Evaluating Quasi-stationary Behaviour of Epidemic Models by Means of Parallel Aggregation/Disaggregation

M. S. BEBBINGTON*, University of Queensland

Abstract We describe how an aggregation/disaggregation method for finding quasi-stationary distributions of continuous-time Markov chains can be implemented on a massively parallel computer. The method is similar to an algebraic multigrid, using restriction operators that depend on the current iteration of the "solution", and Jacobi smoothers at each level of the multigrid. The method is illustrated using a simple epidemic model, and the performance compared to a sequential implementation as the size of the population increases. We find that the parallel implementation is superior in terms of time to convergence of the residual error, although inferior in terms of iterations required.

1. INTRODUCTION

Epidemics are essentially evanescent in nature, with eventual extinction certain, if only because the susceptible population is eradicated. However, the time required for this to occur may be very long, and the epidemic appears to observers to be in a stable state. This can be modelled by a continuous-time Markov chain with an absorbing class, which is evanescent on the remainder of the state-space. In most practical applications the time to absorption is large, and we are interested in predicting the apparently stable behaviour before evanescence. The equilibrium distribution of the Markov chain is of no use, being concentrated on the absorbing state. Hence we use the quasi-stationary distribution, which is basically the long-term distribution, conditional on non-absorption.

Finding the quasi-stationary distribution is equivalent to finding the eigenvector corresponding to the smallest eigenvalue of the transition-rate matrix. This has the incidental benefit of calculating this eigenvalue, which determines the persistence time for the epidemic. If the model is to be useful, the transition matrix is usually large, and even approximating the process by truncating the matrix leaves upwards of a hundred million elements. Obviously, numerical methods, efficient ones, are required.

A multigrid-type method for determining the smallest eigenvalue and associated eigenvector has recently been developed and shown to be sequentially efficient for a large class of problems (Stewart and Bebbington [1995]). The method is a natural candidate for implementation on a parallel machine, containing many inherently parallel elements. We will describe the algorithm and its parallel implementation and, using a simple epidemic model, compare its performance with the sequential algorithm. Means of further improvement will be briefly discussed.

2. QUASI-STATIONARY DISTRIBUTIONS

A continuous-time Markov chain is a stochastic process, which at time t is in a state $X(t) \in S$, where S is a countable state-space. If $p_i(t) = \Pr(X(t) = i)$, then the transition-rate matrix, or q-matrix, Q is specified by

$$\frac{d}{dt} p(t) = Q' p(t) \qquad (1)$$

where $p(t)$ is the vector $(p_i(t), i \in S)$ and the prime denotes the transpose. The off-diagonal elements of Q are non-negative, while the diagonal elements are non-positive. Now let us suppose that $C \subset S$ is an irreducible transient class, that is, $S = C \cup \overline{C}$ where $\overline{C}$ is an absorbing set. If S is finite then, under certain conditions, the q-matrix restricted to C has eigenvalues with negative real parts, the eigenvalue with maximal real part is real and of multiplicity one, and the corresponding left eigenvector, $m = (m_j, j \in C)$ has positive entries. Further, the *stationary conditional quasi-stationary distribution*,

$$\lim_{t\to\infty} \Pr(X(t) = j \,|\, X(0) = i, X(t) \in C) = \frac{m_j}{\sum_{k\in C} m_k}, \qquad (2)$$

$i, j \in C$, exists (Darroch and Seneta [1967]). Thus, the eigenvector m, when normalised, is an estimate of the long-term probability of being in each state *given that the process has not yet been absorbed*, and hence describes the apparent behaviour of the process to an observer. We note also that m is the only left eigenvector of the restricted q-matrix with all positive entries, and attracts trajectories on the space of probability distributions over C.

When S is infinite, it is usual to truncate the restricted q-matrix to an $n \times n$ matrix $Q^{(n)}$, and construct a sequence $\{m^{(n)}\}$ of corresponding left eigenvectors. The desired accuracy in the quasi-stationary distribution is achieved by making n as large as necessary (Tweedie [1973]).

3. MULTIGRID FOR M-MATRICES

Our objective is to find the left-eigenvector, corresponding to the eigenvalue with maximal real part, of the transition-rate

*Postal address: Department of Statistics, Massey University, Private Bag 11222, Palmerston North, New Zealand

matrix restricted to the transient class. That is, we must solve for x in the equation

$$x'Q = -\lambda x' \tag{3}$$

while simultaneously minimising λ, subject to $\lambda > 0, \sum_i x_i = 1$.

There are a number of techniques for computing extremal eigenvalues and their corresponding eigenvector. Factorisation techniques are limited, with dense matrix methods, to smaller scale Markov chains. Sparse matrix factorisation techniques, while allowing larger chains, also require additional memory due to the non-zero entries created by factorisation ("fill-in"), and are very difficult to parallelise. For these reasons, iterative techniques are used to find quasi-stationary distributions. However, the sheer size of the problem (Q might contain upwards of 10^5 non-zero elements) makes simple relaxation schemes impracticable.

Standard multigrid methods are developed with the object of solving systems of differential equations. The approach taken here was developed for Markov chain eigenvalue problems by Stewart [1992], and Stewart and Bebbington [1995], based on the fact that $-Q'$, which we shall henceforth denote as A, is a singular M-matrix (see, for example, Seneta [1981]) with non-negative column sums.

In common with standard multigrid methods we must construct a hierarchy of finite dimensional vector spaces indexed by m, $X^{(m)}$, a matrix $A^{(m)}$ and corresponding eigenvector $x^{(m)} \in X^{(m)}$ on each space, restriction and prolongation (or interpolation) operators (respectively r_m and p_m) between spaces, and a smoother operator S_m on each space. The most common multigrid algorithm is known as the "V-cycle". Beginning with data at the unaggregated, or finest, level, it progressively transfers data to coarser levels, smoothing at each step. Subsequently, this smoothed data is used to update progressively finer levels until the finest level is again reached. This "down and up" procedure is often visualised as a "V". The smoothing on progressively coarser grids reduces lower frequency components of the error (see, for example, Hackbusch [1985]).

The basic V-cycle algorithm for finding the eigenvalue and associated eigenvector of an M-matrix can be described in the following way:

function Vcycle($A^{(m)}, x^{(m)}, \lambda, m$)
  repeat ν_1 times
    $(x^{(m)}, \lambda) \leftarrow S_m(A^{(m)}, x^{(m)}, \lambda)$
  $x^{(m)} \leftarrow r_{m-1}(x^{(m)})$
  if $m \neq 0$ **then**
    construct coarse-grid operator $A^{(m-1)}$
    $x^{(m)} \leftarrow p_{m-1}(\text{Vcycle}(A^{(m-1)}, x^{(m-1)}, \lambda, m-1), x^{(m)})$
  repeat ν_2 times
    $(x^{(m)}, \lambda) \leftarrow S_m(A^{(m)}, x^{(m)}, \lambda)$
  return $(x^{(m)}, \lambda)$

The restriction and prolongation operators are defined in the context of a pair of "grids". The coarse grid is formed as a systematic, pre-defined, partition of the fine grid. Suppose the fine grid (level m, say) can be partitioned as $G = \{G_1, G_2, \ldots, G_n\}$, where no G_i is empty, and $G_i \cap G_j = \emptyset$ whenever $i \neq j$. The coarse-grid vector $r_{m-1}(x)$ given by the restriction operator is a n-dimensional vector with components

$$r_m(x^{(m)})_i = \sum_{j \in G_i} x_j^{(m)} \tag{4}$$

When prolongating, the probability is distributed proportionately to the (original) x_j values, and thus

$$p_{m-1}(y)_j = \frac{y_i x_j^{(m)}}{\sum_{k \in G_i} x_k^{(m)}} \tag{5}$$

where $j \in G_i$. This is necessary in order to ensure that a correct solution on a finer grid is still correct after a V-cycle. If $x_j^{(m)} = 0$ for all $j \in G_i$, then the prolongation operator distributes the probability uniformly over $j \in G_i$. Finally, the coarse-grid matrix is the $n \times n$ matrix given by

$$A_{ij}^{(m-1)}(x^{(m)}) = \frac{\sum_{k \in G_i} \sum_{l \in G_j} A_{kl}^{(m)} x_l^{(m)}}{\sum_{l \in G_j} x_l^{(m)}} \tag{6}$$

This choice, of $A^{(m-1)} = r_{m-1} A^{(m)} p_{m-1}$, is the Galerkin coarse-grid operator, a standard choice in multigrid solvers because of its convergence and consistency properties (Hackbusch [1985]). Note that $A^{(m)}$ must be calculated anew at each restriction step, is unchanged in the prolongation step, and that the matrix at the finest level is invariant.

The properties of the aggregation/disaggregation procedure are investigated by Stewart and Bebbington [1995], where the algorithm is shown to converge for a variety of models with widely differing behaviours. It will suffice for our purposes to observe that if $A^{(m)}, x^{(m)}, \lambda$ are such that $A^{(m)} x^{(m)} = \lambda x^{(m)}$, where λ is the smallest eigenvalue of $A^{(m)}$, then this relationship is preserved by the restriction and prolongation operators. The same is necessarily true of the smoother, and so the solution (the eigenvector of A corresponding to the smallest eigenvalue) is a fixed point of the multigrid algorithm.

Since the operators above are defined between grids, complete formulation of the algorithm now requires only a procedure for partitioning the state-space, creating a hierarchy of finer and coarser grids. It is desirable in Markov chain models that the states with the strongest interactions be lumped together (see, for example, Cao and Stewart [1985], Meyer [1989]). However, since the set of non-zero transition rates is largely translation invariant, and the models such that we usually have only nearest-neighbour transitions, it is appropriate to

partition the state-space into sets of adjacent states. Since, to conform to the processor geometry of the machine, the truncated state space will be two-dimensional, the simplest procedure is for each 2×2 square of states in the fine grid to form a single state in the coarse grid.

4. IMPLEMENTATION ON THE MASPAR MP-1204 MACHINE

In this section we will discuss how the multigrid algorithm operates on a 4096 (64×64) processor SIMD machine.

A few remarks about the geometry of the problem are now in order. Due to memory constraints, the present program is designed to handle up to (256×256) states. Since this is more than the number of processors we obviously have two regimes: Up to a certain level of the multigrid each state has sole occupancy of a processor. All levels below this are equally fast. Above this level we must have multiple states on each processor, hence we have either a sequential element on each processor, or sequential activation of processors (depending on the allocation of states to processors), and the speed decreases with increasing level. Conversely, at all multigrid levels below 64×64 states some of the processors are unused during the V-cycle, to the extent where at the coarsest level only 4 processors out of the 4096 are in use.

For purposes of reference, in terms of the grid structure outlined above, level 0 will be the coarsest level with 2×2 states, level 5 has 64×64 states, and level 7 is the finest level with 256×256 states. At each of level 6 and 7, states are indexed by an array.

The smoother will be the Jacobi smoother, as follows:

function $(\hat{x},\hat{\lambda}) = \text{smoother}(x, A, \lambda)$
  $y \leftarrow x/\lambda$
  solve $D\hat{y} = (L+U)y + x$ for $\hat{y}$
  $\hat{x} \leftarrow \hat{y} / 1'\hat{y}$
  $\hat{\lambda} \leftarrow (1'\hat{y})^{-1}$

where D, L, U are respectively the diagonal, lower triangular and upper triangular parts of A. The smoother is obtained as a Jacobi iteration of the linear equations for the inverse power method

$$A\hat{y} = x, \quad \hat{x} = \hat{y}/\|\hat{y}\|_1 \tag{7}$$

We see that a natural consequence of the algorithm is the determination of the eigenvalue, the decay parameter for the process. That is, the probability that the epidemic is extant at time t is of order $e^{-\lambda t}$.

The Jacobi smoother is uniquely suited to a parallel implementation since it requires only one communication with neighbouring sites, and all sites (or at least 4096 of them) can be updated simultaneously. However, this is a relatively inefficient smoother, and in order not to degrade the parallel efficiency of the procedure, we shall limit ourselves on levels 6 and 7 to only one application of the smoother at each step, i.e., $\nu_1 = \nu_2 = 1$ (see also Section 7). The optimum number of smoothing repetitions at the coarser levels will be the first aspect considered.

The matrix A and the (Hadamard-type product) matrix $B = A \otimes x$, defined by $B_{ij} = A_{ij}x_j$, are represented at each point of the state space by a one-dimensional array, indexed by direction (1 = E, 2 = S, et cetera, the diagonal elements having index 0). The interpolation operator (5) is simply implemented. The restriction operators (4) and (6) proceed as follows:

function restrict $(A^{(m+1)}, x^{(m+1)})$
  $x^{(m)} \leftarrow r_m(x^{(m+1)})$
  $B \leftarrow A^{(m+1)} \otimes x^{(m+1)}$
  for i, j
    $A_{ij}^{(m)} \leftarrow \sum_{k\in G_i}\sum_{l\in G_j} B_{kl}^{(k+1)} \Big/ x_j^{(m)}$
  return $(x^{(m)}, A^{(m)})$

Finally, the smoother operator is implemented as:

function smoother $(A^{(m)}, x^{(m)}, \lambda)$
  $B^{(m)} \leftarrow A^{(m)} \otimes x^{(m)}$
  for i
    $x_i^{(m)} \leftarrow \left(x_i^{(m)} - \sum_{j\neq i} B_{ij}^{(m)}/\lambda \right) \Big/ A_{ii}^{(m)}$
  $x^{(m)} \leftarrow x^{(m)} / 1'x^{(m)}$
  $\lambda \leftarrow \sum_i \sum_j A_{ij}^{(m)} x_j^{(m)}$
  return $(x^{(m)}, \lambda)$

The algorithm terminates when the relative residual error, $\|Ax - \lambda x\|_2 / \|x\|_2$, becomes less than a specified tolerance, usually 10^{-7}. In the example following, the norm of the matrix A is of the order of 10^4 to 10^5, and so at termination the residuals are of order 10^{-11} times the norm of the matrix.

5. TEST PROBLEM AND COMPUTATIONAL RESULTS

Each state will be a pair of nonnegative integers, (u, v). Truncation is handled by setting all transition rates out of the truncated space equal to zero, corresponding to a reflecting boundary. This is reasonable provided the bulk of the quasi-stationary distribution is within the truncated state space. However, the eigenvalue computed will be larger than that for the full state-space, since we omit the "tail" of the distribution furthest from the absorbing set.

Our example is the simple epidemic model of Ridler-Rowe [1967]. Here u is the number of people susceptible to a

particular disease, and v is the number of people infected (and infective). If $i=(u,v)$ then the non-zero transition rates are

$$q_{ij}=\begin{cases}\alpha & \text{if } j=(u+1,v)\\ \beta uv & \text{if } j=(u-1,v+1)\\ \gamma v & \text{if } j=(u,v-1)\end{cases} \qquad (8)$$

Thus a susceptible becomes infected at rate βuv, an infective recovers (or dies) at rate γv, and a new individual becomes susceptible at rate α. A sequential procedure for finding the quasi-stationary distribution, based on the Arnoldi algorithm, has recently been developed by Pollett and Stewart [1994].

The restriction operator (6) requires all non-zero values of $A_{ij}^{(m)}$. For the epidemic problem, at the finest level, there are four non-zero components, the three given by (8), and the exit rate(s) $A_{ii}^{(m)}$. However, when using a rectangular grid with diagonal transitions, there may be additional possible transitions at coarser levels, as illustrated in Figure 1.

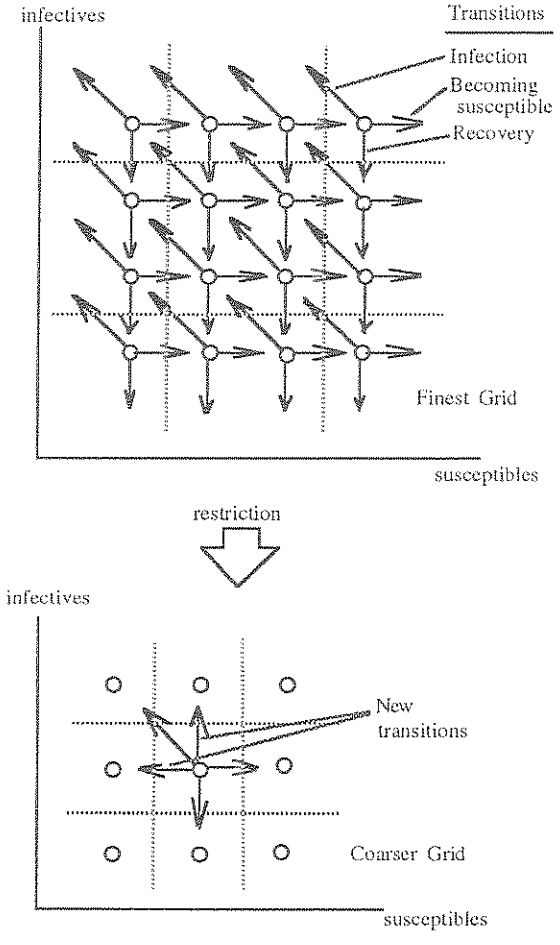

Figure 1: Partitioning for the epidemic problem. Solid lines represent non-zero values of A_{ij}.

It is obvious from the rates (8) that $\{(u,v): v=0\}$ is an absorbing set, corresponding to the extinction of the disease.

The associated deterministic system of differential equations,

$$\frac{d}{dt}u=\alpha-\beta uv, \quad \frac{d}{dt}v=\beta uv-\gamma v \qquad (9)$$

has $\{(u,v): v=0\}$ as an unstable invariant set, while if $v(0)>0$, the trajectory approaches the unique stable equilibrium point $(\bar{u},\bar{v})=(\gamma/\beta,\alpha/\gamma)$. Although, in the Markov chain model, eventual absorption is certain, trajectories will usually remain in the neighbourhood of $(\bar{u},\bar{v})$ for long periods of time, as in figure 2.

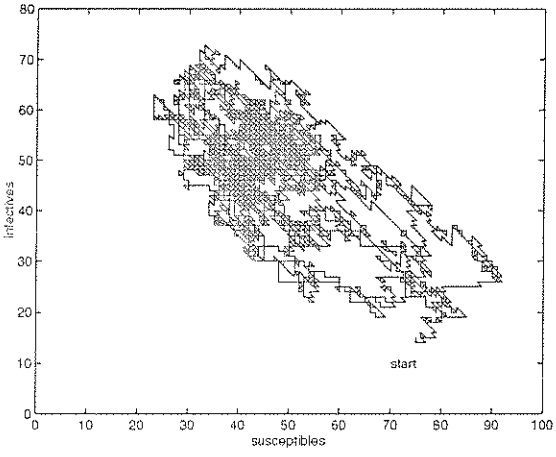

Figure 2: A partial realisation of the simple epidemic with $\alpha=2025$, $\beta=1$, $\gamma=45$.

It is this noisy behaviour around $(\bar{u},\bar{v})$ that the quasi-stationary distribution, shown in Figure 3, describes.

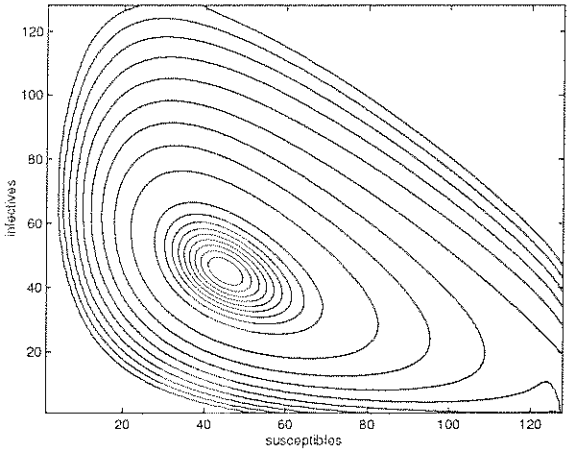

Figure 3: The quasi-stationary distribution of the epidemic model with $\alpha=2025$, $\beta=1$, $\gamma=45$, for $N=128$. Contours are at $10^{-10},10^{-9},\ldots,10^{-2}$, $0.1,0.2,\ldots,0.9$ of the maximum probability.

We will use this test problem to investigate the properties of the parallel implementation. As mentioned in the previous section, we have two distinct regimes, one in which the

smoother is a "single" parallel operation, and one in which the smoother contains sequential elements. In order to gain maximum benefit from the parallelism (as opposed to the algorithm), we will use only one smoother repetition at levels 6 and 7. This leaves open the question of the optimum number of smoothings at coarser levels. Table 1 gives the number of V-cycles required, and the CPU time used, for the epidemic model with parameters $\alpha = 2025$, $\beta = 1$, $\gamma = 45$, for $N = 64$, and varying numbers of smoother repetitions.

Iterations	V-cycles	CPU (s)
1	320	57
2	152	27
3	155	28
4	140	25
5	143	25
10	151	27
15	155	28

Table 1: Effect of multiple smoothing.

We see that multiple smoothing is a very efficient (in terms of time) operation at coarser levels. It also seems from the table that excessive smoothing is detrimental, probably due to the effects on the coarsest levels.

Having identified $\nu_1 = \nu_2 = 4$ as an optimum number of smoothing repetitions, we can now consider how the algorithm performs with the epidemic problem given above, for varying values of N. For comparison, we provide performance details of the sequential implementation with the Gauss-Seidel smoother (Stewart and Bebbington [1995]). The parallel implementation was on the MasPar MP-1204 machine (peak performance 135 MFlops), and the sequential implementation on a SPARC 10 (peak performance 10 MFlops). However, we are really interested in how the performance of the parallel and sequential versions vary with N. Hence we allow each version those advantages inherent to them, notably a more powerful smoother for the sequential, and multiple smoothing at coarser levels for the parallel. Other improvements should apply more or less equally.

		Parallel (Jacobi)		Sequential (Gauss-Seidel)	
N	λ	V-cycles	CPU (s)	V-cycles	CPU (s)
64	8.87×10^{-4}	140	25	105	36
128	4.66×10^{-6}	129	24	126	182
256	4.21×10^{-6}	205	38	133	791

Table 2: Dependence of convergence on N. *Note*: The sequential implementation with the Jacobi smoother takes 2.5 to 3 times as long to converge as that with the Gauss-Seidel smoother. Using multiple smoothing iterations in the sequential implementation results in a speed-up of 10% or so (and a 40% reduction in V-cycles).

From Table 2 we observe that using only one repetition of the smoother at the finer levels, and the less powerful smoother, affects the number of V-cycles required . Unlike the sequential case, where the number required increases only slowly with N, the parallel implementation requires half again as many cycles when increasing N from 128 to 256. On the other hand, the computation time, as opposed to iterations, does not exhibit the geometric increase of the sequential case. An alternative sequential algorithm with differing behaviour is the iterative Arnoldi method of Pollett and Stewart [1994], although the criteria for convergence are not well understood at present. This can also be used as a smoother in the V-cycle (see Stewart and Bebbington [1995]).

Some unexplained variations in Table 2 are at least partly due to the truncation of the state space, and hence of the matrix A, as is indicated by Table 3. Here we have again used the epidemic problem with $\beta = 1$, but now the remaining parameters are chosen so that $(\overline{u}, \overline{v}) = (N/4, N/4)$.

N	λ	V-cycles	CPU (s)
64	3.44×10^{-2}	120	22
128	2.88×10^{-4}	132	24
256	7.05×10^{-9}	181	32

Table 3: Effect of $(\overline{u}, \overline{v})$ on the speed of the parallel implementation.

Stewart [1992] indicates that the rate of convergence for the sequential implementation is roughly proportional to $1/\sqrt{N}$ when $(\overline{u}, \overline{v})$ is scaled proportionally to N. Here we see that the larger problems converge faster than one would expect, because the processors are idle less. Tables 2 and 3 would also seem to indicate that the algorithm does not suffer from having only one smoother repetition at the finest level, the increase in V-cycles occurring when a second level with single smoothing is introduced.

6. CONCLUSIONS

We have seen in the previous section that our two objectives, parallel and numerical efficiency, are somewhat opposed. A policy of minimising sequential elements hampers convergence, increasing the number of iterations required.

N	64	128	256
Speed-up	0.12	0.57	1.35

Table 4: "Speed-up" from parallel implementation, calculated from Table 2 and Table 3.

Taking the relative machine performances into account, we see in Table 4 that the parallel implementation becomes efficient as N increases, partly because the processors are

"busier", but largely because the sequential implementation becomes slower geometrically. However, at some point the sequential elements on the finer grids of the parallel version will produce an effect.

7. EXTENSIONS AND OTHER MODELS

It is possible to accelerate the algorithm by using a W-cycle rather than a V-cycle. This is based on the fact that smoothing on the coarser grids has a greater effect than smoothing on the finer grids. The V-cycle is so called because the eigenvector is restricted down to the coarsest level, and then prolongated back to the finest level. If, instead, we only prolongate partway back to the finest level, and then restrict again, eventually returning to the finest level, we produce a "coarse-grid intensive", or W-cycle, as shown in Figure 4.

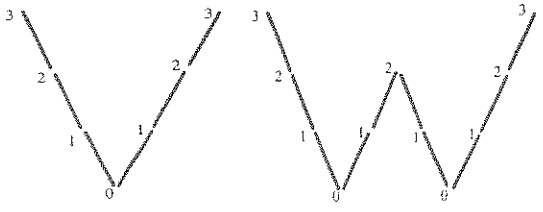

Figure 4: Left, a V-cycle, and right, a 30203 W-cycle

Labelling these cycles in the obvious notation (e.g., 703050307 restricts from level 7 to level 0, prolongates to level 3, restricts back to level 0 and so on), Table 5 shows the effect of various W-cycles for the epidemic process with $N = 256$, $\beta = 1$, $\overline{u} = \overline{v} = 45$.

W-cycle	cycles	CPU (s)
703050307	193	35
705030507	178	32
735030537	182	33
705050507	171	33
70503030507	173	31
(V-cycle)	205	38

Table 5: Effect of W-cycles

The small improvement, on the order of 15%, is partly due to the fact that our implementation is already "coarse-grid intensive". Also, in order for the W-cycle to work well, the V-cycle must converge at a sufficiently fast rate (Hackbusch [1985]), a requirement difficult to achieve with the Jacobi smoother.

Although the Jacobi smoother is suited to parallel implementation, it is also possible to use a multi-colour Gauss-Seidel smoother (Hackbusch [1985]). This uses the Gauss-Seidel smoother, but in a single "pass" updates only those sites which have no communication with each other in the form of non-zero transition rates. We see from Figure 1 that the epidemic model will therefore require 3 passes in order to update all the sites. An obvious potential improvement is to use the multicolour Gauss-Seidel at the finest (levels 6 and 7) levels, and Jacobi below this.

The algorithm has been tested on a number of other two-dimensional problems. Among these are the Ross malaria model (see Näsell [1991]), the cancer model of Bartoszynski and Puri [1983], predator-prey and competition models. The cubic autocatalator model (Gray and Scott [1984]) also demonstrated that the algorithm could handle multimodal distributions (Stewart and Bebbington [1995]).

ACKNOWLEDGMENTS

I would like to thank David Stewart for many fruitful discussions, and Phil Pollett for his helpful suggestions. This worked was funded by The Australian Research Council.

REFERENCES

Bartoszynski, R., and Puri, P.S., On two classes of interacting stochastic processes arising in cancer modeling, *J. Appl. Prob., 15,* 695-712, 1983.

Cao, W., and Stewart, W.J., Iterative aggregation/disaggregation techniques for nearly uncoupled Markov chains, *J.A.C.M., 32,* 702-719, 1985.

Darroch, J.N., and Seneta, E., On quasi-stationary distributions in absorbing continuous-time finite Markov chains, *J. Appl. Prob., 4,* 192-196, 1967.

Gray, P., and Scott, S.K., Autocatalytic reactions in the CSTR: oscillations and instabilities in the system $A + 2B \rightarrow 3B$, $B \rightarrow C$, *Chem. Eng. Sci., 39,* 1087-1097, 1984.

Hackbusch, W., *Multi-Grid Methods and Applications,* Springer-Verlag, Berlin, 1985.

Meyer, C.D., Stochastic complementation, uncoupling Markov chains and the theory of nearly reducible systems, *SIAM Review, 31,* 240-272, 1989.

Näsell, I., On the stationary distribution of the Ross malaria model, *Math. Biosciences, 107,* 187-207, 1991.

Pollett, P.K., and Stewart, D.E., An efficient procedure for computing quasi-stationary distributions of Markov chains with sparse transition structure, *Adv. Appl. Prob., 26,* 68-79, 1994.

Ridler-Rowe, C.J., On a stochastic model of an epidemic, *J. Appl. Prob., 4,* 19-33, 1967.

Seneta, E., *Non-negative Matrices and Markov Chains,* 2nd ed, Springer-Verlag, New York, 1981.

Stewart, D.E., *A multigrid method for computing quasi-stationary distributions of continuous-time Markov chains,* ANU Advanced Computation Report ACTR-6-04-1992, 1992.

Stewart, D.E., and Bebbington, M.S., An iterative aggregation/disaggregation procedure for modelling the long term behaviour of continuous-time evanescent random processes, Submitted for Publication, 1995.

Tweedie, R.L., The calculation of limit probabilities for denumerable Markov processes from infinitesimal properties, *J. Appl. Prob., 10,* 84-99, 1973.