Parallel Low Discrepancy Parameter Sweep for Public Health Policy

Sudheer Chunduri*, Meysam Ghaffari†, Mehran Sadeghi Lahijani†, Ashok Srinivasan† and Sirish Namilae‡
*Argonne National Laboratory
†Florida State University
‡Embry-Riddle Aeronautical University

***Abstract*—Numerical simulations are used to analyze the effectiveness of alternate public policy choices in limiting the spread of infections. In practice, it is usually not feasible to predict their precise impacts due to inherent uncertainties, especially at the early stages of an epidemic. One option is to parameterize the sources of uncertainty and carry out a parameter sweep to identify their robustness under a variety of possible scenarios. The Self Propelled Entity Dynamics (SPED) model has used this approach successfully to analyze the robustness of different airline boarding and deplaning procedures. However, the time taken by this approach is too large to answer questions raised during the course of a decision meeting. In this paper, we use a modified approach that pre-computes simulations of passenger movement, performing only the disease-specific analysis in real time. A novel contribution of this paper lies in using a low discrepancy sequence (LDS) in the parameter sweep, and demonstrating that it can lead to a reduction in analysis time by one to three orders of magnitude over the conventional lattice-based parameter sweep. However, its parallelization suffers from greater load imbalance than the conventional approach. We examine this and relate it to number-theoretic properties of the LDS. We then propose solutions to this problem. Our approach and analysis are applicable to other parameter sweep problems too. The primary contributions of this paper lie in the new approach of low discrepancy parameter sweep and in exploring solutions to challenges in its parallelization, evaluated in the context of an important public health application.[1]**

***Index Terms*—parameter sweep, low discrepancy sequence, parallel computing, public health, air travel**

I. Introduction

Air travel has been identified as a leading factor in the spread of several infectious diseases [1], [2] including influenza, severe acute respiratory syndrome (SARS), tuberculosis, and measles. This has motivated calls for limitations on air travel, for example during the 2014 Ebola outbreak in West Africa. However, such limitations carry considerable economic and human costs. Moreover, the benefits of such travel restrictions are questionable [3].

The goal of Project VIPRA[2] is to produce a science-based analysis of public policy options that can lead to mitigating the spread of diseases without disrupting air travel [3]–[8]. A major challenge in predicting the consequence of policy or procedural change arises from the paucity of data in the early stages of a potential epidemic and due to uncertainties in human response to policy changes. Project VIPRA uses a new approach in which sources of uncertainty are parameterized, and the parameter space is explored to identify vulnerabilities in a policy under a variety of possible scenarios. Rather than come up with a precise prediction, the goal is to identify the different possible outcomes, from which decision makers can evaluate the robustness of a policy to the different scenarios that could potentially arise.

In this manuscript, we focus on a critical model in VIPRA – the Self Propelled Entity Dynamics (SPED) model – which is used to simulate the movement of passengers in an airplane. This model can be used to analyze the impact of boarding and deplaning procedures on the spread of proximity-based diseases, such as Ebola and SARS. This model is described further in Section II.

A major limitation of the SPED implementation is that the simulation time taken is around 20 minutes using thousands of nodes of a parallel system even for moderate problem sizes. Consequently, all analysis is now pre-computed, and this model cannot answer new questions that are raised in the course of a policy meeting, where a response time in the span of a few seconds is desirable.

Our solution is to pre-compute the SPED simulation results, which provides a large set of potential passenger trajectories while boarding or deplaning. During the course of a decision meeting, only the disease-specific analysis would then need to be performed in real time. We describe our proposed use of SPED in a decision meeting scenario further in Section III.

This scenario assumes that decision meetings are called at the initial stages of a new disease outbreak. Some analysis parameters depend on the particular strain of the disease considered, such as the distance threshold within which a human may transmit the infection to another human. Experts may wish to determine the impact of different possible disease-specific parameters. The same set of passenger trajectories will be used in different decision meetings. Consequently, this database of results should cover all conceivable scenarios. That is, we wish to cover the parameter space dealing with passenger trajectories very finely.

The computational effort increases exponentially with dimensionality, that is, as the model is refined to account for more factors. In our current work, we report results with five

[1]This material is based upon work supported by the National Science Foundation grant #1640822 on Petascale Simulation of Viral Infection Propagation Through Air Travel.

[2]www.cs.fsu.edu/vipra. This project has received recent publicity in over 75 news outlets around the world related to results on Ebola transmission in planes.

parameters, in contrast to three parameters in our previous work [5]. It is computationally expensive to perform this parameter sweep with the same fine granularity as we could with three parameters.

Diseases may vary in their sensitivities to the passenger trajectory parameters, and so it is possible that for a specific disease, it is sufficient to analyze a coarser subset of the fine parameter sweep. Ideally, we should be able to start from a coarse parameter sweep, and make it increasingly fine until the results show that we have converged to a sufficiently accurate solution.

The original SPED code uses a lattice-based sweep. That is, parameters are chosen from a uniform d-dimensional grid. This has two limitations. First, for a specific number of points, it does not cover the space efficiently, and second, it does not permit efficient check of convergence, as explained in Section IV. Our goal is to propose a novel alternative to this parameter sweep. Since such lattice-based parameter sweeps are used in other applications too [9]–[12], the results of our work will have a broad impact beyond the specific application considered here.

In Section IV, we also describe our solution, which is to use a low-discrepancy sequence (LDS). A LDS can cover the space efficiently and enable an efficient check for convergence. Traditionally, LDSs have been used for integration, and much theoretical analysis is focused on that. We propose it for the parameter sweep, theoretically justify its promise in Section IV, and empirically demonstrate its effectiveness in Section V. We show that it can lead to a one to three orders of magnitude improvement in performance over the conventional parameter sweep.

The analysis is parallelized by having each core analyze a few passenger trajectory files, synchronize with other cores to verify if the computation has converged, and then repeat the process if the computation has not converged. Load imbalance can impact the analysis time. Load in the analysis of lattice-based parameter sweep can be balanced effectively through a simple cyclic distribution of parameters to core. We show that this is usually not effective for the LDS-based parameter sweep due to number-theoretic features of the LDS. We show alternate approaches that are effective under different scenarios. We discuss load balancing further in Section VI.

While the focus of this manuscript is on computation, rather than on the science, we briefly point out a new discovery enabled by our efficient and large parameter sweep in Section VII. We show that the probability distribution for the number of human interactions (which indicates the potential for infection spread) is bimodal, and arises from the sum of two Gaussian distributions. This is a novel observation, and it gives a direction for epidemiologists to further investigate the mechanisms responsible for it.

We discuss related work on epidemic modeling and LDS in Section VIII. We then summarize our conclusions and present directions for future work in Section IX.

II. SELF-PROPELLED ENTITY DYNAMICS MODEL

A. Modeling Passenger Movement in Planes

The Self-Propelled Entity Dynamics (SPED) model is used to simulate movement of passengers in airplanes [4], [5]. Its basic computational structure is similar to Molecular Dynamics, with each passenger and fixed surface treated as a point particle. Each human, and fixed surfaces such as seats and walls, exerts a repulsive force on other passengers to prevent them from coming too close. This repulsion is implemented using a certain "potential" function, whose parameters can be varied. Unlike with molecular dynamics, there is no attractive force; instead passengers come closer toward each other due to a self-propulsion term. For example, movement toward the exit is the propulsion term when simulating deplaning. These forces are used in Newton's laws of motion and an ordinary differential equation solved to generate trajectories for passengers.

In practice, the above idealized propulsion and repulsion terms are not adequate to model passenger behavior. The above social-dynamics formulation is, therefore, complemented with models for behavioral characteristics, such as passengers stopping to collect luggage from overhead bins, in order to make the simulation realistic. In addition, SPED includes some randomness in the movement of passengers to account for realistic variation in human behavior.

Given the inherent variation and uncertainty in human behavior, a single set of simulation parameters cannot capture the varieties of movement patterns in humans. Consequently, the model parameters are varied to generate a large number of scenarios for passenger trajectories. These trajectories are analyzed to identify passengers who come close to each other. If they come closer than a given threshold for a disease, then it is considered a contact, which is associated with a non-zero probability of infection transmission, if one of those passengers was infected.

The VIPRA approach does not assume that the identity of an infected passenger is known, because then alternate public health strategies, such as quarantining, can be used. Instead, an analysis is performed taking, in turn, each of the passengers as a potential source of the disease. The goal is to identify boarding or deplaning procedures that will be effective in reducing the spread of infection when one does not know which passenger is infected.

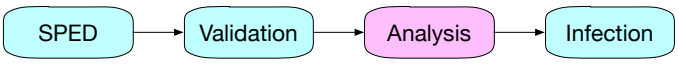


Fig. 1: SPED workflow

The workflow in the computation is shown in Figure 1. Note that the workflow includes a validation step before the analysis, for the following reason. While SPED desires to generate a wide variety of possible scenarios, including extreme cases that are rare, certain combinations of parameters lead to results that are clearly unrealistic. Such scenarios are removed during the validation phase. It is currently integrated with the SPED

simulation for reasons of computational efficiency mentioned in [5].

B. Prior Result with SPED

Prior work [5] involved sweeping a three dimensional parameter space (that is, three parameters were varied) with 41 choices for each parameter, leading to a parameter space of $41^3 = 68921$ parameter combinations. The sequential computational time would be several thousand hours. It was parallelized on the Blue Waters system at NCSA. The schematic for the parallel computation is shown in algorithm 1. Each parameter combination runs on one core, though a single core may run more than one parameter combination. The output for each parameter combination is a file containing the trajectories of all passengers.

The time taken for one parameter combination, to a large extent, depends on the number of iterations required for passengers to deplane, because the simulation stops when all have deplaned. This varies considerably for different parameter combinations, and therefore, dynamic load balancing was performed. This yielded results in around 20 minutes of wall-clock time using 39681 cores (1241 nodes).

Algorithm 1: Parallel scheme for the SPED simulation

```
procedure
    Initialization
for each process do
    Read input (coordinates, input parameters)
    Loop over n time steps
  for each pedestrian do
        Compute desired velocity
        Compute forces
        Find new positions & velocities
        Compute averages, neighbors
    End loop
    Data analysis
end procedure
```

The above parallelization was used to analyze the impact of different policies under a variety of scenarios. For example, we used the contact information to estimate the number of new infections generated in flights with different boarding and deplaning procedures [3].

III. SPED IN A DECISION MEETING SCENARIO

Next, we will discuss some limitations of SPED in a real-time decision making context and our proposed approach to deal with it. Decision makers from different domains typically meet together in a decision meeting. The experts may propose an evaluation of infection profiles based on infectivity, probability of infection transmission, and contact threshold. Computation of the contacts is the major computational bottleneck here. The infection profile can be obtained quickly from formulas once this is computed. In the course of a decision meeting, a real-time response to queries is required; otherwise, it has been observed that people get distracted and digress to other topics.

The current SPED implementation has a couple of computational bottlenecks that make it infeasible for a real-time response. Even a parallel parameter sweep is limited by the speed of the slowest simulation, which is the order of 20 minutes. Thus, SPED currently pre-computes answers to queries too, rather than responding to new queries.

An additional limitation makes even such pre-computations difficult. The current SPED implementation involves five parameters, rather than three. Maintaining the same granularity as with prior results (41 choices per parameter) would require the order of tens of millions of hours of sequential computation time. Even with massive parallelism, computer time allocation limitations make this infeasible. Future versions of SPED are expected to have more parameters, and given the exponential relationship between dimension and number of parameter combinations, this problem can be expected to get worse in the future.

Our solution to the first problem, of real-time response, is to separate the computation of passenger trajectories from the analysis of contacts based on these trajectories. The simulation of the trajectories accounts for most of the computational time. These can be pre-computed, exploring the parameter space exhaustively, so that results are available for analysis of any disease that we may wish to perform in the future. Massive parallelism can be used in the pre-computation phase.

In contrast, the analysis phase will occur in a decision making context, where parallel computing resources will be limited. Supercomputing centers can sometimes provide dedicated reservations for such meetings, however, it would be wasteful to block a large number of nodes for the duration of the meeting when the machines will be used only for short bursts of time.

The current analysis implementation examines all pairs of passengers at each recorded time step from the pre-computed trajectories to check for possible interactions. This is inefficient, and so we modified it using a standard technique. The airplane is covered by a grid, with each cell in the grid having dimensions of the contact threshold. For any passenger in a cell, we examined only the local cell and neighboring ones for possible contact. This resulted in about a 20% decrease in analysis time. While the analysis of a single file is fast, the number of trajectory files to be examined – each file corresponding to one parameter combination – is large, which poses a computational bottleneck; analyzing all the files sequentially in our example application would take tens of hours, and even parallelization on 1000 cores would require a few minutes, which is too long for a real-time response.

We seek to reduce the number of trajectory files that need to be considered. We observe that diseases may vary in their sensitivities to the passenger trajectory parameters, and so it is possible that for a specific disease, it is sufficient to analyze a coarser subset of the fine parameter sweep. Ideally, we should be able to start from a coarse parameter sweep, and make it increasingly fine until results show that we have converged to

a sufficiently accurate solution. That is, although the parameter sweep in generating the trajectories is fine-scaled in order to account for all possible contexts in which it may be used, we may not need the finest granularity in typical situations. The next section examines this issue in greater detail.

IV. LOW DISCREPANCY PARAMATER SWEEP

A. Covering the Parameter Space

We wish to cover the entire parameter space as efficiently as possible. Basically, if we select N points, then the gaps between the points are the uncovered areas. Of course, most of the area will be uncovered. But, intuitively, we want the gaps between the points to be uniformly small.

The conventional parameter sweep is lattice based. It does not cover the space efficiently. For example, we can notice from Figure 2a that the gap in the horizontal direction is lower than the gap in the diagonal direction. This problem worsens in higher dimensions.

Another problem with the lattice-based parameter sweep is that it is inefficient in checking for convergence, as explained below. Let us consider a grid with $N = R^d$ points in d-dimensions. To check for convergence, we check to see if the output of a lattice stops changing much as we make the lattice finer. To check if it has converged with N points, we need to compare it against the results from the next smaller sub-lattice, which would be approximately of size $(R/2)^d = N/2^d$ if 2 is a prime factor of R-1, and even smaller otherwise. This difference in the sizes of consecutive lattices – factor 2^d – is large, and we will not be able to detect a possible convergence after size $N/2^d$ until we reach size N. For example, in 5 dimensions, this is a factor 32 difference. We will, therefore, analyze a much larger number of trajectory files than we could with alternative schemes.

Choosing parameters at random can avoid the convergence problem mentioned above. But, a random sequence does not cover the parameter space efficiently. For example, Figure 2b shows clustering of points and sparsely populated regions.

Formally, the deviation from non-uniformity is characterized by the discrepancy. The most popularly analyzed discrepancy measure is the star discrepancy. The discrepancy reduces with the number of points for any reasonable sequence. The rate at which it reduces varies, though. For a random sequence, the star discrepancy is proportional to $(loglogN)^{1/2}/N^{1/2}$.

Given N points in d dimensions, low discrepancy point sets [13]–[15] cover the space efficiently, with the star discrepancy proportional to $log^{d-1}N/N$, which is asymptotically much better than that of a random sequence. However, the point set requires the entire set of points to achieve this discrepancy. Consequently, we cannot use a subset of a point set and check for convergence. This makes it unsuitable for our application, because we hope to terminate our analysis after only a small fraction of points have been analyzed.

LDS can cover the space efficiently and permit a check for convergence, enabling us to stop quickly once convergence is reached. Their star discrepancy is proportional to log^dN/N [13]–[15]. We can see from Figure 2c that points avoid clustering close to each other, thus, covering the space fairly uniformly. For any value of N, they cover the space efficiently.

B. Choice of Low Discrepancy Sequence

There are several low discrepancy sequences available. We use a Scrambled Halton Sequence. It corrects certain practical defects in the Halton LDS. From the perspective of our aims, it has the following positive feature. In general, points from a d+1-dimensional LDS may be unrelated to the corresponding points from a d-dimensional LDS. However, in the Halton and Scrambled Halton sequences, the first d coordinates of a d+1-dimensional sequence are identical to the corresponding coordinates of a d-dimensional sequence, with only the d+1th coordinate being appended to each point. In the future, if certain fixed parameters in SPED are made variable, then using the scrambled Halton sequence will enable us to reuse the previously computed trajectory files, thus reducing the number of new trajectory files that need to be computed.

Note that each coordinate of a LDS is in (0, 1). The actual parameters for the computation have a different range. We use a standard shift and scaling to map LDS points to our parameter space. For example, if a parameter range is [a, b] and the corresponding coordinate of an LDS point is q, then this point will be mapped to the parameter value a + (b-a)*q.

C. Convergence

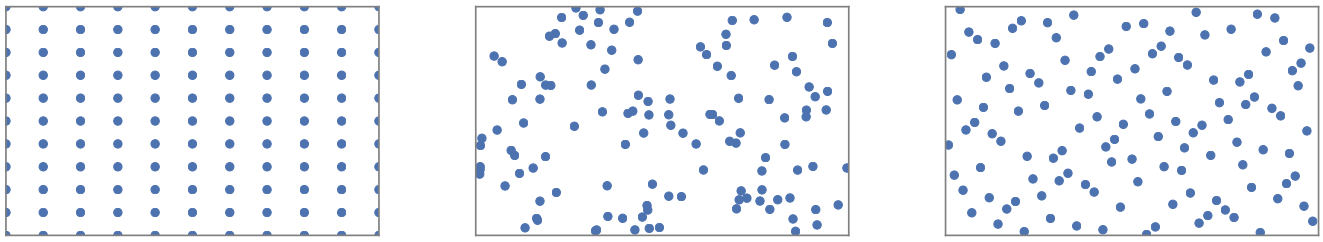
LDS has traditionally been used for integration, and much of the theoretical analysis is focused on that. The Koksma-Hlawka inequality provides an upper bound for the integration error using the star discrepancy [13], [14], [16]. The bound is usually much larger than the actual error. In practice, the error is often close to N^{-1} asymptotically, though it may have an initial phase with a $N^{-1/2}$ convergence rate [13], [14], [16]. While the latter is asymptotically weaker than the Koksma-Hlawka inequality, it is often tighter for realistic values of N.

We propose using LDS for a parameter sweep, rather than for integration. In our convergence analysis, we compare the results for different values of the number of trajectory files analyzed (N) to verify if the solution has converged. We assume an error roughly proportional to $1/N^p$, for unknown p, in view of the results cited above.

This type of problem is a standard one in Grid Convergence (or Refinement) Analysis in Computational Fluid Dynamics [17], where one uses results with different grid granularities to verify convergence. The error there is posed as proportional to h^p where h is the grid spacing. Here, h is analogous to $1/N$ in our application. Numerical solutions are computed for grids with granularity h_0, $h_0/2$, $h_0/4$, etc.

In an analogous manner, we check for convergence by computing a quantity of interest using N_0 trajectory files, then $2N_0, 4N_0, ...$ trajectory files. The ith check is performed with 2^iN_0 trajectory files. We provide the convergence criteria below. Further justification is provided in Appendix A.

Let us denote by x_i the estimate with 2^iN_0 trajectory files of a quantity of interest having an unknown exact value x. Let $e_i = |x_i - x_{i-1}|$. e_i goes to 0 asymptotically. We check if e_i



(a) (b) (c)

Fig. 2: (a) Two-dimensional Lattice. (b) Two-dimensional Random Sequence. (c) Two-dimensional Low Discrepancy Sequence.

is less than 1% of x_i. In order to be a little more stringent, we require e_i to satisfy this constraint for two consecutive values of i.

Next, we need to determine the quantity that needs to be evaluated. This would depend on the scientific results that we wish to compute. In this paper, we focus on computing the histogram of the number of humans who interact with each other in a trajectory file. Each trajectory file will give us one number – the number of human pairs who were within the contact threshold for a disease in one simulation of deplaning. If the histogram of this number over the entire parameter space shows large values as being frequent, then we expect that the disease could be transmitted to many persons. On the other hand, if low values have high probability, then this would indicate that the disease is not likely to spread widely.

We wish to check for the convergence of the histogram by checking if the first four moments of the probability distribution have converged. In practice, we verify this by checking for convergence of estimates of the mean, standard deviation, skewness, and kurtosis. If each of the four has converged, then we consider the histogram as having converged. Note that unlike in solutions of differential equations, we don't require several digits of accuracy in the result. Models for human movement are somewhat inaccurate, given inherent uncertainties in human behavior, and so we just require important features of the probability distribution to be brought out.

Algorithm 2: Histogram Convergence Scheme

```
procedure
    Initialization
for each process do
    Analyze N_0 files
    Update local moments
    Compute global statistics

    i=0
 while not converged and files remain unanalyzed do
        i++
        Analyze (2^i − 2^{i−1})N_0 files
        Update local moments
        Compute global statistics
        Check for convergence
end procedure
```

V. EMPIRICAL ANALYSIS OF PARAMATER SWEEP

In this section, we show that the LDS parameter sweep is much more effective than the conventional lattice sweep.

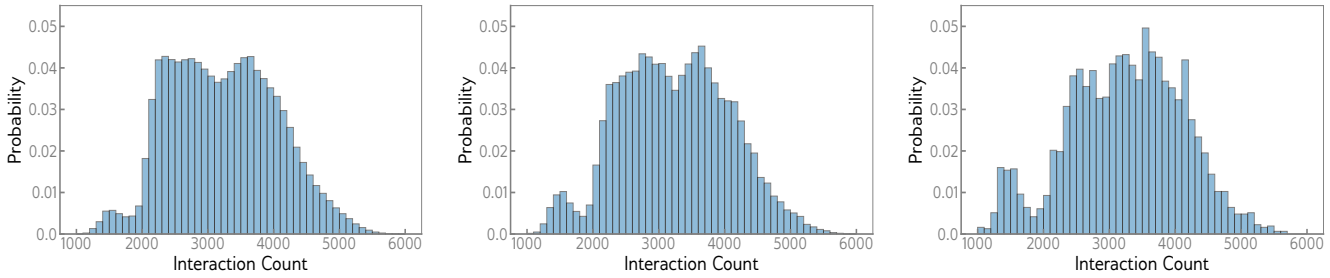
Our prior work used a granularity of 41 points for each parameter with 3 parameters. It is not feasible to use such granularity with 5 parameters due to computational time limitations as explained earlier. Instead, we use 17 points for each parameter. This corresponds to 16 intervals between each value for a parameter. This count has the benefit that subgrids with 8 intervals (9 points) and 4 intervals (with 5 points) can be analyzed to see how the smaller subgrid differs from the larger grid. Use of 32 intervals per parameter would, on the other hand, be computationally too expensive.

Figures 3a, 3b, and 3c show the histograms for the entire grid and the next two smaller subgrids respectively. These figures do not show any apparent convergence.

If we compare Figure 3 with Figure 4a, the latter being the LDS sweep with 17^5 trajectories, then we can see that the large lattice sweep is moving toward the LDS figure, but has not converged to it yet. For example, the first small peak in Figure 3c, which is absent in the LDS, is in the process of being eliminated. The two modes clear in LDS are also being better defined.

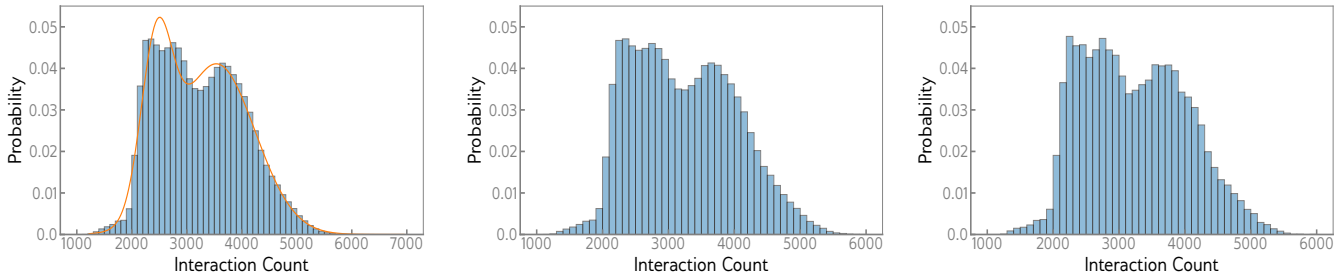
In contrast to the lattice sweep, comparing Figure 4c with Figure 4a, we can see that the LDS sweep appears to converge even with fewer than 33,000 trajectory files. The actual convergence criterion that we specified led to convergence with around 262,000 trajectories, as shown in Figure 4b. These figures suggest that a less restrictive convergence criterion could have been as accurate, while leading to a smaller computational time.

We can make the following observations from these results. The first is that LDS led to a factor 5 improvement in performance over the lattice (around 262,000 trajectory files instead of $17^5 \approx 1.4$ million with the lattice), and it also gave much more accurate results. In order to get more accurate solutions with Lattice, the next bigger size would be a $33^5 \approx 39$ million point lattice. Even if this were accurate, the LDS would have obtained accurate results with two orders of magnitude fewer points with the current convergence criterion. If we could



(a) (b) (c)

Fig. 3: Histograms for Lattice Sweep. (a) Histogram for 17^5 grid. (b) Histogram for the subgrid of size 9^5. (c) Histogram for the subgrid of size 5^5.



(a) (b) (c)

Fig. 4: Histograms for Low Discrepancy Sweep. (a) Histogram for 17^5 points. (b) Histogram for subset of data of size 262144. (c) Histogram for subset of data of size 32768.

change the convergence criterion to stop with around $33,000$ trajectory files, then this would be a three orders of magnitude improvement in performance. This relative advantage increases with dimension, and thus, our results show that LDS is far superior to Lattice.

Figure 5 shows the values of the four convergence criteria (relative differences in certain statistics) as a function of the number of trajectories. It appears that the skewness and kurtosis are particularly effective in checking for convergence, while mean and standard deviation differences are small, even in the unconverged region. Given that the histogram converged by 33,000 trajectory files, it appears that a 2% relative difference threshold for skewness and kurtosis would have been adequate to indicate convergence.

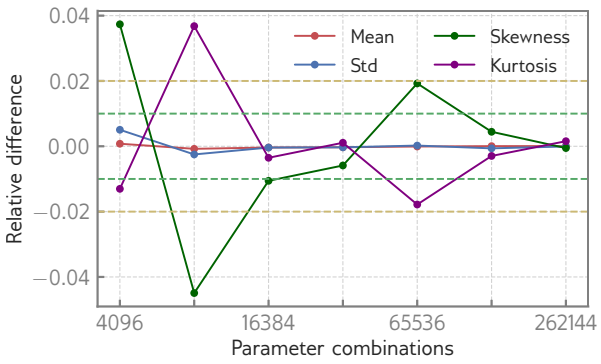


Fig. 5: Low Discrepancy Sweep: Relative difference in statistics used in convergence check

VI. LOAD BALANCING

A. *Parallelization*

We first summarize the parallelization scheme for the analysis, as shown in Algorithm 2, and then discuss the load balancing issue. The SPED simulation produces one output trajectory file for each choice of parameter combination. Each trajectory file from SPED is analyzed sequentially by only one core, but each core analyzes several files.

All cores will synchronize periodically to check for convergence. Frequent checks for convergence are needed to ensure that we don't continue the computation much after convergence has taken place. We output the histogram of contact probabilities once the convergence has taken place. The convergence checks require an Allreduce operation to evaluate the bin probability based on data from all the processes. Each process replicates the convergence test to come to the same conclusion on whether convergence has occurred. Computing the final histogram requires one Reduce operation.

Synchronization happens during the convergence check, at exponentially increasing intervals, with the first synchroniza-

tion happening after each process has analyzed $N_0 = 4$ files. We would like the load to be fairly well balanced between each synchronization point. The load balance at the later convergence checks are more important, because they take more time. However, if the convergence is fast, then the load imbalance during the first few checks would also matter.

The Lattice sweep does not have convergence checks. A simple cyclic distribution of parameters to processes gives good load balance, as shown in Section VI-B. The load for each parameter combination depends, to a large extent, on the number of time steps in the passenger movement simulation, which in turn depends on the values of the parameters. If the parameters are uniformly distributed across the processes, we would expect a good load balance. This, indeed, occurs with the cyclic parameter distribution on Lattice sweep.

If we generate parameters at random, then we can expect such a random parameter sweep to yield a well-balanced load on the cores. However, LDS covers the space uniformly, rather than randomly. Cyclic load balancing is not guaranteed to lead to uniform load distribution for the LDS. The uniformity of the load could also depend on the number of processes used for the following reason. The Halton sequence deals with each dimension independently. It divides the dimension into smaller intervals and distributes points uniformly over these intervals. The number of intervals increases progressively, thereby generating finer resolutions as the sequence progresses. The i^{th} dimension divides regions by some power of the i^{th} prime. The Scrambled Halton sequence just performs some local permutation of the order in which these regions are filled, which does not change the above pattern.

If any of the first d primes (where d is the number of parameters) divides the number of processes, then we may expect cyclic distribution to lead to some resonance with the natural period of the LDS, impacting the load balance. For example, each node on Blue Waters can support 32 processes. If we use 32 processes per node, then we can expect some impact on load balance because 2 is the first prime.

In a block distribution, we assign parameters corresponding to several consecutive points of an LDS sequence to one core. When we include convergence checks in the computation, the set of parameters between each convergence check is distributed in blocks. In some sense, this is like a block-cyclic distribution with block sizes increasing as the analysis progresses. If the block size is sufficiently large, then we expect the load to be well distributed, because parameters within each block will try to cover the parameter space uniformly. On the other hand, in the initial stages, the block sizes will be small, and such uniform coverage may not be obtained.

We also use a master-worker dynamic load balancing algorithm. The master assigns the next parameter to the next process that has completed its previous task. If we ignore the overhead of communicating with the master, then this algorithm has a theoretical worst case bound of twice the best case load, though it is usually much better than this bound in practice [5]. On the other hand, it could suffer from a lack of scalability when a large number of processes are used. In this method, the convergence check is performed by the master. Thus, workers don't synchronize amongst themselves for a convergence check.

B. Empirical Results

1) Computational platform: We analyze the load balancing algorithms on the Blue Waters machine at NCSA. We summarize the computational environment below.

The Blue Waters system is a Cray XE6/XK7 hybrid machine consisting of around 22,500 XE6 compute nodes all connected by the Cray Gemini torus interconnect. The XE6 dual-socket nodes are populated with 2 AMD Interlagos processors with a nominal clock speed of at least 2.3 GHz and 64 GB of physical memory. The file systems on Blue Waters are built with the Lustre file system technology.

The default programming environment, Cray (PrgEnv-cray 5.2.40) compiler suite, was used for compiling the codes. The MPI library used is cray-mpich/7.2.0. MPI timer routines are used for timing the code. The compute nodes use a 64-bit Linux OS. The TORQUE job scheduler was used to submit the batch jobs. The Cray Application launcher (aprun) utility was used to launch applications on compute nodes. We used 32 cores per node, running one process per core.

The contact threshold used in all these computations is 1.2 meters, corresponding to Ebola. The simulations involved deplaning.

2) Results: We define load imbalance as $|MaximumLoad - AverageLoad|/AverageLoad$. This is 0 when a load is perfectly balanced. When we use convergence checks, we replace the maximum load in the above formula by the sum of the maximum load between each convergence step.

Figure 6 compares the convergence of Lattice and LDS with the cyclic distribution for their entire data set in the absence of convergence checks. We can see that Lattice is well balanced, as expected. LDS has a poor load balance with 1000 and 1024 processes. The reason for this is that both of these are products of primes used in the LDS sequence. On the other hand, LDS has a good load balance with 1003 processes. This number was chosen so that it is not a multiple of any of the primes used in the LDS. In fact, LDS performs better than Lattice for this number, likely due to the number of lattice parameter values – 17 – dividing 1003. This is not a real defect in the Lattice sweep, because one would normally not have any reason to use a multiple of 17 processes. The results of this figure show that Lattice is easily load balanced through cyclic distribution, while LDS is not, unless we choose the number of processes carefully. Of course, this choice would leave some cores on a node unused, but the improvement in load balance makes it worthwhile. In addition, even with a poor load balancing choice, LDS would be much more preferable to Lattice due to its faster convergence.

While LDS may show good load balance with cyclic distribution when using a suitable number of processes, it may not necessarily perform well when we use convergence checks

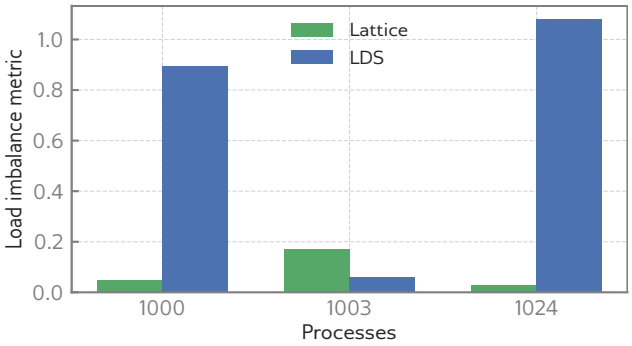


Fig. 6: Load imbalance for the processing the Lattice and LDS sweep outputs

that terminate the computation well before all the data is used; the previous results show the final impact when each process analyzes a large number of files whose varying loads could be smoothed out. This smoothing may not happen when we have a small number of files analyzed between each convergence check.

Figure 7, indeed, shows that even with 1003 processes, the load is not well balanced in the initial stages. Consequently, computations that lead to quick convergence may not be well balanced with a cyclic distribution of parameters.

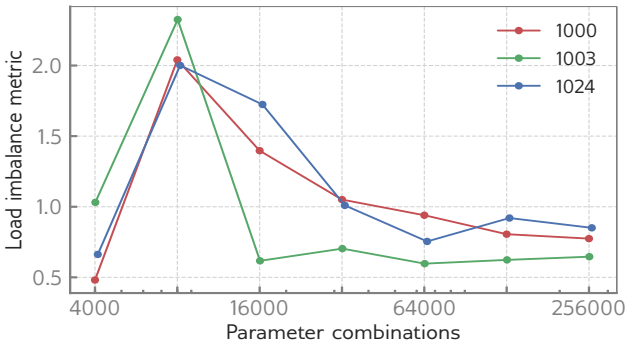


Fig. 7: Load imbalance for LDS with cyclic distribution

Figure 8 shows load imbalance with the block distribution of data. Note that the block distribution happens within each convergence check, and several convergence checks may occur. The results show that the potential advantages of the block distribution are not demonstrated when the block sizes are small. There is a trend toward improved load balance as the number of trajectories analyzed increases, but it is not adequate.

Figure 9 shows that the master-worker dynamic load balancing algorithm yields very good load balance, although it is not as well balanced as Lattice when the number of trajectories is small. But for the number of trajectories required in the problem evaluated in this paper, it has excellent load balance.

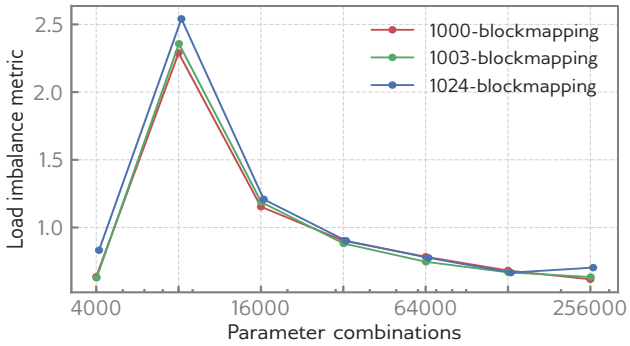


Fig. 8: Load imbalance for LDS with the block distribution

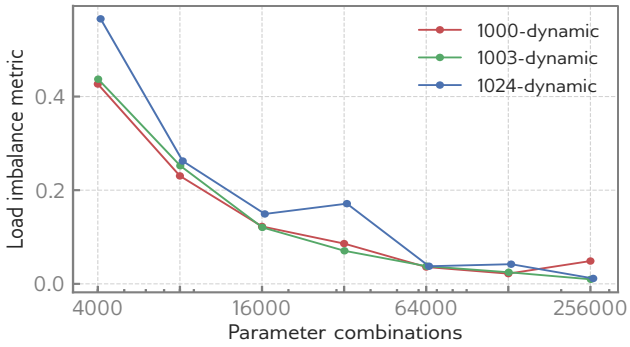


Fig. 9: Load imbalance for LDS with the dynamic load balancing

Finally, we show strong and weak scaling results for dynamic load balancing in Figures 10 and 11 respectively. These results include time for communication with the master, unlike the load balancing results which considered only the load. We can see the it has excellent weak scaling results. Strong scaling results show good efficiency until around 4000 processes, with a sharp drop-off after that.

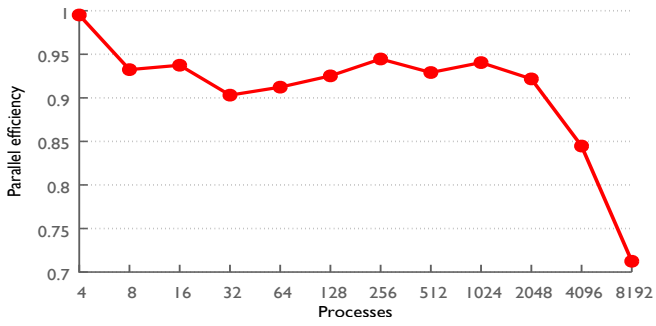


Fig. 10: Strong scaling of LDS with dynamic load balancing until convergence

In summary, LDS tends to have more load imbalance problems than Lattice, but any inefficiency here is more than

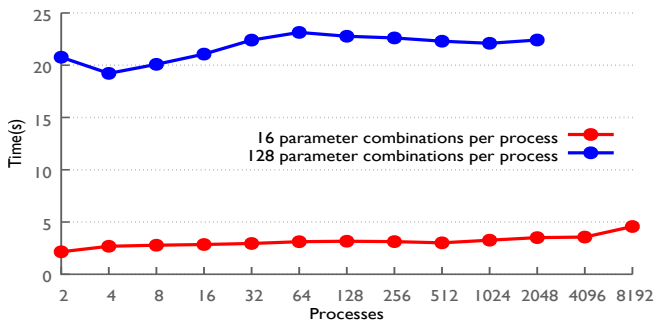


Fig. 11: Weak scaling: Time with 16 and 128 parameter combinations processed per process

compensated for with the improved convergence rate. Dynamic load balancing is effective for LDS, though, unless the number of processes is very large. Such a large number of processes will likely be required when the convergence takes a long time. In that case, it may be worthwhile to use cyclic load balancing with a carefully chosen number of processes, but with less frequent convergences than performed currently. Note that convergence checks for every $2^i N_0$ files can be replaced by checks every $r^i N_0$ files, for a suitable value of r in the theoretical analysis. Using a value of r larger than 2 can lead us to a situation closer to Figure 6 than to Figure 7.

VII. IMPLICATIONS FOR SCIENCE

The efficient LDS parameter sweep has enabled us to obtain application results that could not be obtained otherwise. In particular, the bimodal histogram for contacts, as shown in Figure 4a, has not been seen earlier. This figure also shows that it fits reasonably as the sum of two Gaussians. This suggests two different mechanisms at play. Application scientists ought to explore these further.

VIII. RELATED WORK

In our prior work, we have described the SPED model and scientific results from it, including the impact of boarding and deplaning procedures on infection transmission [3]–[8]. However, we had not discovered the bimodal distribution presented in this paper. Computational optimizations of the SPED model, and its parallelization, were discussed in [5]. The current manuscript, in contrast, is on issues related to a low discrepancy parameter sweep, which enables real-time analysis in the context of a decision meeting.

Conventional models for epidemic spread, such as Wells-Riley, are based on coarse-scale aggregate statistics, and are commonly used to predict disease transmission on aircrafts [18]. Alternatively, at a larger scale, contemporary network epidemic models, such as the meta-population model [19], are used to simulate the movements of diseases within networks, including the air transportation network. The latter models are at a coarse scale and deal with populations of entire cities with associated airports. In both cases, they rely on aggregate analysis, and so cannot account for the consequences of changes in human interaction patterns due to changes in boarding or deplaning procedures. In contrast to differential equation-based models that model aggregate quantities, such as number of infected individuals, agent-based models track each individual. EpiSimdemics [20]–[22] is such a model and has been applied at the scale of the entire nation, using massive parallelism. It is not feasible to track the continuous spatial movement of individuals on this scale. Instead, one looks at certain locations where people could meet, thus, discretizing the space. A social network model is created, and diffusion in this network is used to estimate the interactions between pairs. The advantage of this model is that it can account for individual movement with a large number of individuals, unlike conventional models. A limitation is that space is discretized, thus losing accuracy over a continuous space model like SPED.

There have been extensive theoretical studies on the properties of low-discrepancy sequences, also known as quasi-random sequences, in the context of their application to integration [13], [16]. This results in quasi-Monte Carlo methods for integration, which have better asymptotic properties than conventional numerical integration and also Monte Carlo integration. These methods have also been shown to be effective in certain application domains, such as finance, in practice, with a much better convergence rate than Monte Carlo [14], [15]. We have studied parallel low-discrepancy sequences in [15]. That work is not closely related to the results of the current manuscript; the current work does not parallelize the low-discrepancy sequence, nor does it study integration. Rather, the focus is on parallel computing issues that arise from the use of a low-discrepancy parameter sweep, which is a novel topic.

IX. CONCLUSIONS AND FUTURE WORK

We have demonstrated that the LDS parameter sweep can obtain a substantial reduction in computational effort over the current approach, and make feasible computation that was not feasible earlier. We have complemented theoretical analysis of convergence with empirical results that suggest rules of thumb to improve the convergence criterion. We have explored the load imbalance problem with LDS and related it to the number-theoretical characteristics of LDS. We have also identified techniques, that can lead to good load balancing under different applications scenarios.

The LDS sweep has the attractive property that for any value of N, the parameter space is well covered. However, if we consider the load as an additional dimension, then the resulting space is not well covered, which is the cause for the load imbalance. One direction for future work would be to identify the sensitivity of load to different parameters, and then come up with an LDS sweep that would automatically cover the augmented space of parameters plus load efficiently. This could lead to better load balance.

X. Acknowledgment

The authors thank NCSA for providing use of the Blue Waters supercomputer and the NSF for financial support. This research has been funded in part and used resources of the Argonne Leadership Computing Facility at Argonne National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under contract DE-AC02-06CH11357.

References

[1] Centers for Disease Control and Prevention, "Epidemiological notes and reports. Interstate Importation of Measles Following Transmission in an Airport–California, Washington," *MMWR-Morbidity and Mortality Weekly report*, vol. 32, pp. 210–216, 1982.

[2] M. R. Moser, T. R. Bender, H. S. Margolis, G. R. Noble, A. P. Kendal, and D. G. Ritter, "An outbreak of influenza aboard a commercial airliner." *American journal of epidemiology*, vol. 110, no. 1, pp. 1–6, 7 1979.

[3] S. Namilae, P. Derjany, A. Mubayi, M. Scotch, and A. Srinivasan, "Multiscale model for pedestrian and infection dynamics during air travel," *Physical Review E*, vol. 95, no. 5, p. 052320, 5 2017.

[4] S. Namilae, A. Srinivasan, A. Mubayi, M. Scotch, and R. Pahle, "Self-propelled pedestrian dynamics model: Application to passenger movement and infection propagation in airplanes," *Physica A: Statistical Mechanics and its Applications*, vol. 465, pp. 248–260, 1 2017.

[5] A. Srinivasan, C. D. Sudheer, and S. Namilae, "Optimizing massively parallel simulations of infection spread through air-travel for policy analysis," in *2016 16th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*, May 2016, pp. 136–145.

[6] S. Namilae, A. Srinivasan, A. Mubayi, M. Scotch, and R. Pahle, "Self-propelled pedestrian dynamics model: Application to passenger movement and infection propagation in airplanes," *Physica A: Statistical Mechanics and its Applications*, vol. 465, pp. 248–260, 2017.

[7] P. Derjany, S. Namilae, A. Mubayi, and A. Srinivasan, *Computational Model for Pedestrian Movement and Infectious Diseases Spread During Air Travel*. American Institute of Aeronautics and Astronautics, 2018, p. 0419.

[8] P. Derjany, S. Namilae, A. Mubayi, M. Scotch, and A. Srinivasan, "Effect of Pedestrian Movement on the Spread of Infectious Diseases during Air-travel," in *Transportation Research Forum Proceedings*, 2017.

[9] H. Casanova, D. Zagorodnov, F. Berman, and A. Legrand, "Heuristics for scheduling parameter sweep applications in grid environments," in *Proceedings of the 9th Heterogeneous Computing Workshop*, ser. HCW '00. Washington, DC, USA: IEEE Computer Society, 2000.

[10] Y. Naito, M. Taguchi, and S. Yoden, "A parameter sweep experiment on the effects of the equatorial qbo on stratospheric sudden warming events," *Journal of the atmospheric sciences*, vol. 60, no. 11, pp. 1380–1394, 2003.

[11] T. Kiss, P. Greenwell, H. Heindl, G. Terstyanszky, and N. Weingarten, "Parameter sweep workflows for modelling carbohydrate recognition," *Journal of Grid Computing*, vol. 8, no. 4, pp. 587–601, 2010.

[12] C. Youn and T. Kaiser, "Management of a parameter sweep for scientific applications on cluster environments," *Concurrency and Computation: Practice and Experience*, vol. 22, no. 18, pp. 2381–2400, 2010.

[13] A. Göncü, "Monte Carlo and Quasi-Monte Carlo Methods in Financial Derivative Pricing," *PhD dissertation, Florida State University*, 2009.

[14] W. J. Morokoff and R. E. Caflisch, "Quasi-Random Sequences and Their Discrepancies," *SIAM Journal on Scientific Computing*, vol. 15, no. 6, pp. 1251–1279, 11 1994.

[15] A. Srinivasan, "Parallel and distributed computing issues in pricing financial derivatives through quasi Monte Carlo," in *Proceedings 16th International Parallel and Distributed Processing Symposium*, April 2002.

[16] R. E. Caflisch, "Monte Carlo and quasi-Monte Carlo methods," *Acta Numerica*, vol. 7, p. 1, 1 1998.

[17] NASA, "Examining Spatial (Grid) Convergence." Tutorial on CFD Verification and Validation, 2008.

[18] J. K. Gupta, C.-H. Lin, and Q. Chen, "Risk assessment of airborne infectious diseases in aircraft cabins," *Indoor Air*, vol. 22, no. 5, pp. 388–395, 10 2012.

[19] V. Colizza, A. Barrat, M. Barthelemy, A.-J. Valleron, and A. Vespignani, "Modeling the worldwide spread of pandemic influenza: Baseline case and containment interventions," *PLOS Medicine*, vol. 4, no. 1, pp. 1–16, 01 2007.

[20] C. L. Barrett, K. R. Bisset, S. G. Eubank, X. Feng, and M. V. Marathe, "Episimdemics: An efficient algorithm for simulating the spread of infectious disease over large realistic social networks," in *2008 SC - International Conference for High Performance Computing, Networking, Storage and Analysis*, Nov 2008, pp. 1–12.

[21] J. S. Yeom, A. Bhatele, K. Bisset, E. Bohm, A. Gupta, L. V. Kale, M. Marathe, D. S. Nikolopoulos, M. Schulz, and L. Wesolowski, "Overcoming the Scalability Challenges of Epidemic Simulations on Blue Waters," in *2014 IEEE 28th International Parallel and Distributed Processing Symposium*, May 2014, pp. 755–764.

[22] A. Bhatele, J.-S. Yeom, N. Jain, C. J. Kuhlman, Y. Livnat, K. R. Bisset, L. V. Kale, and M. V. Marathe, "Massively parallel simulations of spread of infectious diseases over realistic social networks," in *Proceedings of the 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, ser. CCGrid '17. Piscataway, NJ, USA: IEEE Press, 2017, pp. 689–694.

Appendix A
Justification of Convergence Criterion

We first explain why we check for convergence with a geometrically increasing number of trajectories. This follows the standard practice as with other numerical methods. We then justify our convergence criteria for the low-discrepancy sequence (LDS) parameter sweep.

We estimate a quantity with an unknown exact value x using varying numbers (N) of trajectory files. Let us denote by x_i the estimate with $2^i N_0$ trajectory files. We consider the error $|x - x_i| \approx k/(2^i N_0)^p$, for some unknown p and k as explained earlier. In the asymptotic region, standard convergence analysis [17] gives:

$$x \approx x_i + \frac{k}{(2^i N_0)^p}.$$

From the above, we get:

$$|x_i - x_{i-1}| \approx k\left(\frac{1}{(2^{i-1}N_0)^p} - \frac{1}{(2^i N_0)^p}\right) = \frac{k}{(2^i N_0^p)}(2^p - 1),$$

which goes to 0 for large i. Note that the form $|x - x_i| \approx k/(2^i N_0)^p$ is more important than the assumption that we are in the asymptotic region. If we are not in the asymptotic region, then $|x_i - x_{i-1}|$ would be at most twice the quantity derived above.

We check for convergence by checking if $|x_i - x_{i-1}|$ is sufficiently small. We could define various metrics for what constitutes being sufficiently small. We choose a relative error $|x_i - x_{i-1}|/x_i$ that is 0.01 (that is, a 1% relative error). We make this condition more stringent by requiring this condition to be met in two consecutive convergence checks. There are two reasons for this more stringent requirement. The first is that LDS are often not in the asymptotic region for realistic values of N. The other is that this is a novel application of LDS, and we wish to be on the safe side. Empirical analysis later showed that we could have used a less stringent convergence criterion.