UNIVERSITY OF CALIFORNIA, SAN DIEGO

The Hybrid Ensemble Smoother (HEnS)
&
Noncartesian Computational Interconnects

A dissertation submitted in partial satisfaction of the
requirements for the degree Doctor of Philosophy

in

Engineering Sciences with Specialization in Computational Science

by

Joseph B. Cessna

Committee in charge:

Professor Thomas Bewley, Chair
Professor Bruce Cornuelle
Professor Philip Gill
Professor Miroslav Krstic
Professor Daniel Tartakovsky

2010

The dissertation of Joseph B. Cessna is approved, and it is acceptable in quality and form for publication on microfilm and electronically:

_____

_____

_____

_____

_____

Chair

University of California, San Diego

2010

To my family.

" . . . never underestimate the estimation problem."

– unknown

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I must first thank my entire family. Without their continued love and support, I would not have the option to pursue my dreams. Jilly, you are the light of my life, my love, and my best friend. Bennett, you are my newest inspiration and motivation. And, Kodi, thanks for always knowing how to make me smile. I love you all.

My parents, Jeff & Sally, and my parents-in-law, George & Sue, thanks for all your love and insight. I appreciate all the backing through the years (both emotional and financial, of course). And while you were half a continent away, our long discussions over the phone always seemed to bridge the distance. Thanks to Casey, for constantly listening and understanding.

I would never have reached as far as I have without the unwavering support of my adviser Tom Bewley. His insight and direction allowed me to pursue many exciting research problems in a ridiculous number of directions, and his sense of humor and easygoing personality allowed me to pretend that they were all related. I value our personal relationship as much as that of our professional one; your compassion and understanding has helped me through many difficult times when most of my support structure was states away.

Thanks to all my friends and colleagues in the Flow Control Lab: Chris Colburn, David Zhang, Paul Belitz, and Rob Krohn. Your friendship has been

invaluable, helping me through the dog-days of grad school. Thanks for putting up with my incessantly sarcastic sense of humor.

Thanks to Bob Skelton and Mauricio de Oliveira for all the crazy experiences one might expect when dealing with tensegrity. Thank you for your continued support both at school and at home.

I gratefully acknowledge Los Alamos National Lab for their generous financial support that allowed me to focus my time solely on research.

The text of this dissertation includes the reprints of the following papers, either accepted of submitted for consideration at the time of publication. The dissertation author was the primary investigator and author of these publications.

[Chapter 1]

Cessna, J. and Bewley, T. (2010) A Hybrid (variational / Kalman) ensemble smoother for the estimation of nonlinear high-dimensional discretizations of PDE systems. *Under preparation, IEEE Transactions on Automatic Control.*

[Chapter 3]

Cessna, J. and Bewley, T. (2009) Structured computational interconnects. *11^{th} IEEE International Workshop on System-Level Prediction*, San Francisco, CA, USA.

[Chapter 4]

Cessna, J. and Bewley, T. (2010) Three general classes of regular, directed graphs. *Under preparation, IEEE Embedded Systems Letters.*

VITA

2010	Doctor of Philosophy in Engineering Sciences with Specialization in Computational Science University of California, San Diego
2007	Master of Science in Engineering Sciences (Engr. Physics), University of California, San Diego
2004	Bachelor of Science in Engineering Mechanics/Astronautics, University of Wisconsin, Madison
2004	Bachelor of Science in Mathematics, University of Wisconsin, Madison

JOURNAL PUBLICATIONS

• Cessna, J. and Bewley, T. (2010) A Hybrid (variational/Kalman) ensemble smoother for the estimation of nonlinear high-dimensional discretizations of PDE systems. *Under preparation, IEEE Transactions on Automatic Control*.

• Cessna, J. and Bewley, T. (2010) Three general classes of regular, directed graphs. *Under preparation, IEEE Embedded Systems Letters*.

• Bewley, T., Cessna, J., and Belitz, P. (2009) New horizons in sphere packing, part I: from dense to rare. *Submitted to SIAM Review*.

• Kammer, D., Cessna, J., and Kostuch, A. (2007) A generalization of effective mass for selecting free-free target modes. *Journal of Vibration and Acoustics*. **129**, 121–127.

SELECT PROCEEDINGS

• Cessna, J. and Bewley, T. (2009) Hexagonally structured computational interconnects and their scalable extension to spherical domains. *Proceedings from 11th IEEE International Workshop on System-Level Prediction*, San Francisco, CA, USA.

• Bewley, T., Cessna, J., Colburn, C., Ham, F., Iaccarino, G., and Wang, Q. (2008) Object oriented implementation of the enve estimation / forecasting algorithm and its application to high performance turbulence codes. *Proceedings from 2008 Stanford CTR Summer Program*, Stanford, CA, USA.

• Cessna, J., Colburn, C., and Bewley, T. (2008) EnVE: A new estimation algorithm for weather forecasting and flow control. *Proceedings from 4th AIAA Flow Control Conference*, Seattle, WA, USA.

• Cessna, J., Colburn, C., and Bewley, T. (2007) Multiscale retrograde estimation and forecasting of chaotic nonlinear systems. *Proceedings from 46th IEEE Conference on Decision and Control*, New Orleans, LA, USA.

SELECT PRESENTATIONS

• Cessna, J. and Bewley, T. (2009) Structured computational interconnects. *11^{th} IEEE International Workshop on System-Level Prediction*, San Francisco, CA, USA.

• Cessna, J. and Bewley, T. (2009) Structured computational interconnects on a sphere for the efficient parallel solution of the 2D shallow-water equations. *3^{rd} Southern California Symposium on Flow Physics*, La Jolla, CA, USA.

• Cessna, J. and Bewley, T. (2009) Structured computational interconnects. *UCSD CSME Seminar Series*, La Jolla, CA, USA.

• Cessna, J., Colburn, C., and Bewley, T. (2009) Ensemble/Variational Estimation (EnVE) theoretical foundations and applications to turbulent flows in complex geometries. *American Meteorological Society annual meeting*, Phoenix, AZ, USA.

• Cessna, J., Colburn, C., and Bewley, T. (2008) Ensemble/Variational Estimation (EnVE) and its applications to turbulent flows in complex geometries. *American Physical Society, Division of Fluid Dynamics*, San Antonio, TX, USA.

• Cessna, J., Colburn, C., and Bewley, T. (2008) EnVE: A consistent hybrid ensemble/variational estimation strategy for multiscale uncertain systems. *SIAM annual meeting*, San Diego, CA, USA.

• Cessna, J., Colburn, C., and Bewley, T. (2008) EnVE: A new estimation algorithm for weather forecasting and flow control. *4th AIAA Flow Control Conference*, Seattle, WA, USA.

• Cessna, J., Colburn, C., and Bewley, T. (2007) Retrograde estimation and forecasting of chaotic systems, part 2: application to chaotic and multiscale model problems. *American Physical Society, Division of Fluid Dynamics*, Salt Lake City , UT, USA.

POSTERS

• Cessna, J. and Bewley, T. (2009) Structured computational interconnects on a sphere. *Jacobs School Research Expo, UCSD*, La Jolla, CA, USA.

• Cessna, et. al. (2008) Hybrid ensemble/variational estimation (EnVE) and adaptive observation (EnVO). *WWRP/THORPEX Workshop on 4D-Var and EnKF Inter-comparisons*, Buenos Aires, Argentina.

• Colburn, C., Cessna, J., and Bewley, T. (2008) Hybrid sequential (enkf) / variational (4DVar) estimation of chaotic systems, part 2: EnVE, advantages and applications. *Jacobs School Research Expo, UCSD*, La Jolla, CA, USA.

• Cessna, J., Colburn, C., and Bewley, T. (2008) Hybrid sequential (enkf) / variational (4DVar) estimation of chaotic systems, part 1: EnVE, theoretical foundations. *Jacobs School Research Expo, UCSD*, La Jolla, CA, USA.

• Cessna, J., Colburn, C., and Bewley, T. (2007) Multiscale retrograde estimation and forecasting of chaotic nonlinear systems. *46th IEEE Conference on Decision and Control*, New Orleans, USA.

• Cessna, J., Colburn, C., and Bewley, T. (2007) Multiscale retrograde estimation and forecasting of chaotic nonlinear systems. *Jacobs School Research Expo, UCSD*, La Jolla, CA, USA.

AWARDS AND HONORS

• Gordon Engineering Leadership Center - Inaugural Gordon Scholar *2009*

• UCSD Jacobs School Research Expo - Best Poster *2009*

• UW–Madison Schoofs Prize - Grand Prize Winner *2004*

ABSTRACT OF THE DISSERTATION

The Hybrid Ensemble Smoother (HEnS)
&
Noncartesian Computational Interconnects

by

Joseph B. Cessna

Doctor of Philosophy in Engineering Sciences with Specialization in Computational Science

University of California, San Diego, 2010

Professor Thomas Bewley, Chair

Two classes of state estimation schemes, variational (4DVar) and ensemble Kalman (EnKF), have been developed and used extensively by the weather forecasting community as tractable alternatives to the standard matrix-based Kalman update equations for the estimation of high-dimensional systems. Variational schemes iteratively minimize a cost function with respect to the state estimate, using efficient vector-based gradient descent methods, but fail to capture the moments of the PDF of this estimate. Ensemble Kalman methods represent well the principal moments of the PDF, accounting for the measurements with a se-

quence of Kalman-like updates with the covariance of the PDF approximated via the ensemble. Here, we first introduce a tractable method for updating an ensemble of estimates in a variational fashion, capturing correctly both the estimate and the leading moments of its PDF. We then extend this variational ensemble framework to facilitate its consistent hybridization with the ensemble Kalman smoother. Finally, it is shown that the resulting Hybrid (variational/Kalman) Ensemble Smoother (HEnS) significantly outperforms the existing 4DVar and EnKF approaches used operationally today for high-dimensional state estimation.

The second part of this dissertation examines the best possible interconnect topologies for switchless multiprocessor computer systems. We focus first on hexagonal interconnect graphs and their extension to problems on the sphere. Eight families of efficient tiled layouts have been discovered that make such interconnects trivial to scale to large cluster sizes while incorporating no long wires. In the resulting switchless interconnect designs, the *physical proximity* of the cells created and the *logical proximity* of the nodes to which these cells are assigned coincide perfectly, so all communication between physically adjacent cells during the PDE simulation require communication over just a single hop in the computational cluster. Lastly, we attempt to generalize two classes of directed graphs into a unified theory in which the well-known cartesian and butterfly graphs are special cases of a more general class of interconnect that better spans the design parameter space.

Part I

The Hybrid Ensemble Smoother

(HEnS)

Chapter 1

Theoretical Foundations

Joseph Cessna and Thomas Bewley

Abstract. Two classes of state estimation schemes, variational (4DVar) and ensemble Kalman (EnKF), have been developed and used extensively by the weather forecasting community as tractable alternatives to the standard matrix-based Kalman update equations for the estimation of high-dimensional nonlinear systems with possibly nongaussian PDFs. Variational schemes iteratively minimize a finite-horizon cost function with respect to the state estimate, using efficient vector-based gradient descent methods, but fail to capture the moments of the PDF of this estimate. Ensemble Kalman methods represent well the principle moments of the PDF, accounting for the measurements with a sequence of

Kalman-like updates with the covariance of the PDF approximated via the ensemble, but fail to provide a mechanism to reinterpret past measurements in light of new data. In this paper, we first introduce a tractable method for updating an ensemble of estimates in a variational fashion, capturing correctly both the estimate (via the ensemble mean) and the leading moments of its PDF (via the ensemble distribution). We then extend this variational ensemble framework to facilitate its consistent hybridization with the ensemble Kalman smoother. Finally, it is shown (on a low-dimensional model problem) that the resulting Hybrid (variational/Kalman) Ensemble Smoother (HEnS), which inherits the tractable extensibility to high-dimensional systems of the component methods upon which it is based, significantly outperforms the existing 4DVar and EnKF approaches used operationally today for high-dimensional state estimation.

1.1 Introduction

The estimation and forecasting of chaotic, multiscale, uncertain fluid systems is one of the most highly visible grand challenge problems of our generation. Specifically, this class of problems includes weather forecasting, climate forecasting, and flow control. The financial impact of a hurricane passing through a major metropolitan center regularly exceeds a billion dollars. Improved atmospheric forecasting techniques provide early and accurate warnings, which are critical to minimize the impact of such events. On longer time scales, the estimation and forecasting of changes in ocean currents and temperatures is essential for an improved understanding of changes to the earth's weather systems. On shorter time scales, feedback control of fluid systems (for reasons such as minimizing drag, maximizing harvested energy, etc.) in mechanical, aerospace, environmental, energy, and chemical engineering settings lead to a variety of similar estimation problems. While this paper makes no claims with regards to addressing the particular details of any of these important applications, it does introduce a new Hybrid (variational/Kalman) Ensemble Smoother (HEnS) for the estimation and forecasting of such multiscale uncertain fluid systems that might well have a transformational effect in all of these areas.

1.1.1 Historical developments

Much of the research today in state estimation (a.k.a. data assimilation) for multiscale uncertain fluid systems is focused on short- to medium-range weather forecasting. Towards this end, the methods available for this class of problems have matured greatly in the past 25 years. To set the stage, we must first mention a few related developments.

The full, correct answer to the state estimation of nonlinear systems with finite (and, thus, nongaussian) uncertainties dates back to the late 1950s (see [55]). As described clearly on page 164 of [32], it combines two simple steps:

(i) between measurement updates, the full probability density function (PDF) in phase space is propagated via the Kolmogorov forward equation (a.k.a. Fokker-Planck equation);

(ii) at measurement updates, the PDF is updated via application of Bayes' theorem.

During step (i), the PDF stretches and diffuses; during step (ii), the PDF is refocused. An efficient modern implementation of this idea using a grid-based method, leveraging effectively the fact that the PDF is usually nearly zero almost everywhere in phase space, is given in [8]; unfortunately, such methods are numerically intractable for systems with states of order $n \gtrsim 10$, even with modern supercomputing resources.

Particle filters (PF; see [52]) approximate the solution of such Bayesian estimation strategies using a Lagrangian approach. With such methods, a set of candidate state trajectories is followed to track the evolution of the probability distribution in time, and associated with each particle is a weight, which is modified via Bayes' theorem at each measurement update (while normalizing such that the weights always add to one). Unfortunately, application of such updates for successive measurements invariably leads to most weights being driven towards zero as the algorithm proceeds, a phenomenon known as *degeneracy*. To counter this tendency in order to maintain adequate resolution of the significant (nonzero) portion of the PDF, resampling of the PDF with a new distribution of particles (with equalized weights) is, from time to time, required. A variety of such resampling algorithms have been proposed. When using a large number of particles (necessary when attempting to resolve a nongaussian PDF of the state estimate), the sampling importance resampling algorithm proposed in [18] is commonly used. When using small number of particles N (specifically, for $N = 2n + 1$, used when considering a state estimate of order n with a Gaussian PDF), an unscented transform (see [33], [34]) can be used to resample while preserving exactly the covariance of the original distribution. Unfortunately, PFs are also numerically intractable in large-scale systems.

Kalman filters (see [57], [55], [36], [37], [3]) substantially simplify the full state estimation problem in the common situation in which the random variables

are all well approximated by Gaussian PDFs. In this case, the PDF of a given random variable, of order n, can be specified completely by keeping track of its mean (of order n) and its covariance (of order n^2), which enormously simplifies the complexity of the state estimation problem. With modern computational resources, Kalman filters can thus be deployed for systems with states of order up to $n \sim 1000$. Note that Extended Kalman filters, designed for nonlinear systems, are simply Kalman filters, designed based on linearization of the nonlinear system about the expected state trajectory, with the nonlinearity tacked back on in the eleventh hour.

Traditional Kalman and extended Kalman filters were investigated by [24] for atmospheric applications, with nonlinear high-dimensional systems of order $n \gtrsim 10^5$. These applications necessitate the computation of a reduced-rank approximation of the covariance matrix at the heart of the Kalman filter in order to be computationally tractable. Such reduced-rank approximations are known in the controls community as Chandresarkhar's method, and were introduced by [35].

1.1.2 Variational methods

Since the mid 1980s, the field of state estimation has seen two revolutionary advancements: variational methods and ensemble Kalman methods. Today, these two classes of methods, in roughly equal proportion worldwide, are used operationally for practical real-time atmospheric forecasting.

The first variational methods introduced were spatial (three-dimensional) variational methods (3DVar; see [42] and [48]), which provide an optimization framework that may be used to fit a large-scale model to a "snapshot" in time of available data. This was soon followed by the development of spatial/temporal (four-dimensional) variational methods (4DVar; see [17] and [50]), in which this optimization framework is extended to account for a time history of observations. This 4DVar framework has the effect of conditioning the resulting estimate on all included data, in a manner consistent with the Kalman Smoother (see [41], [51] and [15]).

Note that 4DVar was developed in parallel, and largely independently, in the controls and weather forecasting communities. In the controls community, the technique is referred to as Moving Horizon Estimation (MHE; see [46]). MHE was developed with low-dimensional ODE systems in mind; implementations of MHE typically search for a small time-varying "state disturbance" or "model error" term in addition to the initial state of the system in order reconcile the measurements with the model over the period of interest. 4DVar, in contrast, was developed with high-fidelity (that is, high-dimensional) discretizations of infinite-dimensional (PDE) systems in mind; in order to maintain numerical tractability, implementations of 4DVar typically do not search for such a time-varying model error term. Both 4DVar and MHE suffer from the fact that they only provide an updated mean trajectory, and not any updated higher-moment statistics. However, during

the minimization process, it is possible to build up an approximation to the cost function Hessian. For convex variational problems, this Hessian is directly related to the inverse of the updated covariance matrix. Several of these methods are outlined in [23] and include the randomization method (which uses the statistics of perturbed gradients), the Lanczos method (which exploits the coupling between Lanczos vectors and conjugate gradient directions), and the BFGS method (which explicitly builds up the Hessian during minimization). All three of these methods fail to provide an effective means for *propagating* the updated statistics forward in time, and thus are not typically tractable for variational schemes that cycle over multiple, successive windows.

Another technique that has been introduced to accelerate MHE/4DVar implementations is multiple shooting (see [40]). With this technique, the horizon of interest is split into two or more subintervals. The initial conditions (and, in some cases, the time-varying model error term) for each subinterval are first initialized and optimized independently; these several independent solutions are then adjusted so that the trajectories coincide at the matching points between the subintervals.

1.1.3 Ensemble Kalman methods

The more recent development of the Ensemble Kalman Filter (EnKF) (see [19], [29], [30], [20]) has focused much attention on an important refinement of

the (sequential) Kalman method in which the estimation statistics are intrinsically represented via the distribution of a cluster or "ensemble" of state estimates in phase space, akin to the particle filters mentioned previously but without separate weights for each ensemble member. As with particle filters, the simultaneous simulation of several perturbed trajectories of the state estimate eliminates the need to propagate the state covariance matrix along with the estimate as required by traditional Kalman and extended Kalman approaches. Instead, this covariance information is approximated based on the spread of the ensemble members (with equal weights) in order to compute Kalman-like updates *to the position of each ensemble member*[1] at the measurement times (for further discussion, see §1.2.3).

Since its introduction, the EnKF has spawned many variations and modifications that seek to improve both its performance and its numerical tractability. For example, Kalman square-root filters update the analysis only once, in a manner different than the traditional perturbed observation method. Some square-root filters introduced include the ensemble adjustment filter of [4], the ensemble transform filter of [9], and the ensemble square-root filter of [60]. Work has also been done (see [39]) to further relax the linear Gaussian assumptions with regards to the interpolation between the observation and the background statistics.

Another essential advancement in the implementation of the EnKF is the idea of covariance localization, as discussed in [27] and [47]. With covariance

[1]That is, rather than updating individual weights for each member separately, as done at in particle filters.

localization, spurious correlations of the uncertainty covariance over large distances are reduced in an ad hoc fashion in order to improve the overall performance of the estimation algorithm. This adjustment is motivated by the rank-deficiency of the ensemble approximation of the covariance matrix, and facilitates parallel implementation of the resulting algorithm.

The Ensemble Kalman Smoother (EnKS) [21] is the analogous ensemble extension of the standard Kalman Smoother. With the EnKS, updates are performed on past estimates based on future observations in a manner similar to the EnKF. With the EnKS, the smoothed updates are a function of time correlations between two ensemble estimates at the appropriate times. Although each individual update is tractable, it becomes infeasible to update entire trajectories after each new observation; as a result, a fixed-lag or fixed-point EnKS is traditionally used in lieu of a full smoother. Another smoother in this class, the Ensemble Smoother (ES; see [59]), uses ensemble statistics to calculate a variance minimizing estimate, but in practice, for nonlinear systems, performs poorly even when compared to the standard EnKF.

For nonlinear systems, the ensemble Kalman framework is suboptimal due to its reliance on a Kalman-like measurement update formula, which is predicated on a Gaussian distribution of the estimate uncertainty. The more general Particle Filter (PF) method described in §1.1.1, in contrast, is a full Bayesian approach, with the PDF approximated in a Largrangian fashion akin to the ensemble Kalman

framework. The Particle Kalman Filter (PKF) method proposed by [28], which attempts to combine the PF and EnKF approaches in order to inherit the nongaussian uncertainty characterization of the PF method and the numerical tractability of the EnKF method, appears to be promising; this method could potentially benefit from further hybridization with a variational approach, as proposed below.

1.1.4 Hybrid methods

The two modern schools of thought in large-scale state estimation for multiscale uncertain systems [namely, space/time variational methods (§1.1.2) and ensemble Kalman methods (§1.1.3)] have, for the most part, remained independent, despite their similar theoretical backgrounds. The weather forecasting community has made considerable efforts to compare and contrast both the performance and the theoretical foundation of these two methods (see, e.g., [43], [11], [38], and [25]). While these comparisons are enlightening, it is quite possible that the optimal method for many large-scale state estimation problems cases may well be a hybridization of the two frameworks, as suggested by [25]. We have identified three recent attempts at such hybridization:

1. the 3DVar/EnKF method of [26],

2. the 4DEnKF method of [31], and

3. the E4DVar method of [61].

The 3DVar/EnKF algorithm introduced by [26] utilizes the ensemble framework to propagate the estimate statistics in a nonlinear setting, but does not exploit the temporal smoothing characteristics of the 4DVar algorithm. The 4DEnKF (4D Ensemble Kalman Filter) introduced by [31] provides a means for assimilating past (and non-uniform) observations in a sequential framework, but does not intrinsically smooth the resulting estimate or fully implement the 4DVar framework. The E4DVar (Ensemble 4DVar) method discussed by [61], which is the closest existing method to the hybrid smoother proposed here, runs a 4DVar and EnKF in parallel, sequentially shifting the mean of the ensemble based on the 4DVar result and providing the background term of the 4DVar algorithm based on the EnKF result; however, this method does not attempt a tighter coupling of the ensemble and variational approaches by using an Ensemble Kalman Smoother to initialize better (and, thus, accelerate) the variational iteration.

The three attempts at hybridization discussed above struggle with the inability of traditional variational iterations to update correctly the statistics of the PDF (covariance, etc.). This is crucial for a consistent[2] hybrid method, thus motivating the precise formulation of ensemble variation methods in §1.3 below. The VAE (Variational Assimilation Ensemble) method of [6] runs a half-dozen per-

[2]The word "consistent" is used in a precise fashion in this paper to mean an estimation method that reduces to exactly the Kalman filter in the case that the system happens to be linear, the disturbances happen to be Gaussian, and (in the case of an ensemble-based method) a sufficient number of ensemble members is used.

turbed decoupled 4DVar or 3DFgat[3] assimilations in parallel to estimate error covariances, and is the closest existing method we have found in the literature to a true ensemble-variation method. However, to the best of our knowledge, the current paper lays out the first complete mathematical foundation for a pure variational method that provides consistent, updated ensemble statistics upon algorithm convergence.

We can now classify the full taxonomy of ensemble-based methods (see Figure 1.1). Until now, these methods have been split into two distinct families: ensemble variation methods (suggested previously, but described formally for perhaps the first time in §1.3) and the well-known ensemble Kalman methods. Each family consists of filter variants (En3DVar and EnKF) and smoother variants (En4DVar and EnKS).

The proposed new algorithm, the Hybrid Ensemble Smoother (HEnS), is a consistent[4] and tightly-coupled hybrid of these two types of ensemble smoothers. HEnS uses the EnKS to precondition an appropriately defined En4DVar iteration. Essentially, the EnKS solution is used as a good initial condition for the ensemble variation problem, which in turn improves upon this smoothed estimate in a manner that would have been impossible using either method independently. In earlier work done by our group (see [12]), the 4DVar/MHE framework was inverted, pro-

[3]That is, 3D First Guess at the Appropriate Time (3DFgat), an intermediate variational method with complexity somewhere between that of 3DVar and 4DVar [see [22]].

[4]Again, meaning that it reduces to exactly the Kalman filter under the appropriate assumptions.

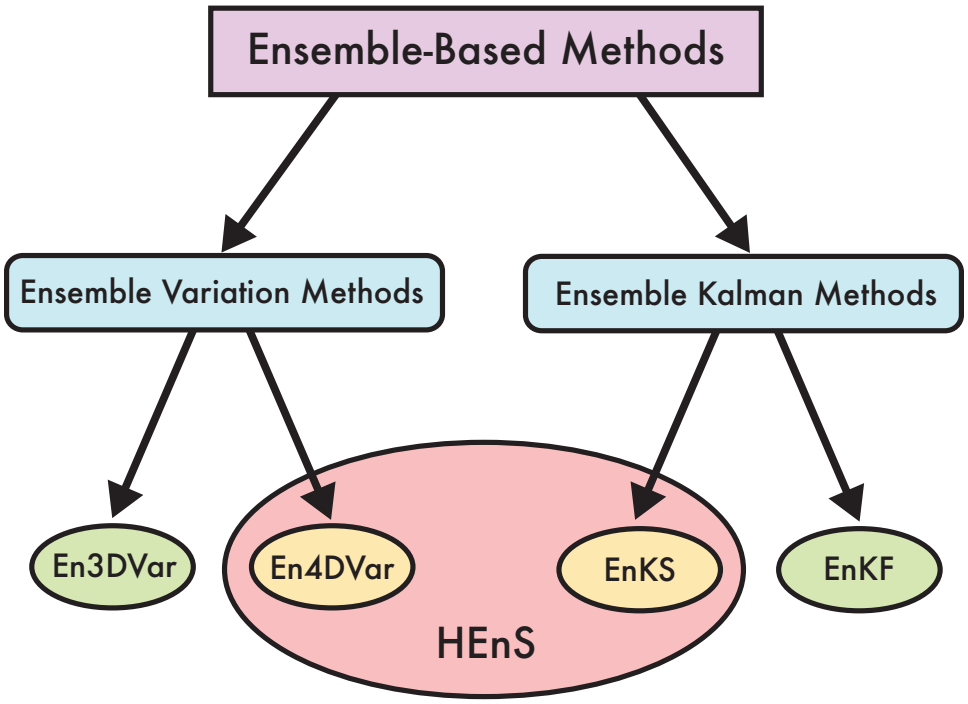

Figure 1.1: Ensemble-based methods can be classified into two distinct families. Ensemble Variation Methods (§1.3) are vector-based methods that iteratively minimize an appropriately-defined cost function to produce either a filtered (En3DVar) or a smoothed (En4DVar) estimate. Ensemble Kalman Methods (§1.2) use the ensemble statistics to approximate the full (but computationally intractable) matrix-based Kalman updates. The new Hybrid Ensemble Smoother (HEnS) is a consistent hybrid of the smoother variants of these two methods, a described in §1.4.

moting retrograde time marches (that is, marching the state estimate backward in time and the corresponding adjoint forward in time), which facilitated an adaptive (i.e., multiscale-in-time) receding-horizon optimization framework, dubbed EnVE (Ensemble Variational Estimation). Though the motivation behind this original work was sound, performance suffered, in part as a result of the inability of the variational formulation used to update correctly the higher-moment statistics of the ensemble; the present formulation corrects this significant shortcoming associated with the EnVE formulation.

Section 1.2 reviews the general forms of both the ensemble Kalman methods and the traditional variational methods. Section 1.3 describes the theoretical foundations for the ensemble variation methods, and identifies their relationship with the well-known KF and KS results. Building upon this, Section 1.4 describes the new hybrid smoother, HEnS, in detail. Finally, Section 1.5 contains a comparative example, performed on the low-dimensional chaotic Lorenz system, showing the performance of the various methods in a time-averaged setting. Two follow-up papers are planned which will detail the implementation of the HEnS algorithm on 1D, 2D, and 3D chaotic PDE systems, and introduce a unique adaptive observation algorithm which builds directly upon the hybrid framework discussed here.

1.2 Theoretical Background

1.2.1 Notation

As described above, the Hybrid Ensemble Smoother (HEnS) is a consistent data assimilation method that combines the key ideas of the sequential Ensemble Kalman Smoother (EnKS) and an ensemble variant of the batch (in time) variational method known as 4DVar in the weather forecasting community and as Moving Horizon Estimation (MHE) in the controls community. These methods are thus first reviewed briefly in a fairly standard form. Without loss of generality, the dynamic model used to introduce these methods is a continuous-time nonlinear ODE system with discrete-time measurements:

$$\frac{d\mathbf{x}(t)}{dt} = f(\mathbf{x}(t), \mathbf{w}(t)), \tag{1.1a}$$

$$\mathbf{y}_k = H\mathbf{x}(t_k) + \mathbf{v}_k, \tag{1.1b}$$

where the measurement noise $\mathbf{v}_k$ is a zero-mean, white, discrete-time random process with auto-correlation

$$R_{\mathbf{v}}(j;k) = E\{\, \mathbf{v}_{k+j}\, \mathbf{v}_k^T \,\} = R\delta_{j0}, \tag{1.2}$$

with covariance $R > 0$, and the state disturbance $\mathbf{w}(t)$ is a zero-mean, "nearly"-white[5] continuous-time random process with auto-correlation

$$R_{\mathbf{w}}(\tau; t) = E\{\, \mathbf{w}(t+\tau)\, \mathbf{w}^T(t)\, \} = Q\delta^\sigma(\tau), \tag{1.3a}$$

$$\text{where} \quad \delta^\sigma(\tau) = \frac{1}{\sigma\sqrt{2\pi}}\, e^{-\tau^2/(2\sigma^2)}, \tag{1.3b}$$

with spectral density $Q \geq 0$ and time correlation σ such that $0 < \sigma \ll 1$. Is also assumed that $\mathbf{w}(t)$ and $\mathbf{v}_k$ are uncorrelated.

The noisy measurements $\mathbf{y}_k$ are assumed to be taken at time $t_k = k\Delta t$ for a fixed sample period Δt. For the purpose of analysis, these observations are assumed available for a long history into the past, up to and including the present time of the system being estimated, which is often renormalized to be $t = t_K = T$. It is useful to think of t_K as the time of the most recent available measurement, so, accordingly, this measurement will be denoted $\mathbf{y}_K$ at the beginning of each analysis step. This sets the basis for the indexing notation used in this paper: $k = K$ represents the index of the most recent measurement, $1 \leq k \leq K$ is the set of indices of all available measurements, and $k > K$ indexes observations that are yet to be taken. Continuous-time trajectories, such as $\mathbf{x}(t)$ (the "truth" model), are defined for all time, but are frequently referenced at the observation times only. Hence, the following notation is used:

$$\mathbf{x}(k\Delta t) = \mathbf{x}(t_k) = \mathbf{x}_k. \tag{1.4}$$

[5]The case for infinitesimal σ is sometimes referred to as "continuous-time white noise", but presents certain technical difficulties [7].

1.2.2 Uncertainty Propagation in Chaotic Systems

Estimation, in general, involves the determination of a probability distribution. This probability distribution describes the likelihood that any particular point in phase space matches the truth model. That is, without knowing the actual state of a system, estimation strategies attempt to represent the probability of any given state using only a time history of noisy observations of a subset of the system and an approximate dynamic model of the system of interest. Given this statistical distribution, estimates can be inferred about the "most likely" state of the system, and how much confidence should be placed in that estimate. Unfortunately, in this most general form, the estimation problem is intractable in most systems. However, given certain justifiable assumptions about the nature of the model and its associated disturbances, simplifications can be applied with regards to how the probability distributions are modeled. Specifically, in linear systems with Gaussian uncertainty of the initial state, Gaussian state disturbances, and Gaussian measurement noise, it can be shown that the probability distribution of the optimal estimate is itself Gaussian [see, e.g., [3]]. Consequently, the entire distribution of the estimate in phase space can be represented exactly by its mean $\bar{\mathbf{x}}$ and its second moment about the mean (that is, its covariance), P, where

$$P = E\big[\,(\mathbf{x} - \bar{\mathbf{x}})(\mathbf{x} - \bar{\mathbf{x}})^T\big]. \tag{1.5}$$

This is the essential piece of theory that leads to the traditional Kalman Filter (KF; see [36] and [37]).

Sequential data assimilation methods provide a method to propagate the mean $\bar{\mathbf{x}}$ and covariance P forward in time, making the appropriate updates to both upon the receipt of each new measurement. Under the assumption of a linear system and white (or, in continuous time, "nearly" white) Gaussian state disturbances and measurement noise, the uncertainty distribution of the optimal estimate is itself Gaussian, and thus is *completely* described by the mean estimate $\bar{\mathbf{x}}$ and the covariance P propagated by the Kalman formulation. It is useful to think of these quantities, at any given time t_k, as being conditioned on a subset of the available measurements. The notation $\bar{\mathbf{x}}_{k|j}$ represents the mean estimate at time t_k given measurements up to and including time t_j. Similarly, $P_{k|j}$ represents the corresponding covariance of this estimate. In particular, $\bar{\mathbf{x}}_{k|k-1}$ and $P_{k|k-1}$ are often called the prediction estimate and prediction covariance, whereas $\bar{\mathbf{x}}_{k|k}$ and $P_{k|k}$ are often called the current estimate and the current covariance. Note that $\bar{\mathbf{x}}_{k|k+K}$, for some $K > 0$, is often called a smoothed estimate, and may be obtained in the sequential setting by a Kalman smoother [see, [51] and [3]].

As mentioned previously, for nonlinear systems with relatively small uncertainties, a common variation on the KF known as the Extended Kalman Filter (EKF) has been developed in which the mean and covariance are propagated, to first-order accuracy, about a linearized trajectory of the full system. Essen-

tially, if a Taylor-series expansion for the nonlinear evolution of the covariance is considered, and all terms higher than quadratic are dropped, what is left is the differential Riccati equation associated with the EKF covariance propagation. Though this approach gives acceptable estimation performance for nonlinear systems when uncertainties are small as compared to the fluctuations of the state itself, EKF estimators often diverge when uncertainties are more substantial, and other techniques are needed.

At its core, the linear thinking associated with the uncertainty propagation in the KF and EKF breaks down in chaotic systems. Chaotic systems are characterized by stable manifolds or "attractors" in n-dimensional phase space. Such attractors are fractional-dimensional subsets (a.k.a. "fractal" subsets) of the entire phase space. Trajectories of chaotic systems are stable with respect to the attractor in the sense that initial conditions off the attractor converge exponentially to the attractor, and trajectories on the attractor remain on the attractor. On the attractor, however, trajectories of chaotic systems are characterized by an *exponential divergence*–along the attractor–of slightly perturbed trajectories. That is, two points infinitesimally close on the attractor at one time will diverge exponentially from one another as the system evolves until they are effectively uncorrelated.

Just as an individual trajectories diverge along the attractor, so does the uncertainty associated with them. This uncertainty diverges in a highly non-Gaussian fashion when such uncertainties are not infinitesimal (see Figure 1.2).

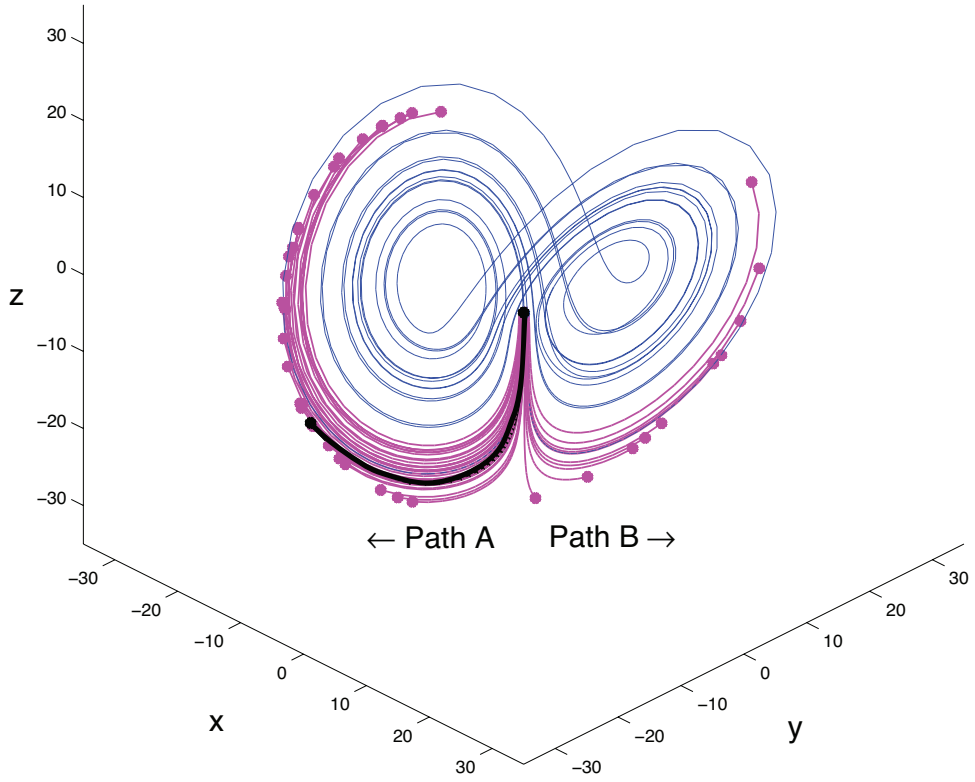

Figure 1.2: Non-Gaussian uncertainty propagation in the Lorenz system. The black point in the center shows a typical point located in a sensitive area of this chaotic system's attractor in phase space, representing a current estimate of the state. The thick black line represents the evolution in time of the trajectory from this estimate. If the uncertainty of the estimate is modeled as a very small cloud of points, centered at the original estimate with an initially Gaussian distribution, then the additional magenta lines show the evolution of each of these perturbed points in time. A Gaussian model of the resulting distribution of points is, clearly, completely invalid.

Estimation techniques that attempt to propagate probability distributions under linear, Gaussian assumptions fail to capture the true uncertainty of the estimate in such settings, and thus improved estimation techniques are required. The Ensemble Kalman Filter, in contrast, accounts properly for the nonlinearities of the chaotic system when propagating estimator uncertainty. This idea is a central component of the hybrid ensemble/variational method proposed in the present work, and is thus reviewed next.

1.2.3 Ensemble Kalman Filtering

The Ensemble Kalman Filter (EnKF) is a sequential data assimilation method useful for nonlinear multiscale systems with substantial uncertainties. In practice, it has been shown repeatedly to provide significantly improved state estimates in systems for which the traditional EKF breaks down. Unlike in the KF and EKF, the statistics of the estimation error in the EnKF are not propagated via a covariance matrix, but rather are approximated implicitly via the appropriate nonlinear propagation of several perturbed trajectories ("ensemble members") centered about the ensemble mean, as illustrated in Figure 1.2. The collection of these ensemble members (itself called the "ensemble"), propagates the statistics of the estimation error exactly in the limit of an infinite number of ensemble members. Realistic approximations arise when the number of ensemble members, N, is (necessarily) finite. Even with a finite ensemble, the propagation of the statistics is

still consistent with the nonlinear nature of the model. Conversely, the EKF propagates only the lowest-order components of the second-moment statistics about some assumed trajectory of the system. This difference is a primary strength of the EnKF.

In practice, the ensemble members $\hat{\mathbf{x}}^j$ in the EnKF are initialized with some known statistics about an initial mean estimate $\bar{\mathbf{x}}$. The ensemble members are propagated forward in time using the fully nonlinear model equation (1.1a), incorporating random forcing $\mathbf{w}^j(t)$ with statistics consistent with those of the actual state disturbances $\mathbf{w}(t)$ [see (1.3)]:

$$\frac{d\hat{\mathbf{x}}^j(t)}{dt} = f(\hat{\mathbf{x}}^j(t), \mathbf{w}^j(t)). \tag{1.6}$$

At the time t_k (for integer k), an observation $\mathbf{y}_k$ is taken [see (1.1b)]. Each ensemble member is updated using this observation, incorporating random forcing $\mathbf{v}_k^j$ with statistics consistent with those of the actual measurement noise, $\mathbf{v}_k$ [see (1.2)]:

$$\mathbf{d}_k^j = \mathbf{y}_k + \mathbf{v}_k^j. \tag{1.7}$$

Given this perturbed observation $\mathbf{d}_k^j$, each ensemble member is updated in a manner consistent[6] with the KF and EKF:

$$\hat{\mathbf{x}}^j_{k|k} = \hat{\mathbf{x}}^j_{k|k-1} + P^e_{k|k-1} H^T (H P^e_{k|k-1} H^T + R)^{-1} (\mathbf{d}_k^j - H\hat{\mathbf{x}}^j_{k|k-1}), \tag{1.8}$$

[6]Note that some authors (see, e.g., [20]) prefer to replace R in (1.8) with R^e, where

$$R^e = \frac{(V_k)\,(V_k)^T}{N-1} \quad \text{and} \quad V_k = \begin{bmatrix} \mathbf{v}_k^1 & \mathbf{v}_k^2 & \cdots & \mathbf{v}_k^N \end{bmatrix}.$$

Our current research has not revealed any clear advantage for using this more computationally expensive form.

Unlike the EKF, in which the entire covariance matrix P is propagated using the appropriate Riccati equation, the EnKF estimate covariance P^e is computed "on the fly" using the second moment of the ensembles from the ensemble mean:

$$P^e = \frac{(\delta\widehat{X})\ (\delta\widehat{X})^T}{N-1}, \text{ where } \delta\widehat{X} = \left[\delta\hat{\mathbf{x}}^1 \quad \delta\hat{\mathbf{x}}^2 \quad \cdots \quad \delta\hat{\mathbf{x}}^N\right],$$
$$\delta\hat{\mathbf{x}}^j = \hat{\mathbf{x}}^j - \bar{\mathbf{x}}, \quad \text{and} \quad \bar{\mathbf{x}} = \frac{1}{N}\sum_j \hat{\mathbf{x}}^j, \tag{1.9}$$

where N is the number of ensemble members, and the time subscripts have been dropped for notational clarity[7].

Thus, like the KF and EKF, the EnKF is propagated with a forecast step (1.6) and an update step (1.8). The ensemble members $\hat{\mathbf{x}}^j(t)$ are propagated forward in time using the system equations [with state disturbances $\mathbf{w}^j(t)$] until a new measurement $\mathbf{y}_k$ is obtained, then each ensemble member $\hat{\mathbf{x}}^j(t_k) = \hat{\mathbf{x}}_k^j$ is updated to include this new information [with measurement noise $\mathbf{v}_k^j$]. The covariance matrix is not propagated explicitly, as its evolution is implicitly represented by the evolution of the ensemble itself.

It is convenient to think of the various estimates during such a data assimilation procedure in terms of the set of measurements that have been included to obtain that estimate. Just as it is possible to propagate the ensemble members forward in time accounting for new measurements, ensemble members can also be propagated *backward* in time, either retaining the effect of each measurement or

[7]Note also that the factor $N-1$ (instead of N) is used in (1.9) to obtain an unbiased estimate of the covariance matrix [see [7]].

subtracting this information back off. In the case of a linear system, the former approach is equivalent to the Kalman smoother, while the later approach simply retraces the forward march of the Kalman filter backward in time. In order to make this distinction clear, the notation $\widehat{X}_{j|k}$ will represent the estimate ensemble at time t_j given measurements up to and including time t_k. Similarly, $\bar{\mathbf{x}}_{j|k}$ will represent the corresponding ensemble mean; that is, the average of the ensemble and the "highest-likelihood" estimate of the system.

While the EnKF significantly outperforms the more traditional EKF for chaotic systems, further approximations need to be made for multiscale systems such as atmospheric models. When assimilating data for 3D PDEs, the discretized state dimension n is many orders of magnitude larger than the number of ensemble members N that is computationally feasible (i.e., $N \ll n$). The consequences of this are twofold. First, the ensemble covariance matrix P^e is guaranteed to be singular, which can lead to difficulty when trying to solve linear systems constructed with this matrix. Second, this singularity combined with an insufficient statistical sample size produces directions in phase space in which no information is gained through the assimilation. This leads to spurious correlations in the covariance that would cause improper updates across the domain of the system. This problem can be significantly diminished via the ad hoc method of "covariance localization" mentioned previously, which artificially suppresses these spurious correlations using a distance-dependent damping function.

1.2.4 Ensemble Kalman Smoother (EnKS)

The Ensemble Kalman Smoother (EnKS) is built upon the theoretical foundations of the EnKF. The key difference lies in its ability to update past estimates based on future observations. Thus, we end up with smoothed estimates $\hat{\mathbf{x}}^j_{p|k}$, where p is not necessarily less than k. Given a new observation $\mathbf{y}_k$ at time t_k and forecasted ensemble $\hat{\mathbf{x}}^j_{k|k-1}$ at that time, the smoothed estimate $\hat{\mathbf{x}}^j_{p|k}$ is given by the following update equation:

$$\hat{\mathbf{x}}^j_{p|k} = \hat{\mathbf{x}}^j_{p|k-1} + S^e_{k-1} H^T (H P^e_{k|k-1} H^T + R)^{-1} (\mathbf{d}^j_k - H\hat{\mathbf{x}}^j_{k|k-1}), \tag{1.10a}$$

where S^e_{k-1} is the time covariance matrix between the estimate at the observation time t_k and the estimate at the smoothing time t_p, which is given by

$$S^e_{k-1} = \frac{(\delta\widehat{X}_{p|k-1})\ (\delta\widehat{X}_{k|k-1})^T}{N-1}, \tag{1.10b}$$

with the definitions for $\delta\widehat{X}$ given in (1.9). Note that, when $t_p = t_k$, the time covariance matrix S^e_{k-1} reduces to the standard covariance matrix $P^e_{k|k-1}$, and thus the EnKS update (1.10a) reduces appropriately to the standard EnKF update (1.8). This highlights an important property of the EnKS: even for highly chaotic, nonlinear systems, the EnKS provides the same estimate at the most recent measurement as the EnKF (in the limit of an infinite number of ensemble members). This result is expected in the linear setting, but is a major shortcoming of the

EnKS when applied to the typical nonlinear systems. This shortcoming is rectified by the hybrid method presented in Section 1.4.

1.2.5 Variational Methods

For high-dimensional systems in which matrix-based methods are computationally infeasible, vector-based variational methods are preferred for data assimilation. 3DVar is a vector-based equivalent to the KF. In both 3DVar and KF, the cost function being minimized is a (quadratic) weighted combination of the uncertainty in the background term and the uncertainty in the new measurement. If the system is linear, the optimal update to the state estimate can be found analytically, though this solution requires matrix-based arithmetic (specifically, the propagation of a Riccati equation), and is the origin of the optimal update gain matrix for the KF. When this matrix is too large for direct computation, a local gradient can instead be found using vector-based arithmetic only; 3DVar uses this local gradient information to determine the optimal update iteratively.

Similarly, 4DVar is the vector-based equivalent to the Kalman Smoother. In 4DVar, a finite time window (or "batch process") of a history of measurements is analyzed together to improve the estimate of the system at one edge of this window (and, thus, the corresponding trajectory of the estimate over the entire window). Unlike sequential methods, a smoother uses all available data over this finite time window to optimize the estimates of the system. This has the con-

sequence of refining past estimates of the system based on future measurements, whereas with sequential methods any given estimate is only conditioned on previous observations.

For analysis, let the variational window be defined on $t \in (\,0, T\,]$. Additionally, let there be K measurements in this interval, with measurement indices given by the set

$$M = \{\, k \mid t_k \in (\,0, T\,]\,\} \quad \Rightarrow \quad M = \{\,1\,, 2\,,\ \cdots\ ,\, K\,\}. \tag{1.11}$$

Without loss of generality, it will be assumed that there are measurements at the right edge of the window (at $t_{_K} = T$), but not at the left (at $t_0 = 0$). Then, the cost function $J(\mathbf{u})$ that 4DVar minimizes (with respect to $\mathbf{u}$) is defined as follows:

$$J(\mathbf{u}) = \frac{1}{2}\,(\,\mathbf{u} - \bar{\mathbf{x}}_{_{0|0}})^T\, P_{_{0|0}}^{-1}\,(\,\mathbf{u} - \bar{\mathbf{x}}_{_{0|0}}) + \frac{1}{2}\,\sum_{k=1}^{K}\big(\,\mathbf{y}_k - H\,\tilde{\mathbf{x}}_k\,\big)^T\, R^{-1}\,\big(\,\mathbf{y}_k - H\,\tilde{\mathbf{x}}_k\,\big), \tag{1.12}$$

where the optimization variable $\mathbf{u}$ is the initial condition on the refined state estimate $\tilde{\mathbf{x}}$ on the interval $t \in (\,0, T\,]$; that is,

$$\frac{d\tilde{\mathbf{x}}(t)}{dt} = f(\tilde{\mathbf{x}}(t), 0), \tag{1.13a}$$

$$\tilde{\mathbf{x}}_{_0} = \mathbf{u}. \tag{1.13b}$$

The first term in the cost function (1.12), known as the "background" term, summarizes the fit of $\mathbf{u}$ with the current probability distribution before the optimization (i.e., the effect of all past measurement updates). Like with the KF, $\bar{\mathbf{x}}_{_{0|0}}$ is

the estimate at time t_0 not including any of the new measurements in the window, and the covariance $P_{0|0}$ quantifies the second moment of the uncertainty in that estimate. Assuming an apriori Gaussian probability distribution of this uncertainty, the background mean and covariance exactly describe this distribution. The second term in the cost function (1.12) summarizes the misfit between the estimated system trajectory and the observations within the variational window. Thus, the solution $\mathbf{u}$ to this optimization problem is the estimate that best "fits" the observations over the variational window while also accounting for the existing information from observations prior to the variational window.

In practice, a 4DVar iteration is usually initialized with the background mean, $\mathbf{u} = \bar{\mathbf{x}}_{0|0}$. Given this initial guess for $\mathbf{u}$, the trajectory $\tilde{\mathbf{x}}(t)$ may be found using the full nonlinear equations for the system (1.13). To find the gradient of the cost function (1.12), consider a small perturbation of the optimization variable, $\mathbf{u} \leftarrow \mathbf{u} + \mathbf{u}'$, and the resulting perturbed trajectory, $\tilde{\mathbf{x}}(t) \leftarrow \tilde{\mathbf{x}}(t) + \tilde{\mathbf{x}}'(t)$, and perturbed cost function, $J(\mathbf{u}) \leftarrow J(\mathbf{u}) + J'(\mathbf{u}')$. The local gradient of (1.12), $\nabla J(\mathbf{u})$, is defined here as the sensitivity of the perturbed cost function $J'(\mathbf{u}')$ to the perturbed optimization variable $\mathbf{u}'$:

$$J'(\mathbf{u}') = \left[\nabla J(\mathbf{u})\right]^T \mathbf{u}'. \tag{1.14}$$

The derivation included in the Appendix illustrates how to write $J'(\mathbf{u}')$ in this simple form, leveraging the definition of an appropriate adjoint field $\mathbf{r}(t)$ on $t \in$

$(\,0, T\,]$, providing the following gradient:

$$\nabla J(\mathbf{u}) = P_{0|0}^{-1}\,(\,\mathbf{u} - \bar{\mathbf{x}}_{0|0}\,) - \mathbf{r}_{0}. \tag{1.15}$$

The resulting gradient can then be used iteratively to update the current estimate via a suitable minimization algorithm (steepest descent, conjugate gradient, limited-memory BFGS, etc.).

Being vector based makes 4DVar well suited for multiscale problems, and as a result is currently used extensively by the weather forecasting community. However, it has several key disadvantages. Most significantly, upon convergence, the algorithm provides an updated mean estimate $\bar{\mathbf{x}}_{0|K}$, but provides no clear formula for computing the updated estimate uncertainty covariance or its inverse, $P_{0|K}^{-1}$. That is, the statistical distribution of the estimate probability is not contained in the output of a traditional 4DVar algorithm. It can be shown that, upon full convergence for a linear system, the resulting analysis covariance $P_{0|K}$ is simply the Hessian of the original cost function (1.12) [see, e.g., [10]]. However, this is merely an analytical curiosity; computing the analysis covariance in this fashion requires as much matrix algebra as would be required to propagate a sequential filter through the entire variational window, defeating the purpose of the vector-based method.

Additionally, as posed above, the width of the variational window is fixed in the traditional 4DVar formulation. Thus, the cost function and associated n-

dimensional minimization surface are also constant throughout the iterations. For nonlinear systems, especially chaotic systems, this makes traditional 4DVar extremely sensitive to initial conditions. Because of the chaotic nature of these systems, the optimization surface, especially if T is large, is highly irregular and nonconvex (that is, fraught with local minima). The gradient-based algorithms associated with 4DVar are only guaranteed to converge to local minima. Thus, if the initial background estimate is located in the region of attraction of one of these local minima, the solution of the 4DVar algorithm will tend to converge to a suboptimal estimate.

1.3 Ensemble Variation Methods

As pointed out in Section 1.2.5, one of the major weaknesses of the standard variational assimilation schemes is the inability of these methods to update the higher moment estimate statistics. Given both a background mean and covariance, 3DVar and 4DVar simply return an updated mean; the corresponding updated covariance has thus far only been found via computationally involved Hessian analysis [10] or schemes coupled with a Kalman-like covariance propagation [61]. Here, we lay out the mathematical foundations for a consistent class of variational methods that, much like the Ensemble Kalman methods, use a finite cloud of points to represent implicitly both the background and analysis estimate

statistics. Unlike the Ensemble Kalman methods, this new class of Ensemble Variation methods uses an *ensemble of variational problems* to solve iteratively the complete data assimilation via the minimization of an appropriately defined cost function. It is shown that, under the standard assumptions of linear dynamics and Gaussian noise and disturbances, these Ensemble Variation methods reduce to the well-known optimal results of the standard Kalman Filter and Kalman Smoother.

1.3.1 Ensemble 3D Variational Assimilation (En3DVar)

Given a measurement $\mathbf{y}_k$ at time t_k, we will represent our estimate statistics with a finite ensemble of N members such that the sample mean and sample covariance are consistent (in the limit as $N \to \infty$) with the (assumed) known background mean and covariance. Thus, we have a collection of ensemble members $\hat{\mathbf{x}}^j_{k|k-1}$ conditioned on all prior observations $\{\ \mathbf{y}_p \mid p < k\ \}$ that build a sample covariance given by $P^e_{k|k-1}$. With this, we can define an En3DVar component cost function for each ensemble member as:

$$\begin{aligned} J_j(\mathbf{u}^j) = &\frac{1}{2}\,(\mathbf{u}^j - \hat{\mathbf{x}}^j_{k|k-1})^T\,(P^e_{k|k-1})^{-1}\,(\mathbf{u}^j - \hat{\mathbf{x}}^j_{k|k-1}) \\ &+ \frac{1}{2}\,(\mathbf{d}^j_k - H\mathbf{u}^j)^T\,R^{-1}\,(\mathbf{d}^j_k - H\mathbf{u}^j), \end{aligned} \tag{1.16}$$

where the control variable $\mathbf{u}^j$ for each ensemble member is (at the minimum) the updated estimate $\hat{\mathbf{x}}^j_{k|k}$, now conditioned on the new measurement $\mathbf{y}_k$. As with the EnKF and EnKS, each ensemble member is assimilated with additional noise

added onto the (already noisy) measurement, i.e.,

$$\mathbf{d}_k^j = \mathbf{y}_k + \mathbf{v}_k^j. \tag{1.17}$$

The total cost function J is given as the sum of the component cost functions J_j for each ensemble member j. Because the component cost functions are only coupled through the specified (and fixed) background ensemble members $\hat{\mathbf{x}}^j_{k|k}$ and the covariance matrix which they approximate, $P^e_{k|k-1}$, each J_j can be minimized *independently*, creating an optimization problem that is trivial to parallelize on modern high performance computing hardware. Similar to traditional 3DVar, each component cost function is minimized by finding the local gradient and then using a suitable descent algorithm; again, these component-wise minimizations are completely decoupled from one ensemble member to the next.

In summary, En3DVar is performed at a given time t_k by assimilating an ensemble of 3DVar problems, one for each ensemble member. Each individual component 3DVar problem is uniquely characterized by its own perturbed background state $\hat{\mathbf{x}}^j_{k|k-1}$ and its own perturbed measurement $\mathbf{d}_k^j$. The component cost functions are coupled through the background ensemble covariance matrix $P^e_{k|k-1}$, the measurement noise covariance matrix R, and the original, unperturbed (but still noisy) measurement $\mathbf{y}_k$. It is shown in the following section that the unique solution to this problem (in the limit as $N \to \infty$) is a new ensemble with corresponding sample statistics (mean and covariance) that are consistent with the

well-known optimal Kalman results.

Theorem 1 (Equivalence of En3DVar to the Kalman Filter)**.** *In the limit of an infinite number of ensemble members (i.e., $N \to \infty$), the En3DVar problem defined above converges to the equivalent Kalman filter solution.*

Proof. Because each component cost function is minimized independently, we will examine the unique solution of just one for the purpose of this proof. Note that (1.16) is convex in $\mathbf{u}^j$. The gradient of the j^{th} component cost function with respect to the initial state $\mathbf{u}^j$ is given by

$$\nabla J_j = (P^e_{k|k-1})^{-1} (\mathbf{u}^j - \hat{\mathbf{x}}^j_{k|k-1}) - H^T R^{-1} (\mathbf{d}^j_k - H\mathbf{u}^j). \tag{1.18}$$

Typically, the cost function is minimized iteratively via a gradient descent method, but for the purpose of analysis here, we can find the minimum directly by setting the $\nabla J_j = 0$ and solving for the updated estimate $\mathbf{u}^j = \hat{\mathbf{x}}^j_{k|k}$ at the minimum:

$$0 = (P^e_{k|k-1})^{-1} (\hat{\mathbf{x}}^j_{k|k} - \hat{\mathbf{x}}^j_{k|k-1}) - H^T R^{-1} (\mathbf{d}^j_k - H\hat{\mathbf{x}}^j_{k|k}) \tag{1.19a}$$

$$\begin{aligned} 0 &= (P^e_{k|k-1})^{-1} (\hat{\mathbf{x}}^j_{k|k} - \hat{\mathbf{x}}^j_{k|k-1}) + H^T R^{-1} H (\hat{\mathbf{x}}^j_{k|k} - \hat{\mathbf{x}}^j_{k|k-1}) \\ &- H^T R^{-1} (\mathbf{d}^j_k - H\hat{\mathbf{x}}^j_{k|k-1}) \end{aligned} \tag{1.19b}$$

$$(\hat{\mathbf{x}}^j_{k|k} - \hat{\mathbf{x}}^j_{k|k-1}) = \left[(P^e_{k|k-1})^{-1} + H^T R^{-1} H \right]^{-1} H^T R^{-1} (\mathbf{d}^j_k - H\hat{\mathbf{x}}^j_{k|k-1}) \tag{1.19c}$$

Assuming that all inverses indicated exist, the identity

$$\left[(P^e_{k|k-1})^{-1} + H^T R^{-1} H \right]^{-1} H^T R^{-1} = P^e_{k|k-1} H^T \left[H P^e_{k|k-1} H^T + R \right]^{-1} \quad (1.19d)$$

can be substituted into (1.19c) to get the form

$$\hat{\mathbf{x}}^j_{k|k} = \hat{\mathbf{x}}^j_{k|k-1} + P^e_{k|k-1} H^T \left[H P^e_{k|k-1} H^T + R \right]^{-1} (\mathbf{d}^j_k - H\hat{\mathbf{x}}^j_{k|k-1}), \quad (1.20a)$$

$$K \equiv P^e_{k|k-1} H^T \left[H P^e_{k|k-1} H^T + R \right]^{-1}, \quad (1.20b)$$

$$\hat{\mathbf{x}}^j_{k|k} = \hat{\mathbf{x}}^j_{k|k-1} + K (\mathbf{d}^j_k - H\hat{\mathbf{x}}^j_{k|k-1}). \quad (1.20c)$$

Recall that (1.20c) is the unique solution for the j^{th} ensemble member. Thus, we can think of the ensemble of solutions $\hat{\mathbf{x}}^j_{k|k}$ as a random variable that is itself conditioned on two other random variables, $\hat{\mathbf{x}}^j_{k|k-1}$ and $\mathbf{d}^j_k$. Note that the gain matrix K is identical to that of the Kalman Filter. To see the rest of the equivalence with the Kalman Filter, we take the sample mean of the result.

$$\begin{aligned} \bar{\mathbf{x}}_{k|k} &= \frac{1}{N} \sum_{j=1}^{N} \hat{\mathbf{x}}^j_{k|k} \\ &= \frac{1}{N} \sum_{j=1}^{N} \hat{\mathbf{x}}^j_{k|k-1} + K \left(\frac{1}{N} \sum_{j=1}^{N} \mathbf{d}^j_k - H \frac{1}{N} \sum_{j=1}^{N} \hat{\mathbf{x}}^j_{k|k-1} \right) \\ &= \bar{\mathbf{x}}_{k|k-1} + K (\mathbf{y}_k - H \bar{\mathbf{x}}_{k|k-1}) \end{aligned} \quad (1.21)$$

Although we obtain the Kalman update (1.21) for the estimate mean by using En3DVar, it is important to note that the traditional 3DVar algorithm (involving only an iterative update of the mean) would also have provided us with this result. The real strength of En3Dvar lies in its ability to also implicitly update the estimate

covariance, something that was not possible with traditional 3DVar. To see this equivalence, we take the sample covariance of the updated ensemble $\hat{\mathbf{x}}^j_{k|k}$,

$$P^e_{k|k} = \frac{1}{N-1}\sum_{j=1}^{N} (\,\hat{\mathbf{x}}^j_{k|k} - \bar{\mathbf{x}}_{k|k}\,)\,(\,\hat{\mathbf{x}}^j_{k|k} - \bar{\mathbf{x}}_{k|k}\,)^T. \tag{1.22a}$$

Substituting in the definitions of $\hat{\mathbf{x}}^j_{k|k}$ from (1.20c) and $\bar{\mathbf{x}}_{k|k}$ from (1.21) and simplifying, we get

$$\begin{aligned} P^e_{k|k} &= (I - KH)\,P^e_{k|k-1}(I - KH)^T + K\,R^e\,K^T + \Phi + \Phi^T, \\ \Phi &= \frac{1}{N-1}\sum_{j=1}^{N}\left\{(\,I - K\,H\,)\,(\,\hat{\mathbf{x}}^j_{k|k-1} - \bar{\mathbf{x}}_{k|k-1})(\,\mathbf{d}^j_k - \mathbf{y}_k)^T\,K^T\right\}. \end{aligned} \tag{1.22b}$$

The final terms $\Phi + \Phi^T$ in (1.22b) arise from spurious correlations between the background error and the measurement noise. In a similar manner to the EnKF, as the number of ensembles increase, these terms disappear, leaving the expected Kalman Filter covariance update equation, i.e.,

$$\lim_{N\to\infty} \Phi = 0, \tag{1.23a}$$

$$P^e_{k|k} = (\,I - K\,H\,)\,P^e_{k|k-1}(\,I - K\,H\,)^T + K\,R^e\,K^T. \tag{1.23b}$$

Thus, we have shown that, by iteratively assimilating an ensemble of 3DVar problems with both perturbed background states and perturbed measurements, we are able to compute both the analysis mean *and* covariance. This algorithm is both tractable for high dimensional systems (in the sense that it is vector-based) and very easily parallelized (in the sense that each individual problem can be solved independently). □

1.3.2 Ensemble 4D Variational Assimilation (En4DVar)

Given a time history of measurements $\{\, \mathbf{y}_k \mid t_k \in (\,0, T\,]\,\}$, as with the En3DVar case, we will represent our estimate statistics with a finite ensemble of N members such that the sample mean and sample covariance are consistent (in the limit as $N \to \infty$) with the (assumed) known background mean and covariance at the left edge of the time window t_0. We can then define an analogous En4DVar cost function over the window, for the j^{th} ensemble member, that balances the misfit between a set of perturbed observations and the deviation from a perturbed background initial condition as follows:

$$\begin{aligned} J_j(\mathbf{u}^j) = &\frac{1}{2}\,(\mathbf{u}^j - \hat{\mathbf{x}}^j_{0|0})^T\,(P^e_{0|0})^{-1}\,(\mathbf{u}^j - \hat{\mathbf{x}}^j_{0|0}) \\ &+ \frac{1}{2}\sum_{k=1}^{K}(\mathbf{d}^j_k - H\,\tilde{\mathbf{x}}^j_k)^T\,R^{-1}\,(\mathbf{d}^j_k - H\,\tilde{\mathbf{x}}^j_k). \end{aligned} \tag{1.24}$$

Much like traditional 4DVar, each ensemble member is constrained over the window by the model, and the control variable $\mathbf{u}^j$ serves as the initial condition for its trajectory.

$$\frac{d\tilde{\mathbf{x}}^j(t)}{dt} = f(\tilde{\mathbf{x}}^j(t), 0), \tag{1.25a}$$

$$\tilde{\mathbf{x}}^j_0 = \mathbf{u}^j. \tag{1.25b}$$

Each initial ensemble member $\hat{\mathbf{x}}^j_{0|0}$ acts as its own perturbed background, and each ensemble member is assimilated with its own set of perturbed measurements $\{\, \mathbf{d}^j_k = \mathbf{y}_k + \mathbf{v}^j_k \mid t_k \in (\,0, T\,]\,\}$.

The total cost function J is given as the sum of the component cost functions for each ensemble member. Because the component cost functions are only coupled through the specified (and fixed) background ensemble members $\hat{\mathbf{x}}^j_{0|0}$ and the covariance matrix which they approximate, $P^e_{0|0}$, each J_j can be minimized *independently*, creating an embarrassingly parallel optimization problem. Similar to traditional 4DVar, each cost function is minimized by finding the local gradient and using a suitable descent algorithm. Finding the gradient of (1.24) requires the use of an appropriately-defined adjoint field. The derivation parallels that of standard 4DVar (as illustrated in the Appendix), and gives the j^{th} gradient as:

$$\nabla J_j(\mathbf{u}^j) = (P^e_{0|0})^{-1}\,(\,\mathbf{u}^j - \hat{\mathbf{x}}^j_{0|0}\,) - \mathbf{r}^j_0, \tag{1.26}$$

where $\mathbf{r}^j_0$ is the initial condition of the j^{th} adjoint field found via a background march from $t_{_K}$ to t_0 of the adjoint equations. Thus each iteration of En4DVar requires a forward march of the ensemble through the optimization window followed by a backward march of an *ensemble of adjoints* to find each component gradient. The decoupled nature of these marches is what facilitates the efficient parallel global solution.

Due to the fact that En4DVar accounts for all observations within the assimilation window, it is, by nature, a smoother. Upon completion of the minimization, we are provided with a new ensemble of points $\hat{\mathbf{x}}^j_{0|K}$, conditioned on these measurements. From this ensemble, statistical measures such as the sample mean and

covariance can be extracted.

Theorem 2 (Equivalence of En4DVar to the Kalman Smoother)**.** *In the limit of an infinite number of ensemble members (i.e. $N \to \infty$) and under the assumptions of linear dynamics and Gaussian noise and disturbances, the En4DVar problem defined above converges to the equivalent Kalman smoother solution.*

Proof. The proof is straightforward and follows directly from that of Theorem 1, however, due to the addition of the time dynamics, it tends to become notationally cumbersome. In the interest of brevity, we have elected to omit it here. □

1.4 Hybrid Ensemble Smoother (HEnS)

As was initially illustrated in Figure 1.1, we have identified two families of ensemble-based assimilation methods: the ensemble variational methods (consisting of En3DVar and En4DVar) and the ensemble Kalman methods (consisting of the EnKF and the EnKS). In theory, both families address the same problem. Further, as we have shown in the simplified case with linear dynamics and Gaussian uncertainties, they converge to the same solution. However, when the system is highly nonlinear, all bets on optimality of the solutions are off, and we do not necessarily expect each method to provide identical solutions.

In the case of nonlinear systems, one might thus wonder which method typically provides the best answer. The best answer, however, may in fact not come

from one individual method, but rather from a consistent hybrid of both methods. This is the idea behind the development of the Hybrid Ensemble Smoother (HEnS), which is a consistent hybrid between the two smoothers, En4DVar and EnKS.

A key motivation for HEnS is the iterative nature of the En4DVar method. Once the component cost functions (1.24) are defined appropriately using the known background ensemble, any initial condition $\mathbf{u}^j$ can be used to begin the gradient-based minimization. Typically, the best guess we have at the start of an iteration is the background itself (i.e., $\mathbf{u}^j = \hat{\mathbf{x}}^j_{0|0}$), because no other information is known. However, if we were to first run the EnKS through the entire window $(\,0,\,T\,]$, we would develop an intermediate smoothed estimate $\hat{\mathbf{x}}^j_{0|K}$. That is, the output of the EnKS at the left edge of the window is the best estimate at that time, given all measurements in the optimization window *as determined by the EnKS framework*. Again, under the appropriate assumptions, this smoothed estimate would be optimal, but, due to the nonlinear nature of the system and the necessarily finite ensemble size, the EnKS typically finds a suboptimal solution to the smoothing problem. Consequently, this intermediate smoothed estimate can then be used as the initial condition for the specified En4DVar minimization problem in lieu of the background state. If there is any more information to be extracted from the observations, this iterative minimization will attempt to do just that.

Theorem 3 (Consistency of HEnS)**.** *In the limit of an infinite number of ensemble members (i.e. $N \to \infty$), and under the assumptions of linear dynamics and*

Gaussian noise and disturbances, the HEnS formulation described above converges to both the Kalman smoother solution and the equivalent En4DVar solution.

Proof. Under the above assumptions, the En4DVar cost function is convex, and thus contains only one minimum. Provided the cost function is defined appropriately, the En4DVar iteration will converge to this global minimum, regardless of initial condition–even if the output from the EnKS is used to initialize the minimization, as done with HEnS. Therefore, HEnS will converge to the same solution as En4DVar under the assumptions stated. The proof of the equivalence to the Kalman smoother then follows immediately from Theorem 2. □

An important consequence of Theorem 3 is that the HEnS framework will do no worse than the EnKS solution alone.

In summary, HEnS is a consistent hybrid of both EnKS and En4DVar. Essentially, HEnS uses a (typically) suboptimal smoothed estimate from the EnKS to initialize an En4DVar minimization. Because the output from the EnKS is much closer to the expected minimum than the original background estimate, the En4DVar iteration is less likely to converge to spurious local minima, far from the optimal estimate, and thus produces more a more accurate estimate than either smoother would by itself. HEnS can be implemented in three straightforward steps:

1. Given a set of measurements $\left\{ \mathbf{y}_1, \cdots, \mathbf{y}_K \right\}$ on the window $[\,0, T\,]$ and a background ensemble $\hat{\mathbf{x}}^j_{0|0}$ at the left edge of the window, define the appropriate

En4DVar component cost functions.

2. March a fixed-point EnKS through the window, smoothing the estimate at the left edge of the window to produce an intermediate smoothed estimate $\hat{\mathbf{x}}^j_{0|K}$ conditioned on all observations in the window.

3. Use the intermediate smoothed estimate output from the EnKS as the initial condition $\mathbf{u}^j$ for an En4DVar minimization over the same window, using the previously-defined component cost functions. This will potentially provide a better smoothed estimate over the entire window for the full nonlinear system.

As with any assimilation strategy, the output of HEnS from a previous window can be used as the background estimate for a subsequent window, cycling the algorithm.

1.4.1 Advantages in forecasting

In data assimilation applications that involve forecasting, the most important estimate is always the most recent one. This is, of course, the estimate that is used as an initial condition for any open-loop forecast (into the future).

In the linear setting, the most recent filtered estimate is identical to the most recent smoothed estimate because they have both been conditioned on the same set of measurements (which are all necessarily in the past). It is for this

reason alone that smoothers have been largely ignored by the operational weather forecasting community. Up until now, their computational expense has not been justified by providing an improved estimate upon which a forecast could be made.

Although this conclusion is certainly true in the linear world, much information is to be gained, even at the most recent time, by revisiting old measurements in light of new data. Even in the EnKF setting, the suboptimal updates made at any given time are a function of the ensemble member trajectories used. If those trajectories were improved (say, through the use of a smoother), then it might be possible to perform more accurate updates, which in turn would increase the accuracy of the estimate at a future time.

Unfortunately, the formulation of the EnKS does not leverage this idea, and (in the limit of an infinite ensemble size) returns exactly the same estimate at the right edge of the window. HEnS, however, is not so constrained. By improving the smoothed estimate initial condition at the left edge of the window, HEnS also improves the estimate throughout the window, and specifically reduces the error in the most recent estimate, providing a more accurate long-term forecast.

1.4.2 Hybrid Ensemble Filter (HEnF)

It is worth noting that, in a manner analogous to the hybridization of the two smoothers (En4DVar and the EnKS), a hybrid filter can be defined by combining En3DVar and the EnKF. The resulting algorithm, appropriately dubbed

HEnF, would use the EnKF to precondition an En3DVar iteration. That is, at a given measurement time, the EnKF update would be used as an initial condition for the subsequent En3DVar minimization. However, because neither filter update necessarily takes into account the dynamics of the system, it is not apparent to the authors that such additional computation would provide a better solution. As a result, we have neglected to highlight the HEnF in this discussion.

1.5 Representative Example

The two primary new ideas described in this paper, En4DVar and HEnS, are now compared, via computational experiments, to both EnKF and EnKS. We have already shown analytically that, in the linear setting, all four methods provide consistent solutions; however, in a nonlinear setting with significant uncertainties, these different approaches provide substantially different results.

The Lorenz equation (see [44]) is used here as a simple model of a nonlinear system with self-sustained chaotic unsteadiness[8] in order to perform this comparison. The Lorenz equation is a system of three coupled, nonlinear ordinary

[8]That is, the system considered maintains its nonperiodic, finite, bounded unsteady motion with no externally-applied stochastic forcing.

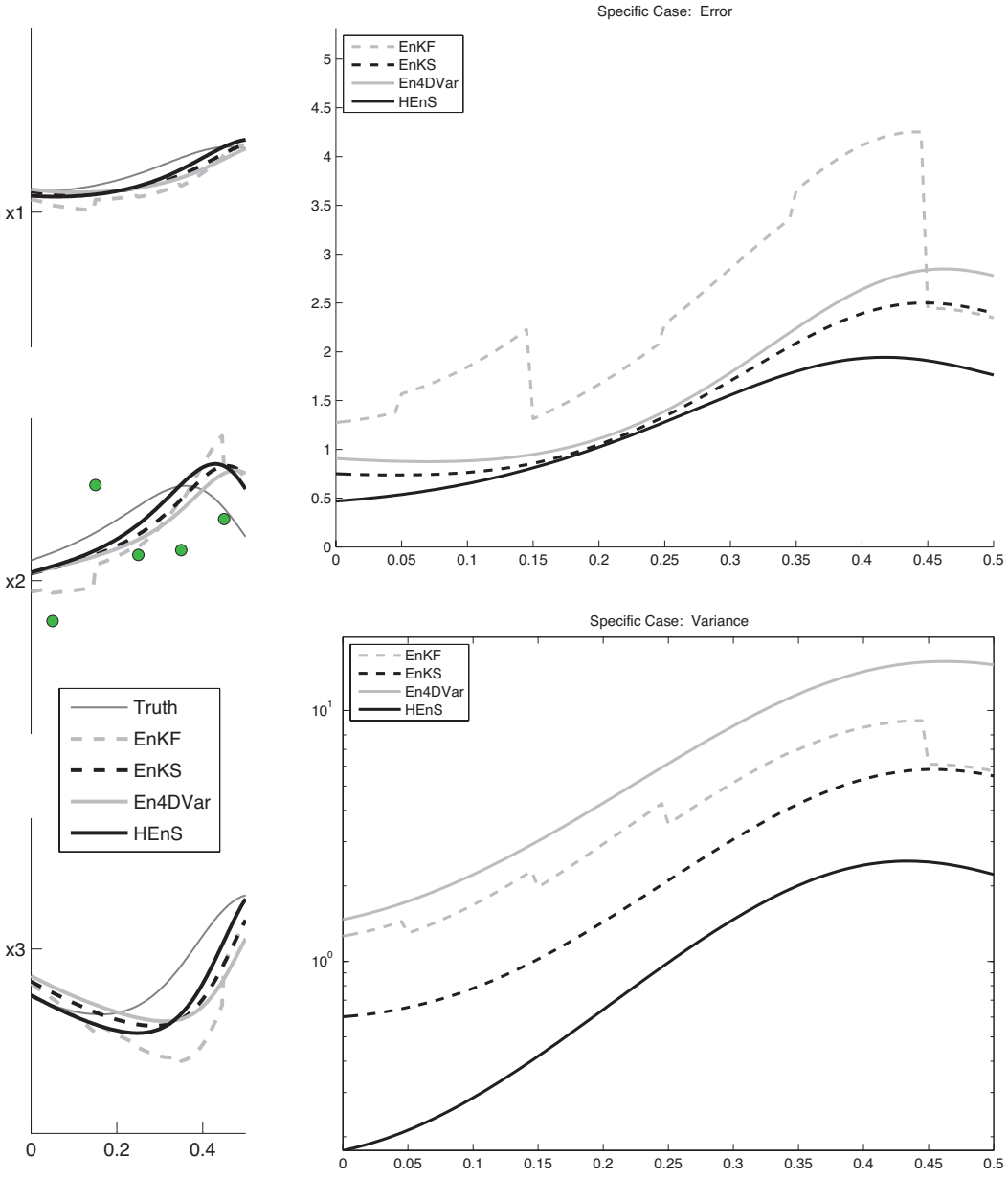

Figure 1.3: A typical run is shown for the Lorenz experiment. On the left, each assimilation method is compared to the truth trajectory via very noisy measurements of x_2. From this, the estimate error (top-right) and error energy (bottom-right) are calculated. Note that the EnKF and EnKS are not necessarily equivalent at the right edge of the window due to the finite ensemble size.

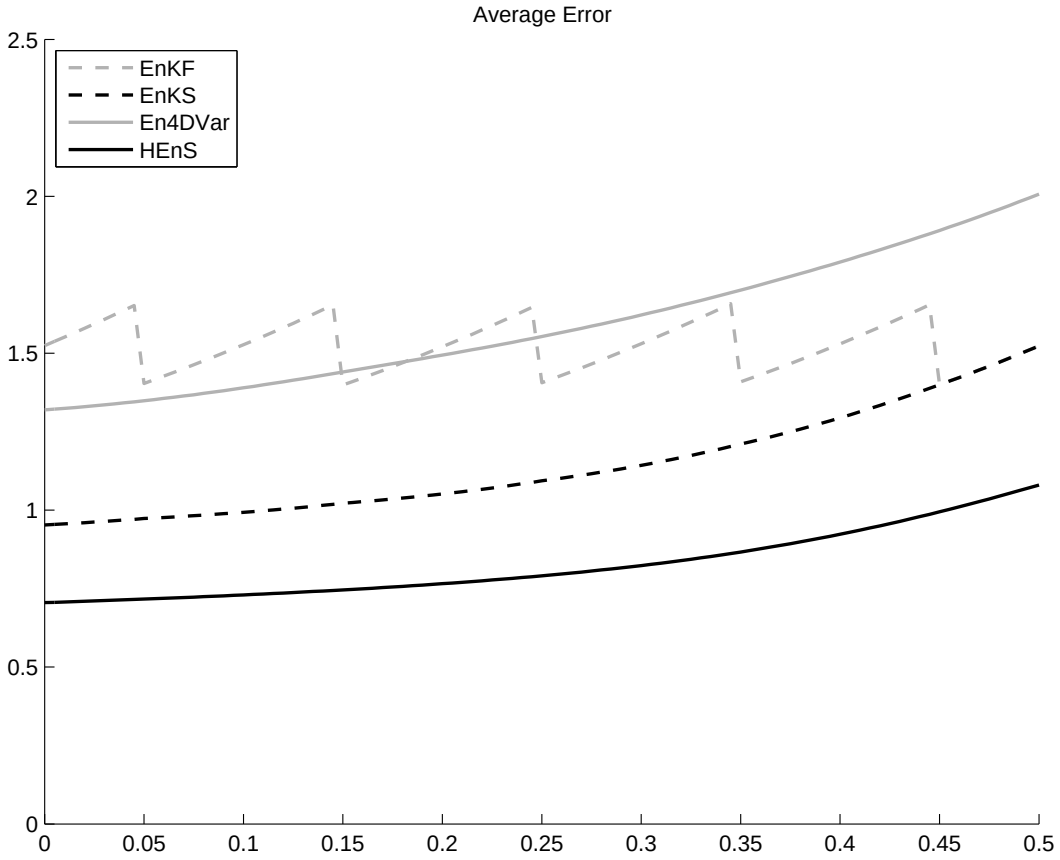

Figure 1.4: This is the converged error plot after several weeks of statistically averaging on a modern desktop computer (3 GHz Core Duo) for the experimental Lorenz test case on EnKF, EnKS, En4DVar, and HEnS. At the left edge of the window, we see that En4DVar decreases the error from the background, but convergence to local minima prevents it from competing with EnKS and HEnS. By initializing an En4DVar minimization with the output from the EnKS, we can see that, statistically, HEnS reduces the error by an additional 50%.

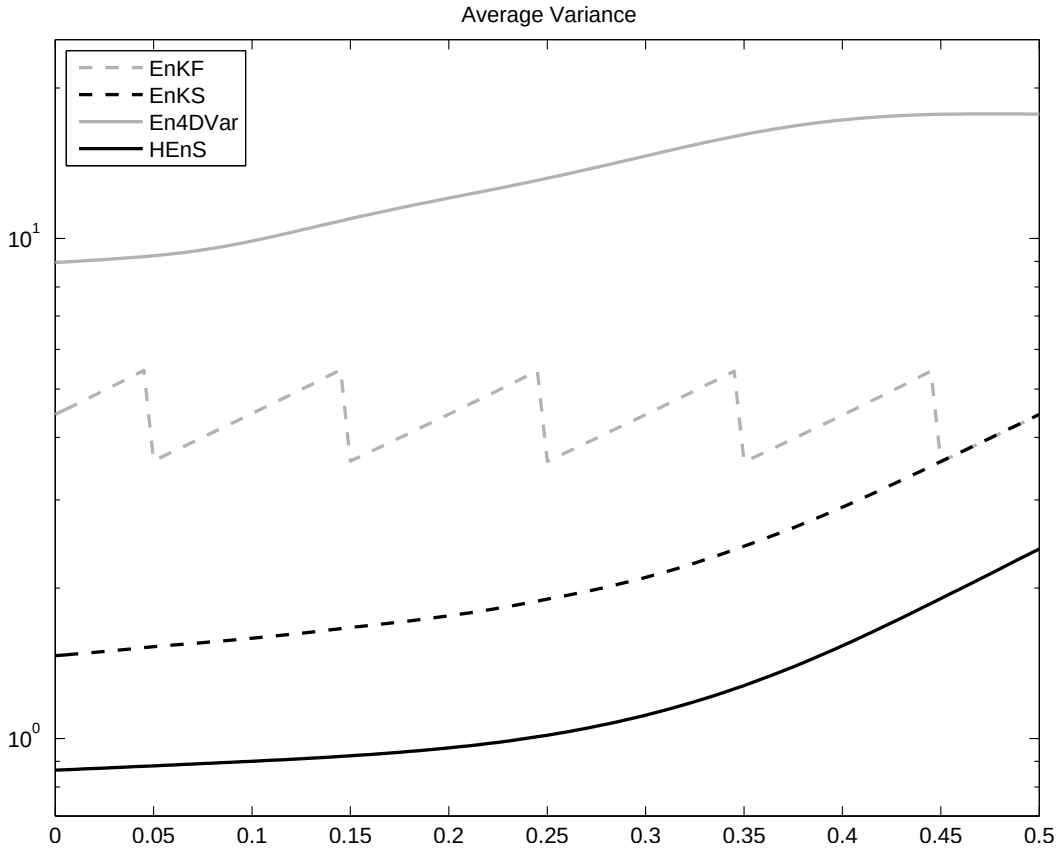

Figure 1.5: This is the converged variance plot for the experimental Lorenz test case on EnKF, EnKS, En4DVar, and HEnS. The trace of the covariance matrix for each method is plotted as a function of time (averaged over several runs). En4DVar on its own does not perform as well as the other methods. However, when En4DVar is combined with EnKS to make HEnS, a substantially improved time-averaged performance is realized.

differential equations given by:

$$\frac{d\mathbf{x}(t)}{dt} = \begin{pmatrix} \sigma\left(x_2 - x_1\right) \\ -x_2 - x_1 x_3 \\ -\beta x_3 + x_1 x_2 - \beta\phi \end{pmatrix}, \tag{1.27}$$

where σ, β, and ϕ are tunable parameters. Solutions of these equations approach a well-defined manifold or *attractor* of dimension slightly higher than two. Perturbed trajectories converge exponentially back onto the attractor, while adjacent trajectories diverge exponentially within the attractor, creating the familiar chaotic motion in the Lorenz system. Note that, for convenience, the system equations of (1.27) are transformed slightly from the traditional form such that the attractor is approximately centered at the origin.

In this comparison, we quantify the time-averaged statistics of the four assimilation methods under consideration. That is, we run a very large number of trials and calculate both the average error of the estimate as well as the average energy of the estimation error (a.k.a. the variance) over all of these trials. A typical run simulates a truth trajectory over a fixed window, taking noisy measurements at set intervals; then, each method (EnKF, EnKS, En4DVar, and HEnS) is used on the resulting data set, initialized with its own independent background. Appealing to the ergodicic nature of the Lorenz system, the output at the right edge of the window for each method (and the truth model) is then used as the input for the next run on the subsequent time window. Consequently, a series of assimilation

windows are evaluated on various intervals all over the attractor. The statistics of this process are then used to calculate the expected performance characteristics on this nonlinear chaotic system.

For the results shown, the model parameters used are $\sigma = 4$, $\beta = 1$, and $\phi = 48$. Only the second state x_2 is measured, and the measurement noise variance used $R = 5$ (which, as seen in Figure 1.3, is quite substantial). The assimilation window has width $T = 0.5$, and five observations are taken (at intervals of $\Delta t = 0.1$) centered in this window. The starting conditions for the truth and estimate ensemble backgrounds are not significant, as the converged statistics are not a function of the starting point used in the simulation. The nonlinear model is assumed to be perfect (that is, $Q = 0$), and thus any ensemble forecasting would be done with a simple evolution of the unforced equations from the most recent state estimate. All of the cases were run with $N = 300$ ensemble members, but similar results were found with significantly fewer ensemble members. A typical run is shown in Figure 1.3.

Statistical steady state was achieved via several weeks of statistical averaging on a modern desktop computer (3 GHz Core Duo). The converged statistics above are illustrated in Figures 1.4 and 1.5. One can see immediately that, on average, HEnS *significantly* outperforms the other three methods in terms of both accuracy (lower error) and precision (lower variance).

1.6 Conclusion

A new, hybrid method (dubbed the Hybrid Ensemble Smoother, or HEnS) for state estimation in nonlinear chaotic systems has been proposed. This new method is based on a new variational formulation of the ensemble smoother, dubbed En4DVar, initialized leveraging the (traditional, non-variational) formulation of the ensemble Kalman smoother (EnKS). In essence, a traditional EnKS is used to initialize the new variational formulation of the ensemble smoother. The method introduced are proven to be *consistent*, meaning that they all reduce to the Kalman filter under the appropriate simplifying assumptions (that is, linear system, Gaussian state disturbances and measurement noise, and a sufficiently large number of ensemble members). However, the proposed new estimation method, HEnS, is shown, on average, to *significantly* outperform existing methods in a representative estimation problem on a nonlinear chaotic system.

The reason for this remarkable success is that *HEnS provides an effective mechanism for revisiting past measurements in light of new data*, leveraging a smoother effectively to reinterpret past measurements based upon a more refined past state estimate, and thereby improving significantly the present state estimate. In essence, HEnS combines the powerful retrospective analysis of a variational method with the effective synthesis of the principle directions of uncertainty, as summarized by an ensemble-based method. An important ingredient

to the method's operational effectiveness is the initialization of the variational analysis with the solution from a EnKS computation, which is far better than initializing this variational analysis simply with an EnKF computation based on older measurements.

Note finally that the HEnS method is based solely on variational and ensemble Kalman components that are already used heavily for operational weather forecasting, and are applied routinely, in real time, to systems with state dimension larger than 10^6. Thus, HEnS naturally inherits this effective scalability to very large-scale chaotic systems, and holds significant promise to improve the accuracy of such estimation and forecasting efforts.

1.7 Acknowledgements

The authors gratefully acknowledge the generous financial support of the National Security Education Center (NSEC) at Los Alamos National Laboratory (LANL) and numerous helpful discussions with Chris Colburn (UCSD), David Zhang (UCSD), Dr. Frank Alexander (LANL), and Prof. Daniel Tartakovsky (UCSD).

This chapter is, in part, a reprint of the material as it will appear in:
Cessna, J. and Bewley, T. (2010) A Hybrid (variational / Kalman) ensemble smoother for the estimation of nonlinear high-dimensional discretizations of PDE

systems. *Under preparation, IEEE Transactions on Automatic Control*.

The dissertation author is the primary investigator and author of this publication.

Chapter 2

Numerical Implementation

An important property of the HEnS algorithm is its ability to scale well to high dimensional systems. Much like the standard EnKF, when dealing with multiscale systems, we can only afford to compute a small number of ensemble members relative to the state dimension, i.e. $N \ll n$. In these cases, the only tractable method of computing an analysis requires a scalable parallel implementation of the algorithm. Typically, each ensemble member will be placed on a subset of the available nodes in the computational cluster, and information will be shared through a suitable message passing routine, e.g. MPI. In the following sections, we illustrate some of the key numerical issues involved with the implementation of each of the HEnS components in an MPI setting.

2.1 EnKF Implementation

Much research has been done in the direction of implementing the EnKF (and consequently the EnKS) in these high dimensional situations. The key challenge for the ensemble Kalman methods is their dependence on state correlations found through sample statistics of the available ensemble. When the number of ensembles is necessarily small relative to the state dimension, these sample correlations are often poor and lead to filter divergence. This problem is averted through the ad hoc method of covariance localization, as is discussed in further detail in Section 1.1.3. Due to the ad hoc nature and wide range of such methods, we will defer the implementation details of such a filter to their respective authors. This section will focus specifically on the parallel implementation of the EnKF update equations–without localization–using the Message Passing Interface (MPI), allowing for uniform load distribution on, and minimal communication between, the massively parallel computational resources required to apply the EnVE algorithm to multiscale systems.

In general, the ensemble $\widehat{X}_{i|k}$ is comprised of $N \ll n$ ensemble members. Each of these ensemble members $\hat{\mathbf{x}}^j_{i|k}$ is located on its own processor (or processors) with a corresponding process number. In practice, for testing purposes, an additional process is also used for the "truth" model simulation, which is done in parallel with the EnKF march. Thus, the MPI environment is constructed of $N+1$

processes, with process j denoted by p^j. For convenience, the "truth" model is run on p^0, while each ensemble member $\hat{\mathbf{x}}^j_{i|k}$ is run on its corresponding process, p^j.

The EnKF consists of two main steps: a forward march of the ensemble to predict the estimate at the next measurement, and an appropriate update to the forecasted estimate due to each measurement. Recall that the discretized system of interest is given by (1.1a) and (1.1b). The forecasting step of the EnKF is the march from $\widehat{X}_{k-1|k-1}$ to $\widehat{X}_{k|k-1}$ (not including the measurement update). In the MPI setting, this is done by simply marching each ensemble member forward in time–using an appropriate time-stepping algorithm–according to the governing equation:

$$\frac{d\hat{\mathbf{x}}^j(t)}{dt} = f(\hat{\mathbf{x}}^j(t), \mathbf{w}(t)). \tag{2.1}$$

The disturbances $\mathbf{w}(t)$ are modeled appropriately using an appropriate random-number generator, and each ensemble member is disturbed independently from the other ensemble members. In an MPI setting, the computation time of each process is assumed independent from the other processes. Hence, the time required to propagate the N ensemble members in this framework is equivalent to a single simulation on a single processor.

Next, the measurement update at time t_k must be performed. To update the ensemble $\widehat{X}_{k|k-1}$ to reflect the newest measurement (thereby giving $\widehat{X}_{k|k}$), a

corresponding update must be done on each individual ensemble member as follows:

$$\hat{\mathbf{x}}^j_{k|k} = \hat{\mathbf{x}}^j_{k|k-1} + P^e_{k|k-1}\, H^H (\, H\, P^e_{k|k-1}\, H^H + R)^{-1} (\mathbf{d}^j_k - H\, \hat{\mathbf{x}}^j_{k|k-1}). \tag{2.2}$$

To evaluate this equation, the three main components of the update are first developed independently as:

$$\hat{\mathbf{x}}^j_{k|k} = \hat{\mathbf{x}}^j_{k|k-1} + L^{(1)}_k \left(L^{(2)}_k\right)^{-1} \mathbf{z}^j_k, \tag{2.3}$$

$$L^{(1)}_k = P^e_{k|k-1}\, H^H \qquad\qquad L^{(1)}_k \in \Re^{n\times m}, \tag{2.4}$$

$$L^{(2)}_k = H\, P^e_{k|k-1}\, H^H + R \qquad\qquad L^{(2)}_k \in \Re^{m\times m}, \tag{2.5}$$

$$\mathbf{z}^j_k = \mathbf{d}^j_k - H\, \hat{\mathbf{x}}^j_{k|k-1} \qquad\qquad \mathbf{z}^j_k \in \Re^m. \tag{2.6}$$

Note that the matrices $L^{(1)}_k$ and $L^{(2)}_k$ depend upon the entire ensemble.

First, examine the structure of $P^e_{k|k-1}$. This covariance is built up from the individual ensemble members such that:

$$\begin{aligned} P^e_{k|k-1} =& \frac{1}{N-1} \left[(\, \hat{\mathbf{x}}^1_{k|k-1} - \bar{\mathbf{x}}_{k|k-1}) \;\; \cdots \;\; (\, \hat{\mathbf{x}}^N_{k|ik1} - \bar{\mathbf{x}}_{k|k-1}) \right] \times \\ & \left[(\, \hat{\mathbf{x}}^1_{k|k-1} - \bar{\mathbf{x}}_{k|k-1}) \;\; \cdots \;\; (\, \hat{\mathbf{x}}^N_{k|k-1} - \bar{\mathbf{x}}_{k|k-1}) \right]^H \\ =& \frac{1}{N-1} \left[\delta\hat{\mathbf{x}}^1_{k|k-1} \;\; \cdots \;\; \delta\hat{\mathbf{x}}^N_{k|k-1} \right] \left[\delta\hat{\mathbf{x}}^1_{k|k-1} \;\; \cdots \;\; \delta\hat{\mathbf{x}}^N_{k|k-1} \right]^H, \\ \Rightarrow \quad P^e_{k|k-1} =& \frac{1}{N-1} \sum_{j=1}^{N} \delta\hat{\mathbf{x}}^j_{k|k-1} (\delta\hat{\mathbf{x}}^j_{k|k-1})^H. \end{aligned} \tag{2.7}$$

Note that $P^e_{k|k-1} \in \Re^{n\times n}$; for high-dimensional systems, building up this matrix is computationally intractable but, as shown below, unnecessary in the implementation if the terms are computed in the appropriate order. As is seen in (2.7),

the covariance can be computed as a sum of outer products of the deviations of each ensemble member from the ensemble mean (that is, of the ensemble state perturbation vectors $\delta\hat{\mathbf{x}}^j_{k|k-1}$). Thus, (2.4) can be written:

$$\begin{aligned} L_k^{(1)} =& P^e_{k|k-1} H^H \\ =& \big(H P^e_{k|k-1} \big)^H \\ =& \frac{1}{N-1} \Big(H \sum_{j=1}^{N} \delta\hat{\mathbf{x}}^j_{k|k-1} (\delta\hat{\mathbf{x}}^j_{k|k-1})^H \Big)^H \\ =& \frac{1}{N-1} \sum_{j=1}^{N} \delta\hat{\mathbf{x}}^j_{k|k-1} (H \, \delta\hat{\mathbf{x}}^j_{k|k-1})^H, \\ \Rightarrow \quad L_k^{(1)} =& \frac{1}{N-1} \sum_{j=1}^{N} \delta\hat{\mathbf{x}}^j_{k|k-1} (\delta\hat{\mathbf{y}}^j_{k|k-1})^H, \end{aligned} \tag{2.8}$$

where $H \, \delta\hat{\mathbf{x}}^j_{k|k-1} = \delta\hat{\mathbf{y}}^j_{k|k-1} \in \Re^m$ is the ensemble output perturbation vector. The matrix H is the linearization of the output operator $h : \Re^n \rightarrow \Re^m$. Note that, for the multiscale chaotic systems of interest, $m \ll n$ (that is, the number of measurements is much smaller than the dimension of the state), so the storage and communication of the output perturbation vectors $\delta\hat{\mathbf{y}}^j_{k|k-1}$ can be assumed to be negligible compared to the storage and communication of the state and state perturbation vectors. At this point, locally on each process p^j, the ensemble state perturbation $\delta\hat{\mathbf{x}}^j_{k|k-1}$ must be computed along with the ensemble output perturbation $\delta\hat{\mathbf{y}}^j_{k|k-1}$.

Similarly, the first term in $L_k^{(2)}$, namely $H \, P^e_{k|k-1} \, H^H$, can be computed in a manner consistent with $L_k^{(1)}$, exploiting the structure of the ensemble covariance

matrix.

$$
\begin{aligned}
H\,P^{e}_{k|k-1}\,H^{H} =& H\left(\frac{1}{N-1}\sum_{j=1}^{N}\delta\hat{\mathbf{x}}^{j}_{k|k-1}(\delta\hat{\mathbf{x}}^{j}_{k|k-1})^{H}\right)H^{H} \\
=& \frac{1}{N-1}\sum_{j=1}^{N}(H\,\delta\hat{\mathbf{x}}^{j}_{k|k-1})(H\,\delta\hat{\mathbf{x}}^{j}_{k|k-1})^{H}, \\
\Rightarrow \quad H\,P^{e}_{k|k-1}\,H^{H} =& \frac{1}{N-1}\sum_{j=1}^{N}\delta\hat{\mathbf{y}}^{j}_{k|k-1}(\delta\hat{\mathbf{y}}^{j}_{k|k-1})^{H}. \qquad (2.9)
\end{aligned}
$$

This term is calculated as a sum over all the processes of the outer product of the ensemble output perturbation with itself (recall that this vector has already been computed on each process). In addition to the $H\,P^{e}_{k|k-1}\,H^{H}$ term, $L_k^{(2)}$ contains the measurement covariance matrix R. This matrix, in general, may be a function of time, but a model for R is assumed to be known.

The structure of many MPI clusters facilitates reasonably efficient all-to-all communication (in which data is passed from every node to every other node in the cluster at the same time). For instance, in a cluster with a toroidal switchless interconnect, all-to-all communication is only slightly more expensive than one-to-all communication (in which one node sends data to ever other node). This is because, in a switchless interconnect torus, during one-to-all communication the data is sent sequential from one node to the next, down the line, while all the other nodes wait. Thus, the time required for a one-to-all communication is the time required for the data to travel all the way down the line of nodes. However, during all-to-all communication, data is cycled down the line from every node. Thus,

every node is always busy, but the total communication time is still only the time it takes for data to travel once down the line.

In the interest of minimizing data transfer, all the ensemble output perturbation vectors $\delta\hat{\mathbf{y}}^j_{k|k-1}$ are thus transferred to every node, where $L_k^{(2)}$ can be computed locally. This requires only one all-to-all communication call for the ensemble output perturbation vectors. Conversely, if the summation components of $L_k^{(2)}$ were computed locally, an all-to-all communication of the entire matrix would be necessary, increasing communication significantly while decreasing computation only slightly.

In the EnKF framework, each individual ensemble member is assimilated with a noisy measurement. The noisy measurement on process p^j is denoted $\mathbf{d}_k^j$ and is found by adding random noise on top of the original measurement (from the truth model), with statistics consistent with the known properties of the sensors:

$$\mathbf{d}_k^j = \mathbf{y}_k + \mathbf{v}_k^j. \tag{2.10}$$

The statistics of the added noise mirror the known measurement noise of (1.1b). This gives the forcing to each ensemble member estimate $\mathbf{z}_k^j$ as

$$\mathbf{z}_k^j = \mathbf{d}_k^j - H\,\hat{\mathbf{x}}^j_{k|k-1} = \ \mathbf{y}_k + \mathbf{v}_k^j - \hat{\mathbf{y}}^j_{k|k-1}, \tag{2.11}$$

where $\hat{\mathbf{y}}^j_{k|k-1}$ is the ensemble output vector on each process. Hence, the calculation of this vector can be done locally; no message passing is required, other than to provide each process with the truth model measurement $\mathbf{y}_k$.

At this point, a simple linear system needs to be solved [due to the $\left(L_k^{(2)}\right)^{-1}$ term] on each process. This solve is straightforward because $L_k^{(2)} \in \Re^{m \times m}$ is both symmetric and relatively small. Many algorithms exist for the efficient solution of such systems. Note that, with the assumption $R > 0$, the matrix $L_k^{(2)}$ is, in general, nonsingular, and thus the solution to the following system exists and is unique:

$$\mathbf{u}_k^j = \left(L_k^{(2)}\right)^{-1} \mathbf{z}_k^j \Rightarrow \quad L_k^{(2)}\, \mathbf{u}_k^j = \mathbf{z}_k^j. \tag{2.12}$$

With the computation of $\mathbf{u}_k^j$ done locally on each process, the update equation (2.3) can again be rewritten as:

$$\hat{\mathbf{x}}_{k|k}^j = \hat{\mathbf{x}}_{k|k-1}^j + L_k^{(1)}\, \mathbf{u}_k^j. \tag{2.13}$$

Substituting in the definition of $L_k^{(1)}$ from (2.8), this update becomes:

$$\begin{aligned}
\hat{\mathbf{x}}_{k|k}^j &= \hat{\mathbf{x}}_{k|k-1}^j + \left[\frac{1}{N-1} \sum_{i=1}^{N} \delta\hat{\mathbf{x}}_{k|k-1}^i \left(\delta\hat{\mathbf{y}}_{k|k-1}^i \right)^H \right] \mathbf{u}_k^j \\
&= \hat{\mathbf{x}}_{k|k-1}^j + \sum_{i=1}^{N} \left[\frac{\left(\delta\hat{\mathbf{y}}_{k|k-1}^i \right)^H \mathbf{u}_k^j}{N-1} \right] \delta\hat{\mathbf{x}}_{k|k-1}^i, \\
\Rightarrow \quad \hat{\mathbf{x}}_{k|k}^j &= \hat{\mathbf{x}}_{k|k-1}^j + \sum_{i=1}^{N} \gamma_k^{ij}\, \delta\hat{\mathbf{x}}_{k|k-1}^i \qquad (2.14a) \\
\text{where} \quad \gamma_k^{ij} &= \frac{\left(\delta\hat{\mathbf{y}}_{k|k-1}^i \right)^H \mathbf{u}_k^j}{N-1}. \qquad (2.14b)
\end{aligned}$$

In its final form, the measurement update equation (2.14) updates each ensemble member via a linear combination of each ensemble state perturbation vector $\delta\hat{\mathbf{x}}_{k|k-1}^j$. This form eliminates the need for any additional storage arrays. The

update can be computed in an all-to-all round robin format, where the ensemble state perturbation vector on each process is shifted one hop to the adjacent process. Then, the corresponding update is computed on every process, and the data is shifted again. Overall, the total communication is equivalent to a single all-to-all send of the ensemble state perturbation vector, but because the computation is done in between each message hop, there is no accumulating storage necessary.

2.1.1 EnKS Implementation

The full implementation of the EnKS is omitted here to due its similarity with that of the EnKF update as discussed in the previous section. Recall that the EnKS update equation is given by

$$\hat{\mathbf{x}}^j_{p|k} = \hat{\mathbf{x}}^j_{p|k-1} + S^e_{k-1} H^T (H P^e_{k|k-1} H^T + R)^{-1} (\mathbf{d}^j_k - H\hat{\mathbf{x}}^j_{k|k-1}), \tag{2.15a}$$

where S^e_{k-1} is the time covariance matrix between the estimate at the observation time t_k and the estimate at the smoothing time t_p, it is given by

$$S^e_{k-1} = \frac{(\delta\widehat{X}_{p|k-1})\ (\delta\widehat{X}_{k|k-1})^T}{N-1}. \tag{2.15b}$$

Using this form of the update equation, the numerical steps necessary for its implementation follow directly from that of the standard EnKF. Thus the full implementation of the fixed-lag EnKS (necessary in the HEnS algorithm) requires two ensemble Kalman updates at each new measurement time: first, the smoothed estimate at the back edge of the window is updated according to the above equations

(which required the un-updated filtered ensemble); second, the filtered estimate at the front edge of the window is updated according to the traditional EnKF equations. Note that many of the computational steps of the two updates are identical, and thus the total cost is much less than that of two independent updates. The completion of this process over a batch of measurements provides two ensemble estimates, both conditioned on all available measurements, one at the front edge of the window and one at the back edge of the window (which, of course, will then be used as the initial condition for the En4DVar step of the HEnS).

2.2 En4DVar Implementation

In this section, we will first go over the general implementation of En4DVar for high-dimensional systems. This assumes the only available initial condition is the ensemble background. Lastly, we discuss the steps necessary to initialize En4DVar with something besides the background (e.g. a smoothed estimate found from the EnKS, as is done with the HEnS).

In either case, the delicate issue is the treatment of the background term in the cost function, $P^e_{0|0}$. In the traditional En4DVar cost function,

$$J_j(\mathbf{u}^j) = \frac{1}{2}\,(\mathbf{u}^j - \hat{\mathbf{x}}^j_{0|0})^T\,(P^e_{0|0})^{-1}\,(\mathbf{u}^j - \hat{\mathbf{x}}^j_{0|0}) + \frac{1}{2}\sum_{k=1}^{K}(\mathbf{d}^j_k - H\,\tilde{\mathbf{x}}^j_k)^T\,R^{-1}\,(\mathbf{d}^j_k - H\,\tilde{\mathbf{x}}^j_k), \tag{2.16}$$

this background terms serves to penalize deviations from the background mean

inversely proportional to the uncertainty in that direction. However, due to the small number of tractable ensemble members, this matrix is in general singular, and thus non-invertable. In these cases, we must use the pseudo-inverse, giving a new cost function,

$$J_j(\mathbf{u}^j) = \frac{1}{2}\,(\mathbf{u}^j - \hat{\mathbf{x}}^j_{0|0})^T\,(P^e_{0|0})^+\,(\mathbf{u}^j - \hat{\mathbf{x}}^j_{0|0}) + \frac{1}{2}\sum_{k=1}^{K}(\mathbf{d}^j_k - H\,\tilde{\mathbf{x}}^j_k)^T\,R^{-1}\,(\mathbf{d}^j_k - H\,\tilde{\mathbf{x}}^j_k). \tag{2.17}$$

In directions of uncertainty that are spanned by our reduced subset of ensembles, the pseudo-inverse behaves identically to that of the inverse, giving us the desired behavior. However, in the null space of $P^e_{0|0}$, the pseudo-inverse applies no penalty to deviations in that direction. This is the opposite of what we would desire. The reduced number of ensemble members that we have all lie on a low dimensional manifold of the high-dimensional state space. Because none of these members, by definition, lie off this manifold, we can expect to have some certainty that the solutions we find tend to stay in the vicinity of this manifold as well. Consequently, we should desire a *heavy* penalty for any solution moving off this manifold. To combat this contradiction that arises from using the pseudo-inverse, we must put a hard constraint on the optimization, not allowing any updates in the directions off the manifold, or, better put, require all updates to be in the range of the

ensemble perturbations $\delta\widehat{X}_{0|0}$ i.e.

$$\mathbf{u}^j \leftarrow \hat{\mathbf{x}}^j_{0|0} + \delta\widehat{X}_{0|0}\mathbf{w}^j \tag{2.18a}$$

$$\text{where} \quad \mathbf{w}^j \in \Re^N \tag{2.18b}$$

$$\text{and thus} \quad (\mathbf{u}^j - \hat{\mathbf{x}}^j_{0|0}) \;=\; \delta\widehat{X}_{0|0}\mathbf{w}^j. \tag{2.18c}$$

Plugging this hard constraint (2.18c) back into our cost function (2.17), we get a new cost function parametrized by the weights on the columns of $\delta\widehat{X}_{0|0}$,

$$\begin{aligned} J_j(\mathbf{w}^j) = {} & \frac{1}{2}\,(\mathbf{w}^j)^T\,(\delta\widehat{X}_{0|0})^T\,(P^e_{0|0})^+\,\delta\widehat{X}_{0|0}\,\mathbf{w}^j \\ & + \frac{1}{2}\sum_{k=1}^{K}(\mathbf{d}^j_k - H\,\tilde{\mathbf{x}}^j_k)^T\,R^{-1}\,(\mathbf{d}^j_k - H\,\tilde{\mathbf{x}}^j_k). \end{aligned} \tag{2.19}$$

This cost function can be further simplified by constructing the reduced singular value decomposition (SVD) of $\delta\widehat{X}_{0|0}$ and recalling its relationship to the background sample covariance $(P^e_{0|0})^+$, i.e.

$$\delta\widehat{X}_{0|0} = U\,\Sigma\,V^T \quad \text{where} \quad U^T\,U \;=\; V^T\,V \;=\; I, \tag{2.20a}$$

$$P^e_{0|0} = \frac{1}{N-1}\,(\delta\widehat{X}_{0|0})\,(\delta\widehat{X}_{0|0})^T \;=\; \frac{1}{N-1}\,U\,\Sigma^2\,U^T, \tag{2.20b}$$

$$\text{and thus} \quad (P^e_{0|0})^+ \;=\; (N-1)\,U\,\Sigma^{-2}\,U^T. \tag{2.20c}$$

Now, substituting (2.20) into (2.19), we get a simplified expression for the cost,

$$J_j(\mathbf{w}^j) = \frac{1}{2}\,(\mathbf{w}^j)^T\,V\,\Sigma\,U^T\,U\,\Sigma^{-2}\,U^T\,U\,\Sigma\,V^T\,\mathbf{w}^j + \frac{N-1}{2}\sum_{k=1}^{K}(\mathbf{d}_k^j - H\,\tilde{\mathbf{x}}_k^j)^T\,R^{-1}\,(\mathbf{d}_k^j - H\,\tilde{\mathbf{x}}_k^j). \tag{2.21a}$$

$$\implies \quad J_j(\mathbf{w}^j) = \frac{N-1}{2}\,(\mathbf{w}^j)^T\,V\,V^T\,\mathbf{w}^j + \frac{1}{2}\sum_{k=1}^{K}(\mathbf{d}_k^j - H\,\tilde{\mathbf{x}}_k^j)^T\,R^{-1}\,(\mathbf{d}_k^j - H\,\tilde{\mathbf{x}}_k^j). \tag{2.21b}$$

Lastly, defining a new variable $\boldsymbol{\omega}^j \in \Re^{N-1}$ such that $\boldsymbol{\omega}^j = V^T\,\mathbf{w}^j$ we arrive at the final reduced cost function

$$J_j(\boldsymbol{\omega}^j) = \frac{N-1}{2}\,(\boldsymbol{\omega}^j)^T\,\boldsymbol{\omega}^j + \frac{1}{2}\sum_{k=1}^{K}(\mathbf{d}_k^j - H\,\tilde{\mathbf{x}}_k^j)^T\,R^{-1}\,(\mathbf{d}_k^j - H\,\tilde{\mathbf{x}}_k^j). \tag{2.22}$$

As with traditional En4DVar, each ensemble member is constrained by the underlying model, and the control variable $\boldsymbol{\omega}^j$ serves to define the initial condition for its trajectory.

$$\frac{d\tilde{\mathbf{x}}^j(t)}{dt} = f(\tilde{\mathbf{x}}^j(t), 0), \tag{2.23a}$$

$$\tilde{\mathbf{x}}_0^j = \hat{\mathbf{x}}_{0|0}^j + \delta\widehat{X}_{0|0}\mathbf{w}^j$$

$$\tilde{\mathbf{x}}_0^j = \hat{\mathbf{x}}_{0|0}^j + U\,\Sigma\,V^T\,\mathbf{w}^j$$

$$\tilde{\mathbf{x}}_0^j = \hat{\mathbf{x}}_{0|0}^j + U\,\Sigma\,\boldsymbol{\omega}^j$$

In practice, however, the columns of U are of dimension n. Therefore, they are too expensive to compute for multiscale systems. On the other hand, the columns of V are of dimension N and can be easily found via an eigendecomposition of the

symmetric $(N \times N)$ matrix $(\delta\widehat{X}_{0|0})^T\,\delta\widehat{X}_{0|0}$. In addition, returning to our definition of the SVD (2.20a), we see that $U = \delta\widehat{X}_{0|0}\,V\,\Sigma^{-1}$. Thus each ensemble trajectory can be initialized as

$$\tilde{\mathbf{x}}_0^j = \hat{\mathbf{x}}_{0|0}^j + \delta\widehat{X}_{0|0}\,V\,\boldsymbol{\omega}^j \tag{2.23b}$$

For standard En4DVar, we can proceed from here, initializing $\boldsymbol{\omega}^j = 0$ and thus using the background as the only known initial condition. The adjoint is found and forced in an identical manner to that previously described for traditional 4DVar. Thus, at each iteration, after a forward march of the state and the appropriate backward march of the adjoint, we are provided with the sensitivity of the cost function to the initial condition. To extract the necessary gradient (with respect to $\boldsymbol{\omega}^j$) we utilize (2.23b), to find a clear relationship between our low dimensional optimization space and the initial state. Thus, the final gradient is given by:

$$\nabla J_j(\boldsymbol{\omega}^j) = (N-1)\,\boldsymbol{\omega}^j - V^T\,(\delta\widehat{X}_{0|0})^T\,\mathbf{r}_0^j. \tag{2.24}$$

2.2.1 Extension to HEnS

With traditional En4DVar, we initialize the optimization with the background ensemble and thus, according to (2.23b), the initial condition for the optimization $\boldsymbol{\omega}_0^j = 0$. From here, the optimization is straightforward. With HEnS, though, we want to use the output from the smoother $\hat{\mathbf{x}}_{0|K}^j$ as the initial condition. However, this smoothed estimate is not guaranteed to lie on the subspace spanned

by the ensemble perturbation, i.e., it is not necessarily true that

$$\left[\; \exists \, \boldsymbol{\omega}^j \in \Re^{N-1} \mid \hat{\mathbf{x}}^j_{0|K} = \hat{\mathbf{x}}^j_{0|0} + \delta \widehat{X}_{0|0} \, V \, \boldsymbol{\omega}^j \; \right]. \tag{2.25}$$

In this situation, the best we can do is to find the closest estimate to that of the smoothed one, such that it does lie in the range of $\delta \widehat{X}_{0|0}$, this is done through the proper use of the pseudo-inverse as follows,

$$\boldsymbol{\omega}^j_0 = \Sigma^{-2} \, V^T \, (\delta \widehat{X}_{0|0})^T \, (\, \hat{\mathbf{x}}^j_{0|K} - \hat{\mathbf{x}}^j_{0|0} \,). \tag{2.26}$$

Part II

Noncartesian Computational Interconnects

Chapter 3

Structured Computational Interconnects

Joseph Cessna and Thomas Bewley

Abstract. The present paper is part of a larger effort to redesign, from the ground up, the best possible interconnect topologies for switchless multiprocessor computer systems. We focus here specifically on hexagonal interconnect graphs and their extension to problems on the sphere, as motivated by the design of special-purpose computational clusters for global weather forecasting. Eight families of efficient tiled layouts have been discovered which make such interconnects trivial to scale to large cluster sizes while incorporating no long wires. In the

resulting switchless interconnect designs, the *physical proximity* of the cells created (in the PDE discretization of the physical domain) and the *logical proximity* of the nodes to which these cells are assigned (in the computational cluster) coincide perfectly, so all communication between physically adjacent cells during the PDE simulation require communication over just a single hop in the computational cluster.

3.1 Introduction

There are two paradigms for interconnecting processing elements in multiprocessor computer systems: switched and switchless.

Switched multiprocessor computer systems are the easiest to field and use in general-purpose applications, and are thus today the most popular. Fast cluster switching hardware has been developed by Infiniband, Myrinet, and Quadrics, and inexpensive ("commodity") switching hardware is available leveraging the standard gigabit ethernet protocol from Cisco. Unfortunately, in a switched computer system, the switch itself is a restrictive bottleneck in the system when attempting to scale to large cluster sizes, as messages between any two nodes must pass through the switch, and thus the throughput demands on the switch increase rapidly as the cluster size is increased. Nonuniform memory access (NUMA) architectures, first pioneered by Silicon Graphics, attempt to circumvent this quagmire by introducing a hierarchy of switches, thus allowing some of the "local" messages (that is, between two nodes on the same "branch" of a tree-like structure) to avoid passing through the full cascade of switches (that is, to avoid going all the way back to the "trunk"). This NUMA paradigm certainly helps, but does not eliminate the bottlenecks inherent to switch-based architectures.

Switchless multiprocessor computer systems, on the other hand, introduce a "graph" (typically, some sort of n-dimensional "grid") to interconnect the nodes

of the system. In such a system, messages between any two nodes are relayed along an appropriate path in the graph, from the source node to the destination node. To accomplish such an interconnection in a beowulf cluster, relatively inexpensive PCI cards are available from Dolphin ICS [1]; however, the use of such hardware in today's high-performance clusters is fairly uncommon. The massively parallel high-performance Blue Gene design, by IBM, is a switchless three-dimensional torus network with dynamic virtual cut-through routing [2].

In the history of high-performance computing, switchless interconnect architectures have gone by a variety of descriptive names, including the 2D torus, the 3D torus, and the hypercube. Almost all such designs, including the IBM Blue Gene and the Dolphin ICS designs discussed above, imply an underlying Cartesian (that is, rectangular) grid topology in two, three, or $n > 3$ dimensions.

Quite recently, the startup SiCortex broke away from the dominant Cartesian interconnect paradigm, launching a novel family of switchless multiprocessor computer systems designed around the Kautz graph [54]. The Kautz graph is the optimal interconnect solution in terms of connecting the largest number of nodes of a given "degree" (that is, with a given number of incoming and outgoing wires at each node) for any prescribed maximum graph "diameter" (that is, the maximum number of hops between any two nodes in the graph). If one considers the wide range of possible graphs that may be used to interconnect a large number of computational nodes, the Cartesian graph may be identified as one extreme, with

the simplest local structure possible but a poor graph diameter, whereas the Kautz graph may be considered the other extreme, with a complex logical structure that sacrifices local order but exhibits the optimal graph diameter.

In certain unstructured applications, the optimal graph diameter offered by the Kautz graph is attractive, though such systems become difficult to build as the cluster size is increased due to the intricate weave of long wires spanning the entire system.

Many problems of interest in high performance computing, however, have a regular structure associated with them. A prime example is the discretization of a partial differential equation (PDE). When distributing such a discretization on a switchless multiprocessor computer systems for its parallel solution, one generally divides the domain of interest into a number of finite regions, or Voronoi cells, assigning one such cell to each computational node. An important observation is that such computations usually require *much* more communication between neighboring cells than they do between cells that are physically distant from one another. Thus, the practical effectiveness of proposed solutions to (i) the definition of the Voronoi cells, and (ii) the distribution of these cells over the nodes of the cluster (together referred to as the "load balancing problem") is closely related to both the physical proximity of the cells created in the PDE discretization and the logical proximity of the nodes to which these cells are assigned in the computational cluster. A graph with local structure, such as the Cartesian graph, can

drastically reduce the average number of hops of the messages it must pass during the simulation of the PDE by laying out the problem in such a way that these two proximity conditions coincide; a graph without such local structure, such as the Kautz graph, does not admit an efficient layout which achieves this condition.

The present line of research thus considers alternative (noncartesian) graphs with local structure exploitable by PDE discretizations, while keeping to a minimum both (a) the number of wires per node [to minimize the complexity/expense of the cluster], and (b) the graph diameter [to minimize the cost of whatever multi-hop communication is required during the PDE simulation].

This particular paper is motivated by the needs presented by global weather forecasting problems defined over a sphere; note that some of the largest purpose-built computational clusters in the world are dedicated to this application. Loosely speaking, the present paper explores the best ways to put a fine hexagonal grid on a sphere, and then explores how to realize this discretization efficiently on an easily-scaled layout of computational hardware without using any long wires.

The work considered may be applied immediately at the system level. With the further development of appropriate hardware, it may also be applied at the board level or even the chip level. Other chip-level noncartesian interconnect strategies which have been investigated in the literature include the Y architecture and the X architecture. The Y architecture for on-chip interconnects is based on the use of three uniform wiring directions (0^o, 120^o, and 240^o) to exploit on-chip

routing resources more efficiently than the traditional Cartesian (a.k.a. Manhattan) wiring architecture [13, 14]. The X architecture is an integrated-circuit wiring architecture based on the pervasive use of diagonal wires. Note that, compared with the traditional Cartesian architecture, the X architecture demonstrates a wire length reduction of more than 20% [58].

3.2 Cartesian interconnects

Two criteria by which switchless interconnects are measured are cluster diameter and maximum wire length [16]. Loosely speaking, the former affects the speed at which information is passed throughout the graph, whereas the latter affects the cost of each wire used to construct the interconnect, as described further below.

Because each node can communicate directly only with its logical neighbors, we characterize information as moving in hops: it takes one hop for information to travel from a given node to its immediate neighbor, two hops for information to travel to a neighbor of a neighbor, etc. The diameter of a graph is the maximum number of hops between any two nodes in the graph. For example, Figure 3.1 illustrates a 1D Cartesian graph with a periodic connection. Each node can send information to its neighbor to the right, and receive information from its neighbor to the left. This type of connection is called a unidirectional link, because infor-

mation can only flow in one direction. Many switchless clusters use bidirectional links, through which a node can both send and receive data.

The expense of the hardware required to complete a hop often increases quickly with the physical length of the wire between the nodes. To reduce this cost, it is thus desirable to minimize the maximum wire length. In Figure 3.1, the link connecting nodes 1 and N traverses N nodes, which makes scaling this layout to large N costly. The problem of long wires can be circumvented by folding the graph. By keeping the same logical connection, but folding the graph onto itself along its axis of symmetry, one can produce the graph shown in Figure 3.2. Here, the interconnect is identical to that of Figure 3.1 (and, thus, so is the graph diameter), but now the longest wire only spans the distance between two nodes, independent of N, thus facilitating scaling of the cluster to large N.

Noting the repetitive pattern in Figure 3.2, we identify a self-similar tile that can be used to build the interconnect. This tile is composed of two nodes and four wires (two sending and two receiving). Figure 3.3a illustrates how four of these tiles, along with simple end caps, can be combined to produce the original interconnect of Figure 3.1. An important feature of the tiled configuration is its scalability; note how it can be extended to much larger interconnects with the same topology, such as the 32 node graph in Figure 3.3b.

We now consider a four-connected periodic 2D Cartesian graph, known as a torus, in which each node has four unidirectional links, two for sending and two

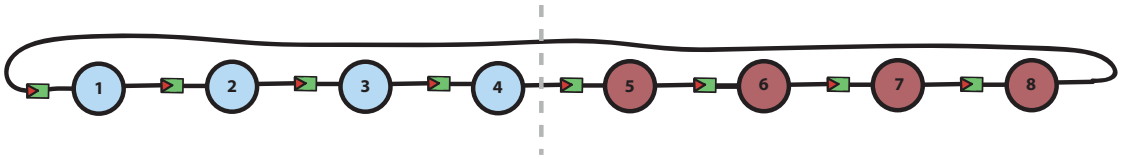

Figure 3.1: A simple 1D periodic Cartesian interconnect with unidirectional links. The diameter of this graph is seven hops. Note that a long wire is needed to make the periodic connection; the length of this wire increases as the cluster size increases.

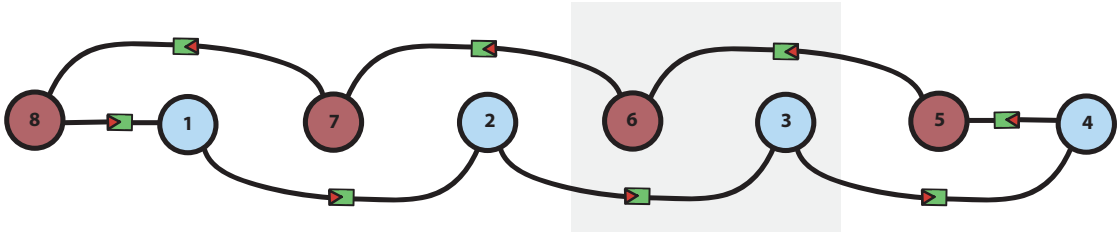

Figure 3.2: By folding the simple 1D interconnect of Figure 3.2 in half while keeping the same logical connection, the effect of the periodic connection may be localized. In this case, the longest wires only span the distance between two nodes in the folded structure, regardless of the number of nodes in the graph.

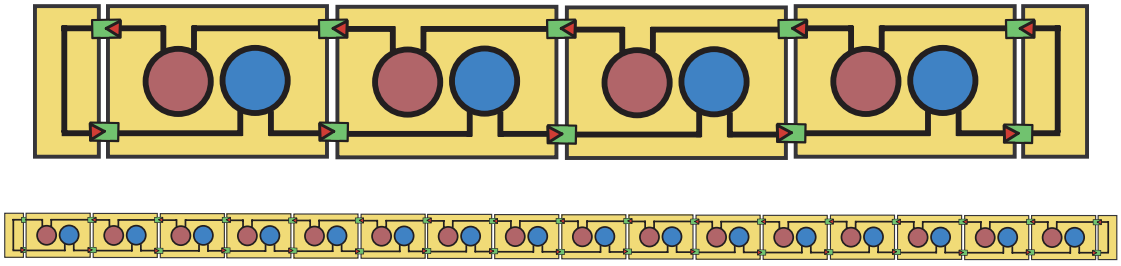

Figure 3.3: By identifying the local structure of the folded 1D interconnect of Figure 3.2, a tile may be designed that contains two nodes and four wires. This tile, together with simple end caps, may be extended to larger interconnects with the same topology. Here, we extend from 8 nodes (top) to 32 nodes (bottom).

for receiving, as illustrated in Figure 3.4a. Similar to the 1D graph of Figure 3.1, the diameter of this 2D graph is six hops, but now interconnects 16 nodes instead of 8. The periodic connections of this 2D graph create many long wires that span the entire width of the interconnect. These long wires can be eliminated by folding the graph onto itself along both axes of symmetry.

From the folded 2D graph, we again identify local structure that facilitates tiling. The tiles for the 2D Cartesian torus contain four nodes and the associated communication links. Figure 3.5 shows how these tiles, together with simple end caps, may be assembled to produce the original 2D periodic Cartesian graph, and scaled to larger graphs of the same topology.

The three key steps illustrated by example in this section are:

(i) folding a graph to minimize the maximum wire length,

(ii) identifying repetitive local structure in the folded graph, and

(iii) defining a self-similar tiling to facilitate scaling.

The remainder of this paper extends these three steps to hexagonal interconnects with a variety of useful periodic closures. The three-connected graphs so generated, in which each node is connected to an odd number of nearest neighbors, lack the symmetry required to configure an effective interconnect using unidirectional links. As a result, each link in the remainder of this discussion is intended to represent either a bidirectional link or a pair of unidirectional links (one in each direction).

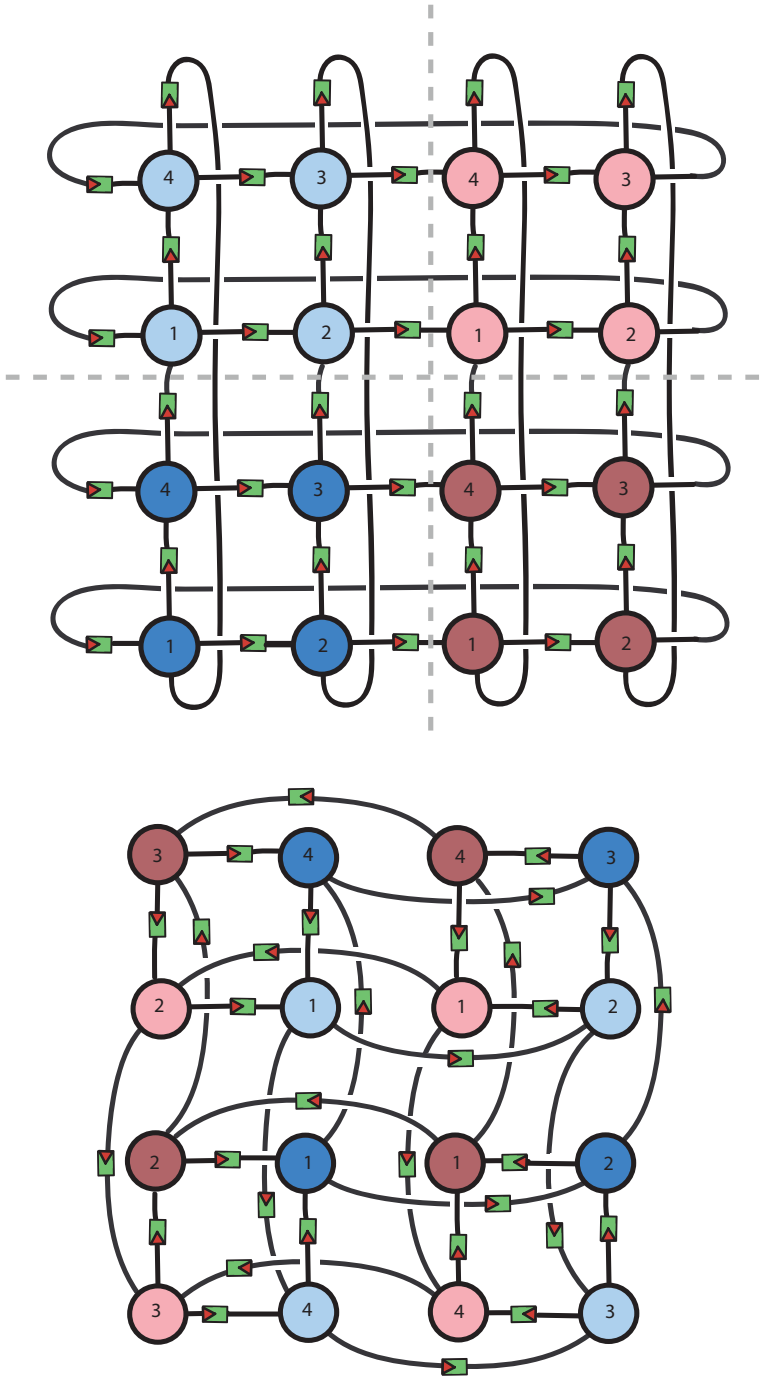

Figure 3.4: The idea of folding the interconnect to minimize the maximum wire length extends directly to higher dimensions. Here, a 2D periodic Cartesian graph (top) is folded onto itself in both directions of symmetry. Again, the longest wires only span the distance between two nodes in the folded structure.

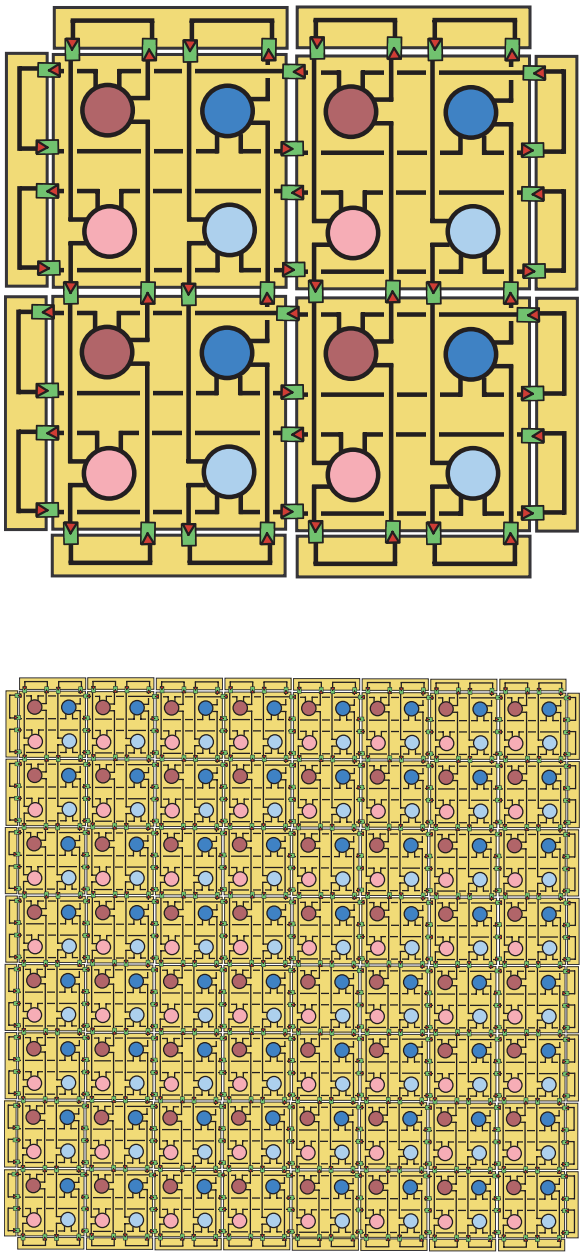

Figure 3.5: A four-node tile can be extracted from the interconnect of Figure 3.4b. This tile, combined with simple end caps, can be extended to larger interconnects with the same topology. Here, we extend from 16 nodes (top) to 256 nodes (bottom).

In §3.3.1, we consider the interconnects that arise from periodically connecting the hexagonal graph in the directions of the Cartesian unit vectors, much like the graphs of Figure 3.4, to produce a toroidal class of interconnects. In §3.3.2, we then examine the tilings that arise by periodically closing a hexagonal graph in three directions instead of two. Finally, §3.3.3 examines a variety of methods for wrapping a sphere with a (mostly) hexagonal graph.

3.3 Hexagonal interconnects

Hexagonal (three-connected) graphs may be used in lieu of Cartesian (four-connected) graphs to create 2D interconnects with a significantly reduced number of wires (and, thus, a significantly reduced cost). We now show that such graphs can be developed with essentially no increase in the overall complexity of the layout, and can easily be extended from toroidal closures, as discussed above, to triply-periodic closures, and then to spherical closures.

3.3.1 Toroidal closure

The simplest method for laying out a 2D periodic hexagonal interconnect is shown in Figure 3.6a. By making a periodic connection in the direction of one of the Cartesian unit vectors, we make a tubular topology similar to that of a carbon nanotube. This nanotube-like structure is then closed upon itself about its other

axis of symmetry, forming a torus. Like the Cartesian torus, one can modify this closure by applying varying amounts of twist in one or both periodic directions before closing the graph. Such twists might prove useful in future applications, but for brevity are not considered further here.

Tiling the toroidal closure. As with the Cartesian torus, the periodic connections in the toroidal nanotube illustrated in Figure 3.6a create long wires that span the width of the entire graph, thus hindering scalability. To eliminate these long wires, a folding strategy is again used to reveal local structure and identify a tiling, as illustrated in Figure 3.6b. Unlike the Cartesian case (due primarily to the anisotropy of the topology with respect to the closure applied), we now define two distinct tiles, each with four nodes. However, the overall complexity of the tiling is on par with that of the Cartesian interconnect discussed previously. It is observed that the hexagonal tilings of the family illustrated in Figure 3.6 (with bidirectional links) have essentially the same diameter as the corresponding Cartesian tilings with the same number of nodes. Hence, by moving to a hexagonal interconnect, we develop a graph with the same diameter but only 3/4 of the wiring cost/complexity.

3.3.2 Triply-periodic closure

Although we can improve upon Cartesian interconnects with hexagonal graphs while still using a Cartesian closure strategy, as discussed above, it is more

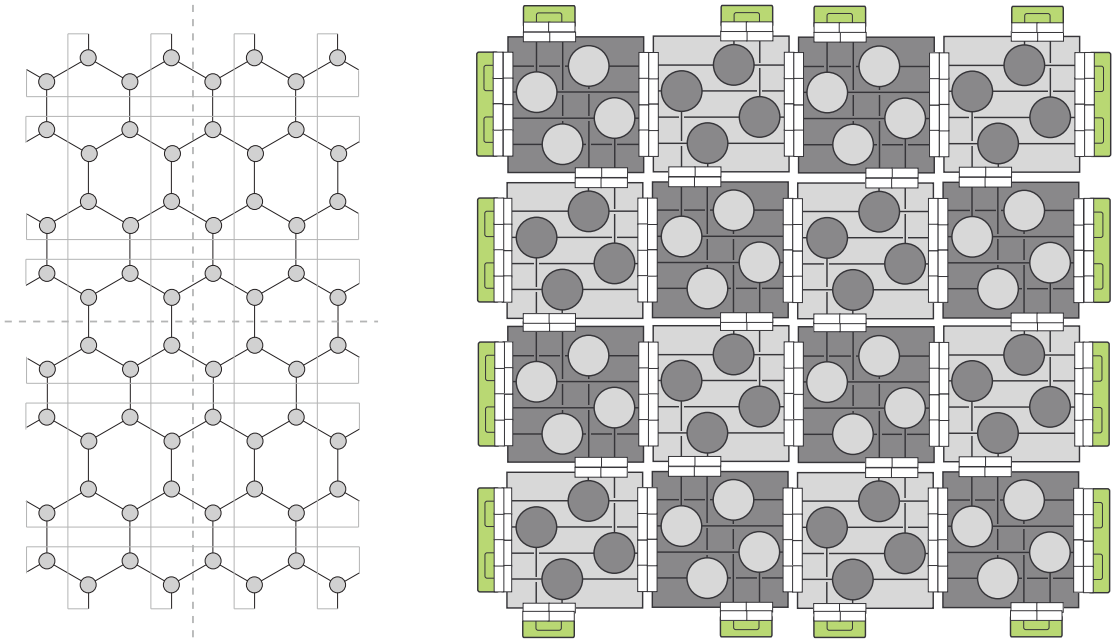

Figure 3.6: A 2D hexagonal grid is anisotropic about its Cartesian axes of symmetry. Nonetheless, periodic connections can be made in the directions of the Cartesian unit vectors (left). This closure produces a topology called a toroidal nanotube. By folding about the Cartesian axes, two distinct tiles can be identified to construct the interconnect (right). This interconnect has the same diameter the corresponding Cartesian torus with the same number of nodes, but uses 25% fewer wires.

natural to select a closure for the hexagonal graph that better reflects its inherent symmetries. Towards this end, we now examine the triply-periodic closure of the plane hexagonal graph. This study forms the foundation upon which our study of spherical closures (§3.3.3) is based. Solutions to this problem build from two distinct classes of hexagonal graphs on the equilateral triangle, denoted in this work as Class A and Class B structures:

A) The Class A structures place the midpoints of the links on the edges of the equilateral triangle, as illustrated in Figure 3.7. The degree of this structure is defined as the number of midpoints that lie on each edge of the triangle.

B) The Class B structures place the edges of the hexagons on the edges of the equilateral triangle, as illustrated in Figure 3.8. The degree of this structure is defined as the number of hexagons touching each edge of the triangle.

It is straightforward to join six Class A or Class B structures, as illustrated in Figures 3.7 and 3.8, to form a hexagon, as illustrated in Figure 3.9. The periodic connections on this hexagon are easily applied: in the case of Class A, the wires on opposite sides of the hexagon are connected; in the case of Class B, the nodes on opposite sides of the hexagon are taken to be identical (in both cases, moving orthogonal to each side of the hexagon, not diagonally through the center point). The resulting graphs are denoted $PA*$ and $PB*$, where the $*$ denotes the degree of the six Class A or Class B structures from which the triply-periodic hexagonal

graph is built.

Tiling the triply-periodic closure. As in the previous examples, the triply-periodic graphs of Figure 3.9 can be tiled via folding about the three axes of symmetry and identifying the repetitive local structure of the folded graph. This process eliminates all long wires which grow as the graph size is increased. For Class A, the new tile so constructed, denoted Tile E in Figure 3.10, contains six nodes instead of four, leading to the tiled $PA*$ family illustrated in the first four subfigures of Figure 3.19. For Class B, the new tile so constructed contains 18 nodes; as illustrated in the last four subfigures of Figure 3.19, and for later convenience, we may immediately split this 18-node tile into three identical smaller tiles, denoted Tile F in Figure 3.10. The tilings are completed with simple end caps.

3.3.3 Spherical closure

We now discuss the most uniform techniques available to cover a sphere with a (mostly) hexagonal grid.

Note first that each An structure of Figure 3.7 has $V = n^2$ vertices (that is, nodes) and $E = 1.5n^2$ edges (that is, wires between nodes), whereas each Bn structure of Figure 3.8 has $V = 3n^2$ vertices and $E = 4.5n^2$ edges. If each corner of an An structure is joined with five other identical An structures, then each An structure contributes effectively $F = 0.5n^2$ faces (that is, hexagons) to the

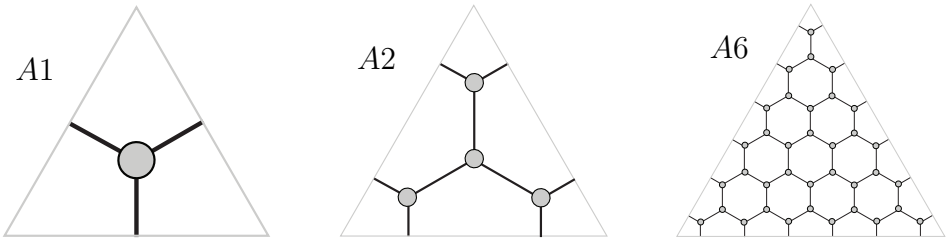

Figure 3.7: Three Class A structures (that is, hexagonal graphs on the equilateral triangle) with degree = 1, 2, and 6.

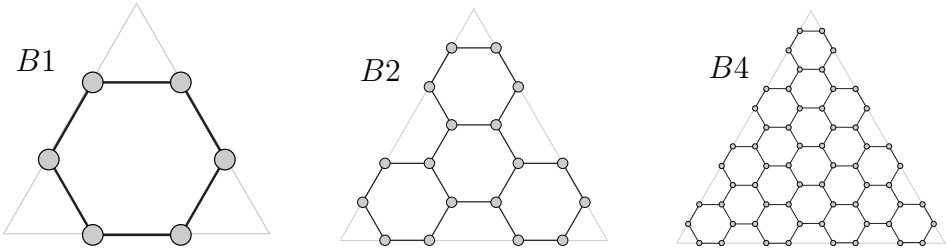

Figure 3.8: Three Class B structures with degree = 1, 2, and 4.

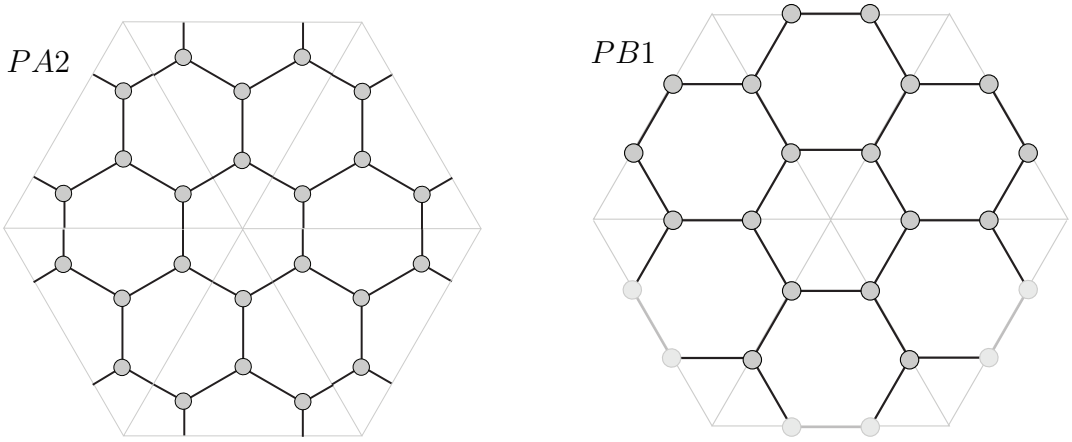

Figure 3.9: The two families of triply-periodic hexagonal interconnects. The $PA*$ family is built from six Class A structures and has connected edge links whereas the $PB*$ family is built from six Class B structures and has coincident edge nodes.

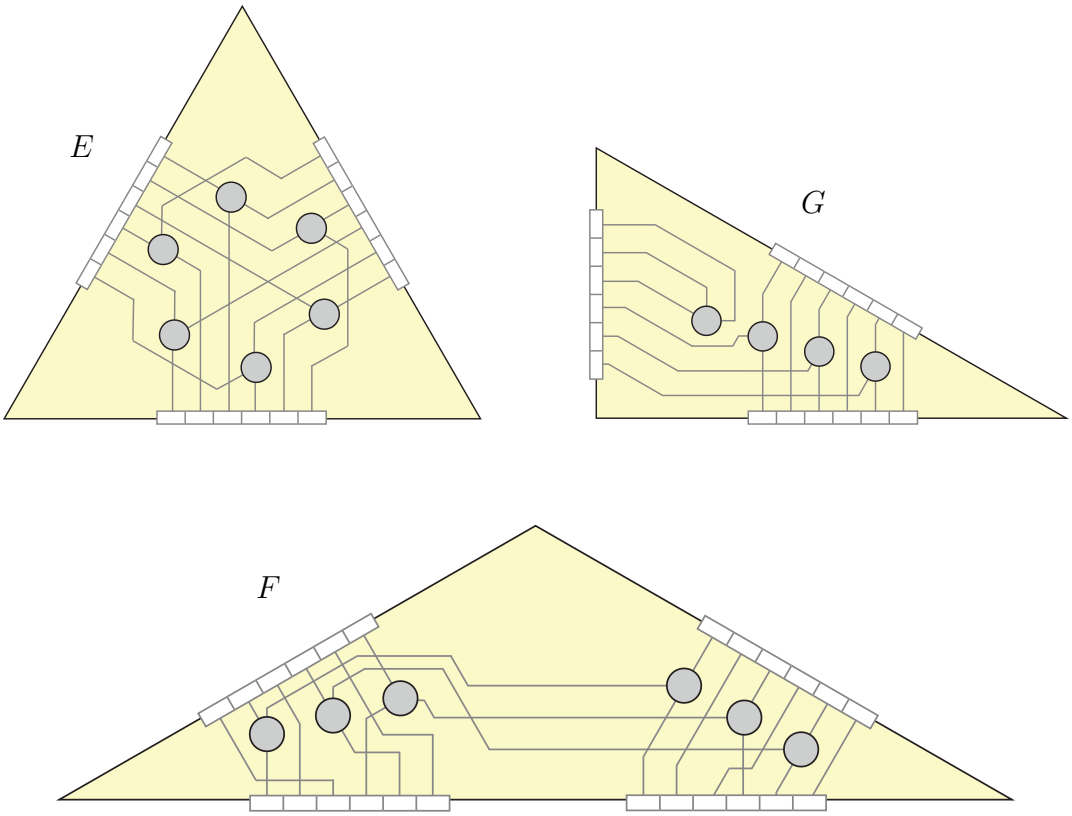

Figure 3.10: The three fundamental tiles, denoted E, F, and G, upon which the tilings of the triply-periodic (§3.3.2) and spherical (§3.3.3) closures of the hexagonal interconnect are based.

overall graph, whereas if each corner of a Bn structure is joined with five other Bn structures, then each Bn structure contributes $F = 1.5n^2$ faces to the overall graph. In both cases, we have $V - E + F = 0$, which is characteristic of a planar graph.

Euler's formula $V - E + F = 2$ relates the numbers of vertices, edges, and faces of any convex polyhedron. The upshot of Euler's formula in the present problem is that it is impossible to cover a sphere perfectly with a hexagonal grid. By this formula, any attempt to map a hexagonal grid onto the sphere will lead to a predictable number of "defects" (that is, faces which are not hexagons).

The three most uniform constructions available for generating a mostly hexagonal graph on a sphere are given by pasting a set of identical An structures of Figure 3.7, or Bn structures of Figure 3.8, onto the triangular faces of a Tetrahedron, Octahedron, or Icosahedron (see Figure 3.11); these three constructions form the basis for the remainder of this study. Note that this approach joins each corner of the structure selected with two, three, or four other identical structures, and leads to 4 triangular faces, 6 square faces, or 12 pentagonal faces embedded within an otherwise hexagonal graph. The resulting graph is easily projected onto the sphere. All graphs so constructed, of course, satisfy Euler's formula. Some of the graphs so constructed occur in nature as nearly spherical carbon molecules known as Buckyballs.

Table 3.1 shows the number of nodes in these graphs as a function of their

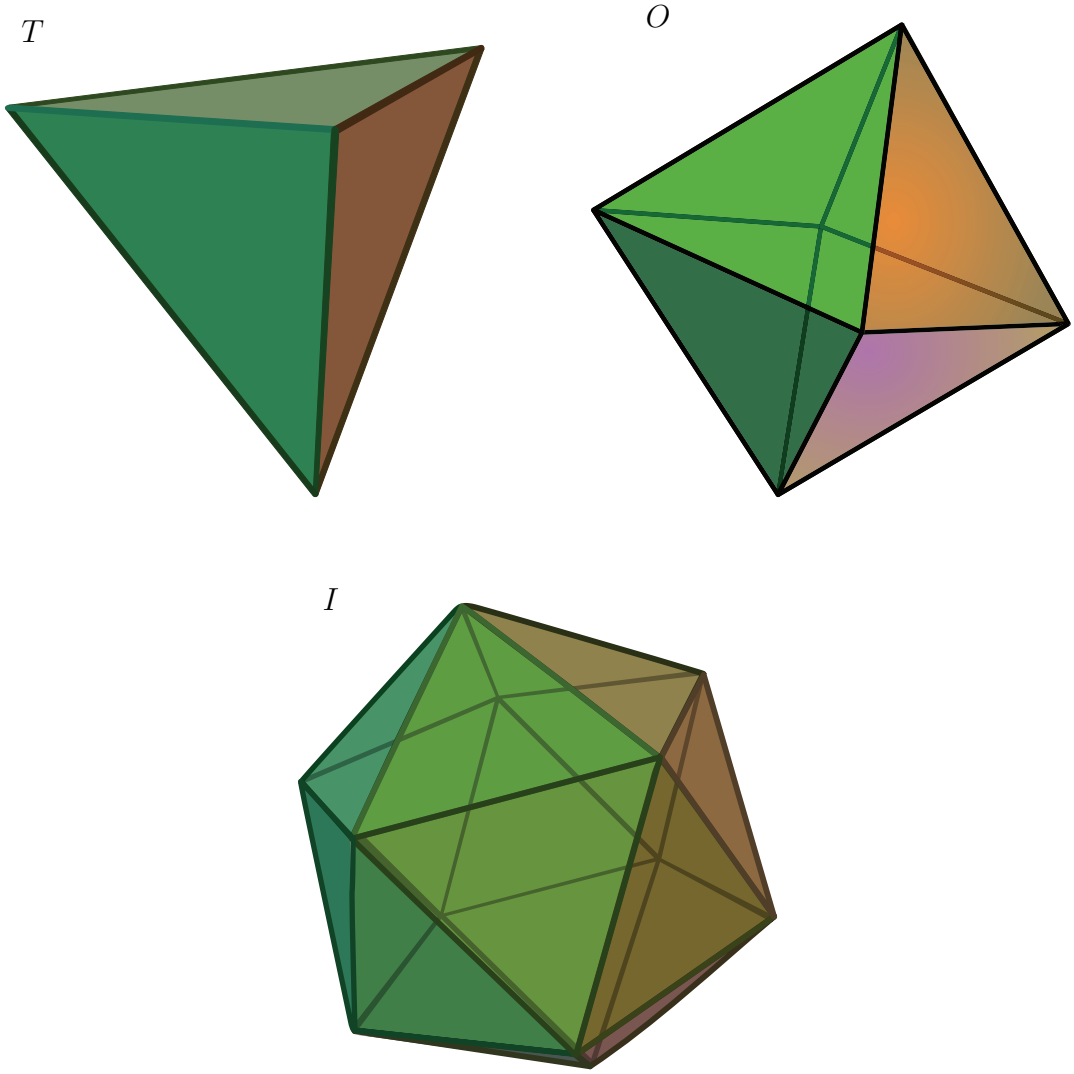

Figure 3.11: The three Platonic solids with triangular faces. From left to right: Tetrahedron (4 faces), Octahedron (8 faces), and Icosahedron (20 faces). The faces of these polyhedra can be gridded with either Class A or Class B triangular graphs (see Figures 3.7 and 3.8) to build tiled spherical interconnects based on the fundamental tiles introduced in Figure 3.10.

Table 3.1: Number of nodes in a hexagonal interconnect. For the same symmetry and degree, the Class B topologies have three times as many nodes as Class A. Note also that $OA1$ is a cube, $IA1$ is a dodecahedron, and $IB1$ is a buckminsterfullerene.

Degree	**Symmetry and Class**							
	Periodic		*Tetrahedral*		*Octahedral*		*Icosahedral*	
	A	B	A	B	A	B	A	B
1	6	18	4	12	8	24	20	60
2	24	72	16	48	32	96	80	240
3	54	162	36	108	72	216	180	540
4	96	288	64	192	128	384	320	960
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
k	$6\,k^2$	$18\,k^2$	$4\,k^2$	$12\,k^2$	$8\,k^2$	$24\,k^2$	$20\,k^2$	$60\,k^2$

symmetry (P, T, O, or I), class (A or B), and degree (k). For each case, the number of nodes increases with k^2. For a given symmetry and degree, the Class B graph contains three times as many nodes as the Class A graph.

For general-purpose computing on switchless multiprocessor computer systems, spherical interconnects (§3.3) are not quite as efficient as planar interconnects (§3.2) with either toroidal closure or triply-periodic closure, as spherical interconnects have slightly higher diameters for a given number of nodes. However, as mentioned previously, spherical interconnects are poised to make a major impact for the specific problem of solving PDEs on the sphere, which is an important class of computational grand challenge problems central to a host of important questions related to global weather forecasting, climate prediction, and solar physics. In fact, some of the largest purpose-built supercomputers in the world are dedicated to such applications, and thus it is logical to design some computational clusters with switchless interconnects which are naturally suited to this particular class of applications.

Some of the graphs developed here lend themselves particularly well to efficient numerical solution of elliptical PDEs via the iterative geometric multigrid method; Figure 3.12 illustrates six graphs that may be used for such a purpose ($OA2^p$ for $p = \{0, 1, \cdots, 5\}$). Note that the Octahedral symmetry of these grids introduces a key property missing from the Tetrahedral and Icosahedral families of spherical interconnects: due to the square defect regions of the Octahedral graphs,

a red/black ordering of points is maintained over the entire graph. That is, half of the points may be labeled as red and the other half labeled as black, with red points having only black neighbors and black points having only red neighbors. This ordering allows an iterative smoothing algorithm known as Red/Black Gauss Seidel to be applied at each sub step of the multigrid algorithm and facilitates remarkable multigrid convergence rates and efficient scaling on multiprocessor machines to very large grids.

Although the idea of building the interconnect of a multiprocessor computer system in the form of a hexagonal spherical graph is still in its infancy, people have known about and used such graphs to discretize the sphere in the numerical weather prediction community for many years; see, e.g., [5], [53], and [56]. In particular, the model developed by the Deutscher Wetterdienst (DWD) is an example of an operational meteorological code that implements the dual of a hexagonal graph with icosahedral symmetry (see [45]). These previous investigations have almost exclusively used graphs with Icosahedral symmetry, as such graphs undergo the least deformation when being projected from the polyhedron (on which they are built) onto the sphere (where they are applied). Minimizing such graph deformation is beneficial for improving accuracy in numerical simulations. However, as mentioned above, our research indicates that Octahedral graphs, which admit a valuable red/black ordering not available in Icosahedral graphs, might, on balance, prove to be significantly better suited in many applications.

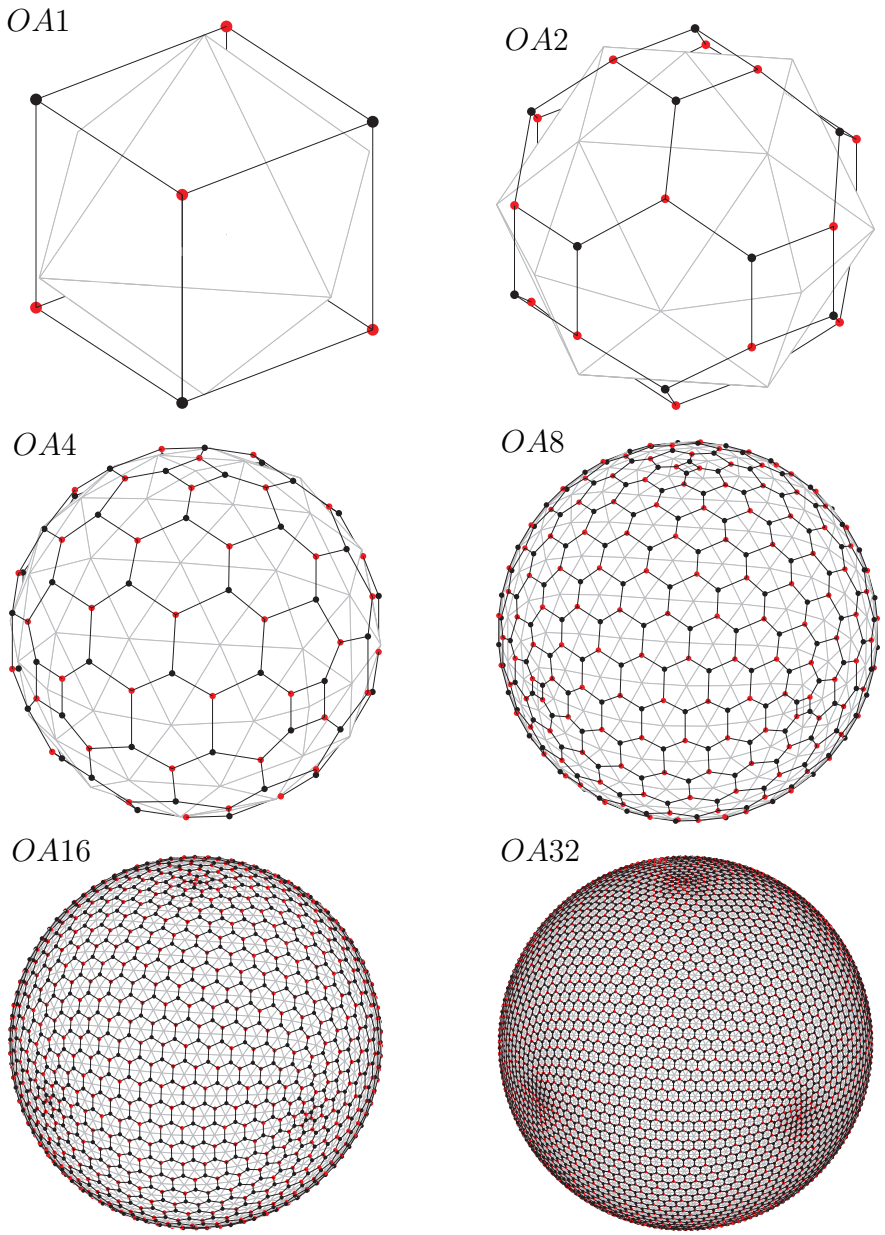

Figure 3.12: A representative family of spherical interconnects. All six graphs are members of the $OA2^p$ family (three of the square "defect" regions are visible in each graph). From top-left to bottom-right the degree is $2^0, 2^1, 2^2, 2^3, 2^4$, and 2^5. The thick black lines show the logical interconnect. The light gray lines illustrate the respective Voronoi cell for each node, indicating the region of the physical domain that each node is responsible for in a PDE simulation on a sphere.

Tiling the spherical closure. As illustrated in Figure 3.12, spherical graphs are quite complicated; their practical deployment for large N would be quite difficult without a scalable tiling strategy. Thus, in a manner analogous to that used in §3.3.1 and §3.3.2, we now transform these graphs to discover local patterns and identify self-similar tilings that make such constructions tractable and scalable.

To illustrate the process, consider first the $TB2$ interconnect as a representative example, as depicted in Figure 3.13. Note first that we replace the "folding" idea used in §2, §3.1, and §3.2 with a "stretching" and "grouping" strategy. The stretching step may be visualized by imagining all links in the spherical graph under consideration as elastic, then imagining the transformation that would result from grabbing all of the nodes near one of the vertices of the polyhedron, or near the center of one of the faces of the polyhedron, and pulling them out to the side (while maintaining the connectivity) until the resulting stretched graph may be laid flat. The resulting spider web-like structure is known as a Schlegel diagram.

For the $TB2$ Schlegel diagram in the top-right of Figure 3.13, the outer edge of the diagram is a triangle that corresponds to the nodes near one of the vertices of the tetrahedron of the unstretched graph, and near the center of the diagram is a hexagon that corresponds to nodes near the middle of the opposite face of the tetrahedron.

The next step in tiling a spherical graph is to identify repetitive structure.

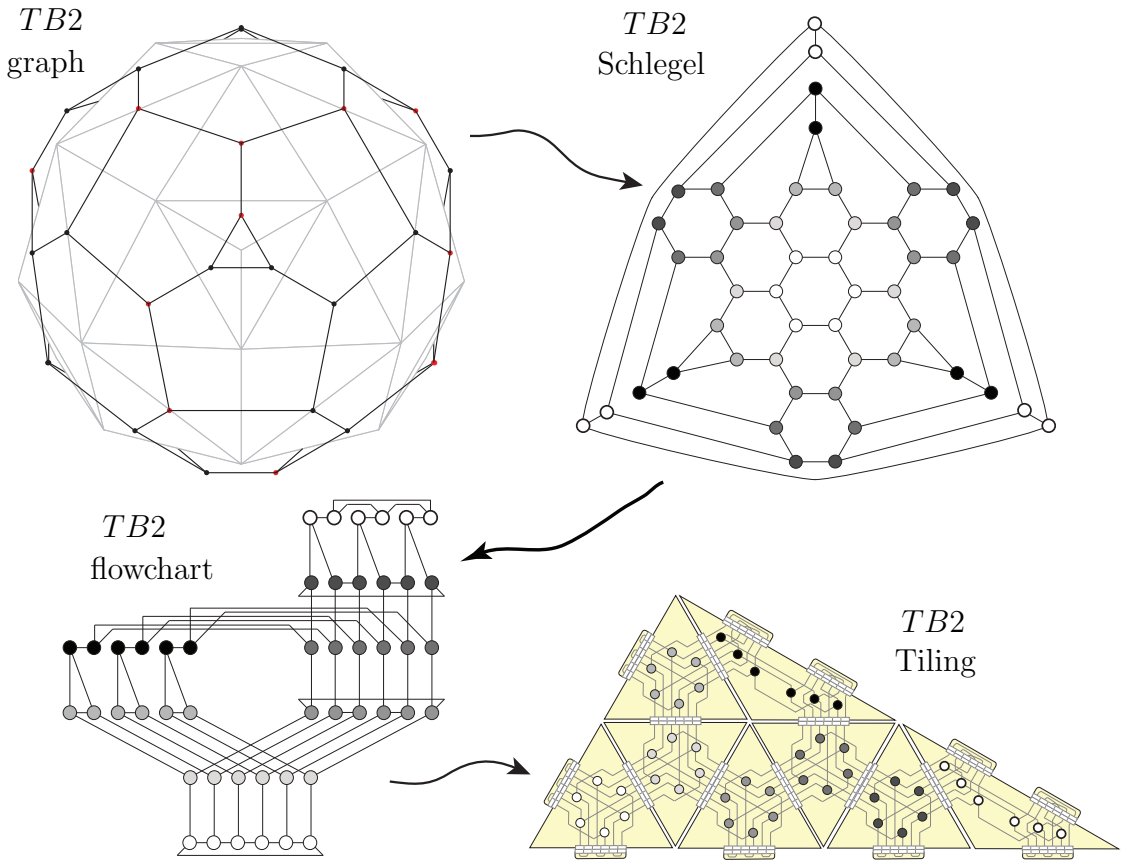

Figure 3.13: In order to determine a tiling for a given spherical interconnect graph (top-left), a Schlegel diagram (top-right) is first constructed by stretching and flattening the graph onto the plane while maintaining the same logical connections. From this, locally similar groups of nodes can be grouped into a revealing "flowchart" (bottom-left) that ultimately makes the tiling evident (bottom-right). In the case of $TB2$ illustrated here, with the exception of the top of each column of the flowchart, the nodes are connected identically to the $PA*$ case, leading to a tiling with the E tiles of Figure 3.10. This tiling is then completed with F tiles from Figure 3.10 along one side, together with the appropriate end caps.

In the Schlegel diagram of Figure 3.13, after considerable deliberation, we have grouped the nodes in sets of six. This grouping is not unique, but it is with the choice shown that a simple tiled structure reveals itself. In general, nodes that are approximately the same distance from the center of the Schlegel diagram tend to be lumped together. From the selected grouping, we can create the flowchart shown on the lower-left of Figure 3.13. In this flowchart, all obvious hexagonal structure is removed, but now the groups of paired nodes are brought together in what will eventually become their tiled form. Significantly, the logical structure of the flowchart (as well as the Schlegel diagram) is identical to the original spherical interconnect. Note that flow from bottom to top on the flowchart corresponds to in-to-out flow in the Schlegel diagram.

With the exception of the top set of nodes in each column of the flowchart, the connections between the other six sets of nodes match those of the $PA*$ case. That is, the first node in each group is connected to the first node in the neighboring groups, etc. As a result, we can use the same E tiles (see Figure 3.10) to construct this part of the graph as were used in the $PA*$ constructions. The remaining two sets of nodes can be incorporated using the F tile used in the $PB*$ constructions. After making simple end connections, the completed tiling is shown on the bottom-right of Figure 3.13. In this case, the overall tiling builds out to a right triangle.

For each interconnect family, we perform a similar procedure for the first few graph degrees until a pattern emerges, first stretching/flattening the graph to

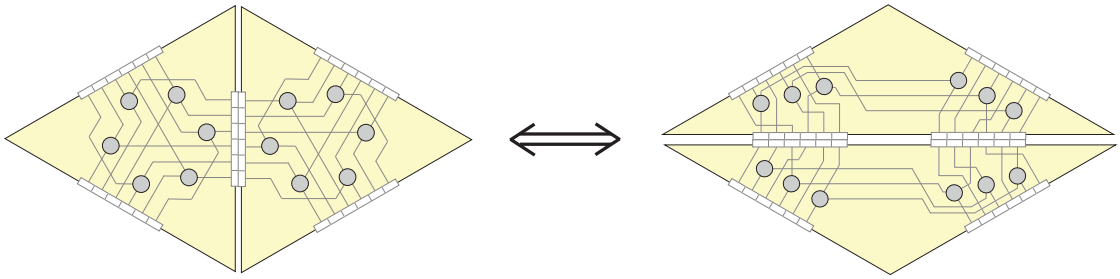

Figure 3.14: An equivalence between a pair of E tiles and a pair of F tiles. Typically, we use the E tiles whenever possible; however, the equivalence between these two pairings helps to unify the Class B tilings of the various different families.

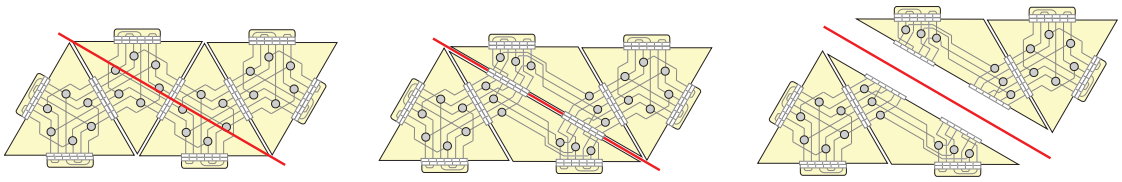

Figure 3.15: For any of the $OB*$ family of tilings (here, $OB1$ is shown), we can replace the E tiles along the shorter diagonal using the equivalence depicted in Figure 3.14. This allows us to identify the bifurcation axis that splits the tiling into two smaller tilings in the $TB*$ tilings of the appropriate degree.

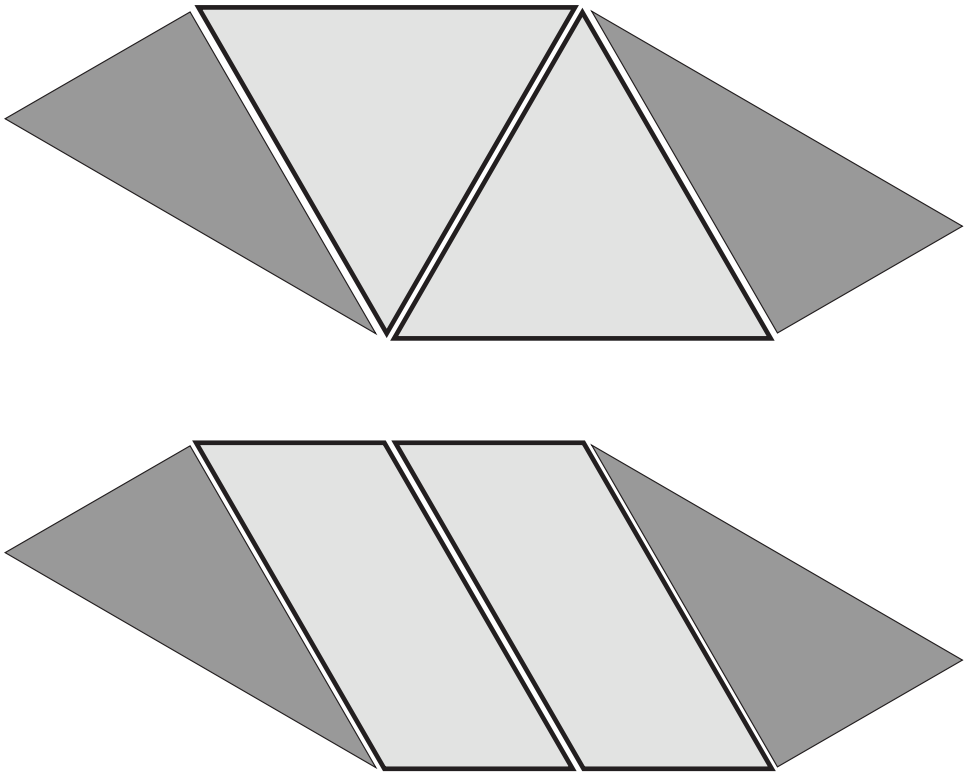

Figure 3.16: (*top*) The IBn graph may be subdivided into four smaller tilings [two TBn and two PBn] via a series of simple switches applied to the electrical connections between the tiles along the three diagonals shown; a similar subdivision to four smaller tilings [two TAn and two PAn] may also be accomplished with the IAn tiling after the transformation illustrated in Figure 3.14 is applied along the appropriate diagonals. (*bottom*) The IBn tiling, for n even, may also be subdivided a different way [to two TBn and two $OA(n/2)$]; a similar subdivision [to two TAn and two $OB(3n/2)$] may also be accomplished with the IAn tiling, for n even, after the transformation in Figure 3.14 is applied appropriately. Such subdivisions are useful for running four small jobs on the cluster simultaneously.

make a Schlegel diagram, then identifying repetitive local structure by grouping the nodes in an in-to-out flowchart of the Schlegel diagram, and finally defining a tiling to facilitate scaling. Though this was an involved process, by so doing, we have discovered the simple and easily scaled families of tilings depicted in Figures 3.21 - 3.25.

Interesting and surprising correlations exist between the various simple tilings which we have discovered. We first observe that each symmetry family shares a common overall shape: the $T**$ family builds out to right triangles, the $O**$ family builds out to parallelograms, and the $I**$ family builds out to "six-sided parallelograms".

By identifying an equivalence between a pair of E tiles and a pair of F tiles, as depicted in Figure 3.14, we are also able to identify interfamily relationships between the tilings as well. Applying this substitution along the shortest diagonal of the $OB*$ family, as illustrated in Figure 3.15, shows immediately how the $OB*$ tiling can be built up from two tilings of the $TB*$ family. Equivalently, the $OA*$ family can be built from two $TA*$ tilings of the appropriate degree.

Similarly, as illustrated in Figure 3.16, the $I**$ family of tilings can be built up from two $T**$ tilings and either two $P**$ tilings or two $O**$ tilings. Such subdivisions could be useful for running four smaller jobs on the cluster simultaneously without reconfiguration.

We have focused in §3.3.3 on applying the triangular Class A and Class B

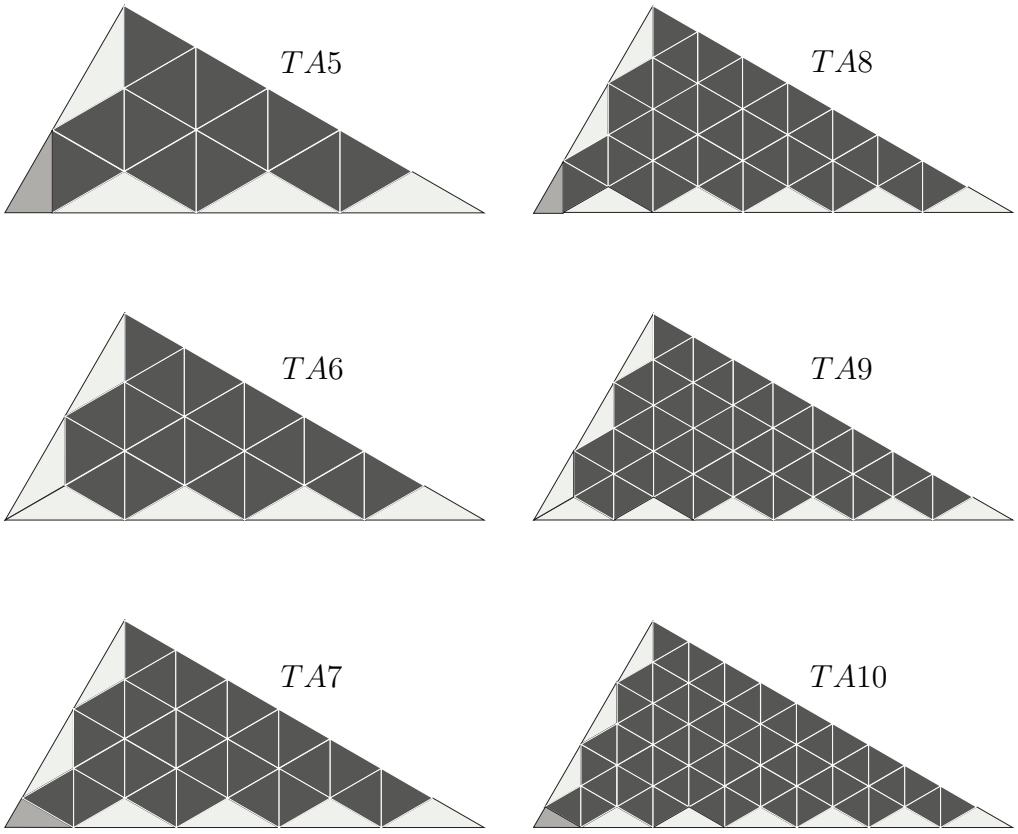

Figure 3.17: The pattern for the $TA*$ interconnects is not immediately apparent, so we have included the next six tilings for clarity. Note that the longest leg has an obtuse tile at the point and enough equilateral triangles to bring the total tile count along that edge equal to the degree of the tiling. By allowing non-equilateral tiles only along the edges of the tiling, the resulting tiling is unique for each degree. This pattern will reemerge as we look at interconnects with octahedral and icosahedral symmetry.

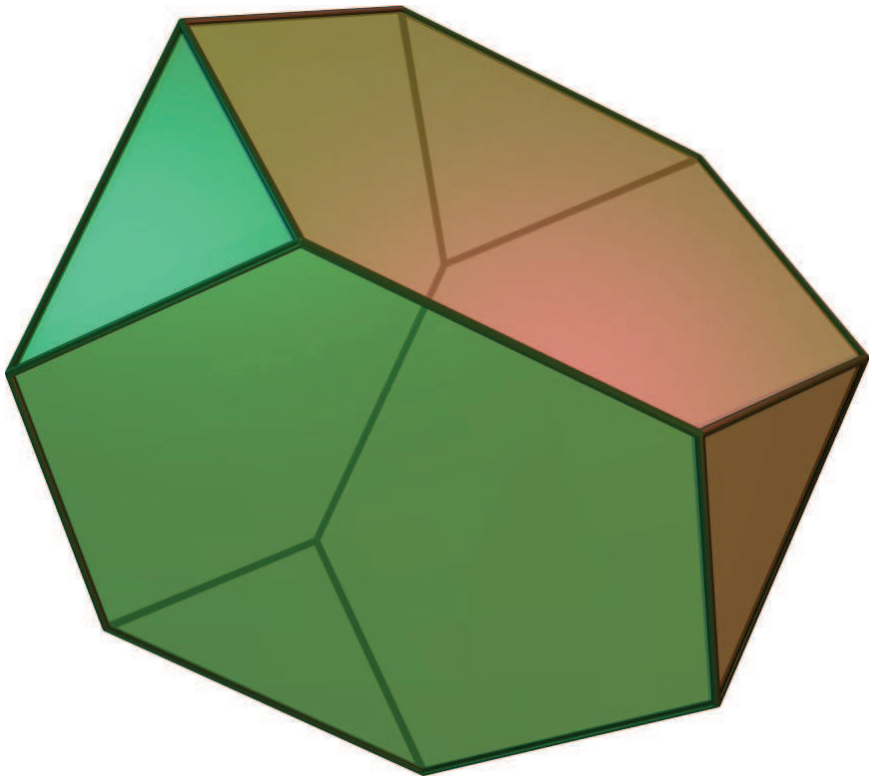

Figure 3.18: The truncated tetrahedron, the only Archimedean solid with only triangular and hexagonal sides.

structures from Figures 3.7 and 3.8 to the triangular faces of three of the Platonic solids (Figure 3.11). However, other constructions are also possible. For instance, the truncated tetrahedron (Figure 3.18) has only hexagonal and triangular sides, and thus can be covered with triangular Class A or Class B structures on its triangular faces and six Class A or Class B structures (see Figure 3.9) on its hexagonal sides. The resulting Class A graphs have $28k^2$ nodes, and the resulting Class B graphs have $84k^2$ nodes (cf. Table 3.1). These graphs each have 12 pentagonal faces embedded within an otherwise hexagonal graph, just like the icosahedron; however, they are slightly less uniform and do not admit a red/black ordering of points—it is currently unclear whether or not such alternative constructions have any significant advantages.

3.4 Summary

The present line of research considers the topology of the interconnection of processing elements in switchless multiprocessor computer systems. The design of such switchless interconnects is a problem that has been considered for decades; however, the question of scalability of such designs is of heightened importance today, as modern computer systems take parallelization to new levels. Note that the three fastest computer systems in the world today each has hundreds of thousands of nodes, whereas typical computer clusters in academic and industrial settings are

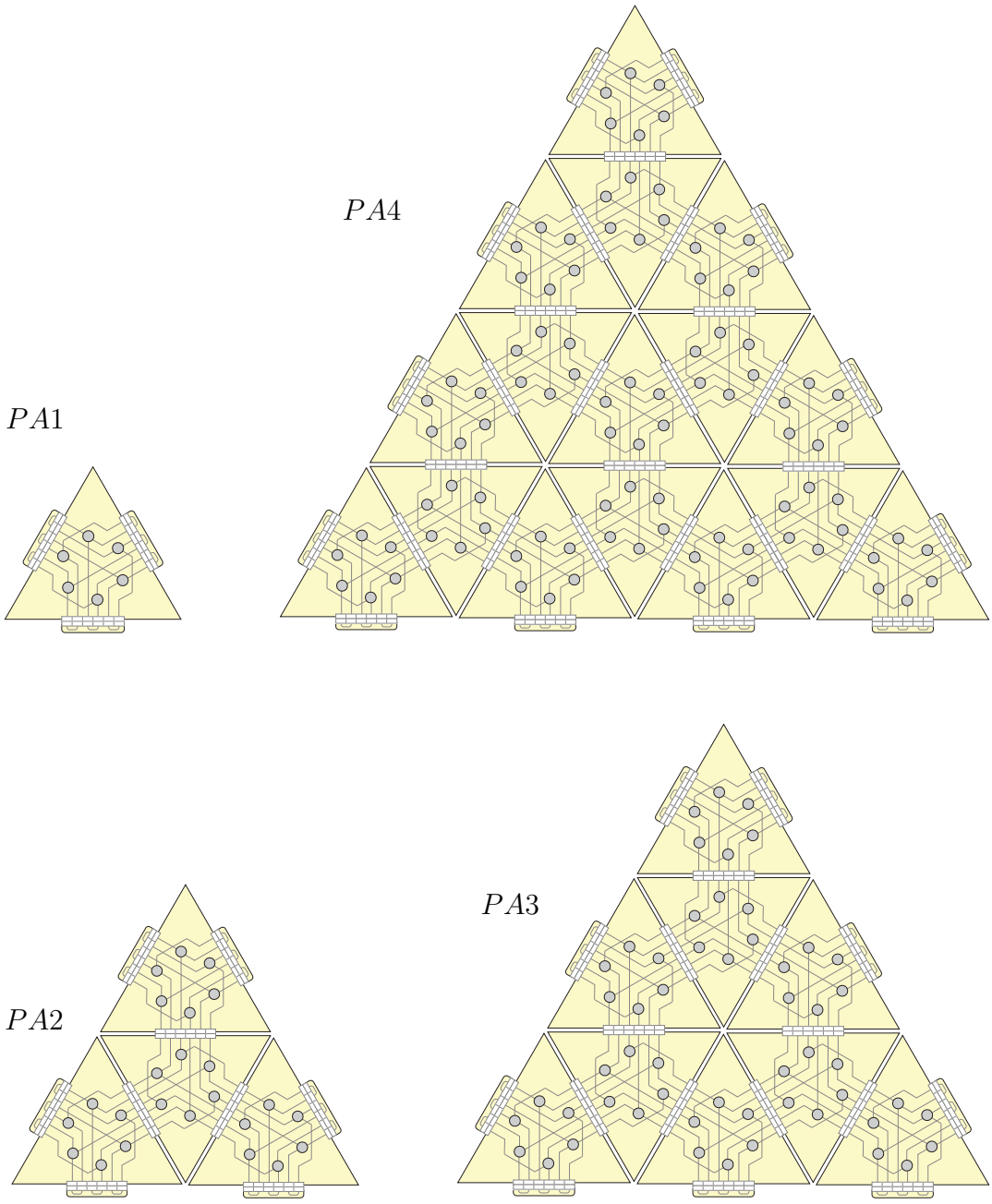

Figure 3.19: The first four members of the $PA*$ family of interconnects; each builds out to an equilateral triangle. Note that the $PA*$ construction illustrated here is built using E tiles (see Figure 3.10).

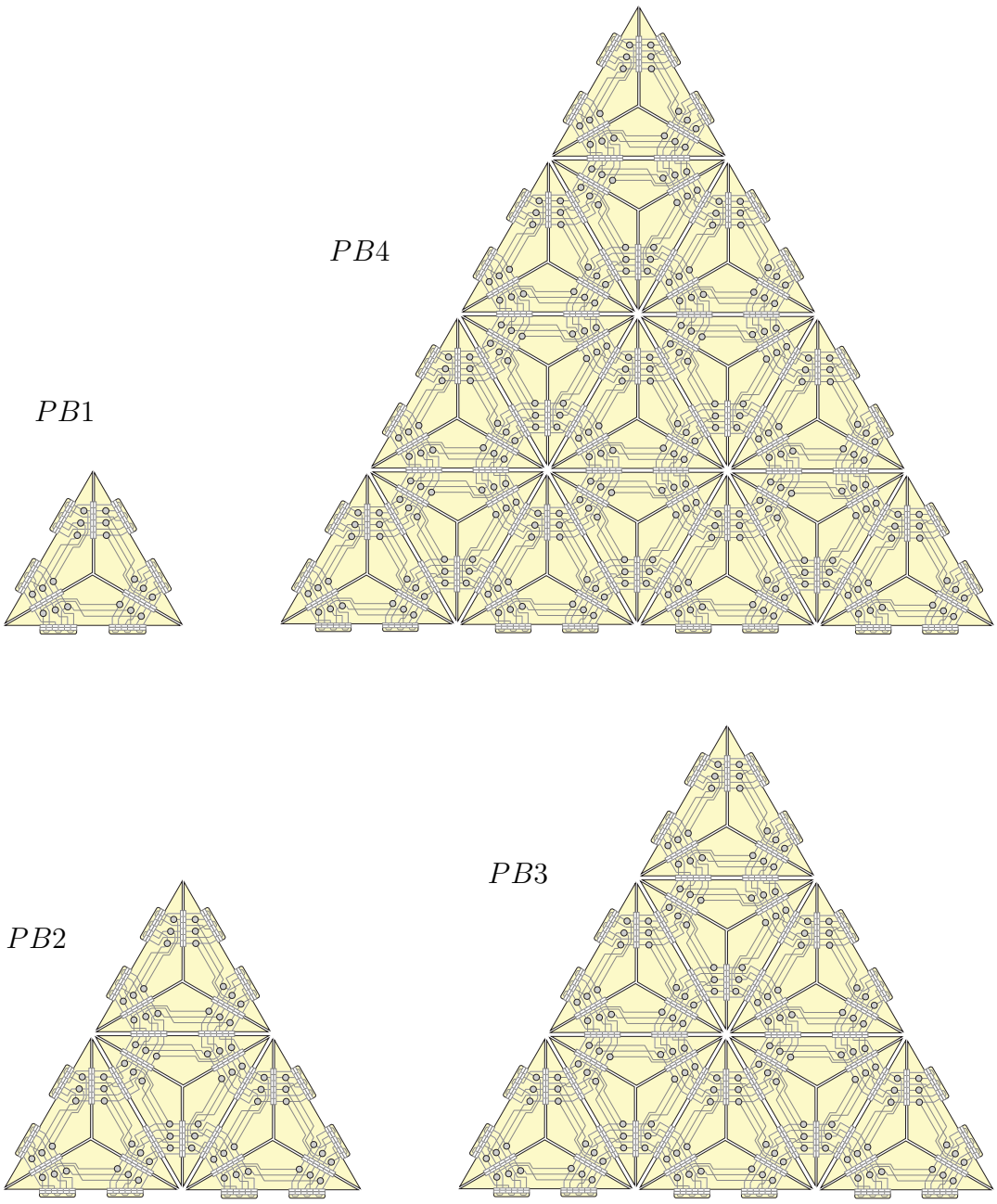

Figure 3.20: The first four members of the $PB*$ family of interconnects; each builds out to an equilateral triangle. Note that the $PB*$ construction illustrated here is built using F tiles (see Figure 3.10).

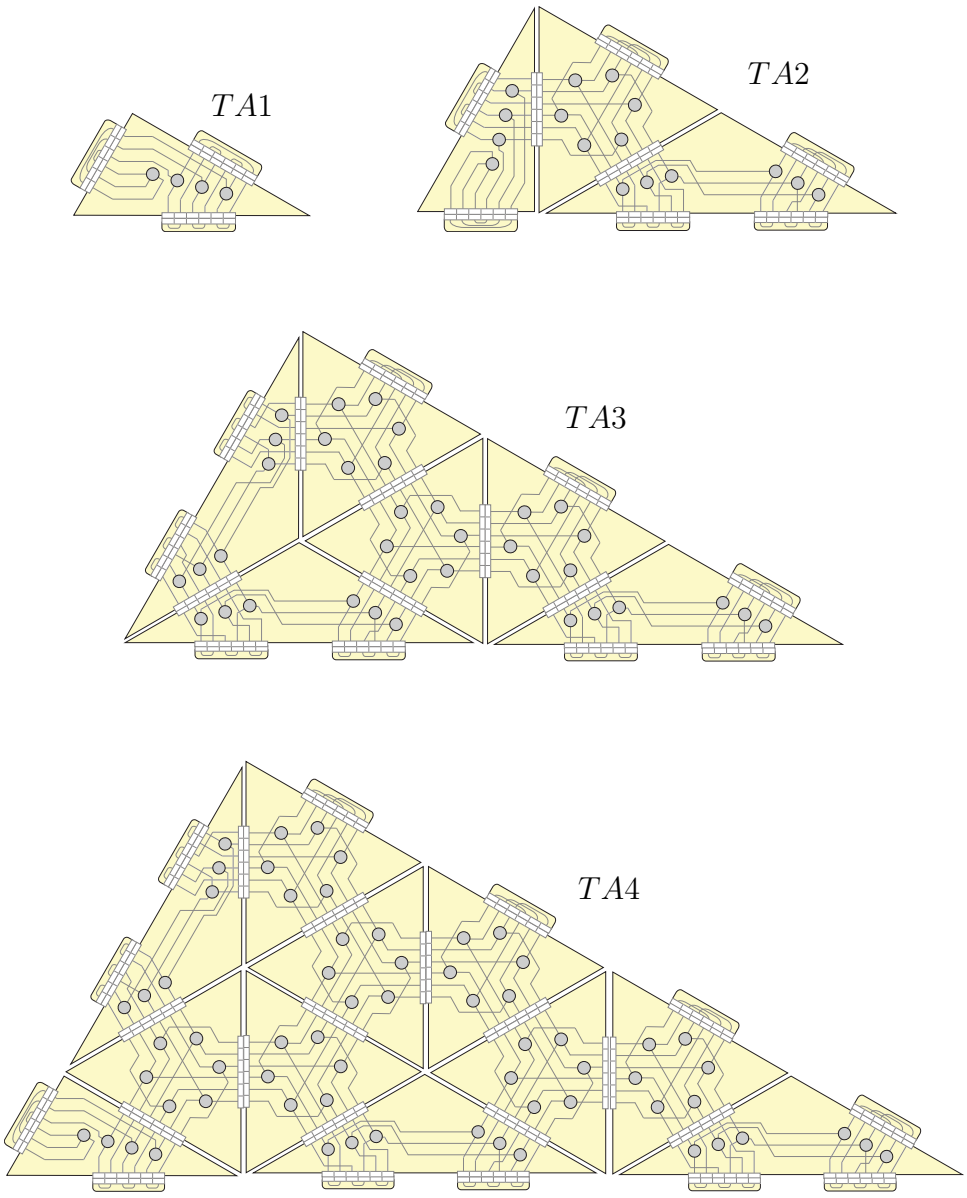

Figure 3.21: The first four members of the $TA*$ family of interconnects; each builds out to a right triangle.

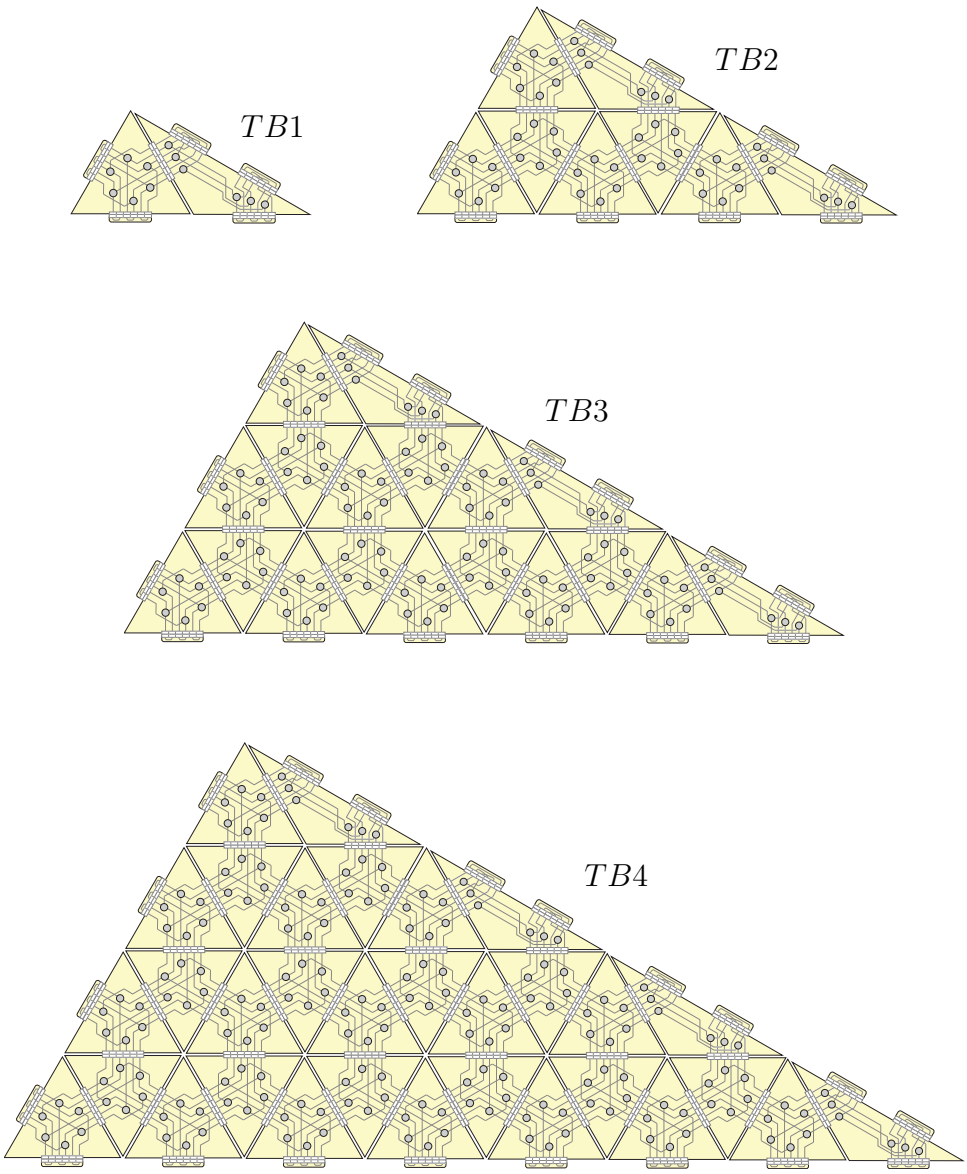

Figure 3.22: The first four members of the $TB*$ family of interconnects; each builds out to a right triangle.

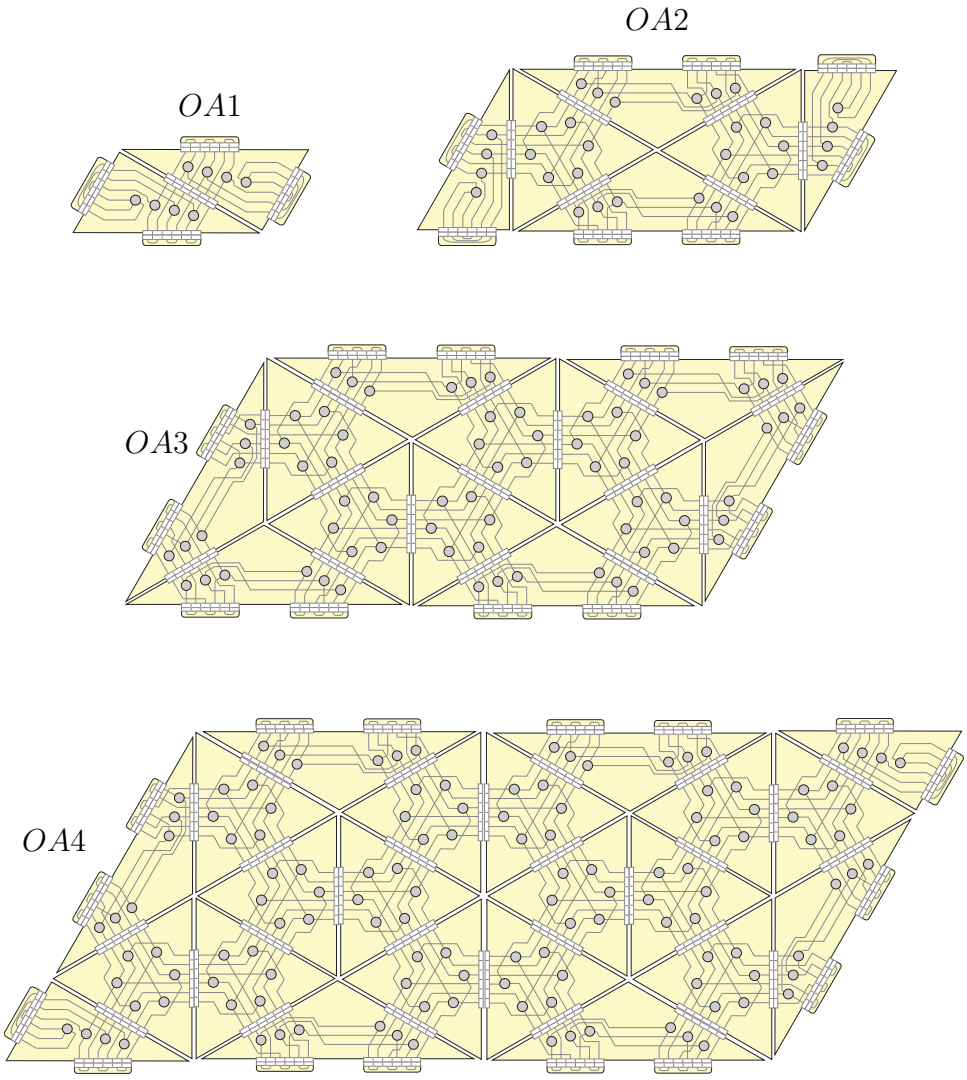

Figure 3.23: The first four members of the $OA*$ family of interconnects; each builds out to a parallelogram. This family may be constructed by connecting two $TA*$ tilings along the longest leg, as described in Figure 3.15.

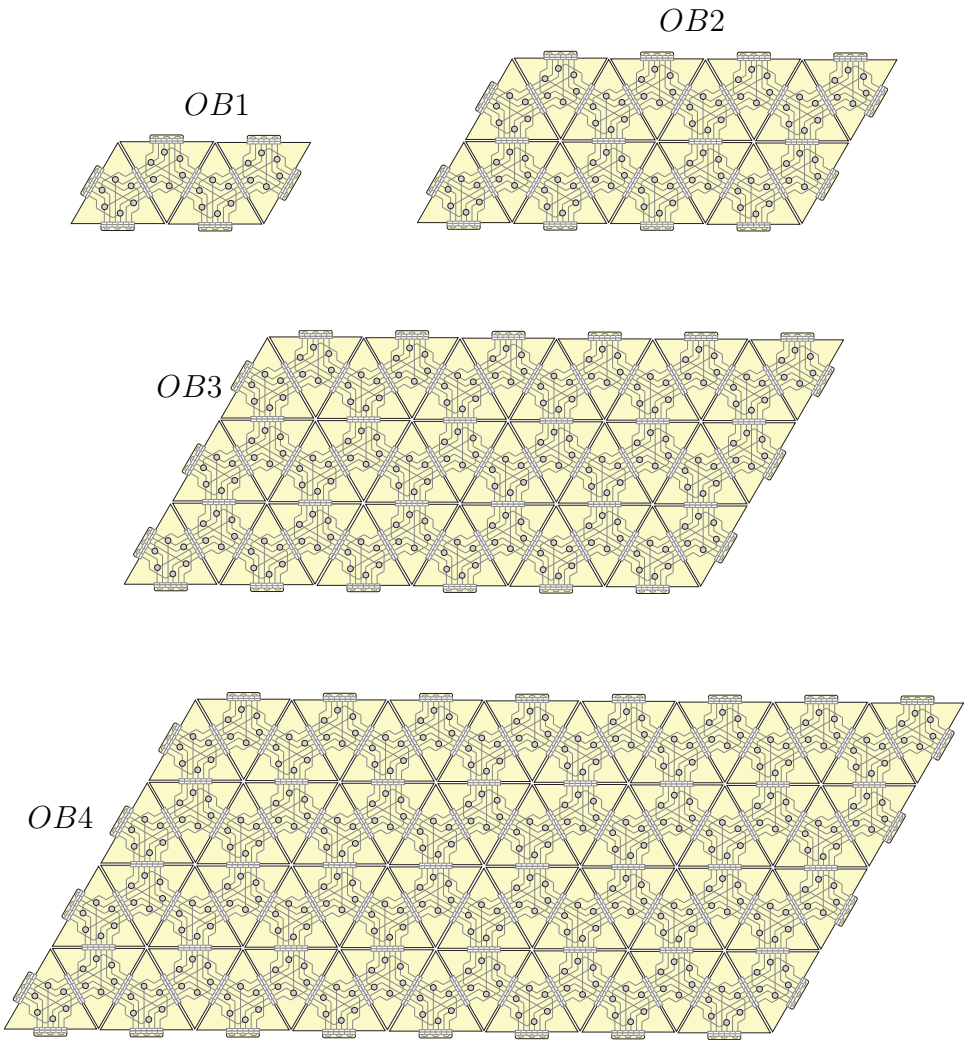

Figure 3.24: The first four members of the $OB*$ family of interconnects; each builds out to a parallelogram. This family may be constructed by connecting two $TB*$ tilings along the longest leg, as described in Figure 3.15.

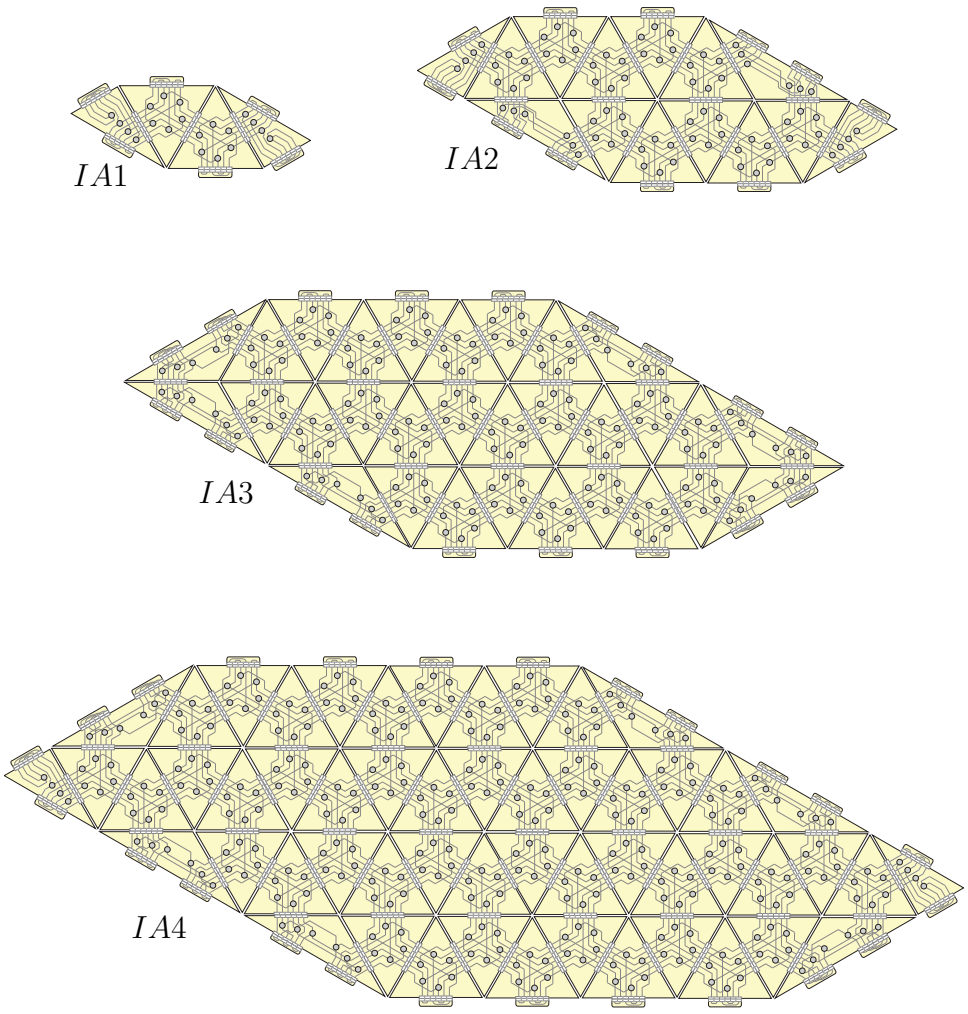

Figure 3.25: The first four members of the $IA*$ family of interconnects; each builds out to a "six-sided parallelogram". This family may be constructed by connecting two $TA*$ and two $PA*$ tilings, or two $TA*$ and two $OB*$ tilings, as described in Figure 3.16.

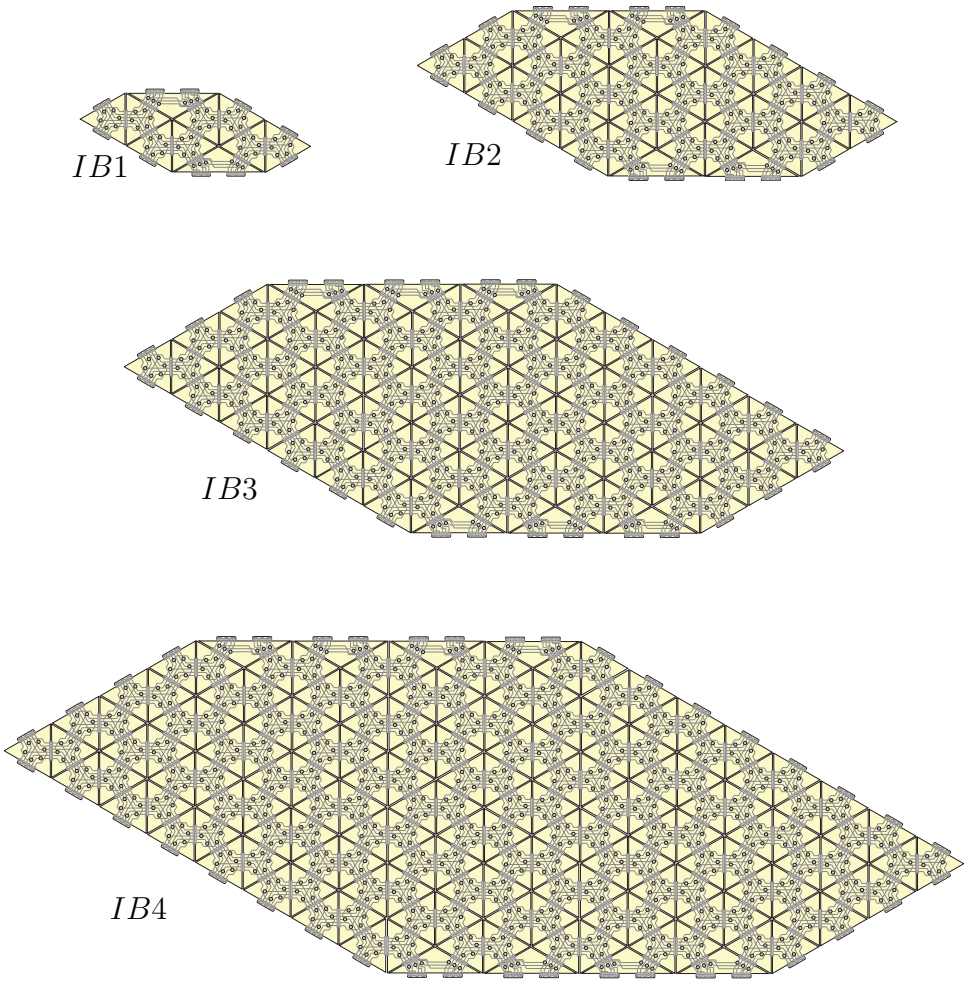

Figure 3.26: The first four members of the $IB*$ family of interconnects; each builds out to a "six-sided parallelogram". This family may be constructed by connecting two $TB*$ and two $PB*$ tilings, or two $TB*$ and two $OA*$ tilings, as described in Figure 3.16.

today growing from thousands to tens of thousands of nodes. It is for this reason that, we believe, a revisiting of the topology used in switchless multiprocessor computer systems is in order, and special-purpose topologies for clusters built for special-purpose applications, such as global weather forecasting, are warranted.

The present paper focused on the optimal ways to cover a sphere with a fine hexagonal grid. Euler's formula leads to a predictable number of "defects" (non-hexagons) within the otherwise hexagonal grid; the three choices that may be developed with maximal uniformly incorporate 4 triangles (distributed at the corners of a Tetrahedron), 8 squares (distributed at the corners of an Octahedron), or 12 pentagons (distributed at the corners of an Icosahedron) into the otherwise hexagonal grid. The present work systematically examined all three of these cases (denoted $T**$, $O**$, and $I**$), and discovered two convenient families of tilings for each case (denoted $*A*$ and $*B*$). These tilings are repetitive structures, which greatly eases their scaling to large cluster sizes ($**n$ for $n \gg 1$), and contain no long wires, thereby facilitating fast link speeds and reduced cluster cost and complexity even as the cluster size scales to hundreds of thousands of nodes.

In the resulting switchless interconnect designs, the *physical proximity* of the cells created (in the PDE discretization on the sphere) and the *logical proximity* of the nodes to which these cells are assigned (in the computational cluster) coincide perfectly, so all communication between physically adjacent cells during the PDE simulation require communication over just a single hop in the compu-

tational cluster. Such an interconnect should provide near-perfect scaling with cluster size in operational problems; that is, refining the overall grid by a factor of n while also increasing the overall cluster size by a factor of n will result in an essentially unchanged execution speed. Such scaling is the holy grail of parallel computing, and is enabled in the present case by the careful design of a computational interconnect topology that is well matched to the physical problem to which the computational cluster is dedicated.

3.5 Acknowledgements

This chapter, in full, is a reprint of the material as it appeared in:

Cessna, J. and Bewley, T. (2009) Structured computational interconnects. *11^{th} IEEE International Workshop on System-Level Prediction*, San Francisco, CA, USA.

The dissertation author was the primary investigator/author of this publication.

Chapter 4

Three Classes of Directed Graphs

Joseph Cessna and Thomas Bewley

Abstract. With directed graph topologies, such as those used in FPGA applications, two competing design constraints are the ability to spread information across the graph in a global sense versus the requirement to have path redundancy and local (nearest neighbor) communication. For information spread, the general class of butterfly graphs are used because they optimally connect each node to every other node in a minimal manner. As a consequence, butterfly graphs lack any path diversity or sense of locality without the addition of subsequent stages. In the other extreme, cartesian-based graphs contain much local structure and high path redundancy. However, they do this by relinquishing the ability to quickly

communicate across the graph. This paper attempts to generalize these two classes of directed graphs into a unified theory in which the cartesian and butterfly graphs are special cases of a more general class of interconnect that better spans the design parameter space. We seek not to suggest a correct solution, but rather provide useful trade-offs between the desired properties that will necessarily vary from application to application.

4.1 Background & Terminology

Regular directed graphs consist of a set of nodes containing an equal number of emanating and terminating directed links. These type of graphs are useful for general information theory, and, more specifically for field-programmable gate array (FPGA) applications. In FPGAs, a large amount of parallel information is fed through the graph, one stage at a time, to efficiently solve a global problem. A better understanding of the design trade-offs of these graphs with respect to information spread and path redundancy could help lead to more efficient, problem-specific designs.

The directional graphs we will consider are derived from more general $(n+1)$-dimensional topologies that contain a flow direction $\mathbf{x}_0$, along which we can locate discrete sets of nodes lying in the sequence of orthogonal n-dimensional hyperplanes. We think of information flowing forward, along this direction, passing, in turn, through each orthogonal hyperplane (or *stage*) of the graph. For finite graphs, each stage contains the same number of nodes. We can then define the *total cardinality* of the graph as $(N_0 \times N_1 \times \cdots \times N_n)$, where N_0 is the number of stages in the graph, $M = (N_1 \times \cdots \times N_n)$ is the number of nodes in each stage, and N_k is the k^{th} *cardinality* (i.e. the number of nodes in the k^{th} dimension of each stage). Thus, the *size* (total number of nodes) of the graph is $N_0 \times M$.

Necessarily, a node in a given stage must only connect to nodes in the

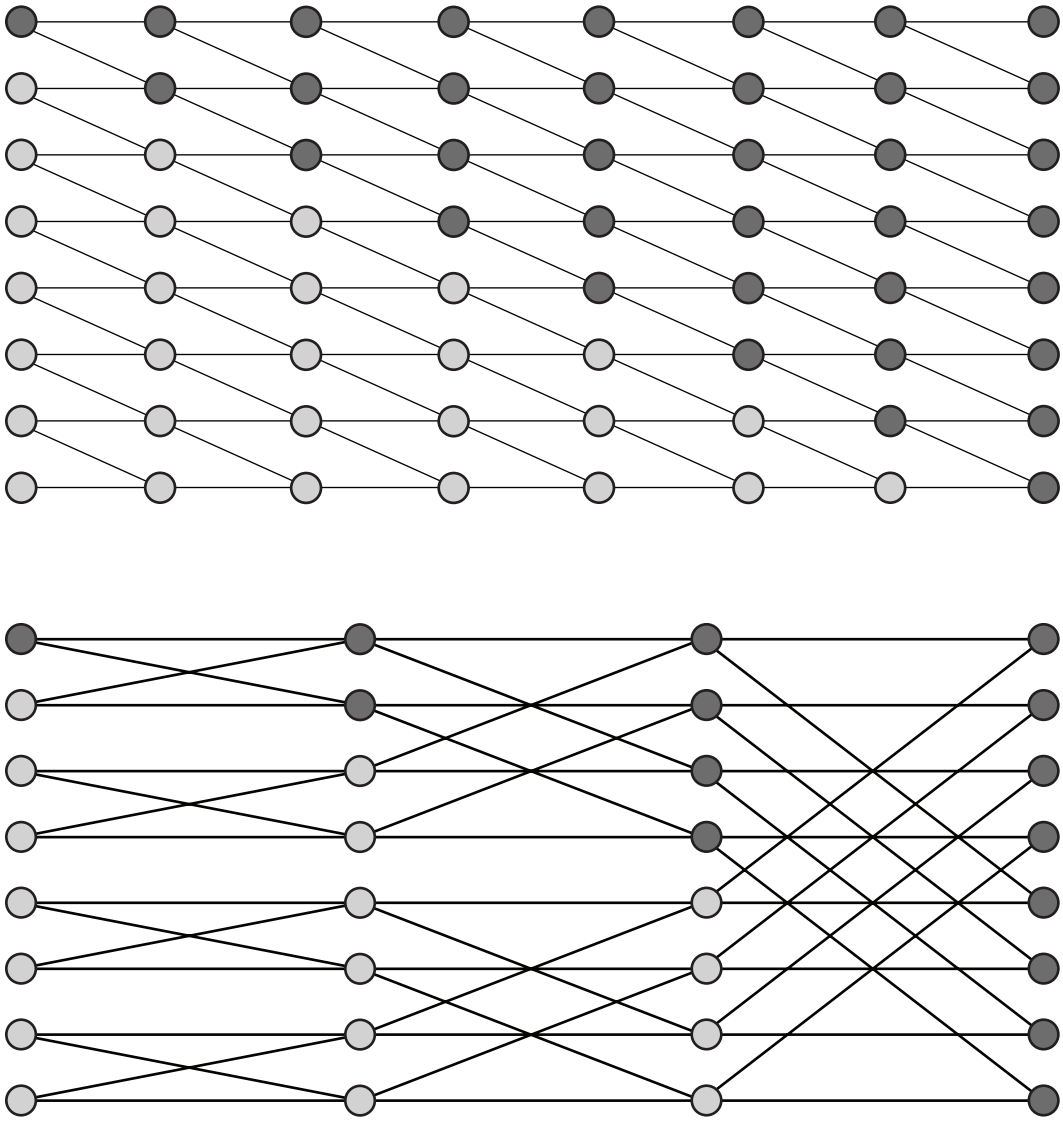

Figure 4.1: Two traditional directed graphs are shown; information is assumed to flow from left to right. The 2D cartesian graph (top, 8×8) and the *2-ary 4-fly* (bottom, $4 \times 2 \times 2 \times 2$) represent the opposite extremes with regards to performance characteristics. The cartesian graph contains exclusively local links, and consequently has a lot of path diversity. In contrast, the butterfly graph spreads information across the nodes much more efficiently, but lacks any sense of locality from stage to stage and contains no path diversity.

immediately preceding and following stages; it may not connect to any nodes in its same stage. The connection between stages is governed by the overall topology of the graph and will vary from stage to stage in a repeating pattern. The number of distinct connections between stages is a function of the dimension n of the stages themselves. For the directed graphs of interest, a node will connect to exactly s nodes in both the forward and backward direction. Thus, s defines the degree of the graph. For this paper, we will focus exclusively on directed graphs of degree $s = 2$ and $s = 3$.

A node in a given stage connects to its s forward neighbors along a *channel*. Two nodes are said to be connected if there exists an ordered set of channels *in the flow direction only* (i.e. a *path*) that connects a node in one stage to another node in a subsequent stage. It is important to note that paths only move in the flow direction. Redundant paths occur when two nodes can be connected by more than one unique path. These redundant paths are important because they decrease bottlenecks in the graph and thus increase the graph's overall robustness. A graph with many redundant paths is usually efficient at performing local data operations such as simple comparisons or additions.

On the other hand, an increase in path redundancy necessarily is associated with a decrease in the *spread* of a graph. The spread is a measure of how quickly information from one node reaches the other nodes in subsequent stages. *Saturation* occurs when the information from one node spreads to every node in a future stage.

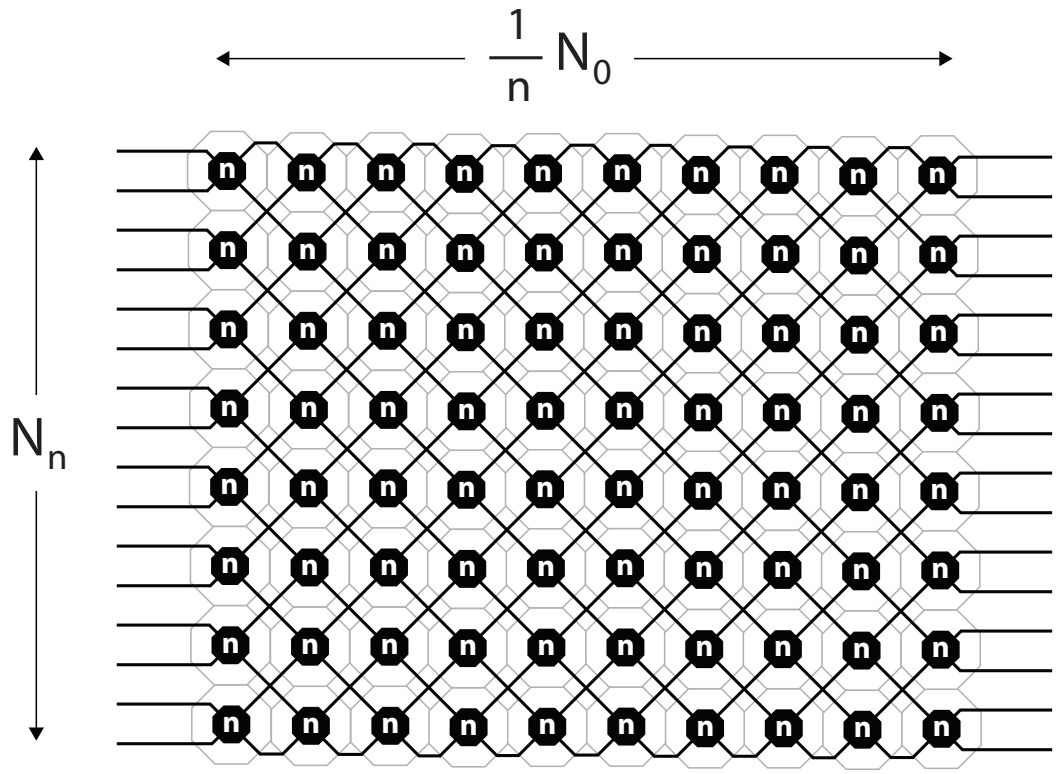

Figure 4.2: The generalized degree-2 directed graph. Each stage is n-dimensional, and the total cardinality is $(N_0 \times N_1 \times \cdots \times N_n)$. When $n = 1$ this graph reduces to the 2D cartesian graph and when $N_k = 2, \ \ \forall k > 0$, this graph reduces to the well-known *2-ary* butterfly. The full connectivity is defined recursively in Figure 4.3.

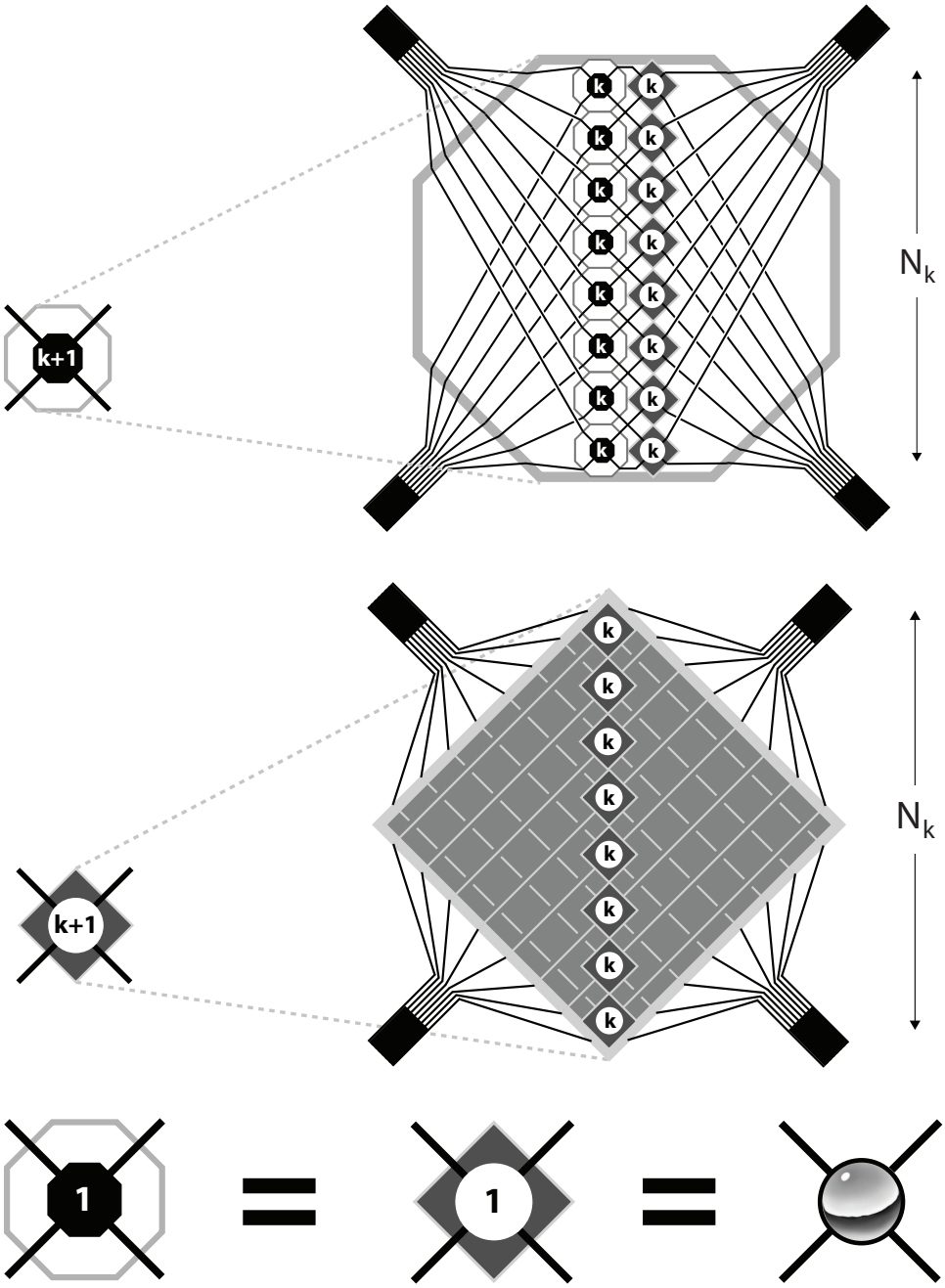

Figure 4.3: The generalized degree-2 directed graph is best graphically defined recursively through the combination of the two basic building blocks shown above. At the lowest level, these building blocks reduce to a single node.

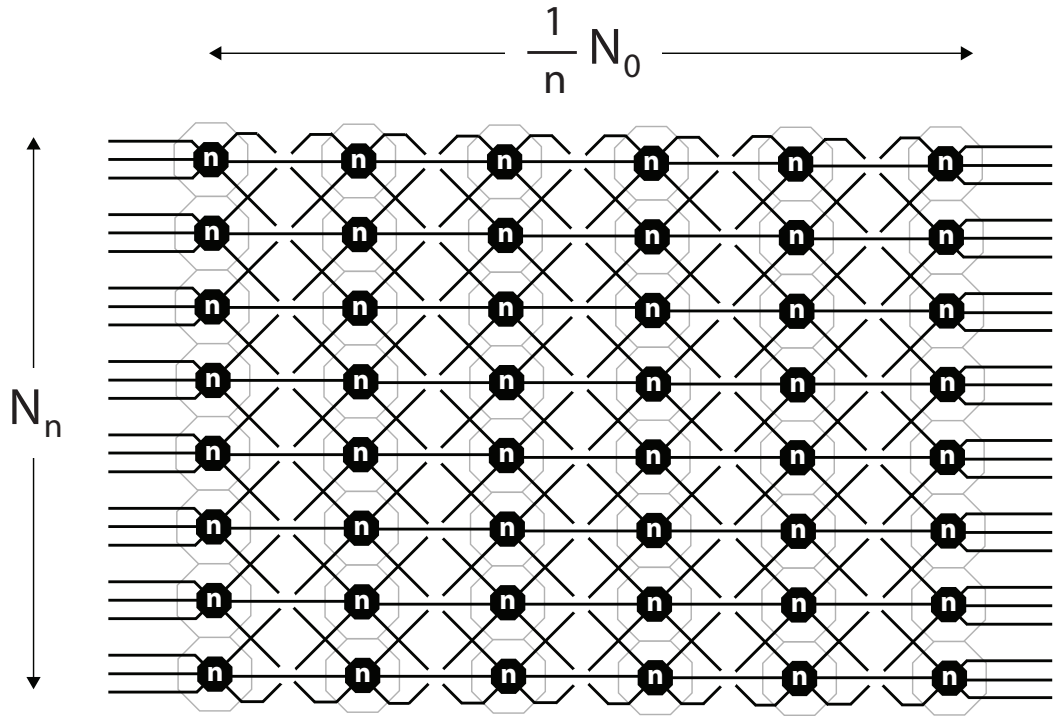

Figure 4.4: The generalized degree-3 directed graph. Each stage is n-dimensional, and the total cardinality is $(N_0 \times N_1 \times \cdots \times N_n)$. When $N_k = 3, \ \ \forall k > 0$, this graph reduces to the well-known *3-ary* butterfly. The full connectivity is defined recursively in Figure 4.5.

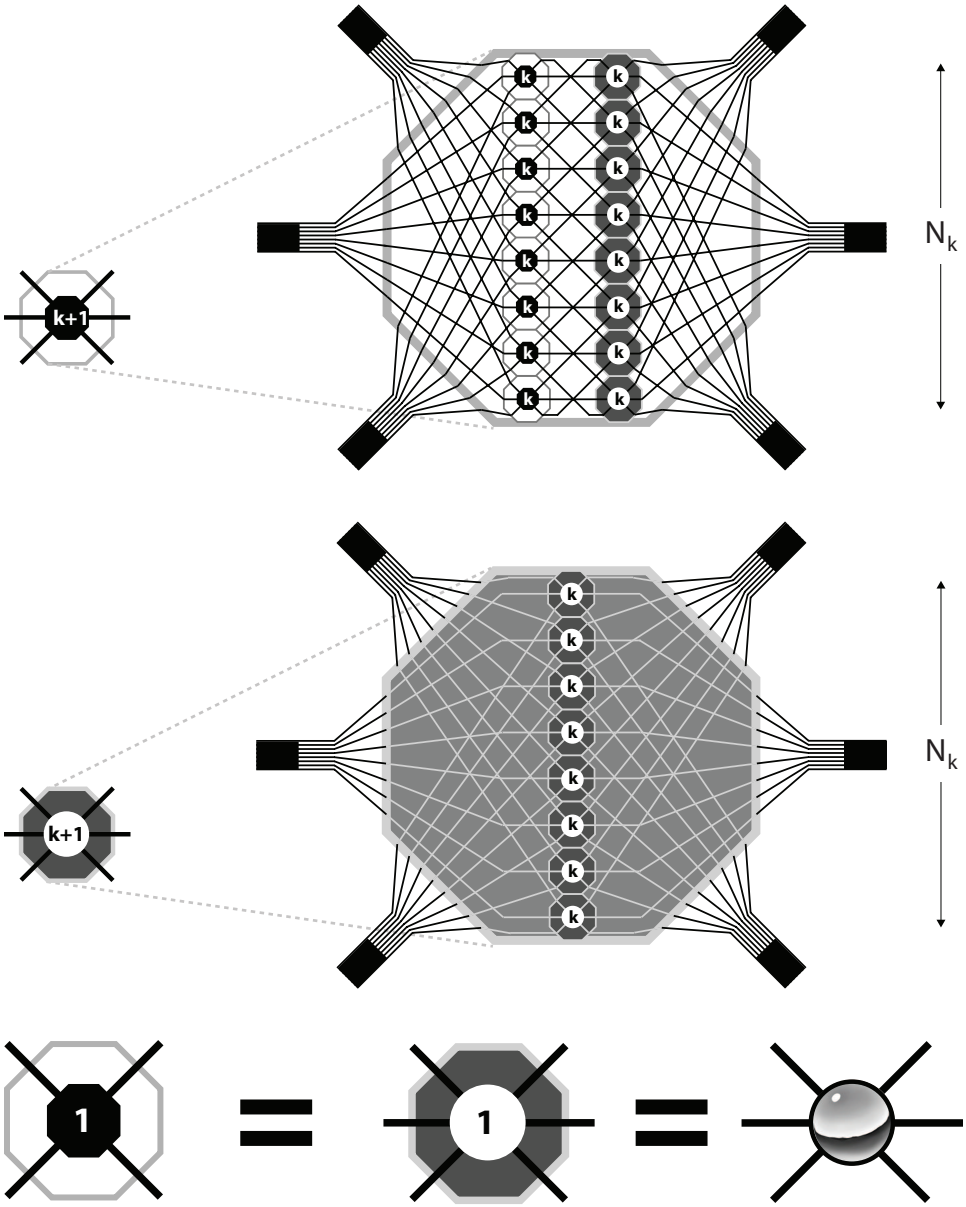

Figure 4.5: The generalized degree-3 directed graph is best graphically defined recursively through the combination of the two basic building blocks shown above. At the lowest level, these building blocks reduce to a single node.

The *saturation length* is the number of traversed stages required until saturation is achieved. A graph with a low saturation length has a high communication rate across all nodes and thus can efficiently perform global data operations such as data transposes.

4.2 General Classes

A 2D cartesian graph (see top of Figure 4.1) has nodal degree $s = 2$ and contains $N_0 = q$ stages with dimension $n = 1$. The total cardinality of the graph is $(\, q \times p \,)$, there are $M = p$ nodes per stage, and the size of the graph is $q\,p$. The cartesian graph has a high saturation length of $q - 1$ when compared to a comparable butterfly graph due to the high number of redundant paths. These paths are a byproduct of the local nature of the cartesian interconnect.

In contrast, a traditional *p-ary q-fly* butterfly graph[16] (see bottom of Figure 4.1) has nodal degree $s = p$ and contains $N_0 = q$ stages with dimension $n = q-1$. The total cardinality of the graph is $(\, q \times p \times \cdots \times p \,)$, there are $M = p^{\,q-1}$ nodes per stage, each flow normal cardinality is equal to the nodal degree $N_k = p$, and the size of the graph is qM. As a result, it can be shown that the butterfly graph provides the minimum saturation length of any graph with stage size $p^{\,q-1}$. However, each node has only one unique path to the nodes of the saturated stage and thus lacks any path diversity (and consequently local nature).

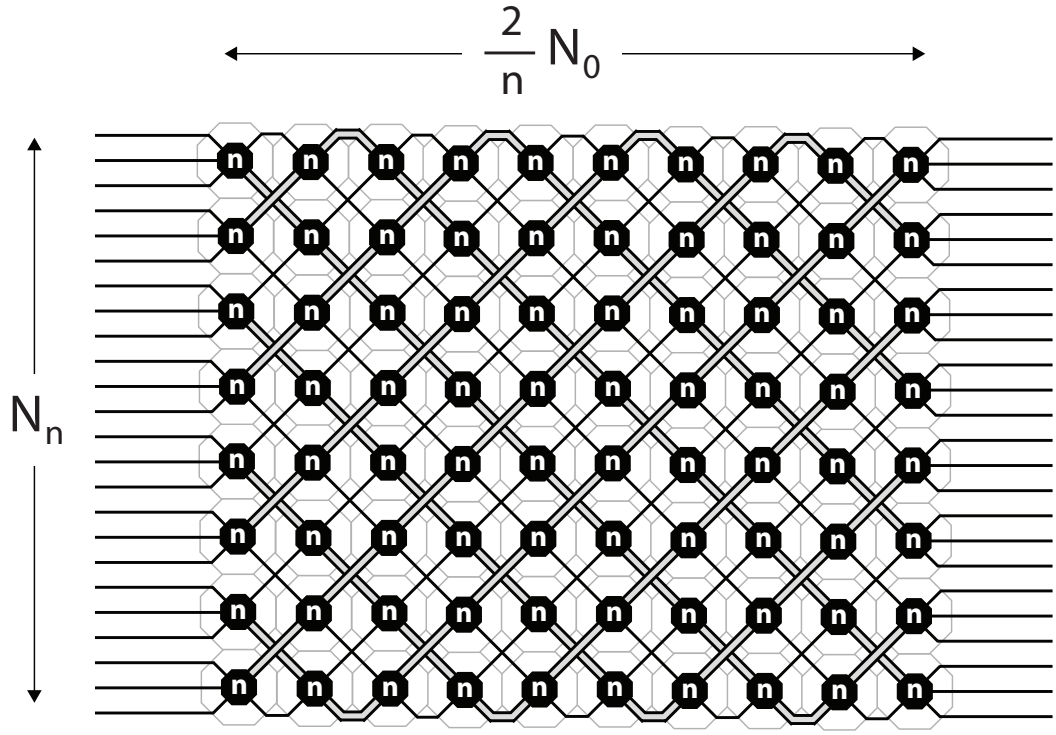

Figure 4.6: The generalized degree-3 double-twist directed graph. Each stage is n-dimensional (for n even), and the total cardinality is $(N_0 \times N_1 \times \cdots \times N_n)$. When $n = 2$ this graph reduces to the well-known 3D cartesian graph. The full connectivity is defined recursively in Figure 4.7

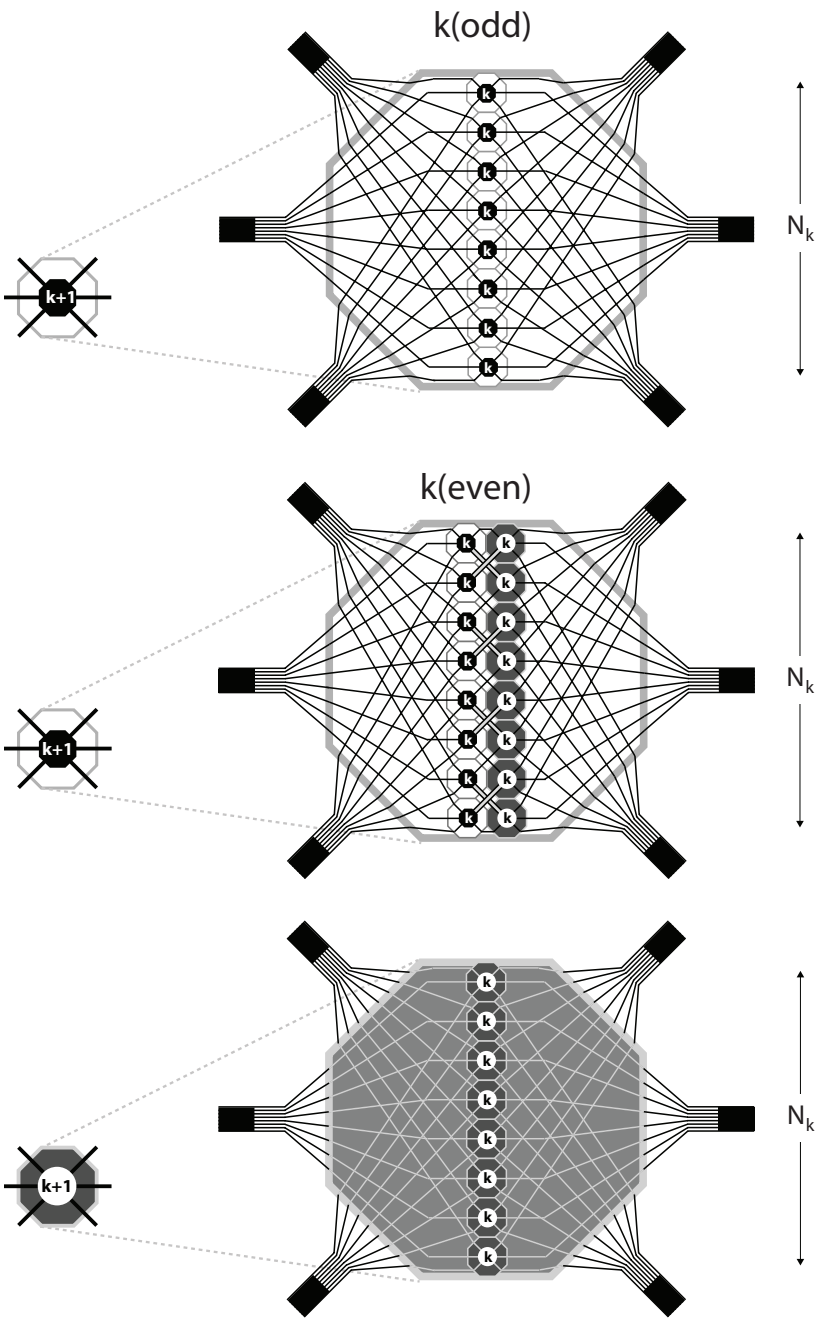

Figure 4.7: The generalized degree-3 double-twist directed graph is best graphically defined recursively through the combination of the two basic building blocks shown above. For one of the building blocks, the definition varies at the odd and even levels. At the lowest level, though, these building blocks still reduce to the basic node.

To achieve the minimal saturation length, the traditional degree-2 butterfly graph places many unnecessary restrictions on the parameters of the graph (e.g. the dimension of each stage, the cardinality of each dimension). While keeping the nodal degree, overall size, and number of stages of the graph constant, we can relax the constraints on the dimension and allow freedom to pick each dimension's cardinality individually, subject to

$$\prod_{k=1}^{n} N_k = M. \tag{4.1}$$

This allows us to define an entire family of degree-2 graphs that include on one extreme the butterfly graph and on the other the simple 2D cartesian graph. This family is illustrated in Figures 4.2 and 4.3. As one might expect, the choices of both dimension and cardinality have a direct effect on the local versus global nature of the resulting graph, and therefore allow for much design flexibility.

In a similar manner, the degree-3 butterfly graph can be generalized by also removing the dimensionality and cardinality constraints, subject to the overall size constraint of (4.1). This produces another family of directed graphs that span the range from entirely local to fully global connectivity. However, unlike the degree-2 case, this family does not reduce to 3D cartesian, as one might expect. This family of directed graphs is shown in Figures 4.4 and 4.5.

In both the generalized degree-2 and degree-3 classes, the connectivity between stages can be thought of as spanning only one dimension at a time, in turn cycling through each dimension. Thus, the connectivity pattern repeats itself every

n stages. An interesting third class of directed graphs can be found by allowing the channels between adjacent stages to span two dimensions at a time. This requires a degree-3 node and an even number of dimensions in the graph's stages. With this design, the interconnect pattern repeats itself every $n/2$ stages. As a result, we are able to obtain more local path redundancy with higher-dimensional stages that in turn lead to shorter saturation lengths. This family of graphs is illustrated in Figures 4.6, 4.7 and 4.8. As it turns out, for $n = 2$ this family of graphs reduces to the well known 3D cartesian topology.

4.3 Discussion & Summary

The properties of the three families of graphs are summarized in Table 4.1. For a given graph, we look at the spread of information, starting from a single node and flowing forward through the graph. To separate the issue of periodic connections, an infinite cardinality was assumed during the calculations. In general, adding the necessary constraint of finite cardinality will increase overall path redundancy at the cost of reducing the expected spread shown in the table. The finite cardinality is addressed further in Figure 4.9. The shaded cells highlight the areas of optimal spread in each graph; in these regions, the information is spreading to the maximum number of nodes possible. As a result, there is no path redundancy. The unshaded cells illustrate where each graph diverges from the

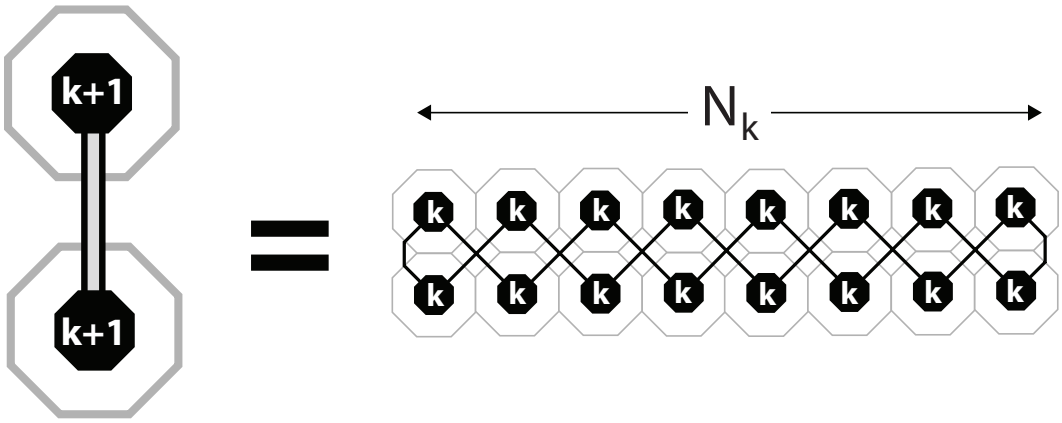

Figure 4.8: The weave in the $(k+1)^{th}$ dimension is combined with an additional weave in the k^{th} dimension as defined above. This shortens the repetitive interconnect pattern between stages and allows for more path diversity while maintaining a sufficient global spread.

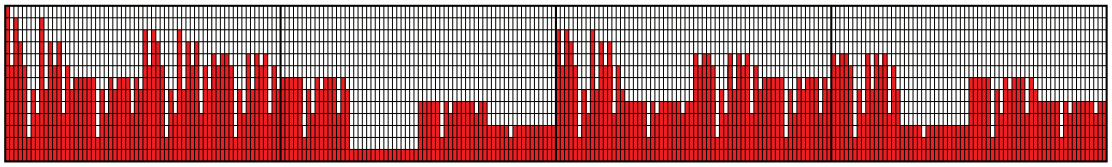

Figure 4.9: The signal propagation through a directed graph with the A_5^+ topology. The illustrated graph has a total cardinality of $(\,12 \times 4 \times 4 \times 4 \times 4\,)$, thus each of the 12 stages of the graph contain $M = 256$ nodes with periodic connections across each of the flow normal dimensions. Here, a signal is injected into the upper-left corner and travels down through the graph. The shaded blocks represent the gates that are reached by the signal at each subsequent stage. In this example, the original signal saturates all 256 gates at the 12^{th} stage. This illustrates how the periodic connections associated with finite cardinality serve to further reduce the predicted spread of Table 4.1 by increasing path redundancy.

Table 4.1: Optimal information spread for select directed graphs.

Directed Graph		n	Gates Reached at Each Stage ($N_1 = N_2 = \cdots = N_n = \infty$)										g_{10}	g_{20}
Generalized degree-2	$\mathbb{Z}_2$, V_2^{90}	1	2	3	4	5	6	7	8	9	10	11	66	231
	A_3^+, V_3^{90}	2	2	4	6	9	12	16	20	25	30	36	161	946
	V_4^{90}	3	2	4	8	12	18	27	36	48	64	80	300	2,800
	V_5^{90}	4	2	4	8	16	24	36	54	81	108	144	478	6,797
	V_6^{90}	5	2	4	8	16	32	48	72	108	162	243	696	14,325
	V_7^{90}	6	2	4	8	16	32	64	96	144	216	324	907	27,110
	V_8^{90}	7	2	4	8	16	32	64	128	192	288	432	1,167	46,836
	V_9^{90}	8	2	4	8	16	32	64	128	256	384	576	1,471	76,126
	V_{10}^{90}	9	2	4	8	16	32	64	128	256	512	768	1,791	119,772
	V_{11}^{90}	10	2	4	8	16	32	64	128	256	512	1,024	2,047	176,122
Generalized degree-3	W_2^{90}	1	3	5	7	9	11	13	15	17	19	21	121	441
	W_3^{90}	2	3	9	15	25	35	49	63	81	99	121	501	3,301
	W_4^{90}	3	3	9	27	45	75	125	175	245	343	441	1,489	17,185
	W_5^{90}	4	3	9	27	81	135	225	375	625	875	1,225	3,581	70,857
	W_6^{90}	5	3	9	27	81	243	405	675	1,125	1,875	3,125	7,569	245,545
	W_7^{90}	6	3	9	27	81	243	729	1,215	2,025	3,375	5,625	13,333	741,161
	W_8^{90}	7	3	9	27	81	243	729	2,187	3,645	6,075	10,125	23,125	1,978,545
	W_9^{90}	8	3	9	27	81	243	729	2,187	6,561	10,935	18,225	39,001	4,855,001
	W_{10}^{90}	9	3	9	27	81	243	729	2,187	6,561	19,683	32,805	62,329	$\sim 1.1 \times 10^7$
	W_{11}^{90}	10	3	9	27	81	243	729	2,187	6,561	19,683	59,049	88,573	$\sim 2.5 \times 10^7$
s_3	$\mathbb{Z}_3$	2	3	6	10	15	21	28	36	45	55	66	286	1,771
	A_5^+	4	3	9	18	36	60	100	150	225	315	441	1,358	22,165

standard butterfly graph, introducing more locality and robustness to the graph at the expense of overall global spread. The final two columns of the table show the total number of nodes reached in stages 0–10 and 0–20, respectively. These numbers give a measure of the short versus long range spread of each particular graph; the difference between these two spreads is highlighted when comparing the W_4^{90} graph to the A_5^+ topology. Across the board, through the first ten stages, the W_4^{90} graph reaches more nodes, giving it a larger short range spread. Thus, the A_5^+ graph contains more local structure and more path redundancy over this range. However, comparing g_{20} for each graph, one can see that past ten stages, the spread of A_5^+ quickly outpaces W_4^{90}. This means that the third family in some sense has the best of both worlds: a relatively high local path redundancy followed by a large long-range spread. While this may seem obviously superior, there still is no perfect graph. Rather, one must understand the needs of a particular application and select a topology appropriately. This research attempts simply to unify some of the existing design decision and to provide intermediate alternatives.

4.4 Acknowledgements

This chapter is, in part, a reprint of the material as it will appear in:

Cessna, J. and Bewley, T. (2010) Three general classes of regular, directed graphs. *Under preparation, IEEE Embedded Systems Letters.*

The dissertation author is the primary investigator and author of this publication.

Appendix A

Mixed Discrete/Continuous Adjoint Derivation

The full derivation of the gradient $\nabla J(\mathbf{u})$ is included here due to the unusual setting considered (that is, of a continuous-time system with discrete-time measurements). Perturbing the nonlinear model equations (1.1a) and linearizing about $\tilde{\mathbf{x}}(t)$ gives:

$$\frac{d\tilde{\mathbf{x}}'(t)}{dt} = A\big(\tilde{\mathbf{x}}(t)\big)\,\tilde{\mathbf{x}}'(t) \quad \text{with} \quad \tilde{\mathbf{x}}'_{_{-K}} = \mathbf{u}' \tag{A.1}$$

$$\Rightarrow \quad \mathcal{L}\,\tilde{\mathbf{x}}' = 0 \quad \text{where} \quad \mathcal{L} = \frac{d}{dt} - A\big(\tilde{\mathbf{x}}(t)\big). \tag{A.2}$$

Similarly, the perturbed cost function is:

$$J'(\mathbf{u}') = (\mathbf{u} - \bar{\mathbf{x}}_{_{0|0}})^T\, P_{_{0|0}}^{-1}\,\mathbf{u}' - \sum_{k=1}^{K} (\mathbf{y}_k - H\tilde{\mathbf{x}}_k)^T\, R^{-1}\, H\tilde{\mathbf{x}}'_k. \tag{A.3}$$

The perturbed cost function (A.3) is not quite in the form necessary to extract the gradient, as illustrated in (1.14). However, there is an implicitly-defined linear relationship between $\mathbf{u}'$ and $\tilde{\mathbf{x}}'(t)$ on $t \in (\,0, T\,]$ given by (A.1). To re-express this relationship, a set of K adjoint functions $\mathbf{r}^{(k)}(t)$ are defined over the measurement intervals such that, for all $k \in [\,1\,,\,K\,]$, the adjoint function $\mathbf{r}^{(k)}(t)$ is defined on the closed interval $t \in \left[\, t_{k-1}\,,\, t_{k}\,\right]$. These adjoint functions will be used to identify the gradient. Towards this end, a suitable duality pairing is defined here as:

$$\langle\ \mathbf{r}^{(k)}\,,\,\tilde{\mathbf{x}}'\ \rangle = \int_{t_{k-1}}^{t_k} (\mathbf{r}^{(k)})^T\,\tilde{\mathbf{x}}'\,dt. \tag{A.4}$$

Then, the necessary adjoint identity is given by

$$\langle\ \mathbf{r}^{(k)}\,,\,\mathcal{L}\,\tilde{\mathbf{x}}'\ \rangle = \langle\ \mathcal{L}^*\,\mathbf{r}^{(k)}\,,\,\tilde{\mathbf{x}}'\ \rangle + b^{(k)}. \tag{A.5a}$$

Using the definition of the operator $\mathcal{L}$ given by (A.2) and the appropriate integration by parts, it is easily shown that

$$\mathcal{L}^*\,\mathbf{r}^{(k)} = -\frac{d\mathbf{r}^{(k)}(t)}{dt} - A\big(\tilde{\mathbf{x}}(t)\big)^T\,\mathbf{r}^{(k)}(t), \tag{A.5b}$$

$$b^{(k)} = (\mathbf{r}_{k}^{(k)})^T\ \tilde{\mathbf{x}}'_{k} - (\mathbf{r}_{k-1}^{(k)})^T\ \tilde{\mathbf{x}}'_{k-1}. \tag{A.5c}$$

Returning to the perturbed cost function, (A.3) can be rewritten as:

$$J'(\mathbf{u}') = (\,\mathbf{u} - \bar{\mathbf{x}}_{0|0}\,)^T\,P_{0|0}^{-1}\ \mathbf{u}' - \sum_{k=1}^{K-1}(\,\mathbf{y}_k - H\,\tilde{\mathbf{x}}_k\,)^T\,R^{-1}\,H\,\tilde{\mathbf{x}}'_k - J'_{K}, \tag{A.6a}$$

$$J'_{K} = \big[H^T\,R^{-1}\,(\,\mathbf{y}_{K} - H\,\tilde{\mathbf{x}}_{K}\,)\big]^T\,\tilde{\mathbf{x}}'_{K}. \tag{A.6b}$$

Looking at the adjoint defined over the last interval, $\mathbf{r}^{(K)}(t)$, the following criteria is enforced:

$$\mathcal{L}^* \, \mathbf{r}^{(K)} = 0 \quad \Rightarrow \quad \langle \, \mathcal{L}^* \, \mathbf{r}^{(K)} \, , \, \tilde{\mathbf{x}}' \, \rangle = 0, \tag{A.7a}$$

$$\mathbf{r}_{K}^{(K)} = H^T R^{-1} \, (\, \mathbf{y}_{K} - H \, \tilde{\mathbf{x}}_{K} \,). \tag{A.7b}$$

Substituting (A.2) and (A.7a) into (A.5a) for $k = K$ gives:

$$\begin{aligned} & b^{(K)} = 0 \\ & \Rightarrow (\mathbf{r}_{K}^{(K)})^T \, \tilde{\mathbf{x}}'_{K} - (\mathbf{r}_{K-1}^{(K)})^T \, \tilde{\mathbf{x}}'_{K-1} = 0, \\ & \Rightarrow \left[H^T \, R^{-1} \, (\mathbf{y}_{K} - H \, \tilde{\mathbf{x}}_{K}) \right]^T \, \tilde{\mathbf{x}}'_{K} = (\mathbf{r}_{K-1}^{(K)})^T \, \tilde{\mathbf{x}}'_{K-1}, \end{aligned} \tag{A.8}$$

which allows us to re-express J'_{K} in (A.6b) as

$$J'_{K} = (\mathbf{r}_{K-1}^{(K)})^T \, \tilde{\mathbf{x}}'_{K-1}. \tag{A.9}$$

Note that (A.7a) and (A.7b) give the full evolution equation and terminal condition for the adjoint $\mathbf{r}^{(K)}$ defined on the interval $t \in [\, t_{K-1}, t_{K} \,]$. Hence, a backward march over this interval will lead to the term $\mathbf{r}_{K-1}^{(K)}$ contained in (A.9).

The perturbed cost function (A.6a) can now be rewritten such that

$$J'(\mathbf{u}') = (\, \mathbf{u} - \bar{\mathbf{x}}_{0|0})^T \, P_{0|0}^{-1} \, \mathbf{u}' - \sum_{k=1}^{K-2} (\, \mathbf{y}_k - H \, \tilde{\mathbf{x}}_k \,)^T \, R^{-1} \, H \, \tilde{\mathbf{x}}'_k - J'_{K-1}, \tag{A.10a}$$

$$\Rightarrow \quad J'_{K-1} = \left[\, H^T \, R^{-1} \, (\, \mathbf{y}_{K-1} - H \, \tilde{\mathbf{x}}_{K-1} \,) + \mathbf{r}_{K-1}^{(K)} \, \right]^T \, \tilde{\mathbf{x}}'_{K-1}. \tag{A.10b}$$

Enforcing the following conditions [cf. (A.7)] for the adjoint on this interval,

$\mathbf{r}^{(K-1)}(t)$,

$$\mathcal{L}^* \, \mathbf{r}^{(K-1)} = 0, \tag{A.11a}$$

$$\mathbf{r}^{(K-1)}_{K-1} = H^T \, R^{-1} \, (\mathbf{y}_{K-1} - H \, \tilde{\mathbf{x}}_{K-1}) + \mathbf{r}^{(K)}_{K-1}, \tag{A.11b}$$

it can be shown via a derivation similar to (A.8) that

$$J'_{K-1} = (\mathbf{r}^{(K-1)}_{K-2})^T \; \tilde{\mathbf{x}}'_{K-2}, \tag{A.12}$$

which is of identical form as (A.9). Thus, it follows that each of the adjoints can be defined in such a way as to collapse the sum in the perturbed cost function (A.3) as above, until the last adjoint equation $\mathbf{r}^{(1)}$ reduces the perturbed cost function to the following:

$$J'(\mathbf{u}') = (\, \mathbf{u} - \bar{\mathbf{x}}_{0|0})^T \, P^{-1}_{0|0} \; \mathbf{u}' - (\mathbf{r}^{(1)}_0)^T \; \tilde{\mathbf{x}}'_0 \tag{A.13}$$

with the adjoints over the K intervals being defined as:

$$\begin{aligned}
\frac{d\mathbf{r}^{(K)}(t)}{dt} &= -A\big(\tilde{\mathbf{x}}(t)\big)^T\,\mathbf{r}^{(K)}(t), \quad \text{where} \\
\mathbf{r}_{K}^{(K)} &= \;0\; + H^T R^{-1}\,(\,\mathbf{y}_{K} - H\,\tilde{\mathbf{x}}_{K}\,), \\
\\
\frac{d\mathbf{r}^{(K-1)}(t)}{dt} &= -A\big(\tilde{\mathbf{x}}(t)\big)^T\,\mathbf{r}^{(K-1)}(t), \quad \text{where} \\
\mathbf{r}_{K-1}^{(K-1)} &= \mathbf{r}_{K-1}^{(K)} + H^T R^{-1}\,(\,\mathbf{y}_{K-1} - H\,\tilde{\mathbf{x}}_{K-1}\,), \\
&\vdots \\
\frac{d\mathbf{r}^{(1)}(t)}{dt} &= -A\big(\tilde{\mathbf{x}}(t)\big)^T\,\mathbf{r}^{(1)}(t), \quad \text{where} \\
\mathbf{r}_{1}^{(1)} &= \mathbf{r}_{1}^{(2)} + H^T R^{-1}\,(\,\mathbf{y}_{1} - H\,\tilde{\mathbf{x}}_{1}\,).
\end{aligned} \tag{A.14}$$

Upon further examination, the system of adjoints (A.14) all have the same form. Each backward-marching adjoint variable $\mathbf{r}^{(k)}$ is endowed with a terminal condition that is the initial condition of the previous adjoint march $\mathbf{r}^{(k+1)}$ plus a correction due to the discrete measurement $\mathbf{y}_k$ at the measurement time t_k. Thus, the total adjoint march can be thought of as one continuous-time march of a single adjoint variable $\mathbf{r}(t)$ backward over the window $[\,t_0, t_K\,]$, with discrete "jumps" in $\mathbf{r}$ at each measurement time t_k. That is, (A.14) can be rewritten as:

$$\frac{d\mathbf{r}(t)}{dt} = -A\big(\tilde{\mathbf{x}}(t)\big)^T\,\mathbf{r}(t), \tag{A.15a}$$

which is marched backward over the entire interval $t \in [\,t_0, t_K\,]$ with $\mathbf{r}_K = 0$. At

the measurement times (t_k for $k \in M$) the adjoint is updated such that

$$\mathbf{r}_k \leftarrow \mathbf{r}_k + H^T R^{-1} (\mathbf{y}_k - H \tilde{\mathbf{x}}_k). \tag{A.15b}$$

Then, this definition of the adjoint can be substituted into (A.13) to give:

$$J'(\mathbf{u}') = (\mathbf{u} - \bar{\mathbf{x}}_{0|0})^T P_{0|0}^{-1} \mathbf{u}' - \mathbf{r}_0^T \tilde{\mathbf{x}}'_0, \tag{A.16}$$

$$\Rightarrow \quad J'(\mathbf{u}') = \left[P_{0|0}^{-1} (\mathbf{u} - \bar{\mathbf{x}}_{0|0}) - \mathbf{r}_0 \right]^T \mathbf{u}', \tag{A.17}$$

where (A.17) is found by noting that $\tilde{\mathbf{x}}'_{-K} = \mathbf{u}'$. Then finally, from (1.14) and (A.17), the gradient sought may be written as:

$$\nabla J(\mathbf{u}) = P_{0|0}^{-1} (\mathbf{u} - \bar{\mathbf{x}}_{0|0}) - \mathbf{r}_0. \tag{A.18}$$

The resulting gradient[1] can then be used iteratively to update the current estimate via a suitable minimization algorithm (steepest descent, conjugate gradient, limited-memory BFGS, etc.).

[1]Omitted in this gradient derivation is the substantial flexibility in the choice of the gradient definition (1.14) and the duality pairing (A.4). There is freedom in the choice of these inner products (e.g. by incorporating derivative and/or integral operators as well as weighting factors) that can serve to better precondition the optimization problem at hand without affecting its minimum points. This ability to precondition the adjoint problem is discussed at length in [49].

Bibliography

[1] The Dolphin SCI interconnect. White paper, Dolphin Interconnect Solutions, 1996.

[2] N. R. Adiga, M. A. Blumrich, D. Chen, P. Coteus, A. Gara, M. E. Giampapa, P. Heidelberger, S. Singh, B. D. Steinmacher-Burow, T. Takken, M. Tsao, and P. Vranas. Blue Gene/L torus interconnection network. *IBM Journal of Research and Development*, 49(2/3):265, 2005.

[3] B. Anderson and J. Moore. *Optimal Filtering*. Dover Publications, 1979.

[4] J. Anderson. An ensemble adjustment Kalman filter for data assimilation. *Monthly Weather Review*, 129:2884–2903, 2001.

[5] J. R. Baumgardner and P. O. Frederickson. Icosahderal discretization of the two-sphere. *SIAM J. Numer. Anal.*, 22(6):1107–1115, 1985.

[6] L. Berre, O. Pannekoucke, G. Desroziers, and G. Stefănescu. A variational assimilation ensemble and the spatial filtering of its error covariances. In *Proceedings of the ECMWF workshop on Flow-dependent aspects of Data Assimilation*, pages 151–168. Reading, 2007.

[7] T. Bewley. *Numerical Renaissance: Simulation, Optimization, and Control*. under preparation, 2009.

[8] T. Bewley and A. Sharma. A tractable framework for grid-based bayesian estimation of nonlinear low-dimensional systems with sparse nongaussian pdfs. *Automatica*, page (submitted).

[9] C. Bishop, B. Etherton, and S. Majumdar. Adaptive sampling with the ensemble transform Kalman filter. Part I: Theoretical aspects. *Monthly Weather Review*, 129:420–436, 2001.

[10] F. Bouttier and P. Courtier. Data assimilation concepts and methods march 1999. *Meteorological training course lecture series. ECMWF*, 2002.

[11] A. Caya, J. Sun, and C. Snyder. A comparison between the 4DVar and the ensemble Kalman filter techniques for radar data assimilation. *Monthly Weather Review*, 133(11):3081–3094, 2005.

[12] J. Cessna, C. Colburn, and T. Bewley. Multiscale retrograde estimation and forecasting of chaotic nonlinear systems. In *Proceedings of the 46th IEEE Conference on Decision and Control*, New Orleans, LA, USA, 2007.

[13] H. Chen, C.-K. Cheng, A. B. Kahng, I. I. Mandoiu, Q. Wang, and B. Yao. The Y architecture for on-chip interconnect: Analysis and methodology. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 24(4):588–599, 2005.

[14] H. Chen, B. Yao, F. Zhou, and C.-K. Cheng. The Y-architecture: yet another on-chip interconnect solution. In *Proceedings of the 2003 Conference on Asia South Pacific Design Automation*, pages 840–847, 2003.

[15] S. Cohn, N. Sivakumaran, and R. Todling. A fixed-lag Kalman smoother for retrospective data assimilation. *Monthly Weather Review*, 122:2838–2867, April 1994.

[16] W. J. Dally and B. Towles. *Principles and Practices of Interconnection Networks*. Elsevier, Burlington, 2004.

[17] F.-X. L. Dimet and O. Talagrand. Variational algorithms for analysis and assimilation of meteorological observations: theoretical aspects. *Tellus*, 38A:97, 1986.

[18] A. Doucet, N. Gordon, and V. Krishnamurthy. Particle filters for state estimation of jump markov linear systems. *IEEE Trans. Signal Processing*, 49:613–624, 1993.

[19] G. Evensen. Sequential data assimilation with a non-linear quasi-geostrophic model using Monte Carlo methods to forecast error statistics. *Journal of Geophysical Research*, 99:10143–10162, May 1994.

[20] G. Evensen. The ensemble Kalman filter: theoretical formulation and practical implementation. *Ocean Dynamics*, 53:343–367, 2003.

[21] G. Evensen and P. van Leeuwen. An ensemble Kalman smoother for nonlinear dynamics. *Monthly Weather Review*, 128:1852–1867, June 2000.

[22] M. Fisher. Assimilation techniques (4): 4dvar, April 2001. In *Meteorological training course lecture series*. ECMWF, 2002.

[23] M. Fisher and P. Courtier. Estimating the covariance matrices of analysis and forecast error in variational data assimilation. Tech Memo 220, European Centre for Medium-Range Weather Forecasts, 1995.

[24] M. Ghil, S. Cohn, J. Tavantzis, K. Bube, and E. Isaacson. Applications of estimation theory to numerical weather prediction. *Dynamic meteorology, data assimilation methods*, pages 139–224, 1981.

[25] N. Gustaffson. Discussions on '4DVar of EnKF?'. *Tellus*, 59A:774–777, 2007.

[26] T. Hamill and C. Snyder. A hybrid ensemble Kalman filter–3D variational analysis scheme. *Monthly Weather Review*, 128(8):2905–2919, 2000.

[27] T. Hamill, J. Whitaker, and C. Snyder. Distance-dependent filtering of background error covariance estimates in an ensemble Kalman filter. *Monthly Weather Review*, 129:2776–2790, 2001.

[28] I. Hoteit, D. Pham, G. Triantafyllou, and G. Korres. Particle Kalman filtering for data assimilation in meteorology and oceanography. *Under Preparation*, 2008.

[29] P. Houtekamer and H. Mitchell. Data assimilation using an ensemble Kalman filter technique. *Monthly Weather Review*, 126:796–811, March 1998.

[30] P. Houtekamer and H. Mitchell. A sequential ensemble Kalman filter for atmospheric data assimilation. *Monthly Weather Review*, 129:123–137, November 2001.

[31] B. Hunt, E. Kalnay, E. Kostelich, E. Ott, D. Patil, T. Sauer, I. Szunyogh, J. Yorke, and A. Zimin. Four-dimensional ensemble Kalman filtering. *Tellus*, 56A:273–277, April 2004.

[32] A. Jazwinski. *Stochastic Processes and Filtering Theory*. Academic Press, 1970.

[33] S. Julier and J. Uhlmann. A new extension of the kalman filter to nonlinear systems. *Proc. AeroSense: 11th Int. Symp. Aerospace/Defence Sensing, Simulation and Controls*, pages 182–193, 1997.

[34] S. Julier and J. Uhlmann. Unscented filtering and nonlinear estimation. *Proc. of the IEEE*, 92:401–422, 2004.

[35] T. Kailath. Some new algorithms for recursive estimation in constant linear systems. *IEEE Transactions on Information Theory*, 19:750–760, 1973.

[36] R. Kalman. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82(1):35–45, 1960.

[37] R. Kalman and R. Bucy. New results in linear filtering and prediction theory. *Journal of Basic Engineering*, 83(3):95–108, 1961.

[38] E. Kalnay, H. Li, T. Miyoshi, S. Yang, and J. Ballabrera-Poy. 4d-var or ensemble kalman filter? *Tellus*, 59A:758–773, 2007.

[39] S. Kim, G. Eyink, J. Restrepo, F. Alexander, and G. Johnson. Ensemble filtering for nonlinear dynamics. *Monthly Weather Review*, 131:2586–2594, November 2003.

[40] T. Kraus, P. Kühl, L. Wirsching, H. Bock, and M. Diehl. A moving horizon state estimation algorithm applied to the Tennessee Eastman Benchmark Process. In *Proceedings of the IEEE Conference on Multisensor Fusion and Integration for Intelligent Systems*, pages 377–382, 2006.

[41] Z. Li and I. Navon. Optimality of 4D-Var and its relationship with the Kalman filter and Kalman smoother. *Quarterly Journal of the Royal Meteorological Society*, 127(572):661–684, January 2001.

[42] A. Lorenc. Analysis methods for numerical weather prediction. *Quarterly Journal of the Royal Meteorological Society*, 112(474):1177–1194, 1986.

[43] A. Lorenc. The potential of the ensemble Kalman filter for NWP-a comparison with 4D-Var. *Quarterly Journal of the Royal Meteorological Society*, 129(595):3183–3203, 2003.

[44] E. Lorenz. Deterministic nonperiodic flow. *Journal of the Atmospheric Sciences*, 20:130–141, January 1963.

[45] D. Majewski, D. Liermann, P. Prohl, B. Ritter, M. Buchold, T. Hanish, G. Paul, and W. Wergen. The operational global icosahedral-hexagonal gridpoint model gme: Description and high-resolution tests. *Monthly Weather Review*, 130:319–338, 2002.

[46] H. Michalska and D. Mayne. Moving horizon observers and observer-based control. *IEEE Transactions on Automatic Control*, 40:995–1006, 1995.

[47] E. Ott, B. Hunt, I. Szunyogh, A. Zimin, E. Kostelich, M. Corazza, E. Kalnay, D. Patil, and J. Yorke. A local ensemble Kalman filter for atmospheric data assimilation. *Tellus*, 56A:415–428, October 2004.

[48] D. Parrish and J. Derber. The National Meteorological Center's spectral statistical-interpolation analysis system. *Monthly Weather Review*, 120:1747–1763, 1992.

[49] B. Protas and T. Bewley. A computational framework for the regularization of adjoint analysis in multiscale pde systems. *Journal of Computational Physics*, 195:49–89, 2004.

[50] F. Rabier, J.-N. Thepaut, and P. Courtier. Extended assimilation and forecast experiments with a four-dimensional variational assimilation system. *Quarterly Journal of the Royal Meteorological Society*, 124(550):1861–1887, July 1998.

[51] H. Rauch, F. Tung, and C. Striebel. Maximum likelihood estimates of linear dynamic systems. *AIAA Journal*, 3:1445–1450, 1965.

[52] N. G. S. Arulampalam, S. Maskell and T. Clapp. A tutorial on particle filters for on-line non-linear/non-gaussian bayesian tracking. *IEEE Transactions on Signal Processing*, 50:174–188, 2002.

[53] R. Sadourny, A. Arakawa, and Y. Mintz. Integration of the nondivergent barotropic vorticity equation with an icosahedral-hexagonal grid for the sphere. *Monthly Weather Review*, 96:351–356, 1968.

[54] L. C. Stewart and D. Gingold. A new generation of cluster interconnet. White paper, SiCortex, Inc., 2006.

[55] R. Stratonovich. Data assimilation and inverse methods in terms of a probabilistic formulation. *Radiofizika*, 2:892—901, 1959.

[56] G. R. Stuhne and W. R. Peltier. New icosahedral grid-point discretizations of the shallow water equations on the sphere. *Journal of Computational Physics*, 148:23–58, 1999.

[57] P. Swerling. A proposed stagewise differential correction procedure for satellite tracking and prediction. *RAND Technical Report, Santa Monica, California*, 1958.

[58] S. L. Teig. The X architecture: Not your father's diagonal wiring. In *Proceedings of the 2002 International Workshop on System-Level Interconnect Predction*, 2002.

[59] P. van Leeuwen and G. Evensen. Data assimilation and inverse methods in terms of a probabilistic formulation. *Monthly Weather Review*, 124:2898–2913, 1996.

[60] J. Whitaker and T. Hamill. Ensemble data assimilation without perturbed observations. *Monthly Weather Review*, 130(7):1913–1924, 2002.

[61] M. Zhang, F. Zhang, and J. Hansen. Coupling ensemble Kalman filter with four dimensional variational data assimilation. In *22nd Conference on Weather Analysis and Forecasting/ 18th Conference on Numerical Weather Prediction*, 2007.