

UC Berkeley

UC Berkeley Previously Published Works

Title

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

Permalink

<https://escholarship.org/uc/item/35g8f29z>

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

1	INTRODUCTION	6
2	PREDICTIVE MODELING OF COUPLED MULTI-PHYSICS SYSTEMS (PM-CMPS)	9
2.1	Introduction	9
2.2	Mathematical Framework	10
2.2.1	<i>A Priori Information for Two Multi-Physics Models</i>	10
2.2.2	<i>Construction of the A Priori Distribution Function $p(\alpha, \beta, r, q)$ as the Maximum Entropy Principle Approximation of the True but Unknown A Priori Distribution Function $P(\alpha, \beta, r, q)$</i>	14
2.2.3	<i>Construction of the A Posteriori Predicted Mean Values and Covariances for the Given Models (Likelihood Function) and Maximum Entropy Prior Distribution</i>	18
2.2.4	<i>Construction of the A Posteriori Predicted Consistency Metrics for Model Validation</i>	26
2.3	Discussion and Particular Cases.....	27
2.3.1	<i>Predictive modeling for a Single Multi-Physics Model</i>	29
2.3.2	<i>Predictive modeling for Model A with β additional parameters, but no additional responses</i>	30
2.3.3	<i>Predictive modeling for Model A with q additional responses, but no additional parameters</i>	31
3	PREDICTIVE MODELING OF A SIMPLE NEUTRON DIFFUSION MODEL	33
4	INVERSE PREDICTIVE MODELING OF RADIATION TRANSPORT THROUGH OPTICALLY THICK MEDIA IN THE PRESENCE OF COUNTING UNCERTAINTIES	40
4.1	Transport of Uncollided Photons through a Slab	40
4.2	Determination of Slab Thickness from Detector Response in the Absence of Uncertainties	42
4.3	Traditional Chi-Square Minimization Method for Determining the Slab's Thickness from Detector Responses in the Presence of Counting Uncertainties	44
4.4	Applying the PM-CMPS Methodology for the Inverse Determination of Slab Thickness in the Presence of Counting Uncertainties.....	53
4.4.1	<i>Prediction of Optically Very Thin Slab (Exact Optical Thickness=0.1)</i>	58
4.4.2	<i>Prediction of Optically Thin Slab (Exact Optical Thickness =1.0)</i>	59
4.4.3	<i>Prediction of Optically Thick Slab (Exact Optical Thickness=3.0)</i>	61
4.4.4	<i>Prediction of Optically Very Thick Slab (Exact Optical Thickness=7.0)</i>	62
4.4.5	<i>Prediction of Extremely Thick Slab (Exact Optical Thickness=10.0)</i>	64

4.4.6	<i>Prediction Limit for Single-Precision Computations: Slab of Exact Optical Thickness=10.0</i>	69
5	PREDICTIVE MODELING APPLICATION TO SAVANNAH RIVER NATIONAL LABORATORY'S F-AREA COOLING TOWERS.....	70
5.1	Introduction.....	70
5.2	Mathematical Model of the Counter-Flow Cooling Tower.....	72
5.3	Adjoint Sensitivity Analysis of Cooling Tower Model.....	89
5.4	Predictive Modeling: Optimal Best-Estimate Results with Reduced Predicted Uncertainties.....	99
6	REFERENCES.....	118
7	MULTI-PRED CODE MODULE.....	123
7.1	Directories.....	123
7.2	Code Compilation and Execution.....	125
7.3	Input and Output File Organization.....	125
7.3.1	<i>Super File</i>	125
7.3.2	<i>File "dimensions.inp"</i>	129
7.3.3	<i>Contents and Organization of Input and Output Files</i>	130
7.4	Input Data Files.....	136
7.4.1	<i>Input Data Files for MULTI-PRED Case 1</i>	136
7.4.2	<i>Input Data Files for MULTI-PRED Case 2</i>	141
7.4.3	<i>Input Data Files for MULTI-PRED Case 3</i>	141
7.4.4	<i>Input Data Files for MULTI-PRED Case 4</i>	142
7.5	Output Data Files.....	143
7.5.1	<i>Output Data Files for MULTI-PRED Case 1</i>	143
7.5.2	<i>Output Data Files for MULTI-PRED Case 2</i>	148
7.5.3	<i>Output Data Files for MULTI-PRED Case 3</i>	148
7.5.4	<i>Output Data Files for MULTI-PRED Case 4</i>	149
8	FORTRAN Source Code for the Program Multi-Pred.....	150
8.1	Main program multi-pred.f90.....	152
8.2	Module ModuleGlobalParameters.f90.....	153
8.3	Module ModuleIO.f90.....	155
8.4	Module ModuleErrors.f90.....	156
8.5	Subroutine Files.f90.....	157

8.6	Module ModuleFiles.f90.....	159
8.7	Subroutine ReadInput.f90.....	177
8.8	Module ModuleReadWrite.f90.....	184
8.9	Subroutine MultiPredSolver.f90.....	190
8.10	Module ModuleMultiPred.f90.....	190
8.11	Module ModuleLapack.f90.....	210
8.12	makefile.....	289

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 this 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 of 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

$$\boldsymbol{\alpha} \triangleq (\alpha_1, \dots, \alpha_{N_\alpha}), \quad (2.1)$$

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

$$\boldsymbol{\beta} \triangleq (\beta_1, \dots, \beta_{N_\beta}). \quad (2.2)$$

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

$$\mathbf{r} \triangleq (r_1, \dots, r_{N_r}), \quad (2.3)$$

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

$$\mathbf{q} \triangleq (q_1, \dots, q_{N_q}). \quad (2.4)$$

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 $P(\mathbf{\alpha}, \mathbf{\beta}, \mathbf{r}, \mathbf{q})$. For large-scale systems, as customarily encountered in practice, the probability distribution $P(\mathbf{\alpha}, \mathbf{\beta}, \mathbf{r}, \mathbf{q})$ 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 $f(\mathbf{\alpha}, \mathbf{\beta}, \mathbf{r}, \mathbf{q})$ over the joint probability distribution $P(\mathbf{\alpha}, \mathbf{\beta}, \mathbf{r}, \mathbf{q})$, i.e.,

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

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 $\mathbf{\alpha}^0$ defined as

$$\mathbf{\alpha}^0 \triangleq (\alpha_1^0, \dots, \alpha_{N_\alpha}^0). \quad (2.6)$$

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 $\mathbf{\beta}^0$ defined as

$$\mathbf{\beta}^0 \triangleq (\beta_1^0, \dots, \beta_{N_\beta}^0). \quad (2.7)$$

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

$$\mathbf{C}_{\alpha\alpha}^{(N_\alpha \times N_\alpha)} \triangleq [\text{cov}(\alpha_i, \alpha_j)]_{N_\alpha \times N_\alpha} \triangleq \langle (\alpha_i - \alpha_i^0)(\alpha_j - \alpha_j^0) \rangle_{N_\alpha \times N_\alpha}; \quad i, j = 1, \dots, N_\alpha, \quad (2.8)$$

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

$$\mathbf{C}_{\beta\beta}^{(N_\beta \times N_\beta)} \triangleq [\text{cov}(\beta_i, \beta_j)]_{N_\beta \times N_\beta} \triangleq \langle (\beta_i - \beta_i^0)(\beta_j - \beta_j^0) \rangle_{N_\beta \times N_\beta}; \quad i, j = 1, \dots, N_\beta. \quad (2.9)$$

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

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

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

$$\mathbf{r}^m \triangleq (r_1^m, \dots, r_{N_r}^m), \quad r_i^m \triangleq \langle r_i \rangle, \quad i = 1, \dots, N_r, \quad (2.11)$$

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

$$\mathbf{C}_{rr}^{(N_r \times N_r)} \triangleq \langle (r_i - r_i^m)(r_j - r_j^m) \rangle_{N_r \times N_r}, \quad i, j = 1, \dots, N_r. \quad (2.12)$$

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

$$\mathbf{q}^m \triangleq (q_1^m, \dots, q_{N_q}^m), \quad q_j^m \triangleq \langle q_j \rangle, \quad j = 1, \dots, N_q, \quad (2.13)$$

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

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

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

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

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

$$\mathbf{C}_{ar}^{(N_a \times N_r)} \triangleq \left\langle (\mathbf{a} - \mathbf{a}^0)(\mathbf{r} - \mathbf{r}^m)^\dagger \right\rangle \triangleq \left[\mathbf{C}_{ra}^{(N_r \times N_a)} \right]^\dagger, \quad (2.16)$$

$$\mathbf{C}_{aq}^{(N_a \times N_q)} \triangleq \left\langle (\mathbf{a} - \mathbf{a}^0)(\mathbf{q} - \mathbf{q}^m)^\dagger \right\rangle \triangleq \left[\mathbf{C}_{qa}^{(N_q \times N_a)} \right]^\dagger, \quad (2.17)$$

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

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

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

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 $p(\mathbf{a}, \mathbf{\beta}, \mathbf{r}, \mathbf{q})$. Mathematically, these means and covariances are functionals of $p(\mathbf{a}, \mathbf{\beta}, \mathbf{r}, \mathbf{q})$, having the generic form

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

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

$$K \triangleq N_\alpha + N_\beta + N_r + N_q + N_\alpha^2 + N_\beta^2 + N_r^2 + N_q^2 + (N_\alpha \times N_\beta) + (N_\alpha \times N_r) + (N_\alpha \times N_q) \\ + (N_\beta \times N_r) + (N_\beta \times N_q) + (N_r \times N_q). \quad (2.21)$$

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

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

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

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

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

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

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

$$\langle F_k \rangle = - \frac{\partial}{\partial \lambda_k} \ln Z, \quad k = 1, 2, \dots, K, \quad (2.25)$$

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 $p(\mathbf{x})$:

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

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

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

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 $(N_r + N_q)$ experimentally measured responses. These computed responses will be considered to be the components of two vectors, denoted as $\mathbf{r}^c(\boldsymbol{\alpha}, \boldsymbol{\beta}) \triangleq (r_1^c, \dots, r_{N_r}^c)$ and $\mathbf{q}^c(\boldsymbol{\alpha}, \boldsymbol{\beta}) \triangleq (q_1^c, \dots, q_{N_q}^c)$, 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, $\mathbf{r}^c(\boldsymbol{\alpha}, \boldsymbol{\beta})$ and $\mathbf{q}^c(\boldsymbol{\alpha}, \boldsymbol{\beta})$ 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 $\mathbf{x} \triangleq (\boldsymbol{\alpha}, \boldsymbol{\beta}, \mathbf{r}, \mathbf{q})$. 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 $\mathbf{r}^c(\boldsymbol{\alpha}, \boldsymbol{\beta})$ and $\mathbf{q}^c(\boldsymbol{\alpha}, \boldsymbol{\beta})$. 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

$$\mathbf{r} = \mathbf{r}^c(\boldsymbol{\alpha}, \boldsymbol{\beta}), \quad \mathbf{q} = \mathbf{q}^c(\boldsymbol{\alpha}, \boldsymbol{\beta}) . \quad (2.28)$$

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:

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

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 $\mathbf{x}^{pred} \triangleq (\boldsymbol{\alpha}^{pred}, \boldsymbol{\beta}^{pred}, \mathbf{r}^{pred}, \mathbf{q}^{pred})$ where the following conditions are simultaneously fulfilled:

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

$$\nabla_{\boldsymbol{\alpha}} P = \mathbf{0}; \quad \nabla_{\boldsymbol{\beta}} P = \mathbf{0}; \quad \nabla_r P = \mathbf{0}; \quad \nabla_q P = \mathbf{0}. \quad (2.31)$$

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:

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

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

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

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

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

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

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

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

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

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

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:

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

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

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 $(\boldsymbol{\alpha}^{pred}, \boldsymbol{\beta}^{pred}, \boldsymbol{\lambda}_r, \boldsymbol{\lambda}_q)$; (ii) from this system, eliminate the quantities $(\boldsymbol{\alpha}^{pred} - \boldsymbol{\alpha}^0)$ and $(\boldsymbol{\beta}^{pred} - \boldsymbol{\beta}^0)$; and (iii) re-arrange the resulting equations to obtain the following coupled equations for the Lagrange multipliers:

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

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:

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

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

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

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

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

Note that the vectors $\mathbf{r}^d(\boldsymbol{\alpha}^0, \boldsymbol{\beta}^0)$ and $\mathbf{q}^d(\boldsymbol{\alpha}^0, \boldsymbol{\beta}^0)$ 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.,

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

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

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

(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.,

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

The Lagrange multipliers λ_r and λ_q are obtained by solving Eq.(2.41), which requires the inverse of the matrix

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

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

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

Where

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

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

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

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

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:

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

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

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

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

where

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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:

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

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:

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

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:

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

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:

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

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

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

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:

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

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

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

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

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

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

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

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

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

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

Note that the quadratic form on the rightmost-side of Eq.(2.86) is distributed according to a χ^2 distribution with $(N_r + N_q)$ 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 , $0 \leq x < \infty$, is defined as

$$P(x < \chi^2 < x + dx)dx = \frac{1}{2^{n/2} \Gamma(n/2)} x^{n/2-1} e^{-x/2} dx, \quad x > 0, \quad (n = 1, 2, \dots). \quad (2.87)$$

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 $\text{var}(x) = 2n$. The value of χ^2 is computed using Eq.(2.86) to obtain

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

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 $(N_r + N_q)^2$. 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 $(\mathbf{D}_{qq} - \mathbf{D}_{rq}^\dagger \mathbf{D}_{rr}^{-1} \mathbf{D}_{rq})$, 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:

$$\mathbf{D}_{rq} = \mathbf{0}, \mathbf{D}_{qr} = \mathbf{0}, \mathbf{D}_{qq} = \mathbf{0}, \mathbf{D}_{rr} = \mathbf{S}_{ra} \mathbf{C}_{\alpha\alpha} \mathbf{S}_{ra}^\dagger - \mathbf{S}_{ra} \mathbf{C}_{ar} - \mathbf{C}_{ar}^\dagger \mathbf{S}_{ra}^\dagger + \mathbf{C}_{rr}. \quad (2.89)$$

$$\mathbf{X}_\alpha \triangleq \mathbf{C}_{\alpha\alpha} \mathbf{S}_{ra}^\dagger - \mathbf{C}_{ar}, \mathbf{Y}_\alpha \triangleq \mathbf{0}, \mathbf{X}_r \triangleq \mathbf{C}_{ar}^\dagger \mathbf{S}_{ra}^\dagger - \mathbf{C}_{rr}, \mathbf{Y}_r = \mathbf{0}. \quad (2.90)$$

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

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

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

$$\mathbf{C}_{\alpha\alpha}^{pred} = \mathbf{C}_{\alpha\alpha} - (\mathbf{C}_{\alpha\alpha} \mathbf{S}_{ra}^\dagger - \mathbf{C}_{ar}) [\mathbf{D}_{rr}]^{-1} (\mathbf{C}_{\alpha\alpha} \mathbf{S}_{ra}^\dagger - \mathbf{C}_{ar})^\dagger, \quad (2.93)$$

$$\mathbf{C}_{rr}^{pred} = \mathbf{C}_{rr} - (\mathbf{C}_{ar}^\dagger \mathbf{S}_{ra}^\dagger - \mathbf{C}_{rr}) [\mathbf{D}_{rr}]^{-1} (\mathbf{C}_{ar}^\dagger \mathbf{S}_{ra}^\dagger - \mathbf{C}_{rr})^\dagger, \quad (2.94)$$

$$\mathbf{C}_{ar}^{pred} = \mathbf{C}_{ar} - (\mathbf{C}_{\alpha\alpha} \mathbf{S}_{ra}^\dagger - \mathbf{C}_{ar}) [\mathbf{D}_{rr}]^{-1} (\mathbf{C}_{ar}^\dagger \mathbf{S}_{ra}^\dagger - \mathbf{C}_{rr})^\dagger. \quad (2.95)$$

Note that if the model is perfect (i.e., $\mathbf{C}_{\alpha\alpha} = \mathbf{0}$ and $\mathbf{C}_{ar} = \mathbf{0}$), then Eqs.(2.91) through (2.95) would yield $\boldsymbol{\alpha}^{pred} = \boldsymbol{\alpha}^0$ and $\mathbf{r}^{pred} = \mathbf{r}^c (\boldsymbol{\alpha}^0, \boldsymbol{\beta}^0)$, predicted ‘‘perfectly,’’ without any accompanying uncertainties (i.e., $\mathbf{C}_{rr}^{pred} = \mathbf{0}$, $\mathbf{C}_{\alpha\alpha}^{pred} = \mathbf{0}$, $\mathbf{C}_{ar}^{pred} = \mathbf{0}$). 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 $\mathbf{C}_{ar} = \mathbf{0}$), but the model were imperfect, then Eqs. (2.91) through (2.95) would yield $\boldsymbol{\alpha}^{pred} = \boldsymbol{\alpha}^0 - \mathbf{C}_{aa} \mathbf{S}_{ra}^\dagger [\mathbf{S}_{ra} \mathbf{C}_{aa} \mathbf{S}_{ra}^\dagger]^{-1} \mathbf{r}^d (\boldsymbol{\alpha}^0)$, $\mathbf{C}_{aa}^{pred} = \mathbf{C}_{aa} - \mathbf{C}_{aa} \mathbf{S}_{ra}^\dagger [\mathbf{S}_{ra} \mathbf{C}_{aa} \mathbf{S}_{ra}^\dagger]^{-1} \mathbf{S}_{ra} \mathbf{C}_{aa}$, $\mathbf{r}^{pred} = \mathbf{r}^m$, $\mathbf{C}_{rr}^{pred} = \mathbf{0}$, $\mathbf{C}_{ar}^{pred} = \mathbf{0}$. 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 β additional parameters, but no additional responses

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

$$\mathbf{D}_{rq} = \mathbf{0}, \mathbf{D}_{qr} = \mathbf{0}, \mathbf{D}_{qq} = \mathbf{0}, \quad (2.96)$$

$$\mathbf{D}_{rr} = \mathbf{S}_{ra} (\mathbf{C}_{aa} \mathbf{S}_{ra}^\dagger + \mathbf{C}_{a\beta} \mathbf{S}_{r\beta}^\dagger - \mathbf{C}_{ar}) + \mathbf{S}_{r\beta} (\mathbf{C}_{a\beta}^\dagger \mathbf{S}_{ra}^\dagger + \mathbf{C}_{\beta\beta} \mathbf{S}_{r\beta}^\dagger - \mathbf{C}_{\beta r}) - \mathbf{C}_{ar}^\dagger \mathbf{S}_{ra}^\dagger - \mathbf{C}_{\beta r}^\dagger \mathbf{S}_{r\beta}^\dagger + \mathbf{C}_{rr}. \quad (2.97)$$

$$\mathbf{X}_\alpha \triangleq \mathbf{C}_{aa} \mathbf{S}_{ra}^\dagger + \mathbf{C}_{a\beta} \mathbf{S}_{r\beta}^\dagger - \mathbf{C}_{ar}, \quad (2.98)$$

$$\mathbf{X}_\beta \triangleq \mathbf{C}_{a\beta}^\dagger \mathbf{S}_{ra}^\dagger + \mathbf{C}_{\beta\beta} \mathbf{S}_{r\beta}^\dagger - \mathbf{C}_{\beta r}, \quad (2.99)$$

$$\mathbf{X}_r \triangleq \mathbf{C}_{ar}^\dagger \mathbf{S}_{ra}^\dagger + \mathbf{C}_{\beta r}^\dagger \mathbf{S}_{r\beta}^\dagger - \mathbf{C}_{rr}, \quad (2.100)$$

$$\mathbf{X}_q \triangleq \mathbf{0}, \mathbf{Y}_\alpha \triangleq \mathbf{0}, \mathbf{Y}_r \triangleq \mathbf{0}, \mathbf{Y}_\beta \triangleq \mathbf{0}, \mathbf{Y}_q \triangleq \mathbf{0}, \quad (2.101)$$

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

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

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

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

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

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

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

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

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

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

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

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

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

$$\mathbf{D}_{qr} = \mathbf{S}_{q\alpha} \mathbf{C}_{\alpha\alpha} \mathbf{S}_{r\alpha}^\dagger - \mathbf{C}_{\alpha q}^\dagger \mathbf{S}_{r\alpha}^\dagger - \mathbf{S}_{q\alpha} \mathbf{C}_{\alpha r} + \mathbf{C}_{qr}^\dagger, \quad \text{Dim}(\mathbf{D}_{qr}) = (N_q \times N_r), \quad (2.114)$$

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

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

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

$$\mathbf{X}_\beta \triangleq \mathbf{0}, \mathbf{Y}_\beta \triangleq \mathbf{0}, \quad (2.118)$$

$$\mathbf{X}_r \triangleq \mathbf{C}_{ar}^\dagger \mathbf{S}_{r\alpha}^\dagger - \mathbf{C}_{rr}, \quad (2.119)$$

$$\mathbf{Y}_r \triangleq \mathbf{C}_{ar}^\dagger \mathbf{S}_{q\alpha}^\dagger - \mathbf{C}_{rq} \quad (2.120)$$

$$\mathbf{X}_q \triangleq \mathbf{C}_{aq}^\dagger \mathbf{S}_{r\alpha}^\dagger - \mathbf{C}_{rq}, \quad (2.121)$$

$$\mathbf{Y}_q \triangleq \mathbf{C}_{aq}^\dagger \mathbf{S}_{q\alpha}^\dagger - \mathbf{C}_{qq}, \quad (2.122)$$

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

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

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

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

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

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

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

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

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

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

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:

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

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

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

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 neutrons/cm³·s within a slab of material of extrapolated thickness $2a$. The linear neutron diffusion equation that models mathematically this problem is

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

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.,

$$\varphi(\pm a) = 0. \quad (3.2)$$

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

$$R(\mathbf{e}) \triangleq \Sigma_d \varphi(b), \quad (3.3)$$

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

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

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

$$\mathbf{e} \triangleq (\varphi, \boldsymbol{\alpha}). \quad (3.5)$$

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

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

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:

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

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 $R(\mathbf{e})$. Even though $R(\mathbf{e})$ 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 $R(\mathbf{e})$ on $\boldsymbol{\alpha}$. This fact is confirmed by the explicit expression of $R(\mathbf{e})$ given in Eq.(3.7).

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

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

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

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

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

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: $\Sigma_a^0 = 0.0197 \text{ cm}^{-1}$, $D^0 = 0.16 \text{ cm}$, containing distributed neutron sources emitting nominally $S^0 = 10^7 \text{ neutrons} \cdot \text{cm}^{-3} \cdot \text{s}^{-1}$. 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 $\Sigma_d^0 = 7.438 \text{ cm}^{-1}$, 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

$$\mathbf{C}_\alpha = \begin{pmatrix} (9.85 \times 10^{-4})^2 & 0 & 0 & 0 \\ 0 & (8.0 \times 10^{-3})^2 & 0 & 0 \\ 0 & 0 & (1.5 \times 10^6)^2 & 0 \\ 0 & 0 & 0 & (7.44 \times 10^{-1})^2 \end{pmatrix}. \quad (3.12)$$

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*”):

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

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

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

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

Thus, the covariance matrix of the measured responses is

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

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

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

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

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

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

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

$$\mathbf{S} \triangleq \left(\frac{\partial R_i}{\partial \alpha_j} \right) = \begin{pmatrix} -1.92 \times 10^{11} & -1.33 \times 10^5 & 3.78 \times 10^2 & 5.08 \times 10^8 \\ -1.92 \times 10^{11} & -1.33 \times 10^5 & 3.78 \times 10^2 & 5.08 \times 10^8 \\ -1.76 \times 10^{11} & -1.24 \times 10^9 & 3.66 \times 10^2 & 4.92 \times 10^8 \\ -1.76 \times 10^{11} & -1.24 \times 10^9 & 3.66 \times 10^2 & 4.92 \times 10^8 \end{pmatrix}, \quad (3.22)$$

$$\mathbf{S}_{rel} \triangleq \left(\frac{\partial R_i}{\partial \alpha_j} \frac{\Delta \alpha_j}{R_i} \right) = \begin{pmatrix} -0.99999 & -5.41 \times 10^{-6} & 1.00 & 1.00 \\ -0.99999 & -5.64 \times 10^{-6} & 1.00 & 1.00 \\ -9.46 \times 10^{-1} & -5.64 \times 10^{-2} & 1.00 & 1.00 \\ -9.46 \times 10^{-1} & -5.41 \times 10^{-2} & 1.00 & 1.00 \end{pmatrix}. \quad (3.23)$$

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:

$$\mathbf{C}_{rc} = \mathbf{S} \mathbf{C}_\alpha \mathbf{S}^\dagger = \begin{pmatrix} 4.99 \times 10^{17} & 4.99 \times 10^{17} & 4.82 \times 10^{17} & 4.82 \times 10^{17} \\ 4.99 \times 10^{17} & 4.99 \times 10^{17} & 4.82 \times 10^{17} & 4.82 \times 10^{17} \\ 4.82 \times 10^{17} & 4.82 \times 10^{17} & 4.66 \times 10^{17} & 4.66 \times 10^{17} \\ 4.82 \times 10^{17} & 4.82 \times 10^{17} & 4.66 \times 10^{17} & 4.66 \times 10^{17} \end{pmatrix} \quad (3.24)$$

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:

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

$$D^{be} = 0.1591 \text{ cm}, \quad rsd(D^{be}) = 5.00\%; \quad (3.26)$$

$$S^{be} = 9.85 \times 10^6 \text{ n} \cdot \text{cm}^{-3} \cdot \text{s}^{-1}, \quad rsd(S^{be}) = 9.21\%; \quad (3.27)$$

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

$$\begin{aligned}
\mathbf{C}_\alpha^{be} = & \begin{pmatrix} 9.50 \times 10^{-4} & 0 & 0 & 0 \\ 0 & 7.99 \times 10^{-3} & 0 & 0 \\ 0 & 0 & 9.08 \times 10^5 & 0 \\ 0 & 0 & 0 & 6.30 \times 10^{-1} \end{pmatrix} \\
& \times \begin{pmatrix} 1.0 & -8.89 \times 10^{-4} & 3.51 \times 10^{-1} & 1.67 \times 10^{-1} \\ -8.89 \times 10^{-4} & 1.0 & 1.02 \times 10^{-2} & 4.84 \times 10^{-3} \\ 3.51 \times 10^{-1} & 1.02 \times 10^{-2} & 1.0 & -8.24 \times 10^{-1} \\ 1.67 \times 10^{-1} & 4.84 \times 10^{-3} & -8.24 \times 10^{-1} & 1.0 \end{pmatrix} \\
& \times \begin{pmatrix} 9.50 \times 10^{-4} & 0 & 0 & 0 \\ 0 & 7.99 \times 10^{-3} & 0 & 0 \\ 0 & 0 & 9.08 \times 10^5 & 0 \\ 0 & 0 & 0 & 6.30 \times 10^{-1} \end{pmatrix}, \tag{3.29}
\end{aligned}$$

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

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

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

$$at (-40 \text{ cm}): r_3^{be} = 3.56 \times 10^9 \text{ n} \cdot \text{cm}^{-3} \cdot \text{sec}^{-1}; \text{rsd}(r_3^{be}) = 2.58\%; \tag{3.32}$$

$$at (40 \text{ cm}): r_4^{be} = 3.56 \times 10^9 \text{ n} \cdot \text{cm}^{-3} \cdot \text{sec}^{-1}; \text{rsd}(r_4^{be}) = 2.58\%; \tag{3.33}$$

$$\mathbf{C}_r^{be} = \begin{pmatrix} 9.04 \times 10^{15} & 9.04 \times 10^{15} & 8.64 \times 10^{15} & 8.64 \times 10^{15} \\ 9.04 \times 10^{15} & 9.04 \times 10^{15} & 8.64 \times 10^{15} & 8.64 \times 10^{15} \\ 8.64 \times 10^{15} & 8.64 \times 10^{15} & 8.45 \times 10^{15} & 8.45 \times 10^{15} \\ 8.64 \times 10^{15} & 8.64 \times 10^{15} & 8.45 \times 10^{15} & 8.45 \times 10^{15} \end{pmatrix} \tag{3.34}$$

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

$$\mathbf{C}_{rx}^{be} = \begin{pmatrix} -7.81 \times 10^3 & 3.89 \times 10^4 & 1.38 \times 10^{13} & 4.57 \times 10^6 \\ -7.81 \times 10^3 & 3.89 \times 10^4 & 1.38 \times 10^{13} & 4.57 \times 10^6 \\ 1.50 \times 10^3 & -4.13 \times 10^4 & 1.64 \times 10^{13} & 5.41 \times 10^6 \\ 1.50 \times 10^3 & -4.13 \times 10^4 & 1.64 \times 10^{13} & 5.41 \times 10^6 \end{pmatrix}. \quad (3.35)$$

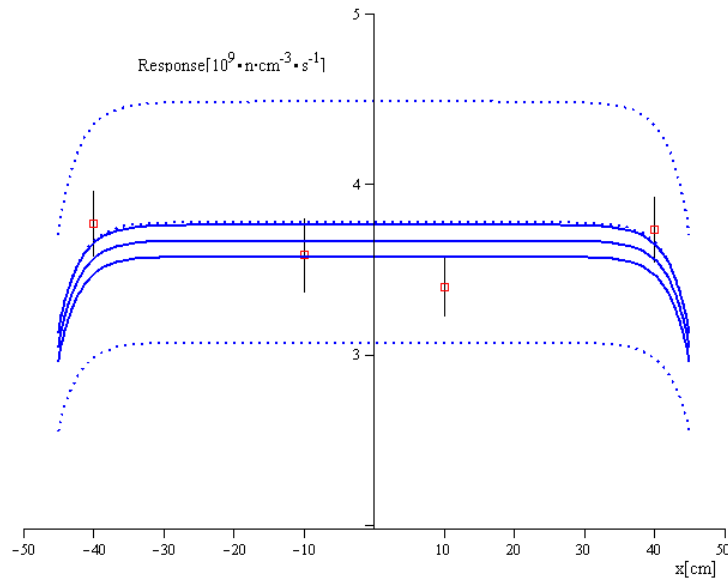


Figure 3.1: Four precise consistent precise measurements ($\chi^2 = 1.21$)

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 $z=a$ [cm], placed in air and characterized by a total interaction coefficient μ [cm⁻¹]. The slab contains a

uniformly distributed source of strength Q [photons / cm³ sec] 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.,

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

$$\psi(0, \omega) = 0. \quad (4.2)$$

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

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

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 Σ_d [cm⁻¹], 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

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

where the exponential-integral function is defined as

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

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.: $Q = 1[\text{photons} / \text{cm}^3 \text{sec}]$, $\Sigma_d = 1[\text{cm}^{-1}]$, $\mu = 1[\text{cm}^{-1}]$. If the detector were perfect and if its response $r(\mu a)$ were the consequence of an exactly-known deterministic counting process, Eq. (4) could be "inverted" to obtain the slab's optical thickness (μa) by solving deterministically the following nonlinear equation:

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

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

$$\frac{dE_2(x)}{dx} = -E_1(x) < 0, \quad 0 < x < \infty. \quad (4.7)$$

The result in Eq.(4.7) indicates that $E_2(x)$ 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 $E_2(x)$ decreases monotonically, at an increasingly slower rate, as x increases. Since $E_2(0) = 1$ and $E_2(x) \xrightarrow{x \rightarrow \infty} 0$, it follows that $E_2(x)$ will take on at most once each value in the interval $1 \geq E_2(x) = C > 0$ as x increases monotonically in the interval $0 \leq x < \infty$. Hence, despite the fact that the axis $x=0$ is asymptotically tangent to $E_2(x)$ in the limit when $x \rightarrow \infty$, Eq.(4.6) admits just a *single real-valued root*. Consequently, for each value of $r(\mu a)$, 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.,:

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

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

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

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

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

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

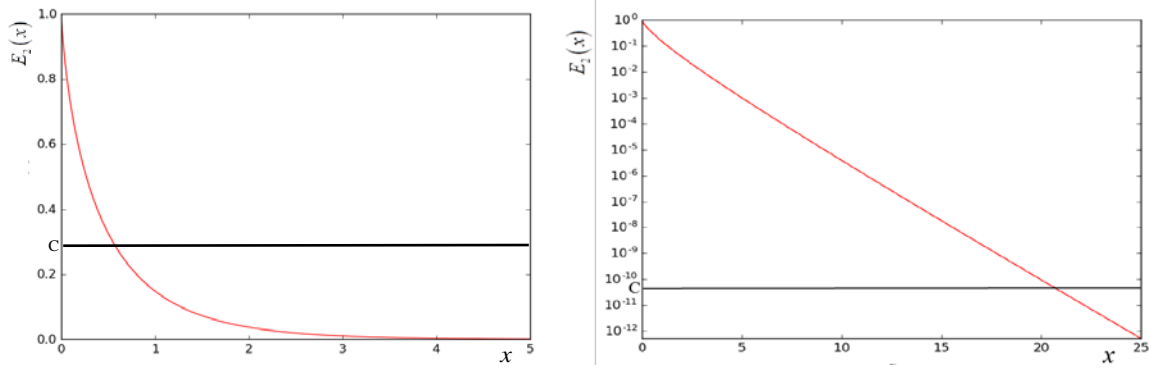


Figure 4.1. Location of the (unique) real root of Eq. (6): Left: linear-linear scale; Right: log-linear scale.

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 $r_{\text{exp}}^{(k)}$, 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

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

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:

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

where, for the slab considered in this Section,

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

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., $Q = 1[\text{photons} / \text{cm}^3 \text{sec}]$, $\Sigma_d = 1[\text{cm}^{-1}]$, $\mu = 1[\text{cm}^{-1}]$. A direct attempt to determine the slab's optical thickness would be by plotting the difference

$$\delta \triangleq (r_{\text{model}} - r_{\text{exp}}), \quad (4.15)$$

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 $\delta \triangleq (r_{\text{model}} - r_{\text{exp}})$ 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 $\delta \triangleq (r_{\text{model}} - r_{\text{exp}})$ 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., $\text{std.dev}(r_{\text{exp}}) = (0.01)r_{\text{exp}}$]. 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 $\delta \triangleq (r_{\text{model}} - r_{\text{exp}})$ are clearly identified 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.

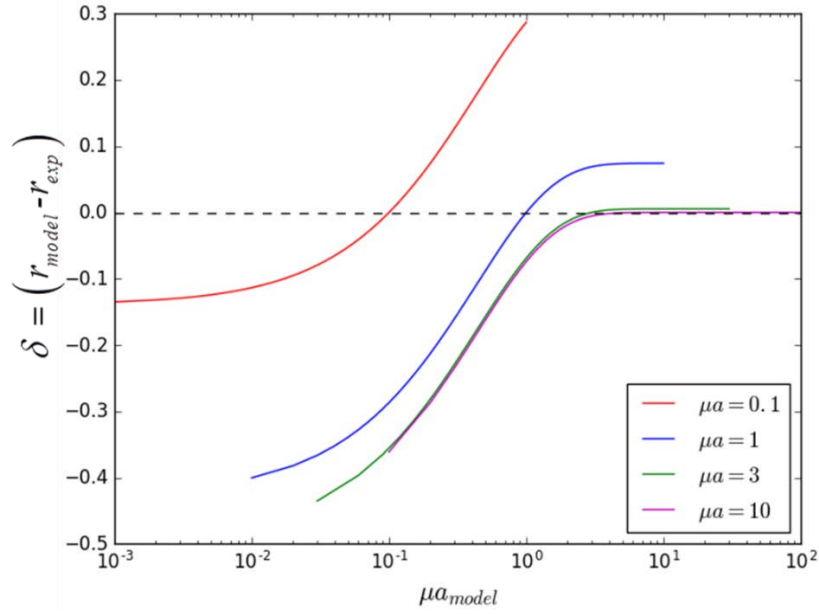


Figure 4.2: Variation of the difference between the computed detector response, r_{model} , and a measured (normally distributed, with a relative standard deviation of 1%) detector response, r_{exp} , as a function of the model’s optical thickness (μa_{model}).

Applying various minimization procedures, the value $(\mu a)_{\text{min}}$ which yields the minimum value, χ_{min}^2 , of χ^2 is considered to be the slab’s optical thickness. Mattingly (2015) has plotted the quantity $\delta^2 \triangleq (r_{\text{model}} - r_{\text{exp}}^{(k)})^2$ 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 $r_{\text{exp}}^{(k)}$, $k=1, \dots, 10$, 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., $\text{std.dev}(r_{\text{exp}}) = (0.01)r_{\text{exp}}$].

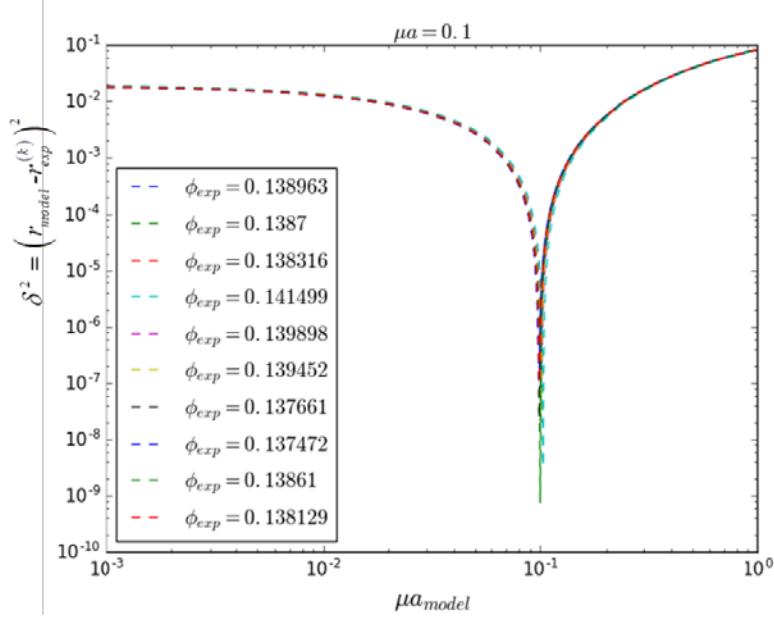


Figure 4.3: Variation of $\delta^2 \triangleq \left(r_{\text{model}} - r_{\text{exp}}^{(k)} \right)^2$ as a function of the model's optical thickness (μa_{model}) for a slab of actual optical thickness $\mu a = 0.1$, for measurements with a relative standard deviation of 1%.

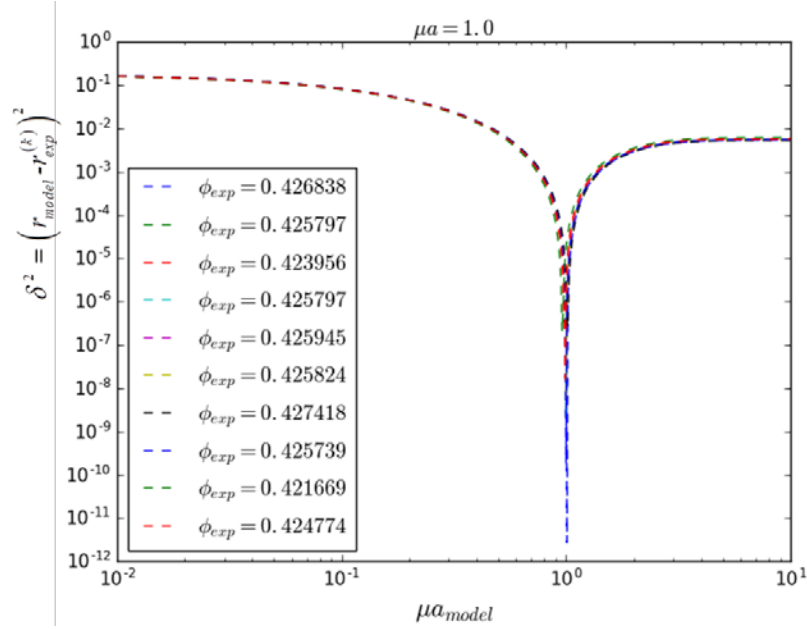


Figure 4.4: Variation of $\delta^2 \triangleq (r_{\text{model}} - r_{\text{exp}}^{(k)})^2$ as a function of the model's optical thickness (μa_{model}) for a slab of actual optical thickness $\mu a = 1.0$, for measurements with a relative standard deviation of 1%.

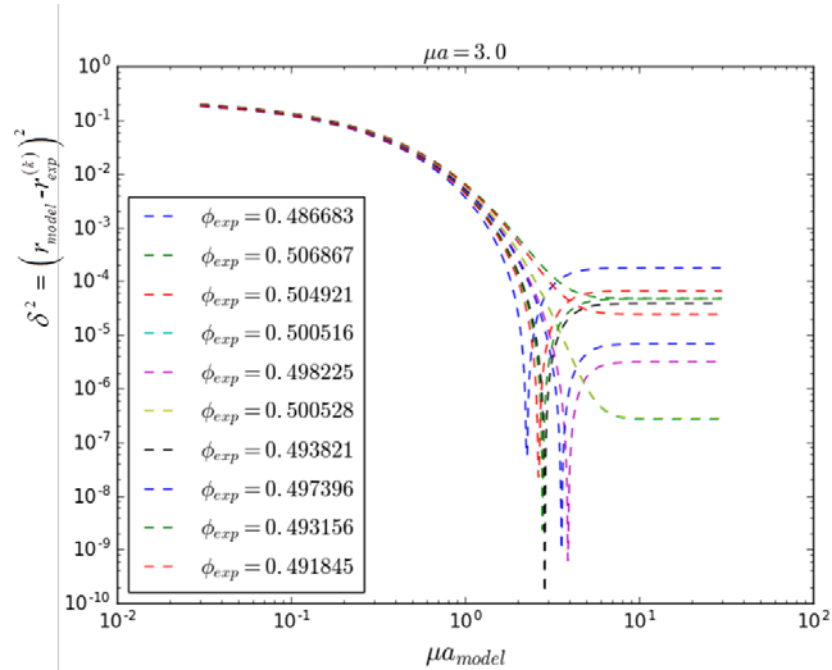


Figure 4.5: Variation of $\delta^2 \triangleq (r_{\text{model}} - r_{\text{exp}}^{(k)})^2$ as a function of the model's optical thickness (μa_{model}) for a slab of actual optical thickness $\mu a = 3.0$, for measurements with a relative standard deviation of 1%.

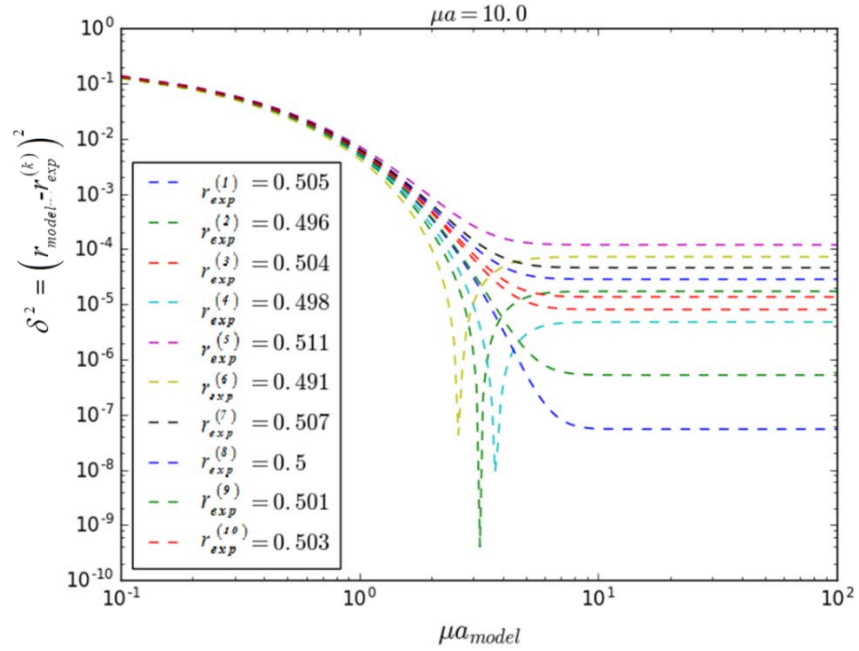


Figure 4.6: Variation of $\delta^2 \triangleq \left(r_{\text{model}} - r_{\text{exp}}^{(k)} \right)^2$ as a function of the model's optical thickness (μa_{model}) for a slab of actual optical thickness $\mu a = 10.0$, for measurements with a relative standard deviation of 1%.

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 $\delta^2 \triangleq (r_{\text{model}} - r_{\text{exp}}^{(k)})^2$ 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 $\delta^2 \triangleq (r_{\text{model}} - r_{\text{exp}}^{(k)})^2$, 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.

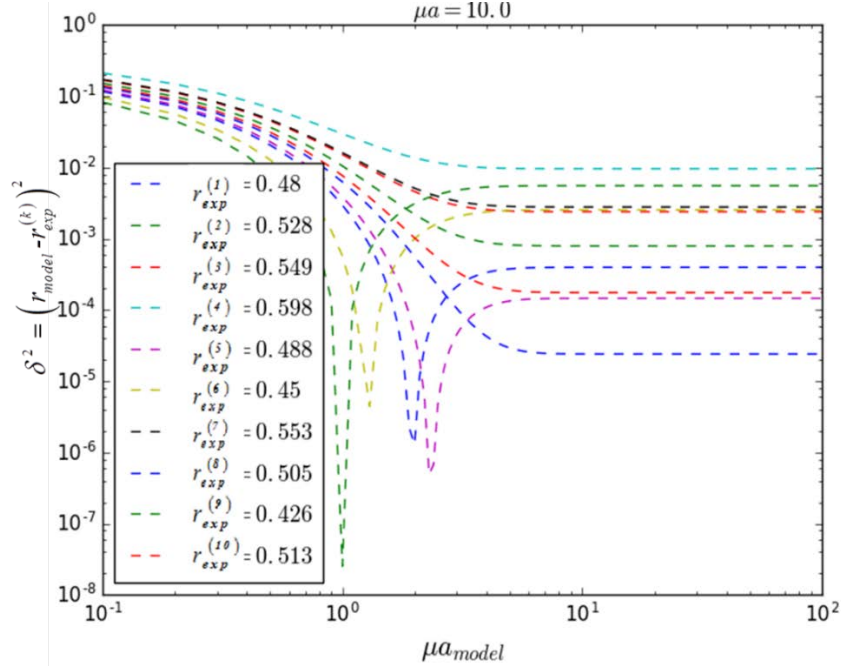


Figure 4.7: Variation of $\delta^2 \triangleq (r_{\text{model}} - r_{\text{exp}}^{(k)})^2$ as a function of the model's optical thickness (μa_{model}) for a slab of actual optical thickness $\mu a = 10.0$, for measurements with a relative standard deviation of 10%.

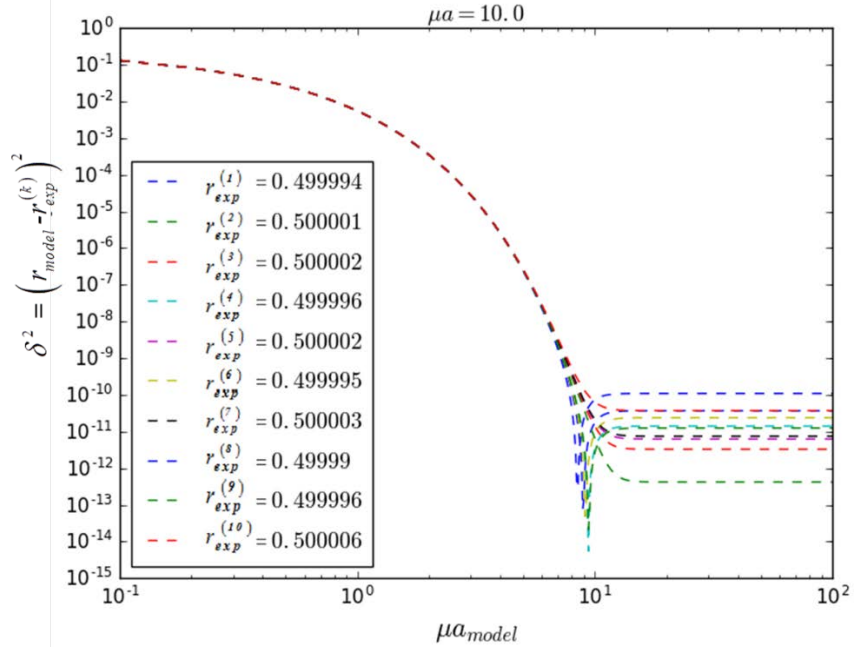


Figure 4.8: Variation of $\delta^2 \triangleq (r_{\text{model}} - r_{\text{exp}}^{(k)})^2$ as a function of the model's optical thickness (μa_{model}) for a slab of actual optical thickness $\mu a = 10.0$, for extremely precise measurements with a relative standard deviation of 0.001%.

Comparing the results depicted in Figure 4.7 with those depicted in Figure 4.6 shows that the quantity $\delta^2 \triangleq (r_{\text{model}} - r_{\text{exp}}^{(k)})^2$ 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 $\delta^2 \triangleq (r_{\text{model}} - r_{\text{exp}}^{(k)})^2$ 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 $(\mu a)_{\text{min}}$. 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 are reproduced below:

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

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

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

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

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

where

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

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

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

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

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

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 $r_m^{(k)} = \text{random normal} \left[r^{exact}, sd \left(r_m^{(exact)} \right) \right]$, $k = 1, \dots, K_n$, 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 $r_m^{(k)} = \text{random normal} \left[r^{exact}, \beta r^{exact} \right]$, $k = 1, \dots, K_0$.
2. Compute the initial “sample average”: $r_{m,ave}^{(K_0)} = \frac{1}{K_0} \sum_{k=1}^{K_0} r_m^{(k)}$.
3. Compute the initial “measurement variance”: $C_{rr}^{(K_0)} = \frac{1}{K_0 - 1} \sum_{k=1}^{K_0} \left[r_m^{(k)} - r_{m,ave}^{(K_0)} \right]^2$.
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 $E_2[\alpha^{(1)}] = 1 - \frac{2\mu r_{m,ave}^{(K_0)}}{Q\Sigma_d}$.
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: $S^{(1)} = \frac{Q\Sigma_d}{2\mu} E_1[\alpha^{(1)}]$.
7. Define the “initial parameter standard deviation”: $sd(\alpha^{(1)}) = \gamma\alpha^{(1)}$ and the variance $C_{\alpha\alpha}^{(1)} = [\gamma\alpha^{(1)}]^2$. 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”: $C_{ar}^{(1)} = C_{\alpha\alpha}^{(1)} S^{(1)\dagger} [D_{rr}^{(1)}]^{-1} C_{rr}^{(K_0)}$.
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 $n = 1, \dots, N$ sets of measurements, $r_m^{(k)}$, $k = 1, \dots, K_n$, , by sampling from the normal distribution $r_m^{(k)} = random\ normal[r^{exact}, sd(r_m^{(exact)})]$.
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”: $C_{rr}^{(K_n)} = \frac{1}{K_1 - 1} \sum_{k=1}^{K_1} [r_m^{(k)} - r_{m,ave}^{(K_n)}]^2$;
- 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:

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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 $\left| \frac{r_{comp}^{(n+1)} - r_{pred}^{(n+1)}}{r_{comp}^{(n+1)}} \right| < \varepsilon$. 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% ($\beta = 10^{-5}$) will be considered as being “very precise.”

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

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

The exact detector response stemming from a slab of optical thickness $\mu\alpha = 0.1$ is $r_{exact} = 1.387275 \times 10^{-1}$ photons/cm²sec, 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/cm²sec, and the exact optical slab thickness within a standard deviation of 8.89×10^{-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 $\mu\alpha = 0.1$ after successively assimilating 2 batches of 100 imprecise experiments ($\beta = 10^{-1}$)

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

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

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 $r_{exact} = 1.387275 \times 10^{-1}$ photons/cm²sec. Using these very precise measurements, the PM-CMPS methodology predicts the exact response value within a standard deviation of 1.3×10^{-6} and the slab thickness within a standard deviation of 2×10^{-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 ($\beta = 10^{-5}$)

$\mu a = 0.1 ; \beta = 10^{-5} ; \varepsilon = 10^{-8} ; K_n = 100 ; \text{ Measured response} = \text{Normal}(r_{exact}, \beta r_{exact})$					
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* ($\beta = 10^{-1}$)

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 ($r_{exact} = 4.257522 \times 10^{-1}$ photons/cm²sec). 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 2.85×10^{-2} , and the exact optical slab thickness is predicted within a standard deviation of 9.65×10^{-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 $\beta = 10^{-1}$.

$\mu a = 1 ; \beta = 10^{-1} ; \varepsilon = 10^{-3} ; K_n = 100 ; \text{ Measured response} = \text{Normal} (r_{\text{exact}}, \beta r_{\text{exact}})$					
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 ($\beta = 10^{-5}$)

Consider a set $K_1 = 100$ of precise measurements (relative standard deviation $\beta = 10^{-5}$) drawn from the same random normal distribution, with $r_{\text{exact}} = 4.257522 \times 10^{-1}$ 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 $\beta = 10^{-5}$.

$\mu a = 1 ; \beta = 10^{-5} ; \varepsilon = 10^{-8} ; K_n = 100 ; \text{ Measured response} = \text{Normal} (r_{\text{exact}}, \beta r_{\text{exact}})$					
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 ($\beta = 10^{-1}$)

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 ($r_{exact} = 4.94679 \times 10^{-1}$ 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 3.25×10^{-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	5.010051×10^{-1}	4.967511×10^{-1}	3.255519×10^{-2}	2.739635	2.736269×10^{-1}
2	4.999093×10^{-1}	4.926791×10^{-1}	2.490237×10^{-3}	2.741372	2.733279×10^{-1}
		Exact Response	Exact Response SD	Exact Parameter	
		4.94679×10^{-1}	4.94679×10^{-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 $r_{exact} = 4.94679 \times 10^{-1}$ photons/cm²sec as the distribution's mean. Using these precise measurements, the PM-CMPS methodology predicts the response within a standard deviation of 4.65×10^{-6} photons/cm²sec, 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 $\beta = 10^{-5}$.

$\mu a = 3; \beta = 10^{-5}; \varepsilon = 10^{-8}; K_n = 100; \text{ Measured response} = \text{Normal}(r_{exact}, \beta r_{exact})$					
n	Experimental Response Mean Value	Predicted Response	Predicted Response SD	Predicted Parameter	Predicted Parameter SD
1	4.946797×10^{-1}	4.946797×10^{-1}	4.647000×10^{-6}	3.000097	9.973716×10^{-5}
		Exact Response	Exact Response SD	Exact Parameter	
		4.94679×10^{-1}	4.94679×10^{-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 ($r_{exact} = 4.999482 \times 10^{-1}$ photons/cm²sec). 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 9.41×10^{-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 = $Normal(r_{exact}, \beta r_{exact})$				
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 ($\beta = 10^{-5}$)

Consider a set $K_1 = 100$ of precise measurements (relative standard deviation $\beta = 10^{-5}$) drawn from the same random normal distribution, with $r_{exact} = 4.999482 \times 10^{-1}$ photons/cm²sec photons/cm²se 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 4.66×10^{-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 = $Normal(r_{exact}, \beta r_{exact})$					
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 ($\beta = 10^{-1}$)

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

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

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

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

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 Parameter Value
1	4.999940×10^{-1}	4.999908×10^{-1}	1.697100×10^{-5}	7.959722
$\mu a = 10 ; \beta = 10^{-4} ; \varepsilon = 10^{-5} ; K_n = 10 ;$				
1	5.000060×10^{-1}	4.999949×10^{-1}	2.146285×10^{-5}	8.334934
$\mu a = 10 ; \beta = 10^{-4} ; \varepsilon = 10^{-5} ; K_n = 50$				
1	4.999974×10^{-1}	4.999958×10^{-1}	3.250068×10^{-5}	9.056938
$\mu a = 10 ; \beta = 10^{-4} ; \varepsilon = 10^{-5} ; K_n = 100 ;$				
1	5.000045×10^{-1}	5.000002×10^{-1}	3.294413×10^{-5}	9.338401
		Exact Response Value	Exact Response SD	Exact Parameter Value
		4.999981×10^{-1}	4.999981×10^{-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	4.999977×10^{-1}	4.999975×10^{-1}	1.796069×10^{-6}	9.563645	6.465910×10^{-1}
$\mu a = 10 ; \beta = 10^{-5} ; \varepsilon = 10^{-8} ; K_n = 10 ;$					
1	4.999989×10^{-1}	4.999981×10^{-1}	2.566375×10^{-6}	9.786590	7.628210×10^{-1}
$\mu a = 10 ; \beta = 10^{-5} ; \varepsilon = 10^{-8} ; K_n = 50 ;$					
1	4.999980×10^{-1}	4.999979×10^{-1}	3.486978×10^{-6}	9.861132	9.236508×10^{-1}
2	4.999986×10^{-1}	4.999981×10^{-1}	1.680819×10^{-6}	9.987424	7.737391×10^{-1}
$\mu a = 10 ; \beta = 10^{-5} ; \varepsilon = 10^{-8} ; K_n = 100 ;$					
1	4.999987×10^{-1}	4.999983×10^{-1}	3.466796×10^{-6}	9.945714	9.359559×10^{-1}
2	4.999986×10^{-1}	4.999981×10^{-1}	1.927486×10^{-6}	9.983171	8.739583×10^{-1}
		Exact Response	Exact Response SD	Exact Parameter	
		4.999981×10^{-1}	4.999981×10^{-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 underpredicts 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
		$4.99999909 \times 10^{-1}$	$4.99999909 \times 10^{-8}$	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.

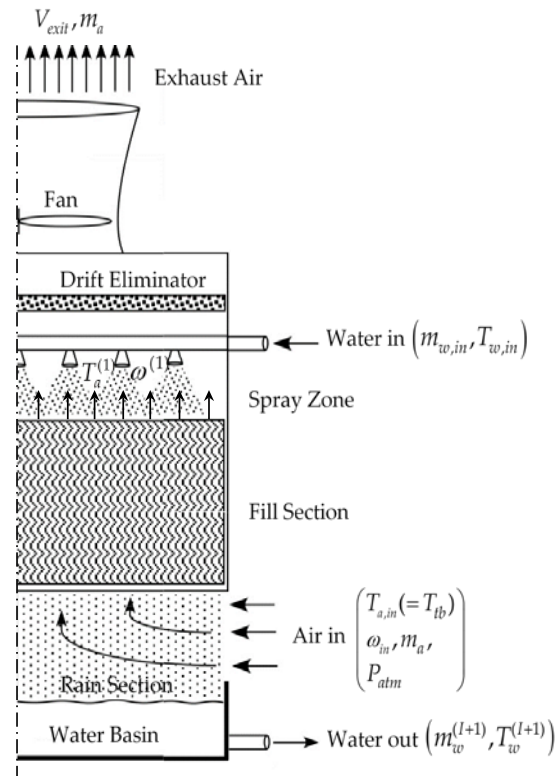


Figure 5.1. Flow through a counter-flow cooling tower.

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.

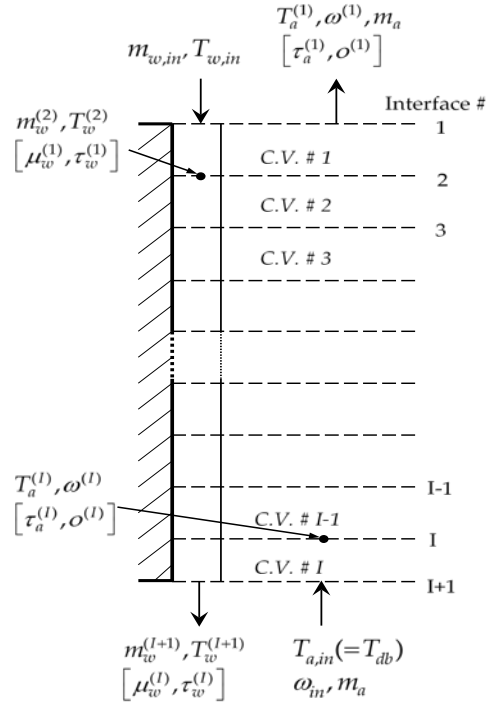


Figure 5.2. Control volumes ($i = 1, \dots, I$) comprising the counter-flow cooling tower, together with the symbols denoting the forward state functions ($m_w^{(i)}, T_w^{(i)}, T_a^{(i)}, \omega^{(i)}$, $i = 1, \dots, I$) and the adjoint state functions ($\mu_w^{(i)}, \tau_w^{(i)}, \tau_a^{(i)}, \mathcal{O}^{(i)}$;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

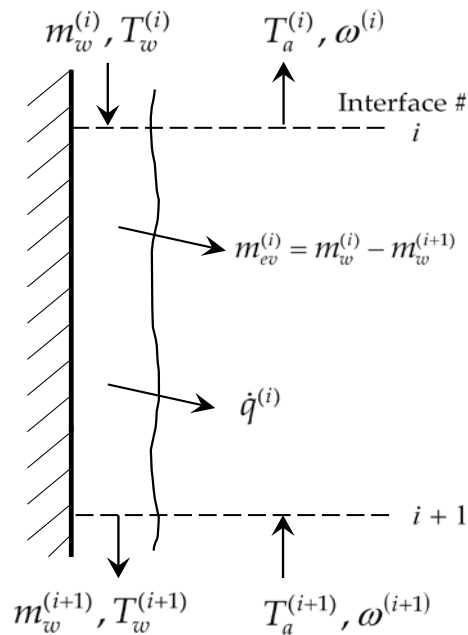


Figure 5.3. Heat and mass transfer between falling water film and rising air in a typical control volume of the cooling tower's fill section.

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 $m_w^{(i)}$ ($i = 2, \dots, 50$), 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 $T_w^{(i)}$ ($i = 2, \dots, 50$), 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 $T_a^{(i)}$ ($i = 1, \dots, 49$), 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 $\omega^{(i)}$ ($i = 1, \dots, 49$), 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:

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

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

A. Liquid continuity equations:

(i) *Control Volume $i=1$:*

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

(ii) *Control Volumes $i=2, \dots, I-1$:*

$$N_1^{(i)}(\mathbf{m}_w, \mathbf{T}_w, \mathbf{T}_a, \omega; \mathbf{a}) \triangleq m_w^{(i+1)} - m_w^{(i)} + \frac{M(m_a, \mathbf{a})}{\bar{R}} \left[\frac{P_{vs}^{(i+1)}(T_w^{(i+1)}, \mathbf{a})}{T_w^{(i+1)}} - \frac{\omega^{(i)} P_{atm}}{T_a^{(i)}(0.622 + \omega^{(i)})} \right] = 0; \quad (5.3)$$

(iii) Control Volume $i=I$:

$$N_1^{(I)}(\mathbf{m}_w, \mathbf{T}_w, \mathbf{T}_a, \omega; \mathbf{a}) \triangleq m_w^{(I+1)} - m_w^{(I)} + \frac{M(m_a, \mathbf{a})}{\bar{R}} \left[\frac{P_{vs}^{(I+1)}(T_w^{(I+1)}, \mathbf{a})}{T_w^{(I+1)}} - \frac{\omega^{(I)} P_{atm}}{T_a^{(I)}(0.622 + \omega^{(I)})} \right] = 0; \quad (5.4)$$

B. Liquid energy balance equations:

(i) Control Volume $i=1$:

$$N_2^{(1)}(\mathbf{m}_w, \mathbf{T}_w, \mathbf{T}_a, \omega; \mathbf{a}) \triangleq m_{w,in} h_f(T_{w,in}, \mathbf{a}) - (T_w^{(2)} - T_a^{(1)})H(m_a, \mathbf{a}) - m_w^{(2)} h_f^{(2)}(T_w^{(2)}, \mathbf{a}) - (m_{w,in} - m_w^{(2)})h_{g,w}^{(2)}(T_w^{(2)}, \mathbf{a}) = 0; \quad (5.5)$$

(ii) Control Volumes $i=2, \dots, I-1$:

$$N_2^{(i)}(\mathbf{m}_w, \mathbf{T}_w, \mathbf{T}_a, \omega; \mathbf{a}) \triangleq m_w^{(i)} h_f^{(i)}(T_w^{(i)}, \mathbf{a}) - (T_w^{(i+1)} - T_a^{(i)})H(m_a, \mathbf{a}) - m_w^{(i+1)} h_f^{(i+1)}(T_w^{(i+1)}, \mathbf{a}) - (m_w^{(i)} - m_w^{(i+1)})h_{g,w}^{(i+1)}(T_w^{(i+1)}, \mathbf{a}) = 0; \quad (5.6)$$

(iii) Control Volume $i=I$:

$$N_2^{(I)}(\mathbf{m}_w, \mathbf{T}_w, \mathbf{T}_a, \omega; \mathbf{a}) \triangleq m_w^{(I)} h_f^{(I)}(T_w^{(I)}, \mathbf{a}) - (T_w^{(I+1)} - T_a^{(I)})H(m_a, \mathbf{a}) - m_w^{(I+1)} h_f^{(I+1)}(T_w^{(I+1)}, \mathbf{a}) - (m_w^{(I)} - m_w^{(I+1)})h_{g,w}^{(I+1)}(T_w^{(I+1)}, \mathbf{a}) = 0; \quad (5.7)$$

C. Water vapor continuity equations:

(i) Control Volume $i=1$:

$$N_3^{(1)}(\mathbf{m}_w, \mathbf{T}_w, \mathbf{T}_a, \omega; \mathbf{a}) \triangleq \omega^{(2)} - \omega^{(1)} + \frac{m_{w,in} - m_w^{(2)}}{|m_a|} = 0; \quad (5.8)$$

(ii) Control Volumes $i=2, \dots, I-1$:

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

(iii) Control Volume $i=I$:

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

D. The air/water vapor energy balance equations:

(i) Control Volume $i=1$:

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

(ii) Control Volumes $i=2, \dots, I-1$:

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

(iii) Control Volume $i=I$:

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

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.,

$$\boldsymbol{\alpha} \triangleq (\alpha_1, \dots, \alpha_{N_\alpha}), \quad (5.14)$$

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

$$\mathbf{N}(\mathbf{u}) = \mathbf{0}, \quad (5.15)$$

where the following definitions are used:

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

- (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 $n = 0, 1, 2, \dots$, iterate over the following steps until convergence:
- (d) Start inner iteration loop: for $m = 1, 2, \dots$, 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 $\delta\mathbf{u}$:

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

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:

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

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

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

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

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

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

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

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

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

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

where

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

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

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

where

$$c_1^{i,i} \triangleq \frac{M(m_a, \boldsymbol{\alpha})}{\bar{R}} \frac{\omega^{(i)} P_{am}}{[T_a^{(i)}]^2 (0.622 + \omega^{(i)})}. \quad (5.27)$$

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

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

Where

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

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

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

Where

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

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

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

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

Where

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

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

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

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

where

$$c_2^{i,i} \triangleq H(m_a, \mathbf{a}); \quad i = 1, \dots, I; \quad j = i. \quad (5.37)$$

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

$$\mathbf{D}_2 \triangleq [d_2^{i,j}]_{I \times I} = \mathbf{0}. \quad (5.38)$$

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

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

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

$$\mathbf{B}_3 \triangleq [b_3^{i,j}]_{I \times I} = \mathbf{0}. \quad (5.40)$$

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

$$\mathbf{C}_3 \triangleq [c_3^{i,j}]_{I \times I} = \mathbf{0}. \quad (5.41)$$

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

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

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

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

where

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

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

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

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

where

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

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

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

where

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

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

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

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

where

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

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

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 ,

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

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:

$$\mathbf{u}_{n+1} = \mathbf{u}_n + \delta \mathbf{u}, \quad (5.55)$$

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:

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

(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 $\mathbf{m}_w \triangleq [m_w^{(2)}, \dots, m_w^{(I+1)}]^\dagger$ of water mass flow rates at the exit of each control volume i , ($i = 1, \dots, 49$);
- (b) the vector $\mathbf{T}_w \triangleq [T_w^{(2)}, \dots, T_w^{(I+1)}]^\dagger$ of water temperatures at the exit of each control volume i , ($i = 1, \dots, 49$);
- (c) the vector $\mathbf{T}_a \triangleq [T_a^{(1)}, \dots, T_a^{(I)}]^\dagger$ of air temperatures at the exit of each control volume i , ($i = 1, \dots, 49$);
- (d) the vector $\mathbf{RH} \triangleq [RH^{(1)}, \dots, RH^{(I)}]^\dagger$, having as components the air relative humidity at the exit of each control volume i , ($i = 1, \dots, 49$).

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 :

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

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.

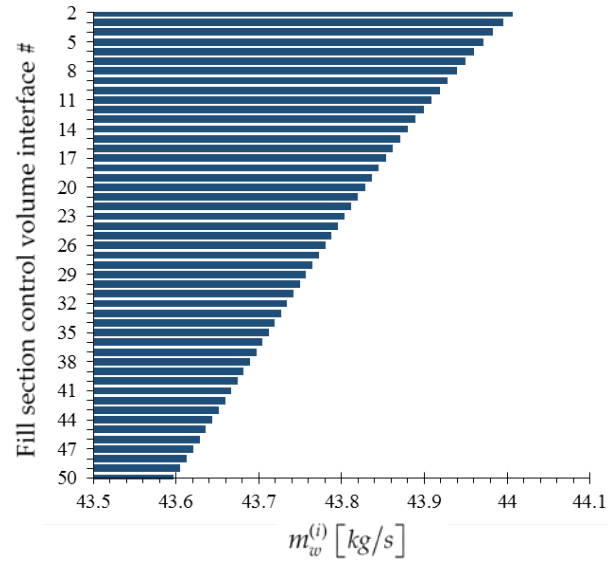


Figure 5.4. Bar plot of the water mass flow rates $m_w^{(i)}$, ($i = 2, \dots, 50$), at the exit of each control volume along the height of the fill section of the cooling tower.

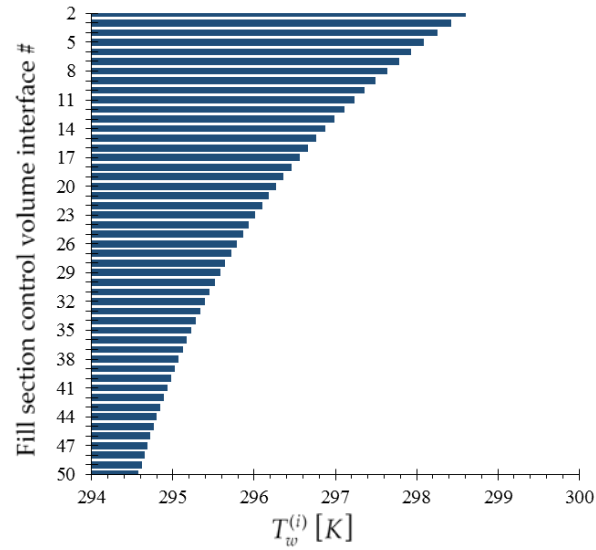


Figure 5.5. Bar plot of the water temperatures $T_w^{(i)}$, ($i = 2, \dots, 50$), at the exit of each control volume along the height of the fill section of the cooling tower.

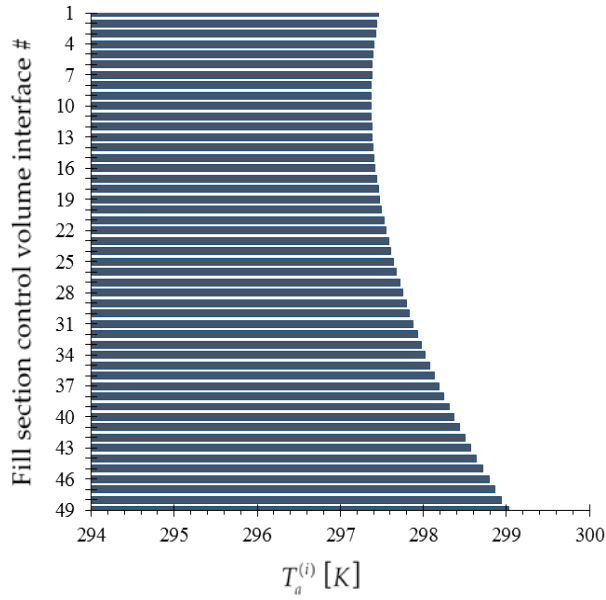


Figure 5.6. Bar plot of the air temperatures $T_a^{(i)}$, ($i = 1, \dots, 49$), at the exit of each control volume along the height of the fill section of the cooling tower.

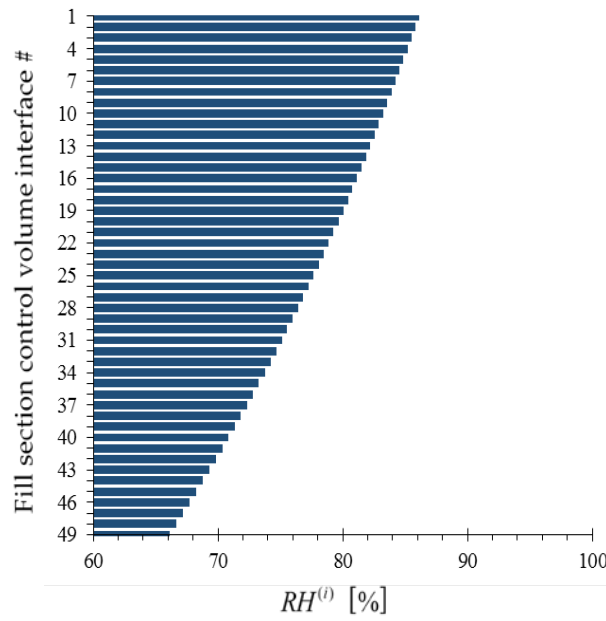


Figure 5.7. Bar plot of the air relative humidity $RH^{(i)}$, ($i = 1, \dots, 49$), at the exit of each control volume along the height of the fill section of the cooling tower.

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 $R(\mathbf{m}_w, \mathbf{T}_w, \mathbf{T}_a, \boldsymbol{\omega}; \boldsymbol{\alpha})$, 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 $\delta \boldsymbol{\alpha} \triangleq (\delta \alpha_1, \dots, \delta \alpha_{N_\alpha})$ 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 $DR(\mathbf{m}_w^0, \mathbf{T}_w^0, \mathbf{T}_a^0, \boldsymbol{\omega}^0; \boldsymbol{\alpha}^0; \delta \mathbf{m}_w, \delta \mathbf{T}_w, \delta \mathbf{T}_a, \delta \boldsymbol{\omega}; \delta \boldsymbol{\alpha})$, which is defined as follows:

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

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

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

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

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

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

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:

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

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

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

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

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

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 $[\boldsymbol{\mu}_w, \boldsymbol{\tau}_w, \boldsymbol{\tau}_a, \mathbf{o}]^\dagger$, having the same structure as the vector $\mathbf{u} \triangleq (\mathbf{m}_w, \mathbf{T}_w, \mathbf{T}_a, \boldsymbol{\omega})^\dagger$, transposing the resulting scalar equation and subsequently using Eq. (5.60). By requiring the vector $[\boldsymbol{\mu}_w, \boldsymbol{\tau}_w, \boldsymbol{\tau}_a, \mathbf{o}]^\dagger$ to satisfy the following adjoint sensitivity system:

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

the “indirect effect” term can be expressed in the following form

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

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 $[\boldsymbol{\mu}_w, \boldsymbol{\tau}_w, \boldsymbol{\tau}_a, \mathbf{o}]^\dagger$. 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:

$$[\boldsymbol{\mu}_w^{(i)}] = \frac{[R]}{[N_1]}; \quad [\boldsymbol{\tau}_w^{(i)}] = \frac{[R]}{[N_2]}; \quad [\boldsymbol{\tau}_a^{(i)}] = \frac{[R]}{[N_3]}; \quad [\mathbf{o}^{(i)}] = \frac{[R]}{[N_4]} \quad (5.67)$$

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

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

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	$[\mu_w^{(i)}]$	$[\tau_w^{(i)}]$	$[\tau_a^{(i)}]$	$[o^{(i)}]$
$R \triangleq T_a^{(1)}$	$K/(kg/s)$	$K/(J/s)$	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 $RS(\alpha_i) \triangleq [dR(\alpha_i)/d\alpha_i][\alpha_i/R(\alpha_i)]$. 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	Prandtl 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_{dp} 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, α_0	25.5943	-0.1949	0.04
5	$P_{vs}(T)$ parameters, α_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, α_{1f}	4186.51	0.0082	0.04
10	$D_{av}(T_{db})$ parameter, $\alpha_{1,dav}$	2.653	0.0071	0.11
11	Enthalpy $h_g(T)$ parameter, α_{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_{lsa}	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): $\pm 2.5\%$ for relative humidity from 10 to 90%; between $\pm 2.5\%$ and $\pm 3.5\%$ for relative humidity from 90% to 95%; and $\pm 3.5\% \sim \pm 4.0\%$ 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 $\pm 4.5\%$ to $\pm 5.0\%$ for relative humidity from 95 to 100%.

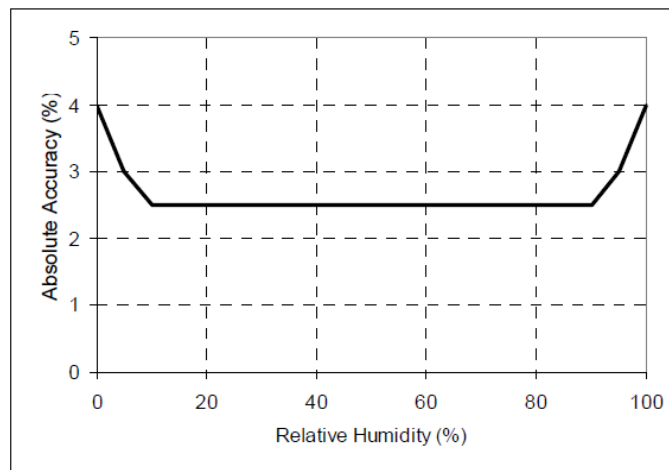


Figure 5.7: Humidity sensor accuracy plot (adopted from the specification of HOBO Pro v2).

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.

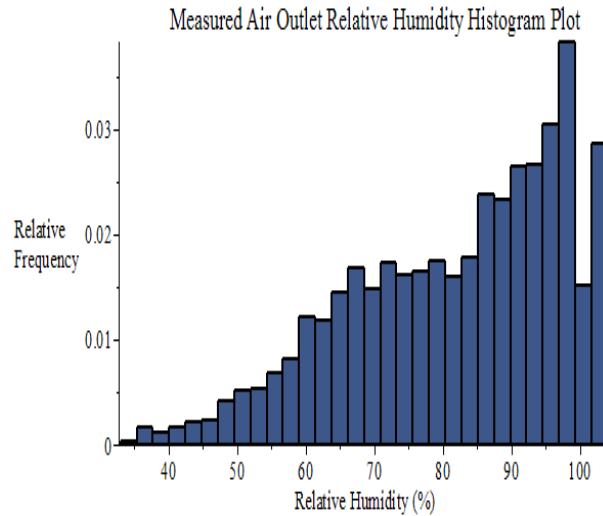


Figure 5.8: Histogram plot of the measured air outlet relative humidity, within the 7688 data sets collected by SRNL from F-Area cooling towers (unsaturated conditions).

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.

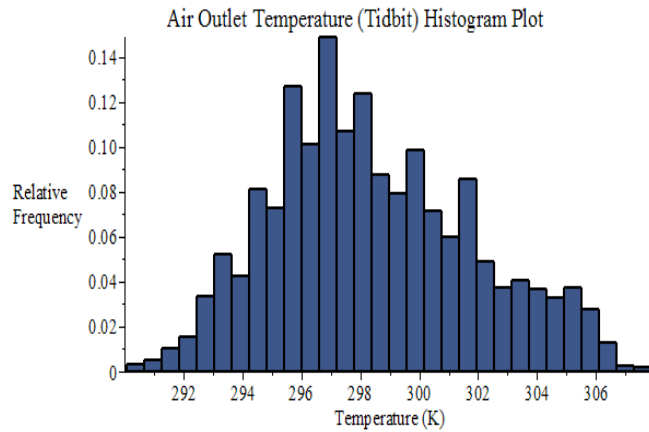


Figure 5.9. Histogram plot of the air outlet temperature measured using “Tidbit” sensors, within the 7688 data sets collected by SRNL from F-Area cooling towers (unsaturated conditions).

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

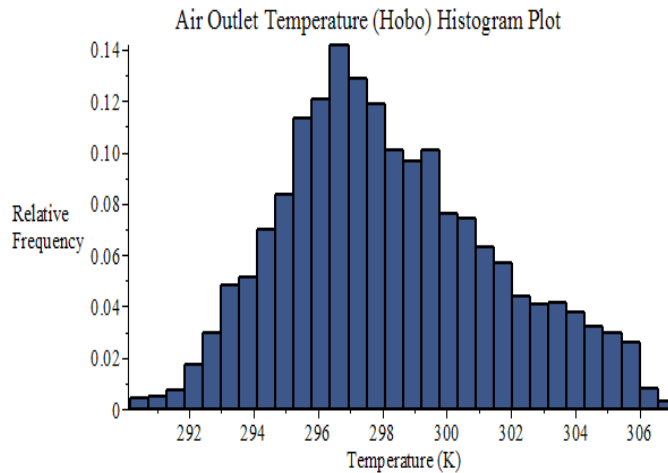


Figure 5.10. Histogram plot of the air outlet temperature measured using “Hobo” sensors, within the 7688 data sets collected by SRNL from F-Area cooling towers (unsaturated conditions).

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

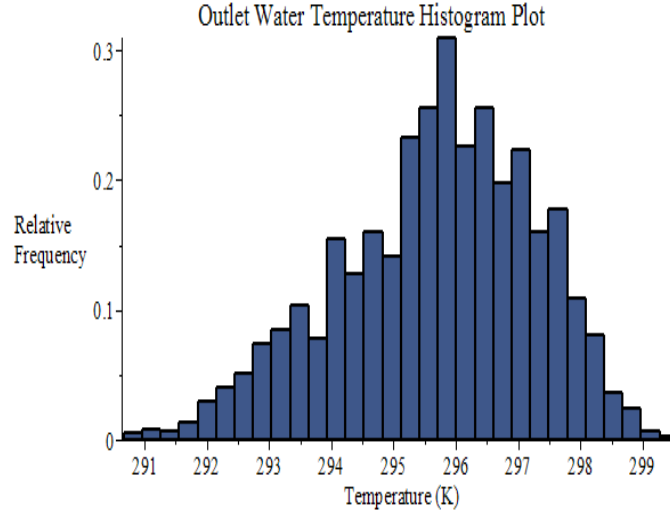


Figure 5.11. Histogram plot of water outlet temperature measurements, within the 7688 data sets collected by SRNL from F-Area cooling towers (unsaturated conditions).

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 $Cov(T_{a,out(Tidbit)}, T_{a,out(Hobo)}, T_{w,out}^{meas}, RH_{out}^{meas})$:

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

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.

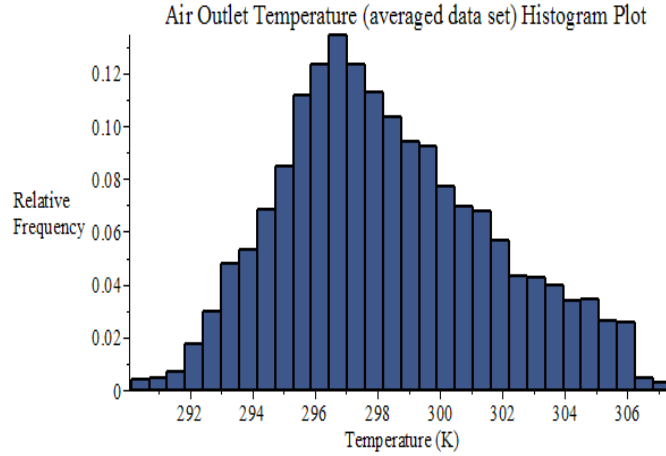


Figure 5.12. Histogram plot of air outlet temperatures

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:

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

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:

$$\sigma = \sqrt{\sigma_{statistic}^2 + \sigma_{sensor}^2}, \quad (5.71)$$

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 $Cov(T_{a,out}^{meas}, T_{w,out}^{meas}, RH_{out}^{meas})$. The final result thus obtained is

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

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

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

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.

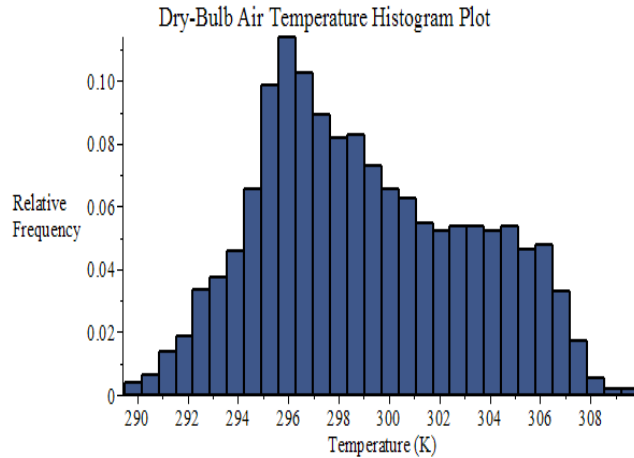


Figure 5.13. Histogram plot of dry-bulb air temperature data collected by SRNL from F-Area cooling towers (unsaturated conditions).

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

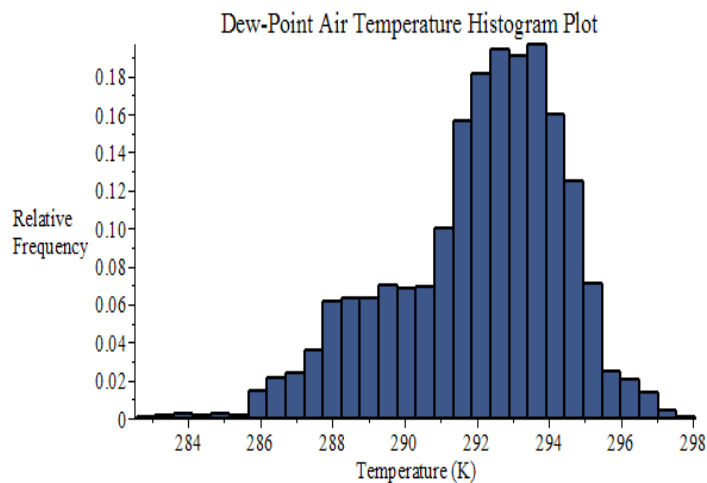


Figure 5.14. Histogram plot of dew-point air temperature data collected by SRNL from F-Area cooling towers (unsaturated conditions).

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

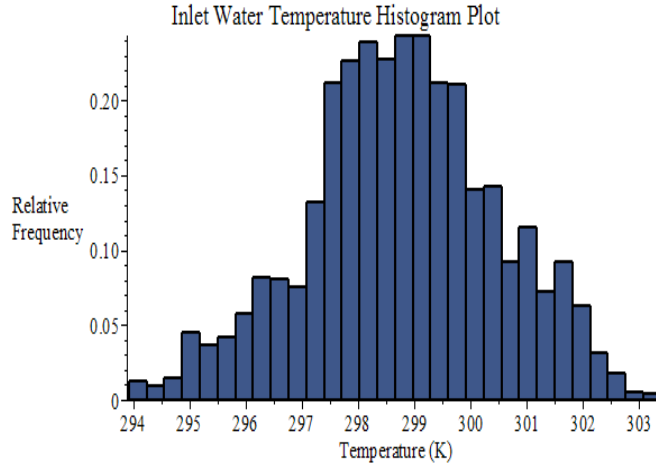


Figure 5.15. Histogram plot of inlet water temperature data collected by SRNL from F-Area cooling towers (unsaturated conditions).

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

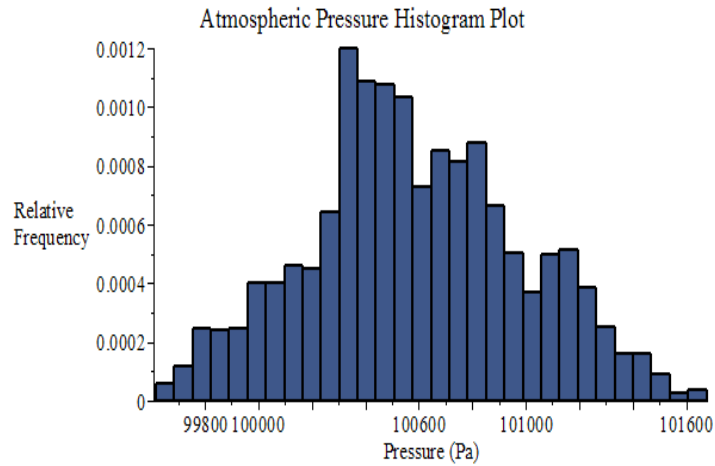


Figure 5.16. Histogram plot of atmospheric pressure data collected by SRNL from F-Area cooling towers (unsaturated conditions).

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:

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

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:

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

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, $T_{a,out}^{meas} \equiv [T_a^{(1)}]^{measured}$; the outlet water temperature, $T_{w,out}^{meas} \equiv [T_w^{(50)}]^{measured}$, and the outlet air relative humidity, $RH_{out}^{meas} \equiv [RH^{(1)}]^{measured}$) is given below:

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

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>i</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	$a_{0,f}$	-1143423.8	543	-1143423.7	543
23		$a_{1,f}$	4186.50768	1.8	4186.50818	1.8

24	$h_g(T)$	$a_{0,g}$	2005743.99	1046	2005743.80	1046
25	parameters	$a_{1,g}$	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>i</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_m	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} :

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

The best-estimate response-parameter correlation matrix, \mathbf{C}_{ar}^{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:

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

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 $\alpha_{49} \triangleq Re_d$.

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:

$$[C_{rr}^{comp}]^{be} = [S_{r\alpha}]^{be} [C_{\alpha\alpha}]^{be} [S_{r\alpha}^\dagger]^{be}. \quad (5.80)$$

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, $T_a^{(i)}$, $i = 2, \dots, I$, at the exit of each control volume; (ii) the water temperature, $T_w^{(i+1)}$, $i = 1, \dots, I - 1$, at the exit of each control volume; and (iii) the air relative humidity, $RH^{(i)}$, $i = 2, \dots, I$, 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, α^{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 $[\mathbf{S}_{ra}]^{pred}$ has been obtained for each of the responses $T_a^{(i)}$, $i = 2, \dots, I$, $T_w^{(i+1)}$, $i = 1, \dots, I - 1$, and $RH^{(i)}$, $i = 2, \dots, I$ 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 $Ta^{(i)}$, ($i = 1, \dots, 49$); water temperature $Tw^{(i)}$, ($i = 2, \dots, 50$); and air humidity $RH^{(i)}$, ($i = 1, \dots, 49$), respectively, along the height of the fill section of the cooling tower.

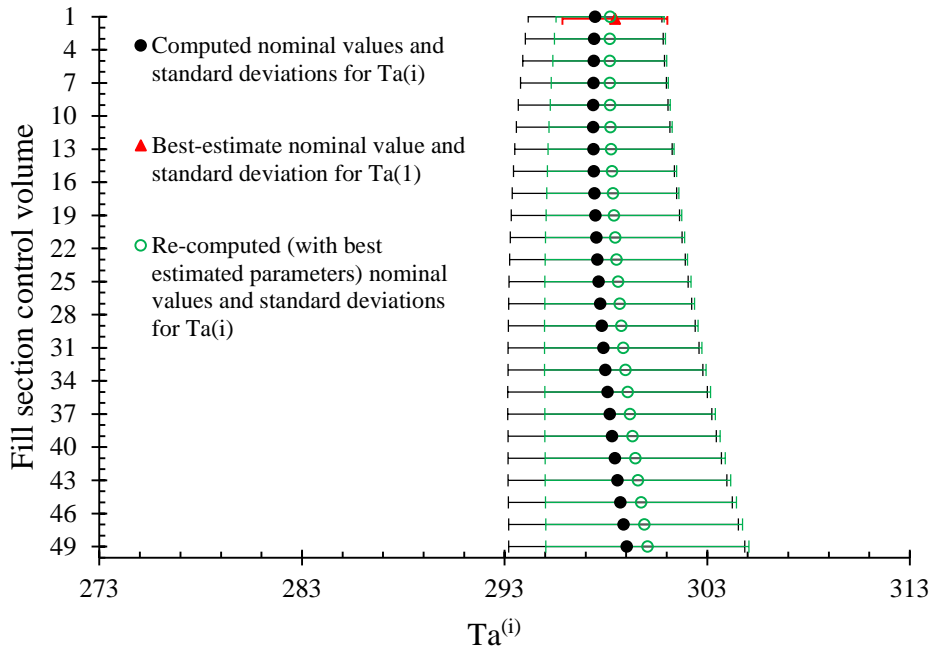


Figure 5.17. Computed (black), best-estimate (red), and re-computed (green; using best-estimate parameter values) nominal values and standard deviations for the air temperature, $Ta^{(i)}$, ($i = 1, \dots, 49$), at the exit of each control volume along the height of the fill section of the cooling tower.

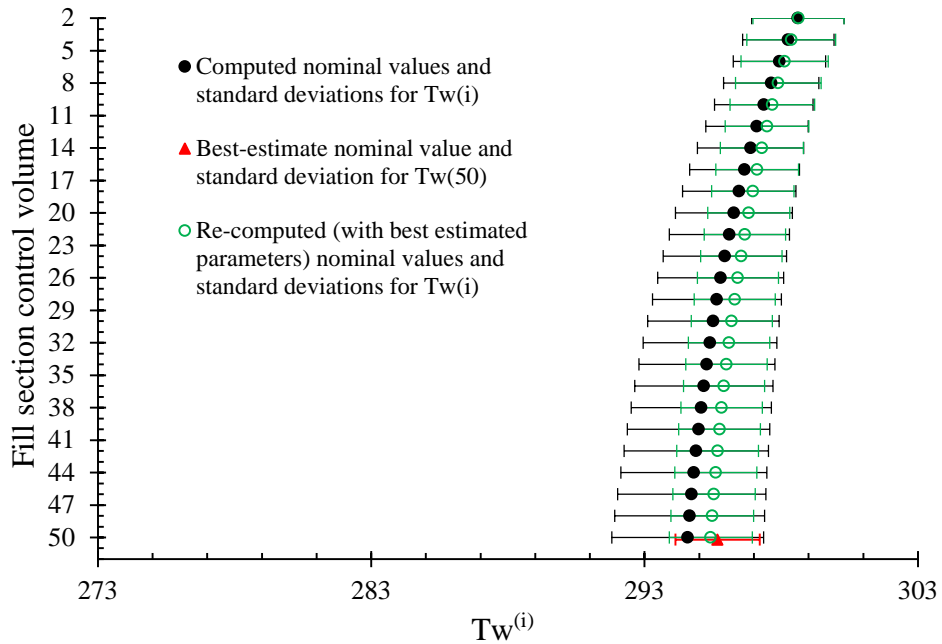


Figure 5.18. Computed (black), best-estimate (red), and re-computed (green; using best-estimate parameter values) nominal values and standard deviations for the water temperature, $Tw^{(i)}$,

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

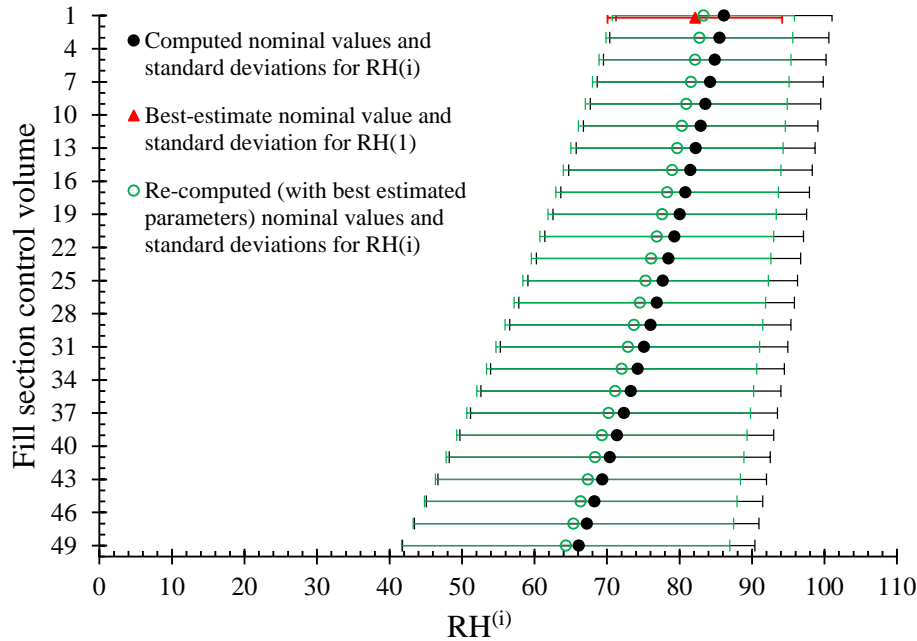


Figure 5.19. Computed (black), best-estimate (red), and re-computed (green; using best-estimate parameter values) nominal values and standard deviations for the air relative humidity, $RH^{(i)}$, ($i = 1, \dots, 49$), at the exit of each control volume along the height of the fill section of the cooling 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 (2nd-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 (2nd-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 (2nd-ASAM) for large-scale nonlinear systems: I. Theory,” *Nucl. Sci. Eng.* 184: 16–30.

Cacuci, D. G. 2016c. Second-order adjoint sensitivity analysis methodology (2nd-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_ga'	'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, $N_a = 42$ and $N_r = 2$. 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, $N_b = 10$ and $N_q = 1$.

```

/ 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, N_a , N_r , N_b and N_q , 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	$\alpha(N_a)$	Nominal values of N_a parameters of model A
rm.inp	22	input	$\mathbf{r}_m(N_r)$	Nominal values of N_r measured responses of model A
rc.inp	23	input	$\mathbf{r}_c(N_r)$	Nominal values of N_r computed responses of model A
Caa.inp	24	input	$\mathbf{C}_{aa}(N_a \times N_a)$	Covariance matrix of N_a parameters of model A
Car.inp	25	input	$\mathbf{C}_{ar}(N_a \times N_r)$	Correlations between N_a parameters and N_r responses of Model A
Crr.inp	26	input	$\mathbf{C}_{rr}(N_r \times N_r)$	Covariance matrix of N_r responses of model A
Sra.inp	27	input	$\mathbf{S}_{ra}(N_r \times N_a)$	Absolute sensitivities of N_r responses of Model A w.r.t N_a parameters of Model A
aBE.out	51	output	$\alpha^{be}(N_a)$	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 N_a parameters of Model A
CrrBE.out	54	output	$\mathbf{C}_{rr}^{be}(N_r \times N_r)$	Predicted covariance matrix of N_r responses of Model A
CarBE.out	55	output	$\mathbf{C}_{ar}^{be}(N_a \times N_r)$	Predicted correlation matrix between the N_a parameters and N_r responses of model A
Crrcomp.out	56	output	$\mathbf{C}_{rr}^{comp}(N_r \times N_r)$	Covariance matrix of N_r 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	$\alpha(N_a)$	Nominal values of N_a parameters of model A
rm.inp	22	Input	$\mathbf{r}_m(N_r)$	Nominal values of N_r measured responses of model A
rc.inp	23	Input	$\mathbf{r}_c(N_r)$	Nominal values of N_r computed responses of model A
Caa.inp	24	Input	$\mathbf{C}_{aa}(N_a \times N_a)$	Covariance matrix of N_a parameters of model A
Car.inp	25	Input	$\mathbf{C}_{ar}(N_a \times N_r)$	Correlations between N_a parameters and N_r 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 N_r computed responses of model A
Caa.inp	24	input	$\mathbf{C}_{aa}(N_a \times N_a)$	Covariance matrix of N_a parameters of model A
Car.inp	25	input	$\mathbf{C}_{ar}(N_a \times N_r)$	Correlations between N_a parameters and N_r responses of Model A
Crr.inp	26	input	$\mathbf{C}_{rr}(N_r \times N_r)$	Covariance matrix of N_r responses of model A
Sra.inp	27	input	$\mathbf{S}_{ra}(N_r \times N_a)$	Absolute sensitivities of N_r responses of Model A w.r.t N_a parameters of Model A
qm.inp	32	input	$\mathbf{q}_m(N_q)$	Nominal values of N_q additional measured responses for model A
qc.inp	33	input	$\mathbf{q}_c(N_q)$	Nominal values of N_q additional computed responses for model A
Cqq.inp	36	input	$\mathbf{C}_{qq}(N_q \times N_q)$	Covariance matrix of N_q additional responses for Model A
Caq.inp	42	input	$\mathbf{C}_{aq}(N_a \times N_q)$	Correlations between N_a parameters of Model A and N_q additional responses for Model A
Crq.inp	44	input	$\mathbf{C}_{rq}(N_r \times N_q)$	Correlations between N_r responses of Model A and N_q additional responses for Model A
Sqa.inp	46	input	$\mathbf{S}_{qa}(N_q \times N_a)$	Absolute sensitivities of N_q additional responses for Model A w.r.t N_a parameters of Model A
aBE.out	51	output	$\mathbf{a}^{be}(N_a)$	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 N_a parameters of Model A
CrrBE.out	54	output	$\mathbf{C}_{rr}^{be}(N_r \times N_r)$	Predicted covariance matrix of N_r responses of Model A
CarBE.out	55	output	$\mathbf{C}_{ar}^{be}(N_a \times N_r)$	Predicted correlation matrix between the N_a parameters and N_r responses of model A
Crrcomp.out	56	output	$\mathbf{C}_{rr}^{comp}(N_r \times N_r)$	Covariance matrix of N_r computed responses of model A
qBE.out	62	output	$\mathbf{q}^{be}(N_q)$	Best-estimate nominal values of N_q additional responses for model A
CqqBE.out	64	output	$\mathbf{C}_{qq}^{be}(N_q \times N_q)$	Predicted covariance matrix of N_q additional responses for model A
Cqqcomp.out	66	output	$\mathbf{C}_{qq}^{comp}(N_q \times N_q)$	Covariance matrix of N_q additional computed responses for model A
CaqBE.out	72	output	$\mathbf{C}_{aq}^{comp}(N_a \times N_q)$	Predicted correlation matrix between the N_a parameters and of Model A and N_q additional responses for model A
CrqBE.out	74	output	$\mathbf{C}_{rq}^{be}(N_r \times N_q)$	Predicted correlation matrix of between N_r responses of Model A and N_q additional responses for model A
Crqcomp.out	75	output	$\mathbf{C}_{rq}^{comp}(N_r \times N_q)$	Correlation matrix of N_r computed responses of Model A and N_q 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	$\mathbf{a}(N_a)$	Nominal values of N_a parameters of model A
rm.inp	22	input	$\mathbf{r}_m(N_r)$	Nominal values of N_r measured responses of model A
rc.inp	23	input	$\mathbf{r}_c(N_r)$	Nominal values of N_r computed responses of model A
Caa.inp	24	input	$\mathbf{C}_{aa}(N_a \times N_a)$	Covariance matrix of N_a parameters of model A
Car.inp	25	input	$\mathbf{C}_{ar}(N_a \times N_r)$	Correlations between N_a parameters and N_r responses of Model A
Crr.inp	26	input	$\mathbf{C}_{rr}(N_r \times N_r)$	Covariance matrix of N_r responses of model A
Sra.inp	27	input	$\mathbf{S}_{ra}(N_r \times N_a)$	Absolute sensitivities of N_r responses of Model A w.r.t N_a parameters of Model A
b.inp	31	input	$\mathbf{b}(N_b)$	Nominal values of N_b parameters of model B
qm.inp	32	input	$\mathbf{q}_m(N_q)$	Nominal values of N_q measured responses of model B
qc.inp	33	input	$\mathbf{q}_c(N_q)$	Nominal values of N_q computed responses of model B
Cbb.inp	34	input	$\mathbf{C}_{bb}(N_b \times N_b)$	Covariance matrix of N_b parameters of model B
Cbq.inp	35	input	$\mathbf{C}_{bq}(N_b \times N_q)$	Correlations between N_b parameters and N_q responses of Model B
Cqq.inp	36	input	$\mathbf{C}_{qq}(N_q \times N_q)$	Covariance matrix of N_q responses of model B
Sqb.inp	37	input	$\mathbf{S}_{qb}(N_q \times N_b)$	Absolute sensitivities of N_q responses of Model B w.r.t N_b parameters of Model B
Cab.inp	41	input	$\mathbf{C}_{ab}(N_a \times N_b)$	Correlation matrix between the N_a parameters of Model A and the N_b parameters of Model B
Caq.inp	42	input	$\mathbf{C}_{aq}(N_a \times N_q)$	Correlation matrix between the N_a parameters and of Model A and N_q responses of model B
Cbr.inp	43	input	$\mathbf{C}_{br}(N_b \times N_r)$	Correlation matrix between the N_b parameters of Model B and N_r responses of model A
Crq.inp	44	input	$\mathbf{C}_{rq}(N_r \times N_q)$	Correlation matrix of between N_r responses of Model A and N_q 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	$\mathbf{a}^{be}(N_a)$	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:

$$\mathbf{a} = \begin{pmatrix} 0.0197 \\ 0.16 \\ 1.0E+07 \\ 7.438 \end{pmatrix}. \quad (7.1)$$

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. For the neutron diffusion model, for example, the nominal values of the $N_r = 4$ measured responses are as follows:

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

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:

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

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:

$$\mathbf{C}_{aa} = \begin{pmatrix} (9.85 \times 10^{-4})^2 & 0 & 0 & 0 \\ 0 & (8.0 \times 10^{-3})^2 & 0 & 0 \\ 0 & 0 & (1.5 \times 10^6)^2 & 0 \\ 0 & 0 & 0 & (7.44 \times 10^{-1})^2 \end{pmatrix}. \quad (7.4)$$

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:

$$\mathbf{C}_{ar} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}. \quad (7.5)$$

The corresponding *Car.inp* is as follows:

```
4      4      0
```

In other applications, the parameters and measured responses are correlated, i.e., $\mathbf{C}_{ar} \neq \mathbf{0}$. 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}_{ar} 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

$$\mathbf{C}_{rr} = \begin{pmatrix} (1.7 \times 10^8)^2 & 0 & 0 & 0 \\ 0 & (2.15 \times 10^8)^2 & 0 & 0 \\ 0 & 0 & (1.89 \times 10^8)^2 & 0 \\ 0 & 0 & 0 & (1.87 \times 10^8)^2 \end{pmatrix}. \quad (7.6)$$

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

$$\mathbf{S} \triangleq \left(\frac{\partial R_i}{\partial \alpha_j} \right) = \begin{pmatrix} -1.92 \times 10^{11} & -1.33 \times 10^5 & 3.78 \times 10^2 & 5.08 \times 10^8 \\ -1.92 \times 10^{11} & -1.33 \times 10^5 & 3.78 \times 10^2 & 5.08 \times 10^8 \\ -1.76 \times 10^{11} & -1.24 \times 10^9 & 3.66 \times 10^2 & 4.92 \times 10^8 \\ -1.76 \times 10^{11} & -1.24 \times 10^9 & 3.66 \times 10^2 & 4.92 \times 10^8 \end{pmatrix}. \quad (7.7)$$

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 $\mathbf{\alpha}^{be}(N_a)$, 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:

$$\mathbf{\alpha}^{be} = \begin{pmatrix} 0.0198 \\ 0.1591 \\ 9.85 \times 10^6 \\ 7.388 \end{pmatrix}. \quad (7.8)$$

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:

$$\mathbf{r}^{be} = \begin{pmatrix} 3.66 \times 10^9 \\ 3.66 \times 10^9 \\ 3.56 \times 10^9 \\ 3.56 \times 10^9 \end{pmatrix}. \quad (7.9)$$

The corresponding output data file *rBE.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 *CaaBE.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 *Caa.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:

$$\mathbf{C}_{aa}^{be} = \begin{pmatrix} 9.03 \times 10^{-7} & 6.75 \times 10^{-9} & 3.03 \times 10^2 & 1.00 \times 10^{-4} \\ 6.75 \times 10^{-9} & 6.38 \times 10^{-5} & 7.37 \times 10^1 & 2.44 \times 10^{-5} \\ 3.03 \times 10^2 & 7.37 \times 10^1 & 8.24 \times 10^{11} & -4.71 \times 10^5 \\ 1.00 \times 10^{-4} & 2.44 \times 10^{-5} & -4.71 \times 10^5 & 3.97 \times 10^{-1} \end{pmatrix} \quad (7.10)$$

The corresponding output data file *CaaBE.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 *CarBE.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 *Car.inp*. For the neutron diffusion model, for example, the correlation matrix $\mathbf{C}_{ar}^{be}(N_a \times N_r)$ has the following structure:

$$\mathbf{C}_{ar}^{be} = \begin{pmatrix} -7.81 \times 10^3 & -7.81 \times 10^3 & 1.50 \times 10^3 & 1.50 \times 10^3 \\ 3.89 \times 10^4 & 3.89 \times 10^4 & -4.13 \times 10^4 & -4.13 \times 10^4 \\ 1.38 \times 10^{13} & 1.38 \times 10^{13} & 1.64 \times 10^{13} & 1.64 \times 10^{13} \\ 4.57 \times 10^6 & 4.57 \times 10^6 & 5.41 \times 10^6 & 5.41 \times 10^6 \end{pmatrix}. \quad (7.11)$$

The corresponding output data file *CarBE.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 *CrrBE.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 *Crr.inp*. For the neutron diffusion model, for example, the correlation matrix $\mathbf{C}_{rr}^{be}(N_r \times N_r)$ is as follows:

$$\mathbf{C}_{rr}^{be} = \begin{pmatrix} 9.04 \times 10^{15} & 9.04 \times 10^{15} & 8.64 \times 10^{15} & 8.64 \times 10^{15} \\ 9.04 \times 10^{15} & 9.04 \times 10^{15} & 8.64 \times 10^{15} & 8.64 \times 10^{15} \\ 8.64 \times 10^{15} & 8.64 \times 10^{15} & 8.45 \times 10^{15} & 8.45 \times 10^{15} \\ 8.64 \times 10^{15} & 8.64 \times 10^{15} & 8.45 \times 10^{15} & 8.45 \times 10^{15} \end{pmatrix} \quad (7.12)$$

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 *Crrcomp.out* contains the nonzero components of the covariance matrix $C_{rr}^{comp}(N_r \times N_r)$ 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 $C_{rr}^{comp}(N_r \times N_r)$ is as follows:

$$C_{rr}^{comp} = \begin{pmatrix} 4.99 \times 10^{17} & 4.99 \times 10^{17} & 4.82 \times 10^{17} & 4.82 \times 10^{17} \\ 4.99 \times 10^{17} & 4.99 \times 10^{17} & 4.82 \times 10^{17} & 4.82 \times 10^{17} \\ 4.82 \times 10^{17} & 4.82 \times 10^{17} & 4.66 \times 10^{17} & 4.66 \times 10^{17} \\ 4.82 \times 10^{17} & 4.82 \times 10^{17} & 4.66 \times 10^{17} & 4.66 \times 10^{17} \end{pmatrix} \quad (7.13)$$

The corresponding output data file *Crrcomp.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*:

$$\begin{aligned} \chi^2 &= 4.852 \\ \chi^2_d = (\chi^2)/(\text{number of responses}) &= 1.213 \end{aligned}$$

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.

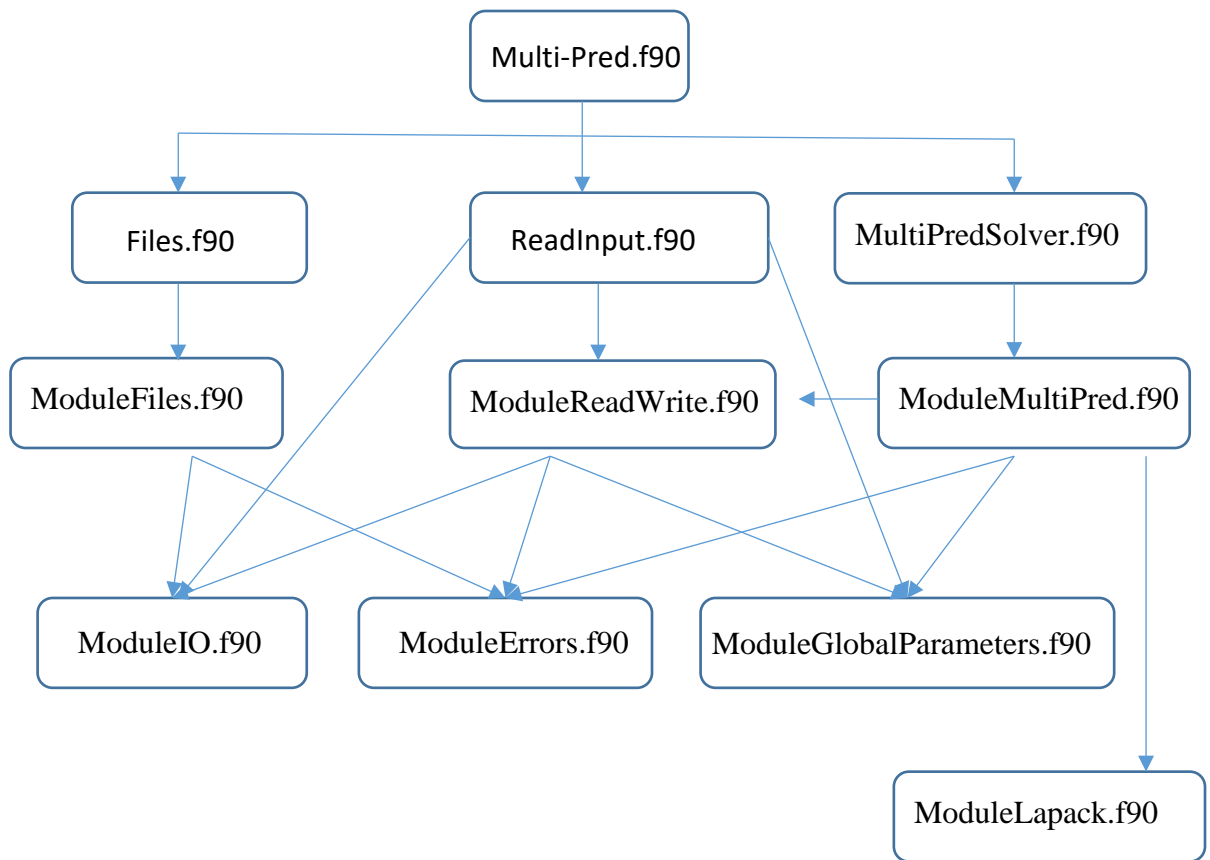


Figure 8.1 Multi-Pred Code Structure

8.1 Main program multi-pred.f90

```
1 PROGRAM multipred
2
3 !*****
4 !* multi-pred: This program is a computational implementation of the      *
5 !*     "predictive modeling for coupled multi-physics systems"          *
6 !*     methodology developed by Cacuci, based on the work "predictive*
7 !*     modeling for coupled multi-physics systems: I. Theory," Annals*
8 !*     of Nuclear Energy. 70, 266-278 (2014).                            *
9 !*                                                                       *
10 !*     The multi-pred has the following fundamental features:           *
11 !*     (i) it uses the maximum entropy principle to combine all         *
12 !*     available experimental and computational information to          *
13 !*     calibrate simultaneously all uncertain quantities,               *
14 !*     including model parameters, initial conditions, boundary        *
15 !*     conditions, and observed model responses;                       *
16 !*     (ii) it provides explicit formulas for the calibrated best      *
17 !*     estimate predicted values for the model responses and           *
18 !*     parameters;                                                      *
19 !*     (iii) it reduces the predicted uncertainties in these            *
20 !*     predicted model responses and parameters, providing              *
21 !*     explicit formulas for the predicted covariance matrices          *
22 !*     of responses and parameters;                                      *
23 !*     (iv) it provides a quantitative indicator --constructed from     *
24 !*     parameter and response covariances and responses                *
25 !*     sensitivities to parameters-- for quantifying the                *
26 !*     consistency (agreement or disagreement) among the a             *
27 !*     priori computational and experimental data.                      *
28 !*                                                                       *
29 !* multi-pred can perform predictive modeling for the following four cases *
30 !* (CaseNumber):                                                       *
31 !* = 1 "One-Model" Case: predictive modeling solely for Model A with Na *
32 !*     model parameters and Nr measured responses;                     *
33 !* = 2 "One-Model" Case: predictive modeling for Model A with Nb additional *
34 !*     parameters, but no additional responses;                         *
35 !* = 3 "One-Model" Case: predictive modeling for Model A with Nq additional *
36 !*     responses, but no additional parameters;                         *
37 !* = 4 "Two-Model" Case: predictive modeling for Model A coupled with Model *
38 !*     B.                                                                *
39 !*                                                                       *
40 !* Developed by the University of South Carolina, Columbia, SC         *
41 !*                                                                       *
42 !* called by: none                                                       *
43 !* calls to: Files, ReadInput, MultiPredSolver                        *
44 !*                                                                       *
45 !*                                                                       *
46 !*****
47
48 IMPLICIT NONE
49
50 ! read superfile and open all files for i/o
51 call Files
52
53 ! read all input data
54 call ReadInput
55
56 ! apply the Multi-Pred formulation and generate outputs
57 call MultiPredSolver
58 END PROGRAM multipred
```

8.2 Module ModuleGlobalParameters.f90

```
1  MODULE ModuleGlobalParameters
2
3  ! Symbolic names for kind types of 4-, 2- and 1-byte integers:
4  INTEGER, PARAMETER :: I4B = SELECTED_INT_KIND(9)
5  INTEGER, PARAMETER :: I2B = SELECTED_INT_KIND(4)
6  INTEGER, PARAMETER :: I1B = SELECTED_INT_KIND(2)
7  ! Symbolic names for kind types of single- and double precision reals:
8  INTEGER, PARAMETER :: SP = KIND(1.0)
9  INTEGER, PARAMETER :: DP = KIND(1.0D0)
10
11 ! Global parameters used in multi-pred:
12 ! alpha = nominal values of Na parameters of Model A (in)
13 ! rm    = nominal values of Nr measured responses of Model A (in)
14 ! rc    = nominal values of Nr computed responses of Model A (in)
15 ! Caa   = covariance matrix of Na parameters of Model A (in)
16 ! Car   = correlations between Na parameters and Nr responses
17 !       of Model A (in)
18 ! Crr   = covariance matrix of Nr responses of Model A (in)
19 ! Sra   = sensitivities of Model B (in)
20 ! beta  = nominal values of Nb parameters of Model B (in)
21 ! qm    = nominal values of Nq measured responses of Model B (in)
22 ! qc    = nominal values of Nq computed responses of Model B (in)
23 ! Cbb   = covariance matrix of Nb parameters of Model B (in)
24 ! Cbq   = correlations between Nb parameters and N1 responses
25 !       of Model B (in)
26 ! Cqq   = covariance matrix of Nq responses of Model B (in)
27 ! Sqb   = sensitivities of Model B (in)
28 ! Cab   = correlations between Na parameters of Model A and Nb
29 !       parameters of Model B (in)
30 ! Caq   = correlations between Na parameters of Model A and Nq
31 !       responses of Model B (in)
32 ! Cbr   = correlations between Nb parameters of Model B and Nr
33 !       responses of Model A (in)
34 ! Crq   = correlations between Nr responses of Model A and Nq
35 !       responses of Model B (in)
36 ! Srb   = sensitivities of Nr responses of Model A w.r.t. Nb
37 !       parameters of Model B (in)
38 ! Sqa   = sensitivities of Nq responses of Model B w.r.t. Na
39 !       parameters of Model A (in)
40 ! aBE   = best-estimate nominal values of Na parameters of Model A (out)
41 ! rBE   = best-estimate nominal values of Nr responses of Model A (out)
42 ! CaaBE = predicted covariance matrix of Na parameters of Model A (out)
43 ! CarBE = predicted correlation matrix between the Na parameters
44 !       and Nr responses of Model A (out)
45 ! CrrBE = predicted covariance matrix of Nr responses of Model A (out)
46 ! Crrcomp = covariance matrix of Nr computed responses of Model A (out)
47 ! bBE   = best-estimate nominal values of Nb parameters of Model B (out)
48 ! qBE   = best-estimate nominal values of Nq responses of Model B (out)
49 ! CbbBE = predicted covariance matrix of Nb parameters of Model B (out)
50 ! CbqBE = predicted correlation matrix between the Nb parameters
51 !       and Nq responses of Model B (out)
52 ! CqqBE = predicted covariance matrix of Nq responses of Model B (out)
53 ! Cqqcomp = covariance matrix of Nq computed responses of Model B (out)
54 ! CabBE = predicted correlation matrix between Na parameters of
55 !       Model A and Nb parameters of Model B (out)
56 ! CaqBE = predicted correlation matrix between Na parameters of
57 !       Model A and Nq responses of Model B (out)
58 ! CbrBE = predicted correlation matrix between Nb parameters of
```

```

59 !           Model B and Nr responses of Model A (out)
60 ! CrqBE    = predicted correlation matrix between Nr responses of
61 !           Model A and Nq responses of Model B (out)
62 ! Cqqcomp  = covariance matrix between Nr computed responses of
63 !           Model A and Nq computed responses of Model B (out)
64 ! chi2     = value of the consistency indicator chi^2   (out)
65
66 PUBLIC
67 REAL(DP), ALLOCATABLE :: alpha(:)
68 REAL(DP), ALLOCATABLE :: rm(:)
69 REAL(DP), ALLOCATABLE :: rc(:)
70 REAL(DP), ALLOCATABLE :: Caa(:, :)
71 REAL(DP), ALLOCATABLE :: Car(:, :)
72 REAL(DP), ALLOCATABLE :: Crr(:, :)
73 REAL(DP), ALLOCATABLE :: Sra(:, :)
74
75 REAL(DP), ALLOCATABLE :: beta(:)
76 REAL(DP), ALLOCATABLE :: qm(:)
77 REAL(DP), ALLOCATABLE :: qc(:)
78 REAL(DP), ALLOCATABLE :: Cbb(:, :)
79 REAL(DP), ALLOCATABLE :: Cbq(:, :)
80 REAL(DP), ALLOCATABLE :: Cqq(:, :)
81 REAL(DP), ALLOCATABLE :: Sqb(:, :)
82
83 REAL(DP), ALLOCATABLE :: Cab(:, :)
84 REAL(DP), ALLOCATABLE :: Caq(:, :)
85 REAL(DP), ALLOCATABLE :: Cbr(:, :)
86 REAL(DP), ALLOCATABLE :: Crq(:, :)
87 REAL(DP), ALLOCATABLE :: Srb(:, :)
88 REAL(DP), ALLOCATABLE :: Sqa(:, :)
89
90 REAL(DP), ALLOCATABLE :: aBE(:)
91 REAL(DP), ALLOCATABLE :: rBE(:)
92 REAL(DP), ALLOCATABLE :: CaaBE(:, :)
93 REAL(DP), ALLOCATABLE :: CarBE(:, :)
94 REAL(DP), ALLOCATABLE :: CrrBE(:, :)
95 REAL(DP), ALLOCATABLE :: Crrcomp(:, :)
96
97 REAL(DP), ALLOCATABLE :: bBE(:)
98 REAL(DP), ALLOCATABLE :: qBE(:)
99 REAL(DP), ALLOCATABLE :: CbbBE(:, :)
100 REAL(DP), ALLOCATABLE :: CbqBE(:, :)
101 REAL(DP), ALLOCATABLE :: CqqBE(:, :)
102 REAL(DP), ALLOCATABLE :: Cqqcomp(:, :)
103
104 REAL(DP), ALLOCATABLE :: CabBE(:, :)
105 REAL(DP), ALLOCATABLE :: CaqBE(:, :)
106 REAL(DP), ALLOCATABLE :: CbrBE(:, :)
107 REAL(DP), ALLOCATABLE :: CrqBE(:, :)
108 REAL(DP), ALLOCATABLE :: Crqcomp(:, :)
109
110 REAL(DP)                :: chi2
111
112 INTEGER(I4B)            :: CaseNumber
113 INTEGER(I4B)            :: Na, Nr, Nb, Nq
114
115 END MODULE ModuleGlobalParameters

```

8.3 Module ModuleIO.f90

```
1  MODULE ModuleIO
2
3  IMPLICIT NONE
4
5  INTEGER, PARAMETER      :: usupr      = 5, &
6                          udims       =20, &
7  !unit for the input of model A
8                          ua_nom      =21, &
9                          ur_mea     =22, &
10                         ur_com     =23, &
11                         uC_aa      =24, &
12                         uC_ar      =25, &
13                         uC_rr      =26, &
14                         uS_ra      =27, &
15
16 !unit for the input of model B
17                         ub_nom     =31, &
18                         uq_mea     =32, &
19                         uq_com     =33, &
20                         uC_bb      =34, &
21                         uC_bq      =35, &
22                         uC_qq      =36, &
23                         uS_qb      =37, &
24
25 !unit for the input of the coupled matrices
26 !between models A & B
27                         uC_ab      =41, &
28                         uC_aq      =42, &
29                         uC_br      =43, &
30                         uC_rq      =44, &
31                         uS_rb      =45, &
32                         uS_qa      =46, &
33
34 !unit for the output of model A
35                         ua_BE      =51, &
36                         ur_BE      =52, &
37                         uC_aaBE     =53, &
38                         uC_rrBE     =54, &
39                         uC_arBE     =55, &
40                         uCrr_comp    =56, &
41
42 !unit for the output of model B
43                         ub_BE      =61, &
44                         uq_BE      =62, &
45                         uC_bbBE     =63, &
46                         uC_qqBE     =64, &
47                         uC_bqBE     =65, &
48                         uCqq_comp    =66, &
49
50 !unit for the output of the coupled matrices
51 !between models A & B
52                         uC_abBE     =71, &
53                         uC_aqBE     =72, &
54                         uC_brBE     =73, &
55                         uC_rqBE     =74, &
56                         uCrq_comp    =75, &
57                         uchi2       =76
58
```

```

59 LOGICAL :: a_nom   =.false., r_mea   =.false., r_com =.false., &
60           C_aa     =.false., C_ar    =.false., C_rr  =.false., &
61           S_ra     =.false., b_nom   =.false., q_mea =.false., &
62           q_com    =.false., C_bb    =.false., C_bq  =.false., &
63           C_qq     =.false., S_qb    =.false., C_ab  =.false., &
64           C_aq     =.false., C_br    =.false., C_rq  =.false., &
65           S_rb     =.false., S_qa    =.false., a_BE  =.false., &
66           r_BE     =.false., C_aaBE  =.false., C_rrBE=.false., &
67           C_arBE  =.false., Crr_comp=.false., b_BE  =.false., &
68           q_BE     =.false., C_bbBE  =.false., C_qqBE=.false., &
69           C_bqBE  =.false., Cqq_comp=.false., C_abBE=.false., &
70           C_aqBE  =.false., C_brBE  =.false., C_rqBE=.false., &
71           Crq_comp=.false., chi_2   =.false., dims  =.false.
72
73 END MODULE ModuleIO

```

8.4 Module ModuleErrors.f90

```

1  MODULE ModuleErrors
2
3  USE ModuleGlobalParameters
4
5  IMPLICIT NONE
6  CHARACTER(LEN=80) errstr(5)
7  INTEGER(I4B) :: alloc_err,ierr = 0
8  LOGICAL      :: lerr = .false.
9
10 DATA errstr/
11 'Error condition during file open', & ! error 1
12 'Error condition during file read', & ! error 2
13 'Error condition during file write', & ! error 3
14 'Unable to allocate storage for array', & ! error 4
15 'Unable to deallocate storage for array' / ! error 5
16
17 CONTAINS
18
19 SUBROUTINE errmsg
20
21     write(*, '(/,a)') '****'//trim(errstr(ierr))//'****'
22
23 END SUBROUTINE errmsg
24
25 END MODULE ModuleErrors

```

8.5 Subroutine Files.f90

```
1  SUBROUTINE Files
2
3  !*****
4  !*
5  !* Open input/output files for multi-pred using super file format. For
6  !* different cases, the required input and output files are different.
7  !*
8  !* called by:  multipred
9  !* calls to:  getarg,errmsg, filescase1, filescase2, filescase3, filescase4
10 !*
11 !*****
12
13  !Global-----
14  USE ModuleFiles
15
16  IMPLICIT NONE
17
18  !Local-----
19  CHARACTER(LEN=128) :: argv,filename
20  CHARACTER(LEN=12)  :: header
21  CHARACTER(LEN=8)   :: category
22  INTEGER(I4B)       :: argc, i
23  CHARACTER(LEN=128) :: commnt
24
25  argc = 1
26  call getarg(argc,argv)
27  filename = argv
28  open(usupr,file=filename,status='old',err=100)
29  read(usupr,'(a12)') header
30  if (header /= 'MultiPredSup') then
31    write(*,*) 'This is not a multi-pred superfile.'
32    stop
33  end if
34  write(*,900) trim(filename)
35  ! read the dimensions.inp file for control variables
36  read(usupr,*) category,filename
37  if (category == 'dims') then
38    open(udims,file=filename,status='old',err=100)
39    write(*,1000) trim(filename)
40    dims = .true.
41  else
42    write(*,*) 'dimensions.inp is specified in the superfile.'
43    stop
44  end if
45
46  ! read CaseNumber
47  do i=1, 9
48    read(udims,*,err=100) commnt
49  end do
50  read (udims,*,err=100) CaseNumber
51
52  ! file I/O for Case 1
53  if(CaseNumber == 1) then
54    call filescase1
55  end if
56
57  ! file I/O for Case 2
58  if(CaseNumber == 2) then
```

```

59     call filescase2
60 end if
61
62 ! file I/O for Case 3
63 if(CaseNumber == 3) then
64     call filescase3
65 end if
66
67 ! file I/O for Case 4
68 if(CaseNumber == 4) then
69     call filescase4
70 end if
71
72 write(*, 1001)
73 if(CaseNumber == 1) then
74     write(*, 1002)
75 else if (CaseNumber == 2) then
76     write(*, 1003)
77 else if (CaseNumber == 3) then
78     write(*, 1004)
79 else if (CaseNumber == 4) then
80     write(*, 1005)
81 end if
82
83 return
84 100 lerr = .true.; ierr = 1; call errmsg; stop
85 900 format(/, &
86     1x, '----- Summary of Multil-Pred -----', /, &
87     1x, ' Input super file .....,', a)
88 1000 format(&
89     1x, ' Input file for Case selection and dimensions control .....,', a)
90 1001 format(/, &
91     1x, 'Case selected for this run:', a)
92 1002 format(&
93     1x, 'Case 1 -- "One-Model" Case: predictive modeling solely for Model A', /, &
94     '          with Na model parameters and Nr measured responses.', a)
95 1003 format(&
96     1x, 'Case 2 -- "One-Model" Case: predictive modeling for Model A with Nb', /, &
97     '          additional parameters, but no additional responses.', a)
98 1004 format(&
99     1x, 'Case 3 -- "One-Model" Case: predictive modeling for Model A with Nq', /, &
100    '          additional responses, but no additional parameters.', a)
101 1005 format(&
102    1x, 'Case 4 -- "Two-Model" Case: predictive modeling for Model A coupled', /, &
103    '          with Model B.', a)
104
105 END SUBROUTINE Files

```

8.6 Module ModuleFiles.f90

```

1  MODULE ModuleFiles
2  !*****
3  !*
4  !* Module ModuleFiles encapsulates subroutines for:
5  !*      subroutine filescase1()
6  !*      subroutine filescase2()
7  !*      subroutine filescase3()
8  !*      subroutine filescase4()
9  !*
10 !*****
11
12 !Global-----
13 USE ModuleErrors
14 USE ModuleIO
15 USE ModuleGlobalParameters
16
17 IMPLICIT NONE
18
19 CONTAINS
20
21 !+++++
22
23 SUBROUTINE filescase1()
24
25 !*****
26 !*
27 !* open needed files for case 1: "One-Model" Case: predictive modeling
28 !* solely for Model A with Na model parameters and Nr
29 !* measured responses.
30 !*
31 !* Open input/output files for multi-pred using super file format
32 !*
33 !* Category      I/O      Description
34 !* -----      -
35 !* 'dims'        input     defines the Case selection and dimensions control
36 !* 'a_nom'        input     nominal values of Na parameters of Model A
37 !* 'r_mea'        input     nominal values of Nr measured responses of Model A
38 !* 'r_com'        input     nominal values of Nr computed responses of Model A
39 !* 'C_aa'         input     covariance matrix of Na parameters of Model A
40 !* 'C_ar'         input     covariance matrix of parameter-response of Model A
41 !* 'C_rr'         input     covariance matrix of Nr responses of Model A
42 !* 'S_ra'         input     sensitivities of Model A
43 !*
44 !* 'a_BE'         output    best-estimate nominal values of Na parameters of
45 !*                                     Model A
46 !* 'r_BE'         output    best-estimate nominal values of Nr responses of
47 !*                                     Model A
48 !* 'C_aaBE'       output    predicted covariance matrix of Na parameters of
49 !*                                     Model A
50 !* 'C_rrBE'       output    predicted covariance matrix of Nr responses of
51 !*                                     Model A
52 !* 'C_arBE'       output    predicted correlation matrix between the Na
53 !*                                     parameters and Nr responses of Model A
54 !* 'Crr_comp'     output    covariance matrix of Nr computed responses of Model A
55 !* 'chi_2'        output    value of the consistency indicator chi^2
56 !*
57 !* called by: Files
58 !* calls to: getarg,errmsg

```



```

59  !*
60  !*****
61  IMPLICIT NONE
62  !Local-----
63  CHARACTER(LEN=128) :: filename
64  CHARACTER(LEN=8)  :: category
65
66  inquire(unit=usupr)
67  do
68      read(usupr,*,end=1) category,filename
69
70  ! INPUT FILES
71  if (category == 'a_nom') then
72      open(ua_nom,file=filename,status='old',err=100)
73      write(*,1010) trim(filename)
74      a_nom = .true.
75  else if (category == 'r_mea') then
76      open(ur_mea,file=filename,status='old',err=100)
77      write(*,1020) trim(filename)
78      r_mea = .true.
79  else if (category == 'r_com') then
80      open(ur_com,file=filename,status='old',err=100)
81      write(*,1030) trim(filename)
82      r_com = .true.
83  else if (category == 'C_aa') then
84      open(uC_aa,file=filename,status='old',err=100)
85      write(*,1040) trim(filename)
86      C_aa = .true.
87  else if (category == 'C_ar') then
88      open(uC_ar,file=filename,status='old',err=100)
89      write(*,1050) trim(filename)
90      C_ar = .true.
91  else if (category == 'C_rr') then
92      open(uC_rr,file=filename,status='old',err=100)
93      write(*,1060) trim(filename)
94      C_rr = .true.
95  else if (category == 'S_ra') then
96      open(uS_ra,file=filename,status='old',err=100)
97      write(*,1070) trim(filename)
98      S_ra = .true.
99
100 ! OUTPUT FILES FOR RESULTS
101 else if (category == 'a_BE') then
102     open (ua_BE,file=filename,status='unknown',err=100)
103     write(*,1100) trim(filename)
104     a_BE = .true.
105 else if (category == 'r_BE') then
106     open (ur_BE,file=filename,status='unknown',err=100)
107     write(*,1110) trim(filename)
108     r_BE = .true.
109 else if (category == 'C_aaBE') then
110     open (uC_aaBE,file=filename,status='unknown',err=100)
111     write(*,1120) trim(filename)
112     C_aaBE = .true.
113 else if (category == 'C_rrBE') then
114     open (uC_rrBE,file=filename,status='unknown',err=100)
115     write(*,1130) trim(filename)
116     C_rrBE = .true.
117 else if (category == 'C_arBE') then
118     open (uC_arBE,file=filename,status='unknown',err=100)
119     write(*,1140) trim(filename)

```

```

120     C_arBE = .true.
121     else if (category == 'Crr_comp') then
122         open (uCrr_comp,file=filename,status='unknown',err=100)
123         write(*,1150) trim(filename)
124         Crr_comp = .true.
125     else if (category == 'chi2') then
126         open (uchi2,file=filename,status='unknown',err=100)
127         write(*,1160) trim(filename)
128         chi_2 = .true.
129 !     else
130 !         write(*,1500) category,filename
131     end if
132 end do
133 1 close(usupr)
134
135 ! CHECK TO SEE IF REQUIRED INPUT/OUTPUT FILES ARE OPEN IN THE SUPERFILE
136 ! INPUT
137 if (.not. a_nom) then
138     write(*,1510) 'a_nom'; stop
139 end if
140 if (.not. r_mea) then
141     write(*,1510) 'r_mea'; stop
142 end if
143 if (.not. r_com) then
144     write(*,1510) 'r_com'; stop
145 end if
146 if (.not. C_aa) then
147     write(*,1510) 'C_aa'; stop
148 end if
149 if (.not. S_ra) then
150     write(*,1510) 'S_ra'; stop
151 end if
152 if (.not. C_ar) then
153     write(*,1510) 'C_ar'; stop
154 end if
155 if (.not. C_rr) then
156     write(*,1510) 'C_rr'; stop
157 end if
158 ! OUTPUT
159 if (.not. Crr_comp) then
160     write(*,1520) 'Crr_comp';stop
161 end if
162 if (.not. a_BE) then
163     write(*,1520) 'a_BE'; stop
164 end if
165 if (.not. r_BE) then
166     write(*,1520) 'r_BE'; stop
167 end if
168 if (.not. C_aaBE) then
169     write(*,1520) 'C_aaBE'; stop
170 end if
171 if (.not. C_rrBE) then
172     write(*,1520) 'C_rrBE'; stop
173 end if
174 if (.not. C_arBE) then
175     write(*,1520) 'C_arBE'; stop
176 end if
177 if (.not. chi_2) then
178     write(*,1520) 'chi_2'; stop
179 end if
180 return

```

```

181
182 100 lerr = .true.;ierr = 1;call errmsg;stop
183 1010 format(&
184 1x,' Input file for parameters nominal values .....',a)
185 1020 format(&
186 1x,' Input file for measured responses nominal values .....',a)
187 1030 format(&
188 1x,' Input file for computed responses nominal values .....',a)
189 1040 format(&
190 1x,' Input file for covariance matrix of parameters .....',a)
191 1050 format(&
192 1x,' Input file for correlation matrix of parameter-response .....',a)
193 1060 format(&
194 1x,' Input file for covariance matrix for responses .....',a)
195 1070 format(&
196 1x,' Input file for sensitivities .....',a)
197 1100 format(&
198 1x,'Output file for best-estimate parameters values .....',a)
199 1110 format(&
200 1x,'Output file for best-estimate response values .....',a)
201 1120 format(&
202 1x,'Output file for predicted covariance matrix of parameters.....',a)
203 1130 format(&
204 1x,'Output file for predicted covariance matrix of responses .....',a)
205 1140 format(&
206 1x,'Output file for predicted correlation of parameter-response ....',a)
207 1150 format(&
208 1x,'Output file for covariance matrix for computed responses .....',a)
209 1160 format(&
210 1x,'Output file for the consistency indicator chi^2 .....',a)
211 1500 format(1x,'Invalid file type:',a,' ....skipping',/, &
212 1x,'Specified file name:',a)
213 1510 format(&
214 1x,a,' input file not specified in superfile!')
215 1520 format(&
216 1x,a,' output file not specified in superfile!')
217
218 END SUBROUTINE filescase1
219
220 !+++++
221
222 SUBROUTINE filescase2()
223
224 !*****
225 !*
226 !* open needed files for case 2: predictive modeling for Model A with Nb
227 !* additional parameters, but no additional responses;
228 !*
229 !* Open input/output files for multi-pred using super file format
230 !*
231 !* Category      I/O      Description
232 !* -----      -
233 !* 'dims'        input    defines the Case selection and dimensions control
234 !* 'a_nom'        input    nominal values of Na parameters of Model A
235 !* 'r_mea'        input    nominal values of Nr measured responses of Model A
236 !* 'r_com'        input    nominal values of Nr computed responses of Model A
237 !* 'C_aa'         input    covariance matrix of Na parameters of Model A
238 !* 'C_ar'         input    covariance matrix of parameter-response of Model A
239 !* 'C_rr'         input    covariance matrix of Nr responses of Model A
240 !* 'S_ra'         input    sensitivities of Model A
241 !*

```

```

242 !* 'b_nom'    input    nominal values of Nb additional parameters for Model A *
243 !* 'C_bb'    input    covariance matrix of Nb additional parameters          *
244 !*                                     *
245 !* 'C_ab'    input    correlations between Na parameters and Nb parameters *
246 !* 'C_br'    input    correlations between Nb parameters and Nr responses *
247 !* 'S_rb'    input    sensitivities of Nr responses wrt Nb parameters    *
248 !*                                     *
249 !* 'a_BE'    output   best-estimate nominal values of Na parameters of    *
250 !*                                     Model A                               *
251 !* 'r_BE'    output   best-estimate nominal values of Nr responses of    *
252 !*                                     Model A                               *
253 !* 'C_aaBE'  output   predicted covariance matrix of Na parameters of    *
254 !*                                     Model A                               *
255 !* 'C_rrBE'  output   predicted covariance matrix of Nr responses of Model A *
256 !* 'C_arBE'  output   predicted correlation matrix between the Na parameters *
257 !*                                     and Nr responses of Model A          *
258 !* 'Crr_comp' output   covariance matrix of Nr computed responses of Model A *
259 !*                                     *
260 !* 'b_BE'    output   best-estimate nominal values of Nb additional    *
261 !*                                     parameters for Model A                *
262 !* 'C_bbBE'  output   predicted covariance matrix of Nb additional    *
263 !*                                     parameters for Model A                *
264 !*                                     *
265 !* 'C_abBE'  output   predicted correlations between Na parameters and Nb    *
266 !*                                     additional parameters for Model A      *
267 !* 'C_brBE'  output   predicted correlations between Nb additional parameters *
268 !*                                     and Nr responses of Model A          *
269 !* 'chi_2'   output   value of the consistency indicator chi^2          *
270 !*                                     *
271 !* called by: Files                                                    *
272 !* calls to:  getarg,errmsg                                           *
273 !*                                     *
274 !*****
275 IMPLICIT NONE
276 !Local-----
277 CHARACTER(LEN=128) :: filename
278 CHARACTER(LEN=8)  :: category
279
280 inquire(unit=usupr)
281 do
282   read(usupr,*,end=1) category,filename
283
284 ! INPUT FILES FOR MODEL A
285 if (category == 'a_nom') then
286   open(ua_nom,file=filename,status='old',err=100)
287   a_nom = .true.
288 else if (category == 'r_mea') then
289   open(ur_mea,file=filename,status='old',err=100)
290   r_mea = .true.
291 else if (category == 'r_com') then
292   open(ur_com,file=filename,status='old',err=100)
293   r_com = .true.
294 else if (category == 'C_aa') then
295   open(uC_aa,file=filename,status='old',err=100)
296   C_aa = .true.
297 else if (category == 'C_ar') then
298   open(uC_ar,file=filename,status='old',err=100)
299   C_ar = .true.
300 else if (category == 'C_rr') then
301   open(uC_rr,file=filename,status='old',err=100)
302   C_rr = .true.

```

```

303     else if (category == 'S_ra') then
304         open(uS_ra,file=filename,status='old',err=100)
305         S_ra = .true.
306
307 ! INPUT FILES FOR ADDITIONAL PARAMETERS
308     else if (category == 'b_nom') then
309         open(ub_nom,file=filename,status='old',err=100)
310         b_nom = .true.
311     else if (category == 'C_bb') then
312         open(uC_bb,file=filename,status='old',err=100)
313         C_bb = .true.
314
315 ! INPUT FILES FOR COUPLED MATRICES
316     else if (category == 'C_ab') then
317         open(uC_ab,file=filename,status='old',err=100)
318         C_ab = .true.
319     else if (category == 'C_br') then
320         open(uC_br,file=filename,status='old',err=100)
321         C_br = .true.
322     else if (category == 'S_rb') then
323         open(uS_rb,file=filename,status='old',err=100)
324         S_rb = .true.
325
326 ! OUTPUT FILES FOR MODEL A
327     else if (category == 'a_BE') then
328         open (ua_BE,file=filename,status='unknown',err=100)
329         a_BE = .true.
330     else if (category == 'r_BE') then
331         open (ur_BE,file=filename,status='unknown',err=100)
332         r_BE = .true.
333     else if (category == 'C_aaBE') then
334         open (uC_aaBE,file=filename,status='unknown',err=100)
335         C_aaBE = .true.
336     else if (category == 'C_rrBE') then
337         open (uC_rrBE,file=filename,status='unknown',err=100)
338         C_rrBE = .true.
339     else if (category == 'C_arBE') then
340         open (uC_arBE,file=filename,status='unknown',err=100)
341         C_arBE = .true.
342     else if (category == 'Crr_comp') then
343         open (uCrr_comp,file=filename,status='unknown',err=100)
344         Crr_comp = .true.
345
346 ! OUTPUT FILES FOR MODEL B
347     else if (category == 'b_BE') then
348         open (ub_BE,file=filename,status='unknown',err=100)
349         b_BE = .true.
350     else if (category == 'C_bbBE') then
351         open (uC_bbBE,file=filename,status='unknown',err=100)
352         C_bbBE = .true.
353
354 ! OUTPUT FILES FOR COUPLED MATRICES BETWEEN MODELS A & B
355     else if (category == 'C_abBE') then
356         open (uC_abBE,file=filename,status='unknown',err=100)
357         C_abBE = .true.
358     else if (category == 'C_brBE') then
359         open (uC_brBE,file=filename,status='unknown',err=100)
360         C_brBE = .true.
361
362     else if (category == 'chi2') then
363         open (uchi2,file=filename,status='unknown',err=100)

```

```

364     chi_2 = .true.
365 !     else
366 !     write(*,1500) category,filename
367     end if
368 end do
369 1 close(usupr)
370
371 ! CHECK TO SEE IF REQUIRED INPUT/OUTPUT FILES ARE OPEN IN THE SUPERFILE
372 ! INPUT
373 if (.not. a_nom) then
374     write(*,1510) 'a_nom'; stop
375 end if
376 if (.not. r_mea) then
377     write(*,1510) 'r_mea'; stop
378 end if
379 if (.not. r_com) then
380     write(*,1510) 'r_com'; stop
381 end if
382 if (.not. C_aa) then
383     write(*,1510) 'C_aa'; stop
384 end if
385 if (.not. C_ar) then
386     write(*,1510) 'C_ar'; stop
387 end if
388 if (.not. C_rr) then
389     write(*,1510) 'C_rr'; stop
390 end if
391 if (.not. S_ra) then
392     write(*,1510) 'S_ra'; stop
393 end if
394 if (.not. b_nom) then
395     write(*,1510) 'b_nom'; stop
396 end if
397 if (.not. C_bb) then
398     write(*,1510) 'C_bb'; stop
399 end if
400 if (.not. C_ab) then
401     write(*,1510) 'C_ab'; stop
402 end if
403 if (.not. C_br) then
404     write(*,1510) 'C_br'; stop
405 end if
406 if (.not. S_rb) then
407     write(*,1510) 'S_rb'; stop
408 end if
409
410 ! OUTPUT
411 if (.not. Crr_comp) then
412     write(*,1520) 'Crr_comp';stop
413 end if
414 if (.not. a_BE) then
415     write(*,1520) 'a_BE'; stop
416 end if
417 if (.not. r_BE) then
418     write(*,1520) 'r_BE'; stop
419 end if
420 if (.not. C_aaBE) then
421     write(*,1520) 'C_aaBE'; stop
422 end if
423 if (.not. C_rrBE) then
424     write(*,1520) 'C_rrBE'; stop

```

```

425     end if
426     if (.not. C_arBE) then
427         write(*,1520) 'C_arBE'; stop
428     end if
429     if (.not. b_BE) then
430         write(*,1520) 'b_BE'; stop
431     end if
432     if (.not. C_bbBE) then
433         write(*,1520) 'C_bbBE'; stop
434     end if
435     if (.not. C_abBE) then
436         write(*,1520) 'C_abBE'; stop
437     end if
438     if (.not. C_brBE) then
439         write(*,1520) 'C_brBE'; stop
440     end if
441     if (.not. chi_2) then
442         write(*,1520) 'chi_2'; stop
443     end if
444
445     return
446
447     100 lerr = .true.; ierr = 1; call errmsg; stop
448     1010 format(&
449         1x, ' Input file for parameters nominal values .....',a)
450     1020 format(&
451         1x, ' Input file for measured responses nominal values .....',a)
452     1030 format(&
453         1x, ' Input file for computed responses nominal values .....',a)
454     1040 format(&
455         1x, ' Input file for covariance matrix of parameters .....',a)
456     1050 format(&
457         1x, ' Input file for correlation matrix of parameter-response .....',a)
458     1060 format(&
459         1x, ' Input file for covariance matrix for responses .....',a)
460     1070 format(&
461         1x, ' Input file for sensitivities .....',a)
462     1100 format(&
463         1x, 'Output file for best-estimate parameters values .....',a)
464     1110 format(&
465         1x, 'Output file for best-estimate response values .....',a)
466     1120 format(&
467         1x, 'Output file for predicted covariance matrix of parameters.....',a)
468     1130 format(&
469         1x, 'Output file for predicted covariance matrix of responses .....',a)
470     1140 format(&
471         1x, 'Output file for predicted correlation of parameter-response ....',a)
472     1150 format(&
473         1x, 'Output file for covariance matrix for computed responses .....',a)
474     1160 format(&
475         1x, 'Output file for the consistency indicator chi^2 .....',a)
476     1500 format(1x, 'Invalid file type:',a, ' ....skipping',/, &
477         1x, 'Specified file name:',a)
478     1510 format(&
479         1x,a, ' input file not specified in superfile!')
480     1520 format(&
481         1x,a, ' output file not specified in superfile!')
482
483     END SUBROUTINE filescase2
484
485     !+++++

```

```

486
487 SUBROUTINE filescase3()
488
489 !*****
490 !*
491 !* open needed files for read and write for Case 3: predictive modeling for
492 !* Model A with Nq additional responses, but no additional parameters.
493 !*
494 !* Open input/output files for multi-pred using super file format
495 !*
496 !* Category      I/O      Description
497 !* -----      -
498 !* 'dims'        input     defines the Case selection and dimensions control
499 !* 'a_nom'        input     nominal values of Na parameters of Model A
500 !* 'r_mea'        input     nominal values of Nr measured responses of Model A
501 !* 'r_com'        input     nominal values of Nr computed responses of Model A
502 !* 'C_aa'         input     covariance matrix of Na parameters of Model A
503 !* 'C_ar'         input     correlations between Na parameters and Nr responses of
504 !*                                     Model A
505 !* 'C_rr'         input     covariance matrix of Nr responses of Model A
506 !* 'S_ra'         input     sensitivities of Model A
507 !*
508 !* 'q_mea'        input     nominal values of Nq measured responses for Model A
509 !* 'q_com'        input     nominal values of Nq computed responses for Model A
510 !* 'C_qq'         input     covariance matrix of Nq additional responses for
511 !*                                     Model A
512 !*
513 !* 'C_aq'         input     correlations between Na parameters of Model A and
514 !*                                     Nq additional responses for Model A
515 !* 'C_rq'         input     correlations between Nr responses of Model A and Nq
516 !*                                     additional responses for Model A
517 !* 'S_qa'         input     sensitivities of Nq additional responses for Model A
518 !*                                     w.r.t. Na parameters of Model A
519 !*
520 !* 'a_BE'         output    best-estimate nominal values of Na parameters of
521 !*                                     Model A
522 !* 'r_BE'         output    best-estimate nominal values of Nr responses of
523 !*                                     Model A
524 !* 'C_aaBE'       output    predicted covariance matrix of Na parameters of
525 !*                                     Model A
526 !* 'C_rrBE'       output    predicted covariance matrix of Nr responses of Model A
527 !* 'C_arBE'       output    predicted correlation matrix between the Na parameters
528 !*                                     and Nr responses of Model A
529 !* 'Crr_comp'     output    covariance matrix of Nr computed responses of Model A
530 !*
531 !* 'q_BE'         output    best-estimate nominal values of Nq additional response
532 !*                                     for Model A
533 !* 'C_qqBE'       output    predicted covariance matrix of Nq additional response
534 !*                                     for Model A
535 !* 'Cqq_comp'     output    covariance matrix for Nq additional computed responses
536 !*                                     of Model A
537 !*
538 !* 'C_aqBE'       output    predicted correlation matrix between Na parameters of
539 !*                                     Model A and Nq additional responses for Model A
540 !* 'C_rqBE'       output    predicted correlation matrix between Nr responses of
541 !*                                     Model A and Nq additional responses for Model A
542 !* 'Crq_comp'     output    correlation matrix between Nr computed responses of
543 !*                                     Model A and Nq additional computed responses of Model A
544 !* 'chi_2'        output    value of the consistency indicator chi^2
545 !*
546 !* called by: Files

```



```

547 !* calls to: getarg,errmsg                                     *
548 !*                                                           *
549 !*****
550
551 IMPLICIT NONE
552 !Local-----
553 CHARACTER(LEN=128) :: filename
554 CHARACTER(LEN=8)   :: category
555
556 inquire(unit=usupr)
557 ! write(*,900) trim(filename)
558 do
559     read(usupr,*,end=1) category,filename
560
561 ! INPUT FILES FOR MODEL A
562 if (category == 'a_nom') then
563     open(ua_nom,file=filename,status='old',err=100)
564     a_nom = .true.
565 else if (category == 'r_mea') then
566     open(ur_mea,file=filename,status='old',err=100)
567     r_mea = .true.
568 else if (category == 'r_com') then
569     open (ur_com,file=filename,status='old',err=100)
570     r_com = .true.
571 else if (category == 'C_aa') then
572     open(uC_aa,file=filename,status='old',err=100)
573     C_aa = .true.
574 else if (category == 'C_ar') then
575     open(uC_ar,file=filename,status='old',err=100)
576     C_ar = .true.
577 else if (category == 'C_rr') then
578     open(uC_rr,file=filename,status='old',err=100)
579     C_rr = .true.
580 else if (category == 'S_ra') then
581     open(uS_ra,file=filename,status='old',err=100)
582     S_ra = .true.
583
584 ! INPUT FILES FOR MODEL B
585 else if (category == 'q_mea') then
586     open(uq_mea,file=filename,status='old',err=100)
587     q_mea = .true.
588 else if (category == 'q_com') then
589     open (uq_com,file=filename,status='old',err=100)
590     q_com = .true.
591 else if (category == 'C_qq') then
592     open(uC_qq,file=filename,status='old',err=100)
593     C_qq = .true.
594
595 ! INPUT FILES FOR COUPLED MATRICES
596 else if (category == 'C_aq') then
597     open(uC_aq,file=filename,status='old',err=100)
598     C_aq = .true.
599 else if (category == 'C_rq') then
600     open(uC_rq,file=filename,status='old',err=100)
601     C_rq = .true.
602 else if (category == 'S_qa') then
603     open(uS_qa,file=filename,status='old',err=100)
604     S_qa = .true.
605
606 ! OUTPUT FILES FOR MODEL A
607 else if (category == 'a_BE') then

```

```

608     open (ua_BE,file=filename,status='unknown',err=100)
609     a_BE = .true.
610     else if (category == 'r_BE') then
611         open (ur_BE,file=filename,status='unknown',err=100)
612         r_BE = .true.
613     else if (category == 'C_aaBE') then
614         open (uC_aaBE,file=filename,status='unknown',err=100)
615         C_aaBE = .true.
616     else if (category == 'C_rrBE') then
617         open (uC_rrBE,file=filename,status='unknown',err=100)
618         C_rrBE = .true.
619     else if (category == 'C_arBE') then
620         open (uC_arBE,file=filename,status='unknown',err=100)
621         C_arBE = .true.
622     else if (category == 'Crr_comp') then
623         open (uCrr_comp,file=filename,status='unknown',err=100)
624         Crr_comp = .true.
625
626 !   OUTPUT FILES FOR MODEL B
627     else if (category == 'q_BE') then
628         open (uq_BE,file=filename,status='unknown',err=100)
629         q_BE = .true.
630     else if (category == 'C_qqBE') then
631         open (uC_qqBE,file=filename,status='unknown',err=100)
632         C_qqBE = .true.
633     else if (category == 'Cqq_comp') then
634         open (uCqq_comp,file=filename,status='unknown',err=100)
635         Cqq_comp = .true.
636
637 !   OUTPUT FILES FOR COUPLED MATRICES BETWEEN MODELS A & B
638     else if (category == 'C_aqBE') then
639         open (uC_aqBE,file=filename,status='unknown',err=100)
640         C_aqBE = .true.
641     else if (category == 'C_rqBE') then
642         open (uC_rqBE,file=filename,status='unknown',err=100)
643         C_rqBE = .true.
644     else if (category == 'Crq_comp') then
645         open (uCrq_comp,file=filename,status='unknown',err=100)
646         Crq_comp = .true.
647     else if (category == 'chi2') then
648         open (uchi2,file=filename,status='unknown',err=100)
649         chi_2 = .true.
650 !   else
651 !       write(*,1500) category,filename
652     end if
653 end do
654 1 close(usupr)
655
656 !   CHECK TO SEE IF REQUIRED INPUT/OUTPUT FILES ARE OPEN IN THE SUPERFILE
657 !   INPUT
658 if (.not. a_nom) then
659     write(*,1510) 'a_nom';   stop
660 end if
661 if (.not. r_mea) then
662     write(*,1510) 'r_mea';   stop
663 end if
664 if (.not. r_com) then
665     write(*,1510) 'r_com';   stop
666 end if
667 if (.not. C_aa) then
668     write(*,1510) 'C_aa';    stop

```

```

669     end if
670     if (.not. C_ar) then
671         write(*,1510) 'C_ar';    stop
672     end if
673     if (.not. C_rr) then
674         write(*,1510) 'C_rr';    stop
675     end if
676     if (.not. S_ra) then
677         write(*,1510) 'S_ra';    stop
678     end if
679     if (.not. q_mea) then
680         write(*,1510) 'q_mea';    stop
681     end if
682     if (.not. q_com) then
683         write(*,1510) 'q_com';    stop
684     end if
685     if (.not. C_qq) then
686         write(*,1510) 'C_qq';    stop
687     end if
688     if (.not. C_aq) then
689         write(*,1510) 'C_aq';    stop
690     end if
691     if (.not. C_rq) then
692         write(*,1510) 'C_rq';    stop
693     end if
694     if (.not. S_qa) then
695         write(*,1510) 'S_qa';    stop
696     end if
697
698     ! OUTPUT
699     if (.not. Crr_comp) then
700         write(*,1520) 'Crr_comp';stop
701     end if
702     if (.not. a_BE) then
703         write(*,1520) 'a_BE';    stop
704     end if
705     if (.not. r_BE) then
706         write(*,1520) 'r_BE';    stop
707     end if
708     if (.not. C_aaBE) then
709         write(*,1520) 'C_aaBE'; stop
710     end if
711     if (.not. C_rrBE) then
712         write(*,1520) 'C_rrBE'; stop
713     end if
714     if (.not. C_arBE) then
715         write(*,1520) 'C_arBE'; stop
716     end if
717     if (.not. Cqq_comp) then
718         write(*,1520) 'Cqq_comp';stop
719     end if
720     if (.not. q_BE) then
721         write(*,1520) 'q_BE';    stop
722     end if
723     if (.not. C_qqBE) then
724         write(*,1520) 'C_qqBE'; stop
725     end if
726     if (.not. Crq_comp) then
727         write(*,1520) 'Crq_comp';stop
728     end if
729     if (.not. C_aqBE) then

```

```

730     write(*,1520) 'C_aqBE'; stop
731 end if
732 if (.not. C_rqBE) then
733     write(*,1520) 'C_rqBE'; stop
734 end if
735 if (.not. chi_2) then
736     write(*,1520) 'chi_2'; stop
737 end if
738
739 return
740
741 100 lerr = .true.; ierr = 1; call errmsg; stop
742 1500 format(1x,'Invalid file type:',a,' ...skipping',/, &
743           1x,'Specified file name:',a)
744 1510 format(&
745           1x,a,' input file not specified in superfile!')
746 1520 format(&
747           1x,a,' output file not specified in superfile!')
748
749 END SUBROUTINE filescase3
750
751 !+++++
752
753 SUBROUTINE filescase4()
754
755 !*****
756 !*
757 !* open needed files for read and write for case 4: "Two-Model" Case:
758 !* predictive modeling for Model A coupled with Model B.
759 !*
760 !* Open input/output files for multi-pred using super file format
761 !*
762 !* Category      I/O      Description
763 !* -----      -
764 !* 'dims'        input     defines the Case selection and dimensions control
765 !* 'a_nom'        input     nominal values of Na parameters of Model A
766 !* 'r_mea'        input     nominal values of Nr measured responses of Model A
767 !* 'r_com'        input     nominal values of Nr computed responses of Model A
768 !* 'C_aa'         input     covariance matrix of Na parameters of Model A
769 !* 'C_ar'         input     correlations between Na parameters and Nr responses of
770 !*                  Model A
771 !* 'C_rr'         input     covariance matrix of Nr responses of Model A
772 !* 'S_ra'         input     sensitivities of Model A
773 !*
774 !* 'b_nom'        input     nominal values of Nb parameters of Model B
775 !* 'q_mea'        input     nominal values of Nq measured responses of Model B
776 !* 'q_com'        input     nominal values of Nq computed responses of Model B
777 !* 'C_bb'         input     covariance matrix of Nb parameters of Model B
778 !* 'C_bq'         input     correlations between Nb parameters and Nq responses of
779 !*                  Model B
780 !* 'C_qq'         input     covariance matrix of Nq responses of Model B
781 !* 'S_qb'         input     sensitivities of Nq responses of Model B w.r.t. Nb
782 !*                  parameters of Model B
783 !*
784 !* 'C_ab'         input     correlations between Na parameters of Model A and Nb
785 !*                  parameters of Model B
786 !* 'C_aq'         input     correlations between Na parameters of Model A and Nq
787 !*                  responses of Model B
788 !* 'C_br'         input     correlations between Nb parameters of Model B and Nr
789 !*                  responses of Model A
790 !* 'C_rq'         input     correlations between Nr responses of Model A and Nq

```

```

791 !*          responses of Model B          *
792 !* 'S_rb'   input  sensitivities of Nr responses of Model A w.r.t. Nb   *
793 !*          parameters of Model B        *
794 !* 'S_qa'   input  sensitivities of Nq responses of Model B w.r.t. Na   *
795 !*          parameters of Model A        *
796 !*                                           *
797 !* 'a_BE'   output  best-estimate nominal values of Na parameters of   *
798 !*          Model A                      *
799 !* 'r_BE'   output  best-estimate nominal values of Nr responses of   *
800 !*          Model A                      *
801 !* 'C_aaBE' output  predicted covariance matrix of Na parameters of   *
802 !*          Model A                      *
803 !* 'C_rrBE' output  predicted covariance matrix of Nr responses of Model A *
804 !* 'C_arBE' output  predicted correlation matrix between the Na parameters *
805 !*          and Nr responses of Model A  *
806 !* 'Crr_comp' output covariance matrix of Nr computed responses of Model A *
807 !*                                           *
808 !* 'b_BE'   output  best-estimate parameters nominal values of Model B  *
809 !* 'q_BE'   output  best-estimate response nominal values of Model B   *
810 !* 'C_bbBE' output  predicted optimal covariance matrix of parameters of *
811 !*          Model B                    *
812 !* 'C_qqBE' output  predicted covariance matrix of responses of Model B  *
813 !* 'C_bqBE' output  predicted correlation matrix for the parameters     *
814 !*          and responses of Model B   *
815 !* 'Cqq_comp' output covariance matrix for computed responses of Model B *
816 !*                                           *
817 !* 'C_abBE' output  predicted correlation matrix between Na parameters of *
818 !*          Model A and Nb parameters of Model B                       *
819 !* 'C_aqBE' output  predicted correlation matrix between Na parameters of *
820 !*          Model A and Nq responses of Model B                       *
821 !* 'C_brBE' output  predicted correlation matrix between Nb parameters of *
822 !*          Model B and Nr responses of Model A                       *
823 !* 'C_rqBE' output  predicted correlation matrix between Nr responses of *
824 !*          Model A and Nq responses of Model B                       *
825 !* 'Crq_comp' output covariance matrix between Nr computed responses of *
826 !*          Model A and Nq computed responses of Model B             *
827 !* 'chi_2'  output  value of the consistency indicator chi^2          *
828 !*                                           *
829 !* called by: Files          *
830 !* calls to:  getarg,errmsg  *
831 !*                                           *
832 !*****
833
834 IMPLICIT NONE
835 !Local-----
836 CHARACTER(LEN=128) :: filename
837 CHARACTER(LEN=8)  :: category
838
839 inquire(unit=usupr)
840 ! write(*,900) trim(filename)
841 do
842     read(usupr,*,end=1) category,filename
843
844 ! INPUT FILES FOR MODEL A
845 if (category == 'a_nom') then
846     open(ua_nom,file=filename,status='old',err=100)
847     a_nom = .true.
848 else if (category == 'r_mea') then
849     open(ur_mea,file=filename,status='old',err=100)
850     r_mea = .true.
851 else if (category == 'r_com') then

```

```

852     open (ur_com,file=filename,status='old',err=100)
853     r_com = .true.
854     else if (category == 'C_aa') then
855         open(uC_aa,file=filename,status='old',err=100)
856         C_aa = .true.
857     else if (category == 'C_ar') then
858         open(uC_ar,file=filename,status='old',err=100)
859         C_ar = .true.
860     else if (category == 'C_rr') then
861         open(uC_rr,file=filename,status='old',err=100)
862         C_rr = .true.
863     else if (category == 'S_ra') then
864         open(uS_ra,file=filename,status='old',err=100)
865         S_ra = .true.
866
867 ! INPUT FILES FOR MODEL B
868     else if (category == 'b_nom') then
869         open(ub_nom,file=filename,status='old',err=100)
870         b_nom = .true.
871     else if (category == 'q_mea') then
872         open(uq_mea,file=filename,status='old',err=100)
873         q_mea = .true.
874     else if (category == 'q_com') then
875         open (uq_com,file=filename,status='old',err=100)
876         q_com = .true.
877     else if (category == 'C_bb') then
878         open(uC_bb,file=filename,status='old',err=100)
879         C_bb = .true.
880     else if (category == 'C_bq') then
881         open(uC_bq,file=filename,status='old',err=100)
882         C_bq = .true.
883     else if (category == 'C_qq') then
884         open(uC_qq,file=filename,status='old',err=100)
885         C_qq = .true.
886     else if (category == 'S_qb') then
887         open(uS_qb,file=filename,status='old',err=100)
888         S_qb = .true.
889
890 ! INPUT FILES FOR COUPLED MATRICES BETWEEN MODELS A & B
891     else if (category == 'C_ab') then
892         open(uC_ab,file=filename,status='old',err=100)
893         C_ab = .true.
894     else if (category == 'C_aq') then
895         open(uC_aq,file=filename,status='old',err=100)
896         C_aq = .true.
897     else if (category == 'C_br') then
898         open(uC_br,file=filename,status='old',err=100)
899         C_br = .true.
900     else if (category == 'C_rq') then
901         open(uC_rq,file=filename,status='old',err=100)
902         C_rq = .true.
903     else if (category == 'S_rb') then
904         open(uS_rb,file=filename,status='old',err=100)
905         S_rb = .true.
906     else if (category == 'S_qa') then
907         open(uS_qa,file=filename,status='old',err=100)
908         S_qa = .true.
909
910 ! OUTPUT FILES FOR MODEL A
911     else if (category == 'a_BE') then
912         open (ua_BE,file=filename,status='unknown',err=100)

```

```

913     a_BE = .true.
914     else if (category == 'r_BE') then
915         open (ur_BE,file=filename,status='unknown',err=100)
916         r_BE = .true.
917     else if (category == 'C_aaBE') then
918         open (uC_aaBE,file=filename,status='unknown',err=100)
919         C_aaBE = .true.
920     else if (category == 'C_rrBE') then
921         open (uC_rrBE,file=filename,status='unknown',err=100)
922         C_rrBE = .true.
923     else if (category == 'C_arBE') then
924         open (uC_arBE,file=filename,status='unknown',err=100)
925         C_arBE = .true.
926     else if (category == 'Crr_comp') then
927         open (uCrr_comp,file=filename,status='unknown',err=100)
928         Crr_comp = .true.
929
930 ! OUTPUT FILES FOR MODEL B
931     else if (category == 'b_BE') then
932         open (ub_BE,file=filename,status='unknown',err=100)
933         b_BE = .true.
934     else if (category == 'q_BE') then
935         open (uq_BE,file=filename,status='unknown',err=100)
936         q_BE = .true.
937     else if (category == 'C_bbBE') then
938         open (uC_bbBE,file=filename,status='unknown',err=100)
939         C_bbBE = .true.
940     else if (category == 'C_qqBE') then
941         open (uC_qqBE,file=filename,status='unknown',err=100)
942         C_qqBE = .true.
943     else if (category == 'C_bqBE') then
944         open (uC_bqBE,file=filename,status='unknown',err=100)
945         C_bqBE = .true.
946     else if (category == 'Cqq_comp') then
947         open (uCqq_comp,file=filename,status='unknown',err=100)
948         Cqq_comp = .true.
949
950 ! OUTPUT FILES FOR COUPLED MATRICES BETWEEN MODELS A & B
951     else if (category == 'C_abBE') then
952         open (uC_abBE,file=filename,status='unknown',err=100)
953         C_abBE = .true.
954     else if (category == 'C_aqBE') then
955         open (uC_aqBE,file=filename,status='unknown',err=100)
956         C_aqBE = .true.
957     else if (category == 'C_brBE') then
958         open (uC_brBE,file=filename,status='unknown',err=100)
959         C_brBE = .true.
960     else if (category == 'C_rqBE') then
961         open (uC_rqBE,file=filename,status='unknown',err=100)
962         C_rqBE = .true.
963     else if (category == 'Crq_comp') then
964         open (uCrq_comp,file=filename,status='unknown',err=100)
965         Crq_comp = .true.
966     else if (category == 'chi2') then
967         open (uchi2,file=filename,status='unknown',err=100)
968         chi_2 = .true.
969     else
970         write(*,1500) category,filename
971     end if
972 end do
973 1 close(usupr)

```

```

974
975 ! CHECK TO SEE IF REQUIRED INPUT/OUTPUT FILES ARE OPEN IN THE SUPERFILE
976 ! INPUT
977 if (.not. a_nom) then
978   write(*,1510) 'a_nom'; stop
979 end if
980 if (.not. r_mea) then
981   write(*,1510) 'r_mea'; stop
982 end if
983 if (.not. r_com) then
984   write(*,1510) 'r_com'; stop
985 end if
986 if (.not. C_aa) then
987   write(*,1510) 'C_aa'; stop
988 end if
989 if (.not. C_ar) then
990   write(*,1510) 'C_ar'; stop
991 end if
992 if (.not. C_rr) then
993   write(*,1510) 'C_rr'; stop
994 end if
995 if (.not. S_ra) then
996   write(*,1510) 'S_ra'; stop
997 end if
998 if (.not. b_nom) then
999   write(*,1510) 'b_nom'; stop
1000 end if
1001 if (.not. q_mea) then
1002   write(*,1510) 'q_mea'; stop
1003 end if
1004 if (.not. q_com) then
1005   write(*,1510) 'q_com'; stop
1006 end if
1007 if (.not. C_bb) then
1008   write(*,1510) 'C_bb'; stop
1009 end if
1010 if (.not. C_bq) then
1011   write(*,1510) 'C_bq'; stop
1012 end if
1013 if (.not. C_qq) then
1014   write(*,1510) 'C_qq'; stop
1015 end if
1016 if (.not. S_qb) then
1017   write(*,1510) 'S_qb'; stop
1018 end if
1019 if (.not. C_ab) then
1020   write(*,1510) 'C_ab'; stop
1021 end if
1022 if (.not. C_aq) then
1023   write(*,1510) 'C_aq'; stop
1024 end if
1025 if (.not. C_br) then
1026   write(*,1510) 'C_br'; stop
1027 end if
1028 if (.not. C_rq) then
1029   write(*,1510) 'C_rq'; stop
1030 end if
1031 if (.not. S_rb) then
1032   write(*,1510) 'S_rb'; stop
1033 end if
1034 if (.not. S_qa) then

```



```

1035     write(*,1510) 'S_qa';    stop
1036 end if
1037
1038 ! OUTPUT
1039 if (.not. Crr_comp) then
1040     write(*,1520) 'Crr_comp';stop
1041 end if
1042 if (.not. a_BE) then
1043     write(*,1520) 'a_BE';    stop
1044 end if
1045 if (.not. r_BE) then
1046     write(*,1520) 'r_BE';    stop
1047 end if
1048 if (.not. C_aaBE) then
1049     write(*,1520) 'C_aaBE'; stop
1050 end if
1051 if (.not. C_rrBE) then
1052     write(*,1520) 'C_rrBE'; stop
1053 end if
1054 if (.not. C_arBE) then
1055     write(*,1520) 'C_arBE'; stop
1056 end if
1057 if (.not. Cqq_comp) then
1058     write(*,1520) 'Cqq_comp';stop
1059 end if
1060 if (.not. b_BE) then
1061     write(*,1520) 'b_BE';    stop
1062 end if
1063 if (.not. q_BE) then
1064     write(*,1520) 'q_BE';    stop
1065 end if
1066 if (.not. C_bbBE) then
1067     write(*,1520) 'C_bbBE'; stop
1068 end if
1069 if (.not. C_qqBE) then
1070     write(*,1520) 'C_qqBE'; stop
1071 end if
1072 if (.not. C_bqBE) then
1073     write(*,1520) 'C_bqBE'; stop
1074 end if
1075 if (.not. Crq_comp) then
1076     write(*,1520) 'Crq_comp';stop
1077 end if
1078 if (.not. C_abBE) then
1079     write(*,1520) 'C_abBE'; stop
1080 end if
1081 if (.not. C_aqBE) then
1082     write(*,1520) 'C_aqBE'; stop
1083 end if
1084 if (.not. C_brBE) then
1085     write(*,1520) 'C_brBE'; stop
1086 end if
1087 if (.not. C_rqBE) then
1088     write(*,1520) 'C_rqBE'; stop
1089 end if
1090 if (.not. chi_2) then
1091     write(*,1520) 'chi_2';    stop
1092 end if
1093
1094 return
1095

```

```

1096 100 lerr = .true.;ierr = 1;call errmsg;stop
1097 1500 format(1x,'Invalid file type:',a,' ...skipping',/, &
1098         1x,'Specified file name:',a)
1099 1510 format(&
1100         1x,a,' input file not specified in superfile!')
1101 1520 format(&
1102         1x,a,' output file not specified in superfile!')
1103
1104 END SUBROUTINE filescase4
1105
1106 !+++++
1107
1108 END MODULE ModuleFiles
1109

```

8.7 Subroutine ReadInput.f90

```

1  SUBROUTINE ReadInput
2
3  !*****
4  !*
5  !* Reads and processes all multi-pred input parameters and data.
6  !*
7  !*****
8  !Global-----
9  USE ModuleReadWrite
10 USE ModuleLapack, ONLY : is_positive_definite
11
12 IMPLICIT NONE
13 !Local-----
14 INTEGER(I4B)      :: i
15 CHARACTER(LEN=128) :: commnt
16
17 ! read dimensions for the vectors and matrices
18 read (udims,*,err=100) commnt
19 read (udims,*,err=100) Na
20 read (udims,*,err=100) commnt
21 read (udims,*,err=100) Nr
22 write(*, 1010) Na, Nr
23 if(CaseNumber /= 1) then
24   do i=1, 5
25     read(udims,*,err=100) commnt
26   end do
27   read (udims,*,err=100) Nb
28   do i=1, 5
29     read(udims,*,err=100) commnt
30   end do
31   read (udims,*,err=100) Nq
32   if(CaseNumber == 2) then
33     write(*, 1011) Nb
34   else if (CaseNumber == 3) then
35     write(*, 1012) Nq
36   else if (CaseNumber == 4) then
37     write(*, 1013) Nb, Nq
38   end if
39 end if
40 close(udims)
41

```

```

42 write(*, 1000)
43 !-----
44 !READ INPUT FILES FOR MODEL A
45 ! read parameters nominal values
46 write(*, 1020)
47 write(*, 1030)
48 allocate(alpha(Na),stat=alloc_err)
49 call readVectorFromFile(ua_nom,alpha)
50
51 ! read measured responses nominal values
52 write(*, 1040)
53 allocate(rm(Nr),stat=alloc_err)
54 call readVectorFromFile(ur_mea,rm)
55
56 ! read computed responses nominal values
57 write(*, 1050)
58 allocate(rc(Nr),stat=alloc_err)
59 call readVectorFromFile(ur_com,rc)
60
61 ! read covariance matrices for parameter-parameter
62 write(*, 1060)
63 allocate(Caa(Na,Na),stat=alloc_err)
64 call readMatrixFromFile(uC_aa,Caa)
65 ! check if Ca positive definite
66 write(*, 1070)
67 if(is_positive_definite(Caa) == 0) then
68     write(*, 1080)
69 else
70     write(*, 1090); stop
71 end if
72
73 ! read covariance matrices for parameter-response
74 write(*, 1100)
75 allocate(Car(Na,Nr),stat=alloc_err)
76 call readMatrixFromFile(uC_ar,Car)
77
78 ! read covariance matrices for response-response
79 write(*, 1110)
80 allocate(Crr(Nr,Nr),stat=alloc_err)
81 call readMatrixFromFile(uC_rr,Crr)
82 ! check if Crr positive definite
83 write(*, 1120)
84 if(is_positive_definite(Crr) == 0) then
85     write(*, 1130);
86 else
87     write(*, 1140); stop
88 end if
89
90 ! read sensitivities
91 write(*, 1150)
92 allocate(Sra(Nr,Na),stat=alloc_err)
93 call readMatrixFromFile(uS_ra,Sra)
94
95 ! for Case 2: additional read inputs to Case 1
96 if(CaseNumber == 2) then
97     !-----
98     !READ INPUT FILES FOR ADDITIONAL PARAMETERS
99     ! read parameters nominal values
100    write(*, 2000)
101    write(*, 2010)
102    allocate(beta(Nb),stat=alloc_err)

```

```

103     call readVectorFromFile(ub_nom,beta)
104
105     ! read covariance matrices for parameter-parameter
106     write(*, 2020)
107     allocate(Cbb(Nb,Nb),stat=alloc_err)
108     call readMatrixFromFile(uC_bb,Cbb)
109     ! check if Cb positive definite
110     write(*, 2030)
111     if(is_positive_definite(Cbb) == 0) then
112         write(*, 2040)
113     else
114         write(*, 2050); stop
115     end if
116
117     !-----
118     !READ INPUT FILES FOR THE COUPLED MATRICES
119     ! read correlations between parameters a and b
120     write(*, 2060)
121     write(*, 2070)
122     allocate(Cab(Na,Nb),stat=alloc_err)
123     call readMatrixFromFile(uC_ab,Cab)
124
125     ! read correlations between parameters b and responses r
126     write(*, 2080)
127     allocate(Cbr(Nb,Nr),stat=alloc_err)
128     call readMatrixFromFile(uC_br,Cbr)
129
130     ! read sensitivities of responses r and parameters b
131     write(*, 2090)
132     allocate(Srb(Nr,Nb),stat=alloc_err)
133     call readMatrixFromFile(uS_rb,Srb)
134
135     end if !case 2
136
137     ! for Case 3: additional read inputs to Case 1
138     if(CaseNumber == 3) then
139         !-----
140         !READ INPUT FILES FOR ADDITIONAL RESPONSES
141         write(*, 3000)
142
143         ! read measured responses nominal values
144         write(*, 3010)
145         allocate(qm(Nq),stat=alloc_err)
146         call readVectorFromFile(uq_mea,qm)
147
148         ! read computed responses nominal values
149         write(*, 3020)
150         allocate(qc(Nq),stat=alloc_err)
151         call readVectorFromFile(uq_com,qc)
152
153         ! read covariance matrices for response-response
154         write(*, 3030)
155         allocate(Cqq(Nq,Nq),stat=alloc_err)
156         call readMatrixFromFile(uC_qq,Cqq)
157         ! check if Cqq positive definite
158         write(*, 3040)
159         if(is_positive_definite(Cqq) == 0) then
160             write(*, 3050)
161         else
162             write(*, 3060); stop
163         end if

```

```

164
165 !-----
166 !READ INPUT FILES FOR THE COUPLED MATRICES
167 write(*, 3070)
168
169 ! read correlations between parameters a and responses q
170 write(*, 3080)
171 allocate(Caq(Na,Nq),stat=alloc_err)
172 call readMatrixFromFile(uC_aq,Caq)
173
174 ! read correlations between responses r and responses q
175 write(*, 3090)
176 allocate(Crq(Nr,Nq),stat=alloc_err)
177 call readMatrixFromFile(uC_rq,Crq)
178
179 ! read sensitivities of responses q and parameters a
180 write(*, 3100)
181 allocate(Sqa(Nq,Na),stat=alloc_err)
182 call readMatrixFromFile(uS_qa,Sqa)
183 end if !case 3
184
185 ! for Case 4: additional read inputs to Case 1
186 if(CaseNumber == 4) then
187 !-----
188 !READ INPUT FILES FOR MODEL B
189 ! read parameters nominal values
190 write(*, 4000)
191 write(*, 4010)
192 allocate(beta(Nb),stat=alloc_err)
193 call readVectorFromFile(ub_nom,beta)
194
195 ! read measured responses nominal values
196 write(*, 4020)
197 allocate(qm(Nq),stat=alloc_err)
198 call readVectorFromFile(uq_mea,qm)
199
200 ! read computed responses nominal values
201 write(*, 4030)
202 allocate(qc(Nq),stat=alloc_err)
203 call readVectorFromFile(uq_com,qc)
204
205 ! read covariance matrices for parameter-parameter
206 write(*, 4040)
207 allocate(Cbb(Nb,Nb),stat=alloc_err)
208 call readMatrixFromFile(uC_bb,Cbb)
209 ! check if Cb positive definite
210 write(*, 4050)
211 if(is_positive_definite(Cbb) == 0) then
212     write(*, 4060)
213 else
214     write(*, 4070); stop
215 end if
216
217 ! read covariance matrices for parameter-response
218 write(*, 4080)
219 allocate(Cbq(Nb,Nq),stat=alloc_err)
220 call readMatrixFromFile(uC_bq,Cbq)
221
222 ! read covariance matrices for response-response
223 write(*, 4090)
224 allocate(Cqq(Nq,Nq),stat=alloc_err)

```

```

225     call readMatrixFromFile(uC_qq,Cqq)
226     ! check if Cqq positive definite
227     write(*, 4100)
228     if(is_positive_definite(Cqq) == 0) then
229         write(*, 4110)
230     else
231         write(*, 4120); stop
232     end if
233
234     ! read sensitivities
235     write(*, 4130)
236     allocate(Sqb(Nq,Nb),stat=alloc_err)
237     call readMatrixFromFile(uS_qb,Sqb)
238
239     !-----
240     !READ INPUT FILES FOR THE COUPLED MATRICES BETWEEN MODELS A & B
241     ! read correlations between parameters a and b
242     write(*, 4140)
243     write(*, 4150)
244     allocate(Cab(Na,Nb),stat=alloc_err)
245     call readMatrixFromFile(uC_ab,Cab)
246
247     ! read correlations between parameters a and responses q
248     write(*, 4160)
249     allocate(Caq(Na,Nq),stat=alloc_err)
250     call readMatrixFromFile(uC_aq,Caq)
251
252     ! read correlations between parameters b and responses r
253     write(*, 4170)
254     allocate(Cbr(Nb,Nr),stat=alloc_err)
255     call readMatrixFromFile(uC_br,Cbr)
256
257     ! read correlations between responses r and responses q
258     write(*, 4180)
259     allocate(Crq(Nr,Nq),stat=alloc_err)
260     call readMatrixFromFile(uC_rq,Crq)
261
262     ! read sensitivities of responses r and parameters b
263     write(*, 4190)
264     allocate(Srb(Nr,Nb),stat=alloc_err)
265     call readMatrixFromFile(uS_rb,Srb)
266
267     ! read sensitivities of responses q and parameters a
268     write(*, 4200)
269     allocate(Sqa(Nq,Na),stat=alloc_err)
270     call readMatrixFromFile(uS_qa,Sqa)
271 end if !case 4
272
273 return
274     100 lerr = .true.;ierr = 1;call errmsg;stop
275     1010 format(/, &
276         1x,'For model A:   Na =',i5,'      Nr =', i5)
277     1011 format(&
278         1x,'Nb =', i5,' parameters added to the Na parameters for Model A',a)
279     1012 format(&
280         1x,'Nq =', i5,' responses added to the Nr reponses for Model A',a)
281     1013 format(&
282         1x,'For model B:   Nb =',i5,'      Nq =', i5)
283     1000 format(/, &
284         1x,'----- Read Inputs into Vectors/Matrices -----',a)
285     ! format reading for model A

```

```

286 1020 format(&
287 1x, ':: Reading inputs for Model A:',a)
288 1030 format(&
289 1x, 'Reading the input for the parameter vector ..... alpha',a)
290 1040 format(&
291 1x, 'Reading the input for measured response vector ..... rm',a)
292 1050 format(&
293 1x, 'Reading the input for computed response vector ..... rc',a)
294 1060 format(&
295 1x, 'Reading the input for covariance matrix of parameters ..... Caa',a)
296 1070 format(&
297 1x, '      Factorizing matrix Caa -- to test if positive definite.',a)
298 1080 format(&
299 1x, '      OK, Caa is positive definite.',a)
300 1090 format(&
301 1x, 'ERROR: The input covariance matrix Caa is not positive'      &
302 'definite, check the input file',a)
303 1100 format(&
304 1x, 'Reading the input for the correlation matrix ..... Car',a)
305 1110 format(&
306 1x, 'Reading the input for covariance matrix of responses ..... Crr',a)
307 1120 format(&
308 1x, '      Factorizing matrix Crr -- to test if positive definite.',a)
309 1130 format(&
310 1x, '      OK, Crr is positive definite.',a)
311 1140 format(&
312 1x, 'ERROR: The input covariance matrix Crr is not positive'      &
313 'definite, check the input file',a)
314 1150 format(&
315 1x, 'Reading the input for the sensitivity matrix ..... Sra',a)
316
317 ! Case 2 -- format reading for Nb additional parameters
318 2000 format(/,&
319 1x, ':: Reading inputs for additional model parameters for Case 2:',a)
320 2010 format(&
321 1x, 'Reading the input for the parameter vector .....beta',a)
322 2020 format(&
323 1x, 'Reading the input for covariance matrix of parameters ..... Cbb',a)
324 2030 format(&
325 1x, '      Factorizing matrix Cbb -- to test if positive definite.',a)
326 2040 format(&
327 1x, '      OK, Cbb is positive definite.',a)
328 2050 format(&
329 1x, 'ERROR: The input covariance matrix Cbb is not positive'      &
330 'definite, check the input file',a)
331
332 ! Case 2 -- format reading for the coupled matrices
333 2060 format(/,&
334 1x, ':: Reading inputs for coupled matrices for Case 2:',a)
335 2070 format(&
336 1x, 'Reading the input for matrix ..... Cab',a)
337 2080 format(&
338 1x, 'Reading the input for matrix ..... Cbr',a)
339 2090 format(&
340 1x, 'Reading the input for matrix ..... Srb',a)
341
342 ! Case 3 -- format reading for additional responses
343 3000 format(/,&
344 1x, ':: Reading inputs for additional model responses for Case 3: ',a)
345 3010 format(&
346 1x, 'Reading the input for additional measured response vector ..... qm',a)

```

```

347 3020 format(&
348 1x,'Reading the input for additional computed response vector ..... qc',a)
349 3030 format(&
350 1x,'Reading the input for covariance matrix of responses ..... Cqq',a)
351 3040 format(&
352 1x,'      Factorizing matrix Cqq -- to test if positive definite.',a)
353 3050 format(&
354 1x,'      OK, Cqq is positive definite.',a)
355 3060 format(&
356 1x,'ERROR: The input covariance matrix Cqq is not positive'      &
357      'definite, check the input file',a)
358
359 ! Case 3 -- format reading for the coupled matrices
360 3070 format(/,&
361 1x,':: Reading inputs for coupled matrices for Case 3:',a)
362 3080 format(&
363 1x,'Reading the input for matrix ..... Caq',a)
364 3090 format(&
365 1x,'Reading the input for matrix ..... Crq',a)
366 3100 format(&
367 1x,'Reading the input for matrix ..... Sqa',a)
368
369 ! Case 4 -- format reading for model B
370 4000 format(/,&
371 1x,':: Reading inputs for Model B: ',a)
372 4010 format(&
373 1x,'Reading the input for the parameter vector .....beta',a)
374 4020 format(&
375 1x,'Reading the input for measured response vector ..... qm',a)
376 4030 format(&
377 1x,'Reading the input for computed response vector ..... qc',a)
378 4040 format(&
379 1x,'Reading the input for covariance matrix of parameters ..... Cbb',a)
380 4050 format(&
381 1x,'      Factorizing matrix Cbb -- to test if positive definite.',a)
382 4060 format(&
383 1x,'      OK, Cbb is positive definite.',a)
384 4070 format(&
385 1x,'ERROR: The input covariance matrix Cbb is not positive'      &
386      'definite, check the input file',a)
387 4080 format(&
388 1x,'Reading the input for the correlation matrix ..... Cbq',a)
389 4090 format(&
390 1x,'Reading the input for covariance matrix of responses ..... Cqq',a)
391 4100 format(&
392 1x,'      Factorizing matrix Cqq -- to test if positive definite.',a)
393 4110 format(&
394 1x,'      OK, Cqq is positive definite.',a)
395 4120 format(&
396 1x,'ERROR: The input covariance matrix Cqq is not positive'      &
397      'definite, check the input file',a)
398 4130 format(&
399 1x,'Reading the input for the sensitivity matrix ..... Sqb',a)
400
401 ! Case 4 -- format reading for the coupled matrices between models A & B
402 4140 format(/,&
403 1x,':: Reading inputs for coupled matrices between Models A & B:',a)
404 4150 format(&
405 1x,'Reading the input for matrix ..... Cab',a)
406 4160 format(&
407 1x,'Reading the input for matrix ..... Caq',a)

```



```

408 4170 format(&
409     1x,'Reading the input for matrix ..... Cbr',a)
410 4180 format(&
411     1x,'Reading the input for matrix ..... Crq',a)
412 4190 format(&
413     1x,'Reading the input for matrix ..... Srb',a)
414 4200 format(&
415     1x,'Reading the input for matrix ..... Sqa',a)
416
417 END SUBROUTINE ReadInput

```

8.8 Module ModuleReadWrite.f90

```

1  MODULE ModuleReadWrite
2  !*****
3  !*
4  !* Module ModuleReadWrite encapsulates subroutines for:
5  !*           readMatrixFromFile (UnitNum, Array)
6  !*           readVectorFromFile (UnitNum, Array)
7  !*           writeVectorToFile (UnitNum, Array)
8  !*           writeMatrixToFile (UnitNum, Array)
9  !*
10 !*****
11
12 !Global-----
13 USE ModuleErrors
14 USE ModuleIO
15 USE ModuleGlobalParameters
16
17 IMPLICIT NONE
18
19 CONTAINS
20
21 SUBROUTINE readMatrixFromFile (UnitNum, Array)
22
23 !*****
24 !*
25 !* read a 2D sparse matrix and to produce a normal one
26 !*
27 !*****
28 IMPLICIT NONE
29
30 ! Arguments-----
31 INTEGER, intent(in) :: UnitNum
32 REAL(DP), ALLOCATABLE :: Array(:, :)
33 !Local-----
34 INTEGER(I4B) :: numRows, numCols, NonzeroElements, Nrow, Ncol
35 INTEGER(I4B) :: i, j, k
36 INTEGER(I4B) :: IOstatus
37 REAL(DP) :: val
38
39 Nrow = size(Array,1)
40 Ncol = size(Array,2)
41 rewind(UnitNum)
42 read (UnitNum,*,err=100) numRows,numCols,NonzeroElements
43 if(numRows /= Nrow .OR. numCols /= Ncol) then
44     write(*, 200) Nrow,Ncol; stop

```

```

45     end if
46     Array = 0.0
47     do k=1, NonzeroElements
48         read (UnitNum,*,IOSTAT=IOstatus) i, j, Array(i,j)
49         if(IOstatus > 0) then !something is wrong, like illegal values
50             Write(*, 210) K+1
51             stop
52         else if (IOstatus < 0) then !end of file reached
53             Write(*, 220) NonzeroElements, K-1
54             stop
55         else !normal reading
56             if(i > Nrow .OR. j > Ncol) then
57                 write(*, 300); stop
58             end if
59         end if
60     end do
61     ! read one more line, to check if end of file reached.
62     ! if not, then the input data file has a inconsistency.
63     ! It has more data than expected.
64     read (UnitNum,*,IOSTAT=IOstatus) i,j,val
65     if(IOstatus == 0) then
66         write(*, 310)NonzeroElements
67         stop
68     end if
69
70     close(UnitNum)
71     return
72     ! bail out for read error
73     100 lerr = .true.; ierr = 2; call ermsg; stop
74     200 format(&
75         1x,'Error: this matrix should have a dimension of ',i8,' -by-', i8)
76     210 format(&
77         1x,'Error: something is wrong while reading line ',i8,'. Maybe illegal',/,&
78         ' data; Check the input.',a)
79     220 format(&
80         1x,'Error: end of file reached earlier than expected. It is expected ',/,&
81         ' to read Nz = ', i8,' nonzero elements, but only read ',i8',/,&
82         ' of them. Check the input.',a)
83     300 format(&
84         1x,'Error: the index for nonzero elements exceeds the matrix size.',a)
85     310 format(&
86         1x,'Error: something is wrong with the input data file. It seems the',/,&
87         ' actual number of nonzero elements in the file exceeds ' ,/,&
88         ' that defined in the 1st line: Nz = ',i8,'. Check the ' ,/,&
89         ' input.',a)
90
91     END SUBROUTINE readMatrixFromFile
92
93     !+++++
94
95     SUBROUTINE readVectorFromFile(UnitNum, Array)
96
97     !*****
98     !*
99     !* read a 1D sparse matrix and to produce a normal one
100    !*
101    !*****
102    IMPLICIT NONE
103
104    ! Arguments-----
105    INTEGER, intent(in) :: UnitNum

```

```

106 REAL(DP), ALLOCATABLE :: Array(:)
107 !Local-----
108 INTEGER(I4B)          :: numRows, numCols, NonzeroElements, Nrow, Ncol
109 INTEGER(I4B)          :: i, j, k
110 INTEGER(I4B)          :: IOstatus
111 REAL(DP)              :: val
112
113 Nrow = size(Array,1)
114 Ncol = 1
115 rewind(UnitNum)
116 read (UnitNum,*,err=100) numRows,numCols,NonzeroElements
117 Array = 0.0
118 if(numRows /= Nrow .OR. numCols /= 1) then
119     write(*, 200) Nrow,Ncol; stop
120 end if
121 do k=1, NonzeroElements
122     read (UnitNum,*,IOSTAT=IOstatus) i, j, Array(i)
123     if(IOstatus > 0) then !something is wrong, like illegal values
124         Write(*, 210) K+1
125         stop
126     else if (IOstatus < 0) then !end of file reached earlier than expected
127         Write(*, 220) NonzeroElements, K-1
128         stop
129     else !normal reading
130         if(i > Nrow .OR. j > Ncol) then
131             write(*, 300); stop
132         end if
133     end if
134 end do
135 ! read one more line, to check if end of file reached
136 ! if not, then the input data file has a inconsistency.
137 ! It has more data than expected.
138 read (UnitNum,*,IOSTAT=IOstatus) i,j,val
139 if(IOstatus == 0) then
140     write(*, 310)NonzeroElements
141     stop
142 end if
143
144 close(UnitNum)
145 return
146 ! bail out for read error
147 100 lerr = .true.;ierr = 2;call errmsg;stop
148 200 format(&
149     1x,'Error: this vector should have a dimension of ',i8,' -by-', i8)
150 210 format(&
151     1x,'Error: something is wrong while reading line ',i8,'. Maybe illegal',/,&
152     ' data; Check the input.',a)
153 220 format(&
154     1x,'Error: end of file reached earlier than expected. It is expected ',/,&
155     ' to read Nz = ', i8,' nonzero elements, but only read ',i8',/,&
156     ' of them. Check the input.',a)
157 300 format(&
158     1x,'Error: the index for nonzero elements exceeds the vector size.',a)
159 310 format(&
160     1x,'Error: something is wrong with the input data file. It seems the',/,&
161     ' actual number of nonzero elements in the file exceeds ' ,/,&
162     ' that defined in the 1st line: Nz = ',i8,'. Check the ' ,/,&
163     ' input.',a)
164
165 END SUBROUTINE readVectorFromFile
166

```

```

167 !+++++
168
169 SUBROUTINE writeVectorToFile(UnitNum, Array)
170
171 !*****
172 !*
173 !* write a full vector into a file as a sparse one
174 !*
175 !*****
176 IMPLICIT NONE
177
178 ! Arguments-----
179 INTEGER, intent(in) :: UnitNum
180 REAL(DP), ALLOCATABLE :: Array(:)
181 !Local-----
182 INTEGER(I4B) :: numRows, numCols, nZ
183 INTEGER(I4B) :: i, j
184
185 numRows = size(Array,1)
186 numCols = 1
187 nZ = 0
188
189 ! write the 1st line for numRows, numCols and the number of nonzero elements
190 do i=1, numRows
191     do j=1, numCols
192         if(Array(i)/= 0.0) then
193             nZ = nZ + 1
194         end if
195     end do
196 end do
197 write(UnitNum,1000,err=200) numRows, numCols, nZ
198
199 ! write the nonzero elements with their associated row and colume coordinates
200 do i=1, numRows
201     do j=1, numCols
202         if(Array(i)/= 0.0) then
203             write(UnitNum,2000,err=200) i, j, Array(i)
204         end if
205     end do
206 end do
207
208 close(UnitNum)
209 return
210 ! bail out for write error
211 200 lerr = .true.;ierr = 3;call errmsg;stop
212 1000 format(3I5)
213 2000 format(2I5,ES20.8)
214
215 END SUBROUTINE writeVectorToFile
216
217 !+++++
218
219 SUBROUTINE writeMatrixToFile(UnitNum, Array)
220
221 !*****
222 !*
223 !* write a 2D full matrix into a file as a sparse one
224 !*
225 !*****
226 IMPLICIT NONE
227

```

```

228 ! Arguments-----
229 INTEGER, intent(in) :: UnitNum
230 REAL(DP), ALLOCATABLE :: Array(:, :)
231 !Local-----
232 INTEGER(I4B) :: numRows, numCols, nZ
233 INTEGER(I4B) :: i, j
234
235 numRows = size(Array,1)
236 numCols = size(Array,2)
237 nZ = 0
238
239 ! write the 1st line for numRows, numCols and the number of nonzero elements
240 do i=1, numRows
241     do j=1, numCols
242         if(Array(i,j)/= 0.0) then
243             nZ = nZ + 1
244         end if
245     end do
246 end do
247 write(UnitNum,1000,err=200) numRows, numCols, nZ
248
249 ! write the nonzero elements with their associated row and colume coordinates
250 do i=1, numRows
251     do j=1, numCols
252         if(Array(i,j)/= 0.0) then
253             write(UnitNum,2000,err=200) i, j, Array(i,j)
254         end if
255     end do
256 end do
257
258 close(UnitNum)
259 return
260 ! bail out for write error
261 200 lerr = .true.;ierr = 3;call errmsg;stop
262 1000 format(I5,I5,I8)
263 2000 format(2I5,ES20.8)
264
265 END SUBROUTINE writeMatrixToFile
266
267 ! =====
268
269 LOGICAL FUNCTION is_NAN_or_Infinity_M(A) result(tf)
270
271 ! -- check the matrix component values not to be Inifinite or NAN
272 ! -- where A is a 2D matrix
273 ! -- developed by University of South Carolina.
274 !
275 INTEGER, PARAMETER :: DP = KIND(1.0D0)
276 INTEGER, PARAMETER :: I4B = SELECTED_INT_KIND(9)
277 ! Arguments-----
278 REAL(DP), dimension(:, :), intent(in) :: A
279 ! LOGICAL :: tf
280 ! Local-----
281 REAL(DP) :: infinity
282 INTEGER(I4B) :: numRows, numCols, i, j
283
284 infinity = 1.e100_dp
285 tf = .false.
286
287 numRows = size(A,1)
288 numCols = size(A,2)

```

```

289
290     do i=1, numRows
291         do j=1, numCols
292             !check if infinity
293             if(A(i,j) > infinity) then
294                 tf = .true.
295             end if
296             ! check if NAN
297             if(A(i,j) /= A(i,j)) then
298                 tf = .true.
299             end if
300         end do
301     end do
302
303 END FUNCTION is_NAN_or_Infinity_M
304
305 ! =====
306
307 LOGICAL FUNCTION is_NAN_or_Infinity_V(A) result(tf)
308
309 ! -- check the matrix component values not to be Inifinite or NAN
310 ! -- where A is a Vector
311 ! -- developed by University of South Carolina.
312 !
313     INTEGER, PARAMETER :: DP = KIND(1.0D0)
314     INTEGER, PARAMETER :: I4B = SELECTED_INT_KIND(9)
315 ! Arguments-----
316 REAL(DP), dimension(:), intent(in)    :: A
317 ! LOGICAL                                :: tf
318 ! Local-----
319 REAL(DP)                                :: infinity
320 INTEGER(I4B)                             :: numRows, i, j
321
322     infinity = 1.e100_dp
323     tf = .false.
324
325     numRows = size(A,1)
326
327     do i=1, numRows
328         !check if infinity
329         if(A(i) > infinity) then
330             tf = .true.
331         end if
332         ! check if NAN
333         if(A(i) /= A(i)) then
334             tf = .true.
335         end if
336     end do
337
338 END FUNCTION is_NAN_or_Infinity_V
339
340 END MODULE ModuleReadWrite

```

8.9 Subroutine MultiPredSolver.f90

```
1 SUBROUTINE MultiPredSolver
2
3 !*****
4 !*
5 !* apply the Multi-Pred formulations and solve
6 !*
7 !*****
8
9 !Global-----
10 USE ModuleMultiPred
11
12 IMPLICIT NONE
13
14 ! call Multi-Pred solver for Case 1: Modle A solely
15 if(CaseNumber == 1) then
16     call solvercase1
17 end if
18
19 ! call Multi-Pred solver for Case 2: Modle A with additional Nb parameters
20 if(CaseNumber == 2) then
21     call solvercase2
22 end if
23
24 ! call Multi-Pred solver for Case 3: Modle A with additional Nc responses
25 if(CaseNumber == 3) then
26     call solvercase3
27 end if
28
29 ! call Multi-Pred solver for Case 4: Coupled Model A and Model B
30 if(CaseNumber == 4) then
31     call solvercase4
32 end if
33
34
35 END SUBROUTINE MultiPredSolver
```

8.10 Module ModuleMultiPred.f90

```
1 MODULE ModuleMultiPred
2 !*****
3 !*
4 !* Module ModuleMultiPred encapsulates subroutines for:
5 !* solvercase1 ()
6 !* solvercase2 ()
7 !* solvercase3 ()
8 !* solvercase4 ()
9 !*
10 !*****
11
12 !Global-----
13 USE ModuleReadWrite
14 USE ModuleLapack
15
16 IMPLICIT NONE
17
```

```

18 CONTAINS
19
20 !+++++
21 SUBROUTINE solvercase1()
22 !
23 ! solver for Casee 1: predictive modeling for Model A solely
24 !
25 !Local-----
26 REAL(DP), ALLOCATABLE :: SraCaa(:,,:), SraT(:,:)
27 REAL(DP), ALLOCATABLE :: Xa(:,,:), Xr(:,,:), Drr(:,,:), D11(:,,:), rd(:)
28 REAL(DP), ALLOCATABLE :: CarT(:,:)
29 REAL(DP), ALLOCATABLE :: Drrinv(:,:)
30 REAL(DP), ALLOCATABLE :: XaD11(:,,:), XrD11(:,,:)
31 REAL(DP), ALLOCATABLE :: D11rd(:)
32
33 write(*,1000)
34
35 !start with computing covariance matrix of Crr_comp
36 !for the computed responses.
37
38 ! Crrcomp = Sra*Caa*Sra'
39 allocate(Crrcomp(Nr,Nr),stat=alloc_err)
40 allocate(SraCaa(Nr,Na),stat=alloc_err)
41 allocate(SraT(Na,Nr),stat=alloc_err)
42 Crrcomp = 0.0; SraCaa = 0.0
43 SraT = 0.0;
44
45 write(*, 1010)
46 SraCaa = multipMM(Sra,Caa)
47 SraT = transpose(Sra)
48 Crrcomp = multipMM(SraCaa,SraT)
49 call writeMatrixToFile(uCrr_comp,Crrcomp)
50
51 ! define intermediate quantities and initialize
52 write(*, 1020)
53 allocate(Xa(Na,Nr),stat=alloc_err)
54 allocate(Xr(Nr,Nr),stat=alloc_err)
55 allocate(Drr(Nr,Nr),stat=alloc_err)
56 allocate(D11(Nr,Nr),stat=alloc_err)
57 allocate(rd(Nr),stat=alloc_err)
58 Xa = 0.0
59 Xr = 0.0
60 Drr = 0.0
61 D11 = 0.0
62 rd = 0.0
63
64 ! Xa=Caa*Sra'-Car
65 Xa = multipMM(Caa,SraT)
66 Xa = Xa - Car
67
68 ! Xr=Car'*Sra'-Crr
69 allocate(CarT(Nr,Na),stat=alloc_err)
70 CarT = 0.0
71 CarT = transpose(Car)
72
73 Xr = multipMM(CarT,SraT)
74 Xr = Xr - Crr
75
76 ! Drr=Sra*Xa-Xr
77 Drr = multipMM(Sra,Xa)
78 Drr = Drr - Xr

```



```

79
80 ! define rd
81 rd = rc -rm
82
83 ! compute Drr^-1, D22^-1
84 allocate(Drrinv(Nr,Nr),stat=alloc_err)
85 Drrinv = 0.0
86 Drrinv = inv(Drr)
87 if(is_NAN_or_Infinity_M(Drr)) then
88     write(*, 1030)
89 end if
90 if(is_NAN_or_Infinity_M(Drrinv)) then
91     write(*, 1040)
92 end if
93
94 ! D11=Drr^-1
95 D11 = Drrinv
96
97 ! best estimated mean values for aBE, rBE
98 write(*, 1050)
99 allocate(aBE(Na),stat=alloc_err)
100 allocate(rBE(Nr),stat=alloc_err)
101 aBE = 0.0
102 rBE = 0.0
103
104 ! aBE = alpha-[Xa*D11]*rd
105 allocate(XaD11(Na,Nr),stat=alloc_err)
106 XaD11 = 0.0
107 XaD11 = multipMM(Xa,D11)
108
109 aBE = multipMV(XaD11,rd)
110 aBE = alpha - aBE
111 call writeVectorToFile(ua_BE,aBE)
112
113 ! rBE = rm-[Xr*D11]*rd
114 write(*, 1060)
115 allocate(XrD11(Nr,Nr),stat=alloc_err)
116 XrD11 = 0.0
117 XrD11 = multipMM(Xr,D11)
118
119 rBE = multipMV(XrD11,rd)
120 rBE = rm - rBE
121 call writeVectorToFile(ur_BE,rBE)
122
123 !calculate coviances for responses and parameters
124 allocate(CaaBE(Na,Na),stat=alloc_err)
125 allocate(CrrBE(Nr,Nr),stat=alloc_err)
126 allocate(CarBE(Na,Nr),stat=alloc_err)
127 CaaBE = 0.0; CrrBE = 0.0; CarBE = 0.0
128
129 ! CaaBE = Caa - [Xa*(D11*Xa')]
130 write(*, 1070)
131 CaaBE = multipMM(Xa,transpose(XaD11))
132 CaaBE = Caa - CaaBE
133 call writeMatrixToFile(uC_aaBE,CaaBE)
134
135 ! CrrBE = Crr - [Xr*(D11*Xr')]
136 write(*, 1080)
137 CrrBE = multipMM(Xr,transpose(XrD11))
138 CrrBE = Crr - CrrBE
139 call writeMatrixToFile(uC_rrBE,CrrBE)

```

```

140
141 ! CarBE = Car - [Xa*(D11*Xr')]
142 write(*, 1090)
143 CarBE = multipMM(Xa,transpose(XrD11))
144 CarBE = Car - CarBE
145 call writeMatrixToFile(uC_arBE,CarBE)
146
147 !calculate the "consistency indicator" chi^2
148 allocate(D11rd(Nr),stat=alloc_err)
149 D11rd = 0.0
150 chi2 = 0.0
151
152 D11rd = multipMV(D11,rd)
153 chi2 = multipVV(rd,D11rd)
154 write(uchi2,3500,err=200) chi2, chi2/Nr
155
156 write(*,3600)
157 return
158 200 lerr = .true.;ierr = 3;call errmsg;stop
159 1000 format(/, &
160 1x,'----- Multi-pred Solving & Output -----')
161 1010 format(&
162 1x,'computing Crrcomp = Sra*Caa*Sra',a)
163 1020 format(&
164 1x,'computing Xa, Xr, D11, Drr, Drr^-1, rd',a)
165 1030 format(&
166 1x,'ERROR: Infinite or NAN in computing Drr, where ' &
167 'Drr=Sra*Xa-(CarT)*(SraT)-Crr',a)
168 1040 format(&
169 1x,'ERROR: Infinite or NAN in computing Drr^-1, where ' &
170 'Drr=Sra*Xa-(CarT)*(SraT)-Crr',a)
171 1050 format(&
172 1x,'computing aBE = alpha-[Caa*Sra"-Car]*Drr^-1*rd',a)
173 1060 format(&
174 1x,'computing rBE = rm-[(Car")*(Sra")-Crr]*Drr^-1*rd',a)
175 1070 format(&
176 1x,'computing CaaBE = Caa-[Caa*Sra"-Car]*Drr^-1*[Caa*Sra"-Car]',a)
177 1080 format(&
178 1x,'computing CrrBE = Crr-[Car"*Sra"-Crr]*Drr^-1*[Car"*Sra"-Crr]',a)
179 1090 format(&
180 1x,'computing CarBE = Car-[Caa*Sra"-Car]*Drr^-1*[Car"*Sra"-Crr]',a)
181 3500 format(&
182 'chi^2 = ' ,F8.3/, &
183 'chi^2_d = (chi^2)/(number of responses) = ' ,F8.3)
184 3600 format(/, &
185 1x,'done.')
```

186

```

187 END SUBROUTINE solvercase1
188
189 !+++++
```

190

```

191 SUBROUTINE solvercase2()
192 !
193 ! solver for Casee 2: predictive modeling for Model A with Nb additional
194 ! parameters, but no additional responses
195 !
196 !Local-----
197 REAL(DP), ALLOCATABLE :: SraCaa(:,,:), SraT(:,,:), SraCab(:,,:), SrbT(:,,:), &
198 SrbCbb(:,:)
199 REAL(DP), ALLOCATABLE :: Xa(:,,:), Xb(:,,:), Xr(:,,:), Drr(:,,:), D11(:,,:), rd(:)
200 REAL(DP), ALLOCATABLE :: CarT(:,,:), CbrT(:,:)
```

```

201 REAL(DP), ALLOCATABLE :: Drrinv(:,,:), XaD11(:,,:), XbD11(:,,:), XrD11(:,,:)
202 REAL(DP), ALLOCATABLE :: D11rd(:)
203
204 write(*,1000)
205
206 !start with computing covariance matrix of Crr_comp
207 !for the computed responses.
208
209 ! Crrcomp = Sra*Caa*Sra'+2*Sra*Cab*Srb'+Srb*Cbb*Srb'
210 allocate(Crrcomp(Nr,Nr),stat=alloc_err)
211 allocate(SraCaa(Nr,Na),stat=alloc_err)
212 allocate(SraT(Na,Nr),stat=alloc_err)
213 allocate(SraCab(Nr,Nb),stat=alloc_err)
214 allocate(SrbT(Nb,Nr),stat=alloc_err)
215 allocate(SrbCbb(Nr,Nb),stat=alloc_err)
216 Crrcomp = 0.0; SraCaa = 0.0
217 SraT = 0.0; SraCab = 0.0
218 SrbT = 0.0; SrbCbb = 0.0
219
220 write(*, 1010)
221 SraCaa = multipMM(Sra,Caa)
222 SraT = transpose(Sra)
223 Crrcomp = multipMM(SraCaa,SraT)
224 SraCab = multipMM(Sra,Cab)
225 SrbT = transpose(Srb)
226 Crrcomp = Crrcomp + 2 * multipMM(SraCab,SrbT)
227 SrbCbb = multipMM(Srb,Cbb)
228 Crrcomp = Crrcomp + multipMM(SrbCbb,SrbT)
229 call writeMatrixToFile(uCrr_comp,Crrcomp)
230
231 ! define intermediate quantities and initialize
232 allocate(Xa(Na,Nr),stat=alloc_err)
233 allocate(Xb(Nb,Nr),stat=alloc_err)
234 allocate(Xr(Nr,Nr),stat=alloc_err)
235 allocate(Drr(Nr,Nr),stat=alloc_err)
236 allocate(D11(Nr,Nr),stat=alloc_err)
237 allocate(rd(Nr),stat=alloc_err)
238 Xa = 0.0
239 Xb = 0.0
240 Xr = 0.0
241 Drr = 0.0
242 D11 = 0.0
243 rd = 0.0
244
245 ! Xa=Caa*Sra'+Cab*Srb'-Car
246 Xa = multipMM(Caa,SraT)
247 Xa = Xa + multipMM(Cab,SrbT)
248 Xa = Xa - Car
249
250 ! Xb=Cab'*Sra'+Cbb*Srb'-Cbr
251 Xb = multipMM(transpose(Cab),SraT)
252 Xb = Xb + multipMM(Cbb,SrbT)
253 Xb = Xb - Cbr
254
255 ! Xr=Car'*Sra'+Cbr'*Srb'-Crr
256 allocate(CarT(Nr,Na),stat=alloc_err)
257 allocate(CbrT(Nr,Nb),stat=alloc_err)
258 CarT = 0.0
259 CbrT = 0.0
260 CarT = transpose(Car)
261 CbrT = transpose(Cbr)

```

```

262
263  Xr = multipMM(CarT,SraT)
264  Xr = Xr + multipMM(CbrT,SrbT)
265  Xr = Xr - Crr
266
267  ! Drr=Sra*Xa+Srb*Xb-Xr
268  write(*, 1020)
269  Drr = multipMM(Sra,Xa)
270  Drr = Drr + multipMM(Srb,Xb)
271  Drr = Drr - Xr
272
273  ! define rd
274  rd = rc -rm
275
276  ! compute Drr^-1
277  allocate(Drrinv(Nr,Nr),stat=alloc_err)
278  Drrinv = 0.0
279  Drrinv = inv(Drr)
280  if(is_NAN_or_Infinity_M(Drr)) then
281    write(*, 1030)
282  end if
283  if(is_NAN_or_Infinity_M(Drrinv)) then
284    write(*, 1040)
285  end if
286
287  ! D11=Drr^-1
288  D11 = Drrinv
289
290  ! best estimated mean values for aBE, bBE, rBE
291  write(*, 1050)
292  allocate(aBE(Na),stat=alloc_err)
293  allocate(bBE(Nb),stat=alloc_err)
294  allocate(rBE(Nr),stat=alloc_err)
295  aBE = 0.0
296  bBE = 0.0
297  rBE = 0.0
298
299  ! aBE = alpha-[Xa*D11]*rd
300  allocate(XaD11(Na,Nr),stat=alloc_err)
301  XaD11 = 0.0
302  XaD11 = multipMM(Xa,D11)
303
304  aBE = multipMV(XaD11,rd)
305  aBE = alpha - aBE
306  call writeVectorToFile(ua_BE,aBE)
307
308  ! bBE = beta-[Xb*D11]*rd
309  write(*, 1060)
310  allocate(XbD11(Nb,Nr),stat=alloc_err)
311  XbD11 = 0.0
312  XbD11 = multipMM(Xb,D11)
313
314  bBE = multipMV(XbD11,rd)
315  bBE = beta - bBE
316  call writeVectorToFile(ub_BE,bBE)
317
318  ! rBE = rm-[Xr*D11]*rd
319  write(*, 1070)
320  allocate(XrD11(Nr,Nr),stat=alloc_err)
321  XrD11 = 0.0
322  XrD11 = multipMM(Xr,D11)

```

```

323
324 rBE = multipMV(XrD11,rd)
325 rBE = rm - rBE
326 call writeVectorToFile(ur_BE,rBE)
327
328 !calculate coviances for responses and parameters
329 allocate(CaaBE(Na,Na),stat=alloc_err)
330 allocate(CrrBE(Nr,Nr),stat=alloc_err)
331 allocate(CarBE(Na,Nr),stat=alloc_err)
332 allocate(CbbBE(Nb,Nb),stat=alloc_err)
333 allocate(CabBE(Na,Nb),stat=alloc_err)
334 allocate(CbrBE(Nb,Nr),stat=alloc_err)
335 CaaBE = 0.0; CrrBE = 0.0; CarBE = 0.0
336 CbbBE = 0.0; CbrBE = 0.0; CbrBE = 0.0
337
338 ! CaaBE = Caa - Xa*(D11*Xa')
339 write(*, 1080)
340 CaaBE = multipMM(Xa,transpose(XaD11))
341 CaaBE = Caa - CaaBE
342 call writeMatrixToFile(uC_aaBE,CaaBE)
343
344 ! CrrBE = Crr - Xr*(D11*Xr')
345 write(*, 1090)
346 CrrBE = multipMM(Xr,transpose(XrD11))
347 CrrBE = Crr - CrrBE
348 call writeMatrixToFile(uC_rrBE,CrrBE)
349
350 ! CarBE = Car - Xa*(D11*Xr')
351 write(*, 2000)
352 CarBE = multipMM(Xa,transpose(XrD11))
353 CarBE = Car - CarBE
354 call writeMatrixToFile(uC_arBE,CarBE)
355
356 ! CbbBE = Cbb - Xb*(D11*Xb')
357 write(*, 2010)
358 CbbBE = multipMM(Xb,transpose(XbD11))
359 CbbBE = Cbb - CbbBE
360 call writeMatrixToFile(uC_bbBE,CbbBE)
361
362 ! CabBE = Cab - Xa*(D11*Xb')
363 write(*, 2020)
364 CabBE = multipMM(Xa,transpose(XbD11))
365 CabBE = Cab - CabBE
366 call writeMatrixToFile(uC_abBE,CabBE)
367
368 ! CbrBE = Cbr - Xb*(D11*Xr')
369 write(*, 2030)
370 CbrBE = multipMM(Xb,transpose(XrD11))
371 CbrBE = Cbr - CbrBE
372 call writeMatrixToFile(uC_brBE,CbrBE)
373
374 !calculate the "consistency indicator" chi^2
375 allocate(D11rd(Nr),stat=alloc_err)
376 D11rd = 0.0
377 chi2 = 0.0
378
379 D11rd = multipMV(D11,rd)
380 chi2 = multipVV(rd,D11rd)
381 write(uchi2,3500,err=200) chi2, chi2/(Nr)
382
383 write(*,3600)

```

```

384 return
385 200 lerr = .true.;ierr = 3;call errmsg;stop
386 1000 format(/, &
387 1x,'----- Multi-pred Solving & Output -----')
388 1010 format(&
389 1x,'computing Crrcomp = Sra*Caa*Sra"+2*Sra*Cab*Srb"+Srb*Cbb*Srb"',a)
390 1020 format(&
391 1x,'computing Xa, Xb, Xr, D11, Drr, Drr^-1, rd',a)
392 1030 format(&
393 1x,'ERROR: Infinite or NAN in computing Drr, where ' &
394 'Drr=Sra*Xa+Srb*Xb-Xr',a)
395 1040 format(&
396 1x,'ERROR: Infinite or NAN in computing Drr^-1',a)
397 1050 format(&
398 1x,'computing aBE = alpha-[Xa*D11]*rd',a)
399 1060 format(&
400 1x,'computing bBE = beta-[Xb*D11]*rd',a)
401 1070 format(&
402 1x,'computing rBE = rm-[Xr*D11]*rd',a)
403 1080 format(&
404 1x,'computing CaaBE = Caa - Xa*(D11*Xa)',a)
405 1090 format(&
406 1x,'computing CrrBE = Crr - Xr*(D11*Xr)',a)
407 2000 format(&
408 1x,'computing CarBE = Car - Xa*(D11*Xr)',a)
409 2010 format(&
410 1x,'computing CbbBE = Cbb - Xb*(D11*Xb)',a)
411 2020 format(&
412 1x,'computing CabBE = Cab - Xa*(D11*Xb)',a)
413 2030 format(&
414 1x,'computing CbrBE = Cbr - Xb*(D11*Xr)',a)
415 3500 format(&
416 'chi^2 = ' ,F8.3/, &
417 'chi^2_d = (chi^2)/(number of responses) = ' ,F8.3)
418 3600 format(/, &
419 1x,'done.')
```

END SUBROUTINE solvercase2

```

422
423 !+++++
```

SUBROUTINE solvercase3()

```

426 !
427 ! solver for Casee 2: predictive modeling for Model A with Nq additional
428 ! responses, but no additional parameters
429 !
430 !Local-----
431 REAL(DP), ALLOCATABLE :: SraCaa(:,,:), SraT(:,,:), SqaCaa(:,,:), SqaT(:,,:)
432 REAL(DP), ALLOCATABLE :: Xa(:,,:), Ya(:,,:), Xr(:,,:), Yr(:,,:), Xq(:,,:), &
433 Yq(:,,:), Drr(:,,:), Drq(:,,:), Dqr(:,,:), Dqq(:,,:), &
434 D11(:,,:), D12(:,,:), D21(:,,:), D22(:,,:), rd(:), qd(:)
435 REAL(DP), ALLOCATABLE :: CarT(:,,:), CaqT(:,,:), CrqT(:,,:)
436 REAL(DP), ALLOCATABLE :: Drrinv(:,,:), DqrDrrinv(:,,:)
437 REAL(DP), ALLOCATABLE :: XaD11plusYaD21(:,,:), XaD12plusYaD22(:,,:)
438 REAL(DP), ALLOCATABLE :: XrD11plusYrD21(:,,:), XrD12plusYrD22(:,,:)
439 REAL(DP), ALLOCATABLE :: XqD11plusYqD21(:,,:), XqD12plusYqD22(:,,:)
440 REAL(DP), ALLOCATABLE :: D11rd(:), D12qd(:), D22qd(:)
441
442 write(*,1000)
443
444 !start with computing covariance matrix of Crr_comp, Cqq_comp and Crq_comp
```

```

445 !for the computed responses.
446
447 ! Crrcomp = Sra*Caa*Sra'
448 write(*,1010)
449 allocate(Crrcomp(Nr,Nr),stat=alloc_err)
450 allocate(SraCaa(Nr,Na),stat=alloc_err)
451 allocate(SraT(Na,Nr),stat=alloc_err)
452 Crrcomp = 0.0
453 SraCaa = 0.0
454 SraT = 0.0
455
456 SraCaa = multipMM(Sra,Caa)
457 SraT = transpose(Sra)
458 Crrcomp = multipMM(SraCaa,SraT)
459 call writeMatrixToFile(uCrr_comp,Crrcomp)
460
461 ! Cqqcomp = Sqa*Caa*Sqa'
462 write(*, 1020)
463 allocate(Cqqcomp(Nq,Nq),stat=alloc_err)
464 allocate(SqaCaa(Nq,Na),stat=alloc_err)
465 allocate(SqaT(Na,Nq),stat=alloc_err)
466 Cqqcomp = 0.0
467 SqaCaa = 0.0
468 SqaT = 0.0
469
470 SqaCaa = multipMM(Sqa,Caa)
471 SqaT = transpose(Sqa)
472 Cqqcomp = multipMM(SqaCaa,SqaT)
473 call writeMatrixToFile(uCqq_comp,Cqqcomp)
474
475 ! Crqcomp = Sra*Caa*Sqa'
476 write(*, 1030)
477 allocate(Crqcomp(Nr,Nq),stat=alloc_err)
478 Crqcomp = 0.0
479
480 Crqcomp = multipMM(SraCaa,SqaT)
481 call writeMatrixToFile(uCrq_comp,Crqcomp)
482
483 ! define intermediate quantities and initialize
484 write(*, 1040)
485 allocate(Xa(Na,Nr),stat=alloc_err)
486 allocate(Ya(Na,Nq),stat=alloc_err)
487 allocate(Xr(Nr,Nr),stat=alloc_err)
488 allocate(Yr(Nr,Nq),stat=alloc_err)
489 allocate(Xq(Nq,Nr),stat=alloc_err)
490 allocate(Yq(Nq,Nq),stat=alloc_err)
491 allocate(Drr(Nr,Nr),stat=alloc_err)
492 allocate(Drq(Nr,Nq),stat=alloc_err)
493 allocate(Dqr(Nq,Nr),stat=alloc_err)
494 allocate(Dqq(Nq,Nq),stat=alloc_err)
495 allocate(D11(Nr,Nr),stat=alloc_err)
496 allocate(D12(Nr,Nq),stat=alloc_err)
497 allocate(D21(Nq,Nr),stat=alloc_err)
498 allocate(D22(Nq,Nq),stat=alloc_err)
499 allocate(rd(Nr),stat=alloc_err)
500 allocate(qd(Nq),stat=alloc_err)
501
502 Xa = 0.0; Ya = 0.0; Xr = 0.0;
503 Yr = 0.0; Xq = 0.0; Yq = 0.0;
504 Drr = 0.0; Drq = 0.0; Dqr = 0.0;
505 Dqq = 0.0; D11 = 0.0; D12 = 0.0;

```

```

506 D21 = 0.0; D22 = 0.0; rd = 0.0;
507 qd = 0.0
508
509 ! Xa=Caa*Sra'-Car
510 Xa = multipMM(Caa,SraT)
511 Xa = Xa - Car
512
513 ! Ya=Caa*Sqa'-Caq
514 Ya = multipMM(Caa,SqaT)
515 Ya = Ya - Caq
516
517 ! Xr=Car'*Sra'-Crr
518 allocate(CarT(Nr,Na),stat=alloc_err)
519 CarT = 0.0
520 CarT = transpose(Car)
521
522 Xr = multipMM(CarT,SraT)
523 Xr = Xr - Crr
524
525 ! Yr=Car'*Sqa'-Crq
526 Yr = multipMM(CarT,SqaT)
527 Yr = Yr - Crq
528
529 ! Xq=Caq'*Sra'-Crq'
530 allocate(CaqT(Nq,Na),stat=alloc_err)
531 allocate(CrqT(Nq,Nr),stat=alloc_err)
532 CaqT = 0.0
533 CrqT = 0.0
534 CaqT = transpose(Caq)
535 CrqT = transpose(Crq)
536
537 Xq = multipMM(CaqT,SraT)
538 Xq = Xq - CrqT
539
540 ! Yq=Caq'*Sqa'-Cqq
541 Yq = multipMM(CaqT,SqaT)
542 Yq = Yq - Cqq
543
544 ! Drr=Sra*Xa-Xr
545 Drr = multipMM(Sra,Xa)
546 Drr = Drr - Xr
547
548 ! Drq=Sra*Ya-Yr
549 Drq = multipMM(Sra,Ya)
550 Drq = Drq - Yr
551
552 ! Drq'
553 Dqr = transpose(Drq)
554
555 ! Dqq=Sqa*Ya-Yq
556 Dqq = multipMM(Sqa,Ya)
557 Dqq = Dqq - Yq
558
559 ! define rd, qd
560 rd = rc -rm
561 qd = qc -qm
562
563 ! compute Drr^-1, Dqq^-1, D22^-1
564 allocate(Drrinv(Nr,Nr),stat=alloc_err)
565 Drrinv = 0.0
566 Drrinv = inv(Drr)

```



```

567  if(is_NAN_or_Infinity_M(Drr)) then
568      write(*, 1050)
569  end if
570  if(is_NAN_or_Infinity_M(Drrinv)) then
571      write(*, 1060)
572  end if
573
574  ! D22=[Dqq-Drq'*Drr^-1*Drq]^-1
575  allocate(DqrDrrinv(Nq,Nr),stat=alloc_err)
576  DqrDrrinv = 0.0
577  DqrDrrinv = multipMM(Dqr,Drrinv)
578  D22 = Dqq - multipMM(DqrDrrinv,Drq)
579  D22 = inv(D22)
580
581  ! D12=-Drr^-1*Drq*D22
582  D12 = multipMM(Drrinv,Drq)
583  D12 = -1.0 * multipMM(D12,D22)
584
585  ! D12'
586  D21 = transpose(D12)
587
588  ! D11=Drr^-1+D12*Drq'*Drr^-1
589  D11 = multipMM(D12,Dqr)
590  D11 = Drrinv - multipMM(D11,Drrinv)
591
592  ! best estimated mean values for aBE, rBE, qBE
593  write(*, 1070)
594  allocate(aBE(Na),stat=alloc_err)
595  allocate(rBE(Nr),stat=alloc_err)
596  allocate(qBE(Nq),stat=alloc_err)
597  aBE = 0.0
598  rBE = 0.0
599  qBE = 0.0
600
601  ! aBE = alpha-[Xa*D11+Ya*D21]*rd-[Xa*D12+Ya*D22]*qd
602  allocate(XaD11plusYaD21(Na,Nr),stat=alloc_err)
603  allocate(XaD12plusYaD22(Na,Nq),stat=alloc_err)
604  XaD11plusYaD21 = 0.0
605  XaD12plusYaD22 = 0.0
606  XaD11plusYaD21 = multipMM(Xa,D11) + multipMM(Ya,D21)
607  XaD12plusYaD22 = multipMM(Xa,D12) + multipMM(Ya,D22)
608
609  aBE = multipMV(XaD11plusYaD21,rd)
610  aBE = alpha - aBE
611  aBE = aBE - multipMV(XaD12plusYaD22,qd)
612  call writeVectorToFile(ua_BE,aBE)
613
614  ! rBE = rm-[Xr*D11+Yr*D21]*rd-[Xr*D12+Yr*D22]*qd
615  write(*, 1080)
616  allocate(XrD11plusYrD21(Nr,Nr),stat=alloc_err)
617  allocate(XrD12plusYrD22(Nr,Nq),stat=alloc_err)
618  XrD11plusYrD21 = 0.0
619  XrD12plusYrD22 = 0.0
620  XrD11plusYrD21 = multipMM(Xr,D11) + multipMM(Yr,D21)
621  XrD12plusYrD22 = multipMM(Xr,D12) + multipMM(Yr,D22)
622
623  rBE = multipMV(XrD11plusYrD21,rd)
624  rBE = rm - rBE
625  rBE = rBE - multipMV(XrD12plusYrD22,qd)
626  call writeVectorToFile(ur_BE,rBE)
627

```

```

628 ! qBE = qm - [Xq*D11+Yq*D21]*rd - [Xq*D12+Yq*D22]*qd
629 write(*, 1090)
630 allocate(XqD11plusYqD21(Nq,Nr),stat=alloc_err)
631 allocate(XqD12plusYqD22(Nq,Nq),stat=alloc_err)
632 XqD11plusYqD21 = 0.0
633 XqD12plusYqD22 = 0.0
634 XqD11plusYqD21 = multipMM(Xq,D11) + multipMM(Yq,D21)
635 XqD12plusYqD22 = multipMM(Xq,D12) + multipMM(Yq,D22)
636
637 qBE = multipMV(XqD11plusYqD21,rd)
638 qBE = qm - qBE
639 qBE = qBE - multipMV(XqD12plusYqD22,qd)
640 call writeVectorToFile(uq_BE,qBE)
641
642 ! calculate coviances for responses and parameters
643 allocate(CaaBE(Na,Na),stat=alloc_err)
644 allocate(CrrBE(Nr,Nr),stat=alloc_err)
645 allocate(CarBE(Na,Nr),stat=alloc_err)
646 allocate(CqqBE(Nq,Nq),stat=alloc_err)
647 allocate(CaqBE(Na,Nq),stat=alloc_err)
648 allocate(CrqBE(Nr,Nq),stat=alloc_err)
649 CaaBE = 0.0; CrrBE = 0.0; CarBE = 0.0
650 CqqBE = 0.0; CaqBE = 0.0; CrqBE = 0.0
651
652 ! CaaBE = Caa - [Xa*(D11*Xa'+D12*Ya')+Ya*(D21*Xa'+D22*Ya')]
653 write(*, 2000)
654 CaaBE = multipMM(Xa,transpose(XaD11plusYaD21))
655 CaaBE = Caa - CaaBE
656 CaaBE = CaaBE - multipMM(Ya,transpose(XaD12plusYaD22))
657 call writeMatrixToFile(uC_aaBE,CaaBE)
658
659 ! CrrBE = Crr - [Xr*(D11*Xr'+D12*Yr')+Yr*(D21*Xr'+D22*Yr')]
660 write(*, 2010)
661 CrrBE = multipMM(Xr,transpose(XrD11plusYrD21))
662 CrrBE = Crr - CrrBE
663 CrrBE = CrrBE - multipMM(Yr,transpose(XrD12plusYrD22))
664 call writeMatrixToFile(uC_rrBE,CrrBE)
665
666 ! CarBE = Car - [Xa*(D11*Xr'+D12*Yr')+Ya*(D21*Xr'+D22*Yr')]
667 write(*, 2020)
668 CarBE = multipMM(Xa,transpose(XrD11plusYrD21))
669 CarBE = Car - CarBE
670 CarBE = CarBE - multipMM(Ya,transpose(XrD12plusYrD22))
671 call writeMatrixToFile(uC_arBE,CarBE)
672
673 ! CqqBE = Cqq - [Xq*(D11*Xq'+D12*Yq')+Yq*(D21*Xq'+D22*Yq')]
674 write(*, 2030)
675 CqqBE = multipMM(Xq,transpose(XqD11plusYqD21))
676 CqqBE = Cqq - CqqBE
677 CqqBE = CqqBE - multipMM(Yq,transpose(XqD12plusYqD22))
678 call writeMatrixToFile(uC_qqBE,CqqBE)
679
680 ! CaqBE = Caq - [Xa*(D11*Xq'+D12*Yq')+Ya*(D21*Xq'+D22*Yq')]
681 write(*, 2040)
682 CaqBE = multipMM(Xa,transpose(XqD11plusYqD21))
683 CaqBE = Caq - CaqBE
684 CaqBE = CaqBE - multipMM(Ya,transpose(XqD12plusYqD22))
685 call writeMatrixToFile(uC_aqBE,CaqBE)
686
687 ! CrqBE = Crq - [Xr*(D11*Xq'+D12*Yq')+Yr*(D21*Xq'+D22*Yq')]
688 write(*, 2050)

```

```

689 CrqBE = multipMM(Xr,transpose(XqD11plusYqD21))
690 CrqBE = Crq - CrqBE
691 CrqBE = CrqBE - multipMM(Yr,transpose(XqD12plusYqD22))
692 call writeMatrixToFile(uC_rqBE,CrqBE)
693
694 lcalculate the "consistency indicator" chi^2
695 allocate(D11rd(Nr),stat=alloc_err)
696 allocate(D12qd(Nr),stat=alloc_err)
697 allocate(D22qd(Nq),stat=alloc_err)
698 D11rd = 0.0
699 D12qd = 0.0
700 D22qd = 0.0
701 chi2 = 0.0
702
703 D11rd = multipMV(D11,rd)
704 D12qd = multipMV(D12,qd)
705 D22qd = multipMV(D22,qd)
706 chi2 = multipVV(rd,D11rd)
707 chi2 = chi2 + 2.0 * multipVV(rd,D12qd)
708 chi2 = chi2 + multipVV(qd,D22qd)
709 write(uchi2,3500,err=200) chi2, chi2/(Nr+Nq)
710
711 write(*,3600)
712 return
713 200 lerr = .true.;ierr = 3;call errmsg;stop
714 1000 format(/, &
715 1x,'----- Multi-pred Solving & Output -----')
716 1010 format(&
717 1x,'computing Crrcomp = Sra*Caa*Sra',a)
718 1020 format(&
719 1x,'computing Cqqcomp = Sqa*Caa*Sqa',a)
720 1030 format(&
721 1x,'computing Cqqcomp = Sqa*Caa*Sqa',a)
722 1040 format(&
723 1x,'computing Xa, Ya, Xr, Yr, Xq, Yq, D11, D12, D22, Drr,Drq, Dqr,/, &
724 1x,' Dqq, rd, qd',a)
725 1050 format(&
726 1x,'ERROR: Infinite or NAN in computing Drr, where Drr=Sra*Xa-Xr',a)
727 1060 format(&
728 1x,'ERROR: Infinite or NAN in computing Drr^-1',a)
729 1070 format(&
730 1x,'computing aBE = alpha-[Xa*D11+Ya*D21]*rd-[Xa*D12+Ya*D22]*qd',a)
731 1080 format(&
732 1x,'computing rBE = rm-[Xr*D11+Yr*D21]*rd-[Xr*D12+Yr*D22]*qd',a)
733 1090 format(&
734 1x,'computing qBE = qm-[Xq*D11+Yq*D21]*rd-[Xq*D12+Yq*D22]*qd',a)
735 2000 format(&
736 1x,'computing CaaBE = Caa-[Xa*(D11*Xa"+D12*Ya")+Ya*(D21*Xa"+D22*Ya")]',a)
737 2010 format(&
738 1x,'computing CrrBE = Crr-[Xr*(D11*Xr"+D12*Yr")+Yr*(D21*Xr"+D22*Yr")]',a)
739 2020 format(&
740 1x,'computing CarBE = Car-[Xa*(D11*Xr"+D12*Yr")+Ya*(D21*Xr"+D22*Yr")]',a)
741 2030 format(&
742 1x,'computing CqqBE = Cqq-[Xq*(D11*Xq"+D12*Yq")+Yq*(D21*Xq"+D22*Yq")]',a)
743 2040 format(&
744 1x,'computing CaqBE = Caq-[Xa*(D11*Xq"+D12*Yq")+Ya*(D21*Xq"+D22*Yq")]',a)
745 2050 format(&
746 1x,'computing CrqBE = Crq-[Xr*(D11*Xq"+D12*Yq")+Yr*(D21*Xq"+D22*Yq")]',a)
747 3500 format(&
748 1x,'chi^2 =',F8.3/, &
749 1x,'chi^2_d = (chi^2)/(number of responses) =',F8.3)

```

```

750 3600 format(/, &
751 1x, 'done. ')
752
753 END SUBROUTINE solvercase3
754
755 !+++++
756
757 SUBROUTINE solvercase4()
758 !
759 ! solver for Casee 4: coupled Models A & B
760 !
761 !Local-----
762 REAL(DP), ALLOCATABLE :: SraCaa(:, :), SraT(:, :), SraCab(:, :), SrbT(:, :), &
763 SrbCbb(:, :)
764 REAL(DP), ALLOCATABLE :: SqaCaa(:, :), SqaT(:, :), SqaCab(:, :), SqbT(:, :), &
765 SqbCbb(:, :)
766 REAL(DP), ALLOCATABLE :: CabT(:, :), SrbCabT(:, :)
767 REAL(DP), ALLOCATABLE :: Xa(:, :), Ya(:, :), Xb(:, :), Yb(:, :), Xr(:, :), &
768 Yr(:, :), Xq(:, :), Yq(:, :), Drr(:, :), Drq(:, :), &
769 Dqr(:, :), Dqq(:, :), D11(:, :), D12(:, :), D21(:, :), &
770 D22(:, :), rd(:, :), qd(:, :)
771 REAL(DP), ALLOCATABLE :: CarT(:, :), CbrT(:, :)
772 REAL(DP), ALLOCATABLE :: CaqT(:, :), CbqT(:, :), CrqT(:, :)
773 REAL(DP), ALLOCATABLE :: Drrinv(:, :), DqrDrrinv(:, :), temp2(:, :), temp3(:, :)
774 REAL(DP), ALLOCATABLE :: XaD11plusYaD21(:, :), XaD12plusYaD22(:, :)
775 REAL(DP), ALLOCATABLE :: XbD11plusYbD21(:, :), XbD12plusYbD22(:, :)
776 REAL(DP), ALLOCATABLE :: XrD11plusYrD21(:, :), XrD12plusYrD22(:, :)
777 REAL(DP), ALLOCATABLE :: XqD11plusYqD21(:, :), XqD12plusYqD22(:, :)
778 REAL(DP), ALLOCATABLE :: D11rd(:, :), D12qd(:, :), D22qd(:, :)
779
780 write(*,1000)
781
782 !start with computing covariance matrix of Crr_comp, Cqq_comp and Crq_comp
783 !for the computed responses.
784
785 ! Crrcomp = Sra*Caa*Sra'+2Sra*Cab*Srb'+Srb*Cbb*Srb'
786 write(*, 1010)
787 allocate(Crrcomp(Nr,Nr),stat=alloc_err)
788 allocate(SraCaa(Nr,Na),stat=alloc_err)
789 allocate(SraT(Na,Nr),stat=alloc_err)
790 allocate(SraCab(Nr,Nb),stat=alloc_err)
791 allocate(SrbT(Nb,Nr),stat=alloc_err)
792 allocate(SrbCbb(Nr,Nb),stat=alloc_err)
793 Crrcomp = 0.0; SraCaa = 0.0
794 SraT = 0.0; SraCab = 0.0
795 SrbT = 0.0; SrbCbb = 0.0
796
797 SraCaa = multipMM(Sra,Caa)
798 SraT = transpose(Sra)
799 Crrcomp = multipMM(SraCaa,SraT)
800 SraCab = multipMM(Sra,Cab)
801 SrbT = transpose(Srb)
802 Crrcomp = Crrcomp + 2 * multipMM(SraCab,SrbT)
803 SrbCbb = multipMM(Srb,Cbb)
804 Crrcomp = Crrcomp + multipMM(SrbCbb,SrbT)
805 call writeMatrixToFile(uCrr_comp,Crrcomp)
806
807 ! Cqqcomp = Sqa*Caa*Sqa'+2Sqa*Cab*Sqb'+Sqb*Cbb*Sqb'
808 write(*, 1020)
809 allocate(Cqqcomp(Nq,Nq),stat=alloc_err)
810 allocate(SqaCaa(Nq,Na),stat=alloc_err)

```

```

811 allocate(SqaT(Na,Nq),stat=alloc_err)
812 allocate(SqaCab(Nq,Nb),stat=alloc_err)
813 allocate(SqbT(Nb,Nq),stat=alloc_err)
814 allocate(SqbCbb(Nq,Nb),stat=alloc_err)
815 Cqqcomp = 0.0; SqaCaa = 0.0
816 SqaT = 0.0; SqaCab = 0.0
817 SqbT = 0.0; SqbCbb = 0.0
818
819 SqaCaa = multipMM(Sqa,Caa)
820 SqaT = transpose(Sqa)
821 Cqqcomp = multipMM(SqaCaa,SqaT)
822 SqaCab = multipMM(Sqa,Cab)
823 SqbT = transpose(Sqb)
824 Cqqcomp = Cqqcomp + 2 * multipMM(SqaCab,SqbT)
825 SqbCbb = multipMM(Sqb,Cbb)
826 Cqqcomp = Cqqcomp + multipMM(SqbCbb,SqbT)
827 call writeMatrixToFile(uCqq_comp,Cqqcomp)
828
829 ! Crqcomp = Sra*Caa*Sqa'+Sra*Cab*Sqb'+Srb*Cab'*Sqa'+Sqb*Cbb*Sqb'
830 write(*, 1030)
831 allocate(Crqcomp(Nr,Nq),stat=alloc_err)
832 allocate(CabT(Nb,Na),stat=alloc_err)
833 allocate(SrbCabT(Nr,Na),stat=alloc_err)
834 Crqcomp = 0.0
835 CabT = 0.0
836 SrbCabT = 0.0
837
838 Crqcomp = multipMM(SraCaa,SqaT)
839 Crqcomp = Crqcomp + multipMM(SraCab,SqbT)
840 CabT = transpose(Cab)
841 SrbCabT = multipMM(Srb,CabT)
842 Crqcomp = Crqcomp + multipMM(SrbCabT,SqaT)
843 Crqcomp = Crqcomp + multipMM(SrbCbb,SqbT)
844 call writeMatrixToFile(uCrq_comp,Crqcomp)
845
846 ! define intermediate quantities and initialize
847 write(*, 1040)
848 allocate(Xa(Na,Nr),stat=alloc_err)
849 allocate(Ya(Na,Nq),stat=alloc_err)
850 allocate(Xb(Nb,Nr),stat=alloc_err)
851 allocate(Yb(Nb,Nq),stat=alloc_err)
852 allocate(Xr(Nr,Nr),stat=alloc_err)
853 allocate(Yr(Nr,Nq),stat=alloc_err)
854 allocate(Xq(Nq,Nr),stat=alloc_err)
855 allocate(Yq(Nq,Nq),stat=alloc_err)
856 allocate(Drr(Nr,Nr),stat=alloc_err)
857 allocate(Drq(Nr,Nq),stat=alloc_err)
858 allocate(Dqr(Nq,Nr),stat=alloc_err)
859 allocate(Dqq(Nq,Nq),stat=alloc_err)
860 allocate(D11(Nr,Nr),stat=alloc_err)
861 allocate(D12(Nr,Nq),stat=alloc_err)
862 allocate(D21(Nq,Nr),stat=alloc_err)
863 allocate(D22(Nq,Nq),stat=alloc_err)
864 allocate(rd(Nr),stat=alloc_err)
865 allocate(qd(Nq),stat=alloc_err)
866
867 Xa = 0.0; Ya = 0.0; Xb = 0.0
868 Yb = 0.0; Xr = 0.0; Yr = 0.0
869 Xq = 0.0; Yq = 0.0; Drr = 0.0
870 Drq = 0.0; Dqr = 0.0; Dqq = 0.0
871 D11 = 0.0; D12 = 0.0; qd = 0.0

```

```

872 D21 = 0.0; D22 = 0.0; rd = 0.0
873
874 ! Xa=Caa*Sra'+Cab*Srb'-Car
875 Xa = multipMM(Caa,SraT)
876 Xa = Xa + multipMM(Cab,SrbT)
877 Xa = Xa - Car
878
879 ! Ya=Caa*Sqa'+Cab*Sqb'-Caq
880 Ya = multipMM(Caa,SqaT)
881 Ya = Ya + multipMM(Cab,SqbT)
882 Ya = Ya - Caq
883
884 ! Xb=Cab'*Sra'+Cbb*Srb'-Cbr
885 Xb = multipMM(CabT,SraT)
886 Xb = Xb + multipMM(Cbb,SrbT)
887 Xb = Xb - Cbr
888
889 ! Yb=Cba*Sqa'+Cbb*Sqb'-Cbq
890 Yb = multipMM(CabT,SqaT)
891 Yb = Yb + multipMM(Cbb,SqbT)
892 Yb = Yb - Cbq
893
894 ! Xr=Car'*Sra'+Cbr'*Srb'-Crr
895 allocate(CarT(Nr,Na),stat=alloc_err)
896 allocate(CbrT(Nr,Nb),stat=alloc_err)
897 CarT = 0.0
898 CbrT = 0.0
899 CarT = transpose(Car)
900 CbrT = transpose(Cbr)
901
902 Xr = multipMM(CarT,SraT)
903 Xr = Xr + multipMM(CbrT,SrbT)
904 Xr = Xr - Crr
905
906 ! Yr=Car'*Sqa'+Cbr'*Sqb'-Crq
907 Yr = multipMM(CarT,SqaT)
908 Yr = Yr + multipMM(CbrT,SqbT)
909 Yr = Yr - Crq
910
911 ! Xq=Caq'*Sra'+Cbq'*Srb'-Crq'
912 allocate(CaqT(Nq,Na),stat=alloc_err)
913 allocate(CbqT(Nq,Nb),stat=alloc_err)
914 allocate(CrqT(Nq,Nr),stat=alloc_err)
915 CaqT = 0.0
916 CbqT = 0.0
917 CrqT = 0.0
918 CaqT = transpose(Caq)
919 CbqT = transpose(Cbq)
920 CrqT = transpose(Crq)
921
922 Xq = multipMM(CaqT,SraT)
923 Xq = Xq + multipMM(CbqT,SrbT)
924 Xq = Xq - CrqT
925
926 ! Yq=Caq'*Sqa'+Cbq'*Sqb'-Cqq
927 Yq = multipMM(CaqT,SqaT)
928 Yq = Yq + multipMM(CbqT,SqbT)
929 Yq = Yq - Cqq
930
931 ! Drr=Sra*Xa+Srb*Xb-Xr
932 Drr = multipMM(Sra,Xa)

```

```

933 Drr = Drr + multipMM(Srb,Xb)
934 Drr = Drr - Xr
935
936 ! Drq=Sra*Ya+Srb*Yb-Yr
937 Drq = multipMM(Sra,Ya)
938 Drq = Drq + multipMM(Srb,Yb)
939 Drq = Drq - Yr
940
941 ! Drq'
942 Dqr = transpose(Drq)
943
944 ! Dqq=Sqa*Ya+Sqb*Yb-Yq
945 Dqq = multipMM(Sqa,Ya)
946 Dqq = Dqq + multipMM(Sqb,Yb)
947 Dqq = Dqq - Yq
948
949 ! define rd, qd
950 rd = rc -rm
951 qd = qc -qm
952
953 ! compute Drr^-1, Dqq^-1, D22^-1
954 allocate(Drrinv(Nr,Nr),stat=alloc_err)
955 Drrinv = 0.0
956 Drrinv = inv(Drr)
957 if(is_NAN_or_Infinity_M(Drr)) then
958     write(*, 1050)
959 end if
960 if(is_NAN_or_Infinity_M(Drrinv)) then
961     write(*, 1060)
962 end if
963
964 ! D22=[Dqq-Drq'*Drr^-1*Drq]^-1
965 allocate(DqrDrrinv(Nq,Nr),stat=alloc_err)
966 DqrDrrinv = 0.0
967 DqrDrrinv = multipMM(Dqr,Drrinv)
968 D22 = Dqq - multipMM(DqrDrrinv,Drq)
969 D22 = inv(D22)
970
971 ! D12=-Drr^-1*Drq*D22
972 D12 = multipMM(Drrinv,Drq)
973 D12 = -1.0 * multipMM(D12,D22)
974
975 ! D12'
976 D21 = transpose(D12)
977
978 ! D11=Drr^-1+D12*Drq'*Drr^-1
979 D11 = multipMM(D12,Dqr)
980 D11 = Drrinv - multipMM(D11,Drrinv)
981
982 ! best estimated mean values for aBE, bBE, rBE, qBE
983 write(*, 1070)
984 allocate(aBE(Na),stat=alloc_err)
985 allocate(bBE(Nb),stat=alloc_err)
986 allocate(rBE(Nr),stat=alloc_err)
987 allocate(qBE(Nq),stat=alloc_err)
988 aBE = 0.0
989 bBE = 0.0
990 rBE = 0.0
991 qBE = 0.0
992
993 ! aBE = alpha-[Xa*D11+Ya*D21]*rd-[Xa*D12+Ya*D22]*qd

```

```

994 allocate(XaD11plusYaD21(Na,Nr),stat=alloc_err)
995 allocate(XaD12plusYaD22(Na,Nq),stat=alloc_err)
996 XaD11plusYaD21 = 0.0
997 XaD12plusYaD22 = 0.0
998 XaD11plusYaD21 = multipMM(Xa,D11) + multipMM(Ya,D21)
999 XaD12plusYaD22 = multipMM(Xa,D12) + multipMM(Ya,D22)
1000
1001 aBE = multipMV(XaD11plusYaD21,rd)
1002 aBE = alpha - aBE
1003 aBE = aBE - multipMV(XaD12plusYaD22,qd)
1004 call writeVectorToFile(ua_BE,aBE)
1005
1006 ! bBE = beta - [Xb*D11+Yb*D21]*rd - [Xb*D12+Yb*D22]*qd
1007 write(*, 1080)
1008 allocate(XbD11plusYbD21(Nb,Nr),stat=alloc_err)
1009 allocate(XbD12plusYbD22(Nb,Nq),stat=alloc_err)
1010 XbD11plusYbD21 = 0.0
1011 XbD12plusYbD22 = 0.0
1012 XbD11plusYbD21 = multipMM(Xb,D11) + multipMM(Yb,D21)
1013 XbD12plusYbD22 = multipMM(Xb,D12) + multipMM(Yb,D22)
1014
1015 bBE = multipMV(XbD11plusYbD21,rd)
1016 bBE = beta - bBE
1017 bBE = bBE - multipMV(XbD12plusYbD22,qd)
1018 call writeVectorToFile(ub_BE,bBE)
1019
1020 ! rBE = rm - [Xr*D11+Yr*D21]*rd - [Xr*D12+Yr*D22]*qd
1021 write(*, 1090)
1022 allocate(XrD11plusYrD21(Nr,Nr),stat=alloc_err)
1023 allocate(XrD12plusYrD22(Nr,Nq),stat=alloc_err)
1024 XrD11plusYrD21 = 0.0
1025 XrD12plusYrD22 = 0.0
1026 XrD11plusYrD21 = multipMM(Xr,D11) + multipMM(Yr,D21)
1027 XrD12plusYrD22 = multipMM(Xr,D12) + multipMM(Yr,D22)
1028
1029 rBE = multipMV(XrD11plusYrD21,rd)
1030 rBE = rm - rBE
1031 rBE = rBE - multipMV(XrD12plusYrD22,qd)
1032 call writeVectorToFile(ur_BE,rBE)
1033
1034 ! qBE = qm - [Xq*D11+Yq*D21]*rd - [Xq*D12+Yq*D22]*qd
1035 write(*, 2000)
1036 allocate(XqD11plusYqD21(Nq,Nr),stat=alloc_err)
1037 allocate(XqD12plusYqD22(Nq,Nq),stat=alloc_err)
1038 XqD11plusYqD21 = 0.0
1039 XqD12plusYqD22 = 0.0
1040 XqD11plusYqD21 = multipMM(Xq,D11) + multipMM(Yq,D21)
1041 XqD12plusYqD22 = multipMM(Xq,D12) + multipMM(Yq,D22)
1042
1043 qBE = multipMV(XqD11plusYqD21,rd)
1044 qBE = qm - qBE
1045 qBE = qBE - multipMV(XqD12plusYqD22,qd)
1046 call writeVectorToFile(uq_BE,qBE)
1047
1048 !calculate coviances for responses and parameters
1049 allocate(CaaBE(Na,Na),stat=alloc_err)
1050 allocate(CrrBE(Nr,Nr),stat=alloc_err)
1051 allocate(CarBE(Na,Nr),stat=alloc_err)
1052 allocate(CbbBE(Nb,Nb),stat=alloc_err)
1053 allocate(CqqBE(Nq,Nq),stat=alloc_err)
1054 allocate(CbqBE(Nb,Nq),stat=alloc_err)

```



```

1055 allocate(CabBE(Na,Nb),stat=alloc_err)
1056 allocate(CaqBE(Na,Nq),stat=alloc_err)
1057 allocate(CbrBE(Nb,Nr),stat=alloc_err)
1058 allocate(CrqBE(Nr,Nq),stat=alloc_err)
1059 CaaBE = 0.0; CrrBE = 0.0; CarBE = 0.0
1060 CbbBE = 0.0; CqqBE = 0.0; CbqBE = 0.0
1061 CaqBE = 0.0; CbrBE = 0.0; CbrBE = 0.0
1062 CrqBE = 0.0
1063
1064 ! CaaBE = Caa - [Xa*(D11*Xa'+D12*Ya')+Ya*(D21*Xa'+D22*Ya')]
1065 write(*, 2010)
1066 CaaBE = multipMM(Xa,transpose(XaD11plusYaD21))
1067 CaaBE = Caa - CaaBE
1068 CaaBE = CaaBE - multipMM(Ya,transpose(XaD12plusYaD22))
1069 call writeMatrixToFile(uC_aaBE,CaaBE)
1070
1071 ! CrrBE = Crr - [Xr*(D11*Xr'+D12*Yr')+Yr*(D21*Xr'+D22*Yr')]
1072 write(*, 2020)
1073 CrrBE = multipMM(Xr,transpose(XrD11plusYrD21))
1074 CrrBE = Crr - CrrBE
1075 CrrBE = CrrBE - multipMM(Yr,transpose(XrD12plusYrD22))
1076 call writeMatrixToFile(uC_rrBE,CrrBE)
1077
1078 ! CarBE = Car - [Xa*(D11*Xr'+D12*Yr')+Ya*(D21*Xr'+D22*Yr')]
1079 write(*, 2030)
1080 CarBE = multipMM(Xa,transpose(XrD11plusYrD21))
1081 CarBE = Car - CarBE
1082 CarBE = CarBE - multipMM(Ya,transpose(XrD12plusYrD22))
1083 call writeMatrixToFile(uC_arBE,CarBE)
1084
1085 ! CbbBE = Cbb - [Xb*(D11*Xb'+D12*Yb')+Yb*(D21*Xb'+D22*Yb')]
1086 write(*, 2040)
1087 CbbBE = multipMM(Xb,transpose(XbD11plusYbD21))
1088 CbbBE = Cbb - CbbBE
1089 CbbBE = CbbBE - multipMM(Yb,transpose(XbD12plusYbD22))
1090 call writeMatrixToFile(uC_bbBE,CbbBE)
1091
1092 ! CqqBE = Cqq - [Xq*(D11*Xq'+D12*Yq')+Yq*(D21*Xq'+D22*Yq')]
1093 write(*, 2050)
1094 CqqBE = multipMM(Xq,transpose(XqD11plusYqD21))
1095 CqqBE = Cqq - CqqBE
1096 CqqBE = CqqBE - multipMM(Yq,transpose(XqD12plusYqD22))
1097 call writeMatrixToFile(uC_qqBE,CqqBE)
1098
1099 ! CbqBE = Cbq - [Xb*(D11*Xq'+D12*Yq')+Yb*(D21*Xq'+D22*Yq')]
1100 write(*, 2060)
1101 CbqBE = multipMM(Xb,transpose(XqD11plusYqD21))
1102 CbqBE = Cbq - CbqBE
1103 CbqBE = CbqBE - multipMM(Yb,transpose(XqD12plusYqD22))
1104 call writeMatrixToFile(uC_bqBE,CbqBE)
1105
1106 ! CabBE = Cab - [Xa*(D11*Xb'+D12*Yb')+Yb*(D21*Xb'+D22*Yb')]
1107 write(*, 2070)
1108 CabBE = multipMM(Xa,transpose(XbD11plusYbD21))
1109 CabBE = Cab - CabBE
1110 CabBE = CabBE - multipMM(Ya,transpose(XbD12plusYbD22))
1111 call writeMatrixToFile(uC_abBE,CabBE)
1112
1113 ! CaqBE = Caq - [Xa*(D11*Xq'+D12*Yq')+Ya*(D21*Xq'+D22*Yq')]
1114 write(*, 2080)
1115 CaqBE = multipMM(Xa,transpose(XqD11plusYqD21))

```

```

1116 CaqBE = Caq - CaqBE
1117 CaqBE = CaqBE - multipMM(Ya,transpose(XqD12plusYqD22))
1118 call writeMatrixToFile(uC_aqBE,CaqBE)
1119
1120 ! CbrBE = Cbr - [Xb*(D11*Xr'+D12*Yr')+Yb*(D21*Xr'+D22*Yr')]
1121 write(*, 2090)
1122 CbrBE = multipMM(Xb,transpose(XrD11plusYrD21))
1123 CbrBE = Cbr - CbrBE
1124 CbrBE = CbrBE - multipMM(Yb,transpose(XrD12plusYrD22))
1125 call writeMatrixToFile(uC_brBE,CbrBE)
1126
1127 ! CrqBE = Crq - [Xr*(D11*Xq'+D12*Yq')+Yr*(D21*Xq'+D22*Yq')]
1128 write(*, 3000)
1129 CrqBE = multipMM(Xr,transpose(XqD11plusYqD21))
1130 CrqBE = Crq - CrqBE
1131 CrqBE = CrqBE - multipMM(Yr,transpose(XqD12plusYqD22))
1132 call writeMatrixToFile(uC_rqBE,CrqBE)
1133
1134
1135 !calculate the "consistency indicator" chi^2
1136 allocate(D11rd(Nr),stat=alloc_err)
1137 allocate(D12qd(Nr),stat=alloc_err)
1138 allocate(D22qd(Nq),stat=alloc_err)
1139 D11rd = 0.0
1140 D12qd = 0.0
1141 D22qd = 0.0
1142 chi2 = 0.0
1143
1144 D11rd = multipMV(D11,rd)
1145 D12qd = multipMV(D12,qd)
1146 D22qd = multipMV(D22,qd)
1147 chi2 = multipVV(rd,D11rd)
1148 chi2 = chi2 + 2.0 * multipVV(rd,D12qd)
1149 chi2 = chi2 + multipVV(qd,D22qd)
1150 write(uchi2,3500,err=200) chi2, chi2/(Nr+Nq)
1151
1152 write(*,3600)
1153 return
1154 200 lerr = .true.;ierr = 3;call errmsg;stop
1155 1000 format(/, &
1156 1x,'----- Multi-pred Solving & Output -----')
1157 1010 format(&
1158 1x,'computing Crrcomp = Sra*Caa*Sra"+2*Sra*Cab*Srb"+Srb*Cbb*Srb"',a)
1159 1020 format(&
1160 1x,'computing Cqqcomp = Sqa*Caa*Sqa"+2*Sqa*Cab*Sqb"+Sqb*Cbb*Sqb"',a)
1161 1030 format(&
1162 1x,'computing Crqcomp =Sra*Caa*Sqa"+Sra*Cab*Sqb"+Srb*Cab"*Sqa"+Sqb*Cbb*Sqb"',a)
1163 1040 format(&
1164 1x,'computing Xa, Ya, Xb, Yb, Xr, Yr, Xq, Yq, D11, D12, D22, Drr, ' /, &
1165 ' Drq, Dqr, Dqq, rd, qd',a)
1166 1050 format(&
1167 1x,'ERROR: Infinite or NAN in computing Drr, where ' &
1168 ' Drr=Sra*Xa+Srb*Xb-Xr',a)
1169 1060 format(&
1170 1x,'ERROR: Infinite or NAN in computing Drr^-1',a)
1171 1070 format(&
1172 1x,'computing aBE = alpha-[Xa*D11+Ya*D21]*rd-[Xa*D12+Ya*D22]*qd',a)
1173 1080 format(&
1174 1x,'computing bBE = beta-[Xb*D11+Yb*D21]*rd-[Xb*D12+Yb*D22]*qd',a)
1175 1090 format(&
1176 1x,'computing rBE = rm-[Xr*D11+Yr*D21]*rd-[Xr*D12+Yr*D22]*qd',a)

```

```

1177 2000 format(&
1178 1x, 'computing qBE = qm-[Xq*D11+Yq*D21]*rd-[Xq*D12+Yq*D22]*qd', a)
1179 2010 format(&
1180 1x, 'computing CaaBE = Caa-[Xa*(D11*Xa"+D12*Ya")+Ya*(D21*Xa"+D22*Ya")]', a)
1181 2020 format(&
1182 1x, 'computing CrrBE = Crr-[Xr*(D11*Xr"+D12*Yr")+Yr*(D21*Xr"+D22*Yr")]', a)
1183 2030 format(&
1184 1x, 'computing CarBE = Car-[Xa*(D11*Xr"+D12*Yr")+Ya*(D21*Xr"+D22*Yr")]', a)
1185 2040 format(&
1186 1x, 'computing CbbBE = Cbb-[Xb*(D11*Xb"+D12*Yb")+Yb*(D21*Xb"+D22*Yb")]', a)
1187 2050 format(&
1188 1x, 'computing CqqBE = Cqq-[Xq*(D11*Xq"+D12*Yq")+Yq*(D21*Xq"+D22*Yq")]', a)
1189 2060 format(&
1190 1x, 'computing CbqBE = Cbq-[Xb*(D11*Xq"+D12*Yq")+Yb*(D21*Xq"+D22*Yq")]', a)
1191 2070 format(&
1192 1x, 'computing CabBE = Cab-[Xa*(D11*Xb"+D12*Yb")+Yb*(D21*Xb"+D22*Yb")]', a)
1193 2080 format(&
1194 1x, 'computing CaqBE = Caq-[Xa*(D11*Xq"+D12*Yq")+Ya*(D21*Xq"+D22*Yq")]', a)
1195 2090 format(&
1196 1x, 'computing CbrBE = Cbr-[Xb*(D11*Xr"+D12*Yr")+Yb*(D21*Xr"+D22*Yr")]', a)
1197 3000 format(&
1198 1x, 'computing CrqBE = Crq-[Xr*(D11*Xq"+D12*Yq")+Yr*(D21*Xq"+D22*Yq")]', a)
1199 3500 format(&
1200 'chi^2 = ' ,F8.3/, &
1201 'chi^2_d = (chi^2)/(number of responses) = ' ,F8.3)
1202 3600 format(/, &
1203 1x, 'done.')
1204
1205 END SUBROUTINE solvercase4
1206
1207 !+++++
1208
1209 END MODULE ModuleMultiPred

```

8.11 Module ModuleLapack.f90

```

1  MODULE ModuleLapack
2  !*****
3  !*
4  !* Module ModuleLapack encapsulates subroutines/functions for matrix/vector
5  !* operations, including multiplication and inverse, which call subroutines
6  !* from lapack.
7  !*
8  !*           FUNCTION inv(A) result(Ainv)
9  !*           FUNCTION multipMM(A, B) result(C)
10 !*           FUNCTION multipMV(A, B) result(C)
11 !*           FUNCTION is_positive_definite(A) result(info)
12 !*
13 !* All dependent subroutines are extracted from lapack and packaged in
14 !* this code. Therefore, no lapack installation is needed to run this
15 !* program.
16 !*
17 !* called by: MultiPredSolver
18 !* calls to: inv,multipMM,multipMV,DGEMM,DTRSM,XERBLA,LSAME,DLASWP,DSCAL,
19 !*           DGETF2,DGETRF,DGER,DLAMCH,DGETRF2,ILAENV,IEECK,DSWAP,DGEMV,
20 !*           DTRTRI,DTRMM,DTRTI2,DTRMV,DGETRI,DPBTF2,DSYR,DPOTF2,DDOT,
21 !*           DISNAN,DLAISNAN,DSYRK,DPBTRF

```

```

22  !*
23  !*****
24
25  IMPLICIT NONE
26
27  CONTAINS
28
29  ! =====
30
31  FUNCTION inv(A) result(Ainv)
32
33  ! -- Returns the inverse of a matrix calculated by finding the LU
34  ! -- decomposition.
35  ! -- developed by University of South Carolina.
36  !
37  INTEGER, PARAMETER :: DP = KIND(1.0D0)
38  ! Arguments-----
39  REAL(DP), dimension(:, :), intent(in) :: A
40  REAL(DP), dimension(size(A,1),size(A,2)) :: Ainv
41  ! Local-----
42  REAL(DP), dimension(size(A,1)) :: work           ! work array for LAPACK
43  INTEGER, dimension(size(A,1)) :: ipiv          ! pivot indices
44  INTEGER :: n, info
45
46  ! External procedures defined in LAPACK
47  !   external DGETRF
48  !   external DGETRI
49
50  ! Store A in Ainv to prevent it from being overwritten by LAPACK
51  Ainv = A
52  n = size(A,1)
53
54  ! DGETRF computes an LU factorization of a general M-by-N matrix A
55  ! using partial pivoting with row interchanges.
56  call DGETRF(n, n, Ainv, n, ipiv, info)
57
58  if (info /= 0) then
59     stop 'Matrix is numerically singular!'
60  end if
61
62  ! DGETRI computes the inverse of a matrix using the LU factorization
63  ! computed by DGETRF.
64  call DGETRI(n, Ainv, n, ipiv, work, n, info)
65
66  if (info /= 0) then
67     stop 'Matrix inversion failed!'
68  end if
69
70  END FUNCTION inv
71
72  ! =====
73
74  FUNCTION multipMM(A, B) result(C)
75
76  ! -- Returns the product of a matrix C= A*B calculated by using DGEMM(),
77  ! -- a subroutine of LAPACK, where A and B are 2D matrices.
78  ! -- developed by University of South Carolina.
79  !
80  INTEGER, PARAMETER :: DP = KIND(1.0D0)
81  ! Arguments-----
82  REAL(DP), dimension(:, :), intent(in) :: A,B

```

```

83     REAL(DP), dimension(size(A,1),size(B,2)) :: C
84     ! Local-----
85     INTEGER                :: M, K, N, LDA, LDB, LDC
86     REAL(DP)               :: ALPHA, BETA
87
88     ! External procedures defined in LAPACK
89     !   external DGEMM
90
91     ! initialize data for matrix multiplication C=A*B
92     C      = 0.0
93     ALPHA  = 1.0
94     BETA   = 0.0
95     M      = size(A,1)
96     N      = size(B,2)
97     K      = size(A,2)
98     LDA    = M
99     LDB    = size(B,1)
100    LDC    = M
101
102    ! Computing matrix product using DGEMM subroutine
103    call DGEMM('N', 'N', M, N, K, ALPHA, A, LDA, B, LDB, BETA, C, LDC)
104
105    END FUNCTION multipMM
106
107    ! =====
108
109    FUNCTION multipMV(A, B) result(C)
110
111    ! -- Returns the product of a matrix C= A*B calculated by using DGEMM(),
112    ! -- a subroutine of LAPACK, where A is a 2D matrix and B is a column vector.
113    ! -- developed by University of South Carolina.
114    !
115    INTEGER, PARAMETER :: DP = KIND(1.0D0)
116    ! Arguments-----
117    REAL(DP), dimension(:, :), intent(in) :: A
118    REAL(DP), dimension(:), intent(in)    :: B
119    REAL(DP), dimension(size(A,1))        :: C
120    ! Local-----
121    INTEGER                :: M, K, N, LDA, LDB, LDC
122    REAL(DP)               :: ALPHA, BETA
123
124    ! External procedures defined in LAPACK
125    !   external DGEMM
126
127    ! initialize data for matrix multiplication C=A*B
128    C      = 0.0
129    ALPHA  = 1.0
130    BETA   = 0.0
131    M      = size(A,1)
132    N      = 1
133    K      = size(A,2)
134    LDA    = M
135    LDB    = size(B,1)
136    LDC    = M
137
138    ! Computing matrix product using DGEMM subroutine
139    call DGEMM('N', 'N', M, N, K, ALPHA, A, LDA, B, LDB, BETA, C, LDC)
140
141    END FUNCTION multipMV
142
143    ! =====

```

```

144
145 FUNCTION multipVV(A, B) result(C)
146
147 ! -- Returns the product of vectors C= A*B calculated by using SDSDOT(),
148 ! -- a subroutine of LAPACK, where A and B are both vectors.
149 ! -- developed by University of South Carolina.
150 !
151 INTEGER, PARAMETER :: DP = KIND(1.0D0)
152 ! Arguments-----
153 REAL(DP), dimension(:), intent(in) :: A(:)
154 REAL(DP), dimension(:), intent(in) :: B(:)
155 REAL(DP) :: C
156 ! Local-----
157 INTEGER :: N, INCX, INCY
158 REAL :: SB
159 ! initialize arguments for SDSDOT()
160 N = size(B,1)
161 SB = 0.0
162 INCX = 1
163 INCY = 1
164
165 ! Computing matrix product using DGEMM subroutine
166 C = SDSDOT(N,SB,A,INCX,B,INCY)
167
168 END FUNCTION multipVV
169 ! =====
170
171 INTEGER FUNCTION is_positive_definite(A) result(info)
172
173 ! -- check if matrix A positive definite, using the Cholesky factorization
174 ! -- and check to see if such a factorization exists.
175 ! -- developed by University of South Carolina.
176 !
177 INTEGER, PARAMETER :: DP = KIND(1.0D0)
178 ! Arguments-----
179 REAL(DP), dimension(:,:), intent(in) :: A
180 ! Local-----
181 INTEGER :: i, ifail, j, kd, ldab, n
182 INTEGER :: ARow, AColumn, nZ
183 LOGICAL :: flag
184 CHARACTER (1) :: uplo
185 REAL(DP), ALLOCATABLE :: AB(:, :)
186
187 !find the array size
188 ARow = size(A,1)
189 AColumn = size(A,2)
190
191 nZ = 0
192 flag = .false.
193 !find the number of superdiagonals of the Upper triangle of A,
194 jloop: do j=1, AColumn
195     iloop: do i=1, ARow-j+1
196         if(A(i,i+j-1)/= 0.0) then
197             flag = .true.
198             exit iloop
199         end if
200     end do iloop
201     if(flag) then
202         nZ = nZ + 1
203     end if
204     flag = .false.

```

```

205     end do jloop
206
207     n = AColumn
208     kd = nZ - 1
209     ldab = kd + 1
210     ALLOCATE (AB(ldab,n))
211     AB = 0.0
212
213     ! write the lower triangle of the symmetric band matrix A, stored in
214     ! the first KD+1 rows of the array, as required by SUBROUTINE DPBTRF.
215     do i=1, ldab
216         do j=1, n-i+1
217             AB(i,j) = A(i+j-1,j)
218         end do
219     end do
220
221     uplo = 'L'
222     ! call dpbtrf to factorize A
223     CALL dpbtrf(uplo,n,kd,AB,ldab,info)
224
225     END FUNCTION is_positive_definite
226
227     ! =====
228     SUBROUTINE DGEMM(TRANSA,TRANSB,M,N,K,ALPHA,A,LDA,B,LDB,BETA,C,LDC)
229     !
230     ! -- Reference BLAS level3 routine (version 3.7.0) --
231     ! -- Reference BLAS is a software package provided by Univ. of Tennessee, --
232     ! -- Univ. of California Berkeley, Univ. of Colorado Denver and NAG Ltd.---
233     ! December 2016
234     !
235     ! .. Scalar Arguments ..
236     DOUBLE PRECISION ALPHA,BETA
237     INTEGER K,LDA,LDB,LDC,M,N
238     CHARACTER TRANSA,TRANSB
239     !
240     ! .. Array Arguments ..
241     DOUBLE PRECISION A(LDA,*),B(LDB,*),C(LDC,*)
242     !
243     !
244     ! =====
245     !
246     ! .. External Functions ..
247     LOGICAL LSAME
248     EXTERNAL LSAME
249     !
250     ! .. External Subroutines ..
251     EXTERNAL XERBLA
252     !
253     ! .. Intrinsic Functions ..
254     INTRINSIC MAX
255     !
256     ! .. Local Scalars ..
257     DOUBLE PRECISION TEMP
258     INTEGER I,INFO,J,L,NCOLA,NROWA,NROWB
259     LOGICAL NOTA,NOTB
260     !
261     ! .. Parameters ..
262     DOUBLE PRECISION ONE,ZERO
263     PARAMETER (ONE=1.0D+0,ZERO=0.0D+0)
264     !
265     !

```

```

266 ! Set NOTA and NOTB as true if A and B respectively are not
267 ! transposed and set NROWA, NCOLA and NROWB as the number of rows
268 ! and columns of A and the number of rows of B respectively.
269 !
270 NOTA = LSAME(TRANSA,'N')
271 NOTB = LSAME(TRANSB,'N')
272 IF (NOTA) THEN
273     NROWA = M
274     NCOLA = K
275 ELSE
276     NROWA = K
277     NCOLA = M
278 END IF
279 IF (NOTB) THEN
280     NROWB = K
281 ELSE
282     NROWB = N
283 END IF
284 !
285 ! Test the input parameters.
286 !
287 INFO = 0
288 IF ((.NOT.NOTA) .AND. (.NOT.LSAME(TRANSA,'C'))) .AND. &
289 & (.NOT.LSAME(TRANSA,'T')) THEN
290     INFO = 1
291 ELSE IF ((.NOT.NOTB) .AND. (.NOT.LSAME(TRANSB,'C'))) .AND. &
292 & (.NOT.LSAME(TRANSB,'T')) THEN
293     INFO = 2
294 ELSE IF (M.LT.0) THEN
295     INFO = 3
296 ELSE IF (N.LT.0) THEN
297     INFO = 4
298 ELSE IF (K.LT.0) THEN
299     INFO = 5
300 ELSE IF (LDA.LT.MAX(1,NROWA)) THEN
301     INFO = 8
302 ELSE IF (LDB.LT.MAX(1,NROWB)) THEN
303     INFO = 10
304 ELSE IF (LDC.LT.MAX(1,M)) THEN
305     INFO = 13
306 END IF
307 IF (INFO.NE.0) THEN
308     CALL XERBLA('DGEMM ',INFO)
309     RETURN
310 END IF
311 !
312 ! Quick return if possible.
313 !
314 IF ((M.EQ.0) .OR. (N.EQ.0) .OR. &
315 & (((ALPHA.EQ.ZERO).OR. (K.EQ.0)).AND. (BETA.EQ.ONE))) RETURN
316 !
317 ! And if alpha.eq.zero.
318 !
319 IF (ALPHA.EQ.ZERO) THEN
320     IF (BETA.EQ.ZERO) THEN
321         DO 20 J = 1,N
322             DO 10 I = 1,M
323                 C(I,J) = ZERO
324             CONTINUE
325         CONTINUE
326     ELSE

```



```

327         DO 40 J = 1,N
328             DO 30 I = 1,M
329                 C(I,J) = BETA*C(I,J)
330             30 CONTINUE
331         40 CONTINUE
332     END IF
333     RETURN
334 END IF
335 !
336 ! Start the operations.
337 !
338 IF (NOTB) THEN
339     IF (NOTA) THEN
340 !
341 ! Form C := alpha*A*B + beta*C.
342 !
343         DO 90 J = 1,N
344             IF (BETA.EQ.ZERO) THEN
345                 DO 50 I = 1,M
346                     C(I,J) = ZERO
347                 50 CONTINUE
348             ELSE IF (BETA.NE.ONE) THEN
349                 DO 60 I = 1,M
350                     C(I,J) = BETA*C(I,J)
351                 60 CONTINUE
352             END IF
353             DO 80 L = 1,K
354                 TEMP = ALPHA*B(L,J)
355                 DO 70 I = 1,M
356                     C(I,J) = C(I,J) + TEMP*A(I,L)
357                 70 CONTINUE
358             80 CONTINUE
359         90 CONTINUE
360     ELSE
361 !
362 ! Form C := alpha*A**T*B + beta*C
363 !
364         DO 120 J = 1,N
365             DO 110 I = 1,M
366                 TEMP = ZERO
367                 DO 100 L = 1,K
368                     TEMP = TEMP + A(L,I)*B(L,J)
369                 100 CONTINUE
370             IF (BETA.EQ.ZERO) THEN
371                 C(I,J) = ALPHA*TEMP
372             ELSE
373                 C(I,J) = ALPHA*TEMP + BETA*C(I,J)
374             END IF
375         110 CONTINUE
376     120 CONTINUE
377     END IF
378 ELSE
379     IF (NOTA) THEN
380 !
381 ! Form C := alpha*A*B**T + beta*C
382 !
383         DO 170 J = 1,N
384             IF (BETA.EQ.ZERO) THEN
385                 DO 130 I = 1,M
386                     C(I,J) = ZERO
387                 130 CONTINUE

```

```

388         ELSE IF (BETA.NE.ONE) THEN
389             DO 140 I = 1,M
390                 C(I,J) = BETA*C(I,J)
391         140         CONTINUE
392         END IF
393         DO 160 L = 1,K
394             TEMP = ALPHA*B(J,L)
395             DO 150 I = 1,M
396                 C(I,J) = C(I,J) + TEMP*A(I,L)
397         150         CONTINUE
398         160         CONTINUE
399         170         CONTINUE
400     ELSE
401 !
402 !         Form C := alpha*A**T*B**T + beta*C
403 !
404         DO 200 J = 1,N
405             DO 190 I = 1,M
406                 TEMP = ZERO
407                 DO 180 L = 1,K
408                     TEMP = TEMP + A(L,I)*B(J,L)
409         180                 CONTINUE
410                 IF (BETA.EQ.ZERO) THEN
411                     C(I,J) = ALPHA*TEMP
412                 ELSE
413                     C(I,J) = ALPHA*TEMP + BETA*C(I,J)
414                 END IF
415         190                 CONTINUE
416         200             CONTINUE
417         END IF
418     END IF
419 !
420     RETURN
421 !
422 !     End of DGEMM .
423 !
424     END SUBROUTINE DGEMM
425
426 !
427 ! =====
428     SUBROUTINE DTRSM(SIDE,UPLO,TRANSA,DIAG,M,N,ALPHA,A,LDA,B,LDB)
429 !
430 ! -- Reference BLAS level3 routine (version 3.7.0) --
431 ! -- Reference BLAS is a software package provided by Univ. of Tennessee, --
432 ! -- Univ. of California Berkeley, Univ. of Colorado Denver and NAG Ltd.--
433 ! December 2016
434 !
435 ! .. Scalar Arguments ..
436     DOUBLE PRECISION ALPHA
437     INTEGER LDA,LDB,M,N
438     CHARACTER DIAG,SIDE,TRANSA,UPLO
439 ! ..
440 ! .. Array Arguments ..
441     DOUBLE PRECISION A(LDA,*),B(LDB,*)
442 ! ..
443 !
444 ! =====
445 !
446 ! .. External Functions ..
447 !     LOGICAL LSAME
448 !     EXTERNAL LSAME

```

```

449 ! ..
450 ! .. External Subroutines ..
451 !   EXTERNAL XERBLA
452 ! ..
453 ! .. Intrinsic Functions ..
454 INTRINSIC MAX
455 ! ..
456 ! .. Local Scalars ..
457 DOUBLE PRECISION TEMP
458 INTEGER I,INFO,J,K,NROWA
459 LOGICAL LSIDE,NOUNIT,UPPER
460 ! ..
461 ! .. Parameters ..
462 DOUBLE PRECISION ONE,ZERO
463 PARAMETER (ONE=1.0D+0,ZERO=0.0D+0)
464 ! ..
465 !
466 ! Test the input parameters.
467 !
468 LSIDE = LSAME(SIDE,'L')
469 IF (LSIDE) THEN
470     NROWA = M
471 ELSE
472     NROWA = N
473 END IF
474 NOUNIT = LSAME(DIAG,'N')
475 UPPER = LSAME(UPLO,'U')
476 !
477 INFO = 0
478 IF ((.NOT.LSIDE) .AND. (.NOT.LSAME(SIDE,'R'))) THEN
479     INFO = 1
480 ELSE IF ((.NOT.UPPER) .AND. (.NOT.LSAME(UPLO,'L'))) THEN
481     INFO = 2
482 ELSE IF ((.NOT.LSAME(TRANSA,'N')) .AND.&
483         (.NOT.LSAME(TRANSA,'T')) .AND.&
484         (.NOT.LSAME(TRANSA,'C'))) THEN
485     INFO = 3
486 ELSE IF ((.NOT.LSAME(DIAG,'U')) .AND. (.NOT.LSAME(DIAG,'N'))) THEN
487     INFO = 4
488 ELSE IF (M.LT.0) THEN
489     INFO = 5
490 ELSE IF (N.LT.0) THEN
491     INFO = 6
492 ELSE IF (LDA.LT.MAX(1,NROWA)) THEN
493     INFO = 9
494 ELSE IF (LDB.LT.MAX(1,M)) THEN
495     INFO = 11
496 END IF
497 IF (INFO.NE.0) THEN
498     CALL XERBLA('DTRSM ',INFO)
499     RETURN
500 END IF
501 !
502 ! Quick return if possible.
503 !
504 IF (M.EQ.0 .OR. N.EQ.0) RETURN
505 !
506 ! And when alpha.eq.zero.
507 !
508 IF (ALPHA.EQ.ZERO) THEN
509     DO 20 J = 1,N

```

```

510             DO 10 I = 1,M
511                 B(I,J) = ZERO
512     10         CONTINUE
513     20         CONTINUE
514             RETURN
515     END IF
516 !
517 !     Start the operations.
518 !
519     IF (LSIDE) THEN
520         IF (LSAME(TRANSA, 'N')) THEN
521 !
522 !         Form B := alpha*inv( A )*B.
523 !
524             IF (UPPER) THEN
525                 DO 60 J = 1,N
526                     IF (ALPHA.NE.ONE) THEN
527                         DO 30 I = 1,M
528                             B(I,J) = ALPHA*B(I,J)
529     30             CONTINUE
530                     END IF
531                 DO 50 K = M,1,-1
532                     IF (B(K,J).NE.ZERO) THEN
533                         IF (NOUNIT) B(K,J) = B(K,J)/A(K,K)
534                         DO 40 I = 1,K - 1
535                             B(I,J) = B(I,J) - B(K,J)*A(I,K)
536     40             CONTINUE
537                     END IF
538                 CONTINUE
539     50         CONTINUE
540             ELSE
541                 DO 100 J = 1,N
542                     IF (ALPHA.NE.ONE) THEN
543                         DO 70 I = 1,M
544                             B(I,J) = ALPHA*B(I,J)
545     70             CONTINUE
546                     END IF
547                 DO 90 K = 1,M
548                     IF (B(K,J).NE.ZERO) THEN
549                         IF (NOUNIT) B(K,J) = B(K,J)/A(K,K)
550                         DO 80 I = K + 1,M
551                             B(I,J) = B(I,J) - B(K,J)*A(I,K)
552     80             CONTINUE
553                     END IF
554                 CONTINUE
555     90         CONTINUE
556     100        CONTINUE
557             END IF
558     ELSE
559 !
560 !         Form B := alpha*inv( A**T )*B.
561 !
562             IF (UPPER) THEN
563                 DO 130 J = 1,N
564                     DO 120 I = 1,M
565                         TEMP = ALPHA*B(I,J)
566                         DO 110 K = 1,I - 1
567                             TEMP = TEMP - A(K,I)*B(K,J)
568     110            CONTINUE
569                         IF (NOUNIT) TEMP = TEMP/A(I,I)
570                         B(I,J) = TEMP
571     120            CONTINUE

```

```

571 130      CONTINUE
572      ELSE
573          DO 160 J = 1,N
574              DO 150 I = M,1,-1
575                  TEMP = ALPHA*B(I,J)
576                  DO 140 K = I + 1,M
577                      TEMP = TEMP - A(K,I)*B(K,J)
578 140          CONTINUE
579                  IF (NOUNIT) TEMP = TEMP/A(I,I)
580                  B(I,J) = TEMP
581 150          CONTINUE
582 160          CONTINUE
583          END IF
584      END IF
585  ELSE
586      IF (LSAME(TRANSA,'N')) THEN
587  !
588  !      Form B := alpha*B*inv( A ).
589  !
590          IF (UPPER) THEN
591              DO 210 J = 1,N
592                  IF (ALPHA.NE.ONE) THEN
593                      DO 170 I = 1,M
594                          B(I,J) = ALPHA*B(I,J)
595 170          CONTINUE
596                  END IF
597                  DO 190 K = 1,J - 1
598                      IF (A(K,J).NE.ZERO) THEN
599                          DO 180 I = 1,M
600                              B(I,J) = B(I,J) - A(K,J)*B(I,K)
601 180          CONTINUE
602                      END IF
603 190          CONTINUE
604                  IF (NOUNIT) THEN
605                      TEMP = ONE/A(J,J)
606                      DO 200 I = 1,M
607                          B(I,J) = TEMP*B(I,J)
608 200          CONTINUE
609                  END IF
610 210          CONTINUE
611          ELSE
612              DO 260 J = N,1,-1
613                  IF (ALPHA.NE.ONE) THEN
614                      DO 220 I = 1,M
615                          B(I,J) = ALPHA*B(I,J)
616 220          CONTINUE
617                  END IF
618                  DO 240 K = J + 1,N
619                      IF (A(K,J).NE.ZERO) THEN
620                          DO 230 I = 1,M
621                              B(I,J) = B(I,J) - A(K,J)*B(I,K)
622 230          CONTINUE
623                      END IF
624 240          CONTINUE
625                  IF (NOUNIT) THEN
626                      TEMP = ONE/A(J,J)
627                      DO 250 I = 1,M
628                          B(I,J) = TEMP*B(I,J)
629 250          CONTINUE
630                  END IF
631 260          CONTINUE

```

```

632         END IF
633     ELSE
634 !
635 !         Form B := alpha*B*inv( A**T ).
636 !
637         IF (UPPER) THEN
638             DO 310 K = N,1,-1
639                 IF (NOUNIT) THEN
640                     TEMP = ONE/A(K,K)
641                     DO 270 I = 1,M
642                         B(I,K) = TEMP*B(I,K)
643 270                     CONTINUE
644                 END IF
645                 DO 290 J = 1,K - 1
646                     IF (A(J,K).NE.ZERO) THEN
647                         TEMP = A(J,K)
648                         DO 280 I = 1,M
649                             B(I,J) = B(I,J) - TEMP*B(I,K)
650 280                         CONTINUE
651                     END IF
652 290                 CONTINUE
653                 IF (ALPHA.NE.ONE) THEN
654                     DO 300 I = 1,M
655                         B(I,K) = ALPHA*B(I,K)
656 300                     CONTINUE
657                 END IF
658 310                 CONTINUE
659             ELSE
660                 DO 360 K = 1,N
661                     IF (NOUNIT) THEN
662                         TEMP = ONE/A(K,K)
663                         DO 320 I = 1,M
664                             B(I,K) = TEMP*B(I,K)
665 320                         CONTINUE
666                     END IF
667                     DO 340 J = K + 1,N
668                         IF (A(J,K).NE.ZERO) THEN
669                             TEMP = A(J,K)
670                             DO 330 I = 1,M
671                                 B(I,J) = B(I,J) - TEMP*B(I,K)
672 330                             CONTINUE
673                         END IF
674 340                     CONTINUE
675                     IF (ALPHA.NE.ONE) THEN
676                         DO 350 I = 1,M
677                             B(I,K) = ALPHA*B(I,K)
678 350                         CONTINUE
679                     END IF
680 360                 CONTINUE
681             END IF
682         END IF
683     END IF
684 !
685     RETURN
686 !
687 !     End of DTRSM .
688 !
689     END SUBROUTINE DTRSM
690 !
691 ! =====
692     SUBROUTINE XERBLA( SRNAME, INFO )

```

```

693 !
694 ! -- Reference BLAS level1 routine (version 3.7.0) --
695 ! -- Reference BLAS is a software package provided by Univ. of Tennessee, --
696 ! -- Univ. of California Berkeley, Univ. of Colorado Denver and NAG Ltd.---
697 ! December 2016
698 !
699 ! .. Scalar Arguments ..
700 ! CHARACTER*(*) SRNAME
701 ! INTEGER INFO
702 ! ..
703 !
704 ! =====
705 !
706 ! .. Intrinsic Functions ..
707 ! INTRINSIC LEN_TRIM
708 ! ..
709 ! .. Executable Statements ..
710 !
711 ! WRITE( *, FMT = 9999 )SRNAME( 1:LEN_TRIM( SRNAME ) ), INFO
712 !
713 ! STOP
714 !
715 ! 9999 FORMAT( ' ** On entry to ', A, ' parameter number ', I2, ' had ',&
716 ! 'an illegal value' )
717 !
718 ! End of XERBLA
719 !
720 ! END SUBROUTINE XERBLA
721 !
722 ! =====
723 ! LOGICAL FUNCTION LSAME(CA,CB)
724 !
725 ! -- Reference BLAS level1 routine (version 3.1) --
726 ! -- Reference BLAS is a software package provided by Univ. of Tennessee, --
727 ! -- Univ. of California Berkeley, Univ. of Colorado Denver and NAG Ltd.---
728 ! December 2016
729 !
730 ! .. Scalar Arguments ..
731 ! CHARACTER CA,CB
732 ! ..
733 !
734 ! =====
735 !
736 ! .. Intrinsic Functions ..
737 ! INTRINSIC ICHAR
738 ! ..
739 ! .. Local Scalars ..
740 ! INTEGER INTA,INTB,ZCODE
741 ! ..
742 !
743 ! Test if the characters are equal
744 !
745 ! LSAME = CA .EQ. CB
746 ! IF (LSAME) RETURN
747 !
748 ! Now test for equivalence if both characters are alphabetic.
749 !
750 ! ZCODE = ICHAR('Z')
751 !
752 ! Use 'Z' rather than 'A' so that ASCII can be detected on Prime
753 ! machines, on which ICHAR returns a value with bit 8 set.

```

```

754 !      ICHAR('A') on Prime machines returns 193 which is the same as
755 !      ICHAR('A') on an EBCDIC machine.
756 !
757 !      INTA = ICHAR(CA)
758 !      INTB = ICHAR(CB)
759 !
760 !      IF (ZCODE.EQ.90 .OR. ZCODE.EQ.122) THEN
761 !
762 !          ASCII is assumed - ZCODE is the ASCII code of either lower or
763 !          upper case 'Z'.
764 !
765 !              IF (INTA.GE.97 .AND. INTA.LE.122) INTA = INTA - 32
766 !              IF (INTB.GE.97 .AND. INTB.LE.122) INTB = INTB - 32
767 !
768 !      ELSE IF (ZCODE.EQ.233 .OR. ZCODE.EQ.169) THEN
769 !
770 !          EBCDIC is assumed - ZCODE is the EBCDIC code of either lower or
771 !          upper case 'Z'.
772 !
773 !              IF (INTA.GE.129 .AND. INTA.LE.137 .OR.&
774 ! &          INTA.GE.145 .AND. INTA.LE.153 .OR.&
775 ! &          INTA.GE.162 .AND. INTA.LE.169) INTA = INTA + 64
776 !              IF (INTB.GE.129 .AND. INTB.LE.137 .OR.&
777 ! &          INTB.GE.145 .AND. INTB.LE.153 .OR.&
778 ! &          INTB.GE.162 .AND. INTB.LE.169) INTB = INTB + 64
779 !
780 !      ELSE IF (ZCODE.EQ.218 .OR. ZCODE.EQ.250) THEN
781 !
782 !          ASCII is assumed, on Prime machines - ZCODE is the ASCII code
783 !          plus 128 of either lower or upper case 'Z'.
784 !
785 !              IF (INTA.GE.225 .AND. INTA.LE.250) INTA = INTA - 32
786 !              IF (INTB.GE.225 .AND. INTB.LE.250) INTB = INTB - 32
787 !      END IF
788 !      LSAME = INTA .EQ. INTB
789 !
790 !      RETURN
791 !
792 !      End of LSAME
793 !
794 !      END FUNCTION LSAME
795 !
796 !
797 !
798 !      =====
799 !      SUBROUTINE DLASWP( N, A, LDA, K1, K2, IPIV, INCX )
800 !
801 !      -- LAPACK auxiliary routine (version 3.7.1) --
802 !      -- LAPACK is a software package provided by Univ. of Tennessee, --
803 !      -- Univ. of California Berkeley, Univ. of Colorado Denver and NAG Ltd.--
804 !      June 2017
805 !
806 !      .. Scalar Arguments ..
807 !      INTEGER          INCX, K1, K2, LDA, N
808 !      ..
809 !      .. Array Arguments ..
810 !      INTEGER          IPIV( * )
811 !      DOUBLE PRECISION A( LDA, * )
812 !      ..
813 !
814 !      =====

```



```

815 !
816 ! .. Local Scalars ..
817 INTEGER I, I1, I2, INC, IP, IX, IX0, J, K, N32
818 DOUBLE PRECISION TEMP
819 !
820 ! .. Executable Statements ..
821 !
822 ! Interchange row I with row IPIV(K1+(I-K1)*abs(INCX)) for each of rows
823 ! K1 through K2.
824 !
825 IF( INCX.GT.0 ) THEN
826     IX0 = K1
827     I1 = K1
828     I2 = K2
829     INC = 1
830 ELSE IF( INCX.LT.0 ) THEN
831     IX0 = K1 + ( K1-K2 )*INCX
832     I1 = K2
833     I2 = K1
834     INC = -1
835 ELSE
836     RETURN
837 END IF
838 !
839 N32 = ( N / 32 ) * 32
840 IF( N32.NE.0 ) THEN
841     DO 30 J = 1, N32, 32
842         IX = IX0
843         DO 20 I = I1, I2, INC
844             IP = IPIV( IX )
845             IF( IP.NE.I ) THEN
846                 DO 10 K = J, J + 31
847                     TEMP = A( I, K )
848                     A( I, K ) = A( IP, K )
849                     A( IP, K ) = TEMP
850             10 CONTINUE
851             END IF
852             IX = IX + INCX
853         20 CONTINUE
854     30 CONTINUE
855     END IF
856     IF( N32.NE.N ) THEN
857         N32 = N32 + 1
858         IX = IX0
859         DO 50 I = I1, I2, INC
860             IP = IPIV( IX )
861             IF( IP.NE.I ) THEN
862                 DO 40 K = N32, N
863                     TEMP = A( I, K )
864                     A( I, K ) = A( IP, K )
865                     A( IP, K ) = TEMP
866             40 CONTINUE
867             END IF
868             IX = IX + INCX
869         50 CONTINUE
870     END IF
871 !
872 RETURN
873 !
874 ! End of DLASWP
875 !

```

```

876      END SUBROUTINE DLASWP
877
878      ! =====
879      SUBROUTINE DSCAL(N,DA,DX,INCX)
880      !
881      ! -- Reference BLAS level1 routine (version 3.8.0) --
882      ! -- Reference BLAS is a software package provided by Univ. of Tennessee, --
883      ! -- Univ. of California Berkeley, Univ. of Colorado Denver and NAG Ltd.--
884      ! November 2017
885      !
886      ! .. Scalar Arguments ..
887      DOUBLE PRECISION DA
888      INTEGER INCX,N
889      !
890      ! .. Array Arguments ..
891      DOUBLE PRECISION DX(*)
892      !
893      ! =====
894      !
895      ! .. Local Scalars ..
896      INTEGER I,M,MP1,NINCX
897      !
898      ! .. Intrinsic Functions ..
899      INTRINSIC MOD
900      !
901      !
902      IF (N.LE.0 .OR. INCX.LE.0) RETURN
903      IF (INCX.EQ.1) THEN
904      !
905      !     code for increment equal to 1
906      !
907      !
908      !     clean-up loop
909      !
910      M = MOD(N,5)
911      IF (M.NE.0) THEN
912      DO I = 1,M
913      DX(I) = DA*DX(I)
914      END DO
915      IF (N.LT.5) RETURN
916      END IF
917      MP1 = M + 1
918      DO I = MP1,N,5
919      DX(I) = DA*DX(I)
920      DX(I+1) = DA*DX(I+1)
921      DX(I+2) = DA*DX(I+2)
922      DX(I+3) = DA*DX(I+3)
923      DX(I+4) = DA*DX(I+4)
924      END DO
925      ELSE
926      !
927      !     code for increment not equal to 1
928      !
929      NINCX = N*INCX
930      DO I = 1,NINCX,INCX
931      DX(I) = DA*DX(I)
932      END DO
933      END IF
934      RETURN
935      END SUBROUTINE DSCAL
936

```

```

937
938 !
939 ! =====
940 !     SUBROUTINE DGETF2( M, N, A, LDA, IPIV, INFO )
941 !
942 ! -- LAPACK computational routine (version 3.7.0) --
943 ! -- LAPACK is a software package provided by Univ. of Tennessee, --
944 ! -- Univ. of California Berkeley, Univ. of Colorado Denver and NAG Ltd...--
945 !     December 2016
946 !
947 !     .. Scalar Arguments ..
948 !     INTEGER          INFO, LDA, M, N
949 !     ..
950 !     .. Array Arguments ..
951 !     INTEGER          IPIV( * )
952 !     DOUBLE PRECISION A( LDA, * )
953 !     ..
954 !
955 ! =====
956 !
957 !     .. Parameters ..
958 !     DOUBLE PRECISION ONE, ZERO
959 !     PARAMETER        ( ONE = 1.0D+0, ZERO = 0.0D+0 )
960 !     ..
961 !     .. Local Scalars ..
962 !     DOUBLE PRECISION SFMIN
963 !     INTEGER          I, J, JP
964 !     ..
965 !     .. External Functions ..
966 !     DOUBLE PRECISION DLAMCH
967 !     INTEGER          IDAMAX
968 !     EXTERNAL         DLAMCH, IDAMAX
969 !     ..
970 !     .. External Subroutines ..
971 !     EXTERNAL         DGER, DSCAL, DSWAP, XERBLA
972 !     ..
973 !     .. Intrinsic Functions ..
974 !     INTRINSIC        MAX, MIN
975 !     ..
976 !     .. Executable Statements ..
977 !
978 !     Test the input parameters.
979 !
980 !     INFO = 0
981 !     IF( M.LT.0 ) THEN
982 !       INFO = -1
983 !     ELSE IF( N.LT.0 ) THEN
984 !       INFO = -2
985 !     ELSE IF( LDA.LT.MAX( 1, M ) ) THEN
986 !       INFO = -4
987 !     END IF
988 !     IF( INFO.NE.0 ) THEN
989 !       CALL XERBLA( 'DGETF2', -INFO )
990 !       RETURN
991 !     END IF
992 !
993 !     Quick return if possible
994 !
995 !     IF( M.EQ.0 .OR. N.EQ.0 )&
996 !       RETURN
997 !

```

```

998 !      Compute machine safe minimum
999 !
1000 SFMIN = DLAMCH('S')
1001 !
1002 DO 10 J = 1, MIN( M, N )
1003 !
1004 !      Find pivot and test for singularity.
1005 !
1006 JP = J - 1 + IDAMAX( M-J+1, A( J, J ), 1 )
1007 IPIV( J ) = JP
1008 IF( A( JP, J ).NE.ZERO ) THEN
1009 !
1010 !      Apply the interchange to columns 1:N.
1011 !
1012 IF( JP.NE.J )&
1013     CALL DSWAP( N, A( J, 1 ), LDA, A( JP, 1 ), LDA )
1014 !
1015 !      Compute elements J+1:M of J-th column.
1016 !
1017 IF( J.LT.M ) THEN
1018     IF( ABS(A( J, J )) .GE. SFMIN ) THEN
1019         CALL DSCAL( M-J, ONE / A( J, J ), A( J+1, J ), 1 )
1020     ELSE
1021         DO 20 I = 1, M-J
1022             A( J+I, J ) = A( J+I, J ) / A( J, J )
1023     20     CONTINUE
1024     END IF
1025     END IF
1026 !
1027 ELSE IF( INFO.EQ.0 ) THEN
1028 !
1029     INFO = J
1030     END IF
1031 !
1032 IF( J.LT.MIN( M, N ) ) THEN
1033 !
1034 !      Update trailing submatrix.
1035 !
1036     CALL DGER( M-J, N-J, -ONE, A( J+1, J ), 1, A( J, J+1 ), LDA,&
1037             A( J+1, J+1 ), LDA )
1038     END IF
1039 10 CONTINUE
1040 RETURN
1041 !
1042 !      End of DGETF2
1043 !
1044 END SUBROUTINE DGETF2
1045 !
1046 ! =====
1047 SUBROUTINE DGETRF( M, N, A, LDA, IPIV, INFO )
1048 !
1049 ! -- LAPACK computational routine (version 3.7.0) --
1050 ! -- LAPACK is a software package provided by Univ. of Tennessee, --
1051 ! -- Univ. of California Berkeley, Univ. of Colorado Denver and NAG Ltd.---
1052 ! December 2016
1053 !
1054 ! .. Scalar Arguments ..
1055 INTEGER          INFO, LDA, M, N
1056 ! ..
1057 ! .. Array Arguments ..
1058 INTEGER          IPIV( * )

```

```

1059     DOUBLE PRECISION  A( LDA, * )
1060 !
1061 !
1062 ! =====
1063 !
1064 ! .. Parameters ..
1065     DOUBLE PRECISION  ONE
1066     PARAMETER          ( ONE = 1.0D+0 )
1067 !
1068 ! .. Local Scalars ..
1069     INTEGER            I, IINFO, J, JB, NB
1070 !
1071 ! .. External Subroutines ..
1072     EXTERNAL           DGEMM, DGETRF2, DLASWP, DTRSM, XERBLA
1073 !
1074 ! .. External Functions ..
1075     INTEGER            ILAENV
1076     EXTERNAL           ILAENV
1077 !
1078 ! .. Intrinsic Functions ..
1079     INTRINSIC          MAX, MIN
1080 !
1081 ! .. Executable Statements ..
1082 !
1083 ! Test the input parameters.
1084 !
1085     INFO = 0
1086     IF( M.LT.0 ) THEN
1087         INFO = -1
1088     ELSE IF( N.LT.0 ) THEN
1089         INFO = -2
1090     ELSE IF( LDA.LT.MAX( 1, M ) ) THEN
1091         INFO = -4
1092     END IF
1093     IF( INFO.NE.0 ) THEN
1094         CALL XERBLA( 'DGETRF', -INFO )
1095         RETURN
1096     END IF
1097 !
1098 ! Quick return if possible
1099 !
1100     IF( M.EQ.0 .OR. N.EQ.0 )&
1101         RETURN
1102 !
1103 ! Determine the block size for this environment.
1104 !
1105     NB = ILAENV( 1, 'DGETRF', ' ', M, N, -1, -1 )
1106     IF( NB.LE.1 .OR. NB.GE.MIN( M, N ) ) THEN
1107 !
1108 ! Use unblocked code.
1109 !
1110         CALL DGETRF2( M, N, A, LDA, IPIV, INFO )
1111     ELSE
1112 !
1113 ! Use blocked code.
1114 !
1115         DO 20 J = 1, MIN( M, N ), NB
1116             JB = MIN( MIN( M, N )-J+1, NB )
1117 !
1118 ! Factor diagonal and subdiagonal blocks and test for exact
1119 ! singularity.

```

```

1120 !
1121         CALL DGETRF2( M-J+1, JB, A( J, J ), LDA, IPIV( J ), IINFO )
1122 !
1123 !       Adjust INFO and the pivot indices.
1124 !
1125         IF( INFO.EQ.0 .AND. IINFO.GT.0 )&
1126             INFO = IINFO + J - 1
1127         DO 10 I = J, MIN( M, J+JB-1 )
1128             IPIV( I ) = J - 1 + IPIV( I )
1129     10     CONTINUE
1130 !
1131 !       Apply interchanges to columns 1:J-1.
1132 !
1133         CALL DLASWP( J-1, A, LDA, J, J+JB-1, IPIV, 1 )
1134 !
1135         IF( J+JB.LE.N ) THEN
1136 !
1137 !           Apply interchanges to columns J+JB:N.
1138 !
1139             CALL DLASWP( N-J-JB+1, A( 1, J+JB ), LDA, J, J+JB-1,&
1140                 IPIV, 1 )
1141 !
1142 !           Compute block row of U.
1143 !
1144             CALL DTRSM( 'Left', 'Lower', 'No transpose', 'Unit', JB,&
1145                 N-J-JB+1, ONE, A( J, J ), LDA, A( J, J+JB ),&
1146                 LDA )
1147             IF( J+JB.LE.M ) THEN
1148 !
1149 !               Update trailing submatrix.
1150 !
1151                 CALL DGEMM( 'No transpose', 'No transpose', M-J-JB+1,&
1152                     N-J-JB+1, JB, -ONE, A( J+JB, J ), LDA, &
1153                     A( J, J+JB ), LDA, ONE, A( J+JB, J+JB ), &
1154                     LDA )
1155             END IF
1156         END IF
1157     20     CONTINUE
1158     END IF
1159     RETURN
1160 !
1161 !       End of DGETRF
1162 !
1163     END SUBROUTINE DGETRF
1164 !
1165 ! =====
1166     SUBROUTINE DGER(M,N,ALPHA,X,INCX,Y,INCY,A,LDA)
1167 !
1168 ! -- Reference BLAS level2 routine (version 3.7.0) --
1169 ! -- Reference BLAS is a software package provided by Univ. of Tennessee, --
1170 ! -- Univ. of California Berkeley, Univ. of Colorado Denver and NAG Ltd.---
1171 !       December 2016
1172 !
1173 !       .. Scalar Arguments ..
1174     DOUBLE PRECISION ALPHA
1175     INTEGER INCX,INCY,LDA,M,N
1176 !
1177 !       .. Array Arguments ..
1178     DOUBLE PRECISION A(LDA,*),X(*),Y(*)
1179 !
1180 !

```

```

1181 ! =====
1182 !
1183 ! .. Parameters ..
1184 DOUBLE PRECISION ZERO
1185 PARAMETER (ZERO=0.0D+0)
1186 ! ..
1187 ! .. Local Scalars ..
1188 DOUBLE PRECISION TEMP
1189 INTEGER I,INFO,IX,J,JY,KX
1190 ! ..
1191 ! .. External Subroutines ..
1192 !   EXTERNAL XERBLA
1193 ! ..
1194 ! .. Intrinsic Functions ..
1195 INTRINSIC MAX
1196 ! ..
1197 !
1198 ! Test the input parameters.
1199 !
1200     INFO = 0
1201     IF (M.LT.0) THEN
1202         INFO = 1
1203     ELSE IF (N.LT.0) THEN
1204         INFO = 2
1205     ELSE IF (INCX.EQ.0) THEN
1206         INFO = 5
1207     ELSE IF (INCY.EQ.0) THEN
1208         INFO = 7
1209     ELSE IF (LDA.LT.MAX(1,M)) THEN
1210         INFO = 9
1211     END IF
1212     IF (INFO.NE.0) THEN
1213         CALL XERBLA('DGER ',INFO)
1214         RETURN
1215     END IF
1216 !
1217 ! Quick return if possible.
1218 !
1219     IF ((M.EQ.0) .OR. (N.EQ.0) .OR. (ALPHA.EQ.ZERO)) RETURN
1220 !
1221 ! Start the operations. In this version the elements of A are
1222 ! accessed sequentially with one pass through A.
1223 !
1224     IF (INCY.GT.0) THEN
1225         JY = 1
1226     ELSE
1227         JY = 1 - (N-1)*INCY
1228     END IF
1229     IF (INCX.EQ.1) THEN
1230         DO 20 J = 1,N
1231             IF (Y(JY).NE.ZERO) THEN
1232                 TEMP = ALPHA*Y(JY)
1233                 DO 10 I = 1,M
1234                     A(I,J) = A(I,J) + X(I)*TEMP
1235             10     CONTINUE
1236             END IF
1237             JY = JY + INCY
1238         20     CONTINUE
1239     ELSE
1240         IF (INCX.GT.0) THEN
1241             KX = 1

```

```

1242         ELSE
1243             KX = 1 - (M-1)*INCX
1244         END IF
1245         DO 40 J = 1,N
1246             IF (Y(JY).NE.ZERO) THEN
1247                 TEMP = ALPHA*Y(JY)
1248                 IX = KX
1249                 DO 30 I = 1,M
1250                     A(I,J) = A(I,J) + X(IX)*TEMP
1251                     IX = IX + INCX
1252             30             CONTINUE
1253                 END IF
1254                 JY = JY + INCY
1255             40             CONTINUE
1256         END IF
1257 !
1258         RETURN
1259 !
1260 !     End of DGER .
1261 !
1262         END SUBROUTINE DGER
1263
1264 !
1265 ! =====
1266         DOUBLE PRECISION FUNCTION DLAMCH( CMACH )
1267 !
1268 ! -- LAPACK auxiliary routine (version 3.7.0) --
1269 ! -- LAPACK is a software package provided by Univ. of Tennessee, --
1270 ! -- Univ. of California Berkeley, Univ. of Colorado Denver and NAG Ltd.--
1271 !     December 2016
1272 !
1273 ! .. Scalar Arguments ..
1274         CHARACTER          CMACH
1275 ! ..
1276 !
1277 ! =====
1278 !
1279 ! .. Parameters ..
1280         DOUBLE PRECISION  ONE, ZERO
1281         PARAMETER          ( ONE = 1.0D+0, ZERO = 0.0D+0 )
1282 ! ..
1283 ! .. Local Scalars ..
1284         DOUBLE PRECISION  RND, EPS, SFMIN, SMALL, RMACH
1285 ! ..
1286 ! .. External Functions ..
1287         LOGICAL           LSAME
1288         EXTERNAL          LSAME
1289 ! ..
1290 ! .. Intrinsic Functions ..
1291         INTRINSIC         DIGITS, EPSILON, HUGE, MAXEXPONENT,&
1292                         MINEXPONENT, RADIX, TINY
1293 ! ..
1294 ! .. Executable Statements ..
1295 !
1296 !
1297 !     Assume rounding, not chopping. Always.
1298 !
1299         RND = ONE
1300 !
1301         IF( ONE.EQ.RND ) THEN
1302             EPS = EPSILON(ZERO) * 0.5

```



```

1303     ELSE
1304         EPS = EPSILON(ZERO)
1305     END IF
1306 !
1307     IF( LSAME( CMACH, 'E' ) ) THEN
1308         RMACH = EPS
1309     ELSE IF( LSAME( CMACH, 'S' ) ) THEN
1310         SFMIN = TINY(ZERO)
1311         SMALL = ONE / HUGE(ZERO)
1312         IF( SMALL.GE.SFMIN ) THEN
1313 !
1314 !             Use SMALL plus a bit, to avoid the possibility of rounding
1315 !             causing overflow when computing 1/sfmin.
1316 !
1317             SFMIN = SMALL*( ONE+EPS )
1318         END IF
1319         RMACH = SFMIN
1320     ELSE IF( LSAME( CMACH, 'B' ) ) THEN
1321         RMACH = RADIX(ZERO)
1322     ELSE IF( LSAME( CMACH, 'P' ) ) THEN
1323         RMACH = EPS * RADIX(ZERO)
1324     ELSE IF( LSAME( CMACH, 'N' ) ) THEN
1325         RMACH = DIGITS(ZERO)
1326     ELSE IF( LSAME( CMACH, 'R' ) ) THEN
1327         RMACH = RND
1328     ELSE IF( LSAME( CMACH, 'M' ) ) THEN
1329         RMACH = MINEXPONENT(ZERO)
1330     ELSE IF( LSAME( CMACH, 'U' ) ) THEN
1331         RMACH = tiny(zero)
1332     ELSE IF( LSAME( CMACH, 'L' ) ) THEN
1333         RMACH = MAXEXPONENT(ZERO)
1334     ELSE IF( LSAME( CMACH, 'O' ) ) THEN
1335         RMACH = HUGE(ZERO)
1336     ELSE
1337         RMACH = ZERO
1338     END IF
1339 !
1340     DLAMCH = RMACH
1341     RETURN
1342 !
1343 !     End of DLAMCH
1344 !
1345     END FUNCTION DLAMCH
1346
1347 !
1348 ! =====
1349     INTEGER FUNCTION IDAMAX(N,DX,INCX)
1350 !
1351 ! -- Reference BLAS level1 routine (version 3.8.0) --
1352 ! -- Reference BLAS is a software package provided by Univ. of Tennessee, --
1353 ! -- Univ. of California Berkeley, Univ. of Colorado Denver and NAG Ltd.---
1354 !     November 2017
1355 !
1356 !     .. Scalar Arguments ..
1357     INTEGER INCX,N
1358 !     ..
1359 !     .. Array Arguments ..
1360     DOUBLE PRECISION DX(*)
1361 !     ..
1362 !
1363 ! =====

```

```

1364 !
1365 ! .. Local Scalars ..
1366 DOUBLE PRECISION DMAX
1367 INTEGER I,IX
1368 !
1369 ! .. Intrinsic Functions ..
1370 INTRINSIC DABS
1371 !
1372 IDAMAX = 0
1373 IF (N.LT.1 .OR. INCX.LE.0) RETURN
1374 IDAMAX = 1
1375 IF (N.EQ.1) RETURN
1376 IF (INCX.EQ.1) THEN
1377 !
1378 !     code for increment equal to 1
1379 !
1380     DMAX = DABS(DX(1))
1381     DO I = 2,N
1382         IF (DABS(DX(I)).GT.DMAX) THEN
1383             IDAMAX = I
1384             DMAX = DABS(DX(I))
1385         END IF
1386     END DO
1387 ELSE
1388 !
1389 !     code for increment not equal to 1
1390 !
1391     IX = 1
1392     DMAX = DABS(DX(1))
1393     IX = IX + INCX
1394     DO I = 2,N
1395         IF (DABS(DX(IX)).GT.DMAX) THEN
1396             IDAMAX = I
1397             DMAX = DABS(DX(IX))
1398         END IF
1399     IX = IX + INCX
1400 END DO
1401 END IF
1402 RETURN
1403 END FUNCTION IDAMAX
1404
1405 !
1406 ! =====
1407 RECURSIVE SUBROUTINE DGETRF2( M, N, A, LDA, IPIV, INFO )
1408 !
1409 ! -- LAPACK computational routine (version 3.7.0) --
1410 ! -- LAPACK is a software package provided by Univ. of Tennessee, --
1411 ! -- Univ. of California Berkeley, Univ. of Colorado Denver and NAG Ltd. ---
1412 !     June 2016
1413 !
1414 ! .. Scalar Arguments ..
1415 INTEGER INFO, LDA, M, N
1416 !
1417 ! .. Array Arguments ..
1418 INTEGER IPIV( * )
1419 DOUBLE PRECISION A( LDA, * )
1420 !
1421 !
1422 ! =====
1423 !
1424 ! .. Parameters ..

```

```

1425     DOUBLE PRECISION    ONE, ZERO
1426     PARAMETER           ( ONE = 1.0D+0, ZERO = 0.0D+0 )
1427 !
1428 ! .. Local Scalars ..
1429     DOUBLE PRECISION    SFMIN, TEMP
1430     INTEGER             I, IINFO, N1, N2
1431 !
1432 ! .. External Functions ..
1433 !     DOUBLE PRECISION    DLAMCH
1434 !     INTEGER             IDAMAX
1435 !     EXTERNAL            DLAMCH, IDAMAX
1436 !
1437 ! .. External Subroutines ..
1438 !     EXTERNAL            DGEMM, DSCAL, DLASWP, DTRSM, XERBLA
1439 !
1440 ! .. Intrinsic Functions ..
1441     INTRINSIC            MAX, MIN
1442 !
1443 ! .. Executable Statements ..
1444 !
1445 ! Test the input parameters
1446 !
1447     INFO = 0
1448     IF( M.LT.0 ) THEN
1449         INFO = -1
1450     ELSE IF( N.LT.0 ) THEN
1451         INFO = -2
1452     ELSE IF( LDA.LT.MAX( 1, M ) ) THEN
1453         INFO = -4
1454     END IF
1455     IF( INFO.NE.0 ) THEN
1456         CALL XERBLA( 'DGETRF2', -INFO )
1457         RETURN
1458     END IF
1459 !
1460 ! Quick return if possible
1461 !
1462     IF( M.EQ.0 .OR. N.EQ.0 )&
1463         RETURN
1464
1465     IF ( M.EQ.1 ) THEN
1466 !
1467 !     Use unblocked code for one row case
1468 !     Just need to handle IPIV and INFO
1469 !
1470         IPIV( 1 ) = 1
1471         IF ( A(1,1).EQ.ZERO )&
1472             INFO = 1
1473 !
1474     ELSE IF( N.EQ.1 ) THEN
1475 !
1476 !     Use unblocked code for one column case
1477 !
1478 !
1479 !     Compute machine safe minimum
1480 !
1481         SFMIN = DLAMCH('S')
1482 !
1483 !     Find pivot and test for singularity
1484 !
1485         I = IDAMAX( M, A( 1, 1 ), 1 )

```

```

1486      IPIV( 1 ) = I
1487      IF( A( I, 1 ).NE.ZERO ) THEN
1488 !
1489 !         Apply the interchange
1490 !
1491      IF( I.NE.1 ) THEN
1492          TEMP = A( 1, 1 )
1493          A( 1, 1 ) = A( I, 1 )
1494          A( I, 1 ) = TEMP
1495      END IF
1496 !
1497 !         Compute elements 2:M of the column
1498 !
1499      IF( ABS(A( 1, 1 )) .GE. SFMIN ) THEN
1500          CALL DSCAL( M-1, ONE / A( 1, 1 ), A( 2, 1 ), 1 )
1501      ELSE
1502          DO 10 I = 1, M-1
1503              A( 1+I, 1 ) = A( 1+I, 1 ) / A( 1, 1 )
1504 10      CONTINUE
1505      END IF
1506 !
1507      ELSE
1508          INFO = 1
1509      END IF
1510 !
1511      ELSE
1512 !
1513 !         Use recursive code
1514 !
1515      N1 = MIN( M, N ) / 2
1516      N2 = N-N1
1517 !
1518 !         [ A11 ]
1519 !         Factor [ --- ]
1520 !         [ A21 ]
1521 !
1522      CALL DGETRF2( M, N1, A, LDA, IPIV, IINFO )
1523
1524      IF ( INFO.EQ.0 .AND. IINFO.GT.0 )&
1525          INFO = IINFO
1526 !
1527 !         [ A12 ]
1528 !         Apply interchanges to [ --- ]
1529 !         [ A22 ]
1530 !
1531      CALL DLASWP( N2, A( 1, N1+1 ), LDA, 1, N1, IPIV, 1 )
1532 !
1533 !         Solve A12
1534 !
1535      CALL DTRSM( 'L', 'L', 'N', 'U', N1, N2, ONE, A, LDA,&
1536          A( 1, N1+1 ), LDA )
1537 !
1538 !         Update A22
1539 !
1540      CALL DGEMM( 'N', 'N', M-N1, N2, N1, -ONE, A( N1+1, 1 ), LDA,&
1541          A( 1, N1+1 ), LDA, ONE, A( N1+1, N1+1 ), LDA )
1542 !
1543 !         Factor A22
1544 !
1545      CALL DGETRF2( M-N1, N2, A( N1+1, N1+1 ), LDA, IPIV( N1+1 ),&
1546          IINFO )

```

```

1547 !
1548 !     Adjust INFO and the pivot indices
1549 !
1550 !     IF ( INFO.EQ.0 .AND. IINFO.GT.0 )&
1551 !         INFO = IINFO + N1
1552 !     DO 20 I = N1+1, MIN( M, N )
1553 !         IPIV( I ) = IPIV( I ) + N1
1554 20  CONTINUE
1555 !
1556 !     Apply interchanges to A21
1557 !
1558 !     CALL DLASWP( N1, A( 1, 1 ), LDA, N1+1, MIN( M, N), IPIV, 1 )
1559 !
1560 !     END IF
1561 !     RETURN
1562 !
1563 !     End of DGETRF2
1564 !
1565 !     END SUBROUTINE DGETRF2
1566 !
1567 ! =====
1568 !     INTEGER FUNCTION ILAENV( ISPEC, NAME, OPTS, N1, N2, N3, N4 )
1569 !
1570 !     -- LAPACK auxiliary routine (version 3.8.0) --
1571 !     -- LAPACK is a software package provided by Univ. of Tennessee, --
1572 !     -- Univ. of California Berkeley, Univ. of Colorado Denver and NAG Ltd.--
1573 !     November 2017
1574 !
1575 !     .. Scalar Arguments ..
1576 !     CHARACTER*( * )   NAME, OPTS
1577 !     INTEGER            ISPEC, N1, N2, N3, N4
1578 !     ..
1579 !
1580 !     =====
1581 !
1582 !     .. Local Scalars ..
1583 !     INTEGER            I, IC, IZ, NB, NBMIN, NX
1584 !     LOGICAL            CNAME, SNAME, TWOSTAGE
1585 !     CHARACTER          C1*1, C2*2, C4*2, C3*3, SUBNAM*16
1586 !     ..
1587 !     .. Intrinsic Functions ..
1588 !     INTRINSIC         CHAR, ICHAR, INT, MIN, REAL
1589 !     ..
1590 !     .. External Functions ..
1591 !     INTEGER            IIEEECK, IPARMQ, IPARAM2STAGE
1592 !     EXTERNAL          IIEEECK, IPARMQ, IPARAM2STAGE
1593 !     ..
1594 !     .. Executable Statements ..
1595 !
1596 !     GO TO ( 10, 10, 10, 80, 90, 100, 110, 120,&
1597 !            130, 140, 150, 160, 160, 160, 160, 160)ISPEC
1598 !
1599 !     Invalid value for ISPEC
1600 !
1601 !     ILAENV = -1
1602 !     RETURN
1603 !
1604 10  CONTINUE
1605 !
1606 !     Convert NAME to upper case if the first character is lower case.
1607 !

```

```

1608     ILAENV = 1
1609     SUBNAM = NAME
1610     IC = ICHAR( SUBNAM( 1: 1 ) )
1611     IZ = ICHAR( 'Z' )
1612     IF( IZ.EQ.90 .OR. IZ.EQ.122 ) THEN
1613 !
1614 !         ASCII character set
1615 !
1616         IF( IC.GE.97 .AND. IC.LE.122 ) THEN
1617             SUBNAM( 1: 1 ) = CHAR( IC-32 )
1618             DO 20 I = 2, 6
1619                 IC = ICHAR( SUBNAM( I: I ) )
1620                 IF( IC.GE.97 .AND. IC.LE.122 )&
1621                     SUBNAM( I: I ) = CHAR( IC-32 )
1622 20         CONTINUE
1623         END IF
1624 !
1625     ELSE IF( IZ.EQ.233 .OR. IZ.EQ.169 ) THEN
1626 !
1627 !         EBCDIC character set
1628 !
1629         IF( ( IC.GE.129 .AND. IC.LE.137 ) .OR.&
1630             ( IC.GE.145 .AND. IC.LE.153 ) .OR.&
1631             ( IC.GE.162 .AND. IC.LE.169 ) ) THEN
1632             SUBNAM( 1: 1 ) = CHAR( IC+64 )
1633             DO 30 I = 2, 6
1634                 IC = ICHAR( SUBNAM( I: I ) )
1635                 IF( ( IC.GE.129 .AND. IC.LE.137 ) .OR.&
1636                     ( IC.GE.145 .AND. IC.LE.153 ) .OR.&
1637                     ( IC.GE.162 .AND. IC.LE.169 ) ) SUBNAM( I:&
1638                     I ) = CHAR( IC+64 )
1639 30         CONTINUE
1640             END IF
1641 !
1642     ELSE IF( IZ.EQ.218 .OR. IZ.EQ.250 ) THEN
1643 !
1644 !         Prime machines: ASCII+128
1645 !
1646         IF( IC.GE.225 .AND. IC.LE.250 ) THEN
1647             SUBNAM( 1: 1 ) = CHAR( IC-32 )
1648             DO 40 I = 2, 6
1649                 IC = ICHAR( SUBNAM( I: I ) )
1650                 IF( IC.GE.225 .AND. IC.LE.250 )&
1651                     SUBNAM( I: I ) = CHAR( IC-32 )
1652 40         CONTINUE
1653             END IF
1654         END IF
1655 !
1656         C1 = SUBNAM( 1: 1 )
1657         SNAME = C1.EQ.'S' .OR. C1.EQ.'D'
1658         CNAME = C1.EQ.'C' .OR. C1.EQ.'Z'
1659         IF( .NOT.( CNAME .OR. SNAME ) )&
1660             RETURN
1661         C2 = SUBNAM( 2: 3 )
1662         C3 = SUBNAM( 4: 6 )
1663         C4 = C3( 2: 3 )
1664         TWOSTAGE = LEN( SUBNAM ) .GE.11&
1665             .AND. SUBNAM( 11: 11 ).EQ.'2'
1666 !
1667         GO TO ( 50, 60, 70 )ISPEC
1668 !

```

```

1669      50 CONTINUE
1670 !
1671 !      ISPEC = 1:  block size
1672 !
1673 !      In these examples, separate code is provided for setting NB for
1674 !      real and complex.  We assume that NB will take the same value in
1675 !      single or double precision.
1676 !
1677      NB = 1
1678 !
1679      IF( C2.EQ.'GE' ) THEN
1680          IF( C3.EQ.'TRF' ) THEN
1681              IF( SNAME ) THEN
1682                  NB = 64
1683              ELSE
1684                  NB = 64
1685              END IF
1686          ELSE IF( C3.EQ.'QRF' .OR. C3.EQ.'RQF' .OR. C3.EQ.'LQF' .OR.&
1687                C3.EQ.'QLF' ) THEN
1688              IF( SNAME ) THEN
1689                  NB = 32
1690              ELSE
1691                  NB = 32
1692              END IF
1693          ELSE IF( C3.EQ.'QR ' ) THEN
1694              IF( N3 .EQ. 1) THEN
1695                  IF( SNAME ) THEN
1696 !          M*N
1697                      IF ((N1*N2.LE.131072).OR.(N1.LE.8192)) THEN
1698                          NB = N1
1699                      ELSE
1700                          NB = 32768/N2
1701                      END IF
1702                      ELSE
1703                          IF ((N1*N2.LE.131072).OR.(N1.LE.8192)) THEN
1704                              NB = N1
1705                          ELSE
1706                              NB = 32768/N2
1707                          END IF
1708                      END IF
1709                  ELSE
1710                      IF( SNAME ) THEN
1711                          NB = 1
1712                      ELSE
1713                          NB = 1
1714                      END IF
1715                  END IF
1716              ELSE IF( C3.EQ.'LQ ' ) THEN
1717                  IF( N3 .EQ. 2) THEN
1718                      IF( SNAME ) THEN
1719 !          M*N
1720                          IF ((N1*N2.LE.131072).OR.(N1.LE.8192)) THEN
1721                              NB = N1
1722                          ELSE
1723                              NB = 32768/N2
1724                          END IF
1725                          ELSE
1726                              IF ((N1*N2.LE.131072).OR.(N1.LE.8192)) THEN
1727                                  NB = N1
1728                              ELSE
1729                                  NB = 32768/N2

```

```

1730         END IF
1731     END IF
1732 ELSE
1733     IF( SNAME ) THEN
1734         NB = 1
1735     ELSE
1736         NB = 1
1737     END IF
1738 END IF
1739 ELSE IF( C3.EQ.'HRD' ) THEN
1740     IF( SNAME ) THEN
1741         NB = 32
1742     ELSE
1743         NB = 32
1744     END IF
1745 ELSE IF( C3.EQ.'BRD' ) THEN
1746     IF( SNAME ) THEN
1747         NB = 32
1748     ELSE
1749         NB = 32
1750     END IF
1751 ELSE IF( C3.EQ.'TRI' ) THEN
1752     IF( SNAME ) THEN
1753         NB = 64
1754     ELSE
1755         NB = 64
1756     END IF
1757 END IF
1758 ELSE IF( C2.EQ.'PO' ) THEN
1759     IF( C3.EQ.'TRF' ) THEN
1760         IF( SNAME ) THEN
1761             NB = 64
1762         ELSE
1763             NB = 64
1764         END IF
1765     END IF
1766 ELSE IF( C2.EQ.'SY' ) THEN
1767     IF( C3.EQ.'TRF' ) THEN
1768         IF( SNAME ) THEN
1769             IF( TWOSTAGE ) THEN
1770                 NB = 192
1771             ELSE
1772                 NB = 64
1773             END IF
1774         ELSE
1775             IF( TWOSTAGE ) THEN
1776                 NB = 192
1777             ELSE
1778                 NB = 64
1779             END IF
1780         END IF
1781 ELSE IF( SNAME .AND. C3.EQ.'TRD' ) THEN
1782     NB = 32
1783 ELSE IF( SNAME .AND. C3.EQ.'GST' ) THEN
1784     NB = 64
1785 END IF
1786 ELSE IF( CNAME .AND. C2.EQ.'HE' ) THEN
1787     IF( C3.EQ.'TRF' ) THEN
1788         IF( TWOSTAGE ) THEN
1789             NB = 192
1790         ELSE

```



```

1791         NB = 64
1792     END IF
1793 ELSE IF( C3.EQ.'TRD' ) THEN
1794     NB = 32
1795 ELSE IF( C3.EQ.'GST' ) THEN
1796     NB = 64
1797 END IF
1798 ELSE IF( SNAME .AND. C2.EQ.'OR' ) THEN
1799     IF( C3( 1: 1 ).EQ.'G' ) THEN
1800         IF( C4.EQ.'QR' .OR. C4.EQ.'RQ' .OR. C4.EQ.'LQ' .OR. C4.EQ.&
1801             'QL' .OR. C4.EQ.'HR' .OR. C4.EQ.'TR' .OR. C4.EQ.'BR' )&
1802             THEN
1803             NB = 32
1804         END IF
1805     ELSE IF( C3( 1: 1 ).EQ.'M' ) THEN
1806         IF( C4.EQ.'QR' .OR. C4.EQ.'RQ' .OR. C4.EQ.'LQ' .OR. C4.EQ.&
1807             'QL' .OR. C4.EQ.'HR' .OR. C4.EQ.'TR' .OR. C4.EQ.'BR' )&
1808             THEN
1809             NB = 32
1810         END IF
1811     END IF
1812 ELSE IF( CNAME .AND. C2.EQ.'UN' ) THEN
1813     IF( C3( 1: 1 ).EQ.'G' ) THEN
1814         IF( C4.EQ.'QR' .OR. C4.EQ.'RQ' .OR. C4.EQ.'LQ' .OR. C4.EQ.&
1815             'QL' .OR. C4.EQ.'HR' .OR. C4.EQ.'TR' .OR. C4.EQ.'BR' )&
1816             THEN
1817             NB = 32
1818         END IF
1819     ELSE IF( C3( 1: 1 ).EQ.'M' ) THEN
1820         IF( C4.EQ.'QR' .OR. C4.EQ.'RQ' .OR. C4.EQ.'LQ' .OR. C4.EQ.&
1821             'QL' .OR. C4.EQ.'HR' .OR. C4.EQ.'TR' .OR. C4.EQ.'BR' )&
1822             THEN
1823             NB = 32
1824         END IF
1825     END IF
1826 ELSE IF( C2.EQ.'GB' ) THEN
1827     IF( C3.EQ.'TRF' ) THEN
1828         IF( SNAME ) THEN
1829             IF( N4.LE.64 ) THEN
1830                 NB = 1
1831             ELSE
1832                 NB = 32
1833             END IF
1834         ELSE
1835             IF( N4.LE.64 ) THEN
1836                 NB = 1
1837             ELSE
1838                 NB = 32
1839             END IF
1840         END IF
1841     END IF
1842 ELSE IF( C2.EQ.'PB' ) THEN
1843     IF( C3.EQ.'TRF' ) THEN
1844         IF( SNAME ) THEN
1845             IF( N2.LE.64 ) THEN
1846                 NB = 1
1847             ELSE
1848                 NB = 32
1849             END IF
1850         ELSE
1851             IF( N2.LE.64 ) THEN

```

```

1852             NB = 1
1853             ELSE
1854             NB = 32
1855             END IF
1856         END IF
1857     END IF
1858 ELSE IF( C2.EQ.'TR' ) THEN
1859     IF( C3.EQ.'TRI' ) THEN
1860     IF( SNAME ) THEN
1861     NB = 64
1862     ELSE
1863     NB = 64
1864     END IF
1865 ELSE IF ( C3.EQ.'EVC' ) THEN
1866     IF( SNAME ) THEN
1867     NB = 64
1868     ELSE
1869     NB = 64
1870     END IF
1871     END IF
1872 ELSE IF( C2.EQ.'LA' ) THEN
1873     IF( C3.EQ.'UUM' ) THEN
1874     IF( SNAME ) THEN
1875     NB = 64
1876     ELSE
1877     NB = 64
1878     END IF
1879     END IF
1880 ELSE IF( SNAME .AND. C2.EQ.'ST' ) THEN
1881     IF( C3.EQ.'EBZ' ) THEN
1882     NB = 1
1883     END IF
1884 ELSE IF( C2.EQ.'GG' ) THEN
1885     NB = 32
1886     IF( C3.EQ.'HD3' ) THEN
1887     IF( SNAME ) THEN
1888     NB = 32
1889     ELSE
1890     NB = 32
1891     END IF
1892     END IF
1893     END IF
1894     ILAENV = NB
1895     RETURN
1896 !
1897     60 CONTINUE
1898 !
1899 !     ISPEC = 2:  minimum block size
1900 !
1901     NBMIN = 2
1902     IF( C2.EQ.'GE' ) THEN
1903     IF( C3.EQ.'QRF' .OR. C3.EQ.'RQF' .OR. C3.EQ.'LQF' .OR. C3.EQ.&
1904     'QLF' ) THEN
1905     IF( SNAME ) THEN
1906     NBMIN = 2
1907     ELSE
1908     NBMIN = 2
1909     END IF
1910 ELSE IF( C3.EQ.'HRD' ) THEN
1911     IF( SNAME ) THEN
1912     NBMIN = 2

```

```

1913         ELSE
1914             NBMIN = 2
1915         END IF
1916     ELSE IF( C3.EQ.'BRD' ) THEN
1917         IF( SNAME ) THEN
1918             NBMIN = 2
1919         ELSE
1920             NBMIN = 2
1921         END IF
1922     ELSE IF( C3.EQ.'TRI' ) THEN
1923         IF( SNAME ) THEN
1924             NBMIN = 2
1925         ELSE
1926             NBMIN = 2
1927         END IF
1928     END IF
1929 ELSE IF( C2.EQ.'SY' ) THEN
1930     IF( C3.EQ.'TRF' ) THEN
1931         IF( SNAME ) THEN
1932             NBMIN = 8
1933         ELSE
1934             NBMIN = 8
1935         END IF
1936     ELSE IF( SNAME .AND. C3.EQ.'TRD' ) THEN
1937         NBMIN = 2
1938     END IF
1939 ELSE IF( CNAME .AND. C2.EQ.'HE' ) THEN
1940     IF( C3.EQ.'TRD' ) THEN
1941         NBMIN = 2
1942     END IF
1943 ELSE IF( SNAME .AND. C2.EQ.'OR' ) THEN
1944     IF( C3( 1: 1 ).EQ.'G' ) THEN
1945         IF( C4.EQ.'QR' .OR. C4.EQ.'RQ' .OR. C4.EQ.'LQ' .OR. C4.EQ.&
1946             'QL' .OR. C4.EQ.'HR' .OR. C4.EQ.'TR' .OR. C4.EQ.'BR' )&
1947             THEN
1948                 NBMIN = 2
1949             END IF
1950     ELSE IF( C3( 1: 1 ).EQ.'M' ) THEN
1951         IF( C4.EQ.'QR' .OR. C4.EQ.'RQ' .OR. C4.EQ.'LQ' .OR. C4.EQ.&
1952             'QL' .OR. C4.EQ.'HR' .OR. C4.EQ.'TR' .OR. C4.EQ.'BR' )&
1953             THEN
1954                 NBMIN = 2
1955             END IF
1956     END IF
1957 ELSE IF( CNAME .AND. C2.EQ.'UN' ) THEN
1958     IF( C3( 1: 1 ).EQ.'G' ) THEN
1959         IF( C4.EQ.'QR' .OR. C4.EQ.'RQ' .OR. C4.EQ.'LQ' .OR. C4.EQ.&
1960             'QL' .OR. C4.EQ.'HR' .OR. C4.EQ.'TR' .OR. C4.EQ.'BR' )&
1961             THEN
1962                 NBMIN = 2
1963             END IF
1964     ELSE IF( C3( 1: 1 ).EQ.'M' ) THEN
1965         IF( C4.EQ.'QR' .OR. C4.EQ.'RQ' .OR. C4.EQ.'LQ' .OR. C4.EQ.&
1966             'QL' .OR. C4.EQ.'HR' .OR. C4.EQ.'TR' .OR. C4.EQ.'BR' )&
1967             THEN
1968                 NBMIN = 2
1969             END IF
1970     END IF
1971 ELSE IF( C2.EQ.'GG' ) THEN
1972     NBMIN = 2
1973     IF( C3.EQ.'HD3' ) THEN

```

```

1974         NBMIN = 2
1975     END IF
1976 END IF
1977 ILAENV = NBMIN
1978 RETURN
1979 !
1980 70 CONTINUE
1981 !
1982 ! ISPEC = 3: crossover point
1983 !
1984 NX = 0
1985 IF( C2.EQ.'GE' ) THEN
1986     IF( C3.EQ.'QRF' .OR. C3.EQ.'RQF' .OR. C3.EQ.'LQF' .OR. C3.EQ.&
1987         'QLF' ) THEN
1988         IF( SNAME ) THEN
1989             NX = 128
1990         ELSE
1991             NX = 128
1992         END IF
1993     ELSE IF( C3.EQ.'HRD' ) THEN
1994         IF( SNAME ) THEN
1995             NX = 128
1996         ELSE
1997             NX = 128
1998         END IF
1999     ELSE IF( C3.EQ.'BRD' ) THEN
2000         IF( SNAME ) THEN
2001             NX = 128
2002         ELSE
2003             NX = 128
2004         END IF
2005     END IF
2006 ELSE IF( C2.EQ.'SY' ) THEN
2007     IF( SNAME .AND. C3.EQ.'TRD' ) THEN
2008         NX = 32
2009     END IF
2010 ELSE IF( CNAME .AND. C2.EQ.'HE' ) THEN
2011     IF( C3.EQ.'TRD' ) THEN
2012         NX = 32
2013     END IF
2014 ELSE IF( SNAME .AND. C2.EQ.'OR' ) THEN
2015     IF( C3( 1: 1 ).EQ.'G' ) THEN
2016         IF( C4.EQ.'QR' .OR. C4.EQ.'RQ' .OR. C4.EQ.'LQ' .OR. C4.EQ.&
2017             'QL' .OR. C4.EQ.'HR' .OR. C4.EQ.'TR' .OR. C4.EQ.'BR' )&
2018             THEN
2019             NX = 128
2020         END IF
2021     END IF
2022 ELSE IF( CNAME .AND. C2.EQ.'UN' ) THEN
2023     IF( C3( 1: 1 ).EQ.'G' ) THEN
2024         IF( C4.EQ.'QR' .OR. C4.EQ.'RQ' .OR. C4.EQ.'LQ' .OR. C4.EQ.&
2025             'QL' .OR. C4.EQ.'HR' .OR. C4.EQ.'TR' .OR. C4.EQ.'BR' )&
2026             THEN
2027             NX = 128
2028         END IF
2029     END IF
2030 ELSE IF( C2.EQ.'GG' ) THEN
2031     NX = 128
2032     IF( C3.EQ.'HD3' ) THEN
2033         NX = 128
2034     END IF

```

```

2035     END IF
2036     ILAENV = NX
2037     RETURN
2038 !
2039     80 CONTINUE
2040 !
2041 !     ISPEC = 4:  number of shifts (used by xHSEQR)
2042 !
2043     ILAENV = 6
2044     RETURN
2045 !
2046     90 CONTINUE
2047 !
2048 !     ISPEC = 5:  minimum column dimension (not used)
2049 !
2050     ILAENV = 2
2051     RETURN
2052 !
2053     100 CONTINUE
2054 !
2055 !     ISPEC = 6:  crossover point for SVD (used by xGELSS and xGESVD)
2056 !
2057     ILAENV = INT( REAL( MIN( N1, N2 ) ) * 1.6E0 )
2058     RETURN
2059 !
2060     110 CONTINUE
2061 !
2062 !     ISPEC = 7:  number of processors (not used)
2063 !
2064     ILAENV = 1
2065     RETURN
2066 !
2067     120 CONTINUE
2068 !
2069 !     ISPEC = 8:  crossover point for multishift (used by xHSEQR)
2070 !
2071     ILAENV = 50
2072     RETURN
2073 !
2074     130 CONTINUE
2075 !
2076 !     ISPEC = 9:  maximum size of the subproblems at the bottom of the
2077 !                 computation tree in the divide-and-conquer algorithm
2078 !                 (used by xGELSD and xGESDD)
2079 !
2080     ILAENV = 25
2081     RETURN
2082 !
2083     140 CONTINUE
2084 !
2085 !     ISPEC = 10: ieee NaN arithmetic can be trusted not to trap
2086 !
2087 !     ILAENV = 0
2088 !     ILAENV = 1
2089 !     IF( ILAENV.EQ.1 ) THEN
2090 !         ILAENV = IIEEECK( 1, 0.0, 1.0 )
2091 !     END IF
2092 !     RETURN
2093 !
2094     150 CONTINUE
2095 !

```

```

2096 !      ISPEC = 11: infinity arithmetic can be trusted not to trap
2097 !
2098 !      ILAENV = 0
2099 !      ILAENV = 1
2100 !      IF( ILAENV.EQ.1 ) THEN
2101 !          ILAENV = IEECK( 0, 0.0, 1.0 )
2102 !      END IF
2103 !      RETURN
2104 !
2105 ! 160 CONTINUE
2106 !
2107 !      12 <= ISPEC <= 16: xHSEQR or related subroutines.
2108 !
2109 !      ILAENV = IPARMQ( ISPEC, NAME, OPTS, N1, N2, N3, N4 )
2110 !      RETURN
2111 !
2112 !      End of ILAENV
2113 !
2114 !      END FUNCTION ILAENV
2115 !
2116 !
2117 ! =====
2118 !      INTEGER          FUNCTION IEECK( ISPEC, ZERO, ONE )
2119 !
2120 ! -- LAPACK auxiliary routine (version 3.7.0) --
2121 ! -- LAPACK is a software package provided by Univ. of Tennessee, --
2122 ! -- Univ. of California Berkeley, Univ. of Colorado Denver and NAG Ltd.---
2123 !      December 2016
2124 !
2125 !      .. Scalar Arguments ..
2126 !      INTEGER          ISPEC
2127 !      REAL             ONE, ZERO
2128 !      ..
2129 !
2130 ! =====
2131 !
2132 !      .. Local Scalars ..
2133 !      REAL             NAN1, NAN2, NAN3, NAN4, NAN5, NAN6, NEGINF,&
2134 !      &                NEGZRO, NEWZRO, POSINF
2135 !      ..
2136 !      .. Executable Statements ..
2137 !      IEECK = 1
2138 !
2139 !      POSINF = ONE / ZERO
2140 !      IF( POSINF.LE.ONE ) THEN
2141 !          IEECK = 0
2142 !          RETURN
2143 !      END IF
2144 !
2145 !      NEGINF = -ONE / ZERO
2146 !      IF( NEGINF.GE.ZERO ) THEN
2147 !          IEECK = 0
2148 !          RETURN
2149 !      END IF
2150 !
2151 !      NEGZRO = ONE / ( NEGINF+ONE )
2152 !      IF( NEGZRO.NE.ZERO ) THEN
2153 !          IEECK = 0
2154 !          RETURN
2155 !      END IF
2156 !

```

```

2157     NEGINF = ONE / NEGZRO
2158     IF( NEGINF.GE.ZERO ) THEN
2159         IEEECK = 0
2160         RETURN
2161     END IF
2162 !
2163     NEWZRO = NEGZRO + ZERO
2164     IF( NEWZRO.NE.ZERO ) THEN
2165         IEEECK = 0
2166         RETURN
2167     END IF
2168 !
2169     POSINF = ONE / NEWZRO
2170     IF( POSINF.LE.ONE ) THEN
2171         IEEECK = 0
2172         RETURN
2173     END IF
2174 !
2175     NEGINF = NEGINF*POSINF
2176     IF( NEGINF.GE.ZERO ) THEN
2177         IEEECK = 0
2178         RETURN
2179     END IF
2180 !
2181     POSINF = POSINF*POSINF
2182     IF( POSINF.LE.ONE ) THEN
2183         IEEECK = 0
2184         RETURN
2185     END IF
2186 !
2187 !
2188 ! Return if we were only asked to check infinity arithmetic
2189 !
2190     IF( ISPEC.EQ.0 )&
2191     & RETURN
2192 !
2193     NAN1 = POSINF + NEGINF
2194 !
2195     NAN2 = POSINF / NEGINF
2196 !
2197     NAN3 = POSINF / POSINF
2198 !
2199     NAN4 = POSINF*ZERO
2200 !
2201     NAN5 = NEGINF*NEGZRO
2202 !
2203     NAN6 = NAN5*ZERO
2204 !
2205     IF( NAN1.EQ.NAN1 ) THEN
2206         IEEECK = 0
2207         RETURN
2208     END IF
2209 !
2210     IF( NAN2.EQ.NAN2 ) THEN
2211         IEEECK = 0
2212         RETURN
2213     END IF
2214 !
2215     IF( NAN3.EQ.NAN3 ) THEN
2216         IEEECK = 0
2217         RETURN

```

```

2218     END IF
2219 !
2220     IF( NAN4.EQ.NAN4 ) THEN
2221         IEEECK = 0
2222         RETURN
2223     END IF
2224 !
2225     IF( NAN5.EQ.NAN5 ) THEN
2226         IEEECK = 0
2227         RETURN
2228     END IF
2229 !
2230     IF( NAN6.EQ.NAN6 ) THEN
2231         IEEECK = 0
2232         RETURN
2233     END IF
2234 !
2235     RETURN
2236     END FUNCTION IEEECK
2237
2238 ! =====
2239     INTEGER FUNCTION IPARMQ( ISPEC, NAME, OPTS, N, ILO, IHI, LWORK )
2240 !
2241 ! -- LAPACK auxiliary routine (version 3.7.1) --
2242 ! -- LAPACK is a software package provided by Univ. of Tennessee, --
2243 ! -- Univ. of California Berkeley, Univ. of Colorado Denver and NAG Ltd.--
2244 !   June 2017
2245 !
2246 ! .. Scalar Arguments ..
2247     INTEGER          IHI, ILO, ISPEC, LWORK, N
2248     CHARACTER        NAME*( * ), OPTS*( * )
2249 !
2250 ! =====
2251 ! .. Parameters ..
2252     INTEGER          INMIN, INWIN, INIBL, ISHFTS, IACC22
2253     PARAMETER       ( INMIN = 12, INWIN = 13, INIBL = 14,&
2254 & ISHFTS = 15, IACC22 = 16 )
2255     INTEGER          NMIN, K22MIN, KACMIN, NIBBLE, KNWSWP
2256     PARAMETER       ( NMIN = 75, K22MIN = 14, KACMIN = 14,&
2257 & NIBBLE = 14, KNWSWP = 500 )
2258     REAL             TWO
2259     PARAMETER       ( TWO = 2.0 )
2260 !
2261 ! .. Local Scalars ..
2262     INTEGER          NH, NS
2263     INTEGER          I, IC, IZ
2264     CHARACTER        SUBNAM*6
2265 !
2266 ! .. Intrinsic Functions ..
2267     INTRINSIC       LOG, MAX, MOD, NINT, REAL
2268 !
2269 ! .. Executable Statements ..
2270     IF( ( ISPEC.EQ.ISHFTS ) .OR. ( ISPEC.EQ.INWIN ) .OR.&
2271 & ( ISPEC.EQ.IACC22 ) ) THEN
2272 !
2273 !     ==== Set the number simultaneous shifts ====
2274 !
2275         NH = IHI - ILO + 1
2276         NS = 2
2277         IF( NH.GE.30 )&
2278 &         NS = 4

```



```

2279      IF( NH.GE.60 )&
2280      &      NS = 10
2281      IF( NH.GE.150 )&
2282      &      NS = MAX( 10, NH / NINT( LOG( REAL( NH ) ) / LOG( TWO ) ) )
2283      IF( NH.GE.590 )&
2284      &      NS = 64
2285      IF( NH.GE.3000 )&
2286      &      NS = 128
2287      IF( NH.GE.6000 )&
2288      &      NS = 256
2289      NS = MAX( 2, NS-MOD( NS, 2 ) )
2290      END IF
2291      !
2292      IF( ISPEC.EQ.INMIN ) THEN
2293      !
2294      !
2295      !      ===== Matrices of order smaller than NMIN get sent
2296      !      .      to xLAHQQR, the classic double shift algorithm.
2297      !      .      This must be at least 11. =====
2298      !
2299      IPARMQ = NMIN
2300      !
2301      ELSE IF( ISPEC.EQ.INIBL ) THEN
2302      !
2303      !      ===== INIBL: skip a multi-shift qr iteration and
2304      !      .      whenever aggressive early deflation finds
2305      !      .      at least (NIBBLE*(window size)/100) deflations. =====
2306      !
2307      IPARMQ = NIBBLE
2308      !
2309      ELSE IF( ISPEC.EQ.ISHFTS ) THEN
2310      !
2311      !      ===== NSHFTS: The number of simultaneous shifts =====
2312      !
2313      IPARMQ = NS
2314      !
2315      ELSE IF( ISPEC.EQ.INWIN ) THEN
2316      !
2317      !      ===== NW: deflation window size. =====
2318      !
2319      IF( NH.LE.KNWSWP ) THEN
2320      IPARMQ = NS
2321      ELSE
2322      IPARMQ = 3*NS / 2
2323      END IF
2324      !
2325      ELSE IF( ISPEC.EQ.IACC22 ) THEN
2326      !
2327      !      ===== IACC22: Whether to accumulate reflections
2328      !      .      before updating the far-from-diagonal elements
2329      !      .      and whether to use 2-by-2 block structure while
2330      !      .      doing it. A small amount of work could be saved
2331      !      .      by making this choice dependent also upon the
2332      !      .      NH=IHI-ILO+1.
2333      !
2334      !
2335      !      Convert NAME to upper case if the first character is lower case.
2336      !
2337      IPARMQ = 0
2338      SUBNAM = NAME
2339      IC = ICHAR( SUBNAM( 1: 1 ) )

```

```

2340      IZ = ICHAR( 'Z' )
2341      IF( IZ.EQ.90 .OR. IZ.EQ.122 ) THEN
2342 !
2343 !          ASCII character set
2344 !
2345      IF( IC.GE.97 .AND. IC.LE.122 ) THEN
2346          SUBNAM( 1: 1 ) = CHAR( IC-32 )
2347          DO I = 2, 6
2348              IC = ICHAR( SUBNAM( I: I ) )
2349              IF( IC.GE.97 .AND. IC.LE.122 )&
2350 &          SUBNAM( I: I ) = CHAR( IC-32 )
2351          END DO
2352      END IF
2353 !
2354      ELSE IF( IZ.EQ.233 .OR. IZ.EQ.169 ) THEN
2355 !
2356 !          EBCDIC character set
2357 !
2358      IF( ( IC.GE.129 .AND. IC.LE.137 ) .OR.&
2359 &      ( IC.GE.145 .AND. IC.LE.153 ) .OR.&
2360 &      ( IC.GE.162 .AND. IC.LE.169 ) ) THEN
2361          SUBNAM( 1: 1 ) = CHAR( IC+64 )
2362          DO I = 2, 6
2363              IC = ICHAR( SUBNAM( I: I ) )
2364              IF( ( IC.GE.129 .AND. IC.LE.137 ) .OR.&
2365 &      ( IC.GE.145 .AND. IC.LE.153 ) .OR.&
2366 &      ( IC.GE.162 .AND. IC.LE.169 ) )SUBNAM( I:&
2367 &      I ) = CHAR( IC+64 )
2368          END DO
2369      END IF
2370 !
2371      ELSE IF( IZ.EQ.218 .OR. IZ.EQ.250 ) THEN
2372 !
2373 !          Prime machines: ASCII+128
2374 !
2375      IF( IC.GE.225 .AND. IC.LE.250 ) THEN
2376          SUBNAM( 1: 1 ) = CHAR( IC-32 )
2377          DO I = 2, 6
2378              IC = ICHAR( SUBNAM( I: I ) )
2379              IF( IC.GE.225 .AND. IC.LE.250 )&
2380 &          SUBNAM( I: I ) = CHAR( IC-32 )
2381          END DO
2382      END IF
2383  END IF
2384 !
2385      IF( SUBNAM( 2:6 ).EQ.'GGHRD' .OR.&
2386 &      SUBNAM( 2:6 ).EQ.'GGHD3' ) THEN
2387          IPARMQ = 1
2388          IF( NH.GE.K22MIN )&
2389 &          IPARMQ = 2
2390      ELSE IF ( SUBNAM( 4:6 ).EQ.'EXC' ) THEN
2391          IF( NH.GE.KACMIN )&
2392 &          IPARMQ = 1
2393          IF( NH.GE.K22MIN )&
2394 &          IPARMQ = 2
2395      ELSE IF ( SUBNAM( 2:6 ).EQ.'HSEQR' .OR.&
2396 &      SUBNAM( 2:5 ).EQ.'LAQR' ) THEN
2397          IF( NS.GE.KACMIN )&
2398 &          IPARMQ = 1
2399          IF( NS.GE.K22MIN )&
2400 &          IPARMQ = 2

```

```

2401         END IF
2402 !
2403     ELSE
2404 !         ===== invalid value of ispec =====
2405         IPARMQ = -1
2406 !
2407     END IF
2408 !
2409 !     ===== End of IPARMQ =====
2410 !
2411     END FUNCTION IPARMQ
2412
2413 !
2414 !     =====
2415     INTEGER FUNCTION IPARAM2STAGE( ISPEC, NAME, OPTS,&
2416 &                                NI, NBI, IBI, NXI )
2417 #if defined(_OPENMP)
2418     use omp_lib
2419 #endif
2420     IMPLICIT NONE
2421 !
2422 !     -- LAPACK auxiliary routine (version 3.8.0) --
2423 !     -- LAPACK is a software package provided by Univ. of Tennessee, --
2424 !     -- Univ. of California Berkeley, Univ. of Colorado Denver and NAG Ltd.---
2425 !     June 2016
2426 !
2427 !     .. Scalar Arguments ..
2428     CHARACTER*( * )    NAME, OPTS
2429     INTEGER            ISPEC, NI, NBI, IBI, NXI
2430 !
2431 !     =====
2432 !     ..
2433 !     .. Local Scalars ..
2434     INTEGER            I, IC, IZ, KD, IB, LHOUS, LWORK, NTHREADS,&
2435 &                    FACTOPTNB, QROPTNB, LQOPTNB
2436     LOGICAL            RPREC, CPREC
2437     CHARACTER          PREC*1, ALGO*3, STAG*5, SUBNAM*12, VECT*1
2438 !     ..
2439 !     .. Intrinsic Functions ..
2440     INTRINSIC          CHAR, ICHAR, MAX
2441 !     ..
2442 !     .. External Functions ..
2443 !     INTEGER            ILAENV
2444 !     EXTERNAL           ILAENV
2445 !     ..
2446 !     .. Executable Statements ..
2447 !
2448 !     Invalid value for ISPEC
2449 !
2450     IF( (ISPEC.LT.17).OR.(ISPEC.GT.21) ) THEN
2451         IPARAM2STAGE = -1
2452         RETURN
2453     ENDIF
2454 !
2455 !     Get the number of threads
2456 !
2457     NTHREADS = 1
2458 #if defined(_OPENMP)
2459 !$OMP PARALLEL
2460     NTHREADS = OMP_GET_NUM_THREADS()
2461 !$OMP END PARALLEL

```

```

2462 #endif
2463 !     WRITE(*,*) 'IPARAM VOICI NTHREADS ISPEC ',NTHREADS, ISPEC
2464 !
2465     IF( ISPEC .NE. 19 ) THEN
2466 !
2467 !         Convert NAME to upper case if the first character is lower case.
2468 !
2469         IPARAM2STAGE = -1
2470         SUBNAM = NAME
2471         IC = ICHAR( SUBNAM( 1: 1 ) )
2472         IZ = ICHAR( 'Z' )
2473         IF( IZ.EQ.90 .OR. IZ.EQ.122 ) THEN
2474 !
2475 !             ASCII character set
2476 !
2477             IF( IC.GE.97 .AND. IC.LE.122 ) THEN
2478                 SUBNAM( 1: 1 ) = CHAR( IC-32 )
2479                 DO 100 I = 2, 12
2480                     IC = ICHAR( SUBNAM( I: I ) )
2481                     IF( IC.GE.97 .AND. IC.LE.122 )&
2482                         SUBNAM( I: I ) = CHAR( IC-32 )
2483             100     CONTINUE
2484             END IF
2485 !
2486         ELSE IF( IZ.EQ.233 .OR. IZ.EQ.169 ) THEN
2487 !
2488 !             EBCDIC character set
2489 !
2490             IF( ( IC.GE.129 .AND. IC.LE.137 ) .OR.&
2491                 & ( IC.GE.145 .AND. IC.LE.153 ) .OR.&
2492                 & ( IC.GE.162 .AND. IC.LE.169 ) ) THEN
2493                 SUBNAM( 1: 1 ) = CHAR( IC+64 )
2494                 DO 110 I = 2, 12
2495                     IC = ICHAR( SUBNAM( I: I ) )
2496                     IF( ( IC.GE.129 .AND. IC.LE.137 ) .OR.&
2497                         & ( IC.GE.145 .AND. IC.LE.153 ) .OR.&
2498                         & ( IC.GE.162 .AND. IC.LE.169 ) )SUBNAM( I:&
2499                         & I ) = CHAR( IC+64 )
2500             110     CONTINUE
2501             END IF
2502 !
2503         ELSE IF( IZ.EQ.218 .OR. IZ.EQ.250 ) THEN
2504 !
2505 !             Prime machines: ASCII+128
2506 !
2507             IF( IC.GE.225 .AND. IC.LE.250 ) THEN
2508                 SUBNAM( 1: 1 ) = CHAR( IC-32 )
2509                 DO 120 I = 2, 12
2510                     IC = ICHAR( SUBNAM( I: I ) )
2511                     IF( IC.GE.225 .AND. IC.LE.250 )&
2512                         SUBNAM( I: I ) = CHAR( IC-32 )
2513             120     CONTINUE
2514             END IF
2515         END IF
2516 !
2517         PREC = SUBNAM( 1: 1 )
2518         ALGO = SUBNAM( 4: 6 )
2519         STAG = SUBNAM( 8:12 )
2520         RPREC = PREC.EQ.'S' .OR. PREC.EQ.'D'
2521         CPREC = PREC.EQ.'C' .OR. PREC.EQ.'Z'
2522 !

```

```

2523 !      Invalid value for PRECISION
2524 !
2525      IF( .NOT.( RPREC .OR. CPREC ) ) THEN
2526          IPARAM2STAGE = -1
2527          RETURN
2528      ENDIF
2529  ENDIF
2530 !      WRITE(*,*),'RPREC,CPREC ',RPREC,CPREC,&
2531 !      &          ' ALGO ',ALGO,' STAGE ',STAG
2532 !
2533 !
2534      IF (( ISPEC .EQ. 17 ) .OR. ( ISPEC .EQ. 18 )) THEN
2535 !
2536 !      ISPEC = 17, 18: block size KD, IB
2537 !      Could be also dependent from N but for now it
2538 !      depend only on sequential or parallel
2539 !
2540          IF( NTHREADS.GT.4 ) THEN
2541              IF( CPREC ) THEN
2542                  KD = 128
2543                  IB = 32
2544              ELSE
2545                  KD = 160
2546                  IB = 40
2547              ENDIF
2548          ELSE IF( NTHREADS.GT.1 ) THEN
2549              IF( CPREC ) THEN
2550                  KD = 64
2551                  IB = 32
2552              ELSE
2553                  KD = 64
2554                  IB = 32
2555              ENDIF
2556          ELSE
2557              IF( CPREC ) THEN
2558                  KD = 16
2559                  IB = 16
2560              ELSE
2561                  KD = 32
2562                  IB = 16
2563              ENDIF
2564          ENDIF
2565          IF( ISPEC.EQ.17 ) IPARAM2STAGE = KD
2566          IF( ISPEC.EQ.18 ) IPARAM2STAGE = IB
2567 !
2568      ELSE IF ( ISPEC .EQ. 19 ) THEN
2569 !
2570 !      ISPEC = 19:
2571 !      LHOUS length of the Householder representation
2572 !      matrix (V,T) of the second stage. should be >= 1.
2573 !
2574 !      Will add the VECT OPTION HERE next release
2575          VECT = OPTS(1:1)
2576          IF( VECT.EQ.'N' ) THEN
2577              LHOUS = MAX( 1, 4*NI )
2578          ELSE
2579 !          This is not correct, it need to call the ALGO and the stage2
2580              LHOUS = MAX( 1, 4*NI ) + IBI
2581          ENDIF
2582          IF( LHOUS.GE.0 ) THEN
2583              IPARAM2STAGE = LHOUS

```

```

2584         ELSE
2585             IPARAM2STAGE = -1
2586         ENDIF
2587 !
2588     ELSE IF ( ISPEC .EQ. 20 ) THEN
2589 !
2590 !     ISPEC = 20: (21 for future use)
2591 !     LWORK length of the workspace for
2592 !     either or both stages for TRD and BRD. should be >= 1.
2593 !     TRD:
2594 !     TRD_stage 1: = LT + LW + LS1 + LS2
2595 !                 = LDT*KD + N*KD + N*MAX(KD,FACTOPTNB) + LDS2*KD
2596 !                 where LDT=LDS2=KD
2597 !                 = N*KD + N*max(KD,FACTOPTNB) + 2*KD*KD
2598 !     TRD_stage 2: = (2NB+1)*N + KD*NTHREADS
2599 !     TRD_both   : = max(stage1,stage2) + AB ( AB=(KD+1)*N )
2600 !                 = N*KD + N*max(KD+1,FACTOPTNB)
2601 !                 + max(2*KD*KD, KD*NTHREADS)
2602 !                 + (KD+1)*N
2603     LWORK          = -1
2604     SUBNAM(1:1)    = PREC
2605     SUBNAM(2:6)    = 'GEQRF'
2606     QROPTNB       = ILAENV( 1, SUBNAM, ' ', NI, NBI, -1, -1 )
2607     SUBNAM(2:6)    = 'GELQF'
2608     LQOPTNB       = ILAENV( 1, SUBNAM, ' ', NBI, NI, -1, -1 )
2609 !     Could be QR or LQ for TRD and the max for BRD
2610     FACTOPTNB     = MAX(QROPTNB, LQOPTNB)
2611     IF( ALGO.EQ.'TRD' ) THEN
2612         IF( STAG.EQ.'2STAG' ) THEN
2613             LWORK = NI*NBI + NI*MAX(NBI+1,FACTOPTNB)&
2614 &             + MAX(2*NBI*NBI, NBI*NTHREADS) &
2615 &             + (NBI+1)*NI
2616         ELSE IF( (STAG.EQ.'HE2HB').OR.(STAG.EQ.'SY2SB' ) ) THEN
2617             LWORK = NI*NBI + NI*MAX(NBI,FACTOPTNB) + 2*NBI*NBI
2618         ELSE IF( (STAG.EQ.'HB2ST').OR.(STAG.EQ.'SB2ST' ) ) THEN
2619             LWORK = (2*NBI+1)*NI + NBI*NTHREADS
2620         ENDIF
2621     ELSE IF( ALGO.EQ.'BRD' ) THEN
2622         IF( STAG.EQ.'2STAG' ) THEN
2623             LWORK = 2*NI*NBI + NI*MAX(NBI+1,FACTOPTNB) &
2624 &             + MAX(2*NBI*NBI, NBI*NTHREADS) &
2625 &             + (NBI+1)*NI
2626         ELSE IF( STAG.EQ.'GE2GB' ) THEN
2627             LWORK = NI*NBI + NI*MAX(NBI,FACTOPTNB) + 2*NBI*NBI
2628         ELSE IF( STAG.EQ.'GB2BD' ) THEN
2629             LWORK = (3*NBI+1)*NI + NBI*NTHREADS
2630         ENDIF
2631     ENDIF
2632     LWORK = MAX ( 1, LWORK )
2633
2634     IF( LWORK.GT.0 ) THEN
2635         IPARAM2STAGE = LWORK
2636     ELSE
2637         IPARAM2STAGE = -1
2638     ENDIF
2639 !
2640     ELSE IF ( ISPEC .EQ. 21 ) THEN
2641 !
2642 !     ISPEC = 21 for future use
2643     IPARAM2STAGE = NXI
2644     ENDIF

```

```

2645 !
2646 !      ==== End of IPARAM2STAGE ====
2647 !
2648 !      END FUNCTION IPARAM2STAGE
2649 !
2650 !
2651 !      =====
2652 !      SUBROUTINE DSWAP(N,DX,INCX,DY,INCY)
2653 !
2654 !      -- Reference BLAS level1 routine (version 3.8.0) --
2655 !      -- Reference BLAS is a software package provided by Univ. of Tennessee,      --
2656 !      -- Univ. of California Berkeley, Univ. of Colorado Denver and NAG Ltd.--
2657 !      November 2017
2658 !
2659 !      .. Scalar Arguments ..
2660 !      INTEGER INCX,INCY,N
2661 !      ..
2662 !      .. Array Arguments ..
2663 !      DOUBLE PRECISION DX(*),DY(*)
2664 !      ..
2665 !
2666 !      =====
2667 !
2668 !      .. Local Scalars ..
2669 !      DOUBLE PRECISION DTEMP
2670 !      INTEGER I,IX,IY,M,MP1
2671 !      ..
2672 !      .. Intrinsic Functions ..
2673 !      INTRINSIC MOD
2674 !      ..
2675 !      IF (N.LE.0) RETURN
2676 !      IF (INCX.EQ.1 .AND. INCY.EQ.1) THEN
2677 !
2678 !          code for both increments equal to 1
2679 !
2680 !
2681 !          clean-up loop
2682 !
2683 !          M = MOD(N,3)
2684 !          IF (M.NE.0) THEN
2685 !              DO I = 1,M
2686 !                  DTEMP = DX(I)
2687 !                  DX(I) = DY(I)
2688 !                  DY(I) = DTEMP
2689 !              END DO
2690 !              IF (N.LT.3) RETURN
2691 !          END IF
2692 !          MP1 = M + 1
2693 !          DO I = MP1,N,3
2694 !              DTEMP = DX(I)
2695 !              DX(I) = DY(I)
2696 !              DY(I) = DTEMP
2697 !              DTEMP = DX(I+1)
2698 !              DX(I+1) = DY(I+1)
2699 !              DY(I+1) = DTEMP
2700 !              DTEMP = DX(I+2)
2701 !              DX(I+2) = DY(I+2)
2702 !              DY(I+2) = DTEMP
2703 !          END DO
2704 !      ELSE
2705 !

```

```

2706 !      code for unequal increments or equal increments not equal
2707 !          to 1
2708 !
2709 !      IX = 1
2710 !      IY = 1
2711 !      IF (INCX.LT.0) IX = (-N+1)*INCX + 1
2712 !      IF (INCY.LT.0) IY = (-N+1)*INCY + 1
2713 !      DO I = 1,N
2714 !          DTEMP = DX(IX)
2715 !          DX(IX) = DY(IY)
2716 !          DY(IY) = DTEMP
2717 !          IX = IX + INCX
2718 !          IY = IY + INCY
2719 !      END DO
2720 !  END IF
2721 !  RETURN
2722 !  END SUBROUTINE DSWAP
2723
2724 !
2725 !  =====
2726 !  SUBROUTINE DGEMV(TRANS,M,N,ALPHA,A,LDA,X,INCX,BETA,Y,INCY)
2727 !
2728 !  -- Reference BLAS level2 routine (version 3.7.0) --
2729 !  -- Reference BLAS is a software package provided by Univ. of Tennessee,    --
2730 !  -- Univ. of California Berkeley, Univ. of Colorado Denver and NAG Ltd.--
2731 !  December 2016
2732 !
2733 !  .. Scalar Arguments ..
2734 !  DOUBLE PRECISION ALPHA,BETA
2735 !  INTEGER INCX,INCY,LDA,M,N
2736 !  CHARACTER TRANS
2737 !  ..
2738 !  .. Array Arguments ..
2739 !  DOUBLE PRECISION A(LDA,*),X(*),Y(*)
2740 !  ..
2741 !
2742 !  =====
2743 !
2744 !  .. Parameters ..
2745 !  DOUBLE PRECISION ONE,ZERO
2746 !  PARAMETER (ONE=1.0D+0,ZERO=0.0D+0)
2747 !  ..
2748 !  .. Local Scalars ..
2749 !  DOUBLE PRECISION TEMP
2750 !  INTEGER I,INFO,IX,IY,J,JX,JY,KX,KY,LENX,LENY
2751 !  ..
2752 !  .. External Functions ..
2753 !  LOGICAL LSAME
2754 !  EXTERNAL LSAME
2755 !  ..
2756 !  .. External Subroutines ..
2757 !  EXTERNAL XERBLA
2758 !  ..
2759 !  .. Intrinsic Functions ..
2760 !  INTRINSIC MAX
2761 !  ..
2762 !
2763 !  Test the input parameters.
2764 !
2765 !  INFO = 0
2766 !  IF (.NOT.LSAME(TRANS,'N') .AND. .NOT.LSAME(TRANS,'T') .AND.&

```



```

2767         .NOT.LSAME(TRANS,'C')) THEN
2768         INFO = 1
2769     ELSE IF (M.LT.0) THEN
2770         INFO = 2
2771     ELSE IF (N.LT.0) THEN
2772         INFO = 3
2773     ELSE IF (LDA.LT.MAX(1,M)) THEN
2774         INFO = 6
2775     ELSE IF (INCX.EQ.0) THEN
2776         INFO = 8
2777     ELSE IF (INCY.EQ.0) THEN
2778         INFO = 11
2779     END IF
2780     IF (INFO.NE.0) THEN
2781         CALL XERBLA('DGEMV ',INFO)
2782         RETURN
2783     END IF
2784 !
2785 !     Quick return if possible.
2786 !
2787     IF ((M.EQ.0) .OR. (N.EQ.0) .OR.&
2788         ((ALPHA.EQ.ZERO).AND. (BETA.EQ.ONE))) RETURN
2789 !
2790 !     Set LENX and LENY, the lengths of the vectors x and y, and set
2791 !     up the start points in X and Y.
2792 !
2793     IF (LSAME(TRANS,'N')) THEN
2794         LENX = N
2795         LENY = M
2796     ELSE
2797         LENX = M
2798         LENY = N
2799     END IF
2800     IF (INCX.GT.0) THEN
2801         KX = 1
2802     ELSE
2803         KX = 1 - (LENX-1)*INCX
2804     END IF
2805     IF (INCY.GT.0) THEN
2806         KY = 1
2807     ELSE
2808         KY = 1 - (LENY-1)*INCY
2809     END IF
2810 !
2811 !     Start the operations. In this version the elements of A are
2812 !     accessed sequentially with one pass through A.
2813 !
2814 !     First form y := beta*y.
2815 !
2816     IF (BETA.NE.ONE) THEN
2817         IF (INCY.EQ.1) THEN
2818             IF (BETA.EQ.ZERO) THEN
2819                 DO 10 I = 1,LENY
2820                     Y(I) = ZERO
2821             10 CONTINUE
2822             ELSE
2823                 DO 20 I = 1,LENY
2824                     Y(I) = BETA*Y(I)
2825             20 CONTINUE
2826             END IF
2827         ELSE

```

```

2828         IY = KY
2829         IF (BETA.EQ.ZERO) THEN
2830             DO 30 I = 1,LENY
2831                 Y(IY) = ZERO
2832                 IY = IY + INCY
2833     30         CONTINUE
2834             ELSE
2835                 DO 40 I = 1,LENY
2836                     Y(IY) = BETA*Y(IY)
2837                     IY = IY + INCY
2838     40         CONTINUE
2839             END IF
2840         END IF
2841     END IF
2842     IF (ALPHA.EQ.ZERO) RETURN
2843     IF (LSAME(TRANS,'N')) THEN
2844 !
2845 !         Form  $y := \alpha * A * x + y.$ 
2846 !
2847         JX = KX
2848         IF (INCY.EQ.1) THEN
2849             DO 60 J = 1,N
2850                 TEMP = ALPHA*X(JX)
2851                 DO 50 I = 1,M
2852                     Y(I) = Y(I) + TEMP*A(I,J)
2853     50                 CONTINUE
2854                 JX = JX + INCX
2855     60                 CONTINUE
2856             ELSE
2857                 DO 80 J = 1,N
2858                     TEMP = ALPHA*X(JX)
2859                     IY = KY
2860                     DO 70 I = 1,M
2861                         Y(IY) = Y(IY) + TEMP*A(I,J)
2862                         IY = IY + INCY
2863     70                 CONTINUE
2864                 JX = JX + INCX
2865     80                 CONTINUE
2866             END IF
2867         ELSE
2868 !
2869 !         Form  $y := \alpha * A ** T * x + y.$ 
2870 !
2871         JY = KY
2872         IF (INCX.EQ.1) THEN
2873             DO 100 J = 1,N
2874                 TEMP = ZERO
2875                 DO 90 I = 1,M
2876                     TEMP = TEMP + A(I,J)*X(I)
2877     90                 CONTINUE
2878                 Y(JY) = Y(JY) + ALPHA*TEMP
2879                 JY = JY + INCY
2880     100                CONTINUE
2881             ELSE
2882                 DO 120 J = 1,N
2883                     TEMP = ZERO
2884                     IX = KX
2885                     DO 110 I = 1,M
2886                         TEMP = TEMP + A(I,J)*X(IX)
2887                         IX = IX + INCX
2888     110                CONTINUE

```

```

2889             Y(JY) = Y(JY) + ALPHA*TEMP
2890             JY = JY + INCY
2891 120          CONTINUE
2892             END IF
2893          END IF
2894 !
2895          RETURN
2896 !
2897 !      End of DGEMV .
2898 !
2899          END SUBROUTINE DGEMV
2900 !
2901 ! =====
2902          SUBROUTINE DTRTRI( UPLO, DIAG, N, A, LDA, INFO )
2903 !
2904 ! -- LAPACK computational routine (version 3.7.0) --
2905 ! -- LAPACK is a software package provided by Univ. of Tennessee, --
2906 ! -- Univ. of California Berkeley, Univ. of Colorado Denver and NAG Ltd.--
2907 !      December 2016
2908 !
2909 ! .. Scalar Arguments ..
2910 CHARACTER          DIAG, UPLO
2911 INTEGER            INFO, LDA, N
2912 ! ..
2913 ! .. Array Arguments ..
2914 DOUBLE PRECISION  A( LDA, * )
2915 ! ..
2916 ! =====
2917 !
2918 ! .. Parameters ..
2919 !
2920 DOUBLE PRECISION  ONE, ZERO
2921 PARAMETER        ( ONE = 1.0D+0, ZERO = 0.0D+0 )
2922 ! ..
2923 ! .. Local Scalars ..
2924 LOGICAL          NOUNIT, UPPER
2925 INTEGER          J, JB, NB, NN
2926 ! ..
2927 ! .. External Functions ..
2928 LOGICAL          LSAME
2929 INTEGER          ILAENV
2930 EXTERNAL         LSAME, ILAENV
2931 ! ..
2932 ! .. External Subroutines ..
2933 EXTERNAL         DTRMM, DTRSM, DTRTI2, XERBLA
2934 ! ..
2935 ! .. Intrinsic Functions ..
2936 INTRINSIC        MAX, MIN
2937 ! ..
2938 ! .. Executable Statements ..
2939 !
2940 !      Test the input parameters.
2941 !
2942          INFO = 0
2943          UPPER = LSAME( UPLO, 'U' )
2944          NOUNIT = LSAME( DIAG, 'N' )
2945          IF( .NOT.UPPER .AND. .NOT.LSAME( UPLO, 'L' ) ) THEN
2946              INFO = -1
2947          ELSE IF( .NOT.NOUNIT .AND. .NOT.LSAME( DIAG, 'U' ) ) THEN
2948              INFO = -2
2949          ELSE IF( N.LT.0 ) THEN

```

```

2950         INFO = -3
2951     ELSE IF( LDA.LT.MAX( 1, N ) ) THEN
2952         INFO = -5
2953     END IF
2954     IF( INFO.NE.0 ) THEN
2955         CALL XERBLA( 'DTRTRI', -INFO )
2956         RETURN
2957     END IF
2958 !
2959 !     Quick return if possible
2960 !
2961     IF( N.EQ.0 )&
2962     &     RETURN
2963 !
2964 !     Check for singularity if non-unit.
2965 !
2966     IF( NOUNIT ) THEN
2967         DO 10 INFO = 1, N
2968             IF( A( INFO, INFO ).EQ.ZERO )&
2969             &     RETURN
2970     10 CONTINUE
2971     INFO = 0
2972     END IF
2973 !
2974 !     Determine the block size for this environment.
2975 !
2976     NB = ILAENV( 1, 'DTRTRI', UPLO // DIAG, N, -1, -1, -1 )
2977     IF( NB.LE.1 .OR. NB.GE.N ) THEN
2978 !
2979 !         Use unblocked code
2980 !
2981         CALL DTRTI2( UPLO, DIAG, N, A, LDA, INFO )
2982     ELSE
2983 !
2984 !         Use blocked code
2985 !
2986         IF( UPPER ) THEN
2987 !
2988 !             Compute inverse of upper triangular matrix
2989 !
2990             DO 20 J = 1, N, NB
2991                 JB = MIN( NB, N-J+1 )
2992 !
2993 !                 Compute rows 1:j-1 of current block column
2994 !
2995                 CALL DTRMM( 'Left', 'Upper', 'No transpose', DIAG, J-1,&
2996             &     JB, ONE, A, LDA, A( 1, J ), LDA )
2997                 CALL DTRSM( 'Right', 'Upper', 'No transpose', DIAG, J-1,&
2998             &     JB, -ONE, A( J, J ), LDA, A( 1, J ), LDA )
2999 !
3000 !                 Compute inverse of current diagonal block
3001 !
3002                 CALL DTRTI2( 'Upper', DIAG, JB, A( J, J ), LDA, INFO )
3003     20 CONTINUE
3004     ELSE
3005 !
3006 !         Compute inverse of lower triangular matrix
3007 !
3008         NN = ( ( N-1 ) / NB ) * NB + 1
3009         DO 30 J = NN, 1, -NB
3010             JB = MIN( NB, N-J+1 )

```

```

3011         IF( J+JB.LE.N ) THEN
3012 !
3013 !           Compute rows j+jb:n of current block column
3014 !
3015         CALL DTRMM( 'Left', 'Lower', 'No transpose', DIAG, &
3016 &                N-J-JB+1, JB, ONE, A( J+JB, J+JB ), LDA,&
3017 &                A( J+JB, J ), LDA )
3018         CALL DTRSM( 'Right', 'Lower', 'No transpose', DIAG, &
3019 &                N-J-JB+1, JB, -ONE, A( J, J ), LDA, &
3020 &                A( J+JB, J ), LDA )
3021         END IF
3022 !
3023 !           Compute inverse of current diagonal block
3024 !
3025         CALL DTRTI2( 'Lower', DIAG, JB, A( J, J ), LDA, INFO )
3026 30 CONTINUE
3027     END IF
3028 END IF
3029 !
3030 RETURN
3031 !
3032 ! End of DTRTRI
3033 !
3034 END SUBROUTINE DTRTRI
3035
3036 !
3037 ! =====
3038 SUBROUTINE DTRMM(SIDE,UPLO,TRANSA,DIAG,M,N,ALPHA,A,LDA,B,LDB)
3039 !
3040 ! -- Reference BLAS level3 routine (version 3.7.0) --
3041 ! -- Reference BLAS is a software package provided by Univ. of Tennessee, --
3042 ! -- Univ. of California Berkeley, Univ. of Colorado Denver and NAG Ltd.---
3043 ! December 2016
3044 !
3045 ! .. Scalar Arguments ..
3046 DOUBLE PRECISION ALPHA
3047 INTEGER LDA,LDB,M,N
3048 CHARACTER DIAG,SIDE,TRANSA,UPLO
3049 ! ..
3050 ! .. Array Arguments ..
3051 DOUBLE PRECISION A(LDA,*),B(LDB,*)
3052 ! ..
3053 !
3054 ! =====
3055 !
3056 ! .. External Functions ..
3057 LOGICAL LSAME
3058 EXTERNAL LSAME
3059 ! ..
3060 ! .. External Subroutines ..
3061 EXTERNAL XERBLA
3062 ! ..
3063 ! .. Intrinsic Functions ..
3064 INTRINSIC MAX
3065 ! ..
3066 ! .. Local Scalars ..
3067 DOUBLE PRECISION TEMP
3068 INTEGER I,INFO,J,K,NROWA
3069 LOGICAL LSIDE,NOUNIT,UPPER
3070 ! ..
3071 ! .. Parameters ..

```

```

3072     DOUBLE PRECISION ONE,ZERO
3073     PARAMETER (ONE=1.0D+0,ZERO=0.0D+0)
3074 !     ..
3075 !
3076 !     Test the input parameters.
3077 !
3078     LSIDE = LSAME(SIDE,'L')
3079     IF (LSIDE) THEN
3080         NROWA = M
3081     ELSE
3082         NROWA = N
3083     END IF
3084     NOUNIT = LSAME(DIAG,'N')
3085     UPPER = LSAME(UPLO,'U')
3086 !
3087     INFO = 0
3088     IF ((.NOT.LSIDE) .AND. (.NOT.LSAME(SIDE,'R'))) THEN
3089         INFO = 1
3090     ELSE IF ((.NOT.UPPER) .AND. (.NOT.LSAME(UPLO,'L'))) THEN
3091         INFO = 2
3092     ELSE IF ((.NOT.LSAME(TRANSA,'N')) .AND.&
3093 &           (.NOT.LSAME(TRANSA,'T')) .AND.&
3094 &           (.NOT.LSAME(TRANSA,'C'))) THEN
3095         INFO = 3
3096     ELSE IF ((.NOT.LSAME(DIAG,'U')) .AND. (.NOT.LSAME(DIAG,'N'))) THEN
3097         INFO = 4
3098     ELSE IF (M.LT.0) THEN
3099         INFO = 5
3100     ELSE IF (N.LT.0) THEN
3101         INFO = 6
3102     ELSE IF (LDA.LT.MAX(1,NROWA)) THEN
3103         INFO = 9
3104     ELSE IF (LDB.LT.MAX(1,M)) THEN
3105         INFO = 11
3106     END IF
3107     IF (INFO.NE.0) THEN
3108         CALL XERBLA('DTRMM ',INFO)
3109         RETURN
3110     END IF
3111 !
3112 !     Quick return if possible.
3113 !
3114     IF (M.EQ.0 .OR. N.EQ.0) RETURN
3115 !
3116 !     And when alpha.eq.zero.
3117 !
3118     IF (ALPHA.EQ.ZERO) THEN
3119         DO 20 J = 1,N
3120             DO 10 I = 1,M
3121                 B(I,J) = ZERO
3122             10 CONTINUE
3123         20 CONTINUE
3124         RETURN
3125     END IF
3126 !
3127 !     Start the operations.
3128 !
3129     IF (LSIDE) THEN
3130         IF (LSAME(TRANSA,'N')) THEN
3131 !
3132 !             Form B := alpha*A*B.

```

```

3133 !
3134         IF (UPPER) THEN
3135             DO 50 J = 1,N
3136                 DO 40 K = 1,M
3137                     IF (B(K,J).NE.ZERO) THEN
3138                         TEMP = ALPHA*B(K,J)
3139                         DO 30 I = 1,K - 1
3140                             B(I,J) = B(I,J) + TEMP*A(I,K)
3141             30         CONTINUE
3142                         IF (NOUNIT) TEMP = TEMP*A(K,K)
3143                         B(K,J) = TEMP
3144                     END IF
3145             40         CONTINUE
3146             50         CONTINUE
3147         ELSE
3148             DO 80 J = 1,N
3149                 DO 70 K = M,1,-1
3150                     IF (B(K,J).NE.ZERO) THEN
3151                         TEMP = ALPHA*B(K,J)
3152                         B(K,J) = TEMP
3153                         IF (NOUNIT) B(K,J) = B(K,J)*A(K,K)
3154                         DO 60 I = K + 1,M
3155                             B(I,J) = B(I,J) + TEMP*A(I,K)
3156             60         CONTINUE
3157                         END IF
3158             70         CONTINUE
3159             80         CONTINUE
3160         END IF
3161     ELSE
3162 !
3163 !         Form B := alpha*A**T*B.
3164 !
3165         IF (UPPER) THEN
3166             DO 110 J = 1,N
3167                 DO 100 I = M,1,-1
3168                     TEMP = B(I,J)
3169                     IF (NOUNIT) TEMP = TEMP*A(I,I)
3170                     DO 90 K = 1,I - 1
3171                         TEMP = TEMP + A(K,I)*B(K,J)
3172             90         CONTINUE
3173                     B(I,J) = ALPHA*TEMP
3174             100        CONTINUE
3175             110        CONTINUE
3176         ELSE
3177             DO 140 J = 1,N
3178                 DO 130 I = 1,M
3179                     TEMP = B(I,J)
3180                     IF (NOUNIT) TEMP = TEMP*A(I,I)
3181                     DO 120 K = I + 1,M
3182                         TEMP = TEMP + A(K,I)*B(K,J)
3183             120        CONTINUE
3184                     B(I,J) = ALPHA*TEMP
3185             130        CONTINUE
3186             140        CONTINUE
3187         END IF
3188     END IF
3189 ELSE
3190     IF (LSAME(TRANSA, 'N')) THEN
3191 !
3192 !         Form B := alpha*B*A.
3193 !

```

```

3194         IF (UPPER) THEN
3195             DO 180 J = N,1,-1
3196                 TEMP = ALPHA
3197                 IF (NOUNIT) TEMP = TEMP*A(J,J)
3198                 DO 150 I = 1,M
3199                     B(I,J) = TEMP*B(I,J)
3200             150 CONTINUE
3201             DO 170 K = 1,J - 1
3202                 IF (A(K,J).NE.ZERO) THEN
3203                     TEMP = ALPHA*A(K,J)
3204                     DO 160 I = 1,M
3205                         B(I,J) = B(I,J) + TEMP*B(I,K)
3206             160 CONTINUE
3207                 END IF
3208             170 CONTINUE
3209             180 CONTINUE
3210         ELSE
3211             DO 220 J = 1,N
3212                 TEMP = ALPHA
3213                 IF (NOUNIT) TEMP = TEMP*A(J,J)
3214                 DO 190 I = 1,M
3215                     B(I,J) = TEMP*B(I,J)
3216             190 CONTINUE
3217             DO 210 K = J + 1,N
3218                 IF (A(K,J).NE.ZERO) THEN
3219                     TEMP = ALPHA*A(K,J)
3220                     DO 200 I = 1,M
3221                         B(I,J) = B(I,J) + TEMP*B(I,K)
3222             200 CONTINUE
3223                 END IF
3224             210 CONTINUE
3225             220 CONTINUE
3226         END IF
3227     ELSE
3228 !
3229 ! Form B := alpha*B*A**T.
3230 !
3231         IF (UPPER) THEN
3232             DO 260 K = 1,N
3233                 DO 240 J = 1,K - 1
3234                     IF (A(J,K).NE.ZERO) THEN
3235                         TEMP = ALPHA*A(J,K)
3236                         DO 230 I = 1,M
3237                             B(I,J) = B(I,J) + TEMP*B(I,K)
3238             230 CONTINUE
3239                     END IF
3240                 240 CONTINUE
3241                 TEMP = ALPHA
3242                 IF (NOUNIT) TEMP = TEMP*A(K,K)
3243                 IF (TEMP.NE.ONE) THEN
3244                     DO 250 I = 1,M
3245                         B(I,K) = TEMP*B(I,K)
3246             250 CONTINUE
3247                 END IF
3248             260 CONTINUE
3249         ELSE
3250             DO 300 K = N,1,-1
3251                 DO 280 J = K + 1,N
3252                     IF (A(J,K).NE.ZERO) THEN
3253                         TEMP = ALPHA*A(J,K)
3254             270 DO I = 1,M

```



```

3255             B(I,J) = B(I,J) + TEMP*B(I,K)
3256 270             CONTINUE
3257             END IF
3258 280             CONTINUE
3259             TEMP = ALPHA
3260             IF (NUNIT) TEMP = TEMP*A(K,K)
3261             IF (TEMP.NE.ONE) THEN
3262                 DO 290 I = 1,M
3263                     B(I,K) = TEMP*B(I,K)
3264 290             CONTINUE
3265             END IF
3266 300             CONTINUE
3267             END IF
3268             END IF
3269             END IF
3270 !
3271             RETURN
3272 !
3273 !     End of DTRMM .
3274 !
3275             END SUBROUTINE DTRMM
3276
3277 !
3278 ! =====
3279             SUBROUTINE DTRTI2( UPLO, DIAG, N, A, LDA, INFO )
3280 !
3281 ! -- LAPACK computational routine (version 3.7.0) --
3282 ! -- LAPACK is a software package provided by Univ. of Tennessee, --
3283 ! -- Univ. of California Berkeley, Univ. of Colorado Denver and NAG Ltd.--
3284 ! December 2016
3285 !
3286 ! .. Scalar Arguments ..
3287 CHARACTER          DIAG, UPLO
3288 INTEGER            INFO, LDA, N
3289 !
3290 ! .. Array Arguments ..
3291 DOUBLE PRECISION  A( LDA, * )
3292 !
3293 !
3294 ! =====
3295 !
3296 ! .. Parameters ..
3297 DOUBLE PRECISION  ONE
3298 PARAMETER         ( ONE = 1.0D+0 )
3299 !
3300 ! .. Local Scalars ..
3301 LOGICAL           NUNIT, UPPER
3302 INTEGER           J
3303 DOUBLE PRECISION  AJJ
3304 !
3305 ! .. External Functions ..
3306 LOGICAL          LSAME
3307 EXTERNAL         LSAME
3308 !
3309 ! .. External Subroutines ..
3310 EXTERNAL         DSCAL, DTRMV, XERBLA
3311 !
3312 ! .. Intrinsic Functions ..
3313 INTRINSIC        MAX
3314 !
3315 ! .. Executable Statements ..

```

```

3316 !
3317 !   Test the input parameters.
3318 !
3319     INFO = 0
3320     UPPER = LSAME( UPLO, 'U' )
3321     NOUNIT = LSAME( DIAG, 'N' )
3322     IF( .NOT.UPPER .AND. .NOT.LSAME( UPLO, 'L' ) ) THEN
3323         INFO = -1
3324     ELSE IF( .NOT.NOUNIT .AND. .NOT.LSAME( DIAG, 'U' ) ) THEN
3325         INFO = -2
3326     ELSE IF( N.LT.0 ) THEN
3327         INFO = -3
3328     ELSE IF( LDA.LT.MAX( 1, N ) ) THEN
3329         INFO = -5
3330     END IF
3331     IF( INFO.NE.0 ) THEN
3332         CALL XERBLA( 'DTRTI2', -INFO )
3333         RETURN
3334     END IF
3335 !
3336     IF( UPPER ) THEN
3337 !
3338 !       Compute inverse of upper triangular matrix.
3339 !
3340         DO 10 J = 1, N
3341             IF( NOUNIT ) THEN
3342                 A( J, J ) = ONE / A( J, J )
3343                 AJJ = -A( J, J )
3344             ELSE
3345                 AJJ = -ONE
3346             END IF
3347 !
3348 !       Compute elements 1:j-1 of j-th column.
3349 !
3350             CALL DTRMV( 'Upper', 'No transpose', DIAG, J-1, A, LDA,&
3351 &                 A( 1, J ), 1 )
3352             CALL DSCAL( J-1, AJJ, A( 1, J ), 1 )
3353 10         CONTINUE
3354         ELSE
3355 !
3356 !       Compute inverse of lower triangular matrix.
3357 !
3358         DO 20 J = N, 1, -1
3359             IF( NOUNIT ) THEN
3360                 A( J, J ) = ONE / A( J, J )
3361                 AJJ = -A( J, J )
3362             ELSE
3363                 AJJ = -ONE
3364             END IF
3365             IF( J.LT.N ) THEN
3366 !
3367 !       Compute elements j+1:n of j-th column.
3368 !
3369                 CALL DTRMV( 'Lower', 'No transpose', DIAG, N-J, &
3370 &                 A( J+1, J+1 ), LDA, A( J+1, J ), 1 )
3371                 CALL DSCAL( N-J, AJJ, A( J+1, J ), 1 )
3372             END IF
3373 20         CONTINUE
3374         END IF
3375 !
3376     RETURN

```

```

3377 !
3378 !   End of DTRTI2
3379 !
3380 !   END SUBROUTINE DTRTI2
3381 !
3382 ! =====
3383 !   SUBROUTINE DTRMV(UPLO,TRANS,DIAG,N,A,LDA,X,INCX)
3384 !
3385 ! -- Reference BLAS level2 routine (version 3.7.0) --
3386 ! -- Reference BLAS is a software package provided by Univ. of Tennessee, --
3387 ! -- Univ. of California Berkeley, Univ. of Colorado Denver and NAG Ltd.---
3388 !   December 2016
3389 !
3390 !   .. Scalar Arguments ..
3391 !   INTEGER INCX,LDA,N
3392 !   CHARACTER DIAG,TRANS,UPLO
3393 !   ..
3394 !   .. Array Arguments ..
3395 !   DOUBLE PRECISION A(LDA,*),X(*)
3396 !   ..
3397 !
3398 ! =====
3399 !
3400 !   .. Parameters ..
3401 !   DOUBLE PRECISION ZERO
3402 !   PARAMETER (ZERO=0.0D+0)
3403 !   ..
3404 !   .. Local Scalars ..
3405 !   DOUBLE PRECISION TEMP
3406 !   INTEGER I,INFO,IX,J,JX,KX
3407 !   LOGICAL NOUNIT
3408 !   ..
3409 !   .. External Functions ..
3410 !   LOGICAL LSAME
3411 !   EXTERNAL LSAME
3412 !   ..
3413 !   .. External Subroutines ..
3414 !   EXTERNAL XERBLA
3415 !   ..
3416 !   .. Intrinsic Functions ..
3417 !   INTRINSIC MAX
3418 !   ..
3419 !
3420 !   Test the input parameters.
3421 !
3422 !   INFO = 0
3423 !   IF (.NOT.LSAME(UPLO,'U') .AND. .NOT.LSAME(UPLO,'L')) THEN
3424 !       INFO = 1
3425 !   ELSE IF (.NOT.LSAME(TRANS,'N') .AND. .NOT.LSAME(TRANS,'T') .AND. &
3426 !   & .NOT.LSAME(TRANS,'C')) THEN
3427 !       INFO = 2
3428 !   ELSE IF (.NOT.LSAME(DIAG,'U') .AND. .NOT.LSAME(DIAG,'N')) THEN
3429 !       INFO = 3
3430 !   ELSE IF (N.LT.0) THEN
3431 !       INFO = 4
3432 !   ELSE IF (LDA.LT.MAX(1,N)) THEN
3433 !       INFO = 6
3434 !   ELSE IF (INCX.EQ.0) THEN
3435 !       INFO = 8
3436 !   END IF
3437 !   IF (INFO.NE.0) THEN

```

```

3438         CALL XERBLA('DTRMV ',INFO)
3439         RETURN
3440     END IF
3441 !
3442 ! Quick return if possible.
3443 !
3444     IF (N.EQ.0) RETURN
3445 !
3446     NOUNIT = LSAME(DIAG,'N')
3447 !
3448 ! Set up the start point in X if the increment is not unity. This
3449 ! will be ( N - 1 )*INCX too small for descending loops.
3450 !
3451     IF (INCX.LE.0) THEN
3452         KX = 1 - (N-1)*INCX
3453     ELSE IF (INCX.NE.1) THEN
3454         KX = 1
3455     END IF
3456 !
3457 ! Start the operations. In this version the elements of A are
3458 ! accessed sequentially with one pass through A.
3459 !
3460     IF (LSAME(TRANS,'N')) THEN
3461 !
3462 ! Form x := A*x.
3463 !
3464         IF (LSAME(UPLO,'U')) THEN
3465             IF (INCX.EQ.1) THEN
3466                 DO 20 J = 1,N
3467                     IF (X(J).NE.ZERO) THEN
3468                         TEMP = X(J)
3469                         DO 10 I = 1,J - 1
3470                             X(I) = X(I) + TEMP*A(I,J)
3471 10                     CONTINUE
3472                     IF (NOUNIT) X(J) = X(J)*A(J,J)
3473                     END IF
3474 20                 CONTINUE
3475             ELSE
3476                 JX = KX
3477                 DO 40 J = 1,N
3478                     IF (X(JX).NE.ZERO) THEN
3479                         TEMP = X(JX)
3480                         IX = KX
3481                         DO 30 I = 1,J - 1
3482                             X(IX) = X(IX) + TEMP*A(I,J)
3483                             IX = IX + INCX
3484 30                     CONTINUE
3485                     IF (NOUNIT) X(JX) = X(JX)*A(J,J)
3486                     END IF
3487                     JX = JX + INCX
3488 40                 CONTINUE
3489             END IF
3490         ELSE
3491             IF (INCX.EQ.1) THEN
3492                 DO 60 J = N,1,-1
3493                     IF (X(J).NE.ZERO) THEN
3494                         TEMP = X(J)
3495                         DO 50 I = N,J + 1,-1
3496                             X(I) = X(I) + TEMP*A(I,J)
3497 50                     CONTINUE
3498                     IF (NOUNIT) X(J) = X(J)*A(J,J)

```

```

3499         END IF
3500 60      CONTINUE
3501      ELSE
3502         KX = KX + (N-1)*INCX
3503         JX = KX
3504         DO 80 J = N,1,-1
3505             IF (X(JX).NE.ZERO) THEN
3506                 TEMP = X(JX)
3507                 IX = KX
3508                 DO 70 I = N,J + 1,-1
3509                     X(IX) = X(IX) + TEMP*A(I,J)
3510                     IX = IX - INCX
3511 70      CONTINUE
3512             IF (NOUNIT) X(JX) = X(JX)*A(J,J)
3513         END IF
3514         JX = JX - INCX
3515 80      CONTINUE
3516     END IF
3517 END IF
3518 ELSE
3519 !
3520 ! Form x := A**T*x.
3521 !
3522     IF (LSAME(UPLO,'U')) THEN
3523         IF (INCX.EQ.1) THEN
3524             DO 100 J = N,1,-1
3525                 TEMP = X(J)
3526                 IF (NOUNIT) TEMP = TEMP*A(J,J)
3527                 DO 90 I = J - 1,1,-1
3528                     TEMP = TEMP + A(I,J)*X(I)
3529 90      CONTINUE
3530                 X(J) = TEMP
3531 100     CONTINUE
3532         ELSE
3533             JX = KX + (N-1)*INCX
3534             DO 120 J = N,1,-1
3535                 TEMP = X(JX)
3536                 IX = JX
3537                 IF (NOUNIT) TEMP = TEMP*A(J,J)
3538                 DO 110 I = J - 1,1,-1
3539                     IX = IX - INCX
3540                     TEMP = TEMP + A(I,J)*X(IX)
3541 110     CONTINUE
3542                 X(JX) = TEMP
3543                 JX = JX - INCX
3544 120     CONTINUE
3545             END IF
3546         ELSE
3547             IF (INCX.EQ.1) THEN
3548                 DO 140 J = 1,N
3549                     TEMP = X(J)
3550                     IF (NOUNIT) TEMP = TEMP*A(J,J)
3551                     DO 130 I = J + 1,N
3552                         TEMP = TEMP + A(I,J)*X(I)
3553 130     CONTINUE
3554                     X(J) = TEMP
3555 140     CONTINUE
3556             ELSE
3557                 JX = KX
3558                 DO 160 J = 1,N
3559                     TEMP = X(JX)

```

```

3560             IX = JX
3561             IF (NOUNIT) TEMP = TEMP*A(J,J)
3562             DO 150 I = J + 1,N
3563                 IX = IX + INCX
3564                 TEMP = TEMP + A(I,J)*X(IX)
3565 150             CONTINUE
3566             X(JX) = TEMP
3567             JX = JX + INCX
3568 160             CONTINUE
3569             END IF
3570             END IF
3571             END IF
3572 !
3573             RETURN
3574 !
3575 !         End of DTRMV .
3576 !
3577             END SUBROUTINE DTRMV
3578
3579 !
3580 ! =====
3581             SUBROUTINE DGETRI( N, A, LDA, IPIV, WORK, LWORK, INFO )
3582 !
3583 ! -- LAPACK computational routine (version 3.7.0) --
3584 ! -- LAPACK is a software package provided by Univ. of Tennessee, --
3585 ! -- Univ. of California Berkeley, Univ. of Colorado Denver and NAG Ltd.--
3586 !         December 2016
3587 !
3588 ! .. Scalar Arguments ..
3589             INTEGER          INFO, LDA, LWORK, N
3590 ! ..
3591 ! .. Array Arguments ..
3592             INTEGER          IPIV( * )
3593             DOUBLE PRECISION A( LDA, * ), WORK( * )
3594 ! ..
3595 ! =====
3596 !
3597 ! .. Parameters ..
3598             DOUBLE PRECISION ZERO, ONE
3599             PARAMETER        ( ZERO = 0.0D+0, ONE = 1.0D+0 )
3600 ! ..
3601 ! .. Local Scalars ..
3602 ! ..
3603             LOGICAL          LQUERY
3604             INTEGER          I, IWS, J, JB, JJ, JP, LDWORK, LWKOPT, NB, &
3605             &                NBMIN, NN
3606 ! ..
3607 ! .. External Functions ..
3608             INTEGER          ILAENV
3609             EXTERNAL         ILAENV
3610 ! ..
3611 ! .. External Subroutines ..
3612             EXTERNAL         DGEMM, DGEMV, DSWAP, DTRSM, DTRTRI, XERBLA
3613 ! ..
3614 ! .. Intrinsic Functions ..
3615             INTRINSIC        MAX, MIN
3616 ! ..
3617 ! .. Executable Statements ..
3618 !
3619 !         Test the input parameters.
3620 !

```

```

3621     INFO = 0
3622     NB = ILAENV( 1, 'DGETRI', ' ', N, -1, -1, -1 )
3623     LWKOPT = N*N*NB
3624     WORK( 1 ) = LWKOPT
3625     LQUERY = ( LWORK.EQ.-1 )
3626     IF( N.LT.0 ) THEN
3627         INFO = -1
3628     ELSE IF( LDA.LT.MAX( 1, N ) ) THEN
3629         INFO = -3
3630     ELSE IF( LWORK.LT.MAX( 1, N ) .AND. .NOT.LQUERY ) THEN
3631         INFO = -6
3632     END IF
3633     IF( INFO.NE.0 ) THEN
3634         CALL XERBLA( 'DGETRI', -INFO )
3635         RETURN
3636     ELSE IF( LQUERY ) THEN
3637         RETURN
3638     END IF
3639 !
3640 !     Quick return if possible
3641 !
3642     IF( N.EQ.0 ) &
3643     &     RETURN
3644 !
3645 !     Form inv(U).  If INFO > 0 from DTRTRI, then U is singular,
3646 !     and the inverse is not computed.
3647 !
3648     CALL DTRTRI( 'Upper', 'Non-unit', N, A, LDA, INFO )
3649     IF( INFO.GT.0 ) &
3650     &     RETURN
3651 !
3652     NBMIN = 2
3653     LDWORK = N
3654     IF( NB.GT.1 .AND. NB.LT.N ) THEN
3655         IWS = MAX( LDWORK*NB, 1 )
3656         IF( LWORK.LT.IWS ) THEN
3657             NB = LWORK / LDWORK
3658             NBMIN = MAX( 2, ILAENV( 2, 'DGETRI', ' ', N, -1, -1, -1 ) )
3659         END IF
3660     ELSE
3661         IWS = N
3662     END IF
3663 !
3664 !     Solve the equation inv(A)*L = inv(U) for inv(A).
3665 !
3666     IF( NB.LT.NBMIN .OR. NB.GE.N ) THEN
3667 !
3668 !         Use unblocked code.
3669 !
3670         DO 20 J = N, 1, -1
3671 !
3672 !             Copy current column of L to WORK and replace with zeros.
3673 !
3674             DO 10 I = J + 1, N
3675                 WORK( I ) = A( I, J )
3676                 A( I, J ) = ZERO
3677 10          CONTINUE
3678 !
3679 !             Compute current column of inv(A).
3680 !
3681             IF( J.LT.N ) &

```

```

3682      &          CALL DGEMV( 'No transpose', N, N-J, -ONE, A( 1, J+1 ), &
3683      &          LDA, WORK( J+1 ), 1, ONE, A( 1, J ), 1 )
3684      20      CONTINUE
3685      ELSE
3686      !
3687      !          Use blocked code.
3688      !
3689      NN = ( ( N-1 ) / NB ) * NB + 1
3690      DO 50 J = NN, 1, -NB
3691          JB = MIN( NB, N-J+1 )
3692      !
3693      !          Copy current block column of L to WORK and replace with
3694      !          zeros.
3695      !
3696      DO 40 JJ = J, J + JB - 1
3697          DO 30 I = JJ + 1, N
3698              WORK( I+( JJ-J ) * LDWORK ) = A( I, JJ )
3699              A( I, JJ ) = ZERO
3700      30      CONTINUE
3701      40      CONTINUE
3702      !
3703      !          Compute current block column of inv(A).
3704      !
3705      IF( J+JB.LE.N ) &
3706      &          CALL DGEMM( 'No transpose', 'No transpose', N, JB,      &
3707      &          N-J-JB+1, -ONE, A( 1, J+JB ), LDA,      &
3708      &          WORK( J+JB ), LDWORK, ONE, A( 1, J ), LDA )
3709      CALL DTRSM( 'Right', 'Lower', 'No transpose', 'Unit', N, JB, &
3710      &          ONE, WORK( J ), LDWORK, A( 1, J ), LDA )
3711      50      CONTINUE
3712      END IF
3713      !
3714      !          Apply column interchanges.
3715      !
3716      DO 60 J = N - 1, 1, -1
3717          JP = IPIV( J )
3718          IF( JP.NE.J ) &
3719      &          CALL DSWAP( N, A( 1, J ), 1, A( 1, JP ), 1 )
3720      60      CONTINUE
3721      !
3722      WORK( 1 ) = IWS
3723      RETURN
3724      !
3725      !          End of DGETRI
3726      !
3727      END SUBROUTINE DGETRI
3728      !
3729      !          =====
3730      SUBROUTINE DPBTF2( UPLO, N, KD, AB, LDAB, INFO )
3731      !
3732      !          -- LAPACK computational routine (version 3.7.0) --
3733      !          -- LAPACK is a software package provided by Univ. of Tennessee, --
3734      !          -- Univ. of California Berkeley, Univ. of Colorado Denver and NAG Ltd. ---
3735      !          December 2016
3736      !
3737      !          .. Scalar Arguments ..
3738      CHARACTER          UPLO
3739      INTEGER            INFO, KD, LDAB, N
3740      !          ..
3741      !          .. Array Arguments ..
3742      DOUBLE PRECISION  AB( LDAB, * )

```



```

3743 ! ..
3744 !
3745 ! =====
3746 !
3747 ! .. Parameters ..
3748 DOUBLE PRECISION ONE, ZERO
3749 PARAMETER ( ONE = 1.0D+0, ZERO = 0.0D+0 )
3750 ! ..
3751 ! .. Local Scalars ..
3752 LOGICAL UPPER
3753 INTEGER J, KLD, KN
3754 DOUBLE PRECISION AJJ
3755 ! ..
3756 ! .. External Functions ..
3757 ! LOGICAL LSAME
3758 ! EXTERNAL LSAME
3759 ! ..
3760 ! .. External Subroutines ..
3761 ! EXTERNAL DSCAL, DSYSR, XERBLA
3762 ! ..
3763 ! .. Intrinsic Functions ..
3764 INTRINSIC MAX, MIN, SQRT
3765 ! ..
3766 ! .. Executable Statements ..
3767 !
3768 ! Test the input parameters.
3769 !
3770 INFO = 0
3771 UPPER = LSAME( UPLO, 'U' )
3772 IF( .NOT.UPPER .AND. .NOT.LSAME( UPLO, 'L' ) ) THEN
3773     INFO = -1
3774 ELSE IF( N.LT.0 ) THEN
3775     INFO = -2
3776 ELSE IF( KD.LT.0 ) THEN
3777     INFO = -3
3778 ELSE IF( LDAB.LT.KD+1 ) THEN
3779     INFO = -5
3780 END IF
3781 IF( INFO.NE.0 ) THEN
3782     CALL XERBLA( 'DPBTF2', -INFO )
3783     RETURN
3784 END IF
3785 !
3786 ! Quick return if possible
3787 !
3788 IF( N.EQ.0 ) &
3789 & RETURN
3790 !
3791 KLD = MAX( 1, LDAB-1 )
3792 !
3793 IF( UPPER ) THEN
3794 !
3795 ! Compute the Cholesky factorization A = U**T*U.
3796 !
3797 DO 10 J = 1, N
3798 !
3799 ! Compute U(J,J) and test for non-positive-definiteness.
3800 !
3801 AJJ = AB( KD+1, J )
3802 IF( AJJ.LE.ZERO ) &
3803 & GO TO 30

```

```

3804         AJJ = SQRT( AJJ )
3805         AB( KD+1, J ) = AJJ
3806 !
3807 !         Compute elements J+1:J+KN of row J and update the
3808 !         trailing submatrix within the band.
3809 !
3810         KN = MIN( KD, N-J )
3811         IF( KN.GT.0 ) THEN
3812             CALL DSCAL( KN, ONE / AJJ, AB( KD, J+1 ), KLD )
3813             CALL DSYR( 'Upper', KN, -ONE, AB( KD, J+1 ), KLD, &
3814 &                 AB( KD+1, J+1 ), KLD )
3815         END IF
10      CONTINUE
3816     ELSE
3817
3818 !
3819 !         Compute the Cholesky factorization A = L*L**T.
3820 !
3821     DO 20 J = 1, N
3822 !
3823 !         Compute L(J,J) and test for non-positive-definiteness.
3824 !
3825         AJJ = AB( 1, J )
3826         IF( AJJ.LE.ZERO ) &
3827 &         GO TO 30
3828         AJJ = SQRT( AJJ )
3829         AB( 1, J ) = AJJ
3830 !
3831 !         Compute elements J+1:J+KN of column J and update the
3832 !         trailing submatrix within the band.
3833 !
3834         KN = MIN( KD, N-J )
3835         IF( KN.GT.0 ) THEN
3836             CALL DSCAL( KN, ONE / AJJ, AB( 2, J ), 1 )
3837             CALL DSYR( 'Lower', KN, -ONE, AB( 2, J ), 1, &
3838 &                 AB( 1, J+1 ), KLD )
3839         END IF
20      CONTINUE
3840     END IF
3841     RETURN
3842
3843 !
3844 30 CONTINUE
3845     INFO = J
3846     RETURN
3847 !
3848 !         End of DPBTF2
3849 !
3850     END SUBROUTINE DPBTF2
3851 !
3852 ! =====
3853     SUBROUTINE DSYR(UPLO,N,ALPHA,X,INCX,A,LDA)
3854 !
3855 ! -- Reference BLAS level2 routine (version 3.7.0) --
3856 ! -- Reference BLAS is a software package provided by Univ. of Tennessee, --
3857 ! -- Univ. of California Berkeley, Univ. of Colorado Denver and NAG Ltd.--
3858 ! December 2016
3859 !
3860 ! .. Scalar Arguments ..
3861     DOUBLE PRECISION ALPHA
3862     INTEGER INCX,LDA,N
3863     CHARACTER UPLO
3864 ! ..

```

```

3865 ! .. Array Arguments ..
3866 DOUBLE PRECISION A(LDA,*),X(*)
3867 ! ..
3868 !
3869 ! =====
3870 !
3871 ! .. Parameters ..
3872 DOUBLE PRECISION ZERO
3873 PARAMETER (ZERO=0.0D+0)
3874 ! ..
3875 ! .. Local Scalars ..
3876 DOUBLE PRECISION TEMP
3877 INTEGER I,INFO,IX,J,JX,KX
3878 ! ..
3879 ! .. External Functions ..
3880 LOGICAL LSAME
3881 EXTERNAL LSAME
3882 ! ..
3883 ! .. External Subroutines ..
3884 EXTERNAL XERBLA
3885 ! ..
3886 ! .. Intrinsic Functions ..
3887 INTRINSIC MAX
3888 ! ..
3889 !
3890 ! Test the input parameters.
3891 !
3892 INFO = 0
3893 IF (.NOT.LSAME(UPLO,'U') .AND. .NOT.LSAME(UPLO,'L')) THEN
3894     INFO = 1
3895 ELSE IF (N.LT.0) THEN
3896     INFO = 2
3897 ELSE IF (INCX.EQ.0) THEN
3898     INFO = 5
3899 ELSE IF (LDA.LT.MAX(1,N)) THEN
3900     INFO = 7
3901 END IF
3902 IF (INFO.NE.0) THEN
3903     CALL XERBLA('DSYR ',INFO)
3904     RETURN
3905 END IF
3906 !
3907 ! Quick return if possible.
3908 !
3909 IF ((N.EQ.0) .OR. (ALPHA.EQ.ZERO)) RETURN
3910 !
3911 ! Set the start point in X if the increment is not unity.
3912 !
3913 IF (INCX.LE.0) THEN
3914     KX = 1 - (N-1)*INCX
3915 ELSE IF (INCX.NE.1) THEN
3916     KX = 1
3917 END IF
3918 !
3919 ! Start the operations. In this version the elements of A are
3920 ! accessed sequentially with one pass through the triangular part
3921 ! of A.
3922 !
3923 IF (LSAME(UPLO,'U')) THEN
3924 !
3925 ! Form A when A is stored in upper triangle.

```

```

3926 !
3927     IF (INCX.EQ.1) THEN
3928         DO 20 J = 1,N
3929             IF (X(J).NE.ZERO) THEN
3930                 TEMP = ALPHA*X(J)
3931                 DO 10 I = 1,J
3932                     A(I,J) = A(I,J) + X(I)*TEMP
3933             10     CONTINUE
3934             END IF
3935         20     CONTINUE
3936     ELSE
3937         JX = KX
3938         DO 40 J = 1,N
3939             IF (X(JX).NE.ZERO) THEN
3940                 TEMP = ALPHA*X(JX)
3941                 IX = KX
3942                 DO 30 I = 1,J
3943                     A(I,J) = A(I,J) + X(IX)*TEMP
3944                     IX = IX + INCX
3945             30     CONTINUE
3946             END IF
3947             JX = JX + INCX
3948         40     CONTINUE
3949     END IF
3950 ELSE
3951 !
3952 !     Form A when A is stored in lower triangle.
3953 !
3954     IF (INCX.EQ.1) THEN
3955         DO 60 J = 1,N
3956             IF (X(J).NE.ZERO) THEN
3957                 TEMP = ALPHA*X(J)
3958                 DO 50 I = J,N
3959                     A(I,J) = A(I,J) + X(I)*TEMP
3960             50     CONTINUE
3961             END IF
3962         60     CONTINUE
3963     ELSE
3964         JX = KX
3965         DO 80 J = 1,N
3966             IF (X(JX).NE.ZERO) THEN
3967                 TEMP = ALPHA*X(JX)
3968                 IX = JX
3969                 DO 70 I = J,N
3970                     A(I,J) = A(I,J) + X(IX)*TEMP
3971                     IX = IX + INCX
3972             70     CONTINUE
3973             END IF
3974             JX = JX + INCX
3975         80     CONTINUE
3976     END IF
3977 END IF
3978 !
3979 RETURN
3980 !
3981 !     End of DSYR .
3982 !
3983 END SUBROUTINE DSYR
3984
3985 !
3986 !     =====

```

```

3987     SUBROUTINE DPOTF2( UPLO, N, A, LDA, INFO )
3988 !
3989 ! -- LAPACK computational routine (version 3.7.0) --
3990 ! -- LAPACK is a software package provided by Univ. of Tennessee, --
3991 ! -- Univ. of California Berkeley, Univ. of Colorado Denver and NAG Ltd.--
3992 ! December 2016
3993 !
3994 ! .. Scalar Arguments ..
3995 CHARACTER          UPLO
3996 INTEGER            INFO, LDA, N
3997 !
3998 ! .. Array Arguments ..
3999 DOUBLE PRECISION   A( LDA, * )
4000 !
4001 !
4002 ! =====
4003 !
4004 ! .. Parameters ..
4005 DOUBLE PRECISION   ONE, ZERO
4006 PARAMETER          ( ONE = 1.0D+0, ZERO = 0.0D+0 )
4007 !
4008 ! .. Local Scalars ..
4009 LOGICAL            UPPER
4010 INTEGER            J
4011 DOUBLE PRECISION   AJJ
4012 !
4013 ! .. External Functions ..
4014 LOGICAL            LSAME, DISNAN
4015 DOUBLE PRECISION   DDOT
4016 EXTERNAL          LSAME, DDOT, DISNAN
4017 !
4018 ! .. External Subroutines ..
4019 EXTERNAL          DGEMV, DSCAL, XERBLA
4020 !
4021 ! .. Intrinsic Functions ..
4022 INTRINSIC          MAX, SQRT
4023 !
4024 ! .. Executable Statements ..
4025 !
4026 ! Test the input parameters.
4027 !
4028 INFO = 0
4029 UPPER = LSAME( UPLO, 'U' )
4030 IF( .NOT.UPPER .AND. .NOT.LSAME( UPLO, 'L' ) ) THEN
4031     INFO = -1
4032 ELSE IF( N.LT.0 ) THEN
4033     INFO = -2
4034 ELSE IF( LDA.LT.MAX( 1, N ) ) THEN
4035     INFO = -4
4036 END IF
4037 IF( INFO.NE.0 ) THEN
4038     CALL XERBLA( 'DPOTF2', -INFO )
4039     RETURN
4040 END IF
4041 !
4042 ! Quick return if possible
4043 !
4044 IF( N.EQ.0 ) &
4045 & RETURN
4046 !
4047 IF( UPPER ) THEN

```

```

4048 !
4049 !     Compute the Cholesky factorization A = U**T *U.
4050 !
4051 !     DO 10 J = 1, N
4052 !
4053 !         Compute U(J,J) and test for non-positive-definiteness.
4054 !
4055 !         AJJ = A( J, J ) - DDOT( J-1, A( 1, J ), 1, A( 1, J ), 1 )
4056 !         IF( AJJ.LE.ZERO.OR.DISNAN( AJJ ) ) THEN
4057 !             A( J, J ) = AJJ
4058 !             GO TO 30
4059 !         END IF
4060 !         AJJ = SQRT( AJJ )
4061 !         A( J, J ) = AJJ
4062 !
4063 !         Compute elements J+1:N of row J.
4064 !
4065 !         IF( J.LT.N ) THEN
4066 !             CALL DGEMV( 'Transpose', J-1, N-J, -ONE, A( 1, J+1 ), &
4067 ! &             LDA, A( 1, J ), 1, ONE, A( J, J+1 ), LDA )
4068 !             CALL DSCAL( N-J, ONE / AJJ, A( J, J+1 ), LDA )
4069 !         END IF
4070 !     10 CONTINUE
4071 ! ELSE
4072 !
4073 !     Compute the Cholesky factorization A = L*L**T.
4074 !
4075 !     DO 20 J = 1, N
4076 !
4077 !         Compute L(J,J) and test for non-positive-definiteness.
4078 !
4079 !         AJJ = A( J, J ) - DDOT( J-1, A( J, 1 ), LDA, A( J, 1 ), &
4080 ! &             LDA )
4081 !         IF( AJJ.LE.ZERO.OR.DISNAN( AJJ ) ) THEN
4082 !             A( J, J ) = AJJ
4083 !             GO TO 30
4084 !         END IF
4085 !         AJJ = SQRT( AJJ )
4086 !         A( J, J ) = AJJ
4087 !
4088 !         Compute elements J+1:N of column J.
4089 !
4090 !         IF( J.LT.N ) THEN
4091 !             CALL DGEMV( 'No transpose', N-J, J-1, -ONE, A( J+1, 1 ), &
4092 ! &             LDA, A( J, 1 ), LDA, ONE, A( J+1, J ), 1 )
4093 !             CALL DSCAL( N-J, ONE / AJJ, A( J+1, J ), 1 )
4094 !         END IF
4095 !     20 CONTINUE
4096 ! END IF
4097 ! GO TO 40
4098 !
4099 ! 30 CONTINUE
4100 ! INFO = J
4101 !
4102 ! 40 CONTINUE
4103 ! RETURN
4104 !
4105 ! End of DPOTF2
4106 !
4107 ! END SUBROUTINE DPOTF2
4108 !

```

```

4109 ! =====
4110     DOUBLE PRECISION FUNCTION DDOT(N,DX,INCX,DY,INCY)
4111 !
4112 ! -- Reference BLAS level1 routine (version 3.8.0) --
4113 ! -- Reference BLAS is a software package provided by Univ. of Tennessee, --
4114 ! -- Univ. of California Berkeley, Univ. of Colorado Denver and NAG Ltd.--
4115 ! November 2017
4116 !
4117 ! .. Scalar Arguments ..
4118     INTEGER INCX,INCY,N
4119 ! ..
4120 ! .. Array Arguments ..
4121     DOUBLE PRECISION DX(*),DY(*)
4122 ! ..
4123 !
4124 ! =====
4125 !
4126 ! .. Local Scalars ..
4127     DOUBLE PRECISION DTEMP
4128     INTEGER I,IX,IY,M,MP1
4129 ! ..
4130 ! .. Intrinsic Functions ..
4131     INTRINSIC MOD
4132 ! ..
4133     DDOT = 0.0d0
4134     DTEMP = 0.0d0
4135     IF (N.LE.0) RETURN
4136     IF (INCX.EQ.1 .AND. INCY.EQ.1) THEN
4137 !
4138 !     code for both increments equal to 1
4139 !
4140 !
4141 !     clean-up loop
4142 !
4143         M = MOD(N,5)
4144         IF (M.NE.0) THEN
4145             DO I = 1,M
4146                 DTEMP = DTEMP + DX(I)*DY(I)
4147             END DO
4148             IF (N.LT.5) THEN
4149                 DDOT=DTEMP
4150             RETURN
4151             END IF
4152         END IF
4153         MP1 = M + 1
4154         DO I = MP1,N,5
4155             DTEMP = DTEMP + DX(I)*DY(I) + DX(I+1)*DY(I+1) + &
4156 &             DX(I+2)*DY(I+2) + DX(I+3)*DY(I+3) + DX(I+4)*DY(I+4)
4157         END DO
4158     ELSE
4159 !
4160 !     code for unequal increments or equal increments
4161 !     not equal to 1
4162 !
4163         IX = 1
4164         IY = 1
4165         IF (INCX.LT.0) IX = (-N+1)*INCX + 1
4166         IF (INCY.LT.0) IY = (-N+1)*INCY + 1
4167         DO I = 1,N
4168             DTEMP = DTEMP + DX(IX)*DY(IY)
4169             IX = IX + INCX

```

```

4170         IY = IY + INCY
4171     END DO
4172 END IF
4173 DDOT = DTEMP
4174 RETURN
4175 END FUNCTION DDOT
4176
4177 !
4178 ! =====
4179     LOGICAL FUNCTION DISNAN( DIN )
4180 !
4181 ! -- LAPACK auxiliary routine (version 3.7.1) --
4182 ! -- LAPACK is a software package provided by Univ. of Tennessee, --
4183 ! -- Univ. of California Berkeley, Univ. of Colorado Denver and NAG Ltd.---
4184 ! -- June 2017
4185 !
4186 ! .. Scalar Arguments ..
4187     DOUBLE PRECISION, INTENT(IN) :: DIN
4188 ! ..
4189 !
4190 ! =====
4191 !
4192 ! .. External Functions ..
4193 !     LOGICAL DLAISNAN
4194 !     EXTERNAL DLAISNAN
4195 ! ..
4196 ! .. Executable Statements ..
4197     DISNAN = DLAISNAN(DIN,DIN)
4198     RETURN
4199 END FUNCTION DISNAN
4200
4201 !
4202 ! =====
4203     LOGICAL FUNCTION DLAISNAN( DIN1, DIN2 )
4204 !
4205 ! -- LAPACK auxiliary routine (version 3.7.1) --
4206 ! -- LAPACK is a software package provided by Univ. of Tennessee, --
4207 ! -- Univ. of California Berkeley, Univ. of Colorado Denver and NAG Ltd.---
4208 ! -- June 2017
4209 !
4210 ! .. Scalar Arguments ..
4211     DOUBLE PRECISION, INTENT(IN) :: DIN1, DIN2
4212 ! ..
4213 !
4214 ! =====
4215 !
4216 ! .. Executable Statements ..
4217     DLAISNAN = (DIN1.NE.DIN2)
4218     RETURN
4219 END FUNCTION DLAISNAN
4220
4221 !
4222 ! =====
4223     SUBROUTINE DSYRK(UPLO,TRANS,N,K,ALPHA,A,LDA,BETA,C,LDC)
4224 !
4225 ! -- Reference BLAS level3 routine (version 3.7.0) --
4226 ! -- Reference BLAS is a software package provided by Univ. of Tennessee, --
4227 ! -- Univ. of California Berkeley, Univ. of Colorado Denver and NAG Ltd.---
4228 ! -- December 2016
4229 !
4230 ! .. Scalar Arguments ..

```



```

4231     DOUBLE PRECISION ALPHA,BETA
4232     INTEGER K,LDA,LDC,N
4233     CHARACTER TRANS,UPLO
4234 !     ..
4235 !     .. Array Arguments ..
4236     DOUBLE PRECISION A(LDA,*),C(LDC,*)
4237 !     ..
4238 !
4239 ! =====
4240 !
4241 !     .. External Functions ..
4242 !     LOGICAL LSAME
4243 !     EXTERNAL LSAME
4244 !     ..
4245 !     .. External Subroutines ..
4246 !     EXTERNAL XERBLA
4247 !     ..
4248 !     .. Intrinsic Functions ..
4249     INTRINSIC MAX
4250 !     ..
4251 !     .. Local Scalars ..
4252     DOUBLE PRECISION TEMP
4253     INTEGER I,INFO,J,L,NROWA
4254     LOGICAL UPPER
4255 !     ..
4256 !     .. Parameters ..
4257     DOUBLE PRECISION ONE,ZERO
4258     PARAMETER (ONE=1.0D+0,ZERO=0.0D+0)
4259 !     ..
4260 !
4261 !     Test the input parameters.
4262 !
4263     IF (LSAME(TRANS,'N')) THEN
4264         NROWA = N
4265     ELSE
4266         NROWA = K
4267     END IF
4268     UPPER = LSAME(UPLO,'U')
4269 !
4270     INFO = 0
4271     IF ((.NOT.UPPER) .AND. (.NOT.LSAME(UPLO,'L'))) THEN
4272         INFO = 1
4273     ELSE IF ((.NOT.LSAME(TRANS,'N')) .AND. &
4274 &          (.NOT.LSAME(TRANS,'T')) .AND. &
4275 &          (.NOT.LSAME(TRANS,'C'))) THEN
4276         INFO = 2
4277     ELSE IF (N.LT.0) THEN
4278         INFO = 3
4279     ELSE IF (K.LT.0) THEN
4280         INFO = 4
4281     ELSE IF (LDA.LT.MAX(1,NROWA)) THEN
4282         INFO = 7
4283     ELSE IF (LDC.LT.MAX(1,N)) THEN
4284         INFO = 10
4285     END IF
4286     IF (INFO.NE.0) THEN
4287         CALL XERBLA('DSYRK ',INFO)
4288         RETURN
4289     END IF
4290 !
4291 !     Quick return if possible.

```

```

4292 !
4293     IF ((N.EQ.0) .OR. (((ALPHA.EQ.ZERO).OR. &
4294 &      (K.EQ.0)).AND. (BETA.EQ.ONE))) RETURN
4295 !
4296 !     And when alpha.eq.zero.
4297 !
4298     IF (ALPHA.EQ.ZERO) THEN
4299         IF (UPPER) THEN
4300             IF (BETA.EQ.ZERO) THEN
4301                 DO 20 J = 1,N
4302                     DO 10 I = 1,J
4303                         C(I,J) = ZERO
4304             10             CONTINUE
4305             20             CONTINUE
4306             ELSE
4307                 DO 40 J = 1,N
4308                     DO 30 I = 1,J
4309                         C(I,J) = BETA*C(I,J)
4310             30             CONTINUE
4311             40             CONTINUE
4312             END IF
4313         ELSE
4314             IF (BETA.EQ.ZERO) THEN
4315                 DO 60 J = 1,N
4316                     DO 50 I = J,N
4317                         C(I,J) = ZERO
4318             50             CONTINUE
4319             60             CONTINUE
4320             ELSE
4321                 DO 80 J = 1,N
4322                     DO 70 I = J,N
4323                         C(I,J) = BETA*C(I,J)
4324             70             CONTINUE
4325             80             CONTINUE
4326             END IF
4327         END IF
4328         RETURN
4329     END IF
4330 !
4331 !     Start the operations.
4332 !
4333     IF (LSAME(TRANS,'N')) THEN
4334 !
4335 !     Form C := alpha*A*A**T + beta*C.
4336 !
4337         IF (UPPER) THEN
4338             DO 130 J = 1,N
4339                 IF (BETA.EQ.ZERO) THEN
4340                     DO 90 I = 1,J
4341                         C(I,J) = ZERO
4342             90             CONTINUE
4343                 ELSE IF (BETA.NE.ONE) THEN
4344                     DO 100 I = 1,J
4345                         C(I,J) = BETA*C(I,J)
4346             100            CONTINUE
4347                 END IF
4348                 DO 120 L = 1,K
4349                     IF (A(J,L).NE.ZERO) THEN
4350                         TEMP = ALPHA*A(J,L)
4351                         DO 110 I = 1,J
4352                             C(I,J) = C(I,J) + TEMP*A(I,L)

```

```

4353 110 CONTINUE
4354 END IF
4355 120 CONTINUE
4356 130 CONTINUE
4357 ELSE
4358 DO 180 J = 1,N
4359 IF (BETA.EQ.ZERO) THEN
4360 DO 140 I = J,N
4361 C(I,J) = ZERO
4362 140 CONTINUE
4363 ELSE IF (BETA.NE.ONE) THEN
4364 DO 150 I = J,N
4365 C(I,J) = BETA*C(I,J)
4366 150 CONTINUE
4367 END IF
4368 DO 170 L = 1,K
4369 IF (A(J,L).NE.ZERO) THEN
4370 TEMP = ALPHA*A(J,L)
4371 DO 160 I = J,N
4372 C(I,J) = C(I,J) + TEMP*A(I,L)
4373 160 CONTINUE
4374 END IF
4375 170 CONTINUE
4376 180 CONTINUE
4377 END IF
4378 ELSE
4379 !
4380 ! Form C := alpha*A**T*A + beta*C.
4381 !
4382 IF (UPPER) THEN
4383 DO 210 J = 1,N
4384 DO 200 I = 1,J
4385 TEMP = ZERO
4386 DO 190 L = 1,K
4387 TEMP = TEMP + A(L,I)*A(L,J)
4388 190 CONTINUE
4389 IF (BETA.EQ.ZERO) THEN
4390 C(I,J) = ALPHA*TEMP
4391 ELSE
4392 C(I,J) = ALPHA*TEMP + BETA*C(I,J)
4393 END IF
4394 200 CONTINUE
4395 210 CONTINUE
4396 ELSE
4397 DO 240 J = 1,N
4398 DO 230 I = J,N
4399 TEMP = ZERO
4400 DO 220 L = 1,K
4401 TEMP = TEMP + A(L,I)*A(L,J)
4402 220 CONTINUE
4403 IF (BETA.EQ.ZERO) THEN
4404 C(I,J) = ALPHA*TEMP
4405 ELSE
4406 C(I,J) = ALPHA*TEMP + BETA*C(I,J)
4407 END IF
4408 230 CONTINUE
4409 240 CONTINUE
4410 END IF
4411 END IF
4412 !
4413 RETURN

```

```

4414 !
4415 !   End of DSYRK .
4416 !
4417 !   END SUBROUTINE DSYRK
4418 !
4419 !
4420 !
4421 ! =====
4422 !   SUBROUTINE DPBTRF( UPLO, N, KD, AB, LDAB, INFO )
4423 !
4424 !   -- LAPACK computational routine (version 3.7.0) --
4425 !   -- LAPACK is a software package provided by Univ. of Tennessee, --
4426 !   -- Univ. of California Berkeley, Univ. of Colorado Denver and NAG Ltd.--
4427 !   December 2016
4428 !
4429 !   .. Scalar Arguments ..
4430 !   CHARACTER          UPLO
4431 !   INTEGER            INFO, KD, LDAB, N
4432 !   ..
4433 !   .. Array Arguments ..
4434 !   DOUBLE PRECISION  AB( LDAB, * )
4435 !   ..
4436 !
4437 ! =====
4438 !
4439 !   .. Parameters ..
4440 !   DOUBLE PRECISION  ONE, ZERO
4441 !   PARAMETER         ( ONE = 1.0D+0, ZERO = 0.0D+0 )
4442 !   INTEGER            NBMAX, LDWORK
4443 !   PARAMETER         ( NBMAX = 32, LDWORK = NBMAX+1 )
4444 !   ..
4445 !   .. Local Scalars ..
4446 !   INTEGER            I, I2, I3, IB, II, J, JJ, NB
4447 !   ..
4448 !   .. Local Arrays ..
4449 !   DOUBLE PRECISION  WORK( LDWORK, NBMAX )
4450 !   ..
4451 !   .. External Functions ..
4452 !   LOGICAL           LSAME
4453 !   INTEGER            ILAENV
4454 !   EXTERNAL          LSAME, ILAENV
4455 !   ..
4456 !   .. External Subroutines ..
4457 !   EXTERNAL          DGEMM, DPBTF2, DPOTF2, DSYRK, DTRSM, XERBLA
4458 !   ..
4459 !   .. Intrinsic Functions ..
4460 !   INTRINSIC         MIN
4461 !   ..
4462 !   .. Executable Statements ..
4463 !
4464 !   Test the input parameters.
4465 !
4466 !   INFO = 0
4467 !   IF( ( .NOT.LSAME( UPLO, 'U' ) ) .AND. &
4468 ! & ( .NOT.LSAME( UPLO, 'L' ) ) ) THEN
4469 !     INFO = -1
4470 !   ELSE IF( N.LT.0 ) THEN
4471 !     INFO = -2
4472 !   ELSE IF( KD.LT.0 ) THEN
4473 !     INFO = -3
4474 !   ELSE IF( LDAB.LT.KD+1 ) THEN

```

```

4475         INFO = -5
4476     END IF
4477     IF( INFO.NE.0 ) THEN
4478         CALL XERBLA( 'DPBTRF', -INFO )
4479         RETURN
4480     END IF
4481 !
4482 !     Quick return if possible
4483 !
4484     IF( N.EQ.0 ) &
4485     & RETURN
4486 !
4487 !     Determine the block size for this environment
4488 !
4489     NB = ILAENV( 1, 'DPBTRF', UPLO, N, KD, -1, -1 )
4490 !
4491 !     The block size must not exceed the semi-bandwidth KD, and must not
4492 !     exceed the limit set by the size of the local array WORK.
4493 !
4494     NB = MIN( NB, NBMAX )
4495 !
4496     IF( NB.LE.1 .OR. NB.GT.KD ) THEN
4497 !
4498 !         Use unblocked code
4499 !
4500         CALL DPBTF2( UPLO, N, KD, AB, LDAB, INFO )
4501     ELSE
4502 !
4503 !         Use blocked code
4504 !
4505         IF( LSAME( UPLO, 'U' ) ) THEN
4506 !
4507 !             Compute the Cholesky factorization of a symmetric band
4508 !             matrix, given the upper triangle of the matrix in band
4509 !             storage.
4510 !
4511 !             Zero the upper triangle of the work array.
4512 !
4513             DO 20 J = 1, NB
4514                 DO 10 I = 1, J - 1
4515                     WORK( I, J ) = ZERO
4516             10 CONTINUE
4517             20 CONTINUE
4518 !
4519 !             Process the band matrix one diagonal block at a time.
4520 !
4521             DO 70 I = 1, N, NB
4522                 IB = MIN( NB, N-I+1 )
4523 !
4524 !                 Factorize the diagonal block
4525 !
4526                 CALL DPOTF2( UPLO, IB, AB( KD+1, I ), LDAB-1, II )
4527                 IF( II.NE.0 ) THEN
4528                     INFO = I + II - 1
4529                     GO TO 150
4530                 END IF
4531                 IF( I+IB.LE.N ) THEN
4532 !
4533 !                     Update the relevant part of the trailing submatrix.
4534 !                     If A11 denotes the diagonal block which has just been
4535 !                     factorized, then we need to update the remaining

```

```

4536 !           blocks in the diagram:
4537 !
4538 !           A11  A12  A13
4539 !                A22  A23
4540 !                    A33
4541 !
4542 !           The numbers of rows and columns in the partitioning
4543 !           are IB, I2, I3 respectively. The blocks A12, A22 and
4544 !           A23 are empty if IB = KD. The upper triangle of A13
4545 !           lies outside the band.
4546 !
4547 !           I2 = MIN( KD-IB, N-I-IB+1 )
4548 !           I3 = MIN( IB, N-I-KD+1 )
4549 !
4550 !           IF( I2.GT.0 ) THEN
4551 !
4552 !               Update A12
4553 !
4554 !               CALL DTRSM( 'Left', 'Upper', 'Transpose',           &
4555 !                   & 'Non-unit', IB, I2, ONE, AB( KD+1, I ), &
4556 !                   & LDAB-1, AB( KD+1-IB, I+IB ), LDAB-1 )
4557 !
4558 !               Update A22
4559 !
4560 !               CALL DSYRK( 'Upper', 'Transpose', I2, IB, -ONE,     &
4561 !                   & AB( KD+1-IB, I+IB ), LDAB-1, ONE,         &
4562 !                   & AB( KD+1, I+IB ), LDAB-1 )
4563 !           END IF
4564 !
4565 !           IF( I3.GT.0 ) THEN
4566 !
4567 !               Copy the lower triangle of A13 into the work array.
4568 !
4569 !               DO 40 JJ = 1, I3
4570 !                   DO 30 II = JJ, IB
4571 !                       WORK( II, JJ ) = AB( II-JJ+1, JJ+I+KD-1 )
4572 !                   30 CONTINUE
4573 !               40 CONTINUE
4574 !
4575 !               Update A13 (in the work array).
4576 !
4577 !               CALL DTRSM( 'Left', 'Upper', 'Transpose',           &
4578 !                   & 'Non-unit', IB, I3, ONE, AB( KD+1, I ), &
4579 !                   & LDAB-1, WORK, LDWORK )
4580 !
4581 !               Update A23
4582 !
4583 !               IF( I2.GT.0 )
4584 !                   & CALL DGEMM( 'Transpose', 'No Transpose', I2, I3, &
4585 !                       & IB, -ONE, AB( KD+1-IB, I+IB ), &
4586 !                       & LDAB-1, WORK, LDWORK, ONE, &
4587 !                       & AB( 1+IB, I+KD ), LDAB-1 )
4588 !
4589 !               Update A33
4590 !
4591 !               CALL DSYRK( 'Upper', 'Transpose', I3, IB, -ONE,     &
4592 !                   & WORK, LDWORK, ONE, AB( KD+1, I+KD ), &
4593 !                   & LDAB-1 )
4594 !
4595 !               Copy the lower triangle of A13 back into place.
4596 !

```

```

4597             DO 60 JJ = 1, I3
4598                 DO 50 II = JJ, IB
4599                     AB( II-JJ+1, JJ+I+KD-1 ) = WORK( II, JJ )
4600             50             CONTINUE
4601             60             CONTINUE
4602                 END IF
4603             END IF
4604             70             CONTINUE
4605         ELSE
4606 !
4607 !         Compute the Cholesky factorization of a symmetric band
4608 !         matrix, given the lower triangle of the matrix in band
4609 !         storage.
4610 !
4611 !         Zero the lower triangle of the work array.
4612 !
4613             DO 90 J = 1, NB
4614                 DO 80 I = J + 1, NB
4615                     WORK( I, J ) = ZERO
4616             80             CONTINUE
4617             90             CONTINUE
4618 !
4619 !         Process the band matrix one diagonal block at a time.
4620 !
4621             DO 140 I = 1, N, NB
4622                 IB = MIN( NB, N-I+1 )
4623 !
4624 !         Factorize the diagonal block
4625 !
4626             CALL DPOTF2( UPLO, IB, AB( 1, I ), LDAB-1, II )
4627             IF( II.NE.0 ) THEN
4628                 INFO = I + II - 1
4629                 GO TO 150
4630             END IF
4631             IF( I+IB.LE.N ) THEN
4632 !
4633 !         Update the relevant part of the trailing submatrix.
4634 !         If A11 denotes the diagonal block which has just been
4635 !         factorized, then we need to update the remaining
4636 !         blocks in the diagram:
4637 !
4638 !             A11
4639 !             A21  A22
4640 !             A31  A32  A33
4641 !
4642 !         The numbers of rows and columns in the partitioning
4643 !         are IB, I2, I3 respectively. The blocks A21, A22 and
4644 !         A32 are empty if IB = KD. The lower triangle of A31
4645 !         lies outside the band.
4646 !
4647             I2 = MIN( KD-IB, N-I-IB+1 )
4648             I3 = MIN( IB, N-I-KD+1 )
4649 !
4650             IF( I2.GT.0 ) THEN
4651 !
4652 !                 Update A21
4653 !
4654             CALL DTRSM( 'Right', 'Lower', 'Transpose',      &
4655             & 'Non-unit', I2, IB, ONE, AB( 1, I ),      &
4656             & LDAB-1, AB( 1+IB, I ), LDAB-1 )
4657 !

```

```

4658 !           Update A22
4659 !
4660           CALL DSYRK( 'Lower', 'No Transpose', I2, IB, -ONE, &
4661 &                AB( 1+IB, I ), LDAB-1, ONE, &
4662 &                AB( 1, I+IB ), LDAB-1 )
4663           END IF
4664 !
4665           IF( I3.GT.0 ) THEN
4666 !
4667 !           Copy the upper triangle of A31 into the work array.
4668 !
4669           DO 110 JJ = 1, IB
4670             DO 100 II = 1, MIN( JJ, I3 )
4671               WORK( II, JJ ) = AB( KD+1-JJ+II, JJ+I-1 )
4672             100           CONTINUE
4673           110           CONTINUE
4674 !
4675 !           Update A31 (in the work array).
4676 !
4677           CALL DTRSM( 'Right', 'Lower', 'Transpose', &
4678 &                'Non-unit', I3, IB, ONE, AB( 1, I ), &
4679 &                LDAB-1, WORK, LDWORK )
4680 !
4681 !           Update A32
4682 !
4683           IF( I2.GT.0 ) &
4684 &           CALL DGEMM( 'No transpose', 'Transpose', I3, I2, &
4685 &                IB, -ONE, WORK, LDWORK, &
4686 &                AB( 1+IB, I ), LDAB-1, ONE, &
4687 &                AB( 1+KD-IB, I+IB ), LDAB-1 )
4688 !
4689 !           Update A33
4690 !
4691           CALL DSYRK( 'Lower', 'No Transpose', I3, IB, -ONE, &
4692 &                WORK, LDWORK, ONE, AB( 1, I+KD ), &
4693 &                LDAB-1 )
4694 !
4695 !           Copy the upper triangle of A31 back into place.
4696 !
4697           DO 130 JJ = 1, IB
4698             DO 120 II = 1, MIN( JJ, I3 )
4699               AB( KD+1-JJ+II, JJ+I-1 ) = WORK( II, JJ )
4700             120           CONTINUE
4701           130           CONTINUE
4702           END IF
4703           END IF
4704           140           CONTINUE
4705           END IF
4706           END IF
4707           RETURN
4708 !
4709           150 CONTINUE
4710           RETURN
4711 !
4712 !           End of DPBTRF
4713 !
4714           END SUBROUTINE DPBTRF
4715 !
4716 ! =====
4717           REAL FUNCTION SDSDOT(N,SB,SX,INCX,SY,INCY)
4718 !

```



```

4719 ! -- Reference BLAS level1 routine (version 3.8.0) --
4720 ! -- Reference BLAS is a software package provided by Univ. of Tennessee, --
4721 ! -- Univ. of California Berkeley, Univ. of Colorado Denver and NAG Ltd.--
4722 ! November 2017
4723 !
4724 ! .. Scalar Arguments ..
4725 REAL SB
4726 INTEGER INCX, INCY, N
4727 ! ..
4728 ! .. Array Arguments ..
4729 DOUBLE PRECISION SX(:), SY(:)
4730 !
4731 ! .. Local Scalars ..
4732 DOUBLE PRECISION DSDOT
4733 INTEGER I, KX, KY, NS
4734 ! ..
4735 ! .. Intrinsic Functions ..
4736 INTRINSIC DBLE
4737 ! ..
4738 DSDOT = SB
4739 IF (N.LE.0) THEN
4740     SDDOT = DSDOT
4741     RETURN
4742 END IF
4743 IF (INCX.EQ.INCY .AND. INCX.GT.0) THEN
4744 !
4745 ! Code for equal and positive increments.
4746 !
4747     NS = N*INCX
4748     DO I = 1, NS, INCX
4749         DSDOT = DSDOT + DBLE(SX(I))*DBLE(SY(I))
4750     END DO
4751 ELSE
4752 !
4753 ! Code for unequal or nonpositive increments.
4754 !
4755     KX = 1
4756     KY = 1
4757     IF (INCX.LT.0) KX = 1 + (1-N)*INCX
4758     IF (INCY.LT.0) KY = 1 + (1-N)*INCY
4759     DO I = 1, N
4760         DSDOT = DSDOT + DBLE(SX(KX))*DBLE(SY(KY))
4761         KX = KX + INCX
4762         KY = KY + INCY
4763     END DO
4764 END IF
4765 SDDOT = DSDOT
4766 RETURN
4767 END FUNCTION SDDOT
4768
4769 END MODULE ModuleLapack

```

8.12 makefile

```
1 all: multi-pred clean
2
3 multi-pred: ModuleIO.o ModuleGlobalParameters.o ModuleErrors.o ModuleFiles.o \
4             files.o ModuleReadWrite.o ModuleLapack.o ReadInput.o ModuleMultiPred.o\
5             MultiPredSolver.o multi-pred.o
6             ifort ModuleIO.o ModuleGlobalParameters.o ModuleErrors.o ModuleFiles.o\
7             files.o ModuleReadWrite.o ModuleLapack.o ReadInput.o ModuleMultiPred.o\
8             MultiPredSolver.o multi-pred.o -o multi-pred
9
10 ModuleIO.o: ModuleIO.f90
11     ifort -O3 -fast -c ModuleIO.f90
12
13 ModuleGlobalParameters.o: ModuleGlobalParameters.f90
14     ifort -O3 -fast -c ModuleGlobalParameters.f90
15
16 ModuleErrors.o: ModuleErrors.f90
17     ifort -O3 -fast -c ModuleErrors.f90
18
19 ModuleFiles.o: ModuleFiles.f90
20     ifort -O3 -fast -c ModuleFiles.f90
21
22 files.o: files.f90
23     ifort -O3 -fast -c files.f90
24
25 ModuleReadWrite.o: ModuleReadWrite.f90
26     ifort -O3 -fast -c ModuleReadWrite.f90
27
28 ModuleLapack.o: ModuleLapack.f90
29     ifort -O3 -fast -cpp -c ModuleLapack.f90
30
31 ReadInput.o: ReadInput.f90
32     ifort -O3 -fast -c ReadInput.f90
33
34 ModuleMultiPred.o: ModuleMultiPred.f90
35     ifort -O3 -fast -c ModuleMultiPred.f90
36
37 MultiPredSolver.o: MultiPredSolver.f90
38     ifort -O3 -fast -c MultiPredSolver.f90
39
40 multi-pred.o: multi-pred.f90
41     ifort -O3 -fast -c multi-pred.f90
42
43 clean:
44     rm -f *.o *.mod
45
```