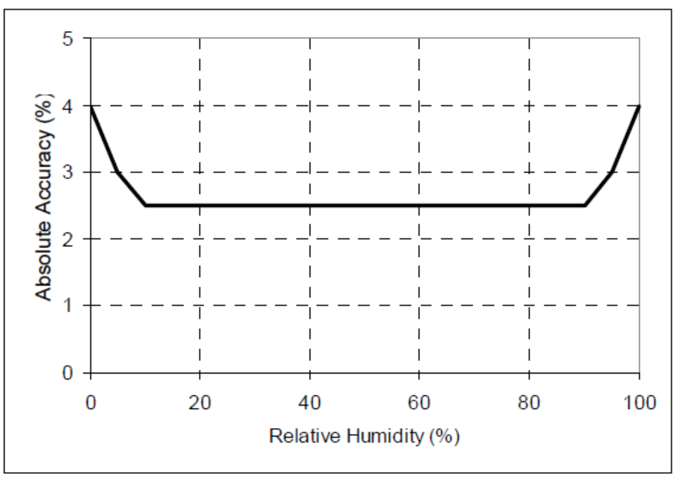

Figure 5.7: Humidity sensor accuracy plot (adopted from the specification of HOBO Pro v2).

The 7668 measured values of the outlet (exit) air relative humidity, RH^{meas}, considered to be "unsaturated," are presented in the histogram plot shown in Figure 5.8. As shown in this figure, although the computed relative humidity for each of the 7668 data sets is less than 100%, the measured relative humidity RH^{meas} actually spans the range from 33.0% to 104.1%; in this range, 6975 data sets have their respective RH^{meas} less than 100% while the other 693 data sets have their respective RH^{meas} over 100%. This situation is nevertheless consistent with the range of the sensors when their tolerances (standard deviations) are taken into account, which would make it possible for a measurement with RH^{meas} =105% to be nevertheless "unsaturated". Consequently, all the 7668 benchmark data sets plotted in Figure 5.8 were considered as "unsaturated", since their respective RH^{meas} was less than 105%. This plot, as well as all of the other histogram plots in this work, have their total respective areas normalized to unity.

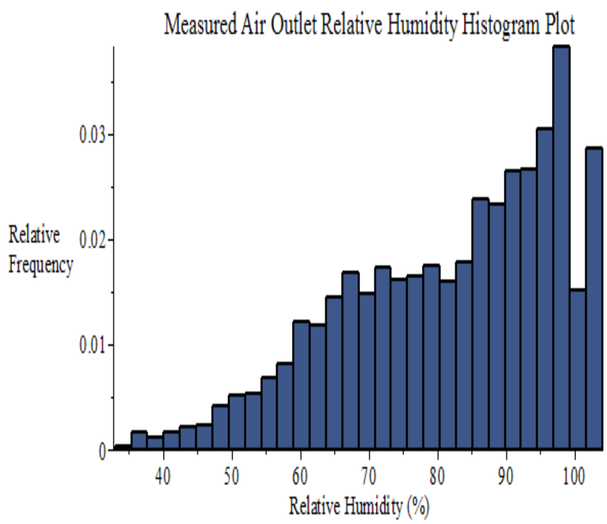

Figure 5.8: Histogram plot of the measured air outlet relative humidity, within the 7688 data sets collected by SRNL from F-Area cooling towers (unsaturated conditions).

The statistical properties of the (measured air outlet relative humidity) distribution shown in Figures 5.8 have been computed using standard packages, and are presented in Table 5.6. These statistical properties will be needed for the uncertainty quantification and predictive modeling computations presented in the main body of this work.

Table 5.6. Statistics of the air outlet relative humidity distribution [%].

Minimum	**Maximum**	**Range**	**Mean**	**Std. Dev.**	**Variance**	**Skewness**	**Kurtosis**
33.0	104.1	71.1	81.98	15.63	244.44	-0.60	2.55

The histogram plots and their corresponding statistical characteristics of the 7668 data sets for the other measurements, namely for: the outlet air temperature [$T_{a,out(Tidbit)}$] measured using the

"Tidbit" sensors; the outlet air temperature [$T_{a,out(Hobo)}$] measured using the "Hobo" sensors; and the outlet water temperature [$T_{w,out}^{meas}$] are reported below in Figures 5.9 through 5.11, and Tables 5.7 through 5.9, respectively.

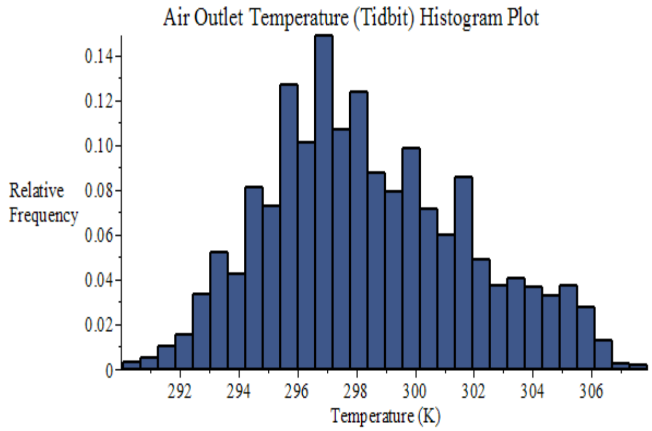

Figure 5.9. Histogram plot of the air outlet temperature measured using "Tidbit" sensors, within the 7688 data sets collected by SRNL from F-Area cooling towers (unsaturated conditions).

Table 5.7. Statistics of the air outlet temperature distribution [K], measured using "Tidbit" sensors.

Minimum	**Maximum**	**Range**	**Mean**	**Std. Dev.**	**Variance**	**Skewness**	**Kurtosis**
290.06	307.89	17.83	298.42	3.42	11.71	0.34	2.52

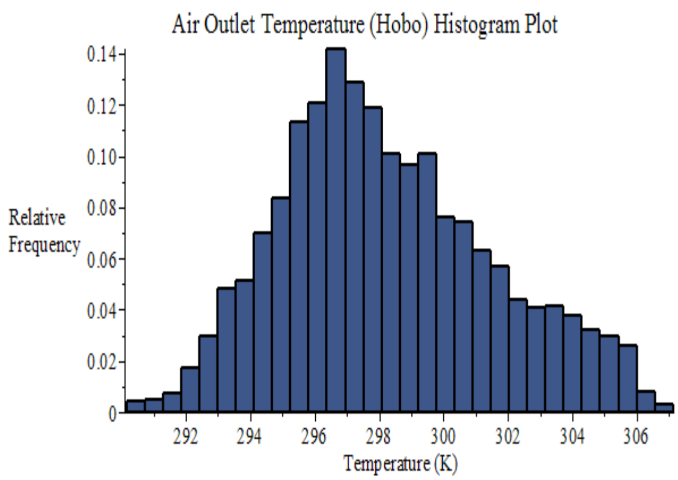

Figure 5.10. Histogram plot of the air outlet temperature measured using "Hobo" sensors, within the 7688 data sets collected by SRNL from F-Area cooling towers (unsaturated conditions).

Table 5.8. Air outlet temperature distribution statistics [K], measured using "Hobo" sensors.

Minimum	Maximum	Range	Mean	Std. Dev.	Variance	Skewness	Kurtosis
290.17	307.13	16.96	298.27	3.30	10.88	0.36	2.56

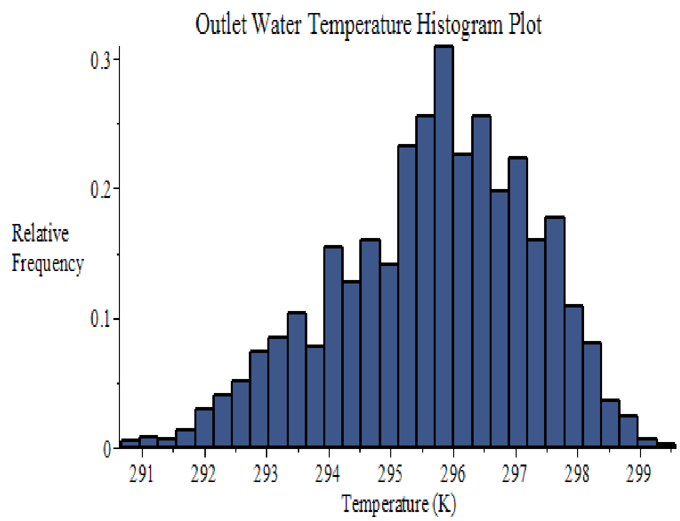

Figure 5.11. Histogram plot of water outlet temperature measurements, within the 7688 data sets collected by SRNL from F-Area cooling towers (unsaturated conditions).

Table 5.9. Water outlet temperature distribution statistics [K].

Minimum	Maximum	Range	Mean	Std. Dev.	Variance	Skewness	Kurtosis
290.67	299.57	8.90	295.68	1.58	2.48	-0.41	2.72

Ordering the above-mentioned four measured responses as follows: (i) outlet air temperature $T_{a,out(Tidbit)}$; (ii) outlet air temperature $T_{a,out(Hobo)}$; (iii) outlet water temperature $T_{w,out}^{meas}$; and (iv) outlet air relative humidity RH_{out}^{meas}, yields the following "measured response covariance matrix", denoted as $Cov\left(T_{a,out(Tidbit)}, T_{a,out(Hobo)}, T_{w,out}^{meas}, RH_{out}^{meas}\right)$:

$$Cov\left(T_{a,out(Tidbit)}, T_{a,out(Hobo)}, T_{w,out}^{meas}, RH_{out}^{meas}\right) = \begin{pmatrix} 11.71 & 11.23 & 3.57 & -44.76 \\ 11.23 & 10.88 & 3.52 & -42.94 \\ 3.57 & 3.52 & 2.48 & -5.31 \\ -44.76 & -42.94 & -5.31 & 244.44 \end{pmatrix}. \tag{5.69}$$

For the purposes of uncertainty quantification, data assimilation, model calibration and predictive modeling, the temperatures measurements provided by the "Tidbit" and "Hobo" sensors can be combined into an "averaged" data set of measured air outlet temperatures, which will be denoted as $T_{a,out}^{meas}$. The histogram plot and corresponding statistical characteristics of this averaged air outlet temperature are presented in Figure 5.12 and Table 5.10, respectively.

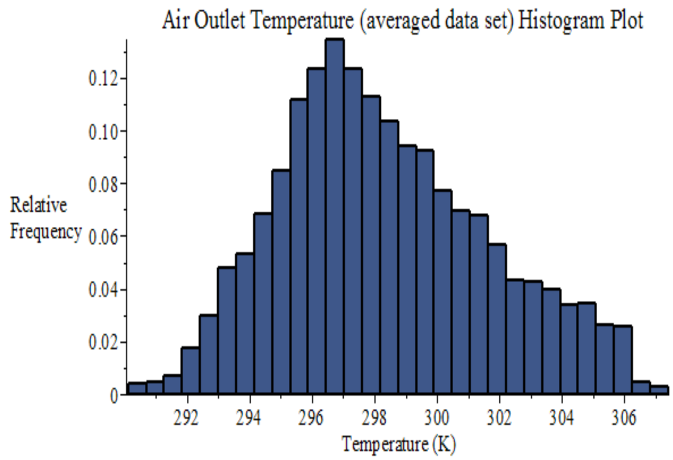

Figure 5.12. Histogram plot of air outlet temperatures

Table 5.10. Statistics of the averaged air outlet temperature distribution [K].

Minimum	**Maximum**	**Range**	**Mean**	**Std. Dev.**	**Variance**	**Skewness**	**Kurtosis**
290.12	307.41	17.30	298.34	3.36	11.27	0.35	2.54

Computing the covariance matrix, denoted as $\left[Cov\left(T_{a,out}^{meas}, T_{w,out}^{meas}, RH_{out}^{meas}\right)\right]_{data}$, for all of the relevant experimental data for the averaged outlet air temperature $\left[T_{a,out}^{meas}\right]$, the outlet water temperature $\left[T_{w,out}^{meas}\right]$, and the outlet air relative humidity $\left[RH_{out}^{meas}\right]$, yields the following result:

$$\left[Cov\left(T_{a,out}^{meas}, T_{w,out}^{meas}, RH_{out}^{meas}\right)\right]_{data} = \begin{pmatrix} 11.27 & 3.55 & -43.85 \\ 3.55 & 2.48 & -5.31 \\ -43.85 & -5.31 & 244.44 \end{pmatrix}. \qquad (5.70)$$

Comparing the results in Eqs. (5.69) and (5.70) shows that eliminating the second column and second row in Eq. (5.69) yields a 3-by-3 matrix which has entries essentially equivalent to the covariance matrix in Eq. (5.70). In turn, this result indicates that the temperature distributions measured by the "Tidbit" and "Hobo" sensors, respectively, need not be treated as separate data sets for the purposes of uncertainty quantification and predictive modeling.

The sensors' standard deviations (namely: $\sigma_{sensor} = 0.2K$ for each of the responses $T_a^{(1)}$ and $T_w^{(50)}$, and $\sigma_{sensor} = 2.8\%$ for the response $RH^{(1)}$) have been taken into account for the data at the 100%-saturation point, by including the 693 data sets that have their respective measured relative humidity, RH^{meas}, between 100% and 104.1%. In addition, the respective sensors' uncertainties (standard deviations) must also be taken into account for the 6975 data sets that have their respective RH^{meas} less than 100%. Since the various measuring methods and devices are independent of each other, the standard deviation, $\sigma_{statistic}$, stemming from the statistical analysis of the 7668 benchmark data sets and the standard deviation, σ_{sensor}, stemming from the instrument's uncertainty are to be combined according to the well-known formula "addition of the variances of uncorrelated variates", namely:

$$\sigma = \sqrt{\sigma_{statistic}^{\ 2} + \sigma_{sensor}^{\ 2}}, \tag{5.71}$$

Using Eq. (5.71) in conjunction with the result presented in Eq.(5.70) will lead to an increase of the variances on the diagonal of the respective "measured covariance matrix", which will be denoted as $Cov\left(T_{a,out}^{meas}, T_{w,out}^{meas}, RH_{out}^{meas}\right)$. The final result thus obtained is

$$Cov\left(T_{a,out}^{meas}, T_{w,out}^{meas}, RH_{out}^{meas}\right) = \begin{pmatrix} 11.29 & 3.55 & -43.85 \\ 3.55 & 2.53 & -5.31 \\ -43.85 & -5.31 & 252.49 \end{pmatrix}. \tag{5.72}$$

The correlation matrix between the measured parameters and responses, denoted as $Cov\left(T_{a,out}^{meas}, T_{w,out}^{meas}, RH^{meas}, \alpha_1, ..., \alpha_{52}\right)$, is presented below:

$$Cov\left(T_{a,out}^{meas}, T_{w,out}^{meas}, RH^{meas}, \alpha_1, ..., \alpha_{52}\right) = \begin{pmatrix} 12.96 & 3.51 & 2.33 & -447.09 & 0 & \cdots & 0 \\ 3.35 & 3.05 & 1.89 & -93.58 & 0 & \cdots & 0 \\ -54.16 & 1.73 & -2.27 & 1831.03 & 0 & \cdots & 0 \end{pmatrix}. \tag{5.73}$$

Parameters α_1 through α_4 (i.e., the dry bulb air temperature, dew point temperature, inlet water temperature, and atmospheric pressure) were also measured at the F-area SRNL site. Among the 8079 measured benchmark data sets, 7688 data sets are considered to represent "unsaturated conditions", which have been used to derive the statistical properties (means, variance and covariance, skewness and kurtosis) for these model parameters, as shown below in Figures 5.13 through 5.16 and Tables 5.11 through 5.14.

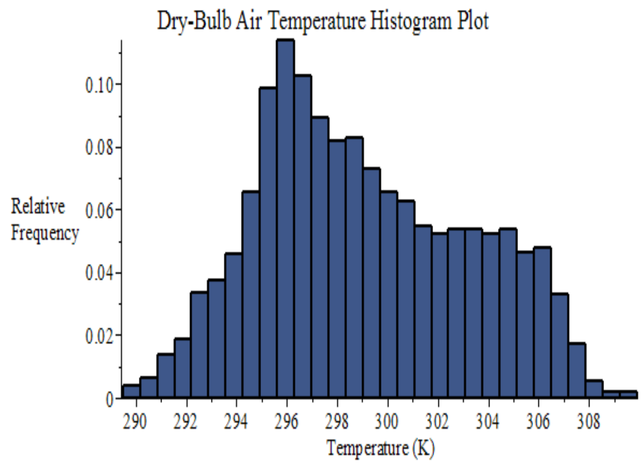

Figure 5.13. Histogram plot of dry-bulb air temperature data collected by SRNL from F-Area cooling towers (unsaturated conditions).

Table 5.11. Statistics of the dry-bulb temperature (set to air inlet temperature) distribution [K].

Minimum	**Maximum**	**Range**	**Mean**	**Std. Dev.**	**Variance**	**Skewness**	**Kurtosis**
289.50	309.91	20.41	299.11	4.17	17.37	0.25	2.18

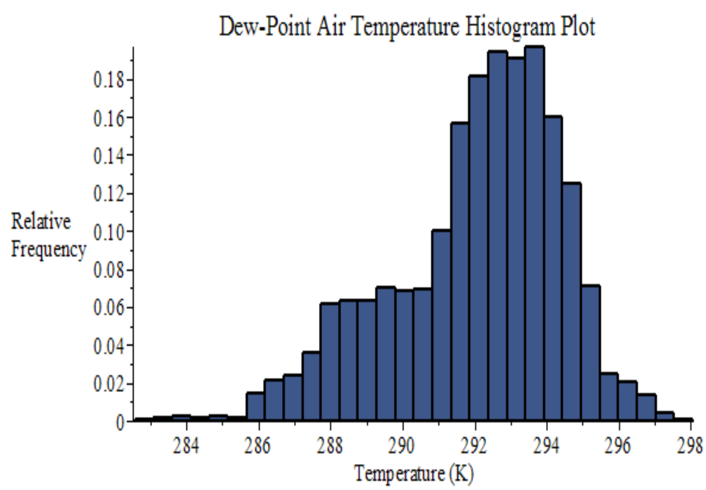

Figure 5.14. Histogram plot of dew-point air temperature data collected by SRNL from F-Area cooling towers (unsaturated conditions).

Table 5.12. Statistics of the dew-point temperature distribution [K].

Minimum	Maximum	Range	Mean	Std. Dev.	Variance	Skewness	Kurtosis
282.58	298.06	15.48	292.05	2.36	5.57	-0.66	3.10

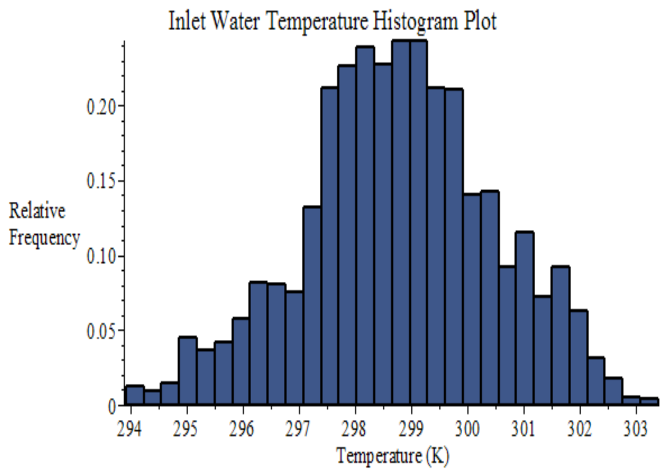

Figure 5.15. Histogram plot of inlet water temperature data collected by SRNL from F-Area cooling towers (unsaturated conditions).

Table 5.13. Statistics of the inlet water temperature distribution [K].

Minimum	Maximum	Range	Mean	Std. Dev.	Variance	Skewness	Kurtosis
293.93	303.39	9.46	298.79	1.70	2.90	-0.12	2.84

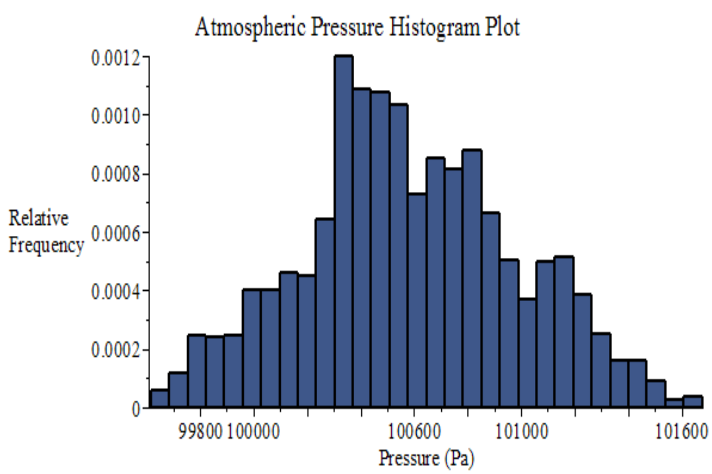

Figure 5.16. Histogram plot of atmospheric pressure data collected by SRNL from F-Area cooling towers (unsaturated conditions).

Table 5.14. Statistics of the atmospheric pressure distribution [Pa].

Minimum	Maximum	Range	Mean	Std. Dev.	Variance	Skewness	Kurtosis
99617	101677	2060	100586	401	160597	0.10	2.58

Using the results presented in Tables 5.11 through 5.14, and ordering these from model parameters as follows: the dry-bulb air temperature, T_{db}; the dew-point air temperature, T_{dp}; the inlet water temperature $T_{w,in}$, and atmospheric air pressure P_{atm}, yields the following 4-by-4 covariance matrix:

$$Cov\left(T_{db};T_{dp};T_{w,in};P_{atm}\right)=\begin{pmatrix} 17.37 & 2.83 & 1.81 & -529.26 \\ 2.83 & 5.56 & 2.31 & -87.16 \\ 1.81 & 2.31 & 2.90 & -47.22 \\ -529.26 & -87.16 & -47.22 & 160597.01 \end{pmatrix}. \quad (5.74)$$

The covariance matrix computed in Eq.(5.74) neglects the uncertainty associated with sensor readings throughout the data collection period. When combining uncertainties by adding variances, the contribution from the sensors is 0.04 K for each of the first three parameters, which accounts for a maximum of ca. 1% of the total variance (for the inlet water temperature, specifically). The uncertainty in the atmospheric pressure sensor is negligibly small. The matrix presented in Eq.(5.74) is used to obtain the following "a priori" parameter covariance matrix, $\mathbf{C}_{\alpha\alpha}$:

$$\mathbf{C}_{\alpha\alpha} \triangleq \begin{pmatrix} Var(\alpha_1) & Cov(\alpha_1,\alpha_2) & \bullet & Cov(\alpha_1,\alpha_{52}) \\ Cov(\alpha_2,\alpha_1) & Var(\alpha_2) & \bullet & Cov(\alpha_2,\alpha_{52}) \\ \bullet & \bullet & \bullet & \bullet \\ Cov(\alpha_{52},\alpha_1) & \bullet & \bullet & Var(\alpha_{52}) \end{pmatrix}$$
$$= \begin{pmatrix} 17.37 & 2.83 & 1.81 & -529.26 & 0 & \bullet & 0 \\ 2.83 & 5.56 & 2.31 & -87.16 & 0 & \bullet & 0 \\ 1.81 & 2.31 & 2.90 & -47.22 & 0 & \bullet & 0 \\ -529.26 & -87.16 & -47.22 & 160597.01 & 0 & \bullet & 0 \\ 0 & 0 & 0 & 0 & \bullet & \bullet & 0 \\ \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\ 0 & 0 & 0 & 0 & 0 & \bullet & 25.81 \end{pmatrix} \quad (5.75)$$

The a priori covariance matrix of the computed responses, $\mathbf{C}_{rr}^{comp}$, is obtained by using Eqs.(4.22) and (5.75) together with the sensitivity results presented in Tables 5.2 through 5.4; the final result is given below:

$$\mathbf{C}_{rr}^{comp} \triangleq Cov\left(T_a^{(1)}, T_w^{(50)}, RH^{(1)}\right) = \mathbf{S}_{r\alpha}\mathbf{C}_{\alpha\alpha}\mathbf{S}_{r\alpha}^{\dagger}$$
$$= \begin{pmatrix} \frac{\partial T_a^{(1)}}{\partial \alpha_1},\ldots,\frac{\partial T_a^{(1)}}{\partial \alpha_{N\alpha}} \\ \frac{\partial T_w^{(50)}}{\partial \alpha_1},\ldots,\frac{\partial T_w^{(50)}}{\partial \alpha_{N\alpha}} \\ \frac{\partial RH^{(1)}}{\partial \alpha_1},\ldots,\frac{\partial RH^{(1)}}{\partial \alpha_{N\alpha}} \end{pmatrix} \begin{pmatrix} Var(\alpha_1) & Cov(\alpha_1,\alpha_2) & \bullet & Cov(\alpha_1,\alpha_{52}) \\ Cov(\alpha_2,\alpha_1) & Var(\alpha_2) & \bullet & Cov(\alpha_2,\alpha_{52}) \\ \bullet & \bullet & \bullet & \bullet \\ Cov(\alpha_{52},\alpha_1) & \bullet & \bullet & Var(\alpha_{52}) \end{pmatrix} \begin{pmatrix} \frac{\partial T_a^{(1)}}{\partial \alpha_1},\ldots,\frac{\partial T_a^{(1)}}{\partial \alpha_{N\alpha}} \\ \frac{\partial T_w^{(50)}}{\partial \alpha_1},\ldots,\frac{\partial T_w^{(50)}}{\partial \alpha_{N\alpha}} \\ \frac{\partial RH^{(1)}}{\partial \alpha_1},\ldots,\frac{\partial RH^{(1)}}{\partial \alpha_{N\alpha}} \end{pmatrix}^{\dagger}$$
$$= \begin{pmatrix} 10.87 & 7.19 & -34.81 \\ 7.19 & 7.72 & -13.97 \\ -34.81 & -13.97 & 221.88 \end{pmatrix}. \tag{5.76}$$

The a priori covariance matrix, $Cov\left(T_{a,out}^{meas}, T_{w,out}^{meas}, RH_{out}^{meas}\right) \triangleq \mathbf{C}_{rr}$, of the measured responses (namely: the outlet air temperature, $T_{a,out}^{meas} \equiv \left[T_a^{(1)}\right]^{measured}$; the outlet water temperature, $T_{w,out}^{meas} \equiv \left[T_w^{(50)}\right]^{measured}$, and the outlet air relative humidity, $RH_{out}^{meas} \equiv \left[RH^{(1)}\right]^{measured}$) is given below:

$$Cov\left(T_{a,out}^{meas}, T_{w,out}^{meas}, RH_{out}^{meas}\right) \triangleq \mathbf{C}_{rr} = \begin{pmatrix} 11.29 & 3.55 & -43.85 \\ 3.55 & 2.53 & -5.31 \\ -43.85 & -5.31 & 252.49 \end{pmatrix}. \tag{5.77}$$

The best-estimate nominal parameter values have been computed using Eq.(4.16) in conjunction with the a priori matrices given in Eqs.(5.73), (5.75) and (5.76) together with the sensitivities presented in Tables 5.2 through 5.5. The resulting best-estimate nominal values are listed in Table 5.15, below. The corresponding best-estimate absolute standard deviations for these parameters are also presented in this table. These values are the square-roots of the diagonal elements of the matrix $\mathbf{C}_{\alpha\alpha}^{pred}$, which is computed using Eq.(4.18) in conjunction with the a priori matrices given in Eqs.(5.73), (5.75) and (5.76) and the sensitivities presented in Tables 5.2 through 5.5. For comparison, the original nominal parameter values and original absolute standard deviations are also listed. As the results in Table 5.15 indicate, the predicted best-estimate standard deviations are all smaller or at most equal to (i.e., left unaffected) the original standard deviations. The parameters are affected proportionally to the magnitudes of their corresponding sensitivities: the parameters experiencing the largest reductions in their predicted standard deviations are those having the largest sensitivities.

Table 5.15. Best-estimated nominal parameter values and their standard deviations.

i	**Scalar Parameter** (α_i)	**Symbol**	**Original Nominal Value**	**Original Absolute Std. Dev.**	**Best-estimated Nominal Value**	**Best-estimated Absolute Std. Dev.**
1	Air temperature (dry bulb), (K)	T_{db}	299.11	4.17	299.37	3.44
2	Dew point temperature (K)	T_{dp}	292.05	2.36	292.23	2.28
3	Inlet water temperature (K)	$T_{w,in}$	298.79	1.70	298.77	1.70
4	Atmospheric pressure (Pa)	P_{atm}	100586	401	100576	389
5	Wetted fraction of fill surface area	w_{tsa}	1	0	1	0
6	Sum of loss coefficients above fill	k_{sum}	10	5	10	5
7	Dynamic viscosity of air at T=300 K (kg/m s)	μ	1.983×10^{-5}	9.676E-7	1.984×10^{-5}	9.668E-7

8	Kinematic viscosity of air at T=300 K (m^2/s)	ν	$1.568x10^{-5}$	$1.895\ x10^{-6}$	$1.564\ x10^{-5}$	$1.893\ x10^{-6}$
9	Thermal conductivity of air at T=300 K (W/m K)	k_{air}	0.02624	$1.584\ x10^{-3}$	0.02625	$1.583\ x10^{-3}$
10	Heat transfer coefficient multiplier	f_{ht}	1	0.5	1.0316	0.47
11	Mass transfer coefficient multiplier	f_{mt}	1	0.5	0.882	0.41
12	Fill section frictional loss multiplier	f	4	2	4	2.00
13	P_{vs}(T) parameters	a_0	25.5943	0.01	25.5943	0.01
14		a_1	-5229.89	4.4	-5229.92	4.40
15	C_{pa}(T) parameters	$a_{0,cpa}$	1030.5	0.2940	1030.5	0.294
16		$a_{1,cpa}$	-0.19975	0.0020	-0.19975	0.0020
17		$a_{2,cpa}$	$3.9734x10^{-4}$	$3.345x10^{-6}$	$3.9734x10^{-4}$	$3.345\ x10^{-6}$
18	D_{av}(T) parameters	$a_{0,dav}$	$7.0608x10^{-9}$	0	$7.06085\ x10^{-9}$	0
19		$a_{1,dav}$	2.65322	0.003	2.65322	0.003
20		$a_{2,dav}$	$-6.1681\ x10^{-3}$	$2.3\ x10^{-5}$	$-6.16806\ x10^{-3}$	$2.3\ x10^{-5}$
21		$a_{3,dav}$	$6.552659\ x10^{-6}$	$3.8\ x10^{-8}$	$6.552688\ x10^{-6}$	$3.8\ x\ 10^{-8}$
22	h_f(T) parameters	a_{0f}	-1143423.8	543	-1143423.7	543
23		a_{1f}	4186.50768	1.8	4186.50818	1.8

24	$h_g(T)$ parameters	a_{0g}	2005743.99	1046	2005743.80	1046
25		a_{1g}	1815.437	3.5	1815.436	3.5
26	Nu parameters	$a_{0,Nu}$	8.235	2.059	8.235	2.059
27		$a_{1,Nu}$	0.00314987	0.001	0.0030475	0.001
28		$a_{2,Nu}$	0.9902987	0.327	0.987827	0.327
29		$a_{3,Nu}$	0.023	0.0088	0.023	0.088
30	Cooling tower deck width in x-dir. (m)	W_{dkx}	8.5	0.085	8.5	0.085
31	Cooling tower deck width in y-dir. (m)	W_{dky}	8.5	0.085	8.5	0.085
32	Cooling tower deck height above ground (m)	Δz_{dk}	10	0.1	10	0.1
33	Fan shroud height (m)	Δz_{fan}	3.0	0.03	3.0	0.03
34	Fan shroud inner diameter (m)	D_{fan}	4.1	0.041	4.1	0.041
35	Fill section height (m)	Δz_{fill}	2.013	0.02013	2.013	0.02013
36	Rain section height (m)	Δz_{rain}	1.633	0.01633	1.633	0.01633
37	Basin section height (m)	Δz_{bs}	1.168	0.01168	1.168	0.01168
38	Drift eliminator thickness (m)	Δz_{de}	0.1524	0.001524	0.1524	0.001524

39	Fill section equivalent diameter (m)	D_h	0.0381	0.000381	0.0381	0.000381
40	Fill section flow area (m^2)	A_{fill}	67.29	6.729	67.507	6.705
41	Fill section surface area (m^2)	A_{surf}	14221	3555.3	13914	3463
42	Prandtl number of air at T=80 C	P_r	0.708	0.005	0.708	0.005
43	Wind speed (m/s)	V_w	1.80	0.92	1.80	0.92
44	Exit air speed at the shroud (m/s)	V_{exit}	10.0	1.0	9.978	1.0
i	**Boundary Param.**	**Symbol**	**Original Nominal Value**	**Absolute Std. Dev.**	**Best-estimated Nominal Value**	**Best-estimated Absolute Std. Dev.**
45	Inlet water mass flow rate (kg/s)	$m_{w,in}$	44.02	2.201	44.05	2.199
46	Inlet air temperature (K)	$T_{a,in}$;	299.11	4.17	300.14	2.64
47	Inlet air mass flow rate (kg/s)	m_a	155.07	15.91	154.70	15.87
48	Inlet air humidity ratio	ω_{in}	0.0138	0.00206	0.0142	0.00137

i	**Special Dependent Parameter**	**Symbol**	**Original Nominal Value**	**Absolute Std. Dev.**	**Best-estimated Nominal Value**	**Best-estimated Absolute Std. Dev.**
49	Reynold's number	Re_d	4428	671.6	4395	666.1
50	Schmidt number	Sc	0.60	0.074	0.5986	0.0739
51	Sherwood number	Sh	14.13	4.84	13.35	4.44
52	Nusselt number	Nu	14.94	5.08	14.34	4.83

Using the a priori matrices given in the a priori matrices given in Eqs.(5.73), (5.75) and (5.76) together with the sensitivities presented in Tables 5.2 through 5.5 in Eq.(4.19) yields the following predicted response covariance matrix, $\mathbf{C}_{rr}^{pred}$:

$$\mathbf{C}_{rr}^{pred} \triangleq Cov\left(\left[T_a^{(1)}\right]^{be},\left[T_w^{(50)}\right]^{be},\left[RH^{(1)}\right]^{be}\right)=\begin{pmatrix} 6.71 & 2.73 & -22.80 \\ 2.73 & 2.37 & -1.79 \\ -22.80 & -1.79 & 145.19 \end{pmatrix}. \tag{5.78}$$

The best-estimate response-parameter correlation matrix, $\mathbf{C}_{\alpha r}^{pred}$, is obtained using Eq.(4.20) together with the a priori matrices given in Eqs.(5.73), (5.75) and (5.76) together with the sensitivities presented in Tables 5.2 through 5.5. The non-zero elements with the largest magnitudes are as follows:

$$\begin{aligned} & rel.\,cor.(R_1,\alpha_4)=-0.278;\;\; rel.\,cor.(R_1,\alpha_{41})=-0.070; \\ & rel.\,cor.(R_1,\alpha_{49})=-0.039; \\ & rel.\,cor.(R_2,\alpha_4)=-0.108;\;\; rel.\,cor.(R_2,\alpha_{41})=-0.019; \\ & rel.\,cor.(R_3,\alpha_4)=0.232;\;\;\;\; rel.\,cor.(R_3,\alpha_{41})=0.127; \\ & rel.\,cor.(R_3,\alpha_{49})=0.072. \end{aligned} \tag{5.79}$$

The notation used in Eq. (5.79) is as follows: $R_1 \triangleq T_a^{(1)}, R_2 \triangleq T_w^{(50)}$, $R_3 \triangleq RH^{(1)}$; $\alpha_4 \triangleq P_{atm}$, $\alpha_{41} \triangleq A_{surf}$, and $\alpha_{49} \triangleq \mathrm{Re}_d$.

The best-estimate nominal values of the (model responses) outlet air temperature, $T_a^{(1)}$; outlet water temperature $T_w^{(50)}$; and outlet air relative humidity, $RH^{(1)}$, have been computed using Eq.(4.17) together with the a priori matrices given in Eqs.(5.73), (5.75) and (5.76) together with the sensitivities presented in Tables 5.2 through 5.5. The resulting best-estimate predicted nominal values are summarized in Table 5.16. To facilitate comparison, the corresponding measured and computed nominal values are also presented in this table. Note that there are no direct measurements for the outlet water flow rate, $m_w^{(50)}$. For this response, therefore, the predicted best-estimate nominal value has been obtained by a forward re-computation using the best-estimate nominal parameter values listed in Table 5.15, while the predicted best estimate standard deviation for this response has been computed by using "best-estimate" values in Eq.(4.22), to obtain:

$$\left[\mathbf{C}_{rr}^{comp}\right]^{be} = \left[\mathbf{S}_{r\alpha}\right]^{be}\left[\mathbf{C}_{\alpha\alpha}\right]^{be}\left[\mathbf{S}_{r\alpha}^{\dagger}\right]^{be}. \tag{5.80}$$

Table 5.16. Computed, measured, and optimal best-estimate nominal values and standard deviations for the outlet air temperature, outlet water temperature, outlet air relative humidity, and outlet water flow rate responses.

Nominal Values and Standard Deviations	$T_a^{(1)}$ **[K]**	$T_w^{(50)}$ **[K]**	$RH^{(1)}$ **[%]**	$m_w^{(50)}$ **[kg/s]**
Measured				
nominal value	298.34	295.68	81.98	---
standard deviation	±3.36	±1.59	±15.89	---
Computed				
nominal value	297.46	294.58	86.12	43.60
standard deviation	±3.30	±2.78	±14.90	±2.21
Best-estimate				
nominal value	298.45	295.67	82.12	43.67
standard deviation	±2.59	±1.54	±12.05	±2.20

The results presented in Table 5.16 indicate that the predicted standard deviations are smaller than either the computed or the experimentally measured ones. This is indeed the consequence of using the PM-CMPS methodology in conjunction with consistent (as opposed to discrepant) computational and experimental information. Often, however, the information is inconsistent, usually due to the presence of unrecognized errors. Solutions for addressing such situations have been proposed by Cacuci and Ionescu-Bujor (2010b). It is also important to note that the PM-CMPS methodology has improved (i.e., reduced, albeit not by a significant amount) the predicted standard deviation for the outlet water flow rate response, for which no measurements were available.

As mentioned in the foregoing, measurements are available only for the three outlet responses: $T_a^{(1)}, T_w^{(50)}$ and $RH^{(1)}$. Otherwise, there are no direct measurements for the internal responses along the height of the fill section, namely: (i) the air temperature, $T_a^{(i)}, i = 2,...,I$, at the exit of each control volume; (ii) the water temperature, $T_w^{(i+1)}, i = 1,...,I-1$, at the exit of each control volume; and (iii) the air relative humidity, $RH^{(i)}, i = 2,...,I$, at the exit of each control volume. For these responses, therefore, the predicted best-estimate nominal value has been obtained by a forward re-computation using the best-estimate nominal parameter values, $\boldsymbol{\alpha}^{pred}$, as listed in Table 5.15. Furthermore, the predicted best estimate standard deviation for these responses have been obtained by using "best-estimate" values in Eq.(5.80), in which the matrix of sensitivities $\left[\mathbf{S}_{r\alpha}\right]^{pred}$ has been obtained for each of the responses $T_a^{(i)}, i = 2,...,I$, $T_w^{(i+1)}, i = 1,...,I-1$, and $RH^{(i)}, i = 2,...,I$ by performing adjoint sensitivity computations using the best-estimate parameter values, rather than at the nominal parameter values. The resulting best-estimate nominal parameter values and standard deviations for these responses are plotted in Figs. 5.17 through 5.19, which depict the computed (black), best-estimate (red), and re-computed (green) nominal values and standard deviations for the air temperature $Ta^{(i)}, (i = 1,...,49)$; water temperature $Tw^{(i)}, (i = 2,...,50)$; and air humidity $RH^{(i)}, (i = 1,...,49)$, respectively, along the height of the fill section of the cooling tower.

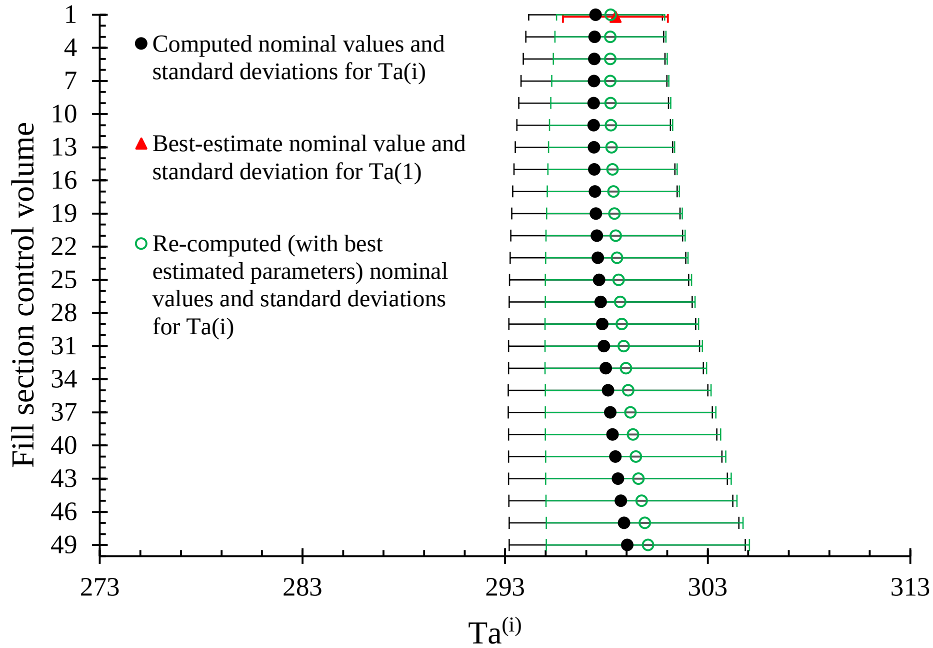

Figure 5.17. Computed (black), best-estimate (red), and re-computed (green; using best-estimate parameter values) nominal values and standard deviations for the air temperature, $Ta^{(i)}$, $(i = 1,...,49)$, at the exit of each control volume along the height of the fill section of the cooling tower.

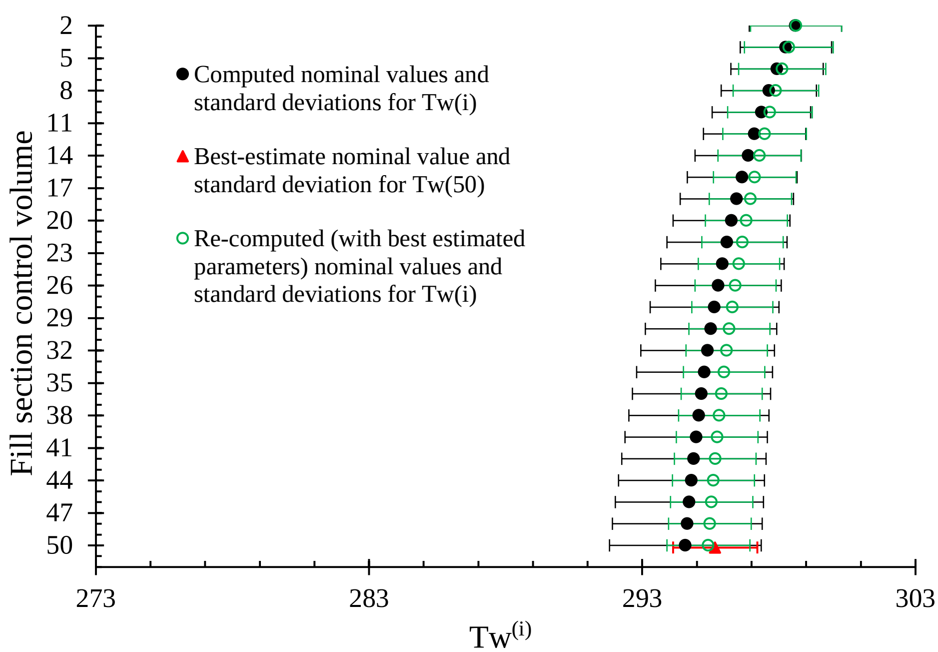

Figure 5.18. Computed (black), best-estimate (red), and re-computed (green; using best-estimate parameter values) nominal values and standard deviations for the water temperature, $Tw^{(i)}$,

$(i = 2,...,50)$, at the exit of each control volume along the height of the fill section of the cooling tower.

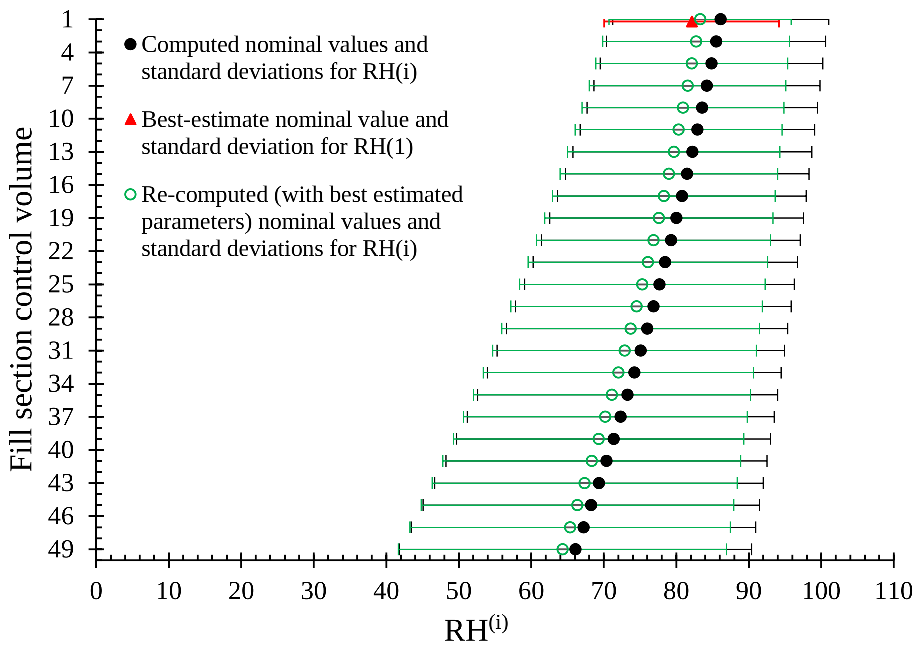

Figure 5.19. Computed (black), best-estimate (red), and re-computed (green; using best-estimate parameter values) nominal values and standard deviations for the air relative humidity, $RH^{(i)}$, $(i = 1,...,49)$, at the exit of each control volume along the height of the fill section of the cooling tower.

The following major conclusions can be drawn from the results presented in this Section:

(i) The results presented in Table 5.16 indicate that the standard deviations predicted by the PM-CMPS are smaller than either the computed or the experimentally measured ones at the locations where measurements are available.

(ii) The results presented in Figs. 5.17 through 5.19 indicate that the PM-CMPS methodology has also improved the predicted standard deviations for the responses inside and along the height of the fill section at locations, for which no measurements were available. As Figs. 5.17 through 5.19 indicate, the PM-CMPS methodology has reduced the uncertainties of the predicted internal responses well below the uncertainties in the computed responses due to uncertainties in the model parameters.

(iii) As depicted in Figs. 5.17 through 5.19, the maximum reductions of uncertainties are always at the boundaries where direct measurements are available, and the amount of reductions decreases toward the inlets along the height of the fill section. For instance, as shown in Fig. 5.17, a maximum of 19% reduction of the uncertainty is achieved for the response $T_a^{(1)}$ at the air exit of the fill section, and this reduction gradually decreases to 14% for the response $T_a^{(49)}$ near the air inlet of the fill section. Similarly, in Fig. 5.18, the maximum reduction of the uncertainty is around 45%, for the response $T_w^{(50)}$ at the water exit of the fill section, and this reduction gradually diminishes to nearly 1% for the response $T_w^{(2)}$ near the water inlet of the fill section. Lastly, for the humidity responses shown in Fig. 5, a maximum of 16% reduction is achieved for the response $RH^{(1)}$ at the air exit of the fill section; this reduction gradually diminishes to around 7% for the response $RH^{(49)}$ near the inlet of the fill section.

Figures 5.17 through 5.19 also indicate that for the internal responses that have no measurements, the assimilation of available experimental information at the boundaries by the PM-CMPS methodology also reduces the predicted uncertainties to be significantly smaller than their computed ones. The maximum reductions of uncertainties occurs at the locations where direct measurements are available (the tower's outlet, in the case considered in this work) and the amount of reductions gradually decrease further away from the locations of the measurements (toward the inlets along the height of the fill section, in the case considered in this work).

6 REFERENCES

Aleman, S.E. and Garrett, A.J. 2015. Operational Cooling Tower Model (CTTool v1.0), SRNL-STI-2015-00039, Revision 0, Savannah River National Laboratory, Savannah River, SC, USA, January 2015.

Arslan E., and D. G. Cacuci, 2014. Predictive Modeling of Liquid-Sodium Thermal-Hydraulics Experiments and Computations, *Ann. Nucl. Energy*, **63C**, 355-370, 2014.

Badea, M. C., D.G. Cacuci, and A.F. Badea, 2012. Best-Estimate Predictions and Model Calibration for Reactor Thermal-Hydraulics", *Nucl. Sci. Eng*., **172**, 1-19, 2012.

Bledsoe, K.C. J. A. Favorite, and T. Aldemir. 2011. Application of the Differential Evolution Method for Solving Inverse Transport Problems, *Nucl. Sci. Eng.*, **169**, 208 (2011).

Cacuci, D. G. 1981a. Sensitivity Theory for Nonlinear Systems: I. Nonlinear Functional Analysis Approach, *J. Math. Phys*. 22: 2794-2802.

Cacuci, D. G. 1981b. Sensitivity Theory for Nonlinear Systems: II. Extensions to additional classes of responses, *J. Math. Phys.* 22: 2803-2812.

Cacuci, D. G. 1988. The forward and the adjoint methods of sensitivity analysis. Chapter 3 in *Uncertainty Analysis*, ed. Y. Ronen, 71-144. Boca Raton: CRC Press, Inc.

Cacuci, D. G. 2003. Sensitivity and Uncertainty Analysis: Theory, **Volume 1.** Boca Raton: Chapman & Hall/CRC.

Cacuci, D. G. 2014. Predictive modeling of coupled multi-physics systems: I. Theory. *Annals of Nuclear Energy* 70: 266–278. See also: Cacuci, D.G. and Badea, M.C. 2014. Predictive modelling of coupled multi-physics systems: II. Illustrative application to reactor physics, *Annals of Nuclear Energy*, **70**, 279-291, 2014.

Cacuci, D. G. 2015a. Second-order adjoint sensitivity analysis methodology (2^{nd}-ASAM) for computing exactly and efficiently first- and second-order sensitivities in large-scale linear systems: I. Computational methodology. *J. Comp. Phys.* 284: 687–699.

Cacuci, D. G. 2015b. Second-order adjoint sensitivity analysis methodology (2^{nd}-ASAM) for computing exactly and efficiently first- and second-order sensitivities in large-scale linear systems: II. Illustrative application to a paradigm particle diffusion problem. *J. Comp. Phys.* 284: 700–717.

Cacuci, D. G. 2016a. Second-order adjoint sensitivity and uncertainty analysis of a benchmark heat transport problem: I. Analytical results. *Nucl. Sci. Eng.* 183: 1-21.

Cacuci, D. G. 2016b. Second-order adjoint sensitivity analysis methodology (2^{nd}-ASAM) for large-scale nonlinear systems: I. Theory," *Nucl. Sci. Eng*. 184: 16–30.

Cacuci, D. G. 2016c. Second-order adjoint sensitivity analysis methodology (2^{nd}-ASAM) for large-scale nonlinear systems: II. Illustrative application to a paradigm nonlinear heat conduction benchmark. *Nucl. Sci. Eng*. 184: 31-52.

Cacuci, D. G. 2016d. A Heat Transport Benchmark Problem for Predicting the Impact of Measurements on Thermal-Hydraulics Experimental Facility Design. *Nucl. Eng. and Design*, 300, 12–27.

Cacuci, D. G. 2016e. Second-Order Adjoint Sensitivity and Uncertainty Analysis of a Benchmark Heat Transport Problem: I. Analytical Results, *Nucl. Sci. Eng.*, 183, 1-21. DOI 10.13182/NSE15-80.

Cacuci, D. G. 2017. Inverse predictive modeling of radiation transport through optically thick media in the presence of counting uncertainties, *Nucl. Sci. Eng*, **186**: 199–223, http://dx.doi.org/10.1080/00295639.2017.1305244, 20 May 2017.

Cacuci D. G., and E. Arslan. 2014. Reducing Uncertainties via Predictive Modeling: FLICA4 Calibration Using BFBT Benchmarks, *Nucl. Sci. Eng*., **176**, 339–349, 2014.

Cacuci, D. G. and Fang, R. 2016. Predictive Modelling of a Paradigm Mechanical Cooling Tower. I: Adjoint Sensitivity Model, *Energies*, **9**, 718 (2016).

Cacuci D.G., and M. Ionescu-Bujor. 2010a. *Sensitivity and Uncertainty Analysis, Data Assimilation and Predictive Best-Estimate Model Calibration*, Chapter 17 in Vol.3, pp 1913 – 2051, *Handbook of Nuclear Engineering*, D. G. Cacuci, Editor, ISBN: 978-0-387-98150-5, Springer New York / Berlin, 2010. See also: D. G. Cacuci and M. Ionescu-Bujor, Model calibration and best-estimate prediction through experimental data assimilation: I. Mathematical framework, *Nucl. Sci. Eng.*, **165** (2010) 18-44.

Cacuci D.G., and M. Ionescu-Bujor. 2010b. On the Evaluation of Discrepant Scientific Data with Unrecognized Errors, *Nucl. Sci. Eng.*, **165**, 1-17, 2010.

Cacuci, D. G., M. Ionescu-Bujor and M. I. Navon. 2005. *Sensitivity and Uncertainty Analysis: Applications to Large Scale Systems*, **Volume 2**. Boca Raton: Chapman & Hall/CRC.

Cacuci, D. G., M. I. Navon and M. Ionescu-Bujor. 2013. *Computational Methods for Data Evaluation and Assimilation*. Boca Raton: Chapman & Hall/CRC.

D.G. Cacuci, C.F. Weber, E.M. Oblow, and J.H. Marable. 1980. Sensitivity Theory for General Systems of Nonlinear Equations," *Nucl. Sci. Eng.*, **75**, 88-110 (1980).See also: Cacuci, D. G., E. Greenspan, J. H. Marable, M. L. Williams. 1980. Developments in sensitivity theory, in: *ANS Topical Conference "1980 Advances in Reactor Physics and Shielding*, 692–704. NAS/70048, 14–17 September 1980. Sun Valley, Idaho.

Fang, R., D. G. Cacuci and M. C. Badea. 2016. Predictive Modelling of a Paradigm Mechanical Cooling Tower. II: Optimal Best-Estimate Predictions with Reduced Uncertainties, *Energies*, **9**, 747 (2016).

Lahoz, W., Khattatov, B., Ménard, R. (Editors), 2010. *Data Assimilation: Making Sense of Observations*, Springer Verlag, Berlin.

Mattingly, J. K. 2015. Private Communication, North Carolina State University, 2015.

McCormick, N.J. 1992. Inverse Radiative Transfer Problems: A Review, *Nucl. Sci. Eng.*, **112**, 185 (1992).

Oppe, T. C., W. D. Joubert, and D. R. Kincaid. 1988. A Package for Solving Large Sparse Linear Systems by Various Iterative Methods", NSPCG User's Guide, Version 1.0. Center for Numerical Analysis, the University of Texas at Austin, April 1988.

Saad, Y. and Schultz, M.H. 1986. GMRES: A Generalized Minimal Residual Algorithm for Solving Nonsymmetric Linear Systems, *SIAM J. Sci. Stat. Comp 7*, No. 3, 856-869, 1986.

Sanchez, R. and N.J. McCormick, 2008. On the Uniqueness on the Inverse Source Problem for Linear Particle Transport Theory, TTSP, **37,** 236 (2008).

Tarantola, A. 2015. *Inverse Problem Theory and Methods for Model Parameter Estimation*, Society for Industrial and Applied Mathematics, Philadelphia, (2005).

Tichonov, A. N. 1963. Regularization of Non-Linear Ill-Posed Problems, *Doklady Akademii Nauk*, **49**(4), 1963. See also: Tichonov, "Solution of Incorrectly Formulated Problems and the Regularization Method", *Soviet Math. Doklady*, **4**, 1035 (1963).

7 MULTI-PRED CODE MODULE

The equations expressing the results of the PM-CMPS methodology developed by Cacuci (2014), namely Eqs. (2.58) through (2.82), which underlie the general case of “two multi-physics models, as well as Eqs. (2.89) through (2.135), which underlie particular situations, have been programed in the computational software module **MULTI-PRED.** All routines in **MULTI-PRED** are written in Fortran 90 and are compatible with most Linux systems, performing predictive modelling computations for the following four cases:

CASE 1: "*One Multi-Physics Model*": predictive modeling solely for Model A with N_a model parameters and N_r measured responses.

CASE 2: "*One Multi-Physics Model with Additional Model Parameters*”: predictive modeling for Model A with N_b additional model parameters, but no additional responses.

CASE 3: "*One Multi-Physics Model with Additional Model Responses*”: predictive modeling for Model A with N_q additional responses, but no additional parameters.

CASE 4: "*Two Multi-Physics Models*”: predictive modelling for Model A coupled with Model B.

7.1 Directories

The computational software module **MULTI-PRED** comprises the following directories:

(1) multi-pred/source/

This folder contains the source codes.

(2) multi-pred/examples/

This folder contains 5 examples specified in the following subfolders.

(i) **../Neutron_Diffusion_Model_Case_1/**

This folder contains the input/output files for Multi-Pred Case 1 for the neutron diffusion model presented in Chapter 3.

(ii) **../Cooling_Tower_Model_Case_1/**

This folder contains the input/output files for Multi-Pred Case 1 for the cooling tower model presented in Chapter 5.

(iii) **../Cooling_Tower_Model_Case_2/**

This folder contains the input/output files for Multi-Pred Case 2 for the cooling tower model presented in Chapter 5.

(iv) **../Cooling_Tower_Model_Case_3/**

This folder contains the input/output files for Multi-Pred Case 3 for the cooling tower model presented in Chapter 5.

(v) **../Cooling_Tower_Model_Case_4/**

This folder contains the input/output files for Multi-Pred Case 4 for the cooling tower model presented in Chapter 5.

(3) multi-pred/matrix_positive_definite_test/

This folder contains the source code for a stand-alone program used to test if a *symmetric* matrix is *positive definite* (SPD). Note that the covariance matrices $\mathbf{C}_{aa}(N_a \times N_a)$, $\mathbf{C}_{rr}(N_r \times N_r)$, $\mathbf{C}_{bb}(N_b \times N_b)$ and $\mathbf{C}_{qq}(N_q \times N_q)$ must be SPD matrices. This program computes the Cholesky factorization of the matrix being tested. If it can be factorized, the program returns a flag indicating that the tested matrix is SPD. Running this test stand-alone program is optional, since the Cholesky factorization has also been implemented in **MULTI-PRED**.

Also included in this folder is an large-scale matrix used for the SPD test. This matrix is a large symmetric positive definite matrix, with seemingly random sparsity pattern. It has a dimension of 60,000 by 60,000 with 410077 nonzero elements. Refer to the following website http://www.cise.ufl.edu/research/sparse/matrices/Andrews/Andrews for detailed information about this matrix.

7.2 Code Compilation and Execution

(1) Compile the software program in Linux

Enter the ***multi-pred/source/*** directory, and use the command *make*, an executable named *multi-pred* will be generated under the source directory.

The compiler used in the *makefile* is ifort (version 12.1.6 and above). It can also be compiled with gfortran (version 4.47 and above). An example makefile with the gfortran compiler, named *makefile.gfortran,* is also included in the *source* directory.

(2) Run the program

To run the program, copy the executable *multi-pred* into the example directories, then use the command:

./multi-pred superfile.inp

where the argument *superfile.inp* contains all the input/output files names. Output files will be generated in the respective example folders.

7.3 Input and Output File Organization

This Section describes the input and output files within the **MULTI-PRED** module.

7.3.1 Super File

The **MULTI-PRED** super-file is a text file that contains the names of input/output files and organizes the individual files for input and output operations. This super-file is read from the command line (UNIT=5) as an argument. The first line of the super-file is reserved for an identifier card, “MultiPredSup”. After the identifier line, each subsequent line is preceded by a category code and a filename. The category code and filename have to be enclosed in single quotes. The filenames can be changed by the user. The second line of the super-file is also reserved for the “dims” category; the corresponding input file defines the dimensions of the matrices and vectors used in **MULTI-PRED**. The lines after the second line are for data files. There are no restrictions regarding the order of the data files and their corresponding categories. Tables 7.1 through 7.4

show the format and complete list of super files for the **MULTI-PRED** Case 1, Case 2, Case 3 and Case 4, respectively.

Table 7.1. Super File Format for Multi-Pred Case 1

Category	File Name
MultiPredSup	
'dims'	'dimensions.inp'
'a_nom'	'a.inp'
'r_mea'	'rm.inp'
'r_com'	'rc.inp'
'C_aa'	'Caa.inp'
'C_ar'	'Car.inp'
'C_rr'	'Crr.inp'
'S_ra'	'Sra.inp'
'a_BE'	'aBE.out'
'r_BE'	'rBE.out'
'C_aaBE'	'CaaBE.out'
'C_rrBE'	'CrrBE.out'
'C_arBE'	'CarBE.out'
'Crr_comp'	''Crrcomp.out'
'chi2'	'chi2.out'

Table 7.2. Super File Format for Multi-Pred Case 2

Category	File Name
MultiPredSup	
'dims'	'dimensions.inp'
'a_nom'	'a.inp'
'r_mea'	'rm.inp'
'r_com'	'rc.inp'
'C_aa'	'Caa.inp'
'C_ar'	'Car.inp'
'C_rr'	'Crr.inp'
'S_ra'	'Sra.inp'
'b_nom'	'b.inp'

'C_bb'	'Cbb.inp'
'C_ab'	'Cab.inp'
'C_br'	'Cbr.inp'
'S_rb'	'Srb.inp'
'a_BE'	'aBE.out'
'r_BE'	'rBE.out'
'C_aaBE'	'CaaBE.out'
'C_rrBE'	'CrrBE.out'
'C_arBE'	'CarBE.out'
'Crr_comp'	"Crrcomp.out'
'b_BE'	'bBE.out'
'C_bbBE'	'CbbBE.out'
'C_abBE'	'CabBE.out'
'C_brBE'	'CbrBE.out'
'chi2'	'chi2.out'

Table 7.3. Super File Format for Multi-Pred Case 3

Category	**File Name**
MultiPredSup	
'dims'	'dimensions.inp'
'a_nom'	'a.inp'
'r_mea'	'rm.inp'
'r_com'	'rc.inp'
'C_aa'	'Caa.inp'
'C_ar'	'Car.inp'
'C_rr'	'Crr.inp'
'S_ra'	'Sra.inp'
'q_mea'	'qm.inp'
'q_com'	'qc.inp'
'C_qq'	'Cqq.inp'
'C_aq'	'Caq.inp'
'S_qa'	'Sqa.inp'
'a_BE'	'aBE.out'
'r_BE'	'rBE.out'

'C_aaBE'	'CaaBE.out'
'C_rrBE'	'CrrBE.out'
'C_arBE'	'CarBE.out'
'Crr_comp'	''Crrcomp.out'
'q_BE'	'qBE.out'
'C_qqBE'	'CqqBE.out'
'Cqq_comp'	''Cqqcomp.out'
'C_aqBE'	'CaqBE.out'
'C_rqBE'	'CrqBE.out'
'Crq_comp'	''Crqcomp.out'
'chi2'	'chi2.out'

Table 7.4. Super File Format for Multi-Pred Case 4

Category	**File Name**
MultiPredSup	
'dims'	'dimensions.inp'
'a_nom'	'a.inp'
'r_mea'	'rm.inp'
'r_com'	'rc.inp'
'C_aa'	'Caa.inp'
'C_ar'	'Car.inp'
'C_rr'	'Crr.inp'
'S_ra'	'Sra.inp'
'b_nom'	'b.inp'
'q_mea'	'qm.inp'
'q_com'	'qc.inp'
'C_bb'	'Cbb.inp'
'C_bq'	'Cbq.inp'
'C_qq'	'Cqq.inp'
'S_qb'	'Sqb.inp'
'C_ab'	'Cab.inp'
'C_aq'	'Caq.inp'
'C_br'	'Cbr.inp'
'C_rq'	'Crq.inp'
'S_rb'	'Srb.inp'

'S_qa'	'Sqa.inp'
'a_BE'	'aBE.out'
'r_BE'	'rBE.out'
'C_aaBE'	'CaaBE.out'
'C_rrBE'	'CrrBE.out'
'C_arBE'	'CarBE.out'
'Crr_comp'	''Crrcomp.out'
'b_BE'	'bBE.out'
'q_BE'	'qBE.out'
'C_bbBE'	'CbbBE.out'
'C_qqBE'	'CqqBE.out'
'C_bqBE'	'CbqBE.out'
'Cqq_comp'	''Cqqcomp.out'
'C_abBE'	'CabBE.out'
'C_aqBE'	'CaqBE.out'
'C_brBE'	'CbrBE.out'
'C_rqBE'	'CrqBE.out'
'Crq_comp'	''Crqcomp.out'
'chi2'	'chi2.out'

7.3.2 File "dimensions.inp"

The file *dimensions.inp* defines the following important control variables:

CaseNumber – Multi-Pred Case selection;

N_a : number of parameters for Model A;

N_r : number of responses for Model A;

N_b : number of additional parameters for Model A (Case 2) or the number of parameters of Model B (Case 4);

N_q : number of additional responses for Model A (Case 3) or the number of responses of Model B (Case 4);

The following is an example of *dimensions.inp* for the Cooling Tower Model Case 4. For this test case, Cooling Tower Model is separated into Model A and Model B. Model A comprises the first 42 parameters (of the total 52 model parameters) and the first 2 responses (of the total 3 model responses). Thus: for Model A, Na = 42 and Nr = 2. Model B comprises the last 10 parameters (of the total 52 model parameters) and the 3rd response (of the total 3 model responses of the Cooling Tower Model). Thus: for Model B, Nb = 10 and Nq =1.

```
/ Case options:
/      = 1 "One-Model" Case: predictive modeling solely for Model A with Na
/                             model parameters and Nr measured responses;
/      = 2 "One-Model" Case: predictive modeling for Model A with Nb additional
/                            parameters, but no additional responses;
/      = 3 "One-Model" Case: predictive modeling for Model A with Nq additional
/                            responses, but no additional parameters;
/      = 4 "Two-Model" Case: predictive modeling for Model A coupled with Model B.
/ Case selection (CaseNumber):
4
/Na -- number of parameters for model A
42
/Nr -- number of responses for model A
2
/Nb -- number of additional parameters:
/       -- for case 1: not used
/       -- for case 2: number of parameters added to the Na parameters for model A
/       -- for case 3: not used
/       -- for case 4: number of parameters of model B
10
/Nq -- number of additional responses:
/       -- for case 1: not used
/       -- for case 2: not used
/       -- for case 3: number of responses added to the Nr responses for model A
/       -- for case 4: number of responses for model B
1
```

The format of *dimensions.inp* is fixed as shown above. The user can change the numbers corresponding to the control variables, namely: CaseNumber, Na, Nr, Nb and Nq, respectively.

7.3.3 Contents and Organization of Input and Output Files

Tables 7.5 through 7.8 describe the contents of the input and output (I/O) files specified within the **MULTI-PRED** super-files listed in Tables 7.1 through 7.4, respectively. The vectors / matrices corresponding to each data file are also listed in Tables 7.5 through 7.8.

Table 7.5. Summary of Input and Output Files for MULTI-PRED Case 1

File	Unit	I/O	Corresponding vector/matrix	Descriptions
superfile.inp	5	input		File organization
dimensions.inp	20	input		Defines the Case selection and dimensions control
a.inp	21	input	$\boldsymbol{\alpha}(N_a)$	Nominal values of Na parameters of model A
rm.inp	22	input	$\mathbf{r}_m(N_r)$	Nominal values of Nr measured responses of model A
rc.inp	23	input	$\mathbf{r}_c(N_r)$	Nominal values of Nr computed responses of model A
Caa.inp	24	input	$\mathbf{C}_{aa}(N_a \times N_a)$	Covariance matrix of Na parameters of model A
Car.inp	25	input	$\mathbf{C}_{ar}(N_a \times N_r)$	Correlations between Na parameters and Nr responses of Model A
Crr.inp	26	input	$\mathbf{C}_{rr}(N_r \times N_r)$	Covariance matrix of Nr responses of model A
Sra.inp	27	input	$\mathbf{S}_{ra}(N_r \times N_a)$	Absolute sensitivities of Nr responses of Model A w.r.t Na parameters of Model A
aBE.out	51	output	$\boldsymbol{\alpha}^{be}(N_a)$	Best-estimate nominal values of parameters of Model A
rBE.out	52	output	$\mathbf{r}^{be}(N_r)$	Best-estimate nominal values of responses of Model A
CaaBE.out	53	output	$\mathbf{C}_{aa}^{be}(N_a \times N_a)$	Predicted covariance matrix of Na parameters of Model A
CrrBE.out	54	output	$\mathbf{C}_{rr}^{be}(N_r \times N_r)$	Predicted covariance matrix of Nr responses of Model A
CarBE.out	55	output	$\mathbf{C}_{ar}^{be}(N_a \times N_r)$	Predicted correlation matrix between the Na parameters and Nr responses of model A
Crrcomp.out	56	output	$\mathbf{C}_{rr}^{comp}(N_r \times N_r)$	Covariance matrix of Nr computed responses of model A
chi2.out	76	output	χ^2, scalar	Value of the consistency indicator

Table 7.6. Summary of Input and Output Files for MULTI-PRED Case 2

File	Unit	I/O	Corresponding vector/matrix	Descriptions
superfile.inp	5	input		File organization
dimensions.inp	20	input		Defines the Case selection and dimensions control
a.inp	21	Input	$\boldsymbol{\alpha}(N_a)$	Nominal values of Na parameters of model A
rm.inp	22	Input	$\mathbf{r}_m(N_r)$	Nominal values of Nr measured responses of model A
rc.inp	23	Input	$\mathbf{r}_c(N_r)$	Nominal values of Nr computed responses of model A
Caa.inp	24	Input	$\mathbf{C}_{aa}(N_a \times N_a)$	Covariance matrix of Na parameters of model A
Car.inp	25	Input	$\mathbf{C}_{ar}(N_a \times N_r)$	Correlations between Na parameters and Nr responses of Model A

File	Unit	I/O	Corresponding vector/matrix	Descriptions
Crr.inp	26	Input	$\mathbf{C}_{rr}(N_r \times N_r)$	Covariance matrix of Nr responses of model A
Sra.inp	27	Input	$\mathbf{S}_{ra}(N_r \times N_a)$	Absolute sensitivities of Nr responses of Model A w.r.t Na parameters of Model A
b.inp	31	Input	$\mathbf{b}(N_b)$	Nominal values of Nb additional parameters for model A
Cbb.inp	34	Input	$\mathbf{C}_{bb}(N_b \times N_b)$	Covariance matrix of Nb additional parameters
Cab.inp	41	Input	$\mathbf{C}_{ab}(N_a \times N_b)$	Correlations between Na parameters of Model A and Nb additional parameters for Model A
Cbr.inp	43	Input	$\mathbf{C}_{br}(N_b \times N_r)$	Correlations between Nb additional parameters for Model A and Nr responses of Model A
Srb.inp	45	input	$\mathbf{S}_{rb}(N_r \times N_b)$	Absolute sensitivities of Nr responses of Model A w.r.t Nb additional parameters for model A
aBE.out	51	output	$\boldsymbol{\alpha}^{be}(N_a)$	Best-estimate nominal values of Na parameters of Model A
rBE.out	52	output	$\mathbf{r}^{be}(N_r)$	Best-estimate nominal values of Nr responses of Model A
CaaBE.out	53	output	$\mathbf{C}_{aa}^{be}(N_a \times N_a)$	Predicted covariance matrix of Na parameters of Model A
CrrBE.out	54	output	$\mathbf{C}_{rr}^{be}(N_r \times N_r)$	Predicted covariance matrix of Nr responses of Model A
CarBE.out	55	output	$\mathbf{C}_{ar}^{be}(N_a \times N_r)$	Predicted correlation matrix between the Na parameters and Nr responses of model A
Crrcomp.out	56	output	$\mathbf{C}_{rr}^{comp}(N_r \times N_r)$	Covariance matrix of Nr computed responses of model A
bBE.out	61	output	$\mathbf{b}^{be}(N_b)$	Best-estimate nominal values of Nb additional parameters
CbbBE.out	63	output	$\mathbf{C}_{bb}^{be}(N_b \times N_b)$	Predicted covariance matrix of Nb parameters of Model A
CabBE.out	71	output	$\mathbf{C}_{ab}^{be}(N_a \times N_b)$	Predicted correlation matrix between the Na parameters of Model A and the Nb additional parameters for Model A
CbrBE.out	73	output	$\mathbf{C}_{br}^{be}(N_b \times N_r)$	Predicted correlation matrix between the Nb additional parameters for Model A and Nr responses of model A
chi2.out	76	output	χ^2, scalar	Value of the consistency indicator

Table 7.7. Summary of Input and Output Files for MULTI-PRED Case 3

File	Unit	I/O	Corresponding vector/matrix	Descriptions
superfile.inp	5	input		File organization
dimensions.inp	20	input		Defines the Case selection and dimensions control
a.inp	21	input	$\boldsymbol{\alpha}(N_a)$	Nominal values of Na parameters of model A
rm.inp	22	input	$\mathbf{r}_m(N_r)$	Nominal values of Nr measured responses of model A

File	Unit	I/O	Corresponding vector/matrix	Descriptions
rc.inp	23	input	$\mathbf{r}_c(N_r)$	Nominal values of Nr computed responses of model A
Caa.inp	24	input	$\mathbf{C}_{aa}(N_a \times N_a)$	Covariance matrix of Na parameters of model A
Car.inp	25	input	$\mathbf{C}_{ar}(N_a \times N_r)$	Correlations between Na parameters and Nr responses of Model A
Crr.inp	26	input	$\mathbf{C}_{rr}(N_r \times N_r)$	Covariance matrix of Nr responses of model A
Sra.inp	27	input	$\mathbf{S}_{ra}(N_r \times N_a)$	Absolute sensitivities of Nr responses of Model A w.r.t Na parameters of Model A
qm.inp	32	input	$\mathbf{q}_m(N_q)$	Nominal values of Nq additional measured responses for model A
qc.inp	33	input	$\mathbf{q}_c(N_q)$	Nominal values of Nq additional computed responses for model A
Cqq.inp	36	input	$\mathbf{C}_{qq}(N_q \times N_q)$	Covariance matrix of Nq additional responses for Model A
Caq.inp	42	input	$\mathbf{C}_{aq}(N_a \times N_q)$	Correlations between Na parameters of Model A and Nq additional responses for Model A
Crq.inp	44	input	$\mathbf{C}_{rq}(N_r \times N_q)$	Correlations between Nr responses of Model A and Nq additional responses for Model A
Sqa.inp	46	input	$\mathbf{S}_{qa}(N_q \times N_a)$	Absolute sensitivities of Nq additional responses for Model A w.r.t Na parameters of Model A
aBE.out	51	output	$\boldsymbol{\alpha}^{be}(N_a)$	Best-estimate nominal values of parameters of Model A
rBE.out	52	output	$\mathbf{r}^{be}(N_r)$	Best-estimate nominal values of responses of Model A
CaaBE.out	53	output	$\mathbf{C}_{aa}^{be}(N_a \times N_a)$	Predicted covariance matrix of Na parameters of Model A
CrrBE.out	54	output	$\mathbf{C}_{rr}^{be}(N_r \times N_r)$	Predicted covariance matrix of Nr responses of Model A
CarBE.out	55	output	$\mathbf{C}_{ar}^{be}(N_a \times N_r)$	Predicted correlation matrix between the Na parameters and Nr responses of model A
Crrcomp.out	56	output	$\mathbf{C}_{rr}^{comp}(N_r \times N_r)$	Covariance matrix of Nr computed responses of model A
qBE.out	62	output	$\mathbf{q}^{be}(N_q)$	Best-estimate nominal values of Nq additional responses for model A
CqqBE.out	64	output	$\mathbf{C}_{qq}^{be}(N_q \times N_q)$	Predicted covariance matrix of Nq additional responses for model A
Cqqcomp.out	66	output	$\mathbf{C}_{qq}^{comp}(N_q \times N_q)$	Covariance matrix of Nq additional computed responses for model A
CaqBE.out	72	output	$\mathbf{C}_{aq}^{comp}(N_a \times N_q)$	Predicted correlation matrix between the Na parameters and of Model A and Nq additional responses for model A
CrqBE.out	74	output	$\mathbf{C}_{rq}^{be}(N_r \times N_q)$	Predicted correlation matrix of between Nr responses of Model A and Nq additional responses for model A
Crqcomp.out	75	output	$\mathbf{C}_{rq}^{comp}(N_r \times N_q)$	Correlation matrix of Nr computed responses of Model A and Nq additional computed responses for model A

File	Unit	I/O	Corresponding vector/matrix	Descriptions
chi2.out	76	output	χ^2, scalar	Value of the consistency indicator

Table 7.8. Summary of Input and Output Files for MULTI-PRED Case 4

File	Unit	I/O	Corresponding vector/matrix	Descriptions
superfile.inp	5	input		File organization
dimensions.inp	20	input		Defines the Case selection and dimensions control
a.inp	21	input	$\boldsymbol{\alpha}(N_a)$	Nominal values of Na parameters of model A
rm.inp	22	input	$\mathbf{r}_m(N_r)$	Nominal values of Nr measured responses of model A
rc.inp	23	input	$\mathbf{r}_c(N_r)$	Nominal values of Nr computed responses of model A
Caa.inp	24	input	$\mathbf{C}_{aa}(N_a \times N_a)$	Covariance matrix of Na parameters of model A
Car.inp	25	input	$\mathbf{C}_{ar}(N_a \times N_r)$	Correlations between Na parameters and Nr responses of Model A
Crr.inp	26	input	$\mathbf{C}_{rr}(N_r \times N_r)$	Covariance matrix of Nr responses of model A
Sra.inp	27	input	$\mathbf{S}_{ra}(N_r \times N_a)$	Absolute sensitivities of Nr responses of Model A w.r.t Na parameters of Model A
b.inp	31	input	$\mathbf{b}(N_b)$	Nominal values of Na parameters of model B
qm.inp	32	input	$\mathbf{q}_m(N_q)$	Nominal values of Nq measured responses of model B
qc.inp	33	input	$\mathbf{q}_c(N_q)$	Nominal values of Nq computed responses of model B
Cbb.inp	34	input	$\mathbf{C}_{bb}(N_b \times N_b)$	Covariance matrix of Nb parameters of model B
Cbq.inp	35	input	$\mathbf{C}_{bq}(N_b \times N_q)$	Correlations between Nb parameters and Nq responses of Model B
Cqq.inp	36	input	$\mathbf{C}_{qq}(N_q \times N_q)$	Covariance matrix of Nq responses of model B
Sqb.inp	37	input	$\mathbf{S}_{qb}(N_q \times N_b)$	Absolute sensitivities of Nq responses of Model B w.r.t Nb parameters of Model B
Cab.inp	41	input	$\mathbf{C}_{ab}(N_a \times N_b)$	Correlation matrix between the Na parameters of Model A and the Nb parameters of Model B
Caq.inp	42	input	$\mathbf{C}_{aq}(N_a \times N_q)$	Correlation matrix between the Na parameters and of Model A and Nq responses of model B
Cbr.inp	43	input	$\mathbf{C}_{br}(N_b \times N_r)$	Correlation matrix between the Nb parameters of Model B and Nr responses of model A
Crq.inp	44	input	$\mathbf{C}_{rq}(N_r \times N_q)$	Correlation matrix of between Nr responses of Model A and Nq responses of model B

File	Unit	I/O	Corresponding vector/matrix	Descriptions
Srb.inp	45	input	$\mathbf{S}_{rb}(N_r \times N_b)$	Absolute sensitivities of Nr responses of Model A w.r.t Nb parameters of model B
Sqa.inp	46	input	$\mathbf{S}_{qa}(N_q \times N_a)$	Absolute sensitivities of Nq responses of Model B w.r.t Na parameters of Model A
aBE.out	51	output	$\boldsymbol{\alpha}^{be}(N_a)$	Best-estimate nominal values of parameters of Model A
rBE.out	52	output	$\mathbf{r}^{be}(N_r)$	Best-estimate nominal values of responses of Model A
CaaBE.out	53	output	$\mathbf{C}_{aa}^{be}(N_a \times N_a)$	Predicted covariance matrix of Na parameters of Model A
CrrBE.out	54	output	$\mathbf{C}_{rr}^{be}(N_r \times N_r)$	Predicted covariance matrix of Nr responses of Model A
CarBE.out	55	output	$\mathbf{C}_{ar}^{be}(N_a \times N_r)$	Predicted correlation matrix between the Na parameters and Nr responses of model A
Crrcomp.out	56	output	$\mathbf{C}_{rr}^{comp}(N_r \times N_r)$	Covariance matrix of Nr computed responses of model A
bBE.out	61	output	$\mathbf{b}^{be}(N_b)$	Best-estimate nominal values of parameters of Model B
qBE.out	62	output	$\mathbf{q}^{be}(N_q)$	Best-estimate nominal values of responses of Model B
CbbBE.out	63	output	$\mathbf{C}_{bb}^{be}(N_b \times N_b)$	Predicted covariance matrix of Nb parameters of Model B
CqqBE.out	64	output	$\mathbf{C}_{qq}^{be}(N_q \times N_q)$	Predicted covariance matrix of Nq responses of Model B
CbqBE.out	65	output	$\mathbf{C}_{bq}^{be}(N_b \times N_q)$	Predicted correlation matrix between the Nb parameters and Nq responses of model B
Cqqcomp.out	66	output	$\mathbf{C}_{qq}^{comp}(N_q \times N_q)$	Covariance matrix of Nq computed responses of model B
CabBE.out	71	output	$\mathbf{C}_{ab}^{be}(N_a \times N_b)$	Predicted correlation matrix between the Na parameters of Model A and the Nb parameters for Model B
CaqBE.out	72	output	$\mathbf{C}_{aq}^{be}(N_a \times N_q)$	Predicted correlation matrix between the Na parameters and of Model A and Nq responses of model B
CbrBE.out	73	output	$\mathbf{C}_{br}^{be}(N_b \times N_r)$	Predicted correlation matrix between the Nb parameters of Model B and Nr responses of model A
CrqBE.out	74	output	$\mathbf{C}_{rq}^{be}(N_r \times N_q)$	Predicted correlation matrix of between Nr responses of Model A and Nq responses of model B
Crqcomp.out	75	output	$\mathbf{C}_{rq}^{comp}(N_r \times N_q)$	Correlation matrix of Nr computed responses of Model A and Nq computed responses of model B
chi2.out	76	output	χ^2, scalar	Value of the consistency indicator

7.4 Input Data Files

This Section describes in detail the *input* files (and their contents) that were listed in Table 7.8. All the data files are in the "sparse triplet matrix" file format, which is a commonly used ASCII file format for storing sparse matrices and compatible with most files in the Matrix Market format.

The *sparse triplet data structure* simply records, for each nonzero entry of the matrix, the row, column and value. The general format is as follows:

Line 1:	**M** **N**	**Nz**	
Line 2:	**Row_index**	**Col_index**	**Val**
Line 3:	**Row_index**	**Col_index**	**Val**
...	...	...	...
Line Nz+1:	**Row_index**	**Col_index**	**Val**

In the above format, the quantities **M** and **N** denote, respectively, the number of rows and columns in the original full matrix; **Nz** denotes total the number of nonzero elements in the matrix; **Row_index** and **Col_index** denote the row and column indices of each nonzero element; and **VAL** denotes the value of the nonzero element.

7.4.1 Input Data Files for MULTI-PRED Case 1

MULTI-PRED Case 1 requires the following 7 input data files as listed in Table 7.9, as well as in Table 7.5.

Table 7.9. Input Data Files for MULTI-PRED Case 1

Input Data File for Model A
a.inp
rm.inp
rc.inp
Caa.inp
Car.inp
Crr.inp
Sra.inp

The file structures for the inputs shown in Table 7.9 are described in detail below.

(1) a.inp

The input file *a.inp* contains the nominal values of all N_a parameters of Model A. For example, for the neutron diffusion model, the nominal values of the $N_a = 4$ parameters are given as follows:

$$\boldsymbol{\alpha} = \begin{pmatrix} 0.0197 \\ 0.16 \\ 1.0E+07 \\ 7.438 \end{pmatrix}. \tag{7.1}$$

The corresponding *a.inp* is as follows.

```
4      1      4
1      1      0.0197
2      1      0.16
3      1      1.0E+07
4      1      7.438
```

(2) rm.inp

The input file *rm.inp* contains the nominal values of N_r measured responses for Model A. T For the neutron diffusion model, for example, the nominal values of the $N_r = 4$ measured responses are as follows:

$$\mathbf{r}_m = \begin{pmatrix} 3.40E+09 \\ 3.59E+09 \\ 3.77E+09 \\ 3.74E+09 \end{pmatrix}. \tag{7.2}$$

The corresponding *rm.inp* is as follows.

```
4      1      4
1      1      3.398068337E+09
2      1      3.586849912E+09
3      1      3.772511377E+09
4      1      3.735885053E+09
```

(3) rc.inp

The input file *rc.inp* contains the nominal values of N_r computed responses of Model A. For the neutron diffusion model, for example, the nominal values of the $N_r = 4$ computed responses are as follows:

$$\mathbf{r}_c = \begin{pmatrix} 3.77\text{E+09} \\ 3.77\text{E+09} \\ 3.66\text{E+09} \\ 3.66\text{E+09} \end{pmatrix}. \tag{7.3}$$

The corresponding *rc.inp* is as follows.

```
4       1      4
1       1      3.775631486E+09
2       1      3.775631486E+09
3       1      3.662632405E+09
4       1      3.662632405E+09
```

(4) Caa.inp

The input file *Caa.inp* contains the nonzero elements of the covariance matrix $\mathbf{C}_{aa}(N_a \times N_a)$ of model parameters of Model A. For the neutron diffusion model, for example, $\mathbf{C}_{aa}$ is:

$$\boldsymbol{C}_{\alpha a} = \begin{pmatrix} \left(9.85\times10^{-4}\right)^2 & 0 & 0 & 0 \\ 0 & \left(8.0\times10^{-3}\right)^2 & 0 & 0 \\ 0 & 0 & \left(1.5\times10^{6}\right)^2 & 0 \\ 0 & 0 & 0 & \left(7.44\times10^{-1}\right)^2 \end{pmatrix}. \tag{7.4}$$

The corresponding *Caa.inp* is as follows.

```
4      4      4
1      1      9.70225E-07
2      2      6.40000E-05
3      3      2.25000E+12
4      4      5.5323844E-01
```

(5) Car.inp

The input file *Car.inp* contains the nonzero elements of the correlation matrix $\mathbf{C}_{ar}(N_a \times N_r)$ between the model parameters and measured responses of Model A. For the neutron diffusion model, for examples, the parameters and measured responses are not correlated; therefore, $\mathbf{C}_{ar}$ has the following structure:

$$\boldsymbol{C}_{\alpha r} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}. \qquad (7.5)$$

The corresponding *Car.inp* is as follows:

```
4      4      0
```

In other applications, the parameters and measured responses are correlated, i.e., $\mathbf{C}_{\alpha r} \neq \mathbf{0}$. An example of a non-zero correlation matrix is provided by the cooling tower model, for which $N_a = 52$, $N_r = 3$, and for which the correlation matrix $\mathbf{C}_{\alpha r}$ comprises 12 nonzero elements. Hence, for this example, *Car.inp* is as follows:

```
52      3      12
1       1      12.957508300000001
1       2      3.3548676099999999
1       3      -54.158679370000002
2       1      3.5102394000000001
2       2      3.0452589900000002
2       3      1.73334787
3       1      2.3294612799999999
3       2      1.8856921
3       3      -2.26657529
4       1      -447.08545706000001
4       2      -93.577718820000001
4       3      1831.03340159
```

(6) Crr.inp

The input file *Crr.inp* contains the nonzero elements of the covariance matrix $\mathbf{C}_{rr}(N_r \times N_r)$ between the model responses of Model A. For the neutron diffusion model, for example, $\mathbf{C}_{rr}$ is

$$\boldsymbol{C}_{rr} = \begin{pmatrix} \left(1.7\times10^{8}\right)^2 & 0 & 0 & 0 \\ 0 & \left(2.15\times10^{8}\right)^2 & 0 & 0 \\ 0 & 0 & \left(1.89\times10^{8}\right)^2 & 0 \\ 0 & 0 & 0 & \left(1.87\times10^{8}\right)^2 \end{pmatrix}. \tag{7.6}$$

The corresponding *Crr.inp* is as follows:

```
4       4      4
1       1      2.886717104E+16
2       2      4.631577224E+16
3       3      3.557960521E+16
4       4      3.489209280E+16
```

(7) Sra.inp

The input file *Sra.inp* contains the nonzero elements of the absolute sensitivities matrix $\mathbf{S}_{ra}(N_r \times N_a)$. For the neutron diffusion model, for example, $\mathbf{S}_{ra}(N_r \times N_a)$ is

$$\boldsymbol{S} \triangleq \left(\frac{\partial R_i}{\partial \alpha_j}\right) = \begin{pmatrix} -1.92\times10^{11} & -1.33\times10^{5} & 3.78\times10^{2} & 5.08\times10^{8} \\ -1.92\times10^{11} & -1.33\times10^{5} & 3.78\times10^{2} & 5.08\times10^{8} \\ -1.76\times10^{11} & -1.24\times10^{9} & 3.66\times10^{2} & 4.92\times10^{8} \\ -1.76\times10^{11} & -1.24\times10^{9} & 3.66\times10^{2} & 4.92\times10^{8} \end{pmatrix}. \tag{7.7}$$

The corresponding *Sra.inp* is as follows:

```
4       4      16
1       1      -1.916553399E+11
1       2      -1.330585230E+5
1       3      3.775631486E+2
1       4      5.076138055E+8
2       1      -1.916553399E+11
2       2      -1.330585230E+5
2       3      3.775631486E+2
2       4      5.076138055E+8
3       1      -1.758565925E+11
```

```
3       2     -1.239109567E+9
3       3     3.662632405E+2
3       4     4.924216731E+8
4       1     -1.758565925E+11
4       2     -1.239109567E+9
4       3     3.662632405E+2
4       4     4.924216731E+8
```

7.4.2 Input Data Files for MULTI-PRED Case 2

Table 7.10 presents the 12 input files required for MULTI-PRED Case 2; these files are also listed in Table 7.6. Of the 12 files listed in Table 7.10, 7 input data files have been previously described in Section 7.4.2; the additional 5 input data files have the same structure as their counterparts for Model A.

Table 7.10. Input Data Files for MULTI-PRED Case 2

Input Data File for Model A	Inputs for the Coupled Matrices	Inputs for the Nb additional parameters for Model A
a.inp		b.inp
rm.inp		
rc.inp		
Caa.inp	Cab.inp	Cbb.inp
Car.inp	Cbr.inp	
Crr.inp		
Sra.inp	Srb.inp	

7.4.3 Input Data Files for MULTI-PRED Case 3

Table 7.11 presents the 13 input files required for MULTI-PRED Case 3; these files are also listed in Table 7.7. Of the 13 files listed in Table 7.10, 7 input data files have been previously described in Section 7.4.2; the additional 6 input data files have the same structure as their counterparts for Model A.

Table 7.11. Input Data Files for MULTI-PRED Case 3

Input Data File for Model A	Inputs for the coupled matrices	Inputs for the Nq additional responses for Model A
a.inp		
rm.inp		qm.inp
rc.inp		qc.inp
Caa.inp		
Car.inp	Caq.inp	
Crr.inp	Crq.inp	Cqq.inp
Sra.inp	Sqa.inp	

7.4.4 Input Data Files for MULTI-PRED Case 4

Table 7.12 presents the 20 input files required for MULTI-PRED Case 3; these files are also listed in Table 7.8. Of the 20 files listed in Table 7.10, 7 input data files have been previously described in Section 7.4.2; the additional 13 input data files have the same structure as their counterparts for Model A.

Table 7.12. Input Data Files for MULTI-PRED Case 4

Input Data File for Model A	Inputs Data Files for the Coupled Matrices between Model A and Model B	Inputs Data Files for Model B
a.inp		b.inp
rm.inp		qm.inp
rc.inp		qc.inp
Caa.inp	Cab.inp	Cbb.inp
Car.inp	Caq.inp, Cbr.inp	Cbq.inp
Crr.inp	Crq.inp	Cqq.inp
Sra.inp	Sqa.inp, Srb.inp	Sqb.inp

7.5 Output Data Files

The model output files are specified in the categories of the super files. All the output files are in the "sparse triplet matrix" file format. In addition, a data file for the consistency indicator, χ^2, is also generated.

7.5.1 Output Data Files for MULTI-PRED Case 1

Table 7.13 lists the output data files generated by MULTI-PRED Case 1; these output files are also listed in Table 7.5.

Table 7.13. Output Data Files for MULTI-PRED Case 1

Output Data File for Model A
aBE.out
rBE.out
CaaBE.out
CrrBE.out
CarBE.out
Crrcomp.out
chi2.out

(1) aBE.out

The output data file *aBE.out* contains the nonzero components of the resulting vector $\boldsymbol{\alpha}^{be}(N_a)$, which provide the best-estimate parameter values for Model A. This file has the same structure as the file *a.inp*. For the neutron diffusion model, for example, the best-estimate parameter values are:

$$\boldsymbol{\alpha}^{be} = \begin{pmatrix} 0.0198 \\ 0.1591 \\ 9.85\times10^{6} \\ 7.388 \end{pmatrix}. \tag{7.8}$$

The corresponding output data file *aBE.out* is as follows:

```
4    1      4
1    1      1.98418101E-02
2    1      1.59118840E-01
3    1      9.84778916E+06
4    1      7.38768248E+00
```

(2) rBE.out

The output file *rBE.out* contains the nonzero components of the vector $\mathbf{r}^{be}(N_r)$, which provide the best-estimate response values for Model A. This output file has the structure as the file *rc.inp*. For the neutron diffusion model, for example, the best-estimate response values are:

$$\mathbf{r}^{be} = \begin{pmatrix} 3.66\times10^{9} \\ 3.66\times10^{9} \\ 3.56\times10^{9} \\ 3.56\times10^{9} \end{pmatrix}. \tag{7.9}$$

The corresponding output data file r*BE.out* is as follows:

```
4    1      4
1    1      3.66544187E+09
2    1      3.66544187E+09
3    1      3.55825935E+09
4    1      3.55825935E+09
```

(3) CaaBE.out

The output file C*aaBE.out* contains the nonzero components of the predicted optimal covariance matrix $\mathbf{C}_{aa}^{be}(N_a \times N_a)$ of parameters for Model A. This output file has the same structure as the file C*aa.inp*. For the neutron diffusion model, for example, the best-estimate covariance matrix $\mathbf{C}_{aa}^{be}(N_a \times N_a)$ has the following form:

$$\boldsymbol{C}_{aa}^{be} = \begin{pmatrix} 9.03\times10^{-7} & 6.75\times10^{-9} & 3.03\times10^{2} & 1.00\times10^{-4} \\ 6.75\times10^{-9} & 6.38\times10^{-5} & 7.37\times10^{1} & 2.44\times10^{-5} \\ 3.03\times10^{2} & 7.37\times10^{1} & 8.24\times10^{11} & -4.71\times10^{5} \\ 1.00\times10^{-4} & 2.44\times10^{-5} & -4.71\times10^{5} & 3.97\times10^{-1} \end{pmatrix} \tag{7.10}$$

The corresponding output data file Caa*BE.out* is as follows.

```
4    4      16
1    1      9.02992937E-07
1    2      6.48311998E-09
1    3      3.03370351E+02
1    4      1.00295979E-04
2    1      6.48311998E-09
2    2      6.38139054E-05
2    3      7.32951010E+01
2    4      2.43980122E-05
3    1      3.03370351E+02
3    2      7.32951010E+01
3    3      8.24405218E+11
3    4     -4.71401727E+05
4    1      1.00295979E-04
4    2      2.43980122E-05
4    3     -4.71401727E+05
4    4      3.97657348E-01
```

(4) CarBE.out

The output file C*arBE.out* contains the nonzero components of the predicted parameter-response correlation matrix $\mathbf{C}_{ar}^{be}(N_a \times N_r)$ for Model A. This output file has the same structure as the file C*ar.inp*. For the neutron diffusion model, for example, the correlation matrix $\mathbf{C}_{ar}^{be}(N_a \times N_r)$ has the following structure:

$$\boldsymbol{C}_{\alpha r}^{be} = \begin{pmatrix} -7.81\times10^{3} & -7.81\times10^{3} & 1.50\times10^{3} & 1.50\times10^{3} \\ 3.89\times10^{4} & 3.89\times10^{4} & -4.13\times10^{4} & -4.13\times10^{4} \\ 1.38\times10^{13} & 1.38\times10^{13} & 1.64\times10^{13} & 1.64\times10^{13} \\ 4.57\times10^{6} & 4.57\times10^{6} & 5.41\times10^{6} & 5.41\times10^{6} \end{pmatrix}. \quad (7.11)$$

The corresponding output data file Car*BE.out* is as follows:

```
4    4      16
1    1     -7.81261058E+03
1    2     -7.81261058E+03
1    3      1.50018202E+03
1    4      1.50018202E+03
2    1      3.88811594E+04
2    2      3.88811594E+04
2    3     -4.12791159E+04
2    4     -4.12791159E+04
3    1      1.38214773E+13
3    2      1.38214773E+13
```

```
3    3      1.63658795E+13
3    4      1.63658795E+13
4    1      4.56907323E+06
4    2      4.56907323E+06
4    3      5.41019607E+06
4    4      5.41019607E+06
```

(5) CrrBE.out

The output file C*rrBE.out* contains the nonzero components of the predicted covariance matrix $\mathbf{C}_{rr}^{be}(N_r \times N_r)$ of responses for Model A. This output file has the same file structure as the file C*rr.inp*. For the neutron diffusion model, for example, the correlation matrix $\mathbf{C}_{rr}^{be}(N_r \times N_r)$ is as follows:

$$\boldsymbol{C}_{rr}^{be} = \begin{pmatrix} 9.04\times10^{15} & 9.04\times10^{15} & 8.64\times10^{15} & 8.64\times10^{15} \\ 9.04\times10^{15} & 9.04\times10^{15} & 8.64\times10^{15} & 8.64\times10^{15} \\ 8.64\times10^{15} & 8.64\times10^{15} & 8.45\times10^{15} & 8.45\times10^{15} \\ 8.64\times10^{15} & 8.64\times10^{15} & 8.45\times10^{15} & 8.45\times10^{15} \end{pmatrix} \tag{7.12}$$

The corresponding output data file *CrrBE.out* is as follows:

```
4    4      16
1    1      9.03512848E+15
1    2      9.03512848E+15
1    3      8.63793079E+15
1    4      8.63793079E+15
2    1      9.03512848E+15
2    2      9.03512848E+15
2    3      8.63793079E+15
2    4      8.63793079E+15
3    1      8.63793079E+15
3    2      8.63793079E+15
3    3      8.44565029E+15
3    4      8.44565029E+15
4    1      8.63793079E+15
4    2      8.63793079E+15
4    3      8.44565029E+15
4    4      8.44565029E+15
```

(6) Crrcomp.out

The output file C*rrcomp.out* contains the nonzero components of the covariance matrix $\mathbf{C}_{rr}^{comp}(N_r \times N_r)$ of computed responses for Model A. This output file has the same structure as the file Crr.*inp*. For the neutron diffusion model, for example, the covariance matrix $\mathbf{C}_{rr}^{comp}(N_r \times N_r)$ is as follows:

$$\boldsymbol{C}_{rr}^{comp} = \begin{pmatrix} 4.99\times10^{17} & 4.99\times10^{17} & 4.82\times10^{17} & 4.82\times10^{17} \\ 4.99\times10^{17} & 4.99\times10^{17} & 4.82\times10^{17} & 4.82\times10^{17} \\ 4.82\times10^{17} & 4.82\times10^{17} & 4.66\times10^{17} & 4.66\times10^{17} \\ 4.82\times10^{17} & 4.82\times10^{17} & 4.66\times10^{17} & 4.66\times10^{17} \end{pmatrix} \tag{7.13}$$

The corresponding output data file C*rrcomp.out* is as follows:

```
4    4      16
1    1      4.98938357E+17
1    2      4.98938357E+17
1    3      4.82134716E+17
1    4      4.82134716E+17
2    1      4.98938357E+17
2    2      4.98938357E+17
2    3      4.82134716E+17
2    4      4.82134716E+17
3    1      4.82134716E+17
3    2      4.82134716E+17
3    3      4.66086473E+17
3    4      4.66086473E+17
4    1      4.82134716E+17
4    2      4.82134716E+17
4    3      4.66086473E+17
4    4      4.66086473E+17
```

(7) Chi2.out

The output file chi2.*out* contains the values for the consistency indicators χ^2 and $\frac{\chi^2}{N_r}$. For the neutron diffusion model, for example, MULTI-PRED outputs the following values for chi2.*out*:

```
chi^2                                  =   4.852
chi^2_d = (chi^2)/(number of responses) =   1.213
```

7.5.2 *Output Data Files for MULTI-PRED Case 2*

Table 7.14 presents the 11 output files generated for MULTI-PRED Case 2; these files are also listed in Table 7.6. Of the 11 output files listed in Table 7.14, 7 output data files have also been listed in Table 7.13; the additional 4 output data files have the same structure as their counterparts for Model A.

Table 7.14. Output Data Files for MULTI-PRED Case 2

Output Data File for Model A	Outputs for the Coupled Matrices	Outputs for the Nb additional parameters for Model A
aBE.out		bBE.out
rBE.out		
CaaBE.out	CabBE.out	CbbBE.out
CrrBE.out		
CarBE.out	CbrBE.out	
Crrcomp.out		
chi2.out		

7.5.3 *Output Data Files for MULTI-PRED Case 3*

Table 7.15 presents the 13 output files generated for MULTI-PRED Case 2; these files are also listed in Table 7.7. Of the 13 output files listed in Table 7.15, 7 output data files have also been listed in Table 7.13; the additional 6 output data files have the same structure as their counterparts for Model A.

Table 7.15. Output Data Files for MULTI-PRED Case 3

Output Data File for Model A	Outputs for the coupled matrices	Outputs for the Nq additional responses for Model A
aBE.out		
rBE.out		qBE.out
CaaBE.out		
CrrBE.out	CrqBE.out	CqqBE.out
CarBE.out	CaqBE.out	
Crrcomp.out	Crqcomp.out	Cqqcomp.out
chi2.out		

7.5.4 Output Data Files for MULTI-PRED Case 4

Table 7.16 presents the 18 output files generated for MULTI-PRED Case 2; these files are also listed in Table 7.8. Of the 18 output files listed in Table 7.15, 7 output data files have also been listed in Table 7.13; the additional 11 output data files have the same structure as their counterparts for Model A.

Table 7.16. Output Data Files for MULTI-PRED Case 4

Output Data File for Model A	**Outputs Data Files for the Coupled Matrices between Model A and Model B**	**Outputs Data Files for Model B**
aBE.out		bBE.out
rBE.out		qBE.out
CaaBE.out	CabBE.out	CbbBE.out
CrrBE.out	CrqBE.out	CqqBE.out
CarBE.out	CaqBE.out, CbrBE.out	CbqBE.out
Crrcomp.out	Crqcomp.out	Cqqcomp.out
chi2.out		

8 FORTRAN Source Code for the Program Multi-Pred

The program Multi-Pred includes the following routines and modules:

- Main program: multi-pred.f90
- Module: ModuleGlobalParameters.f90
- Module: ModuleIO.f90
- Module: ModuleErrors.f90
- Subroutine: Files.f90
- Module: ModuleFiles.f90
- Subroutine: ReadInput.f90
- Module: ModuleReadWrite.f90
- Subroutine: MultiPredSolver.f90
- Module: ModuleMultiPred.f90
- Module: ModuleLapack.f90

The source code for each of them are presented as follows. The structure of the code is organized as shown in Figure 8.1.

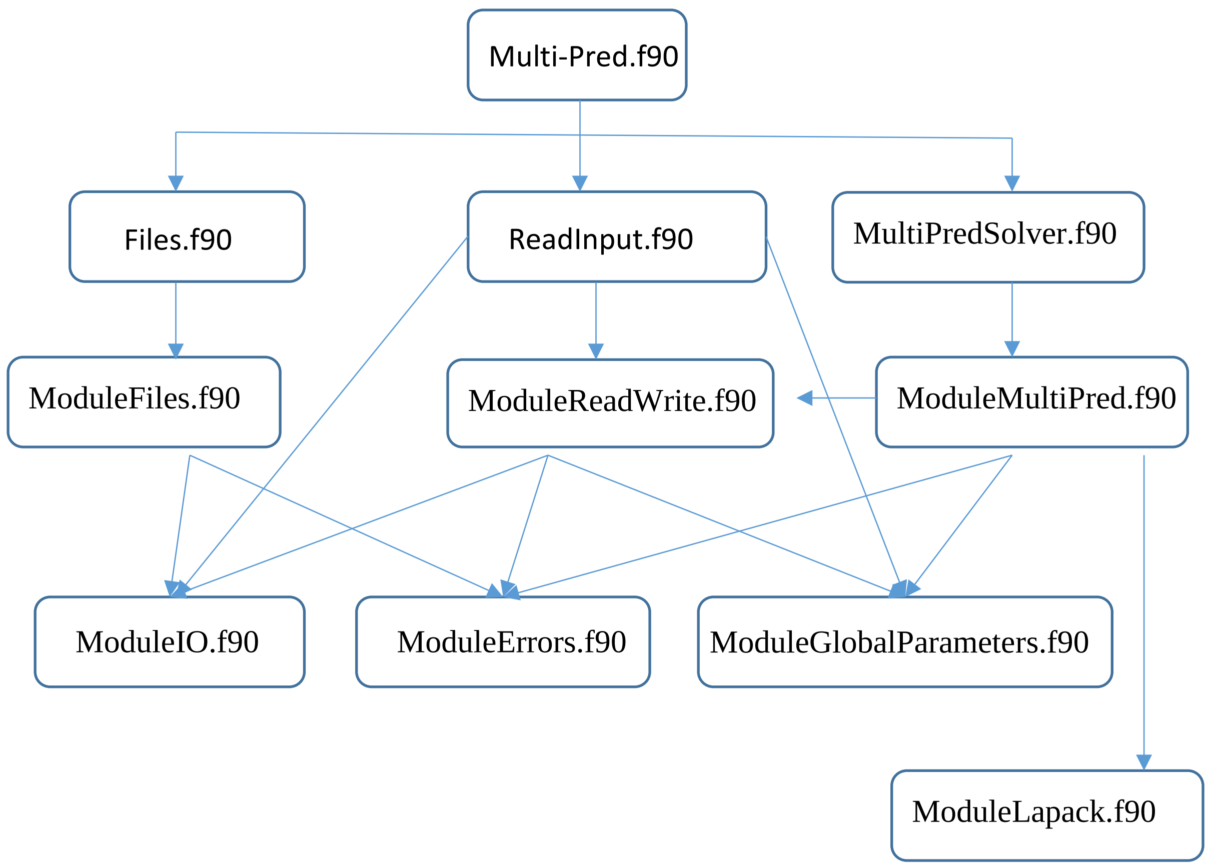

Figure 8.1 Multi-Pred Code Structure

8.1 Main program multi-pred.f90

```
PROGRAM multipred

!*****************************************************************************
!* multi-pred: This program is a computational implementation of the        *
!*             "predictive modeling for coupled multi-physics systems"      *
!*             methodology developed by Cacuci, based on the work "predictive*
!*             modeling for coupled multi-physics systems: I. Theory," Annals*
!*             of Nuclear Energy. 70, 266–278 (2014).                       *
!*                                                                          *
!*             The multi-pred has the following fundamental features:      *
!*             (i) it uses the maximum entropy principle to combine all     *
!*                 available experimental and computational information to  *
!*                 calibrate simultaneously all uncertain quantities,       *
!*                 including model parameters, initial conditions, boundary *
!*                 conditions, and observed model responses;               *
!*            (ii) it provides explicit formulas for the calibrated best    *
!*                 estimate predicted values for the model responses and    *
!*                 parameters;                                              *
!*            (iii)it reduces the predicted uncertainties in these          *
!*                 predicted model responses and parameters, providing      *
!*                 explicit formulas for the predicted covariance matrices  *
!*                 of responses and parameters;                             *
!*            (iv) it provides a quantitative indicator --constructed from  *
!*                 parameter and response covariances and responses         *
!*                 sensitivities to parameters-- for quantifying the        *
!*                 consistency (agreement or disagreement) among the a      *
!*                 priori computational and experimental data.              *
!*                                                                          *
!* multi-pred can perform predictive modeling for the following four cases  *
!* (CaseNumber):                                                            *
!*  = 1 "One-Model" Case: predictive modeling solely for Model A with Na    *
!*                        model parameters and Nr measured responses;       *
!*  = 2 "One-Model" Case: predictive modeling for Model A with Nb additional *
!*                        parameters, but no additional responses;          *
!*  = 3 "One-Model" Case: predictive modeling for Model A with Nq additional *
!*                        responses, but no additional parameters;          *
!*  = 4 "Two-Model" Case: predictive modeling for Model A coupled with Model *
!*                        B.                                                *
!*                                                                          *
!* Developed by the University of South Carolina, Columbia, SC              *
!*                                                                          *
!* called by: none                                                          *
!* calls  to: Files, ReadInput, MultiPredSolver                             *
!*                                                                          *
!*                                                                          *
!*****************************************************************************

  IMPLICIT NONE

! read superfile and open all files for i/o
  call Files

! read all input data
  call ReadInput

! apply the Multi-Pred formulation and generate outputs
  call MultiPredSolver
END PROGRAM multipred
```

8.2 Module ModuleGlobalParameters.f90

```fortran
MODULE ModuleGlobalParameters

! Symbolic names for kind types of 4-, 2- and 1-byte integers:
  INTEGER, PARAMETER :: I4B = SELECTED_INT_KIND(9)
  INTEGER, PARAMETER :: I2B = SELECTED_INT_KIND(4)
  INTEGER, PARAMETER :: I1B = SELECTED_INT_KIND(2)
! Symbolic names for kind types of single- and double precision reals:
  INTEGER, PARAMETER :: SP = KIND(1.0)
  INTEGER, PARAMETER :: DP = KIND(1.0D0)

! Global parameters used in multi-pred:
! alpha   = nominal values of Na parameters of Model A (in)
! rm      = nominal values of Nr measured responses of Model A (in)
! rc      = nominal values of Nr computed responses of Model A (in)
! Caa     = covariance matrix of Na parameters of Model A (in)
! Car     = correlations between Na parameters and Nr responses
!           of Model A (in)
! Crr     = covariance matrix of Nr responses of Model A (in)
! Sra     = sensitivities of Model B (in)
! beta    = nominal values of Nb parameters of Model B (in)
! qm      = nominal values of Nq measured responses of Model B (in)
! qc      = nominal values of Nq computed responses of Model B (in)
! Cbb     = covariance matrix of Nb parameters of Model B (in)
! Cbq     = correlations between Nb parameters and N1 responses
!           of Model B (in)
! Cqq     = covariance matrix of Nq responses of Model B (in)
! Sqb     = sensitivities of Model B (in)
! Cab     = correlations between Na parameters of Model A and Nb
!           parameters of Model B  (in)
! Caq     = correlations between Na parameters of Model A and Nq
!           responses of Model B (in)
! Cbr     = correlations between Nb parameters of Model B and Nr
!           responses of Model A (in)
! Crq     = correlations between Nr responses of Model A and Nq
!           responses of Model B (in)
! Srb     = sensitivities of Nr responses of Model A w.r.t. Nb
!           parameters of Model B (in)
! Sqa     = sensitivities of Nq responses of Model B w.r.t. Na
!           parameters of Model A (in)
! aBE     = best-estimate nominal values of Na parameters of Model A (out)
! rBE     = best-estimate nominal values of Nr responses of Model A (out)
! CaaBE   = predicted covariance matrix of Na parameters of Model A (out)
! CarBE   = predicted correlation matrix between the Na parameters
!           and Nr responses of Model A (out)
! CrrBE   = predicted covariance matrix of Nr responses of Model A (out)
! Crrcomp = covariance matrix of Nr computed responses of Model A (out)
! bBE     = best-estimate nominal values of Nb parameters of Model B (out)
! qBE     = best-estimate nominal values of Nq responses of Model B (out)
! CbbBE   = predicted covariance matrix of Nb parameters of Model B (out)
! CbqBE   = predicted correlation matrix between the Nb parameters
!           and Nq responses of Model B (out)
! CqqBE   = predicted covariance matrix of Nq responses of Model B (out)
! Cqqcomp = covariance matrix of Nq computed responses of Model B (out)
! CabBE   = predicted correlation matrix between Na parameters of
!           Model A and Nb parameters of Model B (out)
! CaqBE   = predicted correlation matrix between Na parameters of
!           Model A and Nq responses of Model B (out)
! CbrBE   = predicted correlation matrix between Nb parameters of
```

```
!            Model B and Nr responses of Model A (out)
! CrqBE   = predicted correlation matrix between Nr responses of
!            Model A and Nq responses of Model B (out)
! Cqqcomp = covariance matrix between Nr computed responses of
!            Model A and Nq computed responses of Model B (out)
! chi2    = value of the consistency indicator chi^2   (out)

  PUBLIC
  REAL(DP), ALLOCATABLE ::  alpha(:)
  REAL(DP), ALLOCATABLE ::  rm(:)
  REAL(DP), ALLOCATABLE ::  rc(:)
  REAL(DP), ALLOCATABLE ::  Caa(:,:)
  REAL(DP), ALLOCATABLE ::  Car(:,:)
  REAL(DP), ALLOCATABLE ::  Crr(:,:)
  REAL(DP), ALLOCATABLE ::  Sra(:,:)

  REAL(DP), ALLOCATABLE ::  beta(:)
  REAL(DP), ALLOCATABLE ::  qm(:)
  REAL(DP), ALLOCATABLE ::  qc(:)
  REAL(DP), ALLOCATABLE ::  Cbb(:,:)
  REAL(DP), ALLOCATABLE ::  Cbq(:,:)
  REAL(DP), ALLOCATABLE ::  Cqq(:,:)
  REAL(DP), ALLOCATABLE ::  Sqb(:,:)

  REAL(DP), ALLOCATABLE ::  Cab(:,:)
  REAL(DP), ALLOCATABLE ::  Caq(:,:)
  REAL(DP), ALLOCATABLE ::  Cbr(:,:)
  REAL(DP), ALLOCATABLE ::  Crq(:,:)
  REAL(DP), ALLOCATABLE ::  Srb(:,:)
  REAL(DP), ALLOCATABLE ::  Sqa(:,:)

  REAL(DP), ALLOCATABLE ::  aBE(:)
  REAL(DP), ALLOCATABLE ::  rBE(:)
  REAL(DP), ALLOCATABLE ::  CaaBE(:,:)
  REAL(DP), ALLOCATABLE ::  CarBE(:,:)
  REAL(DP), ALLOCATABLE ::  CrrBE(:,:)
  REAL(DP), ALLOCATABLE ::  Crrcomp(:,:)

  REAL(DP), ALLOCATABLE ::  bBE(:)
  REAL(DP), ALLOCATABLE ::  qBE(:)
  REAL(DP), ALLOCATABLE ::  CbbBE(:,:)
  REAL(DP), ALLOCATABLE ::  CbqBE(:,:)
  REAL(DP), ALLOCATABLE ::  CqqBE(:,:)
  REAL(DP), ALLOCATABLE ::  Cqqcomp(:,:)

  REAL(DP), ALLOCATABLE ::  CabBE(:,:)
  REAL(DP), ALLOCATABLE ::  CaqBE(:,:)
  REAL(DP), ALLOCATABLE ::  CbrBE(:,:)
  REAL(DP), ALLOCATABLE ::  CrqBE(:,:)
  REAL(DP), ALLOCATABLE ::  Crqcomp(:,:)

  REAL(DP)              ::  chi2

  INTEGER(I4B)          ::  CaseNumber
  INTEGER(I4B)          ::  Na, Nr, Nb, Nq

END MODULE ModuleGlobalParameters
```

8.3 Module ModuleIO.f90

```
MODULE ModuleIO

  IMPLICIT NONE

  INTEGER, PARAMETER      :: usupr      = 5,   &
                             udims      =20,   &
!unit for the input of model A
                             ua_nom     =21,   &
                             ur_mea     =22,   &
                             ur_com     =23,   &
                             uC_aa      =24,   &
                             uC_ar      =25,   &
                             uC_rr      =26,   &
                             uS_ra      =27,   &

!unit for the input of model B
                             ub_nom     =31,   &
                             uq_mea     =32,   &
                             uq_com     =33,   &
                             uC_bb      =34,   &
                             uC_bq      =35,   &
                             uC_qq      =36,   &
                             uS_qb      =37,   &

!unit for the input of the coupled matrices
!between models A & B
                             uC_ab      =41,   &
                             uC_aq      =42,   &
                             uC_br      =43,   &
                             uC_rq      =44,   &
                             uS_rb      =45,   &
                             uS_qa      =46,   &

!unit for the output of model A
                             ua_BE      =51,   &
                             ur_BE      =52,   &
                             uC_aaBE    =53,   &
                             uC_rrBE    =54,   &
                             uC_arBE    =55,   &
                             uCrr_comp  =56,   &

!unit for the output of model B
                             ub_BE      =61,   &
                             uq_BE      =62,   &
                             uC_bbBE    =63,   &
                             uC_qqBE    =64,   &
                             uC_bqBE    =65,   &
                             uCqq_comp  =66,   &

!unit for the output of the coupled matrices
!between models A & B
                             uC_abBE    =71,   &
                             uC_aqBE    =72,   &
                             uC_brBE    =73,   &
                             uC_rqBE    =74,   &
                             uCrq_comp  =75,   &
                             uchi2      =76

```

```
  LOGICAL :: a_nom   =.false., r_mea   =.false., r_com =.false.,    &
             C_aa    =.false., C_ar    =.false., C_rr  =.false.,    &
             S_ra    =.false., b_nom   =.false., q_mea =.false.,    &
             q_com   =.false., C_bb    =.false., C_bq  =.false.,    &
             C_qq    =.false., S_qb    =.false., C_ab  =.false.,    &
             C_aq    =.false., C_br    =.false., C_rq  =.false.,    &
             S_rb    =.false., S_qa    =.false., a_BE  =.false.,    &
             r_BE    =.false., C_aaBE  =.false., C_rrBE=.false.,    &
             C_arBE  =.false., Crr_comp=.false., b_BE  =.false.,    &
             q_BE    =.false., C_bbBE  =.false., C_qqBE=.false.,    &
             C_bqBE  =.false., Cqq_comp=.false., C_abBE=.false.,    &
             C_aqBE  =.false., C_brBE  =.false., C_rqBE=.false.,    &
             Crq_comp=.false., chi_2   =.false., dims  =.false.

END MODULE ModuleIO
```

8.4 Module ModuleErrors.f90

```
MODULE ModuleErrors

  USE ModuleGlobalParameters

  IMPLICIT NONE
  CHARACTER(LEN=80) errstr(5)
  INTEGER(I4B) :: alloc_err,ierr = 0
  LOGICAL      :: lerr = .false.

  DATA errstr/                                        &
  'Error condition during file open',                 & ! error 1
  'Error condition during file read',                 & ! error 2
  'Error condition during file write',                & ! error 3
  'Unable to allocate storage for array',             & ! error 4
  'Unable to deallocate storage for array'/             ! error 5

CONTAINS

  SUBROUTINE errmsg

    write(*,'(/,a)')     '****'//trim(errstr(ierr))//'****'

  END SUBROUTINE errmsg

END MODULE ModuleErrors
```

8.5 Subroutine Files.f90

```
SUBROUTINE Files

!******************************************************************************
!*                                                                            *
!* Open input/output files for multi-pred using super file format. For        *
!* different cases, the required input and output files are different.        *
!*                                                                            *
!* called by:  multipred                                                      *
!* calls  to:  getarg,errmsg, filescase1, filescase2, filescase3, filescase4  *
!*                                                                            *
!******************************************************************************

  !Global---------------------------------------------------------------------
  USE ModuleFiles

  IMPLICIT NONE

  !Local----------------------------------------------------------------------
  CHARACTER(LEN=128) :: argv,filename
  CHARACTER(LEN=12)  :: header
  CHARACTER(LEN=8)   :: category
  INTEGER(I4B)       :: argc, i
  CHARACTER(LEN=128) :: commnt

  argc = 1
  call getarg(argc,argv)
  filename = argv
  open(usupr,file=filename,status='old',err=100)
  read(usupr,'(a12)') header
  if (header /= 'MultiPredSup') then
    write(*,*) 'This is not a multi-pred superfile.'
    stop
  end if
  write(*,900) trim(filename)
  ! read the dimensions.inp file for control variables
  read(usupr,*) category,filename
  if (category == 'dims') then
    open(udims,file=filename,status='old',err=100)
    write(*,1000) trim(filename)
    dims = .true.
  else
    write(*,*) 'dimensions.inp is specified in the superfile.'
    stop
  end if

  ! read CaseNumber
  do i=1, 9
    read(udims,*,err=100) commnt
  end do
  read (udims,*,err=100) CaseNumber

  ! file I/O for Case 1
  if(CaseNumber == 1) then
    call filescase1
  end if

  ! file I/O for Case 2
  if(CaseNumber == 2) then
```

```
    call filescase2
  end if

  ! file I/O for Case 3
  if(CaseNumber == 3) then
    call filescase3
  end if

  ! file I/O for Case 4
  if(CaseNumber == 4) then
    call filescase4
  end if

  write(*, 1001)
  if(CaseNumber == 1) then
     write(*, 1002)
  else if (CaseNumber == 2) then
     write(*, 1003)
  else if (CaseNumber == 3) then
     write(*, 1004)
  else if (CaseNumber == 4) then
     write(*, 1005)
  end if

  return
  100 lerr = .true.;ierr = 1;call errmsg;stop
  900 format(/, &
       1x,'-------------------- Summary of Multil-Pred  ------------------',/, &
       1x,' Input super file ..............................................',a)
  1000 format(&
       1x,' Input file for Case selection and dimensions control ..........',a)
  1001 format(/, &
       1x,'Case selected for this run:',a)
  1002 format(&
       1x,'Case 1 -- "One-Model" Case: predictive modeling solely for Model A' /,&
          '            with Na model parameters and Nr measured responses.', a)
  1003 format(&
       1x,'Case 2 -- "One-Model" Case: predictive modeling for Model A with Nb'/,&
          '            additional parameters, but no additional responses.', a)
  1004 format(&
       1x,'Case 3 -- "One-Model" Case: predictive modeling for Model A with Nq'/,&
          '            additional responses, but no additional parameters.', a)
  1005 format(&
       1x,'Case 4 -- "Two-Model" Case: predictive modeling for Model A coupled'/,&
          '            with Model B.', a)

END SUBROUTINE Files
```

8.6 Module ModuleFiles.f90

```
MODULE ModuleFiles
!***********************************************************************
!*                                                                     *
!* Module ModuleFiles encapsulates subroutines for:                    *
!*                     subroutine filescase1()                         *
!*                     subroutine filescase2()                         *
!*                     subroutine filescase3()                         *
!*                     subroutine filescase4()                         *
!*                                                                     *
!***********************************************************************

  !Global--------------------------------------------------------------
  USE ModuleErrors
  USE ModuleIO
  USE ModuleGlobalParameters

  IMPLICIT NONE

CONTAINS

!++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

SUBROUTINE filescase1()

!***********************************************************************
!*                                                                     *
!* open needed files for case 1: "One-Model" Case: predictive modeling *
!* solely for Model A with Na model parameters and Nr                  *
!* measured responses.                                                 *
!*                                                                     *
!* Open input/output files for multi-pred using super file format      *
!*                                                                     *
!* Category     I/O     Description                                    *
!* --------   -------   -----------                                    *
!* 'dims'     input    defines the Case selection and dimensions control *
!* 'a_nom'    input    nominal values of Na parameters of Model A      *
!* 'r_mea'    input    nominal values of Nr measured responses of Model A *
!* 'r_com'    input    nominal values of Nr computed responses of Model A *
!* 'C_aa'     input    covariance matrix of Na parameters of Model A   *
!* 'C_ar'     input    covariance matrix of parameter-response of Model A *
!* 'C_rr'     input    covariance matrix of Nr responses of Model A    *
!* 'S_ra'     input    sensitivities of Model A                        *
!*                                                                     *
!* 'a_BE'     output   best-estimate nominal values of Na parameters of *
!*                     Model A                                         *
!* 'r_BE'     output   best-estimate nominal values of Nr responses of *
!*                     Model A                                         *
!* 'C_aaBE'   output   predicted covariance matrix of Na parameters of *
!*                     Model A                                         *
!* 'C_rrBE'   output   predicted covariance matrix of Nr responses of  *
!*                     Model A                                         *
!* 'C_arBE'   output   predicted correlation matrix between the Na     *
!*                    parameters and Nr responses of Model A           *
!* 'Crr_comp' output   covariance matrix of Nr computed responses of Model A *
!* 'chi_2'    output   value of the consistency indicator chi^2        *
!*                                                                     *
!* called by:  Files                                                   *
!* calls  to:  getarg,errmsg                                           *
```

```
!*                                                                            *
!******************************************************************************
  IMPLICIT NONE
  !Local----------------------------------------------------------------------
  CHARACTER(LEN=128) :: filename
  CHARACTER(LEN=8)   :: category

  inquire(unit=usupr)
  do
    read(usupr,*,end=1) category,filename

!   INPUT FILES
    if (category == 'a_nom') then
      open(ua_nom,file=filename,status='old',err=100)
      write(*,1010) trim(filename)
      a_nom = .true.
    else if (category == 'r_mea') then
      open(ur_mea,file=filename,status='old',err=100)
      write(*,1020) trim(filename)
      r_mea = .true.
    else if (category == 'r_com') then
      open (ur_com,file=filename,status='old',err=100)
      write(*,1030) trim(filename)
      r_com = .true.
    else if (category == 'C_aa') then
      open(uC_aa,file=filename,status='old',err=100)
      write(*,1040) trim(filename)
      C_aa = .true.
    else if (category == 'C_ar') then
      open(uC_ar,file=filename,status='old',err=100)
      write(*,1050) trim(filename)
      C_ar = .true.
    else if (category == 'C_rr') then
      open(uC_rr,file=filename,status='old',err=100)
      write(*,1060) trim(filename)
      C_rr = .true.
    else if (category == 'S_ra') then
      open(uS_ra,file=filename,status='old',err=100)
      write(*,1070) trim(filename)
      S_ra = .true.

!   OUTPUT FILES FOR RESULTS
    else if (category == 'a_BE') then
      open (ua_BE,file=filename,status='unknown',err=100)
      write(*,1100) trim(filename)
      a_BE = .true.
    else if (category == 'r_BE') then
      open (ur_BE,file=filename,status='unknown',err=100)
      write(*,1110) trim(filename)
      r_BE = .true.
    else if (category == 'C_aaBE') then
      open (uC_aaBE,file=filename,status='unknown',err=100)
      write(*,1120) trim(filename)
      C_aaBE = .true.
    else if (category == 'C_rrBE') then
      open (uC_rrBE,file=filename,status='unknown',err=100)
      write(*,1130) trim(filename)
      C_rrBE = .true.
    else if (category == 'C_arBE') then
      open (uC_arBE,file=filename,status='unknown',err=100)
      write(*,1140) trim(filename)
```

```fortran
      C_arBE = .true.
    else if (category == 'Crr_comp') then
      open (uCrr_comp,file=filename,status='unknown',err=100)
      write(*,1150) trim(filename)
      Crr_comp = .true.
    else if (category == 'chi2') then
      open (uchi2,file=filename,status='unknown',err=100)
      write(*,1160) trim(filename)
      chi_2 = .true.
!    else
!      write(*,1500) category,filename
    end if
  end do
  1 close(usupr)

!   CHECK TO SEE IF REQUIRED INPUT/OUTPUT FILES ARE OPEN IN THE SUPERFILE
  ! INPUT
  if (.not. a_nom) then
    write(*,1510) 'a_nom';   stop
  end if
  if (.not. r_mea) then
    write(*,1510) 'r_mea';   stop
  end if
  if (.not. r_com) then
    write(*,1510) 'r_com';   stop
  end if
  if (.not. C_aa) then
    write(*,1510) 'C_aa';    stop
  end if
  if (.not. S_ra) then
    write(*,1510) 'S_ra';    stop
  end if
  if (.not. C_ar) then
    write(*,1510) 'C_ar';    stop
  end if
  if (.not. C_rr) then
    write(*,1510) 'C_rr';    stop
  end if
  ! OUTPUT
  if (.not. Crr_comp) then
    write(*,1520) 'Crr_comp';stop
  end if
  if (.not. a_BE) then
    write(*,1520) 'a_BE';    stop
  end if
  if (.not. r_BE) then
    write(*,1520) 'r_BE';    stop
  end if
  if (.not. C_aaBE) then
    write(*,1520) 'C_aaBE';  stop
  end if
  if (.not. C_rrBE) then
    write(*,1520) 'C_rrBE';  stop
  end if
  if (.not. C_arBE) then
    write(*,1520) 'C_arBE';  stop
  end if
  if (.not. chi_2) then
    write(*,1520) 'chi_2';   stop
  end if
  return
```

```

  100 lerr = .true.;ierr = 1;call errmsg;stop
  1010 format(&
       1x,' Input file for parameters nominal values ......................',a)
  1020 format(&
       1x,' Input file for measured responses nominal values ..............',a)
  1030 format(&
       1x,' Input file for computed responses nominal values ..............',a)
  1040 format(&
       1x,' Input file for covariance matrix of parameters ................',a)
  1050 format(&
       1x,' Input file for correlation matrix of parameter-response .......',a)
  1060 format(&
       1x,' Input file for covariance matrix for responses ................',a)
  1070 format(&
       1x,' Input file for sensitivities ..................................',a)
  1100 format(&
       1x,'Output file for best-estimate parameters values ................',a)
  1110 format(&
       1x,'Output file for best-estimate response values ..................',a)
  1120 format(&
       1x,'Output file for predicted covariance matrix of parameters.......',a)
  1130 format(&
       1x,'Output file for predicted covariance matrix of responses .......',a)
  1140 format(&
       1x,'Output file for predicted correlation of parameter-response ....',a)
  1150 format(&
       1x,'Output file for covariance matrix for computed responses .......',a)
  1160 format(&
       1x,'Output file for the consistency indicator chi^2 ................',a)
  1500 format(1x,'Invalid file type:',a,' ....skipping',/, &
              1x,'Specified file name:',a)
  1510 format(&
       1x,a,' input file not specified in superfile!')
  1520 format(&
       1x,a,' output file not specified in superfile!')

END SUBROUTINE filescase1

!+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

SUBROUTINE filescase2()

!*****************************************************************************
!*                                                                           *
!* open needed files for case 2: predictive modeling for Model A with Nb     *
!* additional parameters, but no additional responses;                       *
!*                                                                           *
!* Open input/output files for multi-pred using super file format            *
!*                                                                           *
!* Category     I/O     Description                                          *
!* --------   -------   -----------                                          *
!* 'dims'     input    defines the Case selection and dimensions control     *
!* 'a_nom'    input    nominal values of Na parameters of Model A            *
!* 'r_mea'    input    nominal values of Nr measured responses of Model A    *
!* 'r_com'    input    nominal values of Nr computed responses of Model A    *
!* 'C_aa'     input    covariance matrix of Na parameters of Model A         *
!* 'C_ar'     input    covariance matrix of parameter-response of Model A    *
!* 'C_rr'     input    covariance matrix of Nr responses of Model A          *
!* 'S_ra'     input    sensitivities of Model A                              *
!*                                                                           *
```

```
!* 'b_nom'    input    nominal values of Nb additional parameters for Model A *
!* 'C_bb'     input    covariance matrix of Nb additional parameters          *
!*                                                                            *
!* 'C_ab'     input    correlations between Na parameters and Nb parameters   *
!* 'C_br'     input    correlations between Nb parameters and Nr responses    *
!* 'S_rb'     input    sensitivities of Nr responses wrt Nb parameters        *
!*                                                                            *
!* 'a_BE'     output   best-estimate nominal values of Na parameters of       *
!*                     Model A                                                *
!* 'r_BE'     output   best-estimate nominal values of Nr responses of        *
!*                     Model A                                                *
!* 'C_aaBE'   output   predicted covariance matrix of Na parameters of        *
!*                     Model A                                                *
!* 'C_rrBE'   output   predicted covariance matrix of Nr responses of Model A *
!* 'C_arBE'   output   predicted correlation matrix between the Na parameters *
!*       and Nr responses of Model A                                          *
!* 'Crr_comp' output   covariance matrix of Nr computed responses of Model A  *
!*                                                                            *
!* 'b_BE'     output   best-estimate nominal values of Nb additional          *
!*                     parameters for Model A                                 *
!* 'C_bbBE'   output   predicted covariance matrix of Nb additional           *
!*                     parameters for Model A                                 *
!*                                                                            *
!* 'C_abBE'   output   predicted correlations between Na parameters and Nb    *
!*                     additional parameters for Model A                      *
!* 'C_brBE'   output   predicted correlations between Nb additional parameters*
!*                     and Nr responses of Model A                            *
!* 'chi_2'    output   value of the consistency indicator chi^2               *
!*                                                                            *
!* called by:  Files                                                          *
!* calls  to:  getarg,errmsg                                                  *
!*                                                                            *
!******************************************************************************
  IMPLICIT NONE
  !Local---------------------------------------------------------------------
  CHARACTER(LEN=128) :: filename
  CHARACTER(LEN=8)   :: category

  inquire(unit=usupr)
  do
    read(usupr,*,end=1) category,filename

!   INPUT FILES FOR MODEL A
    if (category == 'a_nom') then
      open(ua_nom,file=filename,status='old',err=100)
      a_nom = .true.
    else if (category == 'r_mea') then
      open(ur_mea,file=filename,status='old',err=100)
      r_mea = .true.
    else if (category == 'r_com') then
      open (ur_com,file=filename,status='old',err=100)
      r_com = .true.
    else if (category == 'C_aa') then
      open(uC_aa,file=filename,status='old',err=100)
      C_aa = .true.
    else if (category == 'C_ar') then
      open(uC_ar,file=filename,status='old',err=100)
      C_ar = .true.
    else if (category == 'C_rr') then
      open(uC_rr,file=filename,status='old',err=100)
      C_rr = .true.
```

```
    else if (category == 'S_ra') then
      open(uS_ra,file=filename,status='old',err=100)
      S_ra = .true.

!   INPUT FILES FOR ADDITIONAL PARAMETERS
    else if (category == 'b_nom') then
      open(ub_nom,file=filename,status='old',err=100)
      b_nom = .true.
    else if (category == 'C_bb') then
      open(uC_bb,file=filename,status='old',err=100)
      C_bb = .true.

!   INPUT FILES FOR COUPLED MATRICES
    else if (category == 'C_ab') then
      open(uC_ab,file=filename,status='old',err=100)
      C_ab = .true.
    else if (category == 'C_br') then
      open(uC_br,file=filename,status='old',err=100)
      C_br = .true.
    else if (category == 'S_rb') then
      open(uS_rb,file=filename,status='old',err=100)
      S_rb = .true.

!   OUTPUT FILES FOR MODEL A
    else if (category == 'a_BE') then
      open (ua_BE,file=filename,status='unknown',err=100)
      a_BE = .true.
    else if (category == 'r_BE') then
      open (ur_BE,file=filename,status='unknown',err=100)
      r_BE = .true.
    else if (category == 'C_aaBE') then
      open (uC_aaBE,file=filename,status='unknown',err=100)
      C_aaBE = .true.
    else if (category == 'C_rrBE') then
      open (uC_rrBE,file=filename,status='unknown',err=100)
      C_rrBE = .true.
    else if (category == 'C_arBE') then
      open (uC_arBE,file=filename,status='unknown',err=100)
      C_arBE = .true.
    else if (category == 'Crr_comp') then
      open (uCrr_comp,file=filename,status='unknown',err=100)
      Crr_comp = .true.

!   OUTPUT FILES FOR MODEL B
    else if (category == 'b_BE') then
      open (ub_BE,file=filename,status='unknown',err=100)
      b_BE = .true.
    else if (category == 'C_bbBE') then
      open (uC_bbBE,file=filename,status='unknown',err=100)
      C_bbBE = .true.

!   OUTPUT FILES FOR COUPLED MATRICES BETWEEN MODELS A & B
    else if (category == 'C_abBE') then
      open (uC_abBE,file=filename,status='unknown',err=100)
      C_abBE = .true.
    else if (category == 'C_brBE') then
      open (uC_brBE,file=filename,status='unknown',err=100)
      C_brBE = .true.

    else if (category == 'chi2') then
      open (uchi2,file=filename,status='unknown',err=100)
```

```
      chi_2 = .true.
!    else
!      write(*,1500) category,filename
    end if
  end do
  1 close(usupr)

!   CHECK TO SEE IF REQUIRED INPUT/OUTPUT FILES ARE OPEN IN THE SUPERFILE
  ! INPUT
  if (.not. a_nom) then
    write(*,1510) 'a_nom';   stop
  end if
  if (.not. r_mea) then
    write(*,1510) 'r_mea';   stop
  end if
  if (.not. r_com) then
    write(*,1510) 'r_com';   stop
  end if
  if (.not. C_aa) then
    write(*,1510) 'C_aa';    stop
  end if
  if (.not. C_ar) then
    write(*,1510) 'C_ar';    stop
  end if
  if (.not. C_rr) then
    write(*,1510) 'C_rr';    stop
  end if
  if (.not. S_ra) then
    write(*,1510) 'S_ra';    stop
  end if
  if (.not. b_nom) then
    write(*,1510) 'b_nom';   stop
  end if
  if (.not. C_bb) then
    write(*,1510) 'C_bb';    stop
  end if
  if (.not. C_ab) then
    write(*,1510) 'C_ab';    stop
  end if
  if (.not. C_br) then
    write(*,1510) 'C_br';    stop
  end if
  if (.not. S_rb) then
    write(*,1510) 'S_rb';    stop
  end if

  ! OUTPUT
  if (.not. Crr_comp) then
    write(*,1520) 'Crr_comp';stop
  end if
  if (.not. a_BE) then
    write(*,1520) 'a_BE';    stop
  end if
  if (.not. r_BE) then
    write(*,1520) 'r_BE';    stop
  end if
  if (.not. C_aaBE) then
    write(*,1520) 'C_aaBE';  stop
  end if
  if (.not. C_rrBE) then
    write(*,1520) 'C_rrBE';  stop
```

```fortran
  end if
  if (.not. C_arBE) then
    write(*,1520) 'C_arBE';  stop
  end if
  if (.not. b_BE) then
    write(*,1520) 'b_BE';    stop
  end if
  if (.not. C_bbBE) then
    write(*,1520) 'C_bbBE';  stop
  end if
  if (.not. C_abBE) then
    write(*,1520) 'C_abBE';  stop
  end if
  if (.not. C_brBE) then
    write(*,1520) 'C_brBE';  stop
  end if
  if (.not. chi_2) then
    write(*,1520) 'chi_2';   stop
  end if

  return

  100 lerr = .true.;ierr = 1;call errmsg;stop
  1010 format(&
       1x,' Input file for parameters nominal values ......................',a)
  1020 format(&
       1x,' Input file for measured responses nominal values ..............',a)
  1030 format(&
       1x,' Input file for computed responses nominal values ..............',a)
  1040 format(&
       1x,' Input file for covariance matrix of parameters ................',a)
  1050 format(&
       1x,' Input file for correlation matrix of parameter-response .......',a)
  1060 format(&
       1x,' Input file for covariance matrix for responses ................',a)
  1070 format(&
       1x,' Input file for sensitivities ..................................',a)
  1100 format(&
       1x,'Output file for best-estimate parameters values ................',a)
  1110 format(&
       1x,'Output file for best-estimate response values ..................',a)
  1120 format(&
       1x,'Output file for predicted covariance matrix of parameters.......',a)
  1130 format(&
       1x,'Output file for predicted covariance matrix of responses .......',a)
  1140 format(&
       1x,'Output file for predicted correlation of parameter-response ....',a)
  1150 format(&
       1x,'Output file for covariance matrix for computed responses .......',a)
  1160 format(&
       1x,'Output file for the consistency indicator chi^2 ................',a)
  1500 format(1x,'Invalid file type:',a,' ....skipping',/, &
              1x,'Specified file name:',a)
  1510 format(&
       1x,a,' input file not specified in superfile!')
  1520 format(&
       1x,a,' output file not specified in superfile!')

END SUBROUTINE filescase2

!++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
SUBROUTINE filescase3()

!***************************************************************************
!*                                                                         *
!* open needed files for read and write for Case 3: predictive modeling for   *
!* Model A with Nq additional responses, but no additional parameters.         *
!*                                                                         *
!* Open input/output files for multi-pred using super file format            *
!*                                                                         *
!* Category     I/O     Description                                        *
!* --------    ------   -----------                                        *
!* 'dims'     input    defines the Case selection and dimensions control      *
!* 'a_nom'    input    nominal values of Na parameters of Model A             *
!* 'r_mea'    input    nominal values of Nr measured responses of Model A     *
!* 'r_com'    input    nominal values of Nr computed responses of Model A     *
!* 'C_aa'     input    covariance matrix of Na parameters of Model A          *
!* 'C_ar'     input    correlations between Na parameters and Nr responses of *
!*                     Model A                                             *
!* 'C_rr'     input    covariance matrix of Nr responses of Model A           *
!* 'S_ra'     input    sensitivities of Model A                               *
!*                                                                         *
!* 'q_mea'    input    nominal values of Nq measured responses for Model A    *
!* 'q_com'    input    nominal values of Nq computed responses for Model A    *
!* 'C_qq'     input    covariance matrix of Nq additional responses for       *
!*                     Model A                                             *
!*                                                                         *
!* 'C_aq'     input    correlations between Na parameters of Model A and      *
!*                     Nq additional responses for Model A                 *
!* 'C_rq'     input    correlations between Nr responses of Model A and Nq    *
!*                     additional responses for Model A                    *
!* 'S_qa'     input    sensitivities of Nq additional responses for Model A   *
!*                     w.r.t. Na parameters of Model A                     *
!*                                                                         *
!* 'a_BE'     output   best-estimate nominal values of Na parameters of       *
!*                     Model A                                             *
!* 'r_BE'     output   best-estimate nominal values of Nr responses of        *
!*                     Model A                                             *
!* 'C_aaBE'   output   predicted covariance matrix of Na parameters of        *
!*                     Model A                                             *
!* 'C_rrBE'   output   predicted covariance matrix of Nr responses of Model A *
!* 'C_arBE'   output   predicted correlation matrix between the Na parameters *
!*            and Nr responses of Model A                            *
!* 'Crr_comp' output   covariance matrix of Nr computed responses of Model A  *
!*                                                                         *
!* 'q_BE'     output   best-estimate nominal values of Nq additional response *
!*                     for Model A                                         *
!* 'C_qqBE'   output   predicted covariance matrix of Nq additional response  *
!*                     for Model A                                         *
!* 'Cqq_comp' output   covariance matrix for Nq additional computed responses *
!*                     of Model A                                          *
!*                                                                         *
!* 'C_aqBE'   output   predicted correlation matrix between Na parameters of  *
!*                     Model A and Nq additional responses for Model A     *
!* 'C_rqBE'   output   predicted correlation matrix between Nr responses of   *
!*                     Model A and Nq additional responses for Model A     *
!* 'Crq_comp' output   correlation matrix between Nr computed responses of    *
!*                     Model A and Nq additional computed responses of Model A*
!* 'chi_2'    output   value of the consistency indicator chi^2               *
!*                                                                         *
!* called by:  Files                                                       *
```

```
!* calls  to:  getarg,errmsg                                               *
!*                                                                         *
!**************************************************************************

  IMPLICIT NONE
  !Local------------------------------------------------------------------
  CHARACTER(LEN=128) :: filename
  CHARACTER(LEN=8)   :: category

  inquire(unit=usupr)
!  write(*,900) trim(filename)
  do
    read(usupr,*,end=1) category,filename

!   INPUT FILES FOR MODEL A
    if (category == 'a_nom') then
      open(ua_nom,file=filename,status='old',err=100)
      a_nom = .true.
    else if (category == 'r_mea') then
      open(ur_mea,file=filename,status='old',err=100)
      r_mea = .true.
    else if (category == 'r_com') then
      open (ur_com,file=filename,status='old',err=100)
      r_com = .true.
    else if (category == 'C_aa') then
      open(uC_aa,file=filename,status='old',err=100)
      C_aa = .true.
    else if (category == 'C_ar') then
      open(uC_ar,file=filename,status='old',err=100)
      C_ar = .true.
    else if (category == 'C_rr') then
      open(uC_rr,file=filename,status='old',err=100)
      C_rr = .true.
    else if (category == 'S_ra') then
      open(uS_ra,file=filename,status='old',err=100)
      S_ra = .true.

!   INPUT FILES FOR MODEL B
    else if (category == 'q_mea') then
      open(uq_mea,file=filename,status='old',err=100)
      q_mea = .true.
    else if (category == 'q_com') then
      open (uq_com,file=filename,status='old',err=100)
      q_com = .true.
    else if (category == 'C_qq') then
      open(uC_qq,file=filename,status='old',err=100)
      C_qq = .true.

!   INPUT FILES FOR COUPLED MATRICES
    else if (category == 'C_aq') then
      open(uC_aq,file=filename,status='old',err=100)
      C_aq = .true.
    else if (category == 'C_rq') then
      open(uC_rq,file=filename,status='old',err=100)
      C_rq = .true.
    else if (category == 'S_qa') then
      open(uS_qa,file=filename,status='old',err=100)
      S_qa = .true.

!   OUTPUT FILES FOR MODEL A
    else if (category == 'a_BE') then
```

```
      open (ua_BE,file=filename,status='unknown',err=100)
      a_BE = .true.
    else if (category == 'r_BE') then
      open (ur_BE,file=filename,status='unknown',err=100)
      r_BE = .true.
    else if (category == 'C_aaBE') then
      open (uC_aaBE,file=filename,status='unknown',err=100)
      C_aaBE = .true.
    else if (category == 'C_rrBE') then
      open (uC_rrBE,file=filename,status='unknown',err=100)
      C_rrBE = .true.
    else if (category == 'C_arBE') then
      open (uC_arBE,file=filename,status='unknown',err=100)
      C_arBE = .true.
    else if (category == 'Crr_comp') then
      open (uCrr_comp,file=filename,status='unknown',err=100)
      Crr_comp = .true.

!   OUTPUT FILES FOR MODEL B
    else if (category == 'q_BE') then
      open (uq_BE,file=filename,status='unknown',err=100)
      q_BE = .true.
    else if (category == 'C_qqBE') then
      open (uC_qqBE,file=filename,status='unknown',err=100)
      C_qqBE = .true.
    else if (category == 'Cqq_comp') then
      open (uCqq_comp,file=filename,status='unknown',err=100)
      Cqq_comp = .true.

!   OUTPUT FILES FOR COUPLED MATRICES BETWEEN MODELS A & B
    else if (category == 'C_aqBE') then
      open (uC_aqBE,file=filename,status='unknown',err=100)
      C_aqBE = .true.
    else if (category == 'C_rqBE') then
      open (uC_rqBE,file=filename,status='unknown',err=100)
      C_rqBE = .true.
    else if (category == 'Crq_comp') then
      open (uCrq_comp,file=filename,status='unknown',err=100)
      Crq_comp = .true.
    else if (category == 'chi2') then
      open (uchi2,file=filename,status='unknown',err=100)
      chi_2 = .true.
!    else
!      write(*,1500) category,filename
    end if
  end do
  1 close(usupr)

!   CHECK TO SEE IF REQUIRED INPUT/OUTPUT FILES ARE OPEN IN THE SUPERFILE
  ! INPUT
  if (.not. a_nom) then
    write(*,1510) 'a_nom';   stop
  end if
  if (.not. r_mea) then
    write(*,1510) 'r_mea';   stop
  end if
  if (.not. r_com) then
    write(*,1510) 'r_com';   stop
  end if
  if (.not. C_aa) then
    write(*,1510) 'C_aa';    stop
```

```
end if
if (.not. C_ar) then
  write(*,1510) 'C_ar';     stop
end if
if (.not. C_rr) then
  write(*,1510) 'C_rr';     stop
end if
if (.not. S_ra) then
  write(*,1510) 'S_ra';     stop
end if
if (.not. q_mea) then
  write(*,1510) 'q_mea';    stop
end if
if (.not. q_com) then
  write(*,1510) 'q_com';    stop
end if
if (.not. C_qq) then
  write(*,1510) 'C_qq';     stop
end if
if (.not. C_aq) then
  write(*,1510) 'C_aq';     stop
end if
if (.not. C_rq) then
  write(*,1510) 'C_rq';     stop
end if
if (.not. S_qa) then
  write(*,1510) 'S_qa';     stop
end if

! OUTPUT
if (.not. Crr_comp) then
  write(*,1520) 'Crr_comp';stop
end if
if (.not. a_BE) then
  write(*,1520) 'a_BE';     stop
end if
if (.not. r_BE) then
  write(*,1520) 'r_BE';     stop
end if
if (.not. C_aaBE) then
  write(*,1520) 'C_aaBE';   stop
end if
if (.not. C_rrBE) then
  write(*,1520) 'C_rrBE';   stop
end if
if (.not. C_arBE) then
  write(*,1520) 'C_arBE';   stop
end if
if (.not. Cqq_comp) then
  write(*,1520) 'Cqq_comp';stop
end if
if (.not. q_BE) then
  write(*,1520) 'q_BE';     stop
end if
if (.not. C_qqBE) then
  write(*,1520) 'C_qqBE';   stop
end if
if (.not. Crq_comp) then
  write(*,1520) 'Crq_comp';stop
end if
if (.not. C_aqBE) then
```

```
      write(*,1520) 'C_aqBE';  stop
    end if
    if (.not. C_rqBE) then
      write(*,1520) 'C_rqBE';  stop
    end if
    if (.not. chi_2) then
      write(*,1520) 'chi_2';   stop
    end if

    return

    100 lerr = .true.;ierr = 1;call errmsg;stop
    1500 format(1x,'Invalid file type:',a,' ....skipping',/, &
                1x,'Specified file name:',a)
    1510 format(&
         1x,a,' input file not specified in superfile!')
    1520 format(&
         1x,a,' output file not specified in superfile!')

  END SUBROUTINE filescase3

  !+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

  SUBROUTINE filescase4()

  !*****************************************************************************
  !*                                                                           *
  !* open needed files for read and write for case 4: "Two-Model" Case:        *
  !* predictive modeling for Model A coupled with Model B.                     *
  !*                                                                           *
  !* Open input/output files for multi-pred using super file format            *
  !*                                                                           *
  !* Category     I/O     Description                                          *
  !* --------    ------   -----------                                          *
  !* 'dims'     input    defines the Case selection and dimensions control     *
  !* 'a_nom'    input    nominal values of Na parameters of Model A            *
  !* 'r_mea'    input    nominal values of Nr measured responses of Model A    *
  !* 'r_com'    input    nominal values of Nr computed responses of Model A    *
  !* 'C_aa'     input    covariance matrix of Na parameters of Model A         *
  !* 'C_ar'     input    correlations between Na parameters and Nr responses of *
  !*                     Model A                                               *
  !* 'C_rr'     input    covariance matrix of Nr responses of Model A          *
  !* 'S_ra'     input    sensitivities of Model A                              *
  !*                                                                           *
  !* 'b_nom'    input    nominal values of Nb parameters of Model B            *
  !* 'q_mea'    input    nominal values of Nq measured responses of Model B    *
  !* 'q_com'    input    nominal values of Nq computed responses of Model B    *
  !* 'C_bb'     input    covariance matrix of Nb parameters of Model B         *
  !* 'C_bq'     input    correlations between Nb parameters and Nq responses of *
  !*                     Model B                                               *
  !* 'C_qq'     input    covariance matrix of Nq responses of Model B          *
  !* 'S_qb'     input    sensitivities of Nq responses of Model B w.r.t. Nb    *
  !*                     parameters of Model B                                 *
  !*                                                                           *
  !* 'C_ab'     input    correlations between Na parameters of Model A and Nb  *
  !*                     parameters of Model B                                 *
  !* 'C_aq'     input    correlations between Na parameters of Model A and Nq  *
  !*                     responses of Model B                                  *
  !* 'C_br'     input    correlations between Nb parameters of Model B and Nr  *
  !*                     responses of Model A                                  *
  !* 'C_rq'     input    correlations between Nr responses of Model A and Nq   *
```

```
!*                      responses of Model B                                *
!* 'S_rb'     input     sensitivities of Nr responses of Model A w.r.t. Nb  *
!*                      parameters of Model B                               *
!* 'S_qa'     input     sensitivities of Nq responses of Model B w.r.t. Na  *
!*                      parameters of Model A                               *
!*                                                                          *
!* 'a_BE'     output    best-estimate nominal values of Na parameters of    *
!*                      Model A                                             *
!* 'r_BE'     output    best-estimate nominal values of Nr responses of     *
!*                      Model A                                             *
!* 'C_aaBE'   output    predicted covariance matrix of Na parameters of     *
!*                      Model A                                             *
!* 'C_rrBE'   output    predicted covariance matrix of Nr responses of Model A *
!* 'C_arBE'   output    predicted correlation matrix between the Na parameters *
!*            and Nr responses of Model A                                   *
!* 'Crr_comp' output    covariance matrix of Nr computed responses of Model A  *
!*                                                                          *
!* 'b_BE'     output    best-estimate parameters nominal values of Model B  *
!* 'q_BE'     output    best-estimate response nominal values of Model B   *
!* 'C_bbBE'   output    predicted optimal covariance matrix of parameters of  *
!*                      Model B                                             *
!* 'C_qqBE'   output    predicted covariance matrix of responses of Model B  *
!* 'C_bqBE'   output    predicted correlation matrix for the parameters     *
!*            and responses of Model B                                      *
!* 'Cqq_comp' output    covariance matrix for computed responses of Model B  *
!*                                                                          *
!* 'C_abBE'   output    predicted correlation matrix between Na parameters of  *
!*                      Model A and Nb parameters of Model B                *
!* 'C_aqBE'   output    predicted correlation matrix between Na parameters of  *
!*                      Model A and Nq responses of Model B                 *
!* 'C_brBE'   output    predicted correlation matrix between Nb parameters of  *
!*                      Model B and Nr responses of Model A                 *
!* 'C_rqBE'   output    predicted correlation matrix between Nr responses of  *
!*                      Model A and Nq responses of Model B                 *
!* 'Crq_comp' output    covariance matrix between Nr computed responses of  *
!*                      Model A and Nq computed responses of Model B        *
!* 'chi_2'    output    value of the consistency indicator chi^2            *
!*                                                                          *
!* called by:  Files                                                        *
!* calls  to:  getarg,errmsg                                                *
!*                                                                          *
!****************************************************************************

  IMPLICIT NONE
  !Local-----------------------------------------------------------------------
  CHARACTER(LEN=128) :: filename
  CHARACTER(LEN=8)   :: category

  inquire(unit=usupr)
!  write(*,900) trim(filename)
  do
    read(usupr,*,end=1) category,filename

!   INPUT FILES FOR MODEL A
    if (category == 'a_nom') then
      open(ua_nom,file=filename,status='old',err=100)
      a_nom = .true.
    else if (category == 'r_mea') then
      open(ur_mea,file=filename,status='old',err=100)
      r_mea = .true.
    else if (category == 'r_com') then
```

```fortran
        open (ur_com,file=filename,status='old',err=100)
        r_com = .true.
      else if (category == 'C_aa') then
        open(uC_aa,file=filename,status='old',err=100)
        C_aa = .true.
      else if (category == 'C_ar') then
        open(uC_ar,file=filename,status='old',err=100)
        C_ar = .true.
      else if (category == 'C_rr') then
        open(uC_rr,file=filename,status='old',err=100)
        C_rr = .true.
      else if (category == 'S_ra') then
        open(uS_ra,file=filename,status='old',err=100)
        S_ra = .true.

!     INPUT FILES FOR MODEL B
      else if (category == 'b_nom') then
        open(ub_nom,file=filename,status='old',err=100)
        b_nom = .true.
      else if (category == 'q_mea') then
        open(uq_mea,file=filename,status='old',err=100)
        q_mea = .true.
      else if (category == 'q_com') then
        open (uq_com,file=filename,status='old',err=100)
        q_com = .true.
      else if (category == 'C_bb') then
        open(uC_bb,file=filename,status='old',err=100)
        C_bb = .true.
      else if (category == 'C_bq') then
        open(uC_bq,file=filename,status='old',err=100)
        C_bq = .true.
      else if (category == 'C_qq') then
        open(uC_qq,file=filename,status='old',err=100)
        C_qq = .true.
      else if (category == 'S_qb') then
        open(uS_qb,file=filename,status='old',err=100)
        S_qb = .true.

!     INPUT FILES FOR COUPLED MATRICES BETWEEN MODELS A & B
      else if (category == 'C_ab') then
        open(uC_ab,file=filename,status='old',err=100)
        C_ab = .true.
      else if (category == 'C_aq') then
        open(uC_aq,file=filename,status='old',err=100)
        C_aq = .true.
      else if (category == 'C_br') then
        open(uC_br,file=filename,status='old',err=100)
        C_br = .true.
      else if (category == 'C_rq') then
        open(uC_rq,file=filename,status='old',err=100)
        C_rq = .true.
      else if (category == 'S_rb') then
        open(uS_rb,file=filename,status='old',err=100)
        S_rb = .true.
      else if (category == 'S_qa') then
        open(uS_qa,file=filename,status='old',err=100)
        S_qa = .true.

!     OUTPUT FILES FOR MODEL A
      else if (category == 'a_BE') then
        open (ua_BE,file=filename,status='unknown',err=100)
```

```fortran
   a_BE = .true.
  else if (category == 'r_BE') then
   open (ur_BE,file=filename,status='unknown',err=100)
   r_BE = .true.
  else if (category == 'C_aaBE') then
   open (uC_aaBE,file=filename,status='unknown',err=100)
   C_aaBE = .true.
  else if (category == 'C_rrBE') then
   open (uC_rrBE,file=filename,status='unknown',err=100)
   C_rrBE = .true.
  else if (category == 'C_arBE') then
   open (uC_arBE,file=filename,status='unknown',err=100)
   C_arBE = .true.
  else if (category == 'Crr_comp') then
   open (uCrr_comp,file=filename,status='unknown',err=100)
   Crr_comp = .true.

!   OUTPUT FILES FOR MODEL B
  else if (category == 'b_BE') then
   open (ub_BE,file=filename,status='unknown',err=100)
   b_BE = .true.
  else if (category == 'q_BE') then
   open (uq_BE,file=filename,status='unknown',err=100)
   q_BE = .true.
  else if (category == 'C_bbBE') then
   open (uC_bbBE,file=filename,status='unknown',err=100)
   C_bbBE = .true.
  else if (category == 'C_qqBE') then
   open (uC_qqBE,file=filename,status='unknown',err=100)
   C_qqBE = .true.
  else if (category == 'C_bqBE') then
   open (uC_bqBE,file=filename,status='unknown',err=100)
   C_bqBE = .true.
  else if (category == 'Cqq_comp') then
   open (uCqq_comp,file=filename,status='unknown',err=100)
   Cqq_comp = .true.

!   OUTPUT FILES FOR COUPLED MATRICES BETWEEN MODELS A & B
  else if (category == 'C_abBE') then
   open (uC_abBE,file=filename,status='unknown',err=100)
   C_abBE = .true.
  else if (category == 'C_aqBE') then
   open (uC_aqBE,file=filename,status='unknown',err=100)
   C_aqBE = .true.
  else if (category == 'C_brBE') then
   open (uC_brBE,file=filename,status='unknown',err=100)
   C_brBE = .true.
  else if (category == 'C_rqBE') then
   open (uC_rqBE,file=filename,status='unknown',err=100)
   C_rqBE = .true.
  else if (category == 'Crq_comp') then
   open (uCrq_comp,file=filename,status='unknown',err=100)
   Crq_comp = .true.
  else if (category == 'chi2') then
   open (uchi2,file=filename,status='unknown',err=100)
   chi_2 = .true.
  else
   write(*,1500) category,filename
  end if
 end do
 1 close(usupr)
```

```fortran

!   CHECK TO SEE IF REQUIRED INPUT/OUTPUT FILES ARE OPEN IN THE SUPERFILE
  ! INPUT
  if (.not. a_nom) then
    write(*,1510) 'a_nom';   stop
  end if
  if (.not. r_mea) then
    write(*,1510) 'r_mea';   stop
  end if
  if (.not. r_com) then
    write(*,1510) 'r_com';   stop
  end if
  if (.not. C_aa) then
    write(*,1510) 'C_aa';    stop
  end if
  if (.not. C_ar) then
    write(*,1510) 'C_ar';    stop
  end if
  if (.not. C_rr) then
    write(*,1510) 'C_rr';    stop
  end if
  if (.not. S_ra) then
    write(*,1510) 'S_ra';    stop
  end if
  if (.not. b_nom) then
    write(*,1510) 'b_nom';   stop
  end if
  if (.not. q_mea) then
    write(*,1510) 'q_mea';   stop
  end if
  if (.not. q_com) then
    write(*,1510) 'q_com';   stop
  end if
  if (.not. C_bb) then
    write(*,1510) 'C_bb';    stop
  end if
  if (.not. C_bq) then
    write(*,1510) 'C_bq';    stop
  end if
  if (.not. C_qq) then
    write(*,1510) 'C_qq';    stop
  end if
  if (.not. S_qb) then
    write(*,1510) 'S_qb';    stop
  end if
  if (.not. C_ab) then
    write(*,1510) 'C_ab';    stop
  end if
  if (.not. C_aq) then
    write(*,1510) 'C_aq';    stop
  end if
  if (.not. C_br) then
    write(*,1510) 'C_br';    stop
  end if
  if (.not. C_rq) then
    write(*,1510) 'C_rq';    stop
  end if
  if (.not. S_rb) then
    write(*,1510) 'S_rb';    stop
  end if
  if (.not. S_qa) then
```

```
    write(*,1510) 'S_qa';     stop
  end if

  ! OUTPUT
  if (.not. Crr_comp) then
    write(*,1520) 'Crr_comp';stop
  end if
  if (.not. a_BE) then
    write(*,1520) 'a_BE';     stop
  end if
  if (.not. r_BE) then
    write(*,1520) 'r_BE';     stop
  end if
  if (.not. C_aaBE) then
    write(*,1520) 'C_aaBE';   stop
  end if
  if (.not. C_rrBE) then
    write(*,1520) 'C_rrBE';   stop
  end if
  if (.not. C_arBE) then
    write(*,1520) 'C_arBE';   stop
  end if
  if (.not. Cqq_comp) then
    write(*,1520) 'Cqq_comp';stop
  end if
  if (.not. b_BE) then
    write(*,1520) 'b_BE';     stop
  end if
  if (.not. q_BE) then
    write(*,1520) 'q_BE';     stop
  end if
  if (.not. C_bbBE) then
    write(*,1520) 'C_bbBE';   stop
  end if
  if (.not. C_qqBE) then
    write(*,1520) 'C_qqBE';   stop
  end if
  if (.not. C_bqBE) then
    write(*,1520) 'C_bqBE';   stop
  end if
  if (.not. Crq_comp) then
    write(*,1520) 'Crq_comp';stop
  end if
  if (.not. C_abBE) then
    write(*,1520) 'C_abBE';   stop
  end if
  if (.not. C_aqBE) then
    write(*,1520) 'C_aqBE';   stop
  end if
  if (.not. C_brBE) then
    write(*,1520) 'C_brBE';   stop
  end if
  if (.not. C_rqBE) then
    write(*,1520) 'C_rqBE';   stop
  end if
  if (.not. chi_2) then
    write(*,1520) 'chi_2';    stop
  end if

  return

```

```
 100 lerr = .true.;ierr = 1;call errmsg;stop
 1500 format(1x,'Invalid file type:',a,' ....skipping',/, &
              1x,'Specified file name:',a)
 1510 format(&
       1x,a,' input file not specified in superfile!')
 1520 format(&
       1x,a,' output file not specified in superfile!')

END SUBROUTINE filescase4

!++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

END MODULE ModuleFiles

```

8.7 Subroutine ReadInput.f90

```
SUBROUTINE ReadInput

!****************************************************************************
!*                                                                          *
!* Reads and processes all multi-pred input parameters and data.            *
!*                                                                          *
!****************************************************************************
  !Global--------------------------------------------------------------------
  USE ModuleReadWrite
  USE ModuleLapack, ONLY : is_positive_definite

  IMPLICIT NONE
  !Local---------------------------------------------------------------------
  INTEGER(I4B)          :: i
  CHARACTER(LEN=128)    :: commnt

  ! read dimensions for the vectors and matrices
  read (udims,*,err=100) commnt
  read (udims,*,err=100) Na
  read (udims,*,err=100) commnt
  read (udims,*,err=100) Nr
  write(*, 1010) Na, Nr
  if(CaseNumber /= 1) then
     do i=1, 5
       read(udims,*,err=100) commnt
     end do
     read (udims,*,err=100) Nb
     do i=1, 5
       read(udims,*,err=100) commnt
     end do
     read (udims,*,err=100) Nq
     if(CaseNumber == 2) then
        write(*, 1011) Nb
     else if (CaseNumber == 3) then
        write(*, 1012) Nq
     else if (CaseNumber == 4) then
        write(*, 1013) Nb, Nq
     end if
  end if
  close(udims)

```

```fortran
  write(*, 1000)
  !-----------------------------------------------------------------------
  !READ INPUT FILES FOR MODEL A
  ! read parameters nominal values
  write(*, 1020)
  write(*, 1030)
  allocate(alpha(Na),stat=alloc_err)
  call readVectorFromFile(ua_nom,alpha)

  ! read measured responses nominal values
  write(*, 1040)
  allocate(rm(Nr),stat=alloc_err)
  call readVectorFromFile(ur_mea,rm)

  ! read computed responses nominal values
  write(*, 1050)
  allocate(rc(Nr),stat=alloc_err)
  call readVectorFromFile(ur_com,rc)

  ! read covariance matrices for parameter-parameter
  write(*, 1060)
  allocate(Caa(Na,Na),stat=alloc_err)
  call readMatrixFromFile(uC_aa,Caa)
  ! check if Ca positive definite
  write(*, 1070)
  if(is_positive_definite(Caa) == 0) then
     write(*, 1080)
  else
     write(*, 1090); stop
  end if

! read covariance matrices for parameter-response
  write(*, 1100)
  allocate(Car(Na,Nr),stat=alloc_err)
  call readMatrixFromFile(uC_ar,Car)

  ! read covariance matrices for response-response
  write(*, 1110)
  allocate(Crr(Nr,Nr),stat=alloc_err)
  call readMatrixFromFile(uC_rr,Crr)
  ! check if Crr positive definite
  write(*, 1120)
  if(is_positive_definite(Crr) == 0) then
     write(*, 1130);
  else
     write(*, 1140); stop
  end if

  ! read sensitivities
  write(*, 1150)
  allocate(Sra(Nr,Na),stat=alloc_err)
  call readMatrixFromFile(uS_ra,Sra)

  ! for Case 2: additional read inputs to Case 1
  if(CaseNumber == 2) then
      !-----------------------------------------------------------------------
      !READ INPUT FILES FOR ADDITIONAL PARAMETERS
      ! read parameters nominal values
      write(*, 2000)
      write(*, 2010)
      allocate(beta(Nb),stat=alloc_err)
```

```
      call readVectorFromFile(ub_nom,beta)

      ! read covariance matrices for parameter-parameter
      write(*, 2020)
      allocate(Cbb(Nb,Nb),stat=alloc_err)
      call readMatrixFromFile(uC_bb,Cbb)
      ! check if Cb positive definite
      write(*, 2030)
      if(is_positive_definite(Cbb) == 0) then
         write(*, 2040)
      else
         write(*, 2050); stop
      end if

      !-----------------------------------------------------------------------
      !READ INPUT FILES FOR THE COUPLED MATRICES
      ! read correlations between parameters a and b
      write(*, 2060)
      write(*, 2070)
      allocate(Cab(Na,Nb),stat=alloc_err)
      call readMatrixFromFile(uC_ab,Cab)

      ! read correlations between parameters b and responses r
      write(*, 2080)
      allocate(Cbr(Nb,Nr),stat=alloc_err)
      call readMatrixFromFile(uC_br,Cbr)

      ! read sensitivities of responses r and parameters b
      write(*, 2090)
      allocate(Srb(Nr,Nb),stat=alloc_err)
      call readMatrixFromFile(uS_rb,Srb)

   end if !case 2

   ! for Case 3: additional read inputs to Case 1
   if(CaseNumber == 3) then
      !-----------------------------------------------------------------------
      !READ INPUT FILES FOR ADDITIONAL RESPONSES
      write(*, 3000)

      ! read measured responses nominal values
      write(*, 3010)
      allocate(qm(Nq),stat=alloc_err)
      call readVectorFromFile(uq_mea,qm)

      ! read computed responses nominal values
      write(*, 3020)
      allocate(qc(Nq),stat=alloc_err)
      call readVectorFromFile(uq_com,qc)

      ! read covariance matrices for response-response
      write(*, 3030)
      allocate(Cqq(Nq,Nq),stat=alloc_err)
      call readMatrixFromFile(uC_qq,Cqq)
      ! check if Cqq positive definite
      write(*, 3040)
      if(is_positive_definite(Cqq) == 0) then
         write(*, 3050)
      else
         write(*, 3060); stop
      end if
```

```
    !-------------------------------------------------------------------
    !READ INPUT FILES FOR THE COUPLED MATRICES
    write(*, 3070)

    ! read correlations between parameters a and responses q
    write(*, 3080)
    allocate(Caq(Na,Nq),stat=alloc_err)
    call readMatrixFromFile(uC_aq,Caq)

    ! read correlations between responses r and responses q
    write(*, 3090)
    allocate(Crq(Nr,Nq),stat=alloc_err)
    call readMatrixFromFile(uC_rq,Crq)

    ! read sensitivities of responses q and parameters a
    write(*, 3100)
    allocate(Sqa(Nq,Na),stat=alloc_err)
    call readMatrixFromFile(uS_qa,Sqa)
end if !case 3

! for Case 4: additional read inputs to Case 1
if(CaseNumber == 4) then
    !-------------------------------------------------------------------
    !READ INPUT FILES FOR MODEL B
    ! read parameters nominal values
    write(*, 4000)
    write(*, 4010)
    allocate(beta(Nb),stat=alloc_err)
    call readVectorFromFile(ub_nom,beta)

    ! read measured responses nominal values
    write(*, 4020)
    allocate(qm(Nq),stat=alloc_err)
    call readVectorFromFile(uq_mea,qm)

    ! read computed responses nominal values
    write(*, 4030)
    allocate(qc(Nq),stat=alloc_err)
    call readVectorFromFile(uq_com,qc)

    ! read covariance matrices for parameter-parameter
    write(*, 4040)
    allocate(Cbb(Nb,Nb),stat=alloc_err)
    call readMatrixFromFile(uC_bb,Cbb)
    ! check if Cb positive definite
    write(*, 4050)
    if(is_positive_definite(Cbb) == 0) then
       write(*, 4060)
    else
       write(*, 4070); stop
    end if

    ! read covariance matrices for parameter-response
    write(*, 4080)
    allocate(Cbq(Nb,Nq),stat=alloc_err)
    call readMatrixFromFile(uC_bq,Cbq)

    ! read covariance matrices for response-response
    write(*, 4090)
    allocate(Cqq(Nq,Nq),stat=alloc_err)
```

```
      call readMatrixFromFile(uC_qq,Cqq)
      ! check if Cqq positive definite
      write(*, 4100)
      if(is_positive_definite(Cqq) == 0) then
         write(*, 4110)
      else
         write(*, 4120); stop
      end if

      ! read sensitivities
      write(*, 4130)
      allocate(Sqb(Nq,Nb),stat=alloc_err)
      call readMatrixFromFile(uS_qb,Sqb)

      !-----------------------------------------------------------------------
      !READ INPUT FILES FOR THE COUPLED MATRICES BETWEEN MODELS A & B
      ! read correlations between parameters a and b
      write(*, 4140)
      write(*, 4150)
      allocate(Cab(Na,Nb),stat=alloc_err)
      call readMatrixFromFile(uC_ab,Cab)

      ! read correlations between parameters a and responses q
      write(*, 4160)
      allocate(Caq(Na,Nq),stat=alloc_err)
      call readMatrixFromFile(uC_aq,Caq)

      ! read correlations between parameters b and responses r
      write(*, 4170)
      allocate(Cbr(Nb,Nr),stat=alloc_err)
      call readMatrixFromFile(uC_br,Cbr)

      ! read correlations between responses r and responses q
      write(*, 4180)
      allocate(Crq(Nr,Nq),stat=alloc_err)
      call readMatrixFromFile(uC_rq,Crq)

      ! read sensitivities of responses r and parameters b
      write(*, 4190)
      allocate(Srb(Nr,Nb),stat=alloc_err)
      call readMatrixFromFile(uS_rb,Srb)

      ! read sensitivities of responses q and parameters a
      write(*, 4200)
      allocate(Sqa(Nq,Na),stat=alloc_err)
      call readMatrixFromFile(uS_qa,Sqa)
  end if !case 4

  return
   100 lerr = .true.;ierr = 1;call errmsg;stop
  1010 format(/, &
       1x,'For model A:   Na =' ,i5,'     Nr =', i5)
  1011 format(&
       1x,'Nb =' ,i5,' parameters added to the Na parameters for Model A',a)
  1012 format(&
       1x,'Nq =', i5,' responses added to the Nr reponses for Model A',a)
  1013 format(&
       1x,'For model B:   Nb =' ,i5,'     Nq =', i5)
  1000 format(/, &
       1x,'---------------- Read Inputs into Vectors/Matrices ---------------',a)
  ! format reading for model A
```

```
1020 format(&
     1x,':: Reading inputs for Model A:',a)
1030 format(&
     1x,'Reading the input for the parameter vector .................  alpha',a)
1040 format(&
     1x,'Reading the input for measured response vector ................. rm',a)
1050 format(&
     1x,'Reading the input for computed response vector ................. rc',a)
1060 format(&
     1x,'Reading the input for covariance matrix of parameters .......... Caa',a)
1070 format(&
     1x,'        Factorizing matrix Caa -- to test if positive definite.',a)
1080 format(&
     1x,'        OK, Caa is positive definite.',a)
1090 format(&
     1x,'ERROR: The input covariance matrix Caa is not positive'        &
        'definite, check the input file',a)
1100 format(&
     1x,'Reading the input for the correlation matrix ................... Car',a)
1110 format(&
     1x,'Reading the input for covariance matrix of responses ........... Crr',a)
1120 format(&
     1x,'        Factorizing matrix Crr -- to test if positive definite.',a)
1130 format(&
     1x,'        OK, Crr is positive definite.',a)
1140 format(&
     1x,'ERROR: The input covariance matrix Crr is not positive'         &
        'definite, check the input file',a)
1150 format(&
     1x,'Reading the input for the sensitivity matrix ................... Sra',a)

! Case 2 -- format reading for Nb additional parameters
2000 format(/,&
     1x,':: Reading inputs for additional model parameters for Case 2:',a)
2010 format(&
     1x,'Reading the input for the parameter vector .....................beta',a)
2020 format(&
     1x,'Reading the input for covariance matrix of parameters .......... Cbb',a)
2030 format(&
     1x,'        Factorizing matrix Cbb -- to test if positive definite.',a)
2040 format(&
     1x,'        OK, Cbb is positive definite.',a)
2050 format(&
     1x,'ERROR: The input covariance matrix Cbb is not positive'         &
        'definite, check the input file',a)

! Case 2 -- format reading for the coupled matrices
2060 format(/,&
     1x,':: Reading inputs for coupled matrices for Case 2:',a)
2070 format(&
     1x,'Reading the input for matrix ................................... Cab',a)
2080 format(&
     1x,'Reading the input for matrix ................................... Cbr',a)
2090 format(&
     1x,'Reading the input for matrix ................................... Srb',a)

! Case 3 -- format reading for additional responses
3000 format(/,&
     1x,':: Reading inputs for additional model responses for Case 3: ',a)
3010 format(&
     1x,'Reading the input for additional measured response vector ...... qm',a)
```

```
3020 format(&
     1x,'Reading the input for additional computed response vector ...... qc',a)
3030 format(&
     1x,'Reading the input for covariance matrix of responses ........... Cqq',a)
3040 format(&
     1x,'        Factorizing matrix Cqq -- to test if positive definite.',a)
3050 format(&
     1x,'        OK, Cqq is positive definite.',a)
3060 format(&
     1x,'ERROR: The input covariance matrix Cqq is not positive'         &
        'definite, check the input file',a)

! Case 3 -- format reading for the coupled matrices
3070 format(/,&
     1x,':: Reading inputs for coupled matrices for Case 3:',a)
3080 format(&
     1x,'Reading the input for matrix ................................... Caq',a)
3090 format(&
     1x,'Reading the input for matrix ................................... Crq',a)
3100 format(&
     1x,'Reading the input for matrix ................................... Sqa',a)

! Case 4 -- format reading for model B
4000 format(/,&
     1x,':: Reading inputs for Model B: ',a)
4010 format(&
     1x,'Reading the input for the parameter vector .....................beta',a)
4020 format(&
     1x,'Reading the input for measured response vector ................. qm',a)
4030 format(&
     1x,'Reading the input for computed response vector ................. qc',a)
4040 format(&
     1x,'Reading the input for covariance matrix of parameters .......... Cbb',a)
4050 format(&
     1x,'        Factorizing matrix Cbb -- to test if positive definite.',a)
4060 format(&
     1x,'        OK, Cbb is positive definite.',a)
4070 format(&
     1x,'ERROR: The input covariance matrix Cbb is not positive'         &
        'definite, check the input file',a)
4080 format(&
     1x,'Reading the input for the correlation matrix .................. Cbq',a)
4090 format(&
     1x,'Reading the input for covariance matrix of responses ........... Cqq',a)
4100 format(&
     1x,'        Factorizing matrix Cqq -- to test if positive definite.',a)
4110 format(&
     1x,'        OK, Cqq is positive definite.',a)
4120 format(&
     1x,'ERROR: The input covariance matrix Cqq is not positive'         &
        'definite, check the input file',a)
4130 format(&
     1x,'Reading the input for the sensitivity matrix .................. Sqb',a)

! Case 4 -- format reading for the coupled matrices between models A & B
4140 format(/,&
     1x,':: Reading inputs for coupled matrices between Models A & B:',a)
4150 format(&
     1x,'Reading the input for matrix ................................... Cab',a)
4160 format(&
     1x,'Reading the input for matrix ................................... Caq',a)
```

```
4170 format(&
     1x,'Reading the input for matrix ................................... Cbr',a)
4180 format(&
     1x,'Reading the input for matrix ................................... Crq',a)
4190 format(&
     1x,'Reading the input for matrix ................................... Srb',a)
4200 format(&
     1x,'Reading the input for matrix ................................... Sqa',a)

END SUBROUTINE ReadInput
```

8.8 Module ModuleReadWrite.f90

```
MODULE ModuleReadWrite
!*****************************************************************************
!*                                                                           *
!* Module ModuleReadWrite encapsulates subroutines for:                      *
!*                     readMatrixFromFile (UnitNum, Array)                   *
!*                     readVectorFromFile (UnitNum, Array)                   *
!*                     writeVectorToFile (UnitNum, Array)                    *
!*                     writeMatrixToFile (UnitNum, Array)                    *
!*                                                                           *
!*****************************************************************************

  !Global-------------------------------------------------------------------
  USE ModuleErrors
  USE ModuleIO
  USE ModuleGlobalParameters

  IMPLICIT NONE

CONTAINS

SUBROUTINE readMatrixFromFile (UnitNum, Array)

!*****************************************************************************
!*                                                                           *
!* read a 2D sparse matrix and to produce a normal one                       *
!*                                                                           *
!*****************************************************************************
  IMPLICIT NONE

  ! Arguments---------------------------------------------------------------
    INTEGER, intent(in)    :: UnitNum
    REAL(DP), ALLOCATABLE :: Array(:,:)
  !Local--------------------------------------------------------------------
    INTEGER(I4B)          :: numRows, numCols, NonzeroElements, Nrow, Ncol
    INTEGER(I4B)          :: i, j, k
    INTEGER(I4B)          :: IOstatus
    REAL(DP)              :: val

    Nrow = size(Array,1)
    Ncol = size(Array,2)
    rewind(UnitNum)
    read (UnitNum,*,err=100) numRows,numCols,NonzeroElements
    if(numRows /= Nrow .OR. numCols /= Ncol) then
      write(*, 200) Nrow,Ncol; stop
```

```fortran
    end if
    Array = 0.0
    do k=1, NonzeroElements
       read (UnitNum,*,IOSTAT=IOstatus) i, j, Array(i,j)
       if(Iostatus > 0) then !something is wrong, like illegal values
            Write(*, 210) K+1
            stop
       else if (Iostatus < 0) then    !end of file reached
            Write(*, 220) NonzeroElements, K-1
            stop
       else   !normal reading
          if(i > Nrow .OR. j > Ncol) then
            write(*, 300); stop
          end if
       end if
    end do
    ! read one more line, to check if end of file reached.
    ! if not, then the input data file has a inconsistency.
    ! It has more data than expected.
    read (UnitNum,*,IOSTAT=IOstatus) i,j,val
    if(IOstatus == 0) then
       write(*, 310)NonzeroElements
       stop
    end if

    close(UnitNum)
    return
    ! bail out for read error
    100 lerr = .true.;ierr = 2;call errmsg;stop
    200 format(&
       1x,'Error: this matrix should have a dimension of ',i8,'     -by-', i8)
    210 format(&
       1x,'Error: something is wrong while reading line ',i8,'. Maybe illegal',/,&
          '        data; Check the input.',a)
    220 format(&
       1x,'Error: end of file reached earlier than expected. It is expected ',/,&
          '        to read Nz = ', i8,' nonzero elements, but only read ',i8 ,/,&
          '        of them. Check the input.',a)
    300 format(&
       1x,'Error: the index for nonzero elements exceeds the matrix size.',a)
    310 format(&
       1x,'Error: something is wrong with the input data file. It seems the',/,&
          '        actual number of nonzero elements in the file exceeds '  ,/,&
          '        that defined in the 1st line: Nz = ',i8,'. Check the '   ,/,&
          '        input.',a)

END SUBROUTINE readMatrixFromFile

!++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

SUBROUTINE readVectorFromFile(UnitNum, Array)

!******************************************************************************
!*                                                                            *
!* read a 1D sparse matrix and to produce a normal one                        *
!*                                                                            *
!******************************************************************************
  IMPLICIT NONE

  ! Arguments------------------------------------------------------------
    INTEGER, intent(in)   :: UnitNum
```

```
    REAL(DP), ALLOCATABLE :: Array(:)
  !Local-------------------------------------------------------------------
    INTEGER(I4B)          :: numRows, numCols, NonzeroElements, Nrow, Ncol
    INTEGER(I4B)          :: i, j, k
    INTEGER(I4B)          :: IOstatus
    REAL(DP)              :: val

    Nrow = size(Array,1)
    Ncol = 1
    rewind(UnitNum)
    read (UnitNum,*,err=100) numRows,numCols,NonzeroElements
    Array = 0.0
    if(numRows /= Nrow .OR. numCols /= 1) then
      write(*, 200) Nrow,Ncol; stop
    end if
    do k=1, NonzeroElements
       read (UnitNum,*,IOSTAT=IOstatus) i, j, Array(i)
       if(Iostatus > 0) then !something is wrong, like illegal values
            Write(*, 210) K+1
            stop
       else if (Iostatus < 0) then !end of file reached earlier than expected
            Write(*, 220) NonzeroElements, K-1
            stop
       else   !normal reading
          if(i > Nrow .OR. j > Ncol) then
            write(*, 300); stop
          end if
       end if
    end do
    ! read one more line, to check if end of file reached
    ! if not, then the input data file has a inconstancy.
    ! It has more data than expected.
    read (UnitNum,*,IOSTAT=IOstatus) i,j,val
    if(IOstatus == 0) then
       write(*, 310)NonzeroElements
       stop
    end if

    close(UnitNum)
    return
    ! bail out for read error
    100 lerr = .true.;ierr = 2;call errmsg;stop
    200 format(&
       1x,'Error: this vector should have a dimension of ',i8,'     -by-', i8)
    210 format(&
       1x,'Error: something is wrong while reading line ',i8,'. Maybe illegal',/,&
          '        data; Check the input.',a)
    220 format(&
       1x,'Error: end of file reached earlier than expected. It is expected ',/,&
          '        to read Nz = ', i8,' nonzero elements, but only read ',i8 ,/,&
          '        of them. Check the input.',a)
    300 format(&
       1x,'Error: the index for nonzero elements exceeds the vector size.',a)
    310 format(&
       1x,'Error: something is wrong with the input data file. It seems the',/,&
          '        actual number of nonzero elements in the file exceeds '  ,/,&
          '        that defined in the 1st line: Nz = ',i8,'. Check the '   ,/,&
          '        input.',a)

END SUBROUTINE readVectorFromFile
```

```
!++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

SUBROUTINE writeVectorToFile(UnitNum, Array)

!****************************************************************************
!*                                                                          *
!* write a full vector into a file as a sparse one                          *
!*                                                                          *
!****************************************************************************
  IMPLICIT NONE

  ! Arguments-------------------------------------------------------------------
    INTEGER, intent(in)    :: UnitNum
    REAL(DP), ALLOCATABLE :: Array(:)
  !Local------------------------------------------------------------------------
    INTEGER(I4B)           :: numRows, numCols, nZ
    INTEGER(I4B)           :: i, j

    numRows = size(Array,1)
    numCols = 1
    nZ = 0

 ! write the 1st line for numRows, numCols and the number of nonzero elements
    do i=1, numRows
       do j=1, numCols
          if(Array(i)/= 0.0) then
            nZ = nZ + 1
          end if
       end do
    end do
    write(UnitNum,1000,err=200) numRows, numCols, nZ

 ! write the nonzero elements with their associated row and colume coordinates
    do i=1, numRows
       do j=1, numCols
          if(Array(i)/= 0.0) then
            write(UnitNum,2000,err=200) i, j, Array(i)
          end if
       end do
    end do

    close(UnitNum)
    return
  ! bail out for write error
     200 lerr = .true.;ierr = 3;call errmsg;stop
    1000 format(3I5)
    2000 format(2I5,ES20.8)

END SUBROUTINE writeVectorToFile

!++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

SUBROUTINE writeMatrixToFile(UnitNum, Array)

!****************************************************************************
!*                                                                          *
!* write a 2D full matrix into a file as a sparse one                       *
!*                                                                          *
!****************************************************************************
  IMPLICIT NONE

```

```
  ! Arguments-----------------------------------------------------------------
    INTEGER, intent(in)   :: UnitNum
    REAL(DP), ALLOCATABLE :: Array(:,:)
  !Local-------------------------------------------------------------------
    INTEGER(I4B)          :: numRows, numCols, nZ
    INTEGER(I4B)          :: i, j

    numRows = size(Array,1)
    numCols = size(Array,2)
    nZ = 0

 ! write the 1st line for numRows, numCols and the number of nonzero elements
    do i=1, numRows
       do j=1, numCols
          if(Array(i,j)/= 0.0) then
            nZ = nZ + 1
          end if
       end do
    end do
    write(UnitNum,1000,err=200) numRows, numCols, nZ

 ! write the nonzero elements with their associated row and colume coordinates
    do i=1, numRows
       do j=1, numCols
          if(Array(i,j)/= 0.0) then
            write(UnitNum,2000,err=200) i, j, Array(i,j)
          end if
       end do
    end do

    close(UnitNum)
    return
  ! bail out for write error
     200 lerr = .true.;ierr = 3;call errmsg;stop
    1000 format(I5,I5,I8)
    2000 format(2I5,ES20.8)

END SUBROUTINE writeMatrixToFile

!  ====================================================================

  LOGICAL FUNCTION is_NAN_or_Infinity_M(A) result(tf)

!  -- chech the matrix component values not to be Inifinite or NAN
!  -- where A is a 2D matrix
!  -- developed by University of South Carolina.
!
    INTEGER, PARAMETER :: DP = KIND(1.0D0)
    INTEGER, PARAMETER :: I4B = SELECTED_INT_KIND(9)
  ! Arguments-----------------------------------------------------------------
    REAL(DP), dimension(:,:), intent(in) :: A
!    LOGICAL                              :: tf
  ! Local-------------------------------------------------------------------
    REAL(DP)                         :: infinity
    INTEGER(I4B)                     :: numRows, numCols, i, j

    infinity     = 1.e100_dp
    tf = .false.

    numRows = size(A,1)
    numCols = size(A,2)
```

```

     do i=1, numRows
       do j=1, numCols
          !check if infinity
          if(A(i,j) > infinity) then
            tf = .true.
          end if
          ! check if NAN
          if(A(i,j) /= A(i,j)) then
            tf = .true.
          end if
       end do
    end do

  END FUNCTION is_NAN_or_Infinity_M

!  =====================================================================

  LOGICAL FUNCTION is_NAN_or_Infinity_V(A) result(tf)

!  -- chech the matrix component values not to be Inifinite or NAN
!  -- where A is a Vector
!  -- developed by University of South Carolina.
!
    INTEGER, PARAMETER :: DP = KIND(1.0D0)
    INTEGER, PARAMETER :: I4B = SELECTED_INT_KIND(9)
  ! Arguments-----------------------------------------------------------------
    REAL(DP), dimension(:), intent(in)   :: A
!    LOGICAL                              :: tf
  ! Local---------------------------------------------------------------------
    REAL(DP)                        :: infinity
    INTEGER(I4B)                    :: numRows, i, j

    infinity     = 1.e100_dp
    tf = .false.

    numRows = size(A,1)

     do i=1, numRows
          !check if infinity
          if(A(i) > infinity) then
            tf = .true.
          end if
          ! check if NAN
          if(A(i) /= A(i)) then
            tf = .true.
          end if
    end do

  END FUNCTION is_NAN_or_Infinity_V

END MODULE ModuleReadWrite
```

8.9 Subroutine MultiPredSolver.f90

```
SUBROUTINE MultiPredSolver

!***************************************************************************
!*                                                                         *
!* apply the Multi-Pred formulations and solve                             *
!*                                                                         *
!***************************************************************************

  !Global------------------------------------------------------------------
  USE ModuleMultiPred

  IMPLICIT NONE

  ! call Multi-Pred solver for Case 1: Modle A solely
  if(CaseNumber == 1) then
    call solvercase1
  end if

  ! call Multi-Pred solver for Case 2: Modle A with additional Nb parameters
  if(CaseNumber == 2) then
    call solvercase2
  end if

  ! call Multi-Pred solver for Case 3: Modle A with additional Nc responses
  if(CaseNumber == 3) then
    call solvercase3
  end if

  ! call Multi-Pred solver for Case 4: Coupled Model A and Model B
  if(CaseNumber == 4) then
    call solvercase4
  end if


END SUBROUTINE MultiPredSolver
```

8.10 Module ModuleMultiPred.f90

```
  MODULE ModuleMultiPred
  !***************************************************************************
  !*                                                                         *
  !* Module ModuleMultiPred encapsulates subroutines for:                    *
  !*                   solvercase1 ()                                        *
  !*                   solvercase2 ()                                        *
  !*                   solvercase3 ()                                        *
  !*                   solvercase4 ()                                        *
  !*                                                                         *
  !***************************************************************************

    !Global----------------------------------------------------------------
    USE ModuleReadWrite
    USE ModuleLapack

    IMPLICIT NONE

```

```fortran
CONTAINS

!++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
SUBROUTINE solvercase1()
!
!  solver for Casee 1: predictive modeling for Model A solely
!
  !Local-----------------------------------------------------------------
  REAL(DP), ALLOCATABLE :: SraCaa(:,:), SraT(:,:)
  REAL(DP), ALLOCATABLE :: Xa(:,:), Xr(:,:), Drr(:,:), D11(:,:), rd(:)
  REAL(DP), ALLOCATABLE :: CarT(:,:)
  REAL(DP), ALLOCATABLE :: Drrinv(:,:)
  REAL(DP), ALLOCATABLE :: XaD11(:,:), XrD11(:,:)
  REAL(DP), ALLOCATABLE :: D11rd(:)

  write(*,1000)

  !start with computing covariance matrix of Crr_comp
  !for the computed responses.

  ! Crrcomp = Sra*Caa*Sra'
  allocate(Crrcomp(Nr,Nr),stat=alloc_err)
  allocate(SraCaa(Nr,Na),stat=alloc_err)
  allocate(SraT(Na,Nr),stat=alloc_err)
  Crrcomp  = 0.0;   SraCaa   = 0.0
  SraT     = 0.0;

  write(*, 1010)
  SraCaa   = multipMM(Sra,Caa)
  SraT     = transpose(Sra)
  Crrcomp  = multipMM(SraCaa,SraT)
  call writeMatrixToFile(uCrr_comp,Crrcomp)

  ! define intermediate quantities and initialize
  write(*, 1020)
  allocate(Xa(Na,Nr),stat=alloc_err)
  allocate(Xr(Nr,Nr),stat=alloc_err)
  allocate(Drr(Nr,Nr),stat=alloc_err)
  allocate(D11(Nr,Nr),stat=alloc_err)
  allocate(rd(Nr),stat=alloc_err)
  Xa    = 0.0
  Xr    = 0.0
  Drr   = 0.0
  D11   = 0.0
  rd    = 0.0

  ! Xa=Caa*Sra'-Car
  Xa = multipMM(Caa,SraT)
  Xa = Xa - Car

  ! Xr=Car'*Sra'-Crr
  allocate(CarT(Nr,Na),stat=alloc_err)
  CarT = 0.0
  CarT = transpose(Car)

  Xr = multipMM(CarT,SraT)
  Xr = Xr - Crr

  ! Drr=Sra*Xa-Xr
  Drr   = multipMM(Sra,Xa)
  Drr   = Drr - Xr
```

```
    ! define rd
    rd = rc -rm

    ! compute Drr^-1, D22^-1
    allocate(Drrinv(Nr,Nr),stat=alloc_err)
    Drrinv = 0.0
    Drrinv = inv(Drr)
    if(is_NAN_or_Infinity_M(Drr)) then
       write(*, 1030)
    end if
    if(is_NAN_or_Infinity_M(Drrinv)) then
       write(*, 1040)
    end if

    ! D11=Drr^-1
    D11  = Drrinv

    ! best estimated mean values for aBE, rBE
    write(*, 1050)
    allocate(aBE(Na),stat=alloc_err)
    allocate(rBE(Nr),stat=alloc_err)
    aBE = 0.0
    rBE = 0.0

    ! aBE = alpha-[Xa*D11]*rd
    allocate(XaD11(Na,Nr),stat=alloc_err)
    XaD11 = 0.0
    XaD11 = multipMM(Xa,D11)

    aBE = multipMV(XaD11,rd)
    aBE = alpha - aBE
    call writeVectorToFile(ua_BE,aBE)

    ! rBE = rm-[Xr*D11]*rd
    write(*, 1060)
    allocate(XrD11(Nr,Nr),stat=alloc_err)
    XrD11 = 0.0
    XrD11 = multipMM(Xr,D11)

    rBE = multipMV(XrD11,rd)
    rBE = rm - rBE
    call writeVectorToFile(ur_BE,rBE)

    !calculate coviances for responses and parameters
    allocate(CaaBE(Na,Na),stat=alloc_err)
    allocate(CrrBE(Nr,Nr),stat=alloc_err)
    allocate(CarBE(Na,Nr),stat=alloc_err)
    CaaBE = 0.0; CrrBE = 0.0; CarBE = 0.0

    ! CaaBE = Caa - [Xa*(D11*Xa')]
    write(*, 1070)
    CaaBE = multipMM(Xa,transpose(XaD11))
    CaaBE = Caa - CaaBE
    call writeMatrixToFile(uC_aaBE,CaaBE)

    ! CrrBE = Crr - [Xr*(D11*Xr')]
    write(*, 1080)
    CrrBE = multipMM(Xr,transpose(XrD11))
    CrrBE = Crr - CrrBE
    call writeMatrixToFile(uC_rrBE,CrrBE)
```

```
  ! CarBE = Car - [Xa*(D11*Xr')]
  write(*, 1090)
  CarBE = multipMM(Xa,transpose(XrD11))
  CarBE = Car - CarBE
  call writeMatrixToFile(uC_arBE,CarBE)

  !calculate the "consistency indicator" chi^2
  allocate(D11rd(Nr),stat=alloc_err)
  D11rd     = 0.0
  chi2      = 0.0

  D11rd    = multipMV(D11,rd)
  chi2     = multipVV(rd,D11rd)
  write(uchi2,3500,err=200) chi2, chi2/Nr

  write(*,3600)
  return
   200 lerr = .true.;ierr = 3;call errmsg;stop
  1000 format(/, &
       1x,'-----------------  Multi-pred Solving & Output  ----------------')
  1010 format(&
       1x,'computing Crrcomp = Sra*Caa*Sra"',a)
  1020 format(&
       1x,'computing Xa, Xr, D11, Drr, Drr^-1, rd',a)
  1030 format(&
       1x,'ERROR: Infinite or NAN in computing Drr, where '                    &
          'Drr=Sra*Xa-(CarT)*(SraT)-Crr',a)
  1040 format(&
       1x,'ERROR: Infinite or NAN in computing Drr^-1, where '                 &
          'Drr=Sra*Xa-(CarT)*(SraT)-Crr',a)
  1050 format(&
       1x,'computing aBE   = alpha-[Caa*Sra"-Car]*Drr^-1*rd',a)
  1060 format(&
       1x,'computing rBE   = rm-[(Car")*(Sra")-Crr]*Drr^-1*rd',a)
  1070 format(&
       1x,'computing CaaBE = Caa-[Caa*Sra"-Car]*Drr^-1*[Caa*Sra"-Car]"',a)
  1080 format(&
       1x,'computing CrrBE = Crr-[Car"*Sra"-Crr]*Drr^-1*[Car"*Sra"-Crr]"',a)
  1090 format(&
       1x,'computing CarBE = Car-[Caa*Sra"-Car]*Drr^-1*[Car"*Sra"-Crr]"',a)
  3500 format(&
          'chi^2                                 ='                  ,F8.3/, &
          'chi^2_d = (chi^2)/(number of responses) ='                ,F8.3)
  3600 format(/, &
       1x,'done.')

END SUBROUTINE solvercase1

!++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

SUBROUTINE solvercase2()
!
!  solver for Casee 2: predictive modeling for Model A with Nb additional
!                     parameters, but no additional responses
!
  !Local-----------------------------------------------------------------------
  REAL(DP), ALLOCATABLE :: SraCaa(:,:), SraT(:,:), SraCab(:,:), SrbT(:,:),    &
                           SrbCbb(:,:)
  REAL(DP), ALLOCATABLE :: Xa(:,:), Xb(:,:), Xr(:,:), Drr(:,:), D11(:,:), rd(:)
  REAL(DP), ALLOCATABLE :: CarT(:,:), CbrT(:,:)
```

```
REAL(DP), ALLOCATABLE :: Drrinv(:,:), XaD11(:,:), XbD11(:,:), XrD11(:,:)
REAL(DP), ALLOCATABLE :: D11rd(:)

write(*,1000)

!start with computing covariance matrix of Crr_comp
!for the computed responses.

! Crrcomp = Sra*Caa*Sra'+2*Sra*Cab*Srb'+Srb*Cbb*Srb'
allocate(Crrcomp(Nr,Nr),stat=alloc_err)
allocate(SraCaa(Nr,Na),stat=alloc_err)
allocate(SraT(Na,Nr),stat=alloc_err)
allocate(SraCab(Nr,Nb),stat=alloc_err)
allocate(SrbT(Nb,Nr),stat=alloc_err)
allocate(SrbCbb(Nr,Nb),stat=alloc_err)
Crrcomp  = 0.0;    SraCaa    = 0.0
SraT     = 0.0;    SraCab    = 0.0
SrbT     = 0.0;    SrbCbb    = 0.0

write(*, 1010)
SraCaa   = multipMM(Sra,Caa)
SraT     = transpose(Sra)
Crrcomp  = multipMM(SraCaa,SraT)
SraCab   = multipMM(Sra,Cab)
SrbT     = transpose(Srb)
Crrcomp  = Crrcomp + 2 * multipMM(SraCab,SrbT)
SrbCbb   = multipMM(Srb,Cbb)
Crrcomp  = Crrcomp + multipMM(SrbCbb,SrbT)
call writeMatrixToFile(uCrr_comp,Crrcomp)

! define intermediate quantities and initialize
allocate(Xa(Na,Nr),stat=alloc_err)
allocate(Xb(Nb,Nr),stat=alloc_err)
allocate(Xr(Nr,Nr),stat=alloc_err)
allocate(Drr(Nr,Nr),stat=alloc_err)
allocate(D11(Nr,Nr),stat=alloc_err)
allocate(rd(Nr),stat=alloc_err)
Xa   = 0.0
Xb   = 0.0
Xr   = 0.0
Drr  = 0.0
D11  = 0.0
rd   = 0.0

! Xa=Caa*Sra'+Cab*Srb'-Car
Xa = multipMM(Caa,SraT)
Xa = Xa + multipMM(Cab,SrbT)
Xa = Xa - Car

! Xb=Cab'*Sra'+Cbb*Srb'-Cbr
Xb = multipMM(transpose(Cab),SraT)
Xb = Xb + multipMM(Cbb,SrbT)
Xb = Xb - Cbr

! Xr=Car'*Sra'+Cbr'*Srb'-Crr
allocate(CarT(Nr,Na),stat=alloc_err)
allocate(CbrT(Nr,Nb),stat=alloc_err)
CarT = 0.0
CbrT = 0.0
CarT = transpose(Car)
CbrT = transpose(Cbr)
```

```
Xr = multipMM(CarT,SraT)
Xr = Xr + multipMM(CbrT,SrbT)
Xr = Xr - Crr

! Drr=Sra*Xa+Srb*Xb-Xr
write(*, 1020)
Drr  = multipMM(Sra,Xa)
Drr  = Drr + multipMM(Srb,Xb)
Drr  = Drr - Xr

! define rd
rd = rc -rm

! compute Drr^-1
allocate(Drrinv(Nr,Nr),stat=alloc_err)
Drrinv = 0.0
Drrinv = inv(Drr)
if(is_NAN_or_Infinity_M(Drr)) then
   write(*, 1030)
end if
if(is_NAN_or_Infinity_M(Drrinv)) then
   write(*, 1040)
end if

! D11=Drr^-1
D11  = Drrinv

! best estimated mean values for aBE, bBE, rBE
write(*, 1050)
allocate(aBE(Na),stat=alloc_err)
allocate(bBE(Nb),stat=alloc_err)
allocate(rBE(Nr),stat=alloc_err)
aBE = 0.0
bBE = 0.0
rBE = 0.0

! aBE = alpha-[Xa*D11]*rd
allocate(XaD11(Na,Nr),stat=alloc_err)
XaD11 = 0.0
XaD11 = multipMM(Xa,D11)

aBE = multipMV(XaD11,rd)
aBE = alpha - aBE
call writeVectorToFile(ua_BE,aBE)

! bBE = beta-[Xb*D11]*rd
write(*, 1060)
allocate(XbD11(Nb,Nr),stat=alloc_err)
XbD11 = 0.0
XbD11 = multipMM(Xb,D11)

bBE = multipMV(XbD11,rd)
bBE = beta - bBE
call writeVectorToFile(ub_BE,bBE)

! rBE = rm-[Xr*D11]*rd
write(*, 1070)
allocate(XrD11(Nr,Nr),stat=alloc_err)
XrD11 = 0.0
XrD11 = multipMM(Xr,D11)
```

```
rBE = multipMV(XrD11,rd)
rBE = rm - rBE
call writeVectorToFile(ur_BE,rBE)

!calculate coviances for responses and parameters
allocate(CaaBE(Na,Na),stat=alloc_err)
allocate(CrrBE(Nr,Nr),stat=alloc_err)
allocate(CarBE(Na,Nr),stat=alloc_err)
allocate(CbbBE(Nb,Nb),stat=alloc_err)
allocate(CabBE(Na,Nb),stat=alloc_err)
allocate(CbrBE(Nb,Nr),stat=alloc_err)
CaaBE = 0.0; CrrBE = 0.0; CarBE = 0.0
CbbBE = 0.0; CbrBE = 0.0; CbrBE = 0.0

! CaaBE = Caa - Xa*(D11*Xa')
write(*, 1080)
CaaBE = multipMM(Xa,transpose(XaD11))
CaaBE = Caa - CaaBE
call writeMatrixToFile(uC_aaBE,CaaBE)

! CrrBE = Crr - Xr*(D11*Xr')
write(*, 1090)
CrrBE = multipMM(Xr,transpose(XrD11))
CrrBE = Crr - CrrBE
call writeMatrixToFile(uC_rrBE,CrrBE)

! CarBE = Car - Xa*(D11*Xr')
write(*, 2000)
CarBE = multipMM(Xa,transpose(XrD11))
CarBE = Car - CarBE
call writeMatrixToFile(uC_arBE,CarBE)

! CbbBE = Cbb - Xb*(D11*Xb')
write(*, 2010)
CbbBE = multipMM(Xb,transpose(XbD11))
CbbBE = Cbb - CbbBE
call writeMatrixToFile(uC_bbBE,CbbBE)

! CabBE = Cab - Xa*(D11*Xb')
write(*, 2020)
CabBE = multipMM(Xa,transpose(XbD11))
CabBE = Cab - CabBE
call writeMatrixToFile(uC_abBE,CabBE)

! CbrBE = Cbr - Xb*(D11*Xr')
write(*, 2030)
CbrBE = multipMM(Xb,transpose(XrD11))
CbrBE = Cbr - CbrBE
call writeMatrixToFile(uC_brBE,CbrBE)

!calculate the "consistency indicator" chi^2
allocate(D11rd(Nr),stat=alloc_err)
D11rd    = 0.0
chi2     = 0.0

D11rd   = multipMV(D11,rd)
chi2    = multipVV(rd,D11rd)
write(uchi2,3500,err=200) chi2, chi2/(Nr)

write(*,3600)
```

```
  return
   200 lerr = .true.;ierr = 3;call errmsg;stop
  1000 format(/, &
       1x,'-----------------  Multi-pred Solving & Output  ----------------')
  1010 format(&
       1x,'computing Crrcomp = Sra*Caa*Sra"+2*Sra*Cab*Srb"+Srb*Cbb*Srb"',a)
  1020 format(&
       1x,'computing Xa, Xb, Xr, D11, Drr, Drr^-1, rd',a)
  1030 format(&
       1x,'ERROR: Infinite or NAN in computing Drr, where '                    &
          'Drr=Sra*Xa+Srb*Xb-Xr',a)
  1040 format(&
       1x,'ERROR: Infinite or NAN in computing Drr^-1',a)
  1050 format(&
       1x,'computing aBE   = alpha-[Xa*D11]*rd',a)
  1060 format(&
       1x,'computing bBE   = beta-[Xb*D11]*rd',a)
  1070 format(&
       1x,'computing rBE   = rm-[Xr*D11]*rd',a)
  1080 format(&
       1x,'computing CaaBE = Caa - Xa*(D11*Xa")',a)
  1090 format(&
       1x,'computing CrrBE = Crr - Xr*(D11*Xr")',a)
  2000 format(&
       1x,'computing CarBE = Car - Xa*(D11*Xr")',a)
  2010 format(&
       1x,'computing CbbBE = Cbb - Xb*(D11*Xb")',a)
  2020 format(&
       1x,'computing CabBE = Cab - Xa*(D11*Xb")',a)
  2030 format(&
       1x,'computing CbrBE = Cbr - Xb*(D11*Xr")',a)
  3500 format(&
          'chi^2                                   ='                  ,F8.3/, &
          'chi^2_d = (chi^2)/(number of responses) ='                  ,F8.3)
  3600 format(/, &
       1x,'done.')

END SUBROUTINE solvercase2

!+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

SUBROUTINE solvercase3()
!
!  solver for Casee 2: predictive modeling for Model A with Nq additional
!                      responses, but no additional parameters
!
  !Local-------------------------------------------------------------------
  REAL(DP), ALLOCATABLE ::  SraCaa(:,:), SraT(:,:), SqaCaa(:,:), SqaT(:,:)
  REAL(DP), ALLOCATABLE ::  Xa(:,:),  Ya(:,:),   Xr(:,:),  Yr(:,:),  Xq(:,:), &
                            Yq(:,:), Drr(:,:),  Drq(:,:), Dqr(:,:), Dqq(:,:), &
                           D11(:,:), D12(:,:),  D21(:,:), D22(:,:), rd(:), qd(:)
  REAL(DP), ALLOCATABLE :: CarT(:,:), CaqT(:,:), CrqT(:,:)
  REAL(DP), ALLOCATABLE :: Drrinv(:,:), DqrDrrinv(:,:)
  REAL(DP), ALLOCATABLE :: XaD11plusYaD21(:,:), XaD12plusYaD22(:,:)
  REAL(DP), ALLOCATABLE :: XrD11plusYrD21(:,:), XrD12plusYrD22(:,:)
  REAL(DP), ALLOCATABLE :: XqD11plusYqD21(:,:), XqD12plusYqD22(:,:)
  REAL(DP), ALLOCATABLE :: D11rd(:), D12qd(:), D22qd(:)

  write(*,1000)

  !start with computing covariance matrix of Crr_comp, Cqq_comp and Crq_comp
```

```
!for the computed responses.

! Crrcomp = Sra*Caa*Sra'
write(*,1010)
allocate(Crrcomp(Nr,Nr),stat=alloc_err)
allocate(SraCaa(Nr,Na),stat=alloc_err)
allocate(SraT(Na,Nr),stat=alloc_err)
Crrcomp  = 0.0
SraCaa   = 0.0
SraT     = 0.0

SraCaa   = multipMM(Sra,Caa)
SraT     = transpose(Sra)
Crrcomp  = multipMM(SraCaa,SraT)
call writeMatrixToFile(uCrr_comp,Crrcomp)

! Cqqcomp = Sqa*Caa*Sqa'
write(*, 1020)
allocate(Cqqcomp(Nq,Nq),stat=alloc_err)
allocate(SqaCaa(Nq,Na),stat=alloc_err)
allocate(SqaT(Na,Nq),stat=alloc_err)
Cqqcomp  = 0.0
SqaCaa   = 0.0
SqaT     = 0.0

SqaCaa   = multipMM(Sqa,Caa)
SqaT     = transpose(Sqa)
Cqqcomp  = multipMM(SqaCaa,SqaT)
call writeMatrixToFile(uCqq_comp,Cqqcomp)

! Crqcomp = Sra*Caa*Sqa'
write(*, 1030)
allocate(Crqcomp(Nr,Nq),stat=alloc_err)
Crqcomp  = 0.0

Crqcomp  = multipMM(SraCaa,SqaT)
call writeMatrixToFile(uCrq_comp,Crqcomp)

! define intermediate quantities and initialize
write(*, 1040)
allocate(Xa(Na,Nr),stat=alloc_err)
allocate(Ya(Na,Nq),stat=alloc_err)
allocate(Xr(Nr,Nr),stat=alloc_err)
allocate(Yr(Nr,Nq),stat=alloc_err)
allocate(Xq(Nq,Nr),stat=alloc_err)
allocate(Yq(Nq,Nq),stat=alloc_err)
allocate(Drr(Nr,Nr),stat=alloc_err)
allocate(Drq(Nr,Nq),stat=alloc_err)
allocate(Dqr(Nq,Nr),stat=alloc_err)
allocate(Dqq(Nq,Nq),stat=alloc_err)
allocate(D11(Nr,Nr),stat=alloc_err)
allocate(D12(Nr,Nq),stat=alloc_err)
allocate(D21(Nq,Nr),stat=alloc_err)
allocate(D22(Nq,Nq),stat=alloc_err)
allocate(rd(Nr),stat=alloc_err)
allocate(qd(Nq),stat=alloc_err)

Xa   = 0.0; Ya   = 0.0; Xr   = 0.0;
Yr   = 0.0; Xq   = 0.0; Yq   = 0.0;
Drr  = 0.0; Drq  = 0.0; Dqr  = 0.0;
Dqq  = 0.0; D11  = 0.0; D12  = 0.0;
```

```
D21  = 0.0; D22  = 0.0; rd   = 0.0;
qd   = 0.0

! Xa=Caa*Sra'-Car
Xa = multipMM(Caa,SraT)
Xa = Xa - Car

! Ya=Caa*Sqa'-Caq
Ya = multipMM(Caa,SqaT)
Ya = Ya - Caq

! Xr=Car'*Sra'-Crr
allocate(CarT(Nr,Na),stat=alloc_err)
CarT = 0.0
CarT = transpose(Car)

Xr = multipMM(CarT,SraT)
Xr = Xr - Crr

! Yr=Car'*Sqa'-Crq
Yr = multipMM(CarT,SqaT)
Yr = Yr - Crq

! Xq=Caq'*Sra'-Crq'
allocate(CaqT(Nq,Na),stat=alloc_err)
allocate(CrqT(Nq,Nr),stat=alloc_err)
CaqT = 0.0
CrqT = 0.0
CaqT = transpose(Caq)
CrqT = transpose(Crq)

Xq = multipMM(CaqT,SraT)
Xq = Xq - CrqT

! Yq=Caq'*Sqa'-Cqq
Yq = multipMM(CaqT,SqaT)
Yq = Yq - Cqq

! Drr=Sra*Xa-Xr
Drr  = multipMM(Sra,Xa)
Drr  = Drr - Xr

! Drq=Sra*Ya-Yr
Drq  = multipMM(Sra,Ya)
Drq  = Drq - Yr

! Drq'
Dqr  = transpose(Drq)

! Dqq=Sqa*Ya-Yq
Dqq  = multipMM(Sqa,Ya)
Dqq  = Dqq - Yq

! define rd, qd
rd = rc -rm
qd = qc -qm

! compute Drr^-1, Dqq^-1, D22^-1
allocate(Drrinv(Nr,Nr),stat=alloc_err)
Drrinv = 0.0
Drrinv = inv(Drr)
```

```
if(is_NAN_or_Infinity_M(Drr)) then
   write(*, 1050)
end if
if(is_NAN_or_Infinity_M(Drrinv)) then
   write(*, 1060)
end if

! D22=[Dqq-Drq'*Drr^-1*Drq]^-1
allocate(DqrDrrinv(Nq,Nr),stat=alloc_err)
DqrDrrinv = 0.0
DqrDrrinv = multipMM(Dqr,Drrinv)
D22  = Dqq - multipMM(DqrDrrinv,Drq)
D22  = inv(D22)

! D12=-Drr^-1*Drq*D22
D12  = multipMM(Drrinv,Drq)
D12  = -1.0 * multipMM(D12,D22)

! D12'
D21 = transpose(D12)

! D11=Drr^-1+D12*Drq'*Drr^-1
D11  = multipMM(D12,Dqr)
D11  = Drrinv - multipMM(D11,Drrinv)

! best estimated mean values for aBE, rBE, qBE
write(*, 1070)
allocate(aBE(Na),stat=alloc_err)
allocate(rBE(Nr),stat=alloc_err)
allocate(qBE(Nq),stat=alloc_err)
aBE = 0.0
rBE = 0.0
qBE = 0.0

! aBE = alpha-[Xa*D11+Ya*D21]*rd-[Xa*D12+Ya*D22]*qd
allocate(XaD11plusYaD21(Na,Nr),stat=alloc_err)
allocate(XaD12plusYaD22(Na,Nq),stat=alloc_err)
XaD11plusYaD21 = 0.0
XaD12plusYaD22 = 0.0
XaD11plusYaD21 = multipMM(Xa,D11) + multipMM(Ya,D21)
XaD12plusYaD22 = multipMM(Xa,D12) + multipMM(Ya,D22)

aBE = multipMV(XaD11plusYaD21,rd)
aBE = alpha - aBE
aBE = aBE - multipMV(XaD12plusYaD22,qd)
call writeVectorToFile(ua_BE,aBE)

! rBE = rm-[Xr*D11+Yr*D21]*rd-[Xr*D12+Yr*D22]*qd
write(*, 1080)
allocate(XrD11plusYrD21(Nr,Nr),stat=alloc_err)
allocate(XrD12plusYrD22(Nr,Nq),stat=alloc_err)
XrD11plusYrD21 = 0.0
XrD12plusYrD22 = 0.0
XrD11plusYrD21 = multipMM(Xr,D11) + multipMM(Yr,D21)
XrD12plusYrD22 = multipMM(Xr,D12) + multipMM(Yr,D22)

rBE = multipMV(XrD11plusYrD21,rd)
rBE = rm - rBE
rBE = rBE - multipMV(XrD12plusYrD22,qd)
call writeVectorToFile(ur_BE,rBE)
```

```
! qBE = qm-[Xq*D11+Yq*D21]*rd-[Xq*D12+Yq*D22]*qd
write(*, 1090)
allocate(XqD11plusYqD21(Nq,Nr),stat=alloc_err)
allocate(XqD12plusYqD22(Nq,Nq),stat=alloc_err)
XqD11plusYqD21 = 0.0
XqD12plusYqD22 = 0.0
XqD11plusYqD21 = multipMM(Xq,D11) + multipMM(Yq,D21)
XqD12plusYqD22 = multipMM(Xq,D12) + multipMM(Yq,D22)

qBE = multipMV(XqD11plusYqD21,rd)
qBE = qm - qBE
qBE = qBE - multipMV(XqD12plusYqD22,qd)
call writeVectorToFile(uq_BE,qBE)

!calculate coviances for responses and parameters
allocate(CaaBE(Na,Na),stat=alloc_err)
allocate(CrrBE(Nr,Nr),stat=alloc_err)
allocate(CarBE(Na,Nr),stat=alloc_err)
allocate(CqqBE(Nq,Nq),stat=alloc_err)
allocate(CaqBE(Na,Nq),stat=alloc_err)
allocate(CrqBE(Nr,Nq),stat=alloc_err)
CaaBE = 0.0; CrrBE = 0.0; CarBE = 0.0
CqqBE = 0.0; CaqBE = 0.0; CrqBE = 0.0

! CaaBE = Caa - [Xa*(D11*Xa'+D12*Ya')+Ya*(D21*Xa'+D22*Ya')]
write(*, 2000)
CaaBE = multipMM(Xa,transpose(XaD11plusYaD21))
CaaBE = Caa - CaaBE
CaaBE = CaaBE - multipMM(Ya,transpose(XaD12plusYaD22))
call writeMatrixToFile(uC_aaBE,CaaBE)

! CrrBE = Crr - [Xr*(D11*Xr'+D12*Yr')+Yr*(D21*Xr'+D22*Yr')]
write(*, 2010)
CrrBE = multipMM(Xr,transpose(XrD11plusYrD21))
CrrBE = Crr - CrrBE
CrrBE = CrrBE - multipMM(Yr,transpose(XrD12plusYrD22))
call writeMatrixToFile(uC_rrBE,CrrBE)

! CarBE = Car - [Xa*(D11*Xr'+D12*Yr')+Ya*(D21*Xr'+D22*Yr')]
write(*, 2020)
CarBE = multipMM(Xa,transpose(XrD11plusYrD21))
CarBE = Car - CarBE
CarBE = CarBE - multipMM(Ya,transpose(XrD12plusYrD22))
call writeMatrixToFile(uC_arBE,CarBE)

! CqqBE = Cqq - [Xq*(D11*Xq'+D12*Yq')+Yq*(D21*Xq'+D22*Yq')]
write(*, 2030)
CqqBE = multipMM(Xq,transpose(XqD11plusYqD21))
CqqBE = Cqq - CqqBE
CqqBE = CqqBE - multipMM(Yq,transpose(XqD12plusYqD22))
call writeMatrixToFile(uC_qqBE,CqqBE)

! CaqBE = Caq - [Xa*(D11*Xq'+D12*Yq')+Ya*(D21*Xq'+D22*Yq')]
write(*, 2040)
CaqBE = multipMM(Xa,transpose(XqD11plusYqD21))
CaqBE = Caq - CaqBE
CaqBE = CaqBE - multipMM(Ya,transpose(XqD12plusYqD22))
call writeMatrixToFile(uC_aqBE,CaqBE)

! CrqBE = Crq - [Xr*(D11*Xq'+D12*Yq')+Yr*(D21*Xq'+D22*Yq')]
write(*, 2050)
```

```fortran
CrqBE = multipMM(Xr,transpose(XqD11plusYqD21))
CrqBE = Crq - CrqBE
CrqBE = CrqBE - multipMM(Yr,transpose(XqD12plusYqD22))
call writeMatrixToFile(uC_rqBE,CrqBE)

!calculate the "consistency indicator" chi^2
allocate(D11rd(Nr),stat=alloc_err)
allocate(D12qd(Nr),stat=alloc_err)
allocate(D22qd(Nq),stat=alloc_err)
D11rd     = 0.0
D12qd     = 0.0
D22qd     = 0.0
chi2      = 0.0

D11rd    = multipMV(D11,rd)
D12qd    = multipMV(D12,qd)
D22qd    = multipMV(D22,qd)
chi2     = multipVV(rd,D11rd)
chi2     = chi2 + 2.0 * multipVV(rd,D12qd)
chi2     = chi2 + multipVV(qd,D22qd)
write(uchi2,3500,err=200) chi2, chi2/(Nr+Nq)

write(*,3600)
return
 200 lerr = .true.;ierr = 3;call errmsg;stop
1000 format(/, &
     1x,'-----------------  Multi-pred Solving & Output  ----------------')
1010 format(&
     1x,'computing Crrcomp = Sra*Caa*Sra"',a)
1020 format(&
     1x,'computing Cqqcomp = Sqa*Caa*Sqa"',a)
1030 format(&
     1x,'computing Cqqcomp = Sqa*Caa*Sqa"',a)
1040 format(&
     1x,'computing Xa, Ya, Xr, Yr, Xq, Yq, D11, D12, D22, Drr,Drq, Dqr,'/, &
        '                Dqq, rd, qd',a)
1050 format(&
     1x,'ERROR: Infinite or NAN in computing Drr, where Drr=Sra*Xa-Xr',a)
1060 format(&
     1x,'ERROR: Infinite or NAN in computing Drr^-1',a)
1070 format(&
     1x,'computing aBE    = alpha-[Xa*D11+Ya*D21]*rd-[Xa*D12+Ya*D22]*qd',a)
1080 format(&
     1x,'computing rBE    = rm-[Xr*D11+Yr*D21]*rd-[Xr*D12+Yr*D22]*qd',a)
1090 format(&
     1x,'computing qBE    = qm-[Xq*D11+Yq*D21]*rd-[Xq*D12+Yq*D22]*qd',a)
2000 format(&
     1x,'computing CaaBE = Caa-[Xa*(D11*Xa"+D12*Ya")+Ya*(D21*Xa"+D22*Ya")]',a)
2010 format(&
     1x,'computing CrrBE = Crr-[Xr*(D11*Xr"+D12*Yr")+Yr*(D21*Xr"+D22*Yr")]',a)
2020 format(&
     1x,'computing CarBE = Car-[Xa*(D11*Xr"+D12*Yr")+Ya*(D21*Xr"+D22*Yr")]',a)
2030 format(&
     1x,'computing CqqBE = Cqq-[Xq*(D11*Xq"+D12*Yq")+Yq*(D21*Xq"+D22*Yq")]',a)
2040 format(&
     1x,'computing CaqBE = Caq-[Xa*(D11*Xq"+D12*Yq")+Ya*(D21*Xq"+D22*Yq")]',a)
2050 format(&
     1x,'computing CrqBE = Crq-[Xr*(D11*Xq"+D12*Yq")+Yr*(D21*Xq"+D22*Yq")]',a)
3500 format(&
        'chi^2                                   ='                  ,F8.3/, &
        'chi^2_d = (chi^2)/(number of responses) ='                  ,F8.3)
```

```fortran
  3600 format(/, &
       1x,'done.')

END SUBROUTINE solvercase3

!+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

SUBROUTINE solvercase4()
!
!  solver for Casee 4: coupled Models A & B
!
  !Local---------------------------------------------------------------------
  REAL(DP), ALLOCATABLE :: SraCaa(:,:), SraT(:,:), SraCab(:,:), SrbT(:,:),    &
                           SrbCbb(:,:)
  REAL(DP), ALLOCATABLE :: SqaCaa(:,:), SqaT(:,:), SqaCab(:,:), SqbT(:,:),    &
                           SqbCbb(:,:)
  REAL(DP), ALLOCATABLE :: CabT(:,:), SrbCabT(:,:)
  REAL(DP), ALLOCATABLE :: Xa(:,:),   Ya(:,:),  Xb(:,:),  Yb(:,:),   Xr(:,:), &
                           Yr(:,:),   Xq(:,:),  Yq(:,:), Drr(:,:),  Drq(:,:), &
                           Dqr(:,:), Dqq(:,:), D11(:,:), D12(:,:),  D21(:,:), &
                           D22(:,:),  rd(:),  qd(:)
  REAL(DP), ALLOCATABLE :: CarT(:,:), CbrT(:,:)
  REAL(DP), ALLOCATABLE :: CaqT(:,:), CbqT(:,:), CrqT(:,:)
  REAL(DP), ALLOCATABLE :: Drrinv(:,:), DqrDrrinv(:,:), temp2(:,:), temp3(:,:)
  REAL(DP), ALLOCATABLE :: XaD11plusYaD21(:,:), XaD12plusYaD22(:,:)
  REAL(DP), ALLOCATABLE :: XbD11plusYbD21(:,:), XbD12plusYbD22(:,:)
  REAL(DP), ALLOCATABLE :: XrD11plusYrD21(:,:), XrD12plusYrD22(:,:)
  REAL(DP), ALLOCATABLE :: XqD11plusYqD21(:,:), XqD12plusYqD22(:,:)
  REAL(DP), ALLOCATABLE :: D11rd(:), D12qd(:), D22qd(:)

  write(*,1000)

  !start with computing covariance matrix of Crr_comp, Cqq_comp and Crq_comp
  !for the computed responses.

  ! Crrcomp = Sra*Caa*Sra'+2Sra*Cab*Srb'+Srb*Cbb*Srb'
  write(*, 1010)
  allocate(Crrcomp(Nr,Nr),stat=alloc_err)
  allocate(SraCaa(Nr,Na),stat=alloc_err)
  allocate(SraT(Na,Nr),stat=alloc_err)
  allocate(SraCab(Nr,Nb),stat=alloc_err)
  allocate(SrbT(Nb,Nr),stat=alloc_err)
  allocate(SrbCbb(Nr,Nb),stat=alloc_err)
  Crrcomp  = 0.0;    SraCaa   = 0.0
  SraT     = 0.0;    SraCab   = 0.0
  SrbT     = 0.0;    SrbCbb   = 0.0

  SraCaa   = multipMM(Sra,Caa)
  SraT     = transpose(Sra)
  Crrcomp  = multipMM(SraCaa,SraT)
  SraCab   = multipMM(Sra,Cab)
  SrbT     = transpose(Srb)
  Crrcomp  = Crrcomp + 2 * multipMM(SraCab,SrbT)
  SrbCbb   = multipMM(Srb,Cbb)
  Crrcomp  = Crrcomp + multipMM(SrbCbb,SrbT)
  call writeMatrixToFile(uCrr_comp,Crrcomp)

  ! Cqqcomp = Sqa*Caa*Sqa'+2Sqa*Cab*Sqb'+Sqb*Cbb*Sqb'
  write(*, 1020)
  allocate(Cqqcomp(Nq,Nq),stat=alloc_err)
  allocate(SqaCaa(Nq,Na),stat=alloc_err)
```

```
allocate(SqaT(Na,Nq),stat=alloc_err)
allocate(SqaCab(Nq,Nb),stat=alloc_err)
allocate(SqbT(Nb,Nq),stat=alloc_err)
allocate(SqbCbb(Nq,Nb),stat=alloc_err)
Cqqcomp  = 0.0;   SqaCaa   = 0.0
SqaT     = 0.0;   SqaCab   = 0.0
SqbT     = 0.0;   SqbCbb   = 0.0

SqaCaa   = multipMM(Sqa,Caa)
SqaT     = transpose(Sqa)
Cqqcomp  = multipMM(SqaCaa,SqaT)
SqaCab   = multipMM(Sqa,Cab)
SqbT     = transpose(Sqb)
Cqqcomp  = Cqqcomp + 2 * multipMM(SqaCab,SqbT)
SqbCbb   = multipMM(Sqb,Cbb)
Cqqcomp  = Cqqcomp + multipMM(SqbCbb,SqbT)
call writeMatrixToFile(uCqq_comp,Cqqcomp)

! Crqcomp = Sra*Caa*Sqa'+Sra*Cab*Sqb'+Srb*Cab'*Sqa'+Sqb*Cbb*Sqb'
write(*, 1030)
allocate(Crqcomp(Nr,Nq),stat=alloc_err)
allocate(CabT(Nb,Na),stat=alloc_err)
allocate(SrbCabT(Nr,Na),stat=alloc_err)
Crqcomp  = 0.0
CabT     = 0.0
SrbCabT  = 0.0

Crqcomp  = multipMM(SraCaa,SqaT)
Crqcomp  = Crqcomp + multipMM(SraCab,SqbT)
CabT     = transpose(Cab)
SrbCabT  = multipMM(Srb,CabT)
Crqcomp  = Crqcomp + multipMM(SrbCabT,SqaT)
Crqcomp  = Crqcomp + multipMM(SrbCbb,SqbT)
call writeMatrixToFile(uCrq_comp,Crqcomp)

! define intermediate quantities and initialize
write(*, 1040)
allocate(Xa(Na,Nr),stat=alloc_err)
allocate(Ya(Na,Nq),stat=alloc_err)
allocate(Xb(Nb,Nr),stat=alloc_err)
allocate(Yb(Nb,Nq),stat=alloc_err)
allocate(Xr(Nr,Nr),stat=alloc_err)
allocate(Yr(Nr,Nq),stat=alloc_err)
allocate(Xq(Nq,Nr),stat=alloc_err)
allocate(Yq(Nq,Nq),stat=alloc_err)
allocate(Drr(Nr,Nr),stat=alloc_err)
allocate(Drq(Nr,Nq),stat=alloc_err)
allocate(Dqr(Nq,Nr),stat=alloc_err)
allocate(Dqq(Nq,Nq),stat=alloc_err)
allocate(D11(Nr,Nr),stat=alloc_err)
allocate(D12(Nr,Nq),stat=alloc_err)
allocate(D21(Nq,Nr),stat=alloc_err)
allocate(D22(Nq,Nq),stat=alloc_err)
allocate(rd(Nr),stat=alloc_err)
allocate(qd(Nq),stat=alloc_err)

Xa   = 0.0; Ya   = 0.0; Xb   = 0.0
Yb   = 0.0; Xr   = 0.0; Yr   = 0.0
Xq   = 0.0; Yq   = 0.0; Drr  = 0.0
Drq  = 0.0; Dqr  = 0.0; Dqq  = 0.0
D11  = 0.0; D12  = 0.0; qd   = 0.0
```

```
D21  = 0.0; D22  = 0.0; rd   = 0.0

! Xa=Caa*Sra'+Cab*Srb'-Car
Xa = multipMM(Caa,SraT)
Xa = Xa + multipMM(Cab,SrbT)
Xa = Xa - Car

! Ya=Caa*Sqa'+Cab*Sqb'-Caq
Ya = multipMM(Caa,SqaT)
Ya = Ya + multipMM(Cab,SqbT)
Ya = Ya - Caq

! Xb=Cab'*Sra'+Cbb*Srb'-Cbr
Xb = multipMM(CabT,SraT)
Xb = Xb + multipMM(Cbb,SrbT)
Xb = Xb - Cbr

! Yb=Cba*Sqa'+Cbb*Sqb'-Cbq
Yb = multipMM(CabT,SqaT)
Yb = Yb + multipMM(Cbb,SqbT)
Yb = Yb - Cbq

! Xr=Car'*Sra'+Cbr'*Srb'-Crr
allocate(CarT(Nr,Na),stat=alloc_err)
allocate(CbrT(Nr,Nb),stat=alloc_err)
CarT = 0.0
CbrT = 0.0
CarT = transpose(Car)
CbrT = transpose(Cbr)

Xr = multipMM(CarT,SraT)
Xr = Xr + multipMM(CbrT,SrbT)
Xr = Xr - Crr

! Yr=Car'*Sqa'+Cbr'*Sqb'-Crq
Yr = multipMM(CarT,SqaT)
Yr = Yr + multipMM(CbrT,SqbT)
Yr = Yr - Crq

! Xq=Caq'*Sra'+Cbq'*Srb'-Crq'
allocate(CaqT(Nq,Na),stat=alloc_err)
allocate(CbqT(Nq,Nb),stat=alloc_err)
allocate(CrqT(Nq,Nr),stat=alloc_err)
CaqT = 0.0
CbqT = 0.0
CrqT = 0.0
CaqT = transpose(Caq)
CbqT = transpose(Cbq)
CrqT = transpose(Crq)

Xq = multipMM(CaqT,SraT)
Xq = Xq + multipMM(CbqT,SrbT)
Xq = Xq - CrqT

! Yq=Caq'*Sqa'+Cbq'*Sqb'-Cqq
Yq = multipMM(CaqT,SqaT)
Yq = Yq + multipMM(CbqT,SqbT)
Yq = Yq - Cqq

! Drr=Sra*Xa+Srb*Xb-Xr
Drr  = multipMM(Sra,Xa)
```

```
Drr  = Drr + multipMM(Srb,Xb)
Drr  = Drr - Xr

! Drq=Sra*Ya+Srb*Yb-Yr
Drq  = multipMM(Sra,Ya)
Drq  = Drq + multipMM(Srb,Yb)
Drq  = Drq - Yr

! Drq'
Dqr  = transpose(Drq)

! Dqq=Sqa*Ya+Sqb*Yb-Yq
Dqq  = multipMM(Sqa,Ya)
Dqq  = Dqq + multipMM(Sqb,Yb)
Dqq  = Dqq - Yq

! define rd, qd
rd = rc -rm
qd = qc -qm

! compute Drr^-1, Dqq^-1, D22^-1
allocate(Drrinv(Nr,Nr),stat=alloc_err)
Drrinv = 0.0
Drrinv = inv(Drr)
if(is_NAN_or_Infinity_M(Drr)) then
   write(*, 1050)
end if
if(is_NAN_or_Infinity_M(Drrinv)) then
   write(*, 1060)
end if

! D22=[Dqq-Drq'*Drr^-1*Drq]^-1
allocate(DqrDrrinv(Nq,Nr),stat=alloc_err)
DqrDrrinv = 0.0
DqrDrrinv = multipMM(Dqr,Drrinv)
D22  = Dqq - multipMM(DqrDrrinv,Drq)
D22  = inv(D22)

! D12=-Drr^-1*Drq*D22
D12  = multipMM(Drrinv,Drq)
D12  = -1.0 * multipMM(D12,D22)

! D12'
D21 = transpose(D12)

! D11=Drr^-1+D12*Drq'*Drr^-1
D11  = multipMM(D12,Dqr)
D11  = Drrinv - multipMM(D11,Drrinv)

! best estimated mean values for aBE, bBE, rBE, qBE
write(*, 1070)
allocate(aBE(Na),stat=alloc_err)
allocate(bBE(Nb),stat=alloc_err)
allocate(rBE(Nr),stat=alloc_err)
allocate(qBE(Nq),stat=alloc_err)
aBE = 0.0
bBE = 0.0
rBE = 0.0
qBE = 0.0

! aBE = alpha-[Xa*D11+Ya*D21]*rd-[Xa*D12+Ya*D22]*qd
```

```
allocate(XaD11plusYaD21(Na,Nr),stat=alloc_err)
allocate(XaD12plusYaD22(Na,Nq),stat=alloc_err)
XaD11plusYaD21 = 0.0
XaD12plusYaD22 = 0.0
XaD11plusYaD21 = multipMM(Xa,D11) + multipMM(Ya,D21)
XaD12plusYaD22 = multipMM(Xa,D12) + multipMM(Ya,D22)

aBE = multipMV(XaD11plusYaD21,rd)
aBE = alpha - aBE
aBE = aBE - multipMV(XaD12plusYaD22,qd)
call writeVectorToFile(ua_BE,aBE)

! bBE = beta-[Xb*D11+Yb*D21]*rd-[Xb*D12+Yb*D22]*qd
write(*, 1080)
allocate(XbD11plusYbD21(Nb,Nr),stat=alloc_err)
allocate(XbD12plusYbD22(Nb,Nq),stat=alloc_err)
XbD11plusYbD21 = 0.0
XbD12plusYbD22 = 0.0
XbD11plusYbD21 = multipMM(Xb,D11) + multipMM(Yb,D21)
XbD12plusYbD22 = multipMM(Xb,D12) + multipMM(Yb,D22)

bBE = multipMV(XbD11plusYbD21,rd)
bBE = beta - bBE
bBE = bBE - multipMV(XbD12plusYbD22,qd)
call writeVectorToFile(ub_BE,bBE)

! rBE = rm-[Xr*D11+Yr*D21]*rd-[Xr*D12+Yr*D22]*qd
write(*, 1090)
allocate(XrD11plusYrD21(Nr,Nr),stat=alloc_err)
allocate(XrD12plusYrD22(Nr,Nq),stat=alloc_err)
XrD11plusYrD21 = 0.0
XrD12plusYrD22 = 0.0
XrD11plusYrD21 = multipMM(Xr,D11) + multipMM(Yr,D21)
XrD12plusYrD22 = multipMM(Xr,D12) + multipMM(Yr,D22)

rBE = multipMV(XrD11plusYrD21,rd)
rBE = rm - rBE
rBE = rBE - multipMV(XrD12plusYrD22,qd)
call writeVectorToFile(ur_BE,rBE)

! qBE = qm-[Xq*D11+Yq*D21]*rd-[Xq*D12+Yq*D22]*qd
write(*, 2000)
allocate(XqD11plusYqD21(Nq,Nr),stat=alloc_err)
allocate(XqD12plusYqD22(Nq,Nq),stat=alloc_err)
XqD11plusYqD21 = 0.0
XqD12plusYqD22 = 0.0
XqD11plusYqD21 = multipMM(Xq,D11) + multipMM(Yq,D21)
XqD12plusYqD22 = multipMM(Xq,D12) + multipMM(Yq,D22)

qBE = multipMV(XqD11plusYqD21,rd)
qBE = qm - qBE
qBE = qBE - multipMV(XqD12plusYqD22,qd)
call writeVectorToFile(uq_BE,qBE)

!calculate coviances for responses and parameters
allocate(CaaBE(Na,Na),stat=alloc_err)
allocate(CrrBE(Nr,Nr),stat=alloc_err)
allocate(CarBE(Na,Nr),stat=alloc_err)
allocate(CbbBE(Nb,Nb),stat=alloc_err)
allocate(CqqBE(Nq,Nq),stat=alloc_err)
allocate(CbqBE(Nb,Nq),stat=alloc_err)
```

```
allocate(CabBE(Na,Nb),stat=alloc_err)
allocate(CaqBE(Na,Nq),stat=alloc_err)
allocate(CbrBE(Nb,Nr),stat=alloc_err)
allocate(CrqBE(Nr,Nq),stat=alloc_err)
CaaBE = 0.0; CrrBE = 0.0; CarBE = 0.0
CbbBE = 0.0; CqqBE = 0.0; CbqBE = 0.0
CaqBE = 0.0; CbrBE = 0.0; CbrBE = 0.0
CrqBE = 0.0

! CaaBE = Caa - [Xa*(D11*Xa'+D12*Ya')+Ya*(D21*Xa'+D22*Ya')]
write(*, 2010)
CaaBE = multipMM(Xa,transpose(XaD11plusYaD21))
CaaBE = Caa - CaaBE
CaaBE = CaaBE - multipMM(Ya,transpose(XaD12plusYaD22))
call writeMatrixToFile(uC_aaBE,CaaBE)

! CrrBE = Crr - [Xr*(D11*Xr'+D12*Yr')+Yr*(D21*Xr'+D22*Yr')]
write(*, 2020)
CrrBE = multipMM(Xr,transpose(XrD11plusYrD21))
CrrBE = Crr - CrrBE
CrrBE = CrrBE - multipMM(Yr,transpose(XrD12plusYrD22))
call writeMatrixToFile(uC_rrBE,CrrBE)

! CarBE = Car - [Xa*(D11*Xr'+D12*Yr')+Ya*(D21*Xr'+D22*Yr')]
write(*, 2030)
CarBE = multipMM(Xa,transpose(XrD11plusYrD21))
CarBE = Car - CarBE
CarBE = CarBE - multipMM(Ya,transpose(XrD12plusYrD22))
call writeMatrixToFile(uC_arBE,CarBE)

! CbbBE = Cbb - [Xb*(D11*Xb'+D12*Yb')+Yb*(D21*Xb'+D22*Yb')]
write(*, 2040)
CbbBE = multipMM(Xb,transpose(XbD11plusYbD21))
CbbBE = Cbb - CbbBE
CbbBE = CbbBE - multipMM(Yb,transpose(XbD12plusYbD22))
call writeMatrixToFile(uC_bbBE,CbbBE)

! CqqBE = Cqq - [Xq*(D11*Xq'+D12*Yq')+Yq*(D21*Xq'+D22*Yq')]
write(*, 2050)
CqqBE = multipMM(Xq,transpose(XqD11plusYqD21))
CqqBE = Cqq - CqqBE
CqqBE = CqqBE - multipMM(Yq,transpose(XqD12plusYqD22))
call writeMatrixToFile(uC_qqBE,CqqBE)

! CbqBE = Cbq - [Xb*(D11*Xq'+D12*Yq')+Yb*(D21*Xq'+D22*Yq')]
write(*, 2060)
CbqBE = multipMM(Xb,transpose(XqD11plusYqD21))
CbqBE = Cbq - CbqBE
CbqBE = CbqBE - multipMM(Yb,transpose(XqD12plusYqD22))
call writeMatrixToFile(uC_bqBE,CbqBE)

! CabBE = Cab - [Xa*(D11*Xb'+D12*Yb')+Yb*(D21*Xb'+D22*Yb')]
write(*, 2070)
CabBE = multipMM(Xa,transpose(XbD11plusYbD21))
CabBE = Cab - CabBE
CabBE = CabBE - multipMM(Ya,transpose(XbD12plusYbD22))
call writeMatrixToFile(uC_abBE,CabBE)

! CaqBE = Caq - [Xa*(D11*Xq'+D12*Yq')+Ya*(D21*Xq'+D22*Yq')]
write(*, 2080)
CaqBE = multipMM(Xa,transpose(XqD11plusYqD21))
```

```
CaqBE = Caq - CaqBE
CaqBE = CaqBE - multipMM(Ya,transpose(XqD12plusYqD22))
call writeMatrixToFile(uC_aqBE,CaqBE)

! CbrBE = Cbr - [Xb*(D11*Xr'+D12*Yr')+Yb*(D21*Xr'+D22*Yr')]
write(*, 2090)
CbrBE = multipMM(Xb,transpose(XrD11plusYrD21))
CbrBE = Cbr - CbrBE
CbrBE = CbrBE - multipMM(Yb,transpose(XrD12plusYrD22))
call writeMatrixToFile(uC_brBE,CbrBE)

! CrqBE = Crq - [Xr*(D11*Xq'+D12*Yq')+Yr*(D21*Xq'+D22*Yq')]
write(*, 3000)
CrqBE = multipMM(Xr,transpose(XqD11plusYqD21))
CrqBE = Crq - CrqBE
CrqBE = CrqBE - multipMM(Yr,transpose(XqD12plusYqD22))
call writeMatrixToFile(uC_rqBE,CrqBE)


!calculate the "consistency indicator" chi^2
allocate(D11rd(Nr),stat=alloc_err)
allocate(D12qd(Nr),stat=alloc_err)
allocate(D22qd(Nq),stat=alloc_err)
D11rd    = 0.0
D12qd    = 0.0
D22qd    = 0.0
chi2     = 0.0

D11rd   = multipMV(D11,rd)
D12qd   = multipMV(D12,qd)
D22qd   = multipMV(D22,qd)
chi2    = multipVV(rd,D11rd)
chi2    = chi2 + 2.0 * multipVV(rd,D12qd)
chi2    = chi2 + multipVV(qd,D22qd)
write(uchi2,3500,err=200) chi2, chi2/(Nr+Nq)

write(*,3600)
return
 200 lerr = .true.;ierr = 3;call errmsg;stop
1000 format(/, &
     1x,'-----------------  Multi-pred Solving & Output  ----------------')
1010 format(&
     1x,'computing Crrcomp = Sra*Caa*Sra"+2*Sra*Cab*Srb"+Srb*Cbb*Srb"',a)
1020 format(&
     1x,'computing Cqqcomp = Sqa*Caa*Sqa"+2*Sqa*Cab*Sqb"+Sqb*Cbb*Sqb"',a)
1030 format(&
     1x,'computing Crqcomp =Sra*Caa*Sqa"+Sra*Cab*Sqb"+Srb*Cab"*Sqa"+Sqb*Cbb*Sqb"',a)
1040 format(&
     1x,'computing Xa, Ya, Xb, Yb, Xr, Yr, Xq, Yq, D11, D12, D22, Drr,'   /, &
        '             Drq, Dqr, Dqq, rd, qd',a)
1050 format(&
     1x,'ERROR: Infinite or NAN in computing Drr, where '                    &
        'Drr=Sra*Xa+Srb*Xb-Xr',a)
1060 format(&
     1x,'ERROR: Infinite or NAN in computing Drr^-1',a)
1070 format(&
     1x,'computing aBE   = alpha-[Xa*D11+Ya*D21]*rd-[Xa*D12+Ya*D22]*qd',a)
1080 format(&
     1x,'computing bBE   = beta-[Xb*D11+Yb*D21]*rd-[Xb*D12+Yb*D22]*qd',a)
1090 format(&
     1x,'computing rBE   = rm-[Xr*D11+Yr*D21]*rd-[Xr*D12+Yr*D22]*qd',a)
```

```fortran
  2000 format(&
       1x,'computing qBE   = qm-[Xq*D11+Yq*D21]*rd-[Xq*D12+Yq*D22]*qd',a)
  2010 format(&
       1x,'computing CaaBE = Caa-[Xa*(D11*Xa"+D12*Ya")+Ya*(D21*Xa"+D22*Ya")]',a)
  2020 format(&
       1x,'computing CrrBE = Crr-[Xr*(D11*Xr"+D12*Yr")+Yr*(D21*Xr"+D22*Yr")]',a)
  2030 format(&
       1x,'computing CarBE = Car-[Xa*(D11*Xr"+D12*Yr")+Ya*(D21*Xr"+D22*Yr")]',a)
  2040 format(&
       1x,'computing CbbBE = Cbb-[Xb*(D11*Xb"+D12*Yb")+Yb*(D21*Xb"+D22*Yb")]',a)
  2050 format(&
       1x,'computing CqqBE = Cqq-[Xq*(D11*Xq"+D12*Yq")+Yq*(D21*Xq"+D22*Yq")]',a)
  2060 format(&
       1x,'computing CbqBE = Cbq-[Xb*(D11*Xq"+D12*Yq")+Yb*(D21*Xq"+D22*Yq")]',a)
  2070 format(&
       1x,'computing CabBE = Cab-[Xa*(D11*Xb"+D12*Yb")+Yb*(D21*Xb"+D22*Yb")]',a)
  2080 format(&
       1x,'computing CaqBE = Caq-[Xa*(D11*Xq"+D12*Yq")+Ya*(D21*Xq"+D22*Yq")]',a)
  2090 format(&
       1x,'computing CbrBE = Cbr-[Xb*(D11*Xr"+D12*Yr")+Yb*(D21*Xr"+D22*Yr")]',a)
  3000 format(&
       1x,'computing CrqBE = Crq-[Xr*(D11*Xq"+D12*Yq")+Yr*(D21*Xq"+D22*Yq")]',a)
  3500 format(&
          'chi^2                                   ='                  ,F8.3/, &
          'chi^2_d = (chi^2)/(number of responses) ='                  ,F8.3)
  3600 format(/, &
       1x,'done.')

END SUBROUTINE solvercase4

!++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

END MODULE ModuleMultiPred
```

8.11 Module ModuleLapack.f90

```fortran
MODULE ModuleLapack
!****************************************************************************
!*                                                                          *
!* Module ModuleLapack encapsulates subroutines/functions for matrix/vector *
!* operations, including multiplication and inverse, which call subroutines *
!* from lapack.                                                             *
!*                                                                          *
!*                      FUNCTION inv(A) result(Ainv)                        *
!*                      FUNCTION multipMM(A, B) result(C)                   *
!*                      FUNCTION multipMV(A, B) result(C)                   *
!*                      FUNCTION is_positive_definite(A) result(info)       *
!*                                                                          *
!*      All dependent subroutines are extracted from lapack and packaged in *
!*      this code. Therefore, no lapack installation is needed to run this  *
!*      program.                                                            *
!*                                                                          *
!* called by:  MultiPredSolver                                              *
!* calls  to:  inv,multipMM,multipMV,DGEMM,DTRSM,XERBLA,LSAME,DLASWP,DSCAL, *
!*             DGETF2,DGETRF,DGER,DLAMCH,DGETRF2,ILAENV,IEEECK,DSWAP,DGEMV, *
!*             DTRTRI,DTRMM,DTRTI2,DTRMV,DGETRI,DPBTF2,DSYR,DPOTF2,DDOT,    *
!*             DISNAN,DLAISNAN,DSYRK,DPBTRF                                 *
```

```
!*                                                                            *
!*****************************************************************************

  IMPLICIT NONE

CONTAINS

!  ====================================================================

  FUNCTION inv(A) result(Ainv)

!  -- Returns the inverse of a matrix calculated by finding the LU
!  -- decomposition.
!  -- developed by University of South Carolina.
!
    INTEGER, PARAMETER :: DP = KIND(1.0D0)
  ! Arguments------------------------------------------------------------------
    REAL(DP), dimension(:,:), intent(in) :: A
    REAL(DP), dimension(size(A,1),size(A,2)) :: Ainv
  ! Local----------------------------------------------------------------------
    REAL(DP), dimension(size(A,1)) :: work           ! work array for LAPACK
    INTEGER,  dimension(size(A,1)) :: ipiv           ! pivot indices
    INTEGER                        :: n, info

  ! External procedures defined in LAPACK
!    external DGETRF
!    external DGETRI

  ! Store A in Ainv to prevent it from being overwritten by LAPACK
    Ainv = A
    n = size(A,1)

  ! DGETRF computes an LU factorization of a general M-by-N matrix A
  ! using partial pivoting with row interchanges.
    call DGETRF(n, n, Ainv, n, ipiv, info)

    if (info /= 0) then
       stop 'Matrix is numerically singular!'
    end if

  ! DGETRI computes the inverse of a matrix using the LU factorization
  ! computed by DGETRF.
    call DGETRI(n, Ainv, n, ipiv, work, n, info)

    if (info /= 0) then
       stop 'Matrix inversion failed!'
    end if

  END FUNCTION inv

!  ====================================================================

  FUNCTION multipMM(A, B) result(C)

!  -- Returns the product of a matrix C= A*B calculated by using DGEMM(),
!  -- a subroutine of LAPACK, where A and B are 2D matrices.
!  -- developed by University of South Carolina.
!
    INTEGER, PARAMETER :: DP = KIND(1.0D0)
  ! Arguments------------------------------------------------------------------
    REAL(DP), dimension(:,:), intent(in) :: A,B
```

```
    REAL(DP), dimension(size(A,1),size(B,2)) :: C
  ! Local-------------------------------------------------------------------
    INTEGER                        :: M, K, N, LDA, LDB, LDC
    REAL(DP)                       :: ALPHA, BETA

  ! External procedures defined in LAPACK
!     external DGEMM

  ! initialize data for matrix multiplication C=A*B
    C     = 0.0
    ALPHA = 1.0
    BETA  = 0.0
    M     = size(A,1)
    N     = size(B,2)
    K     = size(A,2)
    LDA   = M
    LDB   = size(B,1)
    LDC   = M

  ! Computing matrix product using DGEMM subroutine
    call DGEMM('N','N',M,N,K,ALPHA,A,LDA,B,LDB,BETA,C,LDC)

  END FUNCTION multipMM

!   ==================================================================

  FUNCTION multipMV(A, B) result(C)

!   -- Returns the product of a matrix C= A*B calculated by using DGEMM(),
!   -- a subroutine of LAPACK, where A is a 2D matrix and B is a column vector.
!   -- developed by University of South Carolina.
!
    INTEGER, PARAMETER :: DP = KIND(1.0D0)
  ! Arguments---------------------------------------------------------------
    REAL(DP), dimension(:,:), intent(in) :: A
    REAL(DP), dimension(:), intent(in)   :: B
    REAL(DP), dimension(size(A,1))       :: C
  ! Local-------------------------------------------------------------------
    INTEGER                        :: M, K, N, LDA, LDB, LDC
    REAL(DP)                       :: ALPHA, BETA

  ! External procedures defined in LAPACK
!     external DGEMM

  ! initialize data for matrix multiplication C=A*B
    C     = 0.0
    ALPHA = 1.0
    BETA  = 0.0
    M     = size(A,1)
    N     = 1
    K     = size(A,2)
    LDA   = M
    LDB   = size(B,1)
    LDC   = M

  ! Computing matrix product using DGEMM subroutine
    call DGEMM('N','N',M,N,K,ALPHA,A,LDA,B,LDB,BETA,C,LDC)

  END FUNCTION multipMV

!   ==================================================================
```

```fortran

  FUNCTION multipVV(A, B) result(C)

!  -- Returns the product of vectors C= A*B calculated by using SDSDOT(),
!  -- a subroutine of LAPACK, where A and B are both vectors.
!  -- developed by University of South Carolina.
!
    INTEGER, PARAMETER :: DP = KIND(1.0D0)
  ! Arguments-------------------------------------------------------------------
    REAL(DP), dimension(:), intent(in)   :: A(:)
    REAL(DP), dimension(:), intent(in)   :: B(:)
    REAL(DP)                             :: C
  ! Local-----------------------------------------------------------------------
    INTEGER                              :: N, INCX, INCY
    REAL                                 :: SB
  ! initialize arguments for SDSDOT()
    N      = size(B,1)
    SB     = 0.0
    INCX   = 1
    INCY   = 1

  ! Computing matrix product using DGEMM subroutine
    C = SDSDOT(N,SB,A,INCX,B,INCY)

  END FUNCTION multipVV
!  =====================================================================

  INTEGER FUNCTION is_positive_definite(A) result(info)

!  -- check if matrix A positive definite, using the Cholesky factorization
!  -- and check to see if such a factorization exists.
!  -- developed by University of South Carolina.
!
    INTEGER, PARAMETER :: DP = KIND(1.0D0)
  ! Arguments-------------------------------------------------------------------
    REAL(DP), dimension(:,:), intent(in) :: A
  ! Local-----------------------------------------------------------------------
    INTEGER                        :: i, ifail, j, kd, ldab, n
    INTEGER                        :: ARow, AColumn, nZ
    LOGICAL                        :: flag
    CHARACTER (1)                  :: uplo
    REAL(DP), ALLOCATABLE          :: AB(:,:)

   !find the array size
    ARow    = size(A,1)
    AColumn = size(A,2)

    nZ = 0
    flag = .false.
   !find the number of superdiagonals of the Upper triangle of A,
    jloop: do j=1, AColumn
       iloop: do i=1, ARow-j+1
                if(A(i,i+j-1)/= 0.0) then
                  flag = .true.
                  exit iloop
                end if
       end do iloop
       if(flag) then
           nZ = nZ + 1
       end if
       flag = .false.
```

```
    end do jloop

    n = AColumn
    kd   = nZ - 1
    ldab = kd + 1
    ALLOCATE (AB(ldab,n))
    AB = 0.0

 ! write the lower triangle of the symmetric band matrix A, stored in
 ! the first KD+1 rows of the array, as required by SUBROUTINE DPBTRF.
    do i=1, ldab
       do j=1, n-i+1
          AB(i,j) = A(i+j-1,j)
       end do
    end do

    uplo = 'L'
  ! call dpbtrf to factorize A
    CALL dpbtrf(uplo,n,kd,AB,ldab,info)

  END FUNCTION is_positive_definite

!  =====================================================================
      SUBROUTINE DGEMM(TRANSA,TRANSB,M,N,K,ALPHA,A,LDA,B,LDB,BETA,C,LDC)
!
!  -- Reference BLAS level3 routine (version 3.7.0) --
!  -- Reference BLAS is a software package provided by Univ. of Tennessee,    --
!  -- Univ. of California Berkeley, Univ. of Colorado Denver and NAG Ltd..--
!     December 2016
!
!     .. Scalar Arguments ..
      DOUBLE PRECISION ALPHA,BETA
      INTEGER K,LDA,LDB,LDC,M,N
      CHARACTER TRANSA,TRANSB
!     ..
!     .. Array Arguments ..
      DOUBLE PRECISION A(LDA,*),B(LDB,*),C(LDC,*)
!     ..
!
!  =====================================================================
!
!     .. External Functions ..
!      LOGICAL LSAME
!      EXTERNAL LSAME
!     ..
!     .. External Subroutines ..
!      EXTERNAL XERBLA
!     ..
!     .. Intrinsic Functions ..
      INTRINSIC MAX
!     ..
!     .. Local Scalars ..
      DOUBLE PRECISION TEMP
      INTEGER I,INFO,J,L,NCOLA,NROWA,NROWB
      LOGICAL NOTA,NOTB
!     ..
!     .. Parameters ..
      DOUBLE PRECISION ONE,ZERO
      PARAMETER (ONE=1.0D+0,ZERO=0.0D+0)
!     ..
!
```

```fortran
!     Set  NOTA  and  NOTB  as  true if  A  and  B  respectively are not
!     transposed and set  NROWA, NCOLA and  NROWB  as the number of rows
!     and  columns of  A  and the  number of  rows  of  B  respectively.
!
      NOTA = LSAME(TRANSA,'N')
      NOTB = LSAME(TRANSB,'N')
      IF (NOTA) THEN
          NROWA = M
          NCOLA = K
      ELSE
          NROWA = K
          NCOLA = M
      END IF
      IF (NOTB) THEN
          NROWB = K
      ELSE
          NROWB = N
      END IF
!
!     Test the input parameters.
!
      INFO = 0
      IF ((.NOT.NOTA) .AND. (.NOT.LSAME(TRANSA,'C')) .AND. &
     &    (.NOT.LSAME(TRANSA,'T'))) THEN
          INFO = 1
      ELSE IF ((.NOT.NOTB) .AND. (.NOT.LSAME(TRANSB,'C')) .AND. &
     &         (.NOT.LSAME(TRANSB,'T'))) THEN
          INFO = 2
      ELSE IF (M.LT.0) THEN
          INFO = 3
      ELSE IF (N.LT.0) THEN
          INFO = 4
      ELSE IF (K.LT.0) THEN
          INFO = 5
      ELSE IF (LDA.LT.MAX(1,NROWA)) THEN
          INFO = 8
      ELSE IF (LDB.LT.MAX(1,NROWB)) THEN
          INFO = 10
      ELSE IF (LDC.LT.MAX(1,M)) THEN
          INFO = 13
      END IF
      IF (INFO.NE.0) THEN
          CALL XERBLA('DGEMM ',INFO)
          RETURN
      END IF
!
!     Quick return if possible.
!
      IF ((M.EQ.0) .OR. (N.EQ.0) .OR. &
     &    (((ALPHA.EQ.ZERO).OR. (K.EQ.0)).AND. (BETA.EQ.ONE))) RETURN
!
!     And if  alpha.eq.zero.
!
      IF (ALPHA.EQ.ZERO) THEN
          IF (BETA.EQ.ZERO) THEN
              DO 20 J = 1,N
                  DO 10 I = 1,M
                      C(I,J) = ZERO
   10             CONTINUE
   20         CONTINUE
          ELSE
```

```
              DO 40 J = 1,N
                  DO 30 I = 1,M
                      C(I,J) = BETA*C(I,J)
   30             CONTINUE
   40         CONTINUE
          END IF
          RETURN
      END IF
!
!     Start the operations.
!
      IF (NOTB) THEN
          IF (NOTA) THEN
!
!           Form  C := alpha*A*B + beta*C.
!
              DO 90 J = 1,N
                  IF (BETA.EQ.ZERO) THEN
                      DO 50 I = 1,M
                          C(I,J) = ZERO
   50                 CONTINUE
                  ELSE IF (BETA.NE.ONE) THEN
                      DO 60 I = 1,M
                          C(I,J) = BETA*C(I,J)
   60                 CONTINUE
                  END IF
                  DO 80 L = 1,K
                      TEMP = ALPHA*B(L,J)
                      DO 70 I = 1,M
                          C(I,J) = C(I,J) + TEMP*A(I,L)
   70                 CONTINUE
   80             CONTINUE
   90         CONTINUE
          ELSE
!
!           Form  C := alpha*A**T*B + beta*C
!
              DO 120 J = 1,N
                  DO 110 I = 1,M
                      TEMP = ZERO
                      DO 100 L = 1,K
                          TEMP = TEMP + A(L,I)*B(L,J)
  100                 CONTINUE
                      IF (BETA.EQ.ZERO) THEN
                          C(I,J) = ALPHA*TEMP
                      ELSE
                          C(I,J) = ALPHA*TEMP + BETA*C(I,J)
                      END IF
  110             CONTINUE
  120         CONTINUE
          END IF
      ELSE
          IF (NOTA) THEN
!
!           Form  C := alpha*A*B**T + beta*C
!
              DO 170 J = 1,N
                  IF (BETA.EQ.ZERO) THEN
                      DO 130 I = 1,M
                          C(I,J) = ZERO
  130                 CONTINUE
```

```fortran
                  ELSE IF (BETA.NE.ONE) THEN
                      DO 140 I = 1,M
                          C(I,J) = BETA*C(I,J)
  140                 CONTINUE
                  END IF
                  DO 160 L = 1,K
                      TEMP = ALPHA*B(J,L)
                      DO 150 I = 1,M
                          C(I,J) = C(I,J) + TEMP*A(I,L)
  150                 CONTINUE
  160             CONTINUE
  170         CONTINUE
          ELSE
!
!           Form  C := alpha*A**T*B**T + beta*C
!
              DO 200 J = 1,N
                  DO 190 I = 1,M
                      TEMP = ZERO
                      DO 180 L = 1,K
                          TEMP = TEMP + A(L,I)*B(J,L)
  180                 CONTINUE
                      IF (BETA.EQ.ZERO) THEN
                          C(I,J) = ALPHA*TEMP
                      ELSE
                          C(I,J) = ALPHA*TEMP + BETA*C(I,J)
                      END IF
  190             CONTINUE
  200         CONTINUE
          END IF
      END IF
!
      RETURN
!
!     End of DGEMM .
!
      END SUBROUTINE DGEMM

!
!  =====================================================================
      SUBROUTINE DTRSM(SIDE,UPLO,TRANSA,DIAG,M,N,ALPHA,A,LDA,B,LDB)
!
!  -- Reference BLAS level3 routine (version 3.7.0) --
!  -- Reference BLAS is a software package provided by Univ. of Tennessee,    --
!  -- Univ. of California Berkeley, Univ. of Colorado Denver and NAG Ltd..--
!     December 2016
!
!     .. Scalar Arguments ..
      DOUBLE PRECISION ALPHA
      INTEGER LDA,LDB,M,N
      CHARACTER DIAG,SIDE,TRANSA,UPLO
!     ..
!     .. Array Arguments ..
      DOUBLE PRECISION A(LDA,*),B(LDB,*)
!     ..
!
!  =====================================================================
!
!     .. External Functions ..
!      LOGICAL LSAME
!      EXTERNAL LSAME
```

```
!      ..
!      .. External Subroutines ..
!       EXTERNAL XERBLA
!      ..
!      .. Intrinsic Functions ..
       INTRINSIC MAX
!      ..
!      .. Local Scalars ..
       DOUBLE PRECISION TEMP
       INTEGER I,INFO,J,K,NROWA
       LOGICAL LSIDE,NOUNIT,UPPER
!      ..
!      .. Parameters ..
       DOUBLE PRECISION ONE,ZERO
       PARAMETER (ONE=1.0D+0,ZERO=0.0D+0)
!      ..
!
!      Test the input parameters.
!
       LSIDE = LSAME(SIDE,'L')
       IF (LSIDE) THEN
           NROWA = M
       ELSE
           NROWA = N
       END IF
       NOUNIT = LSAME(DIAG,'N')
       UPPER = LSAME(UPLO,'U')
!
       INFO = 0
       IF ((.NOT.LSIDE) .AND. (.NOT.LSAME(SIDE,'R'))) THEN
           INFO = 1
       ELSE IF ((.NOT.UPPER) .AND. (.NOT.LSAME(UPLO,'L'))) THEN
           INFO = 2
       ELSE IF ((.NOT.LSAME(TRANSA,'N')) .AND.&
                (.NOT.LSAME(TRANSA,'T')) .AND.&
                (.NOT.LSAME(TRANSA,'C'))) THEN
           INFO = 3
       ELSE IF ((.NOT.LSAME(DIAG,'U')) .AND. (.NOT.LSAME(DIAG,'N'))) THEN
           INFO = 4
       ELSE IF (M.LT.0) THEN
           INFO = 5
       ELSE IF (N.LT.0) THEN
           INFO = 6
       ELSE IF (LDA.LT.MAX(1,NROWA)) THEN
           INFO = 9
       ELSE IF (LDB.LT.MAX(1,M)) THEN
           INFO = 11
       END IF
       IF (INFO.NE.0) THEN
           CALL XERBLA('DTRSM ',INFO)
           RETURN
       END IF
!
!      Quick return if possible.
!
       IF (M.EQ.0 .OR. N.EQ.0) RETURN
!
!      And when  alpha.eq.zero.
!
       IF (ALPHA.EQ.ZERO) THEN
           DO 20 J = 1,N
```

```
              DO 10 I = 1,M
                  B(I,J) = ZERO
   10         CONTINUE
   20     CONTINUE
          RETURN
      END IF
!
!     Start the operations.
!
      IF (LSIDE) THEN
          IF (LSAME(TRANSA,'N')) THEN
!
!           Form  B := alpha*inv( A )*B.
!
              IF (UPPER) THEN
                  DO 60 J = 1,N
                      IF (ALPHA.NE.ONE) THEN
                          DO 30 I = 1,M
                              B(I,J) = ALPHA*B(I,J)
   30                     CONTINUE
                      END IF
                      DO 50 K = M,1,-1
                          IF (B(K,J).NE.ZERO) THEN
                              IF (NOUNIT) B(K,J) = B(K,J)/A(K,K)
                              DO 40 I = 1,K - 1
                                  B(I,J) = B(I,J) - B(K,J)*A(I,K)
   40                         CONTINUE
                          END IF
   50                 CONTINUE
   60             CONTINUE
              ELSE
                  DO 100 J = 1,N
                      IF (ALPHA.NE.ONE) THEN
                          DO 70 I = 1,M
                              B(I,J) = ALPHA*B(I,J)
   70                     CONTINUE
                      END IF
                      DO 90 K = 1,M
                          IF (B(K,J).NE.ZERO) THEN
                              IF (NOUNIT) B(K,J) = B(K,J)/A(K,K)
                              DO 80 I = K + 1,M
                                  B(I,J) = B(I,J) - B(K,J)*A(I,K)
   80                         CONTINUE
                          END IF
   90                 CONTINUE
  100             CONTINUE
              END IF
          ELSE
!
!           Form  B := alpha*inv( A**T )*B.
!
              IF (UPPER) THEN
                  DO 130 J = 1,N
                      DO 120 I = 1,M
                          TEMP = ALPHA*B(I,J)
                          DO 110 K = 1,I - 1
                              TEMP = TEMP - A(K,I)*B(K,J)
  110                     CONTINUE
                          IF (NOUNIT) TEMP = TEMP/A(I,I)
                          B(I,J) = TEMP
  120                 CONTINUE
```

```
  130             CONTINUE
              ELSE
                  DO 160 J = 1,N
                      DO 150 I = M,1,-1
                          TEMP = ALPHA*B(I,J)
                          DO 140 K = I + 1,M
                              TEMP = TEMP - A(K,I)*B(K,J)
  140                     CONTINUE
                          IF (NOUNIT) TEMP = TEMP/A(I,I)
                          B(I,J) = TEMP
  150                 CONTINUE
  160             CONTINUE
              END IF
          END IF
      ELSE
          IF (LSAME(TRANSA,'N')) THEN
!
!           Form  B := alpha*B*inv( A ).
!
              IF (UPPER) THEN
                  DO 210 J = 1,N
                      IF (ALPHA.NE.ONE) THEN
                          DO 170 I = 1,M
                              B(I,J) = ALPHA*B(I,J)
  170                     CONTINUE
                      END IF
                      DO 190 K = 1,J - 1
                          IF (A(K,J).NE.ZERO) THEN
                              DO 180 I = 1,M
                                  B(I,J) = B(I,J) - A(K,J)*B(I,K)
  180                         CONTINUE
                          END IF
  190                 CONTINUE
                      IF (NOUNIT) THEN
                          TEMP = ONE/A(J,J)
                          DO 200 I = 1,M
                              B(I,J) = TEMP*B(I,J)
  200                     CONTINUE
                      END IF
  210             CONTINUE
              ELSE
                  DO 260 J = N,1,-1
                      IF (ALPHA.NE.ONE) THEN
                          DO 220 I = 1,M
                              B(I,J) = ALPHA*B(I,J)
  220                     CONTINUE
                      END IF
                      DO 240 K = J + 1,N
                          IF (A(K,J).NE.ZERO) THEN
                              DO 230 I = 1,M
                                  B(I,J) = B(I,J) - A(K,J)*B(I,K)
  230                         CONTINUE
                          END IF
  240                 CONTINUE
                      IF (NOUNIT) THEN
                          TEMP = ONE/A(J,J)
                          DO 250 I = 1,M
                              B(I,J) = TEMP*B(I,J)
  250                     CONTINUE
                      END IF
  260             CONTINUE
```

```
            END IF
         ELSE
!
!           Form  B := alpha*B*inv( A**T ).
!
            IF (UPPER) THEN
               DO 310 K = N,1,-1
                  IF (NOUNIT) THEN
                     TEMP = ONE/A(K,K)
                     DO 270 I = 1,M
                        B(I,K) = TEMP*B(I,K)
  270                CONTINUE
                  END IF
                  DO 290 J = 1,K - 1
                     IF (A(J,K).NE.ZERO) THEN
                        TEMP = A(J,K)
                        DO 280 I = 1,M
                           B(I,J) = B(I,J) - TEMP*B(I,K)
  280                   CONTINUE
                     END IF
  290             CONTINUE
                  IF (ALPHA.NE.ONE) THEN
                     DO 300 I = 1,M
                        B(I,K) = ALPHA*B(I,K)
  300                CONTINUE
                  END IF
  310          CONTINUE
            ELSE
               DO 360 K = 1,N
                  IF (NOUNIT) THEN
                     TEMP = ONE/A(K,K)
                     DO 320 I = 1,M
                        B(I,K) = TEMP*B(I,K)
  320                CONTINUE
                  END IF
                  DO 340 J = K + 1,N
                     IF (A(J,K).NE.ZERO) THEN
                        TEMP = A(J,K)
                        DO 330 I = 1,M
                           B(I,J) = B(I,J) - TEMP*B(I,K)
  330                   CONTINUE
                     END IF
  340             CONTINUE
                  IF (ALPHA.NE.ONE) THEN
                     DO 350 I = 1,M
                        B(I,K) = ALPHA*B(I,K)
  350                CONTINUE
                  END IF
  360          CONTINUE
            END IF
         END IF
      END IF
!
      RETURN
!
!     End of DTRSM .
!
      END SUBROUTINE DTRSM
!
!  ======================================================================
      SUBROUTINE XERBLA( SRNAME, INFO )
```

```
!
!  -- Reference BLAS level1 routine (version 3.7.0) --
!  -- Reference BLAS is a software package provided by Univ. of Tennessee,    --
!  -- Univ. of California Berkeley, Univ. of Colorado Denver and NAG Ltd..--
!     December 2016
!
!     .. Scalar Arguments ..
      CHARACTER*(*)      SRNAME
      INTEGER            INFO
!     ..
!
! =====================================================================
!
!     .. Intrinsic Functions ..
      INTRINSIC          LEN_TRIM
!     ..
!     .. Executable Statements ..
!
      WRITE( *, FMT = 9999 )SRNAME( 1:LEN_TRIM( SRNAME ) ), INFO
!
      STOP
!
 9999 FORMAT( ' ** On entry to ', A, ' parameter number ', I2, ' had ',&
            'an illegal value' )
!
!     End of XERBLA
!
      END SUBROUTINE XERBLA

!  =====================================================================
      LOGICAL FUNCTION LSAME(CA,CB)
!
!  -- Reference BLAS level1 routine (version 3.1) --
!  -- Reference BLAS is a software package provided by Univ. of Tennessee,    --
!  -- Univ. of California Berkeley, Univ. of Colorado Denver and NAG Ltd..--
!     December 2016
!
!     .. Scalar Arguments ..
      CHARACTER CA,CB
!     ..
!
! =====================================================================
!
!     .. Intrinsic Functions ..
      INTRINSIC ICHAR
!     ..
!     .. Local Scalars ..
      INTEGER INTA,INTB,ZCODE
!     ..
!
!     Test if the characters are equal
!
      LSAME = CA .EQ. CB
      IF (LSAME) RETURN
!
!     Now test for equivalence if both characters are alphabetic.
!
      ZCODE = ICHAR('Z')
!
!     Use 'Z' rather than 'A' so that ASCII can be detected on Prime
!     machines, on which ICHAR returns a value with bit 8 set.
```

```fortran
!     ICHAR('A') on Prime machines returns 193 which is the same as
!     ICHAR('A') on an EBCDIC machine.
!
      INTA = ICHAR(CA)
      INTB = ICHAR(CB)
!
      IF (ZCODE.EQ.90 .OR. ZCODE.EQ.122) THEN
!
!        ASCII is assumed - ZCODE is the ASCII code of either lower or
!        upper case 'Z'.
!
          IF (INTA.GE.97 .AND. INTA.LE.122) INTA = INTA - 32
          IF (INTB.GE.97 .AND. INTB.LE.122) INTB = INTB - 32
!
      ELSE IF (ZCODE.EQ.233 .OR. ZCODE.EQ.169) THEN
!
!        EBCDIC is assumed - ZCODE is the EBCDIC code of either lower or
!        upper case 'Z'.
!
          IF (INTA.GE.129 .AND. INTA.LE.137 .OR.&
     &        INTA.GE.145 .AND. INTA.LE.153 .OR.&
     &        INTA.GE.162 .AND. INTA.LE.169) INTA = INTA + 64
          IF (INTB.GE.129 .AND. INTB.LE.137 .OR.&
     &        INTB.GE.145 .AND. INTB.LE.153 .OR.&
     &        INTB.GE.162 .AND. INTB.LE.169) INTB = INTB + 64
!
      ELSE IF (ZCODE.EQ.218 .OR. ZCODE.EQ.250) THEN
!
!        ASCII is assumed, on Prime machines - ZCODE is the ASCII code
!        plus 128 of either lower or upper case 'Z'.
!
          IF (INTA.GE.225 .AND. INTA.LE.250) INTA = INTA - 32
          IF (INTB.GE.225 .AND. INTB.LE.250) INTB = INTB - 32
      END IF
      LSAME = INTA .EQ. INTB
!
!     RETURN
!
!     End of LSAME
!
      END FUNCTION LSAME



!  =====================================================================
      SUBROUTINE DLASWP( N, A, LDA, K1, K2, IPIV, INCX )
!
!  -- LAPACK auxiliary routine (version 3.7.1) --
!  -- LAPACK is a software package provided by Univ. of Tennessee,    --
!  -- Univ. of California Berkeley, Univ. of Colorado Denver and NAG Ltd..--
!     June 2017
!
!     .. Scalar Arguments ..
      INTEGER            INCX, K1, K2, LDA, N
!     ..
!     .. Array Arguments ..
      INTEGER            IPIV( * )
      DOUBLE PRECISION   A( LDA, * )
!     ..
!
! =====================================================================
```

```
!
!     .. Local Scalars ..
      INTEGER            I, I1, I2, INC, IP, IX, IX0, J, K, N32
      DOUBLE PRECISION   TEMP
!     ..
!     .. Executable Statements ..
!
!     Interchange row I with row IPIV(K1+(I-K1)*abs(INCX)) for each of rows
!     K1 through K2.
!
      IF( INCX.GT.0 ) THEN
         IX0 = K1
         I1 = K1
         I2 = K2
         INC = 1
      ELSE IF( INCX.LT.0 ) THEN
         IX0 = K1 + ( K1-K2 )*INCX
         I1 = K2
         I2 = K1
         INC = -1
      ELSE
         RETURN
      END IF
!
      N32 = ( N / 32 )*32
      IF( N32.NE.0 ) THEN
         DO 30 J = 1, N32, 32
            IX = IX0
            DO 20 I = I1, I2, INC
               IP = IPIV( IX )
               IF( IP.NE.I ) THEN
                  DO 10 K = J, J + 31
                     TEMP = A( I, K )
                     A( I, K ) = A( IP, K )
                     A( IP, K ) = TEMP
   10             CONTINUE
               END IF
               IX = IX + INCX
   20       CONTINUE
   30    CONTINUE
      END IF
      IF( N32.NE.N ) THEN
         N32 = N32 + 1
         IX = IX0
         DO 50 I = I1, I2, INC
            IP = IPIV( IX )
            IF( IP.NE.I ) THEN
               DO 40 K = N32, N
                  TEMP = A( I, K )
                  A( I, K ) = A( IP, K )
                  A( IP, K ) = TEMP
   40          CONTINUE
            END IF
            IX = IX + INCX
   50    CONTINUE
      END IF
!
      RETURN
!
!     End of DLASWP
!
```

```
      END SUBROUTINE DLASWP

!  =====================================================================
      SUBROUTINE DSCAL(N,DA,DX,INCX)
!
!  -- Reference BLAS level1 routine (version 3.8.0) --
!  -- Reference BLAS is a software package provided by Univ. of Tennessee,    --
!  -- Univ. of California Berkeley, Univ. of Colorado Denver and NAG Ltd..--
!     November 2017
!
!     .. Scalar Arguments ..
      DOUBLE PRECISION DA
      INTEGER INCX,N
!     ..
!     .. Array Arguments ..
      DOUBLE PRECISION DX(*)
!     ..
!
!  =====================================================================
!
!     .. Local Scalars ..
      INTEGER I,M,MP1,NINCX
!     ..
!     .. Intrinsic Functions ..
      INTRINSIC MOD
!     ..
      IF (N.LE.0 .OR. INCX.LE.0) RETURN
      IF (INCX.EQ.1) THEN
!
!        code for increment equal to 1
!
!
!        clean-up loop
!
         M = MOD(N,5)
         IF (M.NE.0) THEN
            DO I = 1,M
               DX(I) = DA*DX(I)
            END DO
            IF (N.LT.5) RETURN
         END IF
         MP1 = M + 1
         DO I = MP1,N,5
            DX(I) = DA*DX(I)
            DX(I+1) = DA*DX(I+1)
            DX(I+2) = DA*DX(I+2)
            DX(I+3) = DA*DX(I+3)
            DX(I+4) = DA*DX(I+4)
         END DO
      ELSE
!
!        code for increment not equal to 1
!
         NINCX = N*INCX
         DO I = 1,NINCX,INCX
            DX(I) = DA*DX(I)
         END DO
      END IF
      RETURN
      END SUBROUTINE DSCAL

```

```
!
!  =====================================================================
      SUBROUTINE DGETF2( M, N, A, LDA, IPIV, INFO )
!
!  -- LAPACK computational routine (version 3.7.0) --
!  -- LAPACK is a software package provided by Univ. of Tennessee,    --
!  -- Univ. of California Berkeley, Univ. of Colorado Denver and NAG Ltd..--
!     December 2016
!
!     .. Scalar Arguments ..
      INTEGER            INFO, LDA, M, N
!     ..
!     .. Array Arguments ..
      INTEGER            IPIV( * )
      DOUBLE PRECISION   A( LDA, * )
!     ..
!
!  =====================================================================
!
!     .. Parameters ..
      DOUBLE PRECISION   ONE, ZERO
      PARAMETER          ( ONE = 1.0D+0, ZERO = 0.0D+0 )
!     ..
!     .. Local Scalars ..
      DOUBLE PRECISION   SFMIN
      INTEGER            I, J, JP
!     ..
!     .. External Functions ..
!      DOUBLE PRECISION   DLAMCH
!      INTEGER            IDAMAX
!      EXTERNAL           DLAMCH, IDAMAX
!     ..
!     .. External Subroutines ..
!      EXTERNAL           DGER, DSCAL, DSWAP, XERBLA
!     ..
!     .. Intrinsic Functions ..
      INTRINSIC          MAX, MIN
!     ..
!     .. Executable Statements ..
!
!     Test the input parameters.
!
      INFO = 0
      IF( M.LT.0 ) THEN
         INFO = -1
      ELSE IF( N.LT.0 ) THEN
         INFO = -2
      ELSE IF( LDA.LT.MAX( 1, M ) ) THEN
         INFO = -4
      END IF
      IF( INFO.NE.0 ) THEN
         CALL XERBLA( 'DGETF2', -INFO )
         RETURN
      END IF
!
!     Quick return if possible
!
      IF( M.EQ.0 .OR. N.EQ.0 )&
         RETURN
!
```

```
!     Compute machine safe minimum
!
      SFMIN = DLAMCH('S')
!
      DO 10 J = 1, MIN( M, N )
!
!        Find pivot and test for singularity.
!
         JP = J - 1 + IDAMAX( M-J+1, A( J, J ), 1 )
         IPIV( J ) = JP
         IF( A( JP, J ).NE.ZERO ) THEN
!
!           Apply the interchange to columns 1:N.
!
            IF( JP.NE.J )&
               CALL DSWAP( N, A( J, 1 ), LDA, A( JP, 1 ), LDA )
!
!           Compute elements J+1:M of J-th column.
!
            IF( J.LT.M ) THEN
               IF( ABS(A( J, J )) .GE. SFMIN ) THEN
                  CALL DSCAL( M-J, ONE / A( J, J ), A( J+1, J ), 1 )
               ELSE
                 DO 20 I = 1, M-J
                    A( J+I, J ) = A( J+I, J ) / A( J, J )
  20             CONTINUE
               END IF
            END IF
!
         ELSE IF( INFO.EQ.0 ) THEN
!
            INFO = J
         END IF
!
         IF( J.LT.MIN( M, N ) ) THEN
!
!           Update trailing submatrix.
!
            CALL DGER( M-J, N-J, -ONE, A( J+1, J ), 1, A( J, J+1 ), LDA,&
                       A( J+1, J+1 ), LDA )
         END IF
  10  CONTINUE
      RETURN
!
!     End of DGETF2
!
      END SUBROUTINE DGETF2
!
!  =====================================================================
      SUBROUTINE DGETRF( M, N, A, LDA, IPIV, INFO )
!
!  -- LAPACK computational routine (version 3.7.0) --
!  -- LAPACK is a software package provided by Univ. of Tennessee,    --
!  -- Univ. of California Berkeley, Univ. of Colorado Denver and NAG Ltd..--
!     December 2016
!
!     .. Scalar Arguments ..
      INTEGER            INFO, LDA, M, N
!     ..
!     .. Array Arguments ..
      INTEGER            IPIV( * )
```

```fortran
      DOUBLE PRECISION   A( LDA, * )
!     ..
!
!  =====================================================================
!
!     .. Parameters ..
      DOUBLE PRECISION   ONE
      PARAMETER          ( ONE = 1.0D+0 )
!     ..
!     .. Local Scalars ..
      INTEGER            I, IINFO, J, JB, NB
!     ..
!     .. External Subroutines ..
!      EXTERNAL           DGEMM, DGETRF2, DLASWP, DTRSM, XERBLA
!     ..
!     .. External Functions ..
!      INTEGER            ILAENV
!      EXTERNAL           ILAENV
!     ..
!     .. Intrinsic Functions ..
      INTRINSIC          MAX, MIN
!     ..
!     .. Executable Statements ..
!
!     Test the input parameters.
!
      INFO = 0
      IF( M.LT.0 ) THEN
         INFO = -1
      ELSE IF( N.LT.0 ) THEN
         INFO = -2
      ELSE IF( LDA.LT.MAX( 1, M ) ) THEN
         INFO = -4
      END IF
      IF( INFO.NE.0 ) THEN
         CALL XERBLA( 'DGETRF', -INFO )
         RETURN
      END IF
!
!     Quick return if possible
!
      IF( M.EQ.0 .OR. N.EQ.0 )&
         RETURN
!
!     Determine the block size for this environment.
!
      NB = ILAENV( 1, 'DGETRF', ' ', M, N, -1, -1 )
      IF( NB.LE.1 .OR. NB.GE.MIN( M, N ) ) THEN
!
!        Use unblocked code.
!
         CALL DGETRF2( M, N, A, LDA, IPIV, INFO )
      ELSE
!
!        Use blocked code.
!
         DO 20 J = 1, MIN( M, N ), NB
            JB = MIN( MIN( M, N )-J+1, NB )
!
!           Factor diagonal and subdiagonal blocks and test for exact
!           singularity.
```

```
!
            CALL DGETRF2( M-J+1, JB, A( J, J ), LDA, IPIV( J ), IINFO )
!
            Adjust INFO and the pivot indices.
!
            IF( INFO.EQ.0 .AND. IINFO.GT.0 )&
               INFO = IINFO + J - 1
            DO 10 I = J, MIN( M, J+JB-1 )
               IPIV( I ) = J - 1 + IPIV( I )
   10       CONTINUE
!
            Apply interchanges to columns 1:J-1.
!
            CALL DLASWP( J-1, A, LDA, J, J+JB-1, IPIV, 1 )
!
            IF( J+JB.LE.N ) THEN
!
               Apply interchanges to columns J+JB:N.
!
               CALL DLASWP( N-J-JB+1, A( 1, J+JB ), LDA, J, J+JB-1,&
                            IPIV, 1 )
!
               Compute block row of U.
!
               CALL DTRSM( 'Left', 'Lower', 'No transpose', 'Unit', JB,&
                           N-J-JB+1, ONE, A( J, J ), LDA, A( J, J+JB ),&
                           LDA )
               IF( J+JB.LE.M ) THEN
!
                  Update trailing submatrix.
!
                  CALL DGEMM( 'No transpose', 'No transpose', M-J-JB+1,&
                              N-J-JB+1, JB, -ONE, A( J+JB, J ), LDA,   &
                              A( J, J+JB ), LDA, ONE, A( J+JB, J+JB ), &
                              LDA )
               END IF
            END IF
   20    CONTINUE
      END IF
      RETURN
!
      End of DGETRF
!
      END SUBROUTINE DGETRF
!
!  ====================================================================
      SUBROUTINE DGER(M,N,ALPHA,X,INCX,Y,INCY,A,LDA)
!
!  -- Reference BLAS level2 routine (version 3.7.0) --
!  -- Reference BLAS is a software package provided by Univ. of Tennessee,    --
!  -- Univ. of California Berkeley, Univ. of Colorado Denver and NAG Ltd..--
      December 2016
!
      .. Scalar Arguments ..
      DOUBLE PRECISION ALPHA
      INTEGER INCX,INCY,LDA,M,N
!     ..
!     .. Array Arguments ..
      DOUBLE PRECISION A(LDA,*),X(*),Y(*)
!     ..
!
```

```fortran
!  ====================================================================
!
!     .. Parameters ..
      DOUBLE PRECISION ZERO
      PARAMETER (ZERO=0.0D+0)
!     ..
!     .. Local Scalars ..
      DOUBLE PRECISION TEMP
      INTEGER I,INFO,IX,J,JY,KX
!     ..
!     .. External Subroutines ..
!      EXTERNAL XERBLA
!     ..
!     .. Intrinsic Functions ..
      INTRINSIC MAX
!     ..
!
!     Test the input parameters.
!
      INFO = 0
      IF (M.LT.0) THEN
          INFO = 1
      ELSE IF (N.LT.0) THEN
          INFO = 2
      ELSE IF (INCX.EQ.0) THEN
          INFO = 5
      ELSE IF (INCY.EQ.0) THEN
          INFO = 7
      ELSE IF (LDA.LT.MAX(1,M)) THEN
          INFO = 9
      END IF
      IF (INFO.NE.0) THEN
          CALL XERBLA('DGER  ',INFO)
          RETURN
      END IF
!
!     Quick return if possible.
!
      IF ((M.EQ.0) .OR. (N.EQ.0) .OR. (ALPHA.EQ.ZERO)) RETURN
!
!     Start the operations. In this version the elements of A are
!     accessed sequentially with one pass through A.
!
      IF (INCY.GT.0) THEN
          JY = 1
      ELSE
          JY = 1 - (N-1)*INCY
      END IF
      IF (INCX.EQ.1) THEN
          DO 20 J = 1,N
              IF (Y(JY).NE.ZERO) THEN
                  TEMP = ALPHA*Y(JY)
                  DO 10 I = 1,M
                      A(I,J) = A(I,J) + X(I)*TEMP
   10             CONTINUE
              END IF
              JY = JY + INCY
   20     CONTINUE
      ELSE
          IF (INCX.GT.0) THEN
              KX = 1
```

```
        ELSE
            KX = 1 - (M-1)*INCX
        END IF
        DO 40 J = 1,N
            IF (Y(JY).NE.ZERO) THEN
                TEMP = ALPHA*Y(JY)
                IX = KX
                DO 30 I = 1,M
                    A(I,J) = A(I,J) + X(IX)*TEMP
                    IX = IX + INCX
30              CONTINUE
            END IF
            JY = JY + INCY
40      CONTINUE
    END IF
!
    RETURN
!
!   End of DGER  .
!
    END SUBROUTINE DGER

!
! =====================================================================
    DOUBLE PRECISION FUNCTION DLAMCH( CMACH )
!
! -- LAPACK auxiliary routine (version 3.7.0) --
! -- LAPACK is a software package provided by Univ. of Tennessee,    --
! -- Univ. of California Berkeley, Univ. of Colorado Denver and NAG Ltd..--
!    December 2016
!
!    .. Scalar Arguments ..
    CHARACTER          CMACH
!    ..
!
! =====================================================================
!
!    .. Parameters ..
    DOUBLE PRECISION   ONE, ZERO
    PARAMETER          ( ONE = 1.0D+0, ZERO = 0.0D+0 )
!    ..
!    .. Local Scalars ..
    DOUBLE PRECISION   RND, EPS, SFMIN, SMALL, RMACH
!    ..
!    .. External Functions ..
!     LOGICAL            LSAME
!     EXTERNAL           LSAME
!    ..
!    .. Intrinsic Functions ..
    INTRINSIC          DIGITS, EPSILON, HUGE, MAXEXPONENT,&
                       MINEXPONENT, RADIX, TINY
!    ..
!    .. Executable Statements ..
!
!
!    Assume rounding, not chopping. Always.
!
    RND = ONE
!
    IF( ONE.EQ.RND ) THEN
       EPS = EPSILON(ZERO) * 0.5
```

```
      ELSE
         EPS = EPSILON(ZERO)
      END IF
!
      IF( LSAME( CMACH, 'E' ) ) THEN
         RMACH = EPS
      ELSE IF( LSAME( CMACH, 'S' ) ) THEN
         SFMIN = TINY(ZERO)
         SMALL = ONE / HUGE(ZERO)
         IF( SMALL.GE.SFMIN ) THEN
!
!           Use SMALL plus a bit, to avoid the possibility of rounding
!           causing overflow when computing  1/sfmin.
!
            SFMIN = SMALL*( ONE+EPS )
         END IF
         RMACH = SFMIN
      ELSE IF( LSAME( CMACH, 'B' ) ) THEN
         RMACH = RADIX(ZERO)
      ELSE IF( LSAME( CMACH, 'P' ) ) THEN
         RMACH = EPS * RADIX(ZERO)
      ELSE IF( LSAME( CMACH, 'N' ) ) THEN
         RMACH = DIGITS(ZERO)
      ELSE IF( LSAME( CMACH, 'R' ) ) THEN
         RMACH = RND
      ELSE IF( LSAME( CMACH, 'M' ) ) THEN
         RMACH = MINEXPONENT(ZERO)
      ELSE IF( LSAME( CMACH, 'U' ) ) THEN
         RMACH = tiny(zero)
      ELSE IF( LSAME( CMACH, 'L' ) ) THEN
         RMACH = MAXEXPONENT(ZERO)
      ELSE IF( LSAME( CMACH, 'O' ) ) THEN
         RMACH = HUGE(ZERO)
      ELSE
         RMACH = ZERO
      END IF
!
      DLAMCH = RMACH
      RETURN
!
!     End of DLAMCH
!
      END FUNCTION DLAMCH

!
!  =====================================================================
      INTEGER FUNCTION IDAMAX(N,DX,INCX)
!
!  -- Reference BLAS level1 routine (version 3.8.0) --
!  -- Reference BLAS is a software package provided by Univ. of Tennessee,    --
!  -- Univ. of California Berkeley, Univ. of Colorado Denver and NAG Ltd..--
!     November 2017
!
!     .. Scalar Arguments ..
      INTEGER INCX,N
!     ..
!     .. Array Arguments ..
      DOUBLE PRECISION DX(*)
!     ..
!
!  =====================================================================
```

```
!
!     .. Local Scalars ..
      DOUBLE PRECISION DMAX
      INTEGER I,IX
!     ..
!     .. Intrinsic Functions ..
      INTRINSIC DABS
!     ..
      IDAMAX = 0
      IF (N.LT.1 .OR. INCX.LE.0) RETURN
      IDAMAX = 1
      IF (N.EQ.1) RETURN
      IF (INCX.EQ.1) THEN
!
!        code for increment equal to 1
!
         DMAX = DABS(DX(1))
         DO I = 2,N
            IF (DABS(DX(I)).GT.DMAX) THEN
               IDAMAX = I
               DMAX = DABS(DX(I))
            END IF
         END DO
      ELSE
!
!        code for increment not equal to 1
!
         IX = 1
         DMAX = DABS(DX(1))
         IX = IX + INCX
         DO I = 2,N
            IF (DABS(DX(IX)).GT.DMAX) THEN
               IDAMAX = I
               DMAX = DABS(DX(IX))
            END IF
            IX = IX + INCX
         END DO
      END IF
      RETURN
      END FUNCTION IDAMAX

!
!  =====================================================================
      RECURSIVE SUBROUTINE DGETRF2( M, N, A, LDA, IPIV, INFO )
!
!  -- LAPACK computational routine (version 3.7.0) --
!  -- LAPACK is a software package provided by Univ. of Tennessee,    --
!  -- Univ. of California Berkeley, Univ. of Colorado Denver and NAG Ltd..--
!     June 2016
!
!     .. Scalar Arguments ..
      INTEGER            INFO, LDA, M, N
!     ..
!     .. Array Arguments ..
      INTEGER            IPIV( * )
      DOUBLE PRECISION   A( LDA, * )
!     ..
!
!  =====================================================================
!
!     .. Parameters ..
```

```
      DOUBLE PRECISION   ONE, ZERO
      PARAMETER          ( ONE = 1.0D+0, ZERO = 0.0D+0 )
!     ..
!     .. Local Scalars ..
      DOUBLE PRECISION   SFMIN, TEMP
      INTEGER            I, IINFO, N1, N2
!     ..
!     .. External Functions ..
!      DOUBLE PRECISION   DLAMCH
!      INTEGER            IDAMAX
!      EXTERNAL           DLAMCH, IDAMAX
!     ..
!     .. External Subroutines ..
!      EXTERNAL           DGEMM, DSCAL, DLASWP, DTRSM, XERBLA
!     ..
!     .. Intrinsic Functions ..
      INTRINSIC          MAX, MIN
!     ..
!     .. Executable Statements ..
!
!     Test the input parameters
!
      INFO = 0
      IF( M.LT.0 ) THEN
         INFO = -1
      ELSE IF( N.LT.0 ) THEN
         INFO = -2
      ELSE IF( LDA.LT.MAX( 1, M ) ) THEN
         INFO = -4
      END IF
      IF( INFO.NE.0 ) THEN
         CALL XERBLA( 'DGETRF2', -INFO )
         RETURN
      END IF
!
!     Quick return if possible
!
      IF( M.EQ.0 .OR. N.EQ.0 )&
         RETURN

      IF ( M.EQ.1 ) THEN
!
!        Use unblocked code for one row case
!        Just need to handle IPIV and INFO
!
         IPIV( 1 ) = 1
         IF ( A(1,1).EQ.ZERO )&
            INFO = 1
!
      ELSE IF( N.EQ.1 ) THEN
!
!        Use unblocked code for one column case
!
!
!        Compute machine safe minimum
!
         SFMIN = DLAMCH('S')
!
!        Find pivot and test for singularity
!
         I = IDAMAX( M, A( 1, 1 ), 1 )
```

```
         IPIV( 1 ) = I
         IF( A( I, 1 ).NE.ZERO ) THEN
!
!           Apply the interchange
!
            IF( I.NE.1 ) THEN
               TEMP = A( 1, 1 )
               A( 1, 1 ) = A( I, 1 )
               A( I, 1 ) = TEMP
            END IF
!
!           Compute elements 2:M of the column
!
            IF( ABS(A( 1, 1 )) .GE. SFMIN ) THEN
               CALL DSCAL( M-1, ONE / A( 1, 1 ), A( 2, 1 ), 1 )
            ELSE
               DO 10 I = 1, M-1
                  A( 1+I, 1 ) = A( 1+I, 1 ) / A( 1, 1 )
   10          CONTINUE
            END IF
!
         ELSE
            INFO = 1
         END IF
!
      ELSE
!
!        Use recursive code
!
         N1 = MIN( M, N ) / 2
         N2 = N-N1
!
!               [ A11 ]
!        Factor [ --- ]
!               [ A21 ]
!
         CALL DGETRF2( M, N1, A, LDA, IPIV, IINFO )

         IF ( INFO.EQ.0 .AND. IINFO.GT.0 )&
            INFO = IINFO
!
!                              [ A12 ]
!        Apply interchanges to [ --- ]
!                              [ A22 ]
!
         CALL DLASWP( N2, A( 1, N1+1 ), LDA, 1, N1, IPIV, 1 )
!
!        Solve A12
!
         CALL DTRSM( 'L', 'L', 'N', 'U', N1, N2, ONE, A, LDA,&
                     A( 1, N1+1 ), LDA )
!
!        Update A22
!
         CALL DGEMM( 'N', 'N', M-N1, N2, N1, -ONE, A( N1+1, 1 ), LDA,&
                     A( 1, N1+1 ), LDA, ONE, A( N1+1, N1+1 ), LDA )
!
!        Factor A22
!
         CALL DGETRF2( M-N1, N2, A( N1+1, N1+1 ), LDA, IPIV( N1+1 ),&
                       IINFO )
```

```
!
!        Adjust INFO and the pivot indices
!
         IF ( INFO.EQ.0 .AND. IINFO.GT.0 )&
            INFO = IINFO + N1
         DO 20 I = N1+1, MIN( M, N )
            IPIV( I ) = IPIV( I ) + N1
   20    CONTINUE
!
!        Apply interchanges to A21
!
         CALL DLASWP( N1, A( 1, 1 ), LDA, N1+1, MIN( M, N), IPIV, 1 )
!
      END IF
      RETURN
!
!     End of DGETRF2
!
      END SUBROUTINE DGETRF2

!  =====================================================================
      INTEGER FUNCTION ILAENV( ISPEC, NAME, OPTS, N1, N2, N3, N4 )
!
!  -- LAPACK auxiliary routine (version 3.8.0) --
!  -- LAPACK is a software package provided by Univ. of Tennessee,    --
!  -- Univ. of California Berkeley, Univ. of Colorado Denver and NAG Ltd..--
!     November 2017
!
!     .. Scalar Arguments ..
      CHARACTER*( * )    NAME, OPTS
      INTEGER            ISPEC, N1, N2, N3, N4
!     ..
!
!  =====================================================================
!
!     .. Local Scalars ..
      INTEGER            I, IC, IZ, NB, NBMIN, NX
      LOGICAL            CNAME, SNAME, TWOSTAGE
      CHARACTER          C1*1, C2*2, C4*2, C3*3, SUBNAM*16
!     ..
!     .. Intrinsic Functions ..
      INTRINSIC          CHAR, ICHAR, INT, MIN, REAL
!     ..
!     .. External Functions ..
!      INTEGER            IEEECK, IPARMQ, IPARAM2STAGE
!      EXTERNAL           IEEECK, IPARMQ, IPARAM2STAGE
!     ..
!     .. Executable Statements ..
!
      GO TO ( 10, 10, 10, 80, 90, 100, 110, 120,&
              130, 140, 150, 160, 160, 160, 160, 160)ISPEC
!
!     Invalid value for ISPEC
!
      ILAENV = -1
      RETURN
!
   10 CONTINUE
!
!     Convert NAME to upper case if the first character is lower case.
!
```

```
      ILAENV = 1
      SUBNAM = NAME
      IC = ICHAR( SUBNAM( 1: 1 ) )
      IZ = ICHAR( 'Z' )
      IF( IZ.EQ.90 .OR. IZ.EQ.122 ) THEN
!
!        ASCII character set
!
         IF( IC.GE.97 .AND. IC.LE.122 ) THEN
            SUBNAM( 1: 1 ) = CHAR( IC-32 )
            DO 20 I = 2, 6
               IC = ICHAR( SUBNAM( I: I ) )
               IF( IC.GE.97 .AND. IC.LE.122 )&
                  SUBNAM( I: I ) = CHAR( IC-32 )
   20       CONTINUE
         END IF
!
      ELSE IF( IZ.EQ.233 .OR. IZ.EQ.169 ) THEN
!
!        EBCDIC character set
!
         IF( ( IC.GE.129 .AND. IC.LE.137 ) .OR.&
             ( IC.GE.145 .AND. IC.LE.153 ) .OR.&
             ( IC.GE.162 .AND. IC.LE.169 ) ) THEN
            SUBNAM( 1: 1 ) = CHAR( IC+64 )
            DO 30 I = 2, 6
               IC = ICHAR( SUBNAM( I: I ) )
               IF( ( IC.GE.129 .AND. IC.LE.137 ) .OR.&
                   ( IC.GE.145 .AND. IC.LE.153 ) .OR.&
                   ( IC.GE.162 .AND. IC.LE.169 ) )SUBNAM( I:&
                   I ) = CHAR( IC+64 )
   30       CONTINUE
         END IF
!
      ELSE IF( IZ.EQ.218 .OR. IZ.EQ.250 ) THEN
!
!        Prime machines:  ASCII+128
!
         IF( IC.GE.225 .AND. IC.LE.250 ) THEN
            SUBNAM( 1: 1 ) = CHAR( IC-32 )
            DO 40 I = 2, 6
               IC = ICHAR( SUBNAM( I: I ) )
               IF( IC.GE.225 .AND. IC.LE.250 )&
                  SUBNAM( I: I ) = CHAR( IC-32 )
   40       CONTINUE
         END IF
      END IF
!
      C1 = SUBNAM( 1: 1 )
      SNAME = C1.EQ.'S' .OR. C1.EQ.'D'
      CNAME = C1.EQ.'C' .OR. C1.EQ.'Z'
      IF( .NOT.( CNAME .OR. SNAME ) )&
         RETURN
      C2 = SUBNAM( 2: 3 )
      C3 = SUBNAM( 4: 6 )
      C4 = C3( 2: 3 )
      TWOSTAGE = LEN( SUBNAM ).GE.11&
                 .AND. SUBNAM( 11: 11 ).EQ.'2'
!
      GO TO ( 50, 60, 70 )ISPEC
!
```

```
   50 CONTINUE
!
!     ISPEC = 1:  block size
!
!     In these examples, separate code is provided for setting NB for
!     real and complex.  We assume that NB will take the same value in
!     single or double precision.
!
      NB = 1
!
      IF( C2.EQ.'GE' ) THEN
         IF( C3.EQ.'TRF' ) THEN
            IF( SNAME ) THEN
               NB = 64
            ELSE
               NB = 64
            END IF
         ELSE IF( C3.EQ.'QRF' .OR. C3.EQ.'RQF' .OR. C3.EQ.'LQF' .OR.&
                  C3.EQ.'QLF' ) THEN
            IF( SNAME ) THEN
               NB = 32
            ELSE
               NB = 32
            END IF
         ELSE IF( C3.EQ.'QR ') THEN
            IF( N3 .EQ. 1) THEN
               IF( SNAME ) THEN
!     M*N
                  IF ((N1*N2.LE.131072).OR.(N1.LE.8192)) THEN
                     NB = N1
                  ELSE
                     NB = 32768/N2
                  END IF
               ELSE
                  IF ((N1*N2.LE.131072).OR.(N1.LE.8192)) THEN
                     NB = N1
                  ELSE
                     NB = 32768/N2
                  END IF
               END IF
            ELSE
               IF( SNAME ) THEN
                  NB = 1
               ELSE
                  NB = 1
               END IF
            END IF
         ELSE IF( C3.EQ.'LQ ') THEN
            IF( N3 .EQ. 2) THEN
               IF( SNAME ) THEN
!     M*N
                  IF ((N1*N2.LE.131072).OR.(N1.LE.8192)) THEN
                     NB = N1
                  ELSE
                     NB = 32768/N2
                  END IF
               ELSE
                  IF ((N1*N2.LE.131072).OR.(N1.LE.8192)) THEN
                     NB = N1
                  ELSE
                     NB = 32768/N2
```

```
                  END IF
               END IF
            ELSE
               IF( SNAME ) THEN
                  NB = 1
               ELSE
                  NB = 1
               END IF
            END IF
         ELSE IF( C3.EQ.'HRD' ) THEN
            IF( SNAME ) THEN
               NB = 32
            ELSE
               NB = 32
            END IF
         ELSE IF( C3.EQ.'BRD' ) THEN
            IF( SNAME ) THEN
               NB = 32
            ELSE
               NB = 32
            END IF
         ELSE IF( C3.EQ.'TRI' ) THEN
            IF( SNAME ) THEN
               NB = 64
            ELSE
               NB = 64
            END IF
         END IF
      ELSE IF( C2.EQ.'PO' ) THEN
         IF( C3.EQ.'TRF' ) THEN
            IF( SNAME ) THEN
               NB = 64
            ELSE
               NB = 64
            END IF
         END IF
      ELSE IF( C2.EQ.'SY' ) THEN
         IF( C3.EQ.'TRF' ) THEN
            IF( SNAME ) THEN
               IF( TWOSTAGE ) THEN
                  NB = 192
               ELSE
                  NB = 64
               END IF
            ELSE
               IF( TWOSTAGE ) THEN
                  NB = 192
               ELSE
                  NB = 64
               END IF
            END IF
         ELSE IF( SNAME .AND. C3.EQ.'TRD' ) THEN
            NB = 32
         ELSE IF( SNAME .AND. C3.EQ.'GST' ) THEN
            NB = 64
         END IF
      ELSE IF( CNAME .AND. C2.EQ.'HE' ) THEN
         IF( C3.EQ.'TRF' ) THEN
            IF( TWOSTAGE ) THEN
               NB = 192
            ELSE
```

```
               NB = 64
            END IF
         ELSE IF( C3.EQ.'TRD' ) THEN
            NB = 32
         ELSE IF( C3.EQ.'GST' ) THEN
            NB = 64
         END IF
      ELSE IF( SNAME .AND. C2.EQ.'OR' ) THEN
         IF( C3( 1: 1 ).EQ.'G' ) THEN
            IF( C4.EQ.'QR' .OR. C4.EQ.'RQ' .OR. C4.EQ.'LQ' .OR. C4.EQ.&
                'QL' .OR. C4.EQ.'HR' .OR. C4.EQ.'TR' .OR. C4.EQ.'BR' )&
                 THEN
               NB = 32
            END IF
         ELSE IF( C3( 1: 1 ).EQ.'M' ) THEN
            IF( C4.EQ.'QR' .OR. C4.EQ.'RQ' .OR. C4.EQ.'LQ' .OR. C4.EQ.&
                'QL' .OR. C4.EQ.'HR' .OR. C4.EQ.'TR' .OR. C4.EQ.'BR' )&
                 THEN
               NB = 32
            END IF
         END IF
      ELSE IF( CNAME .AND. C2.EQ.'UN' ) THEN
         IF( C3( 1: 1 ).EQ.'G' ) THEN
            IF( C4.EQ.'QR' .OR. C4.EQ.'RQ' .OR. C4.EQ.'LQ' .OR. C4.EQ.&
                'QL' .OR. C4.EQ.'HR' .OR. C4.EQ.'TR' .OR. C4.EQ.'BR' )&
                 THEN
               NB = 32
            END IF
         ELSE IF( C3( 1: 1 ).EQ.'M' ) THEN
            IF( C4.EQ.'QR' .OR. C4.EQ.'RQ' .OR. C4.EQ.'LQ' .OR. C4.EQ.&
                'QL' .OR. C4.EQ.'HR' .OR. C4.EQ.'TR' .OR. C4.EQ.'BR' )&
                 THEN
               NB = 32
            END IF
         END IF
      ELSE IF( C2.EQ.'GB' ) THEN
         IF( C3.EQ.'TRF' ) THEN
            IF( SNAME ) THEN
               IF( N4.LE.64 ) THEN
                  NB = 1
               ELSE
                  NB = 32
               END IF
            ELSE
               IF( N4.LE.64 ) THEN
                  NB = 1
               ELSE
                  NB = 32
               END IF
            END IF
         END IF
      ELSE IF( C2.EQ.'PB' ) THEN
         IF( C3.EQ.'TRF' ) THEN
            IF( SNAME ) THEN
               IF( N2.LE.64 ) THEN
                  NB = 1
               ELSE
                  NB = 32
               END IF
            ELSE
               IF( N2.LE.64 ) THEN
```

```
                  NB = 1
               ELSE
                  NB = 32
               END IF
            END IF
         END IF
      ELSE IF( C2.EQ.'TR' ) THEN
         IF( C3.EQ.'TRI' ) THEN
            IF( SNAME ) THEN
               NB = 64
            ELSE
               NB = 64
            END IF
         ELSE IF ( C3.EQ.'EVC' ) THEN
            IF( SNAME ) THEN
               NB = 64
            ELSE
               NB = 64
            END IF
         END IF
      ELSE IF( C2.EQ.'LA' ) THEN
         IF( C3.EQ.'UUM' ) THEN
            IF( SNAME ) THEN
               NB = 64
            ELSE
               NB = 64
            END IF
         END IF
      ELSE IF( SNAME .AND. C2.EQ.'ST' ) THEN
         IF( C3.EQ.'EBZ' ) THEN
            NB = 1
         END IF
      ELSE IF( C2.EQ.'GG' ) THEN
         NB = 32
         IF( C3.EQ.'HD3' ) THEN
            IF( SNAME ) THEN
               NB = 32
            ELSE
               NB = 32
            END IF
         END IF
      END IF
      ILAENV = NB
      RETURN
!
   60 CONTINUE
!
!     ISPEC = 2:  minimum block size
!
      NBMIN = 2
      IF( C2.EQ.'GE' ) THEN
         IF( C3.EQ.'QRF' .OR. C3.EQ.'RQF' .OR. C3.EQ.'LQF' .OR. C3.EQ.&
             'QLF' ) THEN
            IF( SNAME ) THEN
               NBMIN = 2
            ELSE
               NBMIN = 2
            END IF
         ELSE IF( C3.EQ.'HRD' ) THEN
            IF( SNAME ) THEN
               NBMIN = 2
```

```
            ELSE
               NBMIN = 2
            END IF
         ELSE IF( C3.EQ.'BRD' ) THEN
            IF( SNAME ) THEN
               NBMIN = 2
            ELSE
               NBMIN = 2
            END IF
         ELSE IF( C3.EQ.'TRI' ) THEN
            IF( SNAME ) THEN
               NBMIN = 2
            ELSE
               NBMIN = 2
            END IF
         END IF
      ELSE IF( C2.EQ.'SY' ) THEN
         IF( C3.EQ.'TRF' ) THEN
            IF( SNAME ) THEN
               NBMIN = 8
            ELSE
               NBMIN = 8
            END IF
         ELSE IF( SNAME .AND. C3.EQ.'TRD' ) THEN
            NBMIN = 2
         END IF
      ELSE IF( CNAME .AND. C2.EQ.'HE' ) THEN
         IF( C3.EQ.'TRD' ) THEN
            NBMIN = 2
         END IF
      ELSE IF( SNAME .AND. C2.EQ.'OR' ) THEN
         IF( C3( 1: 1 ).EQ.'G' ) THEN
            IF( C4.EQ.'QR' .OR. C4.EQ.'RQ' .OR. C4.EQ.'LQ' .OR. C4.EQ.&
                'QL' .OR. C4.EQ.'HR' .OR. C4.EQ.'TR' .OR. C4.EQ.'BR' )&
                 THEN
               NBMIN = 2
            END IF
         ELSE IF( C3( 1: 1 ).EQ.'M' ) THEN
            IF( C4.EQ.'QR' .OR. C4.EQ.'RQ' .OR. C4.EQ.'LQ' .OR. C4.EQ.&
                'QL' .OR. C4.EQ.'HR' .OR. C4.EQ.'TR' .OR. C4.EQ.'BR' )&
                 THEN
               NBMIN = 2
            END IF
         END IF
      ELSE IF( CNAME .AND. C2.EQ.'UN' ) THEN
         IF( C3( 1: 1 ).EQ.'G' ) THEN
            IF( C4.EQ.'QR' .OR. C4.EQ.'RQ' .OR. C4.EQ.'LQ' .OR. C4.EQ.&
                'QL' .OR. C4.EQ.'HR' .OR. C4.EQ.'TR' .OR. C4.EQ.'BR' )&
                 THEN
               NBMIN = 2
            END IF
         ELSE IF( C3( 1: 1 ).EQ.'M' ) THEN
            IF( C4.EQ.'QR' .OR. C4.EQ.'RQ' .OR. C4.EQ.'LQ' .OR. C4.EQ.&
                'QL' .OR. C4.EQ.'HR' .OR. C4.EQ.'TR' .OR. C4.EQ.'BR' )&
                 THEN
               NBMIN = 2
            END IF
         END IF
      ELSE IF( C2.EQ.'GG' ) THEN
         NBMIN = 2
         IF( C3.EQ.'HD3' ) THEN
```

```
            NBMIN = 2
         END IF
      END IF
      ILAENV = NBMIN
      RETURN
!
   70 CONTINUE
!
!     ISPEC = 3:  crossover point
!
      NX = 0
      IF( C2.EQ.'GE' ) THEN
         IF( C3.EQ.'QRF' .OR. C3.EQ.'RQF' .OR. C3.EQ.'LQF' .OR. C3.EQ.&
             'QLF' ) THEN
            IF( SNAME ) THEN
               NX = 128
            ELSE
               NX = 128
            END IF
         ELSE IF( C3.EQ.'HRD' ) THEN
            IF( SNAME ) THEN
               NX = 128
            ELSE
               NX = 128
            END IF
         ELSE IF( C3.EQ.'BRD' ) THEN
            IF( SNAME ) THEN
               NX = 128
            ELSE
               NX = 128
            END IF
         END IF
      ELSE IF( C2.EQ.'SY' ) THEN
         IF( SNAME .AND. C3.EQ.'TRD' ) THEN
            NX = 32
         END IF
      ELSE IF( CNAME .AND. C2.EQ.'HE' ) THEN
         IF( C3.EQ.'TRD' ) THEN
            NX = 32
         END IF
      ELSE IF( SNAME .AND. C2.EQ.'OR' ) THEN
         IF( C3( 1: 1 ).EQ.'G' ) THEN
            IF( C4.EQ.'QR' .OR. C4.EQ.'RQ' .OR. C4.EQ.'LQ' .OR. C4.EQ.&
                'QL' .OR. C4.EQ.'HR' .OR. C4.EQ.'TR' .OR. C4.EQ.'BR' )&
                 THEN
               NX = 128
            END IF
         END IF
      ELSE IF( CNAME .AND. C2.EQ.'UN' ) THEN
         IF( C3( 1: 1 ).EQ.'G' ) THEN
            IF( C4.EQ.'QR' .OR. C4.EQ.'RQ' .OR. C4.EQ.'LQ' .OR. C4.EQ.&
                'QL' .OR. C4.EQ.'HR' .OR. C4.EQ.'TR' .OR. C4.EQ.'BR' )&
                 THEN
               NX = 128
            END IF
         END IF
      ELSE IF( C2.EQ.'GG' ) THEN
         NX = 128
         IF( C3.EQ.'HD3' ) THEN
            NX = 128
         END IF
```

```
      END IF
      ILAENV = NX
      RETURN
!
   80 CONTINUE
!
!     ISPEC = 4:  number of shifts (used by xHSEQR)
!
      ILAENV = 6
      RETURN
!
   90 CONTINUE
!
!     ISPEC = 5:  minimum column dimension (not used)
!
      ILAENV = 2
      RETURN
!
  100 CONTINUE
!
!     ISPEC = 6:  crossover point for SVD (used by xGELSS and xGESVD)
!
      ILAENV = INT( REAL( MIN( N1, N2 ) )*1.6E0 )
      RETURN
!
  110 CONTINUE
!
!     ISPEC = 7:  number of processors (not used)
!
      ILAENV = 1
      RETURN
!
  120 CONTINUE
!
!     ISPEC = 8:  crossover point for multishift (used by xHSEQR)
!
      ILAENV = 50
      RETURN
!
  130 CONTINUE
!
!     ISPEC = 9:  maximum size of the subproblems at the bottom of the
!                 computation tree in the divide-and-conquer algorithm
!                 (used by xGELSD and xGESDD)
!
      ILAENV = 25
      RETURN
!
  140 CONTINUE
!
!     ISPEC = 10: ieee NaN arithmetic can be trusted not to trap
!
!     ILAENV = 0
      ILAENV = 1
      IF( ILAENV.EQ.1 ) THEN
         ILAENV = IEEECK( 1, 0.0, 1.0 )
      END IF
      RETURN
!
  150 CONTINUE
!
```

```
!     ISPEC = 11: infinity arithmetic can be trusted not to trap
!
!     ILAENV = 0
      ILAENV = 1
      IF( ILAENV.EQ.1 ) THEN
         ILAENV = IEEECK( 0, 0.0, 1.0 )
      END IF
      RETURN
!
  160 CONTINUE
!
!     12 <= ISPEC <= 16: xHSEQR or related subroutines.
!
      ILAENV = IPARMQ( ISPEC, NAME, OPTS, N1, N2, N3, N4 )
      RETURN
!
!     End of ILAENV
!
      END FUNCTION ILAENV

!
!  ====================================================================
      INTEGER          FUNCTION IEEECK( ISPEC, ZERO, ONE )
!
!  -- LAPACK auxiliary routine (version 3.7.0) --
!  -- LAPACK is a software package provided by Univ. of Tennessee,    --
!  -- Univ. of California Berkeley, Univ. of Colorado Denver and NAG Ltd..--
!     December 2016
!
!     .. Scalar Arguments ..
      INTEGER            ISPEC
      REAL               ONE, ZERO
!     ..
!
!  ====================================================================
!
!     .. Local Scalars ..
      REAL               NAN1, NAN2, NAN3, NAN4, NAN5, NAN6, NEGINF,&
     &                   NEGZRO, NEWZRO, POSINF
!     ..
!     .. Executable Statements ..
      IEEECK = 1
!
      POSINF = ONE / ZERO
      IF( POSINF.LE.ONE ) THEN
         IEEECK = 0
         RETURN
      END IF
!
      NEGINF = -ONE / ZERO
      IF( NEGINF.GE.ZERO ) THEN
         IEEECK = 0
         RETURN
      END IF
!
      NEGZRO = ONE / ( NEGINF+ONE )
      IF( NEGZRO.NE.ZERO ) THEN
         IEEECK = 0
         RETURN
      END IF
!
```

```
      NEGINF = ONE / NEGZRO
      IF( NEGINF.GE.ZERO ) THEN
         IEEECK = 0
         RETURN
      END IF
!
      NEWZRO = NEGZRO + ZERO
      IF( NEWZRO.NE.ZERO ) THEN
         IEEECK = 0
         RETURN
      END IF
!
      POSINF = ONE / NEWZRO
      IF( POSINF.LE.ONE ) THEN
         IEEECK = 0
         RETURN
      END IF
!
      NEGINF = NEGINF*POSINF
      IF( NEGINF.GE.ZERO ) THEN
         IEEECK = 0
         RETURN
      END IF
!
      POSINF = POSINF*POSINF
      IF( POSINF.LE.ONE ) THEN
         IEEECK = 0
         RETURN
      END IF
!
!
!
!     Return if we were only asked to check infinity arithmetic
!
      IF( ISPEC.EQ.0 )&
     &   RETURN
!
      NAN1 = POSINF + NEGINF
!
      NAN2 = POSINF / NEGINF
!
      NAN3 = POSINF / POSINF
!
      NAN4 = POSINF*ZERO
!
      NAN5 = NEGINF*NEGZRO
!
      NAN6 = NAN5*ZERO
!
      IF( NAN1.EQ.NAN1 ) THEN
         IEEECK = 0
         RETURN
      END IF
!
      IF( NAN2.EQ.NAN2 ) THEN
         IEEECK = 0
         RETURN
      END IF
!
      IF( NAN3.EQ.NAN3 ) THEN
         IEEECK = 0
         RETURN
```

```
      END IF
!
      IF( NAN4.EQ.NAN4 ) THEN
         IEEECK = 0
         RETURN
      END IF
!
      IF( NAN5.EQ.NAN5 ) THEN
         IEEECK = 0
         RETURN
      END IF
!
      IF( NAN6.EQ.NAN6 ) THEN
         IEEECK = 0
         RETURN
      END IF
!
      RETURN
      END FUNCTION IEEECK

!  =====================================================================
      INTEGER FUNCTION IPARMQ( ISPEC, NAME, OPTS, N, ILO, IHI, LWORK )
!
!  -- LAPACK auxiliary routine (version 3.7.1) --
!  -- LAPACK is a software package provided by Univ. of Tennessee,    --
!  -- Univ. of California Berkeley, Univ. of Colorado Denver and NAG Ltd..--
!     June 2017
!
!     .. Scalar Arguments ..
      INTEGER            IHI, ILO, ISPEC, LWORK, N
      CHARACTER          NAME*( * ), OPTS*( * )
!
!  ================================================================
!     .. Parameters ..
      INTEGER            INMIN, INWIN, INIBL, ISHFTS, IACC22
      PARAMETER          ( INMIN = 12, INWIN = 13, INIBL = 14,&
     &                   ISHFTS = 15, IACC22 = 16 )
      INTEGER            NMIN, K22MIN, KACMIN, NIBBLE, KNWSWP
      PARAMETER          ( NMIN = 75, K22MIN = 14, KACMIN = 14,&
     &                   NIBBLE = 14, KNWSWP = 500 )
      REAL               TWO
      PARAMETER          ( TWO = 2.0 )
!     ..
!     .. Local Scalars ..
      INTEGER            NH, NS
      INTEGER            I, IC, IZ
      CHARACTER          SUBNAM*6
!     ..
!     .. Intrinsic Functions ..
      INTRINSIC          LOG, MAX, MOD, NINT, REAL
!     ..
!     .. Executable Statements ..
      IF( ( ISPEC.EQ.ISHFTS ) .OR. ( ISPEC.EQ.INWIN ) .OR.&
     &    ( ISPEC.EQ.IACC22 ) ) THEN
!
!        ==== Set the number simultaneous shifts ====
!
         NH = IHI - ILO + 1
         NS = 2
         IF( NH.GE.30 )&
     &      NS = 4
```

```
         IF( NH.GE.60 )&
     &      NS = 10
         IF( NH.GE.150 )&
     &      NS = MAX( 10, NH / NINT( LOG( REAL( NH ) ) / LOG( TWO ) ) )
         IF( NH.GE.590 )&
     &      NS = 64
         IF( NH.GE.3000 )&
     &      NS = 128
         IF( NH.GE.6000 )&
     &      NS = 256
         NS = MAX( 2, NS-MOD( NS, 2 ) )
      END IF
!
      IF( ISPEC.EQ.INMIN ) THEN
!
!
!        ===== Matrices of order smaller than NMIN get sent
!        .     to xLAHQR, the classic double shift algorithm.
!        .     This must be at least 11. ====
!
         IPARMQ = NMIN
!
      ELSE IF( ISPEC.EQ.INIBL ) THEN
!
!        ==== INIBL: skip a multi-shift qr iteration and
!        .    whenever aggressive early deflation finds
!        .    at least (NIBBLE*(window size)/100) deflations. ====
!
         IPARMQ = NIBBLE
!
      ELSE IF( ISPEC.EQ.ISHFTS ) THEN
!
!        ==== NSHFTS: The number of simultaneous shifts =====
!
         IPARMQ = NS
!
      ELSE IF( ISPEC.EQ.INWIN ) THEN
!
!        ==== NW: deflation window size.  ====
!
         IF( NH.LE.KNWSWP ) THEN
            IPARMQ = NS
         ELSE
            IPARMQ = 3*NS / 2
         END IF
!
      ELSE IF( ISPEC.EQ.IACC22 ) THEN
!
!        ==== IACC22: Whether to accumulate reflections
!        .     before updating the far-from-diagonal elements
!        .     and whether to use 2-by-2 block structure while
!        .     doing it.  A small amount of work could be saved
!        .     by making this choice dependent also upon the
!        .     NH=IHI-ILO+1.
!
!
!        Convert NAME to upper case if the first character is lower case.
!
         IPARMQ = 0
         SUBNAM = NAME
         IC = ICHAR( SUBNAM( 1: 1 ) )
```

```fortran
         IZ = ICHAR( 'Z' )
         IF( IZ.EQ.90 .OR. IZ.EQ.122 ) THEN
!
!           ASCII character set
!
            IF( IC.GE.97 .AND. IC.LE.122 ) THEN
               SUBNAM( 1: 1 ) = CHAR( IC-32 )
               DO I = 2, 6
                  IC = ICHAR( SUBNAM( I: I ) )
                  IF( IC.GE.97 .AND. IC.LE.122 )&
     &               SUBNAM( I: I ) = CHAR( IC-32 )
               END DO
            END IF
!
         ELSE IF( IZ.EQ.233 .OR. IZ.EQ.169 ) THEN
!
!           EBCDIC character set
!
            IF( ( IC.GE.129 .AND. IC.LE.137 ) .OR.&
     &          ( IC.GE.145 .AND. IC.LE.153 ) .OR.&
     &          ( IC.GE.162 .AND. IC.LE.169 ) ) THEN
               SUBNAM( 1: 1 ) = CHAR( IC+64 )
               DO I = 2, 6
                  IC = ICHAR( SUBNAM( I: I ) )
                  IF( ( IC.GE.129 .AND. IC.LE.137 ) .OR.&
     &                ( IC.GE.145 .AND. IC.LE.153 ) .OR.&
     &                ( IC.GE.162 .AND. IC.LE.169 ) )SUBNAM( I:&
     &                I ) = CHAR( IC+64 )
               END DO
            END IF
!
         ELSE IF( IZ.EQ.218 .OR. IZ.EQ.250 ) THEN
!
!           Prime machines:  ASCII+128
!
            IF( IC.GE.225 .AND. IC.LE.250 ) THEN
               SUBNAM( 1: 1 ) = CHAR( IC-32 )
               DO I = 2, 6
                  IC = ICHAR( SUBNAM( I: I ) )
                  IF( IC.GE.225 .AND. IC.LE.250 )&
     &               SUBNAM( I: I ) = CHAR( IC-32 )
               END DO
            END IF
         END IF
!
         IF( SUBNAM( 2:6 ).EQ.'GGHRD' .OR.&
     &       SUBNAM( 2:6 ).EQ.'GGHD3' ) THEN
            IPARMQ = 1
            IF( NH.GE.K22MIN )&
     &         IPARMQ = 2
         ELSE IF ( SUBNAM( 4:6 ).EQ.'EXC' ) THEN
            IF( NH.GE.KACMIN )&
     &         IPARMQ = 1
            IF( NH.GE.K22MIN )&
     &         IPARMQ = 2
         ELSE IF ( SUBNAM( 2:6 ).EQ.'HSEQR' .OR.&
     &             SUBNAM( 2:5 ).EQ.'LAQR' ) THEN
            IF( NS.GE.KACMIN )&
     &         IPARMQ = 1
            IF( NS.GE.K22MIN )&
     &         IPARMQ = 2
```

```
         END IF
!
      ELSE
!        ===== invalid value of ispec =====
         IPARMQ = -1
!
      END IF
!
!     ==== End of IPARMQ ====
!
      END FUNCTION IPARMQ

!
!  ====================================================================
      INTEGER FUNCTION IPARAM2STAGE( ISPEC, NAME, OPTS,&
     &                              NI, NBI, IBI, NXI )
#if defined(_OPENMP)
      use omp_lib
#endif
      IMPLICIT NONE
!
!  -- LAPACK auxiliary routine (version 3.8.0) --
!  -- LAPACK is a software package provided by Univ. of Tennessee,    --
!  -- Univ. of California Berkeley, Univ. of Colorado Denver and NAG Ltd..--
!     June 2016
!
!     .. Scalar Arguments ..
      CHARACTER*( * )    NAME, OPTS
      INTEGER            ISPEC, NI, NBI, IBI, NXI
!
!  ================================================================
!     ..
!     .. Local Scalars ..
      INTEGER            I, IC, IZ, KD, IB, LHOUS, LWORK, NTHREADS,&
     &                   FACTOPTNB, QROPTNB, LQOPTNB
      LOGICAL            RPREC, CPREC
      CHARACTER          PREC*1, ALGO*3, STAG*5, SUBNAM*12, VECT*1
!     ..
!     .. Intrinsic Functions ..
      INTRINSIC          CHAR, ICHAR, MAX
!     ..
!     .. External Functions ..
!      INTEGER            ILAENV
!      EXTERNAL           ILAENV
!     ..
!     .. Executable Statements ..
!
!     Invalid value for ISPEC
!
      IF( (ISPEC.LT.17).OR.(ISPEC.GT.21) ) THEN
          IPARAM2STAGE = -1
          RETURN
      ENDIF
!
!     Get the number of threads
!
      NTHREADS = 1
#if defined(_OPENMP)
!$OMP PARALLEL
      NTHREADS = OMP_GET_NUM_THREADS()
!$OMP END PARALLEL
```

```fortran
#endif
!      WRITE(*,*) 'IPARAM VOICI NTHREADS ISPEC ',NTHREADS, ISPEC
!
      IF( ISPEC .NE. 19 ) THEN
!
!        Convert NAME to upper case if the first character is lower case.
!
         IPARAM2STAGE = -1
         SUBNAM = NAME
         IC = ICHAR( SUBNAM( 1: 1 ) )
         IZ = ICHAR( 'Z' )
         IF( IZ.EQ.90 .OR. IZ.EQ.122 ) THEN
!
!           ASCII character set
!
            IF( IC.GE.97 .AND. IC.LE.122 ) THEN
               SUBNAM( 1: 1 ) = CHAR( IC-32 )
               DO 100 I = 2, 12
                  IC = ICHAR( SUBNAM( I: I ) )
                  IF( IC.GE.97 .AND. IC.LE.122 )&
     &               SUBNAM( I: I ) = CHAR( IC-32 )
  100          CONTINUE
            END IF
!
         ELSE IF( IZ.EQ.233 .OR. IZ.EQ.169 ) THEN
!
!           EBCDIC character set
!
            IF( ( IC.GE.129 .AND. IC.LE.137 ) .OR.&
     &          ( IC.GE.145 .AND. IC.LE.153 ) .OR.&
     &          ( IC.GE.162 .AND. IC.LE.169 ) ) THEN
               SUBNAM( 1: 1 ) = CHAR( IC+64 )
               DO 110 I = 2, 12
                  IC = ICHAR( SUBNAM( I: I ) )
                  IF( ( IC.GE.129 .AND. IC.LE.137 ) .OR.&
     &                ( IC.GE.145 .AND. IC.LE.153 ) .OR.&
     &                ( IC.GE.162 .AND. IC.LE.169 ) )SUBNAM( I:&
     &                I ) = CHAR( IC+64 )
  110          CONTINUE
            END IF
!
         ELSE IF( IZ.EQ.218 .OR. IZ.EQ.250 ) THEN
!
!           Prime machines:  ASCII+128
!
            IF( IC.GE.225 .AND. IC.LE.250 ) THEN
               SUBNAM( 1: 1 ) = CHAR( IC-32 )
               DO 120 I = 2, 12
                 IC = ICHAR( SUBNAM( I: I ) )
                 IF( IC.GE.225 .AND. IC.LE.250 )&
     &             SUBNAM( I: I ) = CHAR( IC-32 )
  120          CONTINUE
            END IF
         END IF
!
         PREC  = SUBNAM( 1: 1 )
         ALGO  = SUBNAM( 4: 6 )
         STAG  = SUBNAM( 8:12 )
         RPREC = PREC.EQ.'S' .OR. PREC.EQ.'D'
         CPREC = PREC.EQ.'C' .OR. PREC.EQ.'Z'
!
```

```
!        Invalid value for PRECISION
!
         IF( .NOT.( RPREC .OR. CPREC ) ) THEN
             IPARAM2STAGE = -1
             RETURN
         ENDIF
      ENDIF
!      WRITE(*,*),'RPREC,CPREC ',RPREC,CPREC,&
!     &           '   ALGO ',ALGO,'    STAGE ',STAG
!
!
      IF (( ISPEC .EQ. 17 ) .OR. ( ISPEC .EQ. 18 )) THEN
!
!     ISPEC = 17, 18:  block size KD, IB
!     Could be also dependent from N but for now it
!     depend only on sequential or parallel
!
         IF( NTHREADS.GT.4 ) THEN
            IF( CPREC ) THEN
               KD = 128
               IB = 32
            ELSE
               KD = 160
               IB = 40
            ENDIF
         ELSE IF( NTHREADS.GT.1 ) THEN
            IF( CPREC ) THEN
               KD = 64
               IB = 32
            ELSE
               KD = 64
               IB = 32
            ENDIF
         ELSE
            IF( CPREC ) THEN
               KD = 16
               IB = 16
            ELSE
               KD = 32
               IB = 16
            ENDIF
         ENDIF
         IF( ISPEC.EQ.17 ) IPARAM2STAGE = KD
         IF( ISPEC.EQ.18 ) IPARAM2STAGE = IB
!
      ELSE IF ( ISPEC .EQ. 19 ) THEN
!
!     ISPEC = 19:
!     LHOUS length of the Houselholder representation
!     matrix (V,T) of the second stage. should be >= 1.
!
!     Will add the VECT OPTION HERE next release
         VECT  = OPTS(1:1)
         IF( VECT.EQ.'N' ) THEN
            LHOUS = MAX( 1, 4*NI )
         ELSE
!           This is not correct, it need to call the ALGO and the stage2
            LHOUS = MAX( 1, 4*NI ) + IBI
         ENDIF
         IF( LHOUS.GE.0 ) THEN
            IPARAM2STAGE = LHOUS
```

```
         ELSE
            IPARAM2STAGE = -1
         ENDIF
!
      ELSE IF ( ISPEC .EQ. 20 ) THEN
!
!     ISPEC = 20: (21 for future use)
!     LWORK length of the workspace for
!     either or both stages for TRD and BRD. should be >= 1.
!     TRD:
!     TRD_stage 1: = LT + LW + LS1 + LS2
!                  = LDT*KD + N*KD + N*MAX(KD,FACTOPTNB) + LDS2*KD
!                    where LDT=LDS2=KD
!                  = N*KD + N*max(KD,FACTOPTNB) + 2*KD*KD
!     TRD_stage 2: = (2NB+1)*N + KD*NTHREADS
!     TRD_both   : = max(stage1,stage2) + AB ( AB=(KD+1)*N )
!                  = N*KD + N*max(KD+1,FACTOPTNB)
!                    + max(2*KD*KD, KD*NTHREADS)
!                    + (KD+1)*N
         LWORK        = -1
         SUBNAM(1:1)  = PREC
         SUBNAM(2:6)  = 'GEQRF'
         QROPTNB      = ILAENV( 1, SUBNAM, ' ', NI, NBI, -1, -1 )
         SUBNAM(2:6)  = 'GELQF'
         LQOPTNB      = ILAENV( 1, SUBNAM, ' ', NBI, NI, -1, -1 )
!        Could be QR or LQ for TRD and the max for BRD
         FACTOPTNB    = MAX(QROPTNB, LQOPTNB)
         IF( ALGO.EQ.'TRD' ) THEN
            IF( STAG.EQ.'2STAG' ) THEN
               LWORK = NI*NBI + NI*MAX(NBI+1,FACTOPTNB)&
     &              + MAX(2*NBI*NBI, NBI*NTHREADS)     &
     &              + (NBI+1)*NI
            ELSE IF( (STAG.EQ.'HE2HB').OR.(STAG.EQ.'SY2SB') ) THEN
               LWORK = NI*NBI + NI*MAX(NBI,FACTOPTNB) + 2*NBI*NBI
            ELSE IF( (STAG.EQ.'HB2ST').OR.(STAG.EQ.'SB2ST') ) THEN
               LWORK = (2*NBI+1)*NI + NBI*NTHREADS
            ENDIF
         ELSE IF( ALGO.EQ.'BRD' ) THEN
            IF( STAG.EQ.'2STAG' ) THEN
               LWORK = 2*NI*NBI + NI*MAX(NBI+1,FACTOPTNB) &
     &              + MAX(2*NBI*NBI, NBI*NTHREADS)        &
     &              + (NBI+1)*NI
            ELSE IF( STAG.EQ.'GE2GB' ) THEN
               LWORK = NI*NBI + NI*MAX(NBI,FACTOPTNB) + 2*NBI*NBI
            ELSE IF( STAG.EQ.'GB2BD' ) THEN
               LWORK = (3*NBI+1)*NI + NBI*NTHREADS
            ENDIF
         ENDIF
         LWORK = MAX ( 1, LWORK )

         IF( LWORK.GT.0 ) THEN
            IPARAM2STAGE = LWORK
         ELSE
            IPARAM2STAGE = -1
         ENDIF
!
      ELSE IF ( ISPEC .EQ. 21 ) THEN
!
!     ISPEC = 21 for future use
         IPARAM2STAGE = NXI
      ENDIF
```

```
!
!     ==== End of IPARAM2STAGE ====
!
      END FUNCTION IPARAM2STAGE

!
!  =====================================================================
      SUBROUTINE DSWAP(N,DX,INCX,DY,INCY)
!
!  -- Reference BLAS level1 routine (version 3.8.0) --
!  -- Reference BLAS is a software package provided by Univ. of Tennessee,    --
!  -- Univ. of California Berkeley, Univ. of Colorado Denver and NAG Ltd..--
!     November 2017
!
!     .. Scalar Arguments ..
      INTEGER INCX,INCY,N
!     ..
!     .. Array Arguments ..
      DOUBLE PRECISION DX(*),DY(*)
!     ..
!
!  =====================================================================
!
!     .. Local Scalars ..
      DOUBLE PRECISION DTEMP
      INTEGER I,IX,IY,M,MP1
!     ..
!     .. Intrinsic Functions ..
      INTRINSIC MOD
!     ..
      IF (N.LE.0) RETURN
      IF (INCX.EQ.1 .AND. INCY.EQ.1) THEN
!
!       code for both increments equal to 1
!
!
!       clean-up loop
!
         M = MOD(N,3)
         IF (M.NE.0) THEN
            DO I = 1,M
               DTEMP = DX(I)
               DX(I) = DY(I)
               DY(I) = DTEMP
            END DO
            IF (N.LT.3) RETURN
         END IF
         MP1 = M + 1
         DO I = MP1,N,3
            DTEMP = DX(I)
            DX(I) = DY(I)
            DY(I) = DTEMP
            DTEMP = DX(I+1)
            DX(I+1) = DY(I+1)
            DY(I+1) = DTEMP
            DTEMP = DX(I+2)
            DX(I+2) = DY(I+2)
            DY(I+2) = DTEMP
         END DO
      ELSE
!
```

```
!       code for unequal increments or equal increments not equal
!         to 1
!
         IX = 1
         IY = 1
         IF (INCX.LT.0) IX = (-N+1)*INCX + 1
         IF (INCY.LT.0) IY = (-N+1)*INCY + 1
         DO I = 1,N
            DTEMP = DX(IX)
            DX(IX) = DY(IY)
            DY(IY) = DTEMP
            IX = IX + INCX
            IY = IY + INCY
         END DO
      END IF
      RETURN
      END SUBROUTINE DSWAP

!
!  =====================================================================
      SUBROUTINE DGEMV(TRANS,M,N,ALPHA,A,LDA,X,INCX,BETA,Y,INCY)
!
!  -- Reference BLAS level2 routine (version 3.7.0) --
!  -- Reference BLAS is a software package provided by Univ. of Tennessee,    --
!  -- Univ. of California Berkeley, Univ. of Colorado Denver and NAG Ltd..--
!     December 2016
!
!     .. Scalar Arguments ..
      DOUBLE PRECISION ALPHA,BETA
      INTEGER INCX,INCY,LDA,M,N
      CHARACTER TRANS
!     ..
!     .. Array Arguments ..
      DOUBLE PRECISION A(LDA,*),X(*),Y(*)
!     ..
!
!  =====================================================================
!
!     .. Parameters ..
      DOUBLE PRECISION ONE,ZERO
      PARAMETER (ONE=1.0D+0,ZERO=0.0D+0)
!     ..
!     .. Local Scalars ..
      DOUBLE PRECISION TEMP
      INTEGER I,INFO,IX,IY,J,JX,JY,KX,KY,LENX,LENY
!     ..
!     .. External Functions ..
!      LOGICAL LSAME
!      EXTERNAL LSAME
!     ..
!     .. External Subroutines ..
!      EXTERNAL XERBLA
!     ..
!     .. Intrinsic Functions ..
      INTRINSIC MAX
!     ..
!
!     Test the input parameters.
!
      INFO = 0
      IF (.NOT.LSAME(TRANS,'N') .AND. .NOT.LSAME(TRANS,'T') .AND.&
```

```
         .NOT.LSAME(TRANS,'C')) THEN
         INFO = 1
     ELSE IF (M.LT.0) THEN
         INFO = 2
     ELSE IF (N.LT.0) THEN
         INFO = 3
     ELSE IF (LDA.LT.MAX(1,M)) THEN
         INFO = 6
     ELSE IF (INCX.EQ.0) THEN
         INFO = 8
     ELSE IF (INCY.EQ.0) THEN
         INFO = 11
     END IF
     IF (INFO.NE.0) THEN
         CALL XERBLA('DGEMV ',INFO)
         RETURN
     END IF
!
!    Quick return if possible.
!
     IF ((M.EQ.0) .OR. (N.EQ.0) .OR.&
         ((ALPHA.EQ.ZERO).AND. (BETA.EQ.ONE))) RETURN
!
!    Set  LENX  and  LENY, the lengths of the vectors x and y, and set
!    up the start points in  X  and  Y.
!
     IF (LSAME(TRANS,'N')) THEN
         LENX = N
         LENY = M
     ELSE
         LENX = M
         LENY = N
     END IF
     IF (INCX.GT.0) THEN
         KX = 1
     ELSE
         KX = 1 - (LENX-1)*INCX
     END IF
     IF (INCY.GT.0) THEN
         KY = 1
     ELSE
         KY = 1 - (LENY-1)*INCY
     END IF
!
!    Start the operations. In this version the elements of A are
!    accessed sequentially with one pass through A.
!
!    First form  y := beta*y.
!
     IF (BETA.NE.ONE) THEN
         IF (INCY.EQ.1) THEN
             IF (BETA.EQ.ZERO) THEN
                 DO 10 I = 1,LENY
                     Y(I) = ZERO
  10             CONTINUE
             ELSE
                 DO 20 I = 1,LENY
                     Y(I) = BETA*Y(I)
  20             CONTINUE
             END IF
         ELSE
```

```fortran
              IY = KY
              IF (BETA.EQ.ZERO) THEN
                  DO 30 I = 1,LENY
                      Y(IY) = ZERO
                      IY = IY + INCY
   30             CONTINUE
              ELSE
                  DO 40 I = 1,LENY
                      Y(IY) = BETA*Y(IY)
                      IY = IY + INCY
   40             CONTINUE
              END IF
          END IF
      END IF
      IF (ALPHA.EQ.ZERO) RETURN
      IF (LSAME(TRANS,'N')) THEN
!
!        Form  y := alpha*A*x + y.
!
          JX = KX
          IF (INCY.EQ.1) THEN
              DO 60 J = 1,N
                  TEMP = ALPHA*X(JX)
                  DO 50 I = 1,M
                      Y(I) = Y(I) + TEMP*A(I,J)
   50             CONTINUE
                  JX = JX + INCX
   60         CONTINUE
          ELSE
              DO 80 J = 1,N
                  TEMP = ALPHA*X(JX)
                  IY = KY
                  DO 70 I = 1,M
                      Y(IY) = Y(IY) + TEMP*A(I,J)
                      IY = IY + INCY
   70             CONTINUE
                  JX = JX + INCX
   80         CONTINUE
          END IF
      ELSE
!
!        Form  y := alpha*A**T*x + y.
!
          JY = KY
          IF (INCX.EQ.1) THEN
              DO 100 J = 1,N
                  TEMP = ZERO
                  DO 90 I = 1,M
                      TEMP = TEMP + A(I,J)*X(I)
   90             CONTINUE
                  Y(JY) = Y(JY) + ALPHA*TEMP
                  JY = JY + INCY
  100         CONTINUE
          ELSE
              DO 120 J = 1,N
                  TEMP = ZERO
                  IX = KX
                  DO 110 I = 1,M
                      TEMP = TEMP + A(I,J)*X(IX)
                      IX = IX + INCX
  110             CONTINUE
```

```fortran
                  Y(JY) = Y(JY) + ALPHA*TEMP
                  JY = JY + INCY
  120         CONTINUE
          END IF
      END IF
!
      RETURN
!
!     End of DGEMV .
!
      END SUBROUTINE DGEMV
!
!  =====================================================================
      SUBROUTINE DTRTRI( UPLO, DIAG, N, A, LDA, INFO )
!
!  -- LAPACK computational routine (version 3.7.0) --
!  -- LAPACK is a software package provided by Univ. of Tennessee,    --
!  -- Univ. of California Berkeley, Univ. of Colorado Denver and NAG Ltd..--
!     December 2016
!
!     .. Scalar Arguments ..
      CHARACTER          DIAG, UPLO
      INTEGER            INFO, LDA, N
!     ..
!     .. Array Arguments ..
      DOUBLE PRECISION   A( LDA, * )
!     ..
!
!  =====================================================================
!
!     .. Parameters ..
      DOUBLE PRECISION   ONE, ZERO
      PARAMETER          ( ONE = 1.0D+0, ZERO = 0.0D+0 )
!     ..
!     .. Local Scalars ..
      LOGICAL            NOUNIT, UPPER
      INTEGER            J, JB, NB, NN
!     ..
!     .. External Functions ..
!      LOGICAL            LSAME
!      INTEGER            ILAENV
!      EXTERNAL           LSAME, ILAENV
!     ..
!     .. External Subroutines ..
!      EXTERNAL           DTRMM, DTRSM, DTRTI2, XERBLA
!     ..
!     .. Intrinsic Functions ..
      INTRINSIC          MAX, MIN
!     ..
!     .. Executable Statements ..
!
!     Test the input parameters.
!
      INFO = 0
      UPPER = LSAME( UPLO, 'U' )
      NOUNIT = LSAME( DIAG, 'N' )
      IF( .NOT.UPPER .AND. .NOT.LSAME( UPLO, 'L' ) ) THEN
         INFO = -1
      ELSE IF( .NOT.NOUNIT .AND. .NOT.LSAME( DIAG, 'U' ) ) THEN
         INFO = -2
      ELSE IF( N.LT.0 ) THEN
```

```
            INFO = -3
         ELSE IF( LDA.LT.MAX( 1, N ) ) THEN
            INFO = -5
         END IF
         IF( INFO.NE.0 ) THEN
            CALL XERBLA( 'DTRTRI', -INFO )
            RETURN
         END IF
!
!        Quick return if possible
!
         IF( N.EQ.0 )&
        &   RETURN
!
!        Check for singularity if non-unit.
!
         IF( NOUNIT ) THEN
            DO 10 INFO = 1, N
               IF( A( INFO, INFO ).EQ.ZERO )&
        &         RETURN
   10       CONTINUE
            INFO = 0
         END IF
!
!        Determine the block size for this environment.
!
         NB = ILAENV( 1, 'DTRTRI', UPLO // DIAG, N, -1, -1, -1 )
         IF( NB.LE.1 .OR. NB.GE.N ) THEN
!
!           Use unblocked code
!
            CALL DTRTI2( UPLO, DIAG, N, A, LDA, INFO )
         ELSE
!
!           Use blocked code
!
            IF( UPPER ) THEN
!
!              Compute inverse of upper triangular matrix
!
               DO 20 J = 1, N, NB
                  JB = MIN( NB, N-J+1 )
!
!                 Compute rows 1:j-1 of current block column
!
                  CALL DTRMM( 'Left', 'Upper', 'No transpose', DIAG, J-1,&
        &                     JB, ONE, A, LDA, A( 1, J ), LDA )
                  CALL DTRSM( 'Right', 'Upper', 'No transpose', DIAG, J-1,&
        &                     JB, -ONE, A( J, J ), LDA, A( 1, J ), LDA )
!
!                 Compute inverse of current diagonal block
!
                  CALL DTRTI2( 'Upper', DIAG, JB, A( J, J ), LDA, INFO )
   20          CONTINUE
            ELSE
!
!              Compute inverse of lower triangular matrix
!
               NN = ( ( N-1 ) / NB )*NB + 1
               DO 30 J = NN, 1, -NB
                  JB = MIN( NB, N-J+1 )
```

```fortran
               IF( J+JB.LE.N ) THEN
!
!                 Compute rows j+jb:n of current block column
!
                  CALL DTRMM( 'Left', 'Lower', 'No transpose', DIAG,  &
     &                        N-J-JB+1, JB, ONE, A( J+JB, J+JB ), LDA,&
     &                        A( J+JB, J ), LDA )
                  CALL DTRSM( 'Right', 'Lower', 'No transpose', DIAG, &
     &                        N-J-JB+1, JB, -ONE, A( J, J ), LDA,     &
     &                        A( J+JB, J ), LDA )
               END IF
!
!              Compute inverse of current diagonal block
!
               CALL DTRTI2( 'Lower', DIAG, JB, A( J, J ), LDA, INFO )
   30       CONTINUE
         END IF
      END IF
!
      RETURN
!
!     End of DTRTRI
!
      END SUBROUTINE DTRTRI

!
!  =====================================================================
      SUBROUTINE DTRMM(SIDE,UPLO,TRANSA,DIAG,M,N,ALPHA,A,LDA,B,LDB)
!
!  -- Reference BLAS level3 routine (version 3.7.0) --
!  -- Reference BLAS is a software package provided by Univ. of Tennessee,    --
!  -- Univ. of California Berkeley, Univ. of Colorado Denver and NAG Ltd..--
!     December 2016
!
!     .. Scalar Arguments ..
      DOUBLE PRECISION ALPHA
      INTEGER LDA,LDB,M,N
      CHARACTER DIAG,SIDE,TRANSA,UPLO
!     ..
!     .. Array Arguments ..
      DOUBLE PRECISION A(LDA,*),B(LDB,*)
!     ..
!
!  =====================================================================
!
!     .. External Functions ..
!      LOGICAL LSAME
!      EXTERNAL LSAME
!     ..
!     .. External Subroutines ..
!      EXTERNAL XERBLA
!     ..
!     .. Intrinsic Functions ..
      INTRINSIC MAX
!     ..
!     .. Local Scalars ..
      DOUBLE PRECISION TEMP
      INTEGER I,INFO,J,K,NROWA
      LOGICAL LSIDE,NOUNIT,UPPER
!     ..
!     .. Parameters ..
```

```fortran
      DOUBLE PRECISION ONE,ZERO
      PARAMETER (ONE=1.0D+0,ZERO=0.0D+0)
!     ..
!
!     Test the input parameters.
!
      LSIDE = LSAME(SIDE,'L')
      IF (LSIDE) THEN
          NROWA = M
      ELSE
          NROWA = N
      END IF
      NOUNIT = LSAME(DIAG,'N')
      UPPER = LSAME(UPLO,'U')
!
      INFO = 0
      IF ((.NOT.LSIDE) .AND. (.NOT.LSAME(SIDE,'R'))) THEN
          INFO = 1
      ELSE IF ((.NOT.UPPER) .AND. (.NOT.LSAME(UPLO,'L'))) THEN
          INFO = 2
      ELSE IF ((.NOT.LSAME(TRANSA,'N')) .AND.&
     &         (.NOT.LSAME(TRANSA,'T')) .AND.&
     &         (.NOT.LSAME(TRANSA,'C'))) THEN
          INFO = 3
      ELSE IF ((.NOT.LSAME(DIAG,'U')) .AND. (.NOT.LSAME(DIAG,'N'))) THEN
          INFO = 4
      ELSE IF (M.LT.0) THEN
          INFO = 5
      ELSE IF (N.LT.0) THEN
          INFO = 6
      ELSE IF (LDA.LT.MAX(1,NROWA)) THEN
          INFO = 9
      ELSE IF (LDB.LT.MAX(1,M)) THEN
          INFO = 11
      END IF
      IF (INFO.NE.0) THEN
          CALL XERBLA('DTRMM ',INFO)
          RETURN
      END IF
!
!     Quick return if possible.
!
      IF (M.EQ.0 .OR. N.EQ.0) RETURN
!
!     And when  alpha.eq.zero.
!
      IF (ALPHA.EQ.ZERO) THEN
          DO 20 J = 1,N
              DO 10 I = 1,M
                  B(I,J) = ZERO
  10          CONTINUE
  20      CONTINUE
          RETURN
      END IF
!
!     Start the operations.
!
      IF (LSIDE) THEN
          IF (LSAME(TRANSA,'N')) THEN
!
!           Form  B := alpha*A*B.
```

```
!
              IF (UPPER) THEN
                  DO 50 J = 1,N
                      DO 40 K = 1,M
                          IF (B(K,J).NE.ZERO) THEN
                              TEMP = ALPHA*B(K,J)
                              DO 30 I = 1,K - 1
                                  B(I,J) = B(I,J) + TEMP*A(I,K)
   30                         CONTINUE
                              IF (NOUNIT) TEMP = TEMP*A(K,K)
                              B(K,J) = TEMP
                          END IF
   40                 CONTINUE
   50             CONTINUE
              ELSE
                  DO 80 J = 1,N
                      DO 70 K = M,1,-1
                          IF (B(K,J).NE.ZERO) THEN
                              TEMP = ALPHA*B(K,J)
                              B(K,J) = TEMP
                              IF (NOUNIT) B(K,J) = B(K,J)*A(K,K)
                              DO 60 I = K + 1,M
                                  B(I,J) = B(I,J) + TEMP*A(I,K)
   60                         CONTINUE
                          END IF
   70                 CONTINUE
   80             CONTINUE
              END IF
          ELSE
!
!           Form  B := alpha*A**T*B.
!
              IF (UPPER) THEN
                  DO 110 J = 1,N
                      DO 100 I = M,1,-1
                          TEMP = B(I,J)
                          IF (NOUNIT) TEMP = TEMP*A(I,I)
                          DO 90 K = 1,I - 1
                              TEMP = TEMP + A(K,I)*B(K,J)
   90                     CONTINUE
                          B(I,J) = ALPHA*TEMP
  100                 CONTINUE
  110             CONTINUE
              ELSE
                  DO 140 J = 1,N
                      DO 130 I = 1,M
                          TEMP = B(I,J)
                          IF (NOUNIT) TEMP = TEMP*A(I,I)
                          DO 120 K = I + 1,M
                              TEMP = TEMP + A(K,I)*B(K,J)
  120                     CONTINUE
                          B(I,J) = ALPHA*TEMP
  130                 CONTINUE
  140             CONTINUE
              END IF
          END IF
      ELSE
          IF (LSAME(TRANSA,'N')) THEN
!
!           Form  B := alpha*B*A.
!
```

```fortran
              IF (UPPER) THEN
                  DO 180 J = N,1,-1
                      TEMP = ALPHA
                      IF (NOUNIT) TEMP = TEMP*A(J,J)
                      DO 150 I = 1,M
                          B(I,J) = TEMP*B(I,J)
 150                  CONTINUE
                      DO 170 K = 1,J - 1
                          IF (A(K,J).NE.ZERO) THEN
                              TEMP = ALPHA*A(K,J)
                              DO 160 I = 1,M
                                  B(I,J) = B(I,J) + TEMP*B(I,K)
 160                          CONTINUE
                          END IF
 170                  CONTINUE
 180              CONTINUE
              ELSE
                  DO 220 J = 1,N
                      TEMP = ALPHA
                      IF (NOUNIT) TEMP = TEMP*A(J,J)
                      DO 190 I = 1,M
                          B(I,J) = TEMP*B(I,J)
 190                  CONTINUE
                      DO 210 K = J + 1,N
                          IF (A(K,J).NE.ZERO) THEN
                              TEMP = ALPHA*A(K,J)
                              DO 200 I = 1,M
                                  B(I,J) = B(I,J) + TEMP*B(I,K)
 200                          CONTINUE
                          END IF
 210                  CONTINUE
 220              CONTINUE
              END IF
          ELSE
!
!           Form  B := alpha*B*A**T.
!
              IF (UPPER) THEN
                  DO 260 K = 1,N
                      DO 240 J = 1,K - 1
                          IF (A(J,K).NE.ZERO) THEN
                              TEMP = ALPHA*A(J,K)
                              DO 230 I = 1,M
                                  B(I,J) = B(I,J) + TEMP*B(I,K)
 230                          CONTINUE
                          END IF
 240                  CONTINUE
                      TEMP = ALPHA
                      IF (NOUNIT) TEMP = TEMP*A(K,K)
                      IF (TEMP.NE.ONE) THEN
                          DO 250 I = 1,M
                              B(I,K) = TEMP*B(I,K)
 250                      CONTINUE
                      END IF
 260              CONTINUE
              ELSE
                  DO 300 K = N,1,-1
                      DO 280 J = K + 1,N
                          IF (A(J,K).NE.ZERO) THEN
                              TEMP = ALPHA*A(J,K)
                              DO 270 I = 1,M
```

```
                                B(I,J) = B(I,J) + TEMP*B(I,K)
  270                       CONTINUE
                        END IF
  280                 CONTINUE
                      TEMP = ALPHA
                      IF (NOUNIT) TEMP = TEMP*A(K,K)
                      IF (TEMP.NE.ONE) THEN
                          DO 290 I = 1,M
                              B(I,K) = TEMP*B(I,K)
  290                     CONTINUE
                      END IF
  300             CONTINUE
              END IF
          END IF
      END IF
!
      RETURN
!
!     End of DTRMM .
!
      END SUBROUTINE DTRMM

!
!  =====================================================================
      SUBROUTINE DTRTI2( UPLO, DIAG, N, A, LDA, INFO )
!
!  -- LAPACK computational routine (version 3.7.0) --
!  -- LAPACK is a software package provided by Univ. of Tennessee,    --
!  -- Univ. of California Berkeley, Univ. of Colorado Denver and NAG Ltd..--
!     December 2016
!
!     .. Scalar Arguments ..
      CHARACTER          DIAG, UPLO
      INTEGER            INFO, LDA, N
!     ..
!     .. Array Arguments ..
      DOUBLE PRECISION   A( LDA, * )
!     ..
!
!  =====================================================================
!
!     .. Parameters ..
      DOUBLE PRECISION   ONE
      PARAMETER          ( ONE = 1.0D+0 )
!     ..
!     .. Local Scalars ..
      LOGICAL            NOUNIT, UPPER
      INTEGER            J
      DOUBLE PRECISION   AJJ
!     ..
!     .. External Functions ..
!      LOGICAL            LSAME
!      EXTERNAL           LSAME
!     ..
!     .. External Subroutines ..
!      EXTERNAL           DSCAL, DTRMV, XERBLA
!     ..
!     .. Intrinsic Functions ..
      INTRINSIC          MAX
!     ..
!     .. Executable Statements ..
```

```
!
!     Test the input parameters.
!
      INFO = 0
      UPPER = LSAME( UPLO, 'U' )
      NOUNIT = LSAME( DIAG, 'N' )
      IF( .NOT.UPPER .AND. .NOT.LSAME( UPLO, 'L' ) ) THEN
         INFO = -1
      ELSE IF( .NOT.NOUNIT .AND. .NOT.LSAME( DIAG, 'U' ) ) THEN
         INFO = -2
      ELSE IF( N.LT.0 ) THEN
         INFO = -3
      ELSE IF( LDA.LT.MAX( 1, N ) ) THEN
         INFO = -5
      END IF
      IF( INFO.NE.0 ) THEN
         CALL XERBLA( 'DTRTI2', -INFO )
         RETURN
      END IF
!
      IF( UPPER ) THEN
!
!        Compute inverse of upper triangular matrix.
!
         DO 10 J = 1, N
            IF( NOUNIT ) THEN
               A( J, J ) = ONE / A( J, J )
               AJJ = -A( J, J )
            ELSE
               AJJ = -ONE
            END IF
!
!           Compute elements 1:j-1 of j-th column.
!
            CALL DTRMV( 'Upper', 'No transpose', DIAG, J-1, A, LDA,&
     &                  A( 1, J ), 1 )
            CALL DSCAL( J-1, AJJ, A( 1, J ), 1 )
   10    CONTINUE
      ELSE
!
!        Compute inverse of lower triangular matrix.
!
         DO 20 J = N, 1, -1
            IF( NOUNIT ) THEN
               A( J, J ) = ONE / A( J, J )
               AJJ = -A( J, J )
            ELSE
               AJJ = -ONE
            END IF
            IF( J.LT.N ) THEN
!
!              Compute elements j+1:n of j-th column.
!
               CALL DTRMV( 'Lower', 'No transpose', DIAG, N-J, &
     &                     A( J+1, J+1 ), LDA, A( J+1, J ), 1 )
               CALL DSCAL( N-J, AJJ, A( J+1, J ), 1 )
            END IF
   20    CONTINUE
      END IF
!
      RETURN
```

```
!
!     End of DTRTI2
!
      END SUBROUTINE DTRTI2
!
!  =====================================================================
      SUBROUTINE DTRMV(UPLO,TRANS,DIAG,N,A,LDA,X,INCX)
!
!  -- Reference BLAS level2 routine (version 3.7.0) --
!  -- Reference BLAS is a software package provided by Univ. of Tennessee,    --
!  -- Univ. of California Berkeley, Univ. of Colorado Denver and NAG Ltd..--
!     December 2016
!
!     .. Scalar Arguments ..
      INTEGER INCX,LDA,N
      CHARACTER DIAG,TRANS,UPLO
!     ..
!     .. Array Arguments ..
      DOUBLE PRECISION A(LDA,*),X(*)
!     ..
!
!  =====================================================================
!
!     .. Parameters ..
      DOUBLE PRECISION ZERO
      PARAMETER (ZERO=0.0D+0)
!     ..
!     .. Local Scalars ..
      DOUBLE PRECISION TEMP
      INTEGER I,INFO,IX,J,JX,KX
      LOGICAL NOUNIT
!     ..
!     .. External Functions ..
!      LOGICAL LSAME
!      EXTERNAL LSAME
!     ..
!     .. External Subroutines ..
!      EXTERNAL XERBLA
!     ..
!     .. Intrinsic Functions ..
      INTRINSIC MAX
!     ..
!
!     Test the input parameters.
!
      INFO = 0
      IF (.NOT.LSAME(UPLO,'U') .AND. .NOT.LSAME(UPLO,'L')) THEN
          INFO = 1
      ELSE IF (.NOT.LSAME(TRANS,'N') .AND. .NOT.LSAME(TRANS,'T') .AND. &
     &          .NOT.LSAME(TRANS,'C')) THEN
          INFO = 2
      ELSE IF (.NOT.LSAME(DIAG,'U') .AND. .NOT.LSAME(DIAG,'N')) THEN
          INFO = 3
      ELSE IF (N.LT.0) THEN
          INFO = 4
      ELSE IF (LDA.LT.MAX(1,N)) THEN
          INFO = 6
      ELSE IF (INCX.EQ.0) THEN
          INFO = 8
      END IF
      IF (INFO.NE.0) THEN
```

```
          CALL XERBLA('DTRMV ',INFO)
          RETURN
      END IF
!
!     Quick return if possible.
!
      IF (N.EQ.0) RETURN
!
      NOUNIT = LSAME(DIAG,'N')
!
!     Set up the start point in X if the increment is not unity. This
!     will be  ( N - 1 )*INCX  too small for descending loops.
!
      IF (INCX.LE.0) THEN
          KX = 1 - (N-1)*INCX
      ELSE IF (INCX.NE.1) THEN
          KX = 1
      END IF
!
!     Start the operations. In this version the elements of A are
!     accessed sequentially with one pass through A.
!
      IF (LSAME(TRANS,'N')) THEN
!
!        Form  x := A*x.
!
          IF (LSAME(UPLO,'U')) THEN
              IF (INCX.EQ.1) THEN
                  DO 20 J = 1,N
                      IF (X(J).NE.ZERO) THEN
                          TEMP = X(J)
                          DO 10 I = 1,J - 1
                              X(I) = X(I) + TEMP*A(I,J)
   10                     CONTINUE
                          IF (NOUNIT) X(J) = X(J)*A(J,J)
                      END IF
   20             CONTINUE
              ELSE
                  JX = KX
                  DO 40 J = 1,N
                      IF (X(JX).NE.ZERO) THEN
                          TEMP = X(JX)
                          IX = KX
                          DO 30 I = 1,J - 1
                              X(IX) = X(IX) + TEMP*A(I,J)
                              IX = IX + INCX
   30                     CONTINUE
                          IF (NOUNIT) X(JX) = X(JX)*A(J,J)
                      END IF
                      JX = JX + INCX
   40             CONTINUE
              END IF
          ELSE
              IF (INCX.EQ.1) THEN
                  DO 60 J = N,1,-1
                      IF (X(J).NE.ZERO) THEN
                          TEMP = X(J)
                          DO 50 I = N,J + 1,-1
                              X(I) = X(I) + TEMP*A(I,J)
   50                     CONTINUE
                          IF (NOUNIT) X(J) = X(J)*A(J,J)
```

```
                    END IF
   60           CONTINUE
            ELSE
                KX = KX + (N-1)*INCX
                JX = KX
                DO 80 J = N,1,-1
                    IF (X(JX).NE.ZERO) THEN
                        TEMP = X(JX)
                        IX = KX
                        DO 70 I = N,J + 1,-1
                            X(IX) = X(IX) + TEMP*A(I,J)
                            IX = IX - INCX
   70                   CONTINUE
                        IF (NOUNIT) X(JX) = X(JX)*A(J,J)
                    END IF
                    JX = JX - INCX
   80           CONTINUE
            END IF
        END IF
    ELSE
!
!        Form  x := A**T*x.
!
        IF (LSAME(UPLO,'U')) THEN
            IF (INCX.EQ.1) THEN
                DO 100 J = N,1,-1
                    TEMP = X(J)
                    IF (NOUNIT) TEMP = TEMP*A(J,J)
                    DO 90 I = J - 1,1,-1
                        TEMP = TEMP + A(I,J)*X(I)
   90               CONTINUE
                    X(J) = TEMP
  100           CONTINUE
            ELSE
                JX = KX + (N-1)*INCX
                DO 120 J = N,1,-1
                    TEMP = X(JX)
                    IX = JX
                    IF (NOUNIT) TEMP = TEMP*A(J,J)
                    DO 110 I = J - 1,1,-1
                        IX = IX - INCX
                        TEMP = TEMP + A(I,J)*X(IX)
  110               CONTINUE
                    X(JX) = TEMP
                    JX = JX - INCX
  120           CONTINUE
            END IF
        ELSE
            IF (INCX.EQ.1) THEN
                DO 140 J = 1,N
                    TEMP = X(J)
                    IF (NOUNIT) TEMP = TEMP*A(J,J)
                    DO 130 I = J + 1,N
                        TEMP = TEMP + A(I,J)*X(I)
  130               CONTINUE
                    X(J) = TEMP
  140           CONTINUE
            ELSE
                JX = KX
                DO 160 J = 1,N
                    TEMP = X(JX)
```

```fortran
                      IX = JX
                      IF (NOUNIT) TEMP = TEMP*A(J,J)
                      DO 150 I = J + 1,N
                          IX = IX + INCX
                          TEMP = TEMP + A(I,J)*X(IX)
  150                 CONTINUE
                      X(JX) = TEMP
                      JX = JX + INCX
  160             CONTINUE
              END IF
          END IF
      END IF
!
      RETURN
!
!     End of DTRMV .
!
      END SUBROUTINE DTRMV

!
!  =====================================================================
      SUBROUTINE DGETRI( N, A, LDA, IPIV, WORK, LWORK, INFO )
!
!  -- LAPACK computational routine (version 3.7.0) --
!  -- LAPACK is a software package provided by Univ. of Tennessee,    --
!  -- Univ. of California Berkeley, Univ. of Colorado Denver and NAG Ltd..--
!     December 2016
!
!     .. Scalar Arguments ..
      INTEGER            INFO, LDA, LWORK, N
!     ..
!     .. Array Arguments ..
      INTEGER            IPIV( * )
      DOUBLE PRECISION   A( LDA, * ), WORK( * )
!     ..
!
!  =====================================================================
!
!     .. Parameters ..
      DOUBLE PRECISION   ZERO, ONE
      PARAMETER          ( ZERO = 0.0D+0, ONE = 1.0D+0 )
!     ..
!     .. Local Scalars ..
      LOGICAL            LQUERY
      INTEGER            I, IWS, J, JB, JJ, JP, LDWORK, LWKOPT, NB, &
     &                   NBMIN, NN
!     ..
!     .. External Functions ..
!      INTEGER            ILAENV
!      EXTERNAL           ILAENV
!     ..
!     .. External Subroutines ..
!      EXTERNAL           DGEMM, DGEMV, DSWAP, DTRSM, DTRTRI, XERBLA
!     ..
!     .. Intrinsic Functions ..
      INTRINSIC          MAX, MIN
!     ..
!     .. Executable Statements ..
!
!     Test the input parameters.
!
```

```
      INFO = 0
      NB = ILAENV( 1, 'DGETRI', ' ', N, -1, -1, -1 )
      LWKOPT = N*NB
      WORK( 1 ) = LWKOPT
      LQUERY = ( LWORK.EQ.-1 )
      IF( N.LT.0 ) THEN
         INFO = -1
      ELSE IF( LDA.LT.MAX( 1, N ) ) THEN
         INFO = -3
      ELSE IF( LWORK.LT.MAX( 1, N ) .AND. .NOT.LQUERY ) THEN
         INFO = -6
      END IF
      IF( INFO.NE.0 ) THEN
         CALL XERBLA( 'DGETRI', -INFO )
         RETURN
      ELSE IF( LQUERY ) THEN
         RETURN
      END IF
!
!     Quick return if possible
!
      IF( N.EQ.0 ) &
     &   RETURN
!
!     Form inv(U).  If INFO > 0 from DTRTRI, then U is singular,
!     and the inverse is not computed.
!
      CALL DTRTRI( 'Upper', 'Non-unit', N, A, LDA, INFO )
      IF( INFO.GT.0 ) &
     &    RETURN
!
      NBMIN = 2
      LDWORK = N
      IF( NB.GT.1 .AND. NB.LT.N ) THEN
         IWS = MAX( LDWORK*NB, 1 )
         IF( LWORK.LT.IWS ) THEN
            NB = LWORK / LDWORK
            NBMIN = MAX( 2, ILAENV( 2, 'DGETRI', ' ', N, -1, -1, -1 ) )
         END IF
      ELSE
         IWS = N
      END IF
!
!     Solve the equation inv(A)*L = inv(U) for inv(A).
!
      IF( NB.LT.NBMIN .OR. NB.GE.N ) THEN
!
!        Use unblocked code.
!
         DO 20 J = N, 1, -1
!
!           Copy current column of L to WORK and replace with zeros.
!
            DO 10 I = J + 1, N
               WORK( I ) = A( I, J )
               A( I, J ) = ZERO
   10       CONTINUE
!
!           Compute current column of inv(A).
!
            IF( J.LT.N ) &
```

```
     &          CALL DGEMV( 'No transpose', N, N-J, -ONE, A( 1, J+1 ), &
     &                      LDA, WORK( J+1 ), 1, ONE, A( 1, J ), 1 )
   20    CONTINUE
      ELSE
!
!        Use blocked code.
!
         NN = ( ( N-1 ) / NB )*NB + 1
         DO 50 J = NN, 1, -NB
            JB = MIN( NB, N-J+1 )
!
!           Copy current block column of L to WORK and replace with
!           zeros.
!
            DO 40 JJ = J, J + JB - 1
               DO 30 I = JJ + 1, N
                  WORK( I+( JJ-J )*LDWORK ) = A( I, JJ )
                  A( I, JJ ) = ZERO
   30          CONTINUE
   40       CONTINUE
!
!           Compute current block column of inv(A).
!
            IF( J+JB.LE.N ) &
     &         CALL DGEMM( 'No transpose', 'No transpose', N, JB,         &
     &                     N-J-JB+1, -ONE, A( 1, J+JB ), LDA,             &
     &                     WORK( J+JB ), LDWORK, ONE, A( 1, J ), LDA )
            CALL DTRSM( 'Right', 'Lower', 'No transpose', 'Unit', N, JB, &
     &                  ONE, WORK( J ), LDWORK, A( 1, J ), LDA )
   50    CONTINUE
      END IF
!
!     Apply column interchanges.
!
      DO 60 J = N - 1, 1, -1
         JP = IPIV( J )
         IF( JP.NE.J ) &
     &      CALL DSWAP( N, A( 1, J ), 1, A( 1, JP ), 1 )
   60 CONTINUE
!
      WORK( 1 ) = IWS
      RETURN
!
!     End of DGETRI
!
      END SUBROUTINE DGETRI
!
!  =====================================================================
      SUBROUTINE DPBTF2( UPLO, N, KD, AB, LDAB, INFO )
!
!  -- LAPACK computational routine (version 3.7.0) --
!  -- LAPACK is a software package provided by Univ. of Tennessee,    --
!  -- Univ. of California Berkeley, Univ. of Colorado Denver and NAG Ltd..--
!     December 2016
!
!     .. Scalar Arguments ..
      CHARACTER          UPLO
      INTEGER            INFO, KD, LDAB, N
!     ..
!     .. Array Arguments ..
      DOUBLE PRECISION   AB( LDAB, * )
```

```
!      ..
!
!  =====================================================================
!
!      .. Parameters ..
      DOUBLE PRECISION   ONE, ZERO
      PARAMETER          ( ONE = 1.0D+0, ZERO = 0.0D+0 )
!      ..
!      .. Local Scalars ..
      LOGICAL            UPPER
      INTEGER            J, KLD, KN
      DOUBLE PRECISION   AJJ
!      ..
!      .. External Functions ..
!       LOGICAL            LSAME
!       EXTERNAL           LSAME
!      ..
!      .. External Subroutines ..
!       EXTERNAL           DSCAL, DSYR, XERBLA
!      ..
!      .. Intrinsic Functions ..
      INTRINSIC          MAX, MIN, SQRT
!      ..
!      .. Executable Statements ..
!
!      Test the input parameters.
!
      INFO = 0
      UPPER = LSAME( UPLO, 'U' )
      IF( .NOT.UPPER .AND. .NOT.LSAME( UPLO, 'L' ) ) THEN
         INFO = -1
      ELSE IF( N.LT.0 ) THEN
         INFO = -2
      ELSE IF( KD.LT.0 ) THEN
         INFO = -3
      ELSE IF( LDAB.LT.KD+1 ) THEN
         INFO = -5
      END IF
      IF( INFO.NE.0 ) THEN
         CALL XERBLA( 'DPBTF2', -INFO )
         RETURN
      END IF
!
!      Quick return if possible
!
      IF( N.EQ.0 ) &
     &   RETURN
!
      KLD = MAX( 1, LDAB-1 )
!
      IF( UPPER ) THEN
!
!         Compute the Cholesky factorization A = U**T*U.
!
         DO 10 J = 1, N
!
!            Compute U(J,J) and test for non-positive-definiteness.
!
            AJJ = AB( KD+1, J )
            IF( AJJ.LE.ZERO ) &
     &         GO TO 30
```

```
            AJJ = SQRT( AJJ )
            AB( KD+1, J ) = AJJ
!
!           Compute elements J+1:J+KN of row J and update the
!           trailing submatrix within the band.
!
            KN = MIN( KD, N-J )
            IF( KN.GT.0 ) THEN
               CALL DSCAL( KN, ONE / AJJ, AB( KD, J+1 ), KLD )
               CALL DSYR( 'Upper', KN, -ONE, AB( KD, J+1 ), KLD, &
     &                    AB( KD+1, J+1 ), KLD )
            END IF
   10    CONTINUE
      ELSE
!
!        Compute the Cholesky factorization A = L*L**T.
!
         DO 20 J = 1, N
!
!           Compute L(J,J) and test for non-positive-definiteness.
!
            AJJ = AB( 1, J )
            IF( AJJ.LE.ZERO ) &
     &         GO TO 30
            AJJ = SQRT( AJJ )
            AB( 1, J ) = AJJ
!
!           Compute elements J+1:J+KN of column J and update the
!           trailing submatrix within the band.
!
            KN = MIN( KD, N-J )
            IF( KN.GT.0 ) THEN
               CALL DSCAL( KN, ONE / AJJ, AB( 2, J ), 1 )
               CALL DSYR( 'Lower', KN, -ONE, AB( 2, J ), 1, &
     &                    AB( 1, J+1 ), KLD )
            END IF
   20    CONTINUE
      END IF
      RETURN
!
   30 CONTINUE
      INFO = J
      RETURN
!
!     End of DPBTF2
!
      END SUBROUTINE DPBTF2
!
!  ====================================================================
      SUBROUTINE DSYR(UPLO,N,ALPHA,X,INCX,A,LDA)
!
!  -- Reference BLAS level2 routine (version 3.7.0) --
!  -- Reference BLAS is a software package provided by Univ. of Tennessee,    --
!  -- Univ. of California Berkeley, Univ. of Colorado Denver and NAG Ltd..--
!     December 2016
!
!     .. Scalar Arguments ..
      DOUBLE PRECISION ALPHA
      INTEGER INCX,LDA,N
      CHARACTER UPLO
!     ..
```

```
!     .. Array Arguments ..
      DOUBLE PRECISION A(LDA,*),X(*)
!     ..
!
!  ====================================================================
!
!     .. Parameters ..
      DOUBLE PRECISION ZERO
      PARAMETER (ZERO=0.0D+0)
!     ..
!     .. Local Scalars ..
      DOUBLE PRECISION TEMP
      INTEGER I,INFO,IX,J,JX,KX
!     ..
!     .. External Functions ..
!      LOGICAL LSAME
!      EXTERNAL LSAME
!     ..
!     .. External Subroutines ..
!      EXTERNAL XERBLA
!     ..
!     .. Intrinsic Functions ..
      INTRINSIC MAX
!     ..
!
!     Test the input parameters.
!
      INFO = 0
      IF (.NOT.LSAME(UPLO,'U') .AND. .NOT.LSAME(UPLO,'L')) THEN
          INFO = 1
      ELSE IF (N.LT.0) THEN
          INFO = 2
      ELSE IF (INCX.EQ.0) THEN
          INFO = 5
      ELSE IF (LDA.LT.MAX(1,N)) THEN
          INFO = 7
      END IF
      IF (INFO.NE.0) THEN
          CALL XERBLA('DSYR  ',INFO)
          RETURN
      END IF
!
!     Quick return if possible.
!
      IF ((N.EQ.0) .OR. (ALPHA.EQ.ZERO)) RETURN
!
!     Set the start point in X if the increment is not unity.
!
      IF (INCX.LE.0) THEN
          KX = 1 - (N-1)*INCX
      ELSE IF (INCX.NE.1) THEN
          KX = 1
      END IF
!
!     Start the operations. In this version the elements of A are
!     accessed sequentially with one pass through the triangular part
!     of A.
!
      IF (LSAME(UPLO,'U')) THEN
!
!        Form  A  when A is stored in upper triangle.
```

```
!
         IF (INCX.EQ.1) THEN
             DO 20 J = 1,N
                 IF (X(J).NE.ZERO) THEN
                     TEMP = ALPHA*X(J)
                     DO 10 I = 1,J
                         A(I,J) = A(I,J) + X(I)*TEMP
   10                CONTINUE
                 END IF
   20        CONTINUE
         ELSE
             JX = KX
             DO 40 J = 1,N
                 IF (X(JX).NE.ZERO) THEN
                     TEMP = ALPHA*X(JX)
                     IX = KX
                     DO 30 I = 1,J
                         A(I,J) = A(I,J) + X(IX)*TEMP
                         IX = IX + INCX
   30                CONTINUE
                 END IF
                 JX = JX + INCX
   40        CONTINUE
         END IF
     ELSE
!
!        Form  A  when A is stored in lower triangle.
!
         IF (INCX.EQ.1) THEN
             DO 60 J = 1,N
                 IF (X(J).NE.ZERO) THEN
                     TEMP = ALPHA*X(J)
                     DO 50 I = J,N
                         A(I,J) = A(I,J) + X(I)*TEMP
   50                CONTINUE
                 END IF
   60        CONTINUE
         ELSE
             JX = KX
             DO 80 J = 1,N
                 IF (X(JX).NE.ZERO) THEN
                     TEMP = ALPHA*X(JX)
                     IX = JX
                     DO 70 I = J,N
                         A(I,J) = A(I,J) + X(IX)*TEMP
                         IX = IX + INCX
   70                CONTINUE
                 END IF
                 JX = JX + INCX
   80        CONTINUE
         END IF
     END IF
!
     RETURN
!
!    End of DSYR  .
!
     END SUBROUTINE DSYR

!
!  =====================================================================
```

```
      SUBROUTINE DPOTF2( UPLO, N, A, LDA, INFO )
!
!  -- LAPACK computational routine (version 3.7.0) --
!  -- LAPACK is a software package provided by Univ. of Tennessee,    --
!  -- Univ. of California Berkeley, Univ. of Colorado Denver and NAG Ltd..--
!     December 2016
!
!     .. Scalar Arguments ..
      CHARACTER          UPLO
      INTEGER            INFO, LDA, N
!     ..
!     .. Array Arguments ..
      DOUBLE PRECISION   A( LDA, * )
!     ..
!
!  =====================================================================
!
!     .. Parameters ..
      DOUBLE PRECISION   ONE, ZERO
      PARAMETER          ( ONE = 1.0D+0, ZERO = 0.0D+0 )
!     ..
!     .. Local Scalars ..
      LOGICAL            UPPER
      INTEGER            J
      DOUBLE PRECISION   AJJ
!     ..
!     .. External Functions ..
!      LOGICAL            LSAME, DISNAN
!      DOUBLE PRECISION   DDOT
!      EXTERNAL           LSAME, DDOT, DISNAN
!     ..
!     .. External Subroutines ..
!      EXTERNAL           DGEMV, DSCAL, XERBLA
!     ..
!     .. Intrinsic Functions ..
      INTRINSIC          MAX, SQRT
!     ..
!     .. Executable Statements ..
!
!     Test the input parameters.
!
      INFO = 0
      UPPER = LSAME( UPLO, 'U' )
      IF( .NOT.UPPER .AND. .NOT.LSAME( UPLO, 'L' ) ) THEN
         INFO = -1
      ELSE IF( N.LT.0 ) THEN
         INFO = -2
      ELSE IF( LDA.LT.MAX( 1, N ) ) THEN
         INFO = -4
      END IF
      IF( INFO.NE.0 ) THEN
         CALL XERBLA( 'DPOTF2', -INFO )
         RETURN
      END IF
!
!     Quick return if possible
!
      IF( N.EQ.0 ) &
     &   RETURN
!
      IF( UPPER ) THEN
```

```fortran
!
!        Compute the Cholesky factorization A = U**T *U.
!
         DO 10 J = 1, N
!
!           Compute U(J,J) and test for non-positive-definiteness.
!
            AJJ = A( J, J ) - DDOT( J-1, A( 1, J ), 1, A( 1, J ), 1 )
            IF( AJJ.LE.ZERO.OR.DISNAN( AJJ ) ) THEN
               A( J, J ) = AJJ
               GO TO 30
            END IF
            AJJ = SQRT( AJJ )
            A( J, J ) = AJJ
!
!           Compute elements J+1:N of row J.
!
            IF( J.LT.N ) THEN
               CALL DGEMV( 'Transpose', J-1, N-J, -ONE, A( 1, J+1 ), &
     &                     LDA, A( 1, J ), 1, ONE, A( J, J+1 ), LDA )
               CALL DSCAL( N-J, ONE / AJJ, A( J, J+1 ), LDA )
            END IF
   10    CONTINUE
      ELSE
!
!        Compute the Cholesky factorization A = L*L**T.
!
         DO 20 J = 1, N
!
!           Compute L(J,J) and test for non-positive-definiteness.
!
            AJJ = A( J, J ) - DDOT( J-1, A( J, 1 ), LDA, A( J, 1 ), &
     &            LDA )
            IF( AJJ.LE.ZERO.OR.DISNAN( AJJ ) ) THEN
               A( J, J ) = AJJ
               GO TO 30
            END IF
            AJJ = SQRT( AJJ )
            A( J, J ) = AJJ
!
!           Compute elements J+1:N of column J.
!
            IF( J.LT.N ) THEN
               CALL DGEMV( 'No transpose', N-J, J-1, -ONE, A( J+1, 1 ), &
     &                     LDA, A( J, 1 ), LDA, ONE, A( J+1, J ), 1 )
               CALL DSCAL( N-J, ONE / AJJ, A( J+1, J ), 1 )
            END IF
   20    CONTINUE
      END IF
      GO TO 40
!
   30 CONTINUE
      INFO = J
!
   40 CONTINUE
      RETURN
!
!     End of DPOTF2
!
      END SUBROUTINE DPOTF2
!
```

```fortran
!  ====================================================================
      DOUBLE PRECISION FUNCTION DDOT(N,DX,INCX,DY,INCY)
!
!  -- Reference BLAS level1 routine (version 3.8.0) --
!  -- Reference BLAS is a software package provided by Univ. of Tennessee,    --
!  -- Univ. of California Berkeley, Univ. of Colorado Denver and NAG Ltd..--
!     November 2017
!
!     .. Scalar Arguments ..
      INTEGER INCX,INCY,N
!     ..
!     .. Array Arguments ..
      DOUBLE PRECISION DX(*),DY(*)
!     ..
!
!  ====================================================================
!
!     .. Local Scalars ..
      DOUBLE PRECISION DTEMP
      INTEGER I,IX,IY,M,MP1
!     ..
!     .. Intrinsic Functions ..
      INTRINSIC MOD
!     ..
      DDOT = 0.0d0
      DTEMP = 0.0d0
      IF (N.LE.0) RETURN
      IF (INCX.EQ.1 .AND. INCY.EQ.1) THEN
!
!        code for both increments equal to 1
!
!
!        clean-up loop
!
         M = MOD(N,5)
         IF (M.NE.0) THEN
            DO I = 1,M
               DTEMP = DTEMP + DX(I)*DY(I)
            END DO
            IF (N.LT.5) THEN
               DDOT=DTEMP
            RETURN
            END IF
         END IF
         MP1 = M + 1
         DO I = MP1,N,5
          DTEMP = DTEMP + DX(I)*DY(I) + DX(I+1)*DY(I+1) + &
     &            DX(I+2)*DY(I+2) + DX(I+3)*DY(I+3) + DX(I+4)*DY(I+4)
         END DO
      ELSE
!
!        code for unequal increments or equal increments
!          not equal to 1
!
         IX = 1
         IY = 1
         IF (INCX.LT.0) IX = (-N+1)*INCX + 1
         IF (INCY.LT.0) IY = (-N+1)*INCY + 1
         DO I = 1,N
            DTEMP = DTEMP + DX(IX)*DY(IY)
            IX = IX + INCX
```

```
            IY = IY + INCY
         END DO
      END IF
      DDOT = DTEMP
      RETURN
      END FUNCTION DDOT

!
!  =====================================================================
      LOGICAL FUNCTION DISNAN( DIN )
!
!  -- LAPACK auxiliary routine (version 3.7.1) --
!  -- LAPACK is a software package provided by Univ. of Tennessee,    --
!  -- Univ. of California Berkeley, Univ. of Colorado Denver and NAG Ltd..--
!     June 2017
!
!     .. Scalar Arguments ..
      DOUBLE PRECISION, INTENT(IN) :: DIN
!     ..
!
!  =====================================================================
!
!  .. External Functions ..
!      LOGICAL DLAISNAN
!      EXTERNAL DLAISNAN
!  ..
!  .. Executable Statements ..
      DISNAN = DLAISNAN(DIN,DIN)
      RETURN
      END FUNCTION DISNAN

!
!  =====================================================================
      LOGICAL FUNCTION DLAISNAN( DIN1, DIN2 )
!
!  -- LAPACK auxiliary routine (version 3.7.1) --
!  -- LAPACK is a software package provided by Univ. of Tennessee,    --
!  -- Univ. of California Berkeley, Univ. of Colorado Denver and NAG Ltd..--
!     June 2017
!
!     .. Scalar Arguments ..
      DOUBLE PRECISION, INTENT(IN) :: DIN1, DIN2
!     ..
!
!  =====================================================================
!
!  .. Executable Statements ..
      DLAISNAN = (DIN1.NE.DIN2)
      RETURN
      END FUNCTION DLAISNAN

!
!  =====================================================================
      SUBROUTINE DSYRK(UPLO,TRANS,N,K,ALPHA,A,LDA,BETA,C,LDC)
!
!  -- Reference BLAS level3 routine (version 3.7.0) --
!  -- Reference BLAS is a software package provided by Univ. of Tennessee,    --
!  -- Univ. of California Berkeley, Univ. of Colorado Denver and NAG Ltd..--
!     December 2016
!
!     .. Scalar Arguments ..
```

```
      DOUBLE PRECISION ALPHA,BETA
      INTEGER K,LDA,LDC,N
      CHARACTER TRANS,UPLO
!     ..
!     .. Array Arguments ..
      DOUBLE PRECISION A(LDA,*),C(LDC,*)
!     ..
!
!  =====================================================================
!
!     .. External Functions ..
!      LOGICAL LSAME
!      EXTERNAL LSAME
!     ..
!     .. External Subroutines ..
!      EXTERNAL XERBLA
!     ..
!     .. Intrinsic Functions ..
      INTRINSIC MAX
!     ..
!     .. Local Scalars ..
      DOUBLE PRECISION TEMP
      INTEGER I,INFO,J,L,NROWA
      LOGICAL UPPER
!     ..
!     .. Parameters ..
      DOUBLE PRECISION ONE,ZERO
      PARAMETER (ONE=1.0D+0,ZERO=0.0D+0)
!     ..
!
!     Test the input parameters.
!
      IF (LSAME(TRANS,'N')) THEN
          NROWA = N
      ELSE
          NROWA = K
      END IF
      UPPER = LSAME(UPLO,'U')
!
      INFO = 0
      IF ((.NOT.UPPER) .AND. (.NOT.LSAME(UPLO,'L'))) THEN
          INFO = 1
      ELSE IF ((.NOT.LSAME(TRANS,'N')) .AND. &
     &         (.NOT.LSAME(TRANS,'T')) .AND. &
     &         (.NOT.LSAME(TRANS,'C'))) THEN
          INFO = 2
      ELSE IF (N.LT.0) THEN
          INFO = 3
      ELSE IF (K.LT.0) THEN
          INFO = 4
      ELSE IF (LDA.LT.MAX(1,NROWA)) THEN
          INFO = 7
      ELSE IF (LDC.LT.MAX(1,N)) THEN
          INFO = 10
      END IF
      IF (INFO.NE.0) THEN
          CALL XERBLA('DSYRK ',INFO)
          RETURN
      END IF
!
!     Quick return if possible.
```

```
!
      IF ((N.EQ.0) .OR. (((ALPHA.EQ.ZERO).OR. &
     &    (K.EQ.0)).AND. (BETA.EQ.ONE))) RETURN
!
!     And when  alpha.eq.zero.
!
      IF (ALPHA.EQ.ZERO) THEN
          IF (UPPER) THEN
              IF (BETA.EQ.ZERO) THEN
                  DO 20 J = 1,N
                      DO 10 I = 1,J
                          C(I,J) = ZERO
   10                 CONTINUE
   20             CONTINUE
              ELSE
                  DO 40 J = 1,N
                      DO 30 I = 1,J
                          C(I,J) = BETA*C(I,J)
   30                 CONTINUE
   40             CONTINUE
              END IF
          ELSE
              IF (BETA.EQ.ZERO) THEN
                  DO 60 J = 1,N
                      DO 50 I = J,N
                          C(I,J) = ZERO
   50                 CONTINUE
   60             CONTINUE
              ELSE
                  DO 80 J = 1,N
                      DO 70 I = J,N
                          C(I,J) = BETA*C(I,J)
   70                 CONTINUE
   80             CONTINUE
              END IF
          END IF
          RETURN
      END IF
!
!     Start the operations.
!
      IF (LSAME(TRANS,'N')) THEN
!
!        Form  C := alpha*A*A**T + beta*C.
!
          IF (UPPER) THEN
              DO 130 J = 1,N
                  IF (BETA.EQ.ZERO) THEN
                      DO 90 I = 1,J
                          C(I,J) = ZERO
   90                 CONTINUE
                  ELSE IF (BETA.NE.ONE) THEN
                      DO 100 I = 1,J
                          C(I,J) = BETA*C(I,J)
  100                 CONTINUE
                  END IF
                  DO 120 L = 1,K
                      IF (A(J,L).NE.ZERO) THEN
                          TEMP = ALPHA*A(J,L)
                          DO 110 I = 1,J
                              C(I,J) = C(I,J) + TEMP*A(I,L)
```

```
110                   CONTINUE
                   END IF
120             CONTINUE
130         CONTINUE
        ELSE
            DO 180 J = 1,N
                IF (BETA.EQ.ZERO) THEN
                    DO 140 I = J,N
                        C(I,J) = ZERO
140                 CONTINUE
                ELSE IF (BETA.NE.ONE) THEN
                    DO 150 I = J,N
                        C(I,J) = BETA*C(I,J)
150                 CONTINUE
                END IF
                DO 170 L = 1,K
                    IF (A(J,L).NE.ZERO) THEN
                        TEMP = ALPHA*A(J,L)
                        DO 160 I = J,N
                            C(I,J) = C(I,J) + TEMP*A(I,L)
160                     CONTINUE
                    END IF
170             CONTINUE
180         CONTINUE
        END IF
    ELSE
!
!       Form  C := alpha*A**T*A + beta*C.
!
        IF (UPPER) THEN
            DO 210 J = 1,N
                DO 200 I = 1,J
                    TEMP = ZERO
                    DO 190 L = 1,K
                        TEMP = TEMP + A(L,I)*A(L,J)
190                 CONTINUE
                    IF (BETA.EQ.ZERO) THEN
                        C(I,J) = ALPHA*TEMP
                    ELSE
                        C(I,J) = ALPHA*TEMP + BETA*C(I,J)
                    END IF
200             CONTINUE
210         CONTINUE
        ELSE
            DO 240 J = 1,N
                DO 230 I = J,N
                    TEMP = ZERO
                    DO 220 L = 1,K
                        TEMP = TEMP + A(L,I)*A(L,J)
220                 CONTINUE
                    IF (BETA.EQ.ZERO) THEN
                        C(I,J) = ALPHA*TEMP
                    ELSE
                        C(I,J) = ALPHA*TEMP + BETA*C(I,J)
                    END IF
230             CONTINUE
240         CONTINUE
        END IF
    END IF
!
    RETURN
```

```
!
!     End of DSYRK .
!
      END SUBROUTINE DSYRK

!
!
!  =====================================================================
      SUBROUTINE DPBTRF( UPLO, N, KD, AB, LDAB, INFO )
!
!  -- LAPACK computational routine (version 3.7.0) --
!  -- LAPACK is a software package provided by Univ. of Tennessee,    --
!  -- Univ. of California Berkeley, Univ. of Colorado Denver and NAG Ltd..--
!     December 2016
!
!     .. Scalar Arguments ..
      CHARACTER          UPLO
      INTEGER            INFO, KD, LDAB, N
!     ..
!     .. Array Arguments ..
      DOUBLE PRECISION   AB( LDAB, * )
!     ..
!
!  =====================================================================
!
!     .. Parameters ..
      DOUBLE PRECISION   ONE, ZERO
      PARAMETER          ( ONE = 1.0D+0, ZERO = 0.0D+0 )
      INTEGER            NBMAX, LDWORK
      PARAMETER          ( NBMAX = 32, LDWORK = NBMAX+1 )
!     ..
!     .. Local Scalars ..
      INTEGER            I, I2, I3, IB, II, J, JJ, NB
!     ..
!     .. Local Arrays ..
      DOUBLE PRECISION   WORK( LDWORK, NBMAX )
!     ..
!     .. External Functions ..
!      LOGICAL            LSAME
!      INTEGER            ILAENV
!      EXTERNAL           LSAME, ILAENV
!     ..
!     .. External Subroutines ..
!      EXTERNAL           DGEMM, DPBTF2, DPOTF2, DSYRK, DTRSM, XERBLA
!     ..
!     .. Intrinsic Functions ..
      INTRINSIC          MIN
!     ..
!     .. Executable Statements ..
!
!     Test the input parameters.
!
      INFO = 0
      IF( ( .NOT.LSAME( UPLO, 'U' ) ) .AND. &
     &    ( .NOT.LSAME( UPLO, 'L' ) ) ) THEN
         INFO = -1
      ELSE IF( N.LT.0 ) THEN
         INFO = -2
      ELSE IF( KD.LT.0 ) THEN
         INFO = -3
      ELSE IF( LDAB.LT.KD+1 ) THEN
```

```
         INFO = -5
      END IF
      IF( INFO.NE.0 ) THEN
         CALL XERBLA( 'DPBTRF', -INFO )
         RETURN
      END IF
!
!     Quick return if possible
!
      IF( N.EQ.0 ) &
     &   RETURN
!
!     Determine the block size for this environment
!
      NB = ILAENV( 1, 'DPBTRF', UPLO, N, KD, -1, -1 )
!
!     The block size must not exceed the semi-bandwidth KD, and must not
!     exceed the limit set by the size of the local array WORK.
!
      NB = MIN( NB, NBMAX )
!
      IF( NB.LE.1 .OR. NB.GT.KD ) THEN
!
!        Use unblocked code
!
         CALL DPBTF2( UPLO, N, KD, AB, LDAB, INFO )
      ELSE
!
!        Use blocked code
!
         IF( LSAME( UPLO, 'U' ) ) THEN
!
!           Compute the Cholesky factorization of a symmetric band
!           matrix, given the upper triangle of the matrix in band
!           storage.
!
!           Zero the upper triangle of the work array.
!
            DO 20 J = 1, NB
               DO 10 I = 1, J - 1
                  WORK( I, J ) = ZERO
   10          CONTINUE
   20       CONTINUE
!
!           Process the band matrix one diagonal block at a time.
!
            DO 70 I = 1, N, NB
               IB = MIN( NB, N-I+1 )
!
!              Factorize the diagonal block
!
               CALL DPOTF2( UPLO, IB, AB( KD+1, I ), LDAB-1, II )
               IF( II.NE.0 ) THEN
                  INFO = I + II - 1
                  GO TO 150
               END IF
               IF( I+IB.LE.N ) THEN
!
!                 Update the relevant part of the trailing submatrix.
!                 If A11 denotes the diagonal block which has just been
!                 factorized, then we need to update the remaining
```

```
!                 blocks in the diagram:
!
!                    A11   A12   A13
!                          A22   A23
!                                A33
!
!                 The numbers of rows and columns in the partitioning
!                 are IB, I2, I3 respectively. The blocks A12, A22 and
!                 A23 are empty if IB = KD. The upper triangle of A13
!                 lies outside the band.
!
                  I2 = MIN( KD-IB, N-I-IB+1 )
                  I3 = MIN( IB, N-I-KD+1 )
!
                  IF( I2.GT.0 ) THEN
!
!                    Update A12
!
                     CALL DTRSM( 'Left', 'Upper', 'Transpose',           &
     &                           'Non-unit', IB, I2, ONE, AB( KD+1, I ), &
     &                           LDAB-1, AB( KD+1-IB, I+IB ), LDAB-1 )
!
!                    Update A22
!
                     CALL DSYRK( 'Upper', 'Transpose', I2, IB, -ONE,     &
     &                           AB( KD+1-IB, I+IB ), LDAB-1, ONE,       &
     &                           AB( KD+1, I+IB ), LDAB-1 )
                  END IF
!
                  IF( I3.GT.0 ) THEN
!
!                    Copy the lower triangle of A13 into the work array.
!
                     DO 40 JJ = 1, I3
                        DO 30 II = JJ, IB
                           WORK( II, JJ ) = AB( II-JJ+1, JJ+I+KD-1 )
  30                    CONTINUE
  40                 CONTINUE
!
!                    Update A13 (in the work array).
!
                     CALL DTRSM( 'Left', 'Upper', 'Transpose',           &
     &                           'Non-unit', IB, I3, ONE, AB( KD+1, I ), &
     &                           LDAB-1, WORK, LDWORK )
!
!                    Update A23
!
                     IF( I2.GT.0 )                                       &
     &                  CALL DGEMM( 'Transpose', 'No Transpose', I2, I3, &
     &                              IB, -ONE, AB( KD+1-IB, I+IB ),       &
     &                              LDAB-1, WORK, LDWORK, ONE,           &
     &                              AB( 1+IB, I+KD ), LDAB-1 )
!
!                    Update A33
!
                     CALL DSYRK( 'Upper', 'Transpose', I3, IB, -ONE,     &
     &                           WORK, LDWORK, ONE, AB( KD+1, I+KD ),    &
     &                           LDAB-1 )
!
!                    Copy the lower triangle of A13 back into place.
!
```

```fortran
                     DO 60 JJ = 1, I3
                        DO 50 II = JJ, IB
                           AB( II-JJ+1, JJ+I+KD-1 ) = WORK( II, JJ )
   50                   CONTINUE
   60                CONTINUE
                  END IF
               END IF
   70       CONTINUE
         ELSE
!
!           Compute the Cholesky factorization of a symmetric band
!           matrix, given the lower triangle of the matrix in band
!           storage.
!
!           Zero the lower triangle of the work array.
!
            DO 90 J = 1, NB
               DO 80 I = J + 1, NB
                  WORK( I, J ) = ZERO
   80          CONTINUE
   90       CONTINUE
!
!           Process the band matrix one diagonal block at a time.
!
            DO 140 I = 1, N, NB
               IB = MIN( NB, N-I+1 )
!
!              Factorize the diagonal block
!
               CALL DPOTF2( UPLO, IB, AB( 1, I ), LDAB-1, II )
               IF( II.NE.0 ) THEN
                  INFO = I + II - 1
                  GO TO 150
               END IF
               IF( I+IB.LE.N ) THEN
!
!                 Update the relevant part of the trailing submatrix.
!                 If A11 denotes the diagonal block which has just been
!                 factorized, then we need to update the remaining
!                 blocks in the diagram:
!
!                    A11
!                    A21   A22
!                    A31   A32   A33
!
!                 The numbers of rows and columns in the partitioning
!                 are IB, I2, I3 respectively. The blocks A21, A22 and
!                 A32 are empty if IB = KD. The lower triangle of A31
!                 lies outside the band.
!
                  I2 = MIN( KD-IB, N-I-IB+1 )
                  I3 = MIN( IB, N-I-KD+1 )
!
                  IF( I2.GT.0 ) THEN
!
!                    Update A21
!
                     CALL DTRSM( 'Right', 'Lower', 'Transpose',         &
     &                           'Non-unit', I2, IB, ONE, AB( 1, I ),    &
     &                           LDAB-1, AB( 1+IB, I ), LDAB-1 )
!
```

```fortran
!                    Update A22
!
                     CALL DSYRK( 'Lower', 'No Transpose', I2, IB, -ONE,  &
     &                           AB( 1+IB, I ), LDAB-1, ONE,             &
     &                           AB( 1, I+IB ), LDAB-1 )
                  END IF
!
                  IF( I3.GT.0 ) THEN
!
!                    Copy the upper triangle of A31 into the work array.
!
                     DO 110 JJ = 1, IB
                        DO 100 II = 1, MIN( JJ, I3 )
                           WORK( II, JJ ) = AB( KD+1-JJ+II, JJ+I-1 )
  100                   CONTINUE
  110                CONTINUE
!
!                    Update A31 (in the work array).
!
                     CALL DTRSM( 'Right', 'Lower', 'Transpose',          &
     &                           'Non-unit', I3, IB, ONE, AB( 1, I ),    &
     &                           LDAB-1, WORK, LDWORK )
!
!                    Update A32
!
                     IF( I2.GT.0 )                                       &
     &                  CALL DGEMM( 'No transpose', 'Transpose', I3, I2, &
     &                              IB, -ONE, WORK, LDWORK,              &
     &                              AB( 1+IB, I ), LDAB-1, ONE,          &
     &                              AB( 1+KD-IB, I+IB ), LDAB-1 )
!
!                    Update A33
!
                     CALL DSYRK( 'Lower', 'No Transpose', I3, IB, -ONE,  &
     &                           WORK, LDWORK, ONE, AB( 1, I+KD ),       &
     &                           LDAB-1 )
!
!                    Copy the upper triangle of A31 back into place.
!
                     DO 130 JJ = 1, IB
                        DO 120 II = 1, MIN( JJ, I3 )
                           AB( KD+1-JJ+II, JJ+I-1 ) = WORK( II, JJ )
  120                   CONTINUE
  130                CONTINUE
                  END IF
               END IF
  140       CONTINUE
         END IF
      END IF
      RETURN
!
  150 CONTINUE
      RETURN
!
!     End of DPBTRF
!
      END SUBROUTINE DPBTRF
!
!  =====================================================================
      REAL FUNCTION SDSDOT(N,SB,SX,INCX,SY,INCY)
!
```

```fortran
!  -- Reference BLAS level1 routine (version 3.8.0) --
!  -- Reference BLAS is a software package provided by Univ. of Tennessee,    --
!  -- Univ. of California Berkeley, Univ. of Colorado Denver and NAG Ltd..--
!     November 2017
!
!     .. Scalar Arguments ..
      REAL SB
      INTEGER INCX,INCY,N
!     ..
!     .. Array Arguments ..
      DOUBLE PRECISION SX(:),SY(:)

!     .. Local Scalars ..
      DOUBLE PRECISION DSDOT
      INTEGER I,KX,KY,NS
!     ..
!     .. Intrinsic Functions ..
      INTRINSIC DBLE
!     ..
      DSDOT = SB
      IF (N.LE.0) THEN
         SDSDOT = DSDOT
         RETURN
      END IF
      IF (INCX.EQ.INCY .AND. INCX.GT.0) THEN
!
!     Code for equal and positive increments.
!
         NS = N*INCX
         DO I = 1,NS,INCX
            DSDOT = DSDOT + DBLE(SX(I))*DBLE(SY(I))
         END DO
      ELSE
!
!     Code for unequal or nonpositive increments.
!
         KX = 1
         KY = 1
         IF (INCX.LT.0) KX = 1 + (1-N)*INCX
         IF (INCY.LT.0) KY = 1 + (1-N)*INCY
         DO I = 1,N
            DSDOT = DSDOT + DBLE(SX(KX))*DBLE(SY(KY))
            KX = KX + INCX
            KY = KY + INCY
         END DO
      END IF
      SDSDOT = DSDOT
      RETURN
      END FUNCTION SDSDOT

END MODULE ModuleLapack
```

8.12 makefile

```
all: multi-pred clean

multi-pred: ModuleIO.o ModuleGlobalParameters.o ModuleErrors.o ModuleFiles.o      \
            files.o ModuleReadWrite.o ModuleLapack.o ReadInput.o ModuleMultiPred.o\
             MultiPredSolver.o multi-pred.o
             ifort ModuleIO.o ModuleGlobalParameters.o ModuleErrors.o ModuleFiles.o\
             files.o ModuleReadWrite.o ModuleLapack.o ReadInput.o ModuleMultiPred.o\
             MultiPredSolver.o multi-pred.o -o multi-pred

ModuleIO.o: ModuleIO.f90
    ifort -O3 -fast -c ModuleIO.f90

ModuleGlobalParameters.o: ModuleGlobalParameters.f90
    ifort -O3 -fast -c ModuleGlobalParameters.f90

ModuleErrors.o: ModuleErrors.f90
    ifort -O3 -fast -c ModuleErrors.f90

ModuleFiles.o: ModuleFiles.f90
    ifort -O3 -fast -c ModuleFiles.f90

files.o: files.f90
    ifort -O3 -fast -c files.f90

ModuleReadWrite.o: ModuleReadWrite.f90
    ifort -O3 -fast -c ModuleReadWrite.f90

ModuleLapack.o: ModuleLapack.f90
    ifort -O3 -fast -cpp -c ModuleLapack.f90

ReadInput.o: ReadInput.f90
    ifort -O3 -fast -c ReadInput.f90

ModuleMultiPred.o: ModuleMultiPred.f90
    ifort -O3 -fast -c ModuleMultiPred.f90

MultiPredSolver.o: MultiPredSolver.f90
    ifort -O3 -fast -c MultiPredSolver.f90

multi-pred.o: multi-pred.f90
    ifort -O3 -fast -c multi-pred.f90

clean:
    rm -f *.o *.mod

```