Available online at www.sciencedirect.com
ScienceDirect
Procedia Computer Science 1 (2010) 327–335
Procedia Computer Science
www.elsevier.com/locate/procedia

International Conference on Computational Science, ICCS 2010

A massively parallel semi-Lagrangian algorithm for solving the transport equation

J.Russell Manson[a*], Dali Wang[b], Steve G Wallis[c], Richard Page[a], Michael J Laielli[a]

[a]*Richard Stockton College of New Jersey, PO Box 195, Pomona, NJ 08240, USA*
[b]*Oak Ridge National Laboratory, P.O. Box 2008, Oak Ridge, TN 37931-6335, USA*
[c]*Heriot-Watt University, Riccarton, Edinburgh, EH14 4AS, UK*

Abstract

The scalar transport equation underpins many models employed in science, engineering, technology and business. Application areas include, but are not restricted to, pollution transport, weather forecasting, video analysis and encoding (the optical flow equation), options and stock pricing (the Black-Scholes equation) and spatially explicit ecological models. Unfortunately finding numerical solutions to this equation which are fast and accurate is not trivial. Moreover, finding such numerical algorithms that can be implemented on high performance computer architectures efficiently is challenging. In this paper the authors describe a massively parallel algorithm for solving the advection portion of the transport equation. We present an approach here which is different to that used in most transport models and which we have tried and tested for various scenarios. The approach employs an intelligent domain decomposition based on the vector field of the system equations and thus automatically partitions the computational domain into algorithmically autonomous regions. The solution of a classic pure advection transport problem is shown to be conservative, monotonic and highly accurate at large time steps. Additionally we demonstrate that the algorithm is highly efficient for high performance computer architectures and thus offers a route towards massively parallel application.

Keywords: advection; semi-lagrangian; parallel; MPI; scalability;

1. Introduction

As computer modeling becomes ever more pervasive so the reliance, indeed dependence, on model results has grown. Yet increasingly people with less expertise use these models and it is therefore incumbent upon model developers to re-examine modeling approaches for accuracy, efficiency and integrity. High model efficiency (i.e. rapid turn-around time) is essential since it allows for: (i) larger domains to be simulated at multiple scales; (ii) more complicated physical, chemical and/or biological mechanisms to be incorporated within the model; (iii) the inherent uncertainty in the model mechanisms to be quantified through a multi-simulation (Monte-Carlo) analysis and (iv) multiple model realizations to be constructed for parameter optimization [1]. There are essentially two ways to achieve this. One way is to improve the underlying algorithm as implemented on a single CPU. The second way is

* Corresponding author. Tel.: +1-609-652-1776; fax: +1-609-626-5515
Email address: russell.manson@stockton.edu

1877-0509 © 2010 Published by Elsevier Ltd.
doi:10.1016/j.procs.2010.04.036

reducing wall-clock time through parallel programming, i.e. running the model on several CPUs simultaneously [2]. The main goal of this research was to take an existing semi-Lagrangian transport modeling algorithm, demonstrate its accuracy and efficiency on a single processor computer architecture and then adapt it for implementation it on a parallel computational platform, such as a CPU cluster in a novel way. In this paper we describe how this particular algorithm may be applied in two-dimensions in just such a massively parallel way. Scalability of the parallel code is achieved through reducing the times of synchronization and eliminating the transfer of data by the introduction of a natural coordinate system (NCS); these are extremely important issues for distributed parallel computation. The approach advocated here employs an intelligent domain decomposition based on the vector field of the system equations and thus automatically partitions the computational domain into algorithmically autonomous regions.

2. Mathematical formulation of transport in the natural coordinate system

The equation describing advective-diffusive transport in a two-dimensional Cartesian system is given below.

$$\frac{\partial c}{\partial t}+\frac{\partial uc}{\partial x}+\frac{\partial vc}{\partial y}=\frac{\partial}{\partial x}\left(D_x\frac{\partial c}{\partial x}\right)+\frac{\partial}{\partial x}\left(D_{xy}\frac{\partial c}{\partial y}\right)+\frac{\partial}{\partial y}\left(D_{xy}\frac{\partial c}{\partial x}\right)+\frac{\partial}{\partial y}\left(D_y\frac{\partial c}{\partial y}\right)+f(c) \quad (1)$$

In equation (1), c (x, y, t) is value of the transported scalar; u(x,y) and v(x,y) are the fluid velocities in the Cartesian coordinate directions, x and y ; D_x , D_{xy} and D_y are the diffusion coefficients; and t is the time. For steady flows this equation may be recast in a natural coordinate system [equation (2)] in which the principal coordinate (ξ) aligns with the flow-field streamlines and the secondary coordinate (η) is mutually orthogonal, [9].

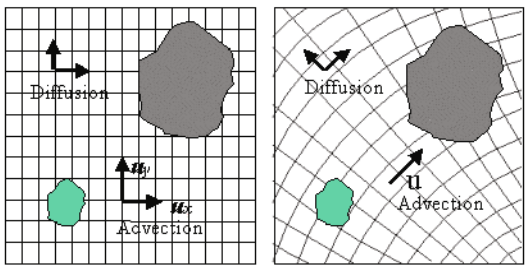
$$\frac{\partial c}{\partial t}+\frac{\partial (qc)}{\partial \xi}=\frac{\partial}{\partial \xi}(D_\xi\frac{\partial c}{\partial \xi})+\frac{\partial}{\partial \eta}(D_\eta\frac{\partial c}{\partial \eta})f(c) \quad (2)$$

In equation (2), q is the resultant velocity, $q=[u^2+v^2]^{0.5}$ where u(x,y) and v(x,y) are the velocities in the x and y directions. D_ξ is the streamwise diffusion coefficient, D_η is the transverse diffusion coefficient, ξ is the streamwise coordinate, η is the transverse coordinate and t is time. The final term in equation (1), f(c), represents some transformation (expressed here as a function of the scalar value, e.g. for a first order transformation, f(c) = -kc). The mapping from (x,y) to (ξ,η) may be found in a number of ways, [9]. For simple flow fields exact relationships may be defined using stream functions.

3. Numerical solution and computational implementation

Now that we have defined the natural coordinate system (ξ,η) we are in a position to define the computational grid. The computational grid is just a spatial discretization of the PDE [equation (2)] but may also be thought of in a physical way. It consists of M streamtubes each containing N computational cells as depicted in figure 1. Note that

Figure 1: Grids for Cartesian and natural coordinate systems



in this natural co-ordinate system there is no transverse advection between streamtubes but there may be transverse diffusion. For the i-th cell in the j-th streamtube the discrete equation may be written (by taking the advection term explicitly and the transverse diffusion term fully implicitly) as,

$$c_{i,j}^{n+1} = c_{i,j}^{n} + \Delta t\left[-\frac{\partial(uc)}{\partial\xi}\right]^{n} + \Delta t\left[\frac{\partial}{\partial\xi}\left(D_{\xi}\frac{\partial c}{\partial\xi}\right)+\frac{\partial}{\partial\eta}\left(D_{\eta}\frac{\partial c}{\partial\eta}\right)\right]^{n+1} \tag{3}$$

In (3) the notation [ ] is taken to mean some kind of spatial discretisation. Equation (3) may be solved by first computing c^{adv} throughout the domain from equation (4) and then solving equation (5) as shown below.

$$c_{ij}^{adv} = c_{i,j} + \Delta t\left[-\frac{\partial(uc)}{\partial\xi}\right]^{n} \tag{4}$$

$$c_{i,j}^{n+1} = c_{ij}^{adv} + \Delta t\left[\frac{\partial}{\partial\xi}\left(D_{\xi}\frac{\partial c}{\partial\xi}\right)+\frac{\partial}{\partial\eta}\left(D_{\eta}\frac{\partial c}{\partial\eta}\right)\right]^{n+1} \tag{5}$$

In these equations Δt is the time step and the superscripts n, adv and n+1 indicate the present, intermediate (advective) and future time levels respectively. The second order spatial derivatives may be replaced with finite differences introducing $\Delta\xi$ as the longitudinal space step and $\Delta\eta$ as the transverse space step (streamtube width). Note that equation (4) is a pure advection equation which is solved using the semi-Lagrangian approach described in more detail elsewhere [3,4,6]. Suffice to say that it is solved using a conservative method of characteristics with third order (cubic) interpolation [3]. A universal limiter is employed to eliminate grid scale oscillations [5,6]. Hereafter this model will be referred to as SL. Once c^{adv} is known throughout the domain equation (5) can be solved for the concentration at the future time level [3,4,6]. With central differences for the transverse diffusion terms equation (5) becomes,

$$c_{ij}^{n+1}[1+2\alpha+2\beta]-\alpha c_{i-1\,j}^{n+1}-\alpha c_{i+1\,j}^{n+1}-\beta c_{ij-1}^{n+1}-\beta c_{ij+1}^{n+1} = c_{ij}^{adv} \tag{6}$$

where

$$\alpha = \frac{D_{\xi}\Delta t}{\Delta\xi^{2}} \qquad \beta = \frac{D_{\eta}\Delta t}{\Delta\eta^{2}} \tag{7}$$

Equation (6) may be written for each control volume and the resulting matrix assembled and solved by a suitable method, e.g. a pre-conditioned conjugate gradient method. Equations (6) and (7) are presented here for completeness but note that all tests reported here were pure advection tests, i.e. the diffusion coefficients were set to zero. Results including diffusion will be reported in a more extensive manuscript elsewhere.

4. Model testing: accuracy

Specifically, this test case consisted of a uniform rectangular flow field with a width (η) of 16m and a length (ξ) of 1000m, with zero longitudinal and transverse diffusion. The streamfunction is defined exactly for this case as,

$$\psi(x, y) = Ay \tag{8}$$

where A = 0.5 m/s. Note that this is a simple case in which the NCS is the same as the Cartesian coordinate, i.e., (x,y) maps to (ξ,η) directly. The flow is a longitudinal uniform flow with velocity u = 0.5 m/s. Boundary conditions consist of zero flux at all boundaries except the outflow boundary where concentration is computed assuming an advection-dominated flow. The initial condition is taken to be Gaussian in the transverse direction and a step function in the streamwise direction, defined by,

$$\begin{aligned} &c(\xi,\eta,0) = 0.25\exp\left(-\frac{(\eta-8.0)^2}{\sigma_0^2}\right), 100\text{m} < \xi < 200\text{m} \\ &c(\xi,\eta,0) = 0, \qquad \xi < 100\text{m} \\ &c(\xi,\eta,0) = 0, \qquad \xi > 200\text{m} \end{aligned} \tag{9}$$

The value of σ_0 was set at 0.5 m. for the initial transverse standard deviation. The mesh had 80 streamtubes each of width $(\Delta\eta)$ equal to 0.2 m. There were 100 grid cells, each of length 10m $(\Delta\xi)$, in the streamwise direction. The duration of each simulation, T, was 1000 seconds. The exact solution is the simply the initial profile translated downstream by 500m.

The purpose of this computational experiment is to demonstrate the accuracy and efficiency of the semi-Lagrangian advection model as compared to four traditional Eulerian models. The four Eulerian methods are: a backward in time, central in space differencing model (BTCS); a backward in time, first order upwind in space differencing model (BTUW); an explicit in time, first order upwind in space differencing model (UW) and an explicit in time, Lax-Wendroff differencing model (LW). Note that only BTCS and BTUW can truly be compared against the semi-Lagrangian model at all timesteps since they are stable at Courant numbers in excess of unity whereas UW and LW are not. We have chosen to compare against these schemes because they are well-known and they were simple to program. A simple rectilinear flow field was used because the numerical error mainly stems from the numerical scheme, and not from other confounding issues such as mesh generation.

The domain-averaged root mean square error (RMSE) between the numerical and exact solutions at the end of the simulation is used as an indicator of numerical accuracy. The RMSE is defined as,

$$RMSE = \frac{1}{MN}\sqrt{\sum_{i=1}^{MN}(c_{exact} - c_{model})^2} \tag{10}$$

where $M\,N$ is the number of computational grid points in the domain. Additionally, the maximum and minimum values are noted as a measure the conservativity and positivity of the algorithm. We always undertake one run with small time step (Courant number, $u\Delta t/\Delta\xi < 1$) and one run with a large time step (Courant number, $u\Delta t/\Delta\xi > 1$).

Figure 2 shows the exact solution (A) and the numerical solutions for all models for the small time step case. The result of the present algorithm (SL) is shown as (B). When the Courant number ($Cr = u\Delta t/\Delta\xi$) is small ($Cr = 0.5$), the SL model delivers very good results; the peak is not damped and the shape is well maintained. The universal limiter is included and so this model delivers no non-physical oscillations in the simulations. For the purposes of comparison, numerical results obtained by the four Eulerian models are also shown in Figures 1 (C) – (F). All are inferior to the SL model but with only the Lax-Wendroff model introducing unphysical oscillations. We highlight that the results of the non-SL models are not novel but are placed here to give some frame of reference as to the quality of the SL results. When the Courant number is small ($Cr = 0.5$), BTUW and BTCS deliver stable results but with significant numerical diffusion. The peak value is damped and the profile spreads out. Model UW performs about the same as BTUW but with substantially less work involved. Table 1 gives some statistics showing the

performance of SL, LW and UW which confirm these findings. The two-dimensional implicit Eulerian models (for BTUW and BTCS) yield a system of finite difference equations, which must be solved simultaneously, even for pure advection. This means that for a parallel implementation communication in unavoidable unlike the SL mode. Although these implicit methods are stable when the Courant number is larger than unity, they usually yield poor results. Acceptable results are found only for very small values of the Courant number (< 1), for which explicit methods are usually more efficient.

Figure 3 shows results for the large time step case (Courant number = 12.5). The SL model (B) is practically indistinguishable from the exact solution, i.e. its accuracy has improved. This feature of SL type models, that their accuracy improves with the time step increase is perhaps counterintuitive but well-known in the SL model community [8]. In contrast the implicit Eulerian models (BTCS and BTUW) provide solutions which bare almost no resemblance to the exact solution.

Figure 2: Numerical result of models at small time step (Courant number of 0.5). A – Exact solution; B – SL model; C – UW model; D – LW model; E – BTCS model; F – BTUW model. Eulerian approaches yield highly inaccurate results however the semi-Lagrangian approach gives close approximation of the exact solution.

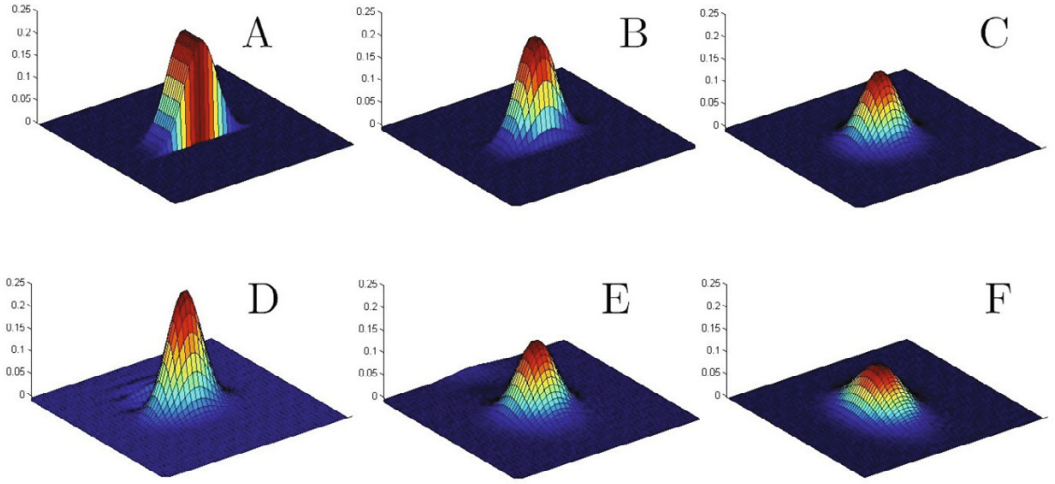


Figure 3: Numerical result of models at large time step (Courant number of 12.5). The solution obtained by two typical Eulerian approaches (BTCS and BTUW) are shown in C and D for comparison. Eulerian approaches yield highly inaccurate results for large timesteps (i.e. Courant numbers in excess of unity). (A) Exact solution; (B) SL model; (C) BTCS model; (D) BTUW model;

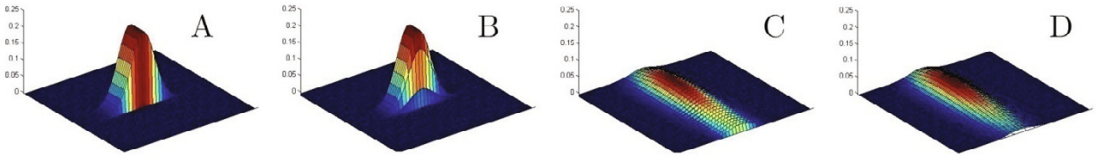


Table 2 shows the statistics for the model agreements at large time step. They reflect the poor visual agreement seen in figure 3. The RMSE of the SL model is always much smaller than those by other models. The peak value is well maintained and no non-physical value exists in the solution. The total mass is also well conserved in simulations; note that BTCS and BTUW theoretically conserve mass but because of the high numerical diffusion some mass has seeped out of the domain.

Table 1: Quantitative measurements of numerical results and exact solution (Cr=0.5)

	RMSE (10^{-4})	Cmax	Cmin	Total Mass (%)	Steps
SL	0.4650	0.2334	0.000	100	100
UW	1.1374	0.1520	0.000	100	100
LW	0.8123	0.2620	-0.045	100	100
Exact	0.0000	0.2334	0.000	100	N/A

The SL model performs so well at the large time step because for a given simulation time, less interpolations are required. So for the large time step and a given simulation time only 4 interpolations are required (table 2). In contrast when it is applied using a lower time step 100 interpolations are required (table 1). Clearly for SL models increasing the time step greatly reduces the advection term error. When other terms are present the logic is more complicated. In many real world problems, negative concentrations are unacceptable. Not only are they nonsense they can also invalidate the source and sink models. As mentioned before, the universal limiter is adopted in the SL model to achieve both monotonicity and shape preservation.

5. Model testing: parallel performance

Computational testing for speed was carried out on a cluster facility at Kean University known as Puma. Puma is a 130 node, 1040 core Dell cluster running Red Hat Enterprise Linux and Rocks+ 4.3. Each node has 2 quad core 2.6 GHz Xeon processors, 16GB RAM, and 350GB local storage, and are connected with Gigabit ethernet. For pure advection the algorithm here becomes “embarrassingly parallel" when the natural coordinate system is employed. The approach employs an intelligent domain decomposition based on the vector field of the system equations and thus automatically partitions the computational domain into algorithmically autonomous regions. In this case if there are M streamtubes and p processors then we simply assign M/p streamtubes to each processor for computation. For passive advection the streamtube computations are completely independent and can proceed to completion with no inter-processor communication. For testing the model we used a similar flow field as before and implemented the parallel algorithm using MPI. The spatial discretization was 1000 in the stream wise (length) direction and was varied in the cross-stream (width) direction (cross-stream discretizations varied from 6400 up to 819200). When the cross-stream discretizations are multiplied by the stream wise we arrive at a total problem size in terms of computational points in the mesh. These are shown in the body of Table 3. If we move down this table diagonally from right to left and top to bottom we see problems of the same size but being executed on an increasing cluster of processors. If we move down the table directly we see problem size increasing in proportion to cluster size. Wall-clock times are shown in table 4 for the sixty-four model runs undertaken.

Table 2: Quantitative measurements of numerical results and exact solution (Cr=12.5)

	RMSE (10^{-4})	Cmax	Cmin	Total Mass (%)	Steps
SL	0.3060	0.2334	0.000	100	4
BTCS	1.9889	0.0400	0.000	89	4
BTUW	1.9941	0.0380	0.000	89	4
Exact	0.0000	0.2334	0.000	100	N/A

Table 3: Problem size (number of computational nodes) computed as the product of the number of streamtubes per processor time the number of processors utilized time 1000.nodes per streamtube longitudinally.

		Number of streamtubes per processor							
		50	***100***	***200***	***400***	***800***	***1600***	***3200***	***6400***
	1	50000	100000	200000	400000	800000	1600000	3200000	6400000
	2	100000	200000	400000	800000	1600000	3200000	6400000	12800000
Number	**4**	200000	400000	800000	1600000	3200000	6400000	12800000	25600000
Of	**8**	400000	800000	1600000	3200000	6400000	12800000	25600000	51200000
processors	**16**	800000	1600000	3200000	6400000	12800000	25600000	51200000	102400000
	32	1600000	3200000	6400000	12800000	25600000	51200000	102400000	204800000
	64	3200000	6400000	12800000	25600000	51200000	102400000	204800000	409600000
	128	6400000	12800000	25600000	51200000	102400000	204800000	409600000	819200000

Table 4: Wall-clock times for the model runs (seconds)

		Number of streamtubes per processor							
		50	***100***	***200***	***400***	***800***	***1600***	***3200***	***6400***
	1	7.965	15.889	31.707	63.405	127.644	253.975	507.79	1013.673
	2	8.98	17.851	32.651	64.345	127.547	254.267	507.986	1012.623
Number	**4**	9.993	16.981	32.779	64.421	127.582	254.742	506.548	1013.569
Of	**8**	9.101	17.003	32.827	64.443	127.992	254.083	506.263	1012.835
processors	**16**	9.148	18.081	34.015	64.812	128.093	255.02	509.124	1013.951
	32	10.175	18.088	34.039	65.796	128.308	256.152	509.917	1013.245
	64	10.261	18.152	33.043	65.886	129.522	256.548	510.59	1016.713
	128	11.062	18.349	34.269	66.437	129.703	256.298	512.213	1018.755

The results in tables 3 and 4 show that the model is performing in a scalable fashion. These results have also been plotted in figures 4 and 5. Figure 4 shows that for a given problem size wall clock time reduces in proportion to the number of processors utilized. Perfect speed-up would be represented by any line parallel to the lines which have been plotted for reference. We can see that for each of the problem size plotted this occurs. Figure 5 is used to demonstrate the algorithm obeying Gustafson's Law with an assumed serial portion of the algorithm at close to 0% , i.e. as problem size increases the time for execution remains the same as long as the number of processors grows in proportion to the problem size.

Conclusions and future work

In this paper we have demonstrated some accuracy characteristics of a semi-Lagrangian transport model (pure advection only). We have demonstrated its efficiency when implemented on a single CPU by virtue of the fact that it permits the use of very large time steps while maintaining accuracy. The solution of a classic pure advection transport problem is shown to be conservative, monotonic and highly accurate at large time steps. We have next proposed an approach to parallelization. The approach employs an intelligent domain decomposition based on the vector field of the system equations and thus automatically partitions the computational domain into algorithmically autonomous regions. We demonstrate that the algorithm is highly efficient for high performance computer architectures and thus offers a route towards massively parallel application.

Acknowledgments

The authors would like to acknowledge Professor David Joiner and Kean University for making the PUMA cluster available to us for model testing and model runs. Professor Joiner also has been helpful in explaining to us

the details of the cluster. The first author would also like to thank the Richard Stockton College of New Jersey for supporting this work in the form of a R&PD grant (AY '09-'10).

References

1. K.Beven.2007. Towards integrated environmental models of everywhere:uncertainty, data and modeling as a learning process. Hydrol.Earth.Syst.Sci., **11**(1), pp460-467.

2. A. Martinez, L. Bergamaschi, M. Caliari, and M. Vianello. 2009. A massively parallel exponential integrator for advection-diffusion models. J. Comput. Appl. Math. **231**(1), pp82-91.

3. J.R. Manson and S.G.Wallis.1995. An accurate numerical algorithm for advective transport, Communications in Numerical Methods in Engineering, **11**, pp1039-1045.

4. S.G.Wallis, J.R. Manson and L.Filippi. 1998. A conservative semi-Lagrangian algorithm for one-dimensional advection-diffusion Communications in Numerical Methods for Engineering, **14** (7), pp671-679.

5. B. P. Leonard. 1991,The ULTIMATE conservative difference scheme applied to unsteady one–dimensional advection. Computer Methods in Applied Mechanics and Engineering, **88**:17–74, June 1991.

6.S.G. Wallis and Manson,J.R. 1997a. Accurate numerical simulation of advection using large timesteps. International Journal of Numerical Methods in Fluids, **24**, pp127-139.

7. O. A. Cirpka, E. O. Frind, and R. Helmig. 1999. Numerical simulation of biodegradation controlled by transverse mixing. Journal of Contaminant Hydrology, **40**(2):159{182, 1999.

8. M.A.Celia, T.F.Russell, I.Herrera, R.E.Ewing. 1990. An Eulerian-Lagrangian localized adjoint method for the advection-diffusion equation. Adv.Water Resources, **13** (4), pp187-206.

9. O. A. Cirpka, E. O. Frind, R. Helmig. 1999. Streamline-oriented grid generation for transport modelling in two-dimensional domains including wells. Advances in Water Resources. **22**(7):pp697-710.

Figure 4: Wall-clock time in seconds versus number of processors used. Solid squares are for a problem size of 3200000 computational nodes; solid circles are for a problem size of 6400000 nodes and solid triangles are for a problem size of 12800000 nodes. All show a reduction in wall clock time in proportion to the number of processors used.

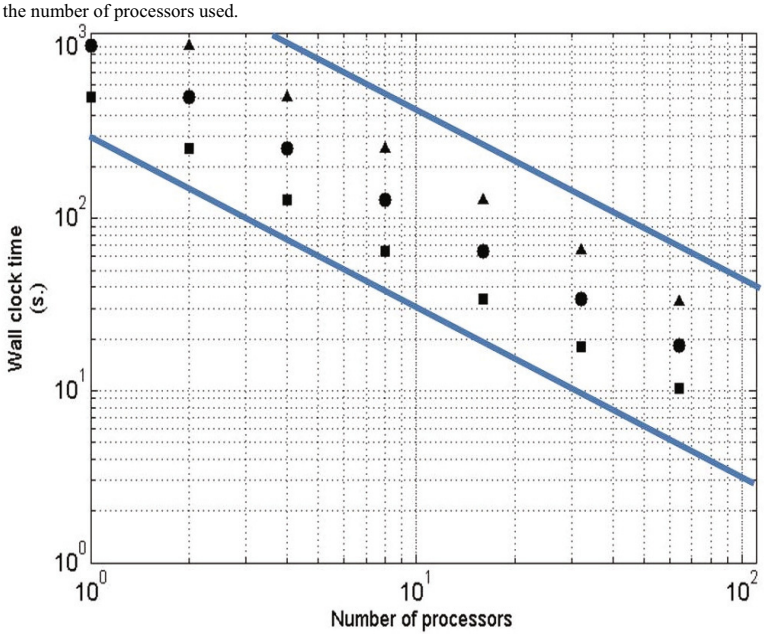


Figure 5: Wall-clock times in seconds for problems wherein the problem size scales proportionally with the number of processors. Wall-clock times remain constant in agreement with Gustafson's Law. Solid squares show problem initial size of 400000; solid circles show problem initial size of 800000; solid triangle (pointing down) show initial problem size of 1600000; solid triangles (pointing up) show initial problem size of 3200000 and solid diamonds show initial problem size of 6400000.