UNIVERSITY OF MINNESOTA

CENTER FOR
TRANSPORTATION
STUDIES

ITS INSTITUTE

PARALLEL TRAFFIC FLOW SIMULATION OF FREEWAY NETWORKS: PHASE 2

Dr. Anthony Chronopoulos
Department of Computer Science

Technical Report Documentation Page

1. Report No.	2.	3. Recipient's Accession No.
CTS 97-02		

4. Title and Subtitle	5. Report Date
Parralel Traffic Flow Simulation of Freeway Networks, Phase 2	July 1997
	6.

7. Author(s)	8. Performing Organization Report No.
Dr. Anthony Chronopoulos	

9. Performing Organization Name and Address	10. Project/Task/Work Unit No.
Department of Computer Science University of Minnesota 4-192 EE/CS Building 200 Union St. SE Minneapolis, MN 55455	11. Contract (C) or Grant (G) No. (C) (G)

12. Sponsoring Organization Name and Address	13. Type of Report and Period Covered
Center for Transportation Studies ITS Institute Program 200 Transportation and Safety Building 511 Washington Avenue SE Minneapolis, MN 55455	Final Report 1994-1995
	14. Sponsoring Agency Code

15. Supplementary Notes

16. Abstract (Limit: 200 words)

Explicit and implicit numerical methods for solving simple macroscopic traffic flow continuum models have been studied and efficiently implemented in traffic simulation codes in the past. We have already studied and implemented explicit methods for solving the high-order flow conservation traffic model. Implicit methods allow much larger time step size than explicit methods, for the same accuracy. However, at each time step a nonlinear system must be solved. We use the Newton method coupled with a linear iterative method (Orthomin). We accelerate the convergence of Orthomin with parallel incomplete LU factorization preconditionings. We implemented this implicit method on a 16 processor nCUBE2 parallel computer and obtained significant execution time speedup.

17. Document Analysis/Descriptors	18. Availability Statement
Traffic Flow Simulation Numerical Methods	No restrictions. Document available from: National Technical Information Services, Springfield, Virginia 22161

19. Security Class (this report)	20. Security Class (this page)	21. No. of Pages	22. Price
Unclassified	Unclassified		

Parallel Traffic Flow Simulation of Freeway Networks

Phase 2 Report

Prepared by
Dr. Anthony Chronopoulos

Department of Computer Science
University of Minnesota

July 1997

Published by

Center for Transportation Studies
University of Minnesota
200 Transportation and Safety Building
511 Washington Avenue S.E.
Minneapolis, MN 55455-0375

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Parallel Traffic Flow Simulation of Freeway Networks (Phase 2 Report)

A. Chronopoulos *

Abstract

Explicit and implicit numerical methods for solving simple macroscopic traffic flow continuum models have been studied and efficiently implemented in traffic simulation codes in the past . We have already studied and implemented explicit methods for solving the high-order flow conservation traffic model. In this project, we studied and implemented an implicit numerical method for solving the high-order flow conservation traffic model. Implicit methods allow much larger time step size than explicit methods, for the same accuracy. However, at each time step a nonlinear system must be solved. We use the Newton method coupled with a linear iterative method (**Orthomin**). We accelerate the convergence of **Orthomin** with parallel incomplete LU factorization preconditionings. We implemented this implicit method on a 16 processor nCUBE2 parallel computer and obtained significant execution time speedup.

*Department of Computer Science, University of Minnesota. The Project was funded through the Center of Transportation Studies. The Computer Science graduate students Choong-Yul Rhee and Gang Wang worked on this project. Dr. Ping Yi of MnDot/GuideStar provided advice for this Project. Mr. David Berg of MnDot provided help with the freeway traffic data.

1 Introduction

Macroscopic continuum traffic models flow based on traffic density, flow and velocity have been proposed and analyzed in the past. Examples include Lighthill and Whitham's (1955) flow conservation model, Payne's momentum model conservation model and Michalopoulos's momentum model [11], [18], [16]. These models involve partial differential equations (PDEs) defined on appropriate domains with suitable boundary conditions which describe various traffic phenomena and road geometries.

The improvement of computational efficiency in the continuum traffic models has been the focal point in the development of traffic simulation programs. It is understood that the computer execution time to solve traffic flow problems depends not only on the size of the freeway and the complexity of roadway geometries, but also on the model equations and numerical schemes used in their discretization.

Explicit numerical methods (for example Lax, Upwind) have been used by Michalopoulos and Lin and Leo and Pretty to compute the solution of traffic flow continuum models [15], [10]. In these explicit schemes the space and time mesh sizes are restricted both by accuracy and numerical stability requirements. In order to reduce the computer execution time and maintain good accuracy, the total number of computations must be reduced. This can be achieved by using larger values of time and space mesh sizes. Implicit numerical methods provide the same accuracy as explicit methods and allow changes in the mesh sizes while maintaining numerical stability [8] [1].

In this work we use an implicit numerical method (Backward Euler) to solve more efficiently the momentum conservation model on a nCUBE parallel computer. Implicit methods allow much larger time step size than explicit methods, for the same accuracy. However, at each time step a nonlinear system must be solved. We use the Newton method coupled with a linear iterative method (Orthomin). We accelerate the convergence of Orthomin with parallel incomplete LU factorization preconditionings. We wrote a code (in C) simulating a freeway multiple entry/exit traffic flow. Tests with real data collected from the I-35W and I-494 freeways in Minneapolis were conducted. Using these data we tested (for accuracy and efficiency) our code on a Sun Sparc1 workstation computer. We have also implemented efficiently the new method on the (16 processor) nCUBE2 parallel computer located at the Department of Computer Science. Each processor of the nCUBE2 is

as powerful as a SUN 3/50 workstation. On the nCUBE2, the parallel new method, on the 16 processors, runs 4 times faster than on the one processor.

The outline of this article is as follows. In section 2, we review the momentum conservation continuum traffic model. In section 3, we review the Euler implicit method. In section 4, we describe the Orthomin(k) iterative method. In section 5, we describe the various theoretical and empirical curves relating the traffic flow and density. In section 6, we describe the multiple entry/exit freeway models. In section 7, we describe a parallel implementation of the Euler method. In section 8, we present the numerical results. Section 9 contains concluding remarks.

2 A Simple and High-Order Continuum Model of Traffic Flow

The following conservation equation has been proposed by Lighthill and Whitham (1995) [11] as a **simple continuum traffic model (LWM)**:

$$\frac{\partial k}{\partial t} + \frac{\partial q}{\partial x} = g(x,t), \tag{1}$$

where $k(x,t)$ and $q(x,t)$ are the traffic density and volume respectively at the space-time point (x,t). The generation term $g(x,t)$ represents the number of cars entering or leaving the traffic flow in a freeway with entries/exits. The **traffic flow volume , density and speed** are related by the equation:

$$q = ku, \tag{2}$$

where the equilibrium speed $u(x,t) = u(k)$ must be provided by a theoretical or empirical u-k model. u-k model The theoretical u-k model, equation of state, can take the general form.

$$u_e = u_f[1 - (k/k_{jam})^{\alpha}]^{\beta}, \tag{3}$$

where u_f is the free flow speed and k_{jam} the jam density [7]. For instance, for $\alpha = 1$ and $\beta = 1$, one obtains the Greenshields'(1934) equation of state.

More information on this and other forms of the u-k relationships can be found elsewhere (Mcshane and Roess 1990) (see [13]).

Since the simple continuum model does not consider acceleration and inertia effects, it does not describe accurately non-equilibrium traffic flow dynamics. Furthermore, the speed-density relationship employed in the simple continuum model is difficult to obtain in practice. Payne introduced a high-order continuum traffic model that includes the momentum equation in addition to the continuity equation of the simple continuum model [18], [19]. The high-order continuum formulation takes into account acceleration and inertia effects by replacing Equation (3) with the momentum equation. The momentum equation in the **improved high-order continuum model (IHOCM)** in [16] is obtained as follows.

$$\frac{du}{dt} = \frac{1}{T}[u_f(x) - u] - G\frac{\partial u}{\partial t} - \nu k^{\beta}\frac{\partial k}{\partial x} \tag{4}$$

where $\frac{du}{dt}$ is the acceleration of an observer moving with the traffic stream and is related to the acceleration $\frac{\partial u}{\partial t}$ of the traffic stream as seen by an observer at a fixed point of the road, i.e.

$$\frac{du}{dt} = \frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} \tag{5}$$

The first term on the right hand side of Eq. 4, $\frac{1}{T}[u_f(x) - u]$, represents the relaxation term, the tendency of traffic flow to adjust speeds due to changes in $u_f(x)$ along the roadway, where relaxation time T is assumed to vary with density k according to

$$T = t_0(1 + \frac{rk}{k_{jam} - rk}) \tag{6}$$

where $t_0 > 0$ and $0 < r < 1$ are constants. The second term, $G\frac{\partial u}{\partial t}$, addresses the traffic friction at freeway ramp junctions due to ramp flows. G is the friction parameter. It is a function of both roadway conditions and the ramp volume entering or leaving the freeway and is derived experimentally as $G = \mu k^{\epsilon} g$, where μ is a geometry parameter depending on the type of road geometry, ϵ is a dimensionless constant, and g is the generation term. The third term, $\nu k^{\beta}\frac{\partial k}{\partial x}$, represents the anticipation term which is the effect

of drivers reacting to downstream traffic conditions. In this term ν is the anticipation parameter. As implied in this example, if downstream density is higher due to congestion, speed has to be decreased accordingly. Conversely, if downstream density is lower, speed can be increased. From equations (4) - (6) one derives a momentum model for the traffic flow described by the following system of PDEs.

$$\frac{\partial \vec{\mathbf{U}}}{\partial \mathbf{t}} + \frac{\partial \vec{\mathbf{E}}}{\partial \mathbf{x}} = \vec{\mathbf{Z}} \tag{7}$$

where $\vec{U}$, $\vec{E}$, and $\vec{Z}$ are the following vectors:

$$\vec{U} = \begin{pmatrix} k \\ q \end{pmatrix}$$

$$\vec{E} = \begin{pmatrix} ku \\ u^2 k + \frac{\nu}{\beta+2} k^{\beta+2} \end{pmatrix}$$

$$\vec{Z} = \begin{pmatrix} g \\ \frac{k}{T}[u_f(x) - u] - Gk\frac{\partial u}{\partial t} + gu \end{pmatrix}$$

We note that the momentum conservation model does not require a $q-k$ curve as in the case of the simple continuum model. However speed data are not available from the the real traffic data then a $q-k$ curve is used to generate the speed data. We chose $G = 0$ in our implementation.

3 Numerical Euler Implicit Method

We consider one high-order implicit method (Euler implicit) which is used in computational fluid dynamics [8]. For each traffic model the road section (the space dimension) is discretized using uniform mesh for all numerical methods; only the time stepsizes differ between methods. We use the following notation:

Δt = time stepsize.

Δx = space stepsize.

k_j^n = density (vehicles/mile/lane) at space node $j\Delta x$ and at time $n\Delta t$.

q_j^n = flow (vehicles/hour/lane) at space node $j\Delta x$ and at time $n\Delta t$.

u_j^n = speed (mile/hour) at space node $j\Delta x$ and at time $n\Delta t$.

The high-order Euler implicit method applied to the nonlinear PDE (1) generates a nonlinear recursion involving all space nodes at each time step. To solve numerically this recursion Beam and Warming have suggested using one Newton linearization steps [8]. Each Newton step constructs a linear system of the banded matrix with unknowns $\Delta\vec{U}_j = \vec{U}_j^{n+1} - \vec{U}_j^n$.

$$-\frac{\Delta t}{2\Delta x}\left(\frac{\partial \vec{E}}{\partial \vec{U}}\right)_{j-1}^{n} \Delta\vec{U}_{j-1} + [1 - \Delta t\left(\frac{\partial \vec{Z}}{\partial \vec{U}}\right)_{j}^{n}]\Delta\vec{U}_j + \frac{\Delta t}{2\Delta x}\left(\frac{\partial \vec{E}}{\partial \vec{U}}\right)_{j+1}^{n} \Delta\vec{U}_{j+1} =$$
$$-\frac{\Delta t}{2\Delta x}\left(\vec{E}_{j+1}^n - \vec{E}_{j-1}^n\right) + \Delta t\vec{Z}_j^n + \Delta t\left(\frac{\partial \vec{Z}}{\partial \vec{U}}\right)_j^n \vec{U}_j^n$$

where

$$\left(\frac{\partial \vec{E}}{\partial \vec{U}}\right)_j^n = \begin{pmatrix} 0 & 1 \\ -u^2 + \nu k^{\beta+1} & 2u \end{pmatrix}_j^n$$

$$\left(\frac{\partial \vec{Z}}{\partial \vec{U}}\right)_j^n = \begin{pmatrix} 0 & 0 \\ \frac{u_f}{t_0 k_{jam}}(k_{jam} - 2rk) + \frac{r}{t_0 k_{jam}} q & -\frac{k_{jam} - rk}{t_0 k_{jam}} \end{pmatrix}_j^n$$

The structure of the matrix is shown in Figure 1. It is a tridiagonal block matrix, whose blocks have size 2 and this matrix has also banded structure with bandwidth equal to 7.

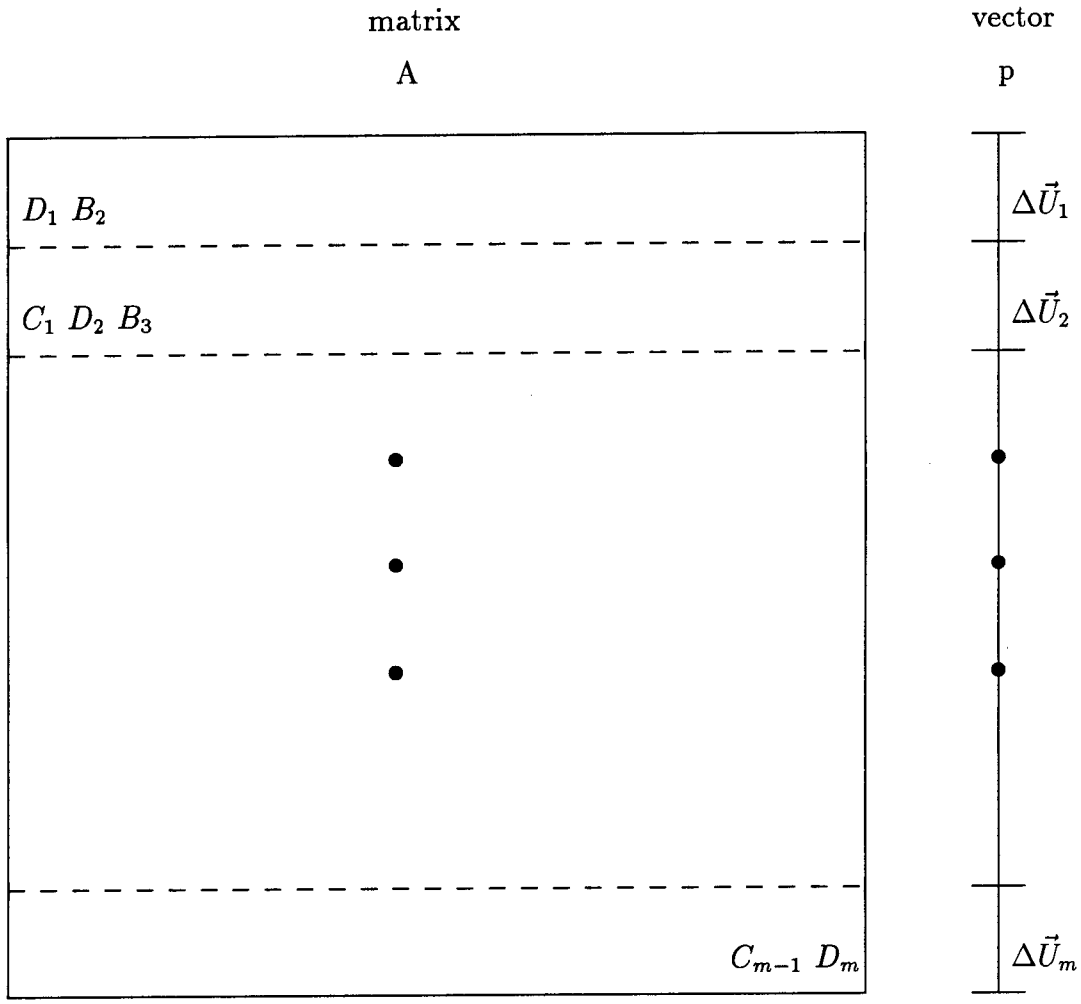


Figure 1. Structure of the matrix-vector of the linear system

where

$$B_j = \begin{pmatrix} 0 & \frac{\Delta t}{2\Delta x} \\ \frac{\Delta t}{2\Delta x}(-u^2+\nu) & \frac{\Delta t}{\Delta x}\frac{q}{k} \end{pmatrix}_j^n$$

$$C_j = \begin{pmatrix} 0 & -\frac{\Delta t}{2\Delta x} \\ -\frac{\Delta t}{2\Delta x}(-u^2+\nu) & -\frac{\Delta t}{\Delta x}\frac{q}{k} \end{pmatrix}_j^n$$

$$D_j = \begin{pmatrix} 1 & 0 \\ \frac{u_f \Delta t}{t_0 k_{jam}}(k_{jam}-2rk) - \frac{r\Delta t}{t_0 k_{jam}} q & 1+\Delta t\left(\frac{k_{jam}-rk}{t_0 k_{jam}}\right) \end{pmatrix}_j^n$$

This linear system of banded matrix is solved by the Orthomin iterative method with preconditioning. On a Sun workstation, we have used the banded LU factorization method as a preconditioning to solve above linear system by Orthomin iterative method. The solution is then advanced to the next time step simultaneously at all space nodes by computing $\vec{U}_j^{n+1} = \vec{U}_j^n + \Delta\vec{U}_j$. This method is of second order accuracy with respect to Δt and it is unconditionally stable. This method will be called **Euler-Momentum** method.

Artificial smoothing is often added to reduce oscillatory behavior in the numerical solution. This is achieved by adding a fourth order damping term d_j to each term $\vec{U}_j$

$$d_j = -\frac{\omega}{8}\left(\vec{U}_{j-2} - 4\vec{U}_{j-1} + 6\vec{U}_j - 4\vec{U}_{j+1} + \vec{U}_{j+2}\right)$$

We have tested several damping coefficients with ω ranging from $\omega = 0$ (no damping) to $\omega = 1$. The choice $\omega = 1$ gave the best results.

4 Orthomin iterative method

We next describe Orthomin iterative method to solve $Ax = rhs$, where A is nonsymmetric matrix of order N. The Orthomin applies to nonsymmetric

linear systems with the symmetric part of A being positive definite. Let k_o be a positive integer. We describe the orthomin iterative method with preconditioning as follows. In this algorithm, $j_i = \max(0, i-k_o+1)$, P_r is the right preconditioner ($P_rA \approx I$) which is obtained by the LU decomposition of the matrix A. For more details, see [12].

Algorithm 1. Orthomin(k_o)

1. Choose $x_0 = 0$.
2. Compute $r_0 = rhs - Ax_0$.
3. $p_0 = r_0$.
   For i = 0 step 1 **Until** convergence **Do**
4. $a_i = \frac{(r_i, Ap_i}{(Ap_i, Ap_i)}$
5. $x_{i+1} = x_i + a_i p_i$
6. $r_{i+1} = r_i - a_i Ap_i$
7. Compute $AP_r r_{i+1}$.
8. $b_j^i = \frac{(AP_r r_{i+1}, Ap_j)}{(Ap_j, Ap_j)}$
9. $p_{i+1} = P_r r_{i+1} + \sum_{j=j_i}^{i} b_j^i p_j$
10. $Ap_{i+1} = AP_r r_{i+1} + \sum_{j=j_i}^{i} b_j^i Ap_j$
    Endfor

5 Volume-Density (q-k) Model Curves

A $q-k$ model curve is an indispensable part of the LWM. This relation can be used to express the volume as a function of the flow density i.e. $q = q(k)$. This function is a nonlinear function which must satisfy some general requirements. The equations that define the $q-k$ curve are used in the programs to convert from density to volume and from volume to density. The Momentum Traffic Flow Model *does not require* a $q-k$ curve. However,

if the speed data are not available from the (upstream/downstream) traffic data then a $q-k$ curve is used to compute the traffic speed.

These general requirements on the $q-k$ curve can be derived from the following observations on traffic flow modeling [13].

- For uncongested flow an increase in density corresponds to an increase in volume, up to a critical density k_c, where the flow becomes congested.
- Maximum volume occurs at the critical density: $q_{max} = q(k_c)$.
- For congested flow an increase in density corresponds to a decrease in volume, up to the jam density k_{jam}, where traffic flow halts.

A q-k model curve must also be adapted to characteristics of the freeway section which it represents. Theoretical q-k model curves can not be adapted to the special roadway characteristics and so such a model function must be constructed from empirical data. Greenshields' q-k curve is derived from equations (2) and (3) and appropriate choices for the free flow speed u_f and jam density $k_{jam} = k_0$. In our applications we chose u_f =(60 miles/hour) and k_0 =(180 vehicles/mile). The Greenshields' curve has the basic features described above but can not be tuned to local characteristics of a freeway section. However, we used Greenshields' q-k curve in the initial development of the programs and as a baseline for comparisons.

5.1 An empirical q-k Model Curve

Field data for constructing the q-k model curve were collected in I-35 W in Minneapolis. With these discrete data a piecewise linear q-k curve was derived [14]. Such a curve must have parameter ranges reflecting the road characteristics of the freeway section it represents [14]. With our discrete data the experimental q-k curve must have following parameter ranges:

- The critical density k_c should be about 70 to 75 vehicles/mile/lane.
- The maximum volume q_{max} should be less than 2500 to 2700 vehicles/hour/lane.
- The slope of the curve at $k = 0$, which represents the free-flow speed u_f, should be approximately 65 to 75 miles/hour.

We have used several curve fitting methods to construct continuous q-k curves from the set of (k, q) discrete data points available. Our objective was to find a general method that produces a curve which is based on the discrete data, has the basic features of a q-k curve, has the parameter ranges (described above), and also works well in the numerical methods for solving (1). We used **least squares** to approximate q-k curves from field data.

Several **least squares** approximations were tried. In this method the data points (k_i, q_i) are used to construct a rectangular matrix with row i composed of powers of k_i and a right-hand-side vector containing the q_i. Then the matrix is reduced using the **Singular Value Decomposition** method (**SVD**) available in theLINPACK package or the Matlab package [5]. The reduced matrix is then used to find the coefficients of the curve that minimizes the total squared error between the data points and the curve. This method will produce curves of any degree up to the number of data points. Quadratic, cubic and quartic least-squares polynomial curves were found using the Matlab's (SVD). The quartic curve

$$q = -1.7156 \times 10^{-5}k^4 + 7.1802 \times 10^{-3}k^3 - 1.2514k^2 + 94.8463k - 69.1588$$

appeared to be the lowest-degree least-squares approximation to the discrete data that satisfies the q-k curve criteria. This quartic polynomial must be evaluated at each node for each time step so it is important to use a polynomial of the least degree.

5.2 Density estimation using occupancy

Another method of estimating traffic density and speed is based on **occupancy** data [6].

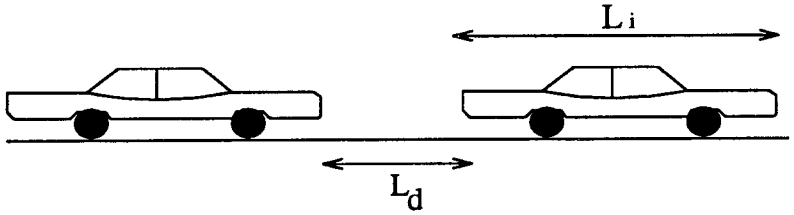


Figure 2.Density/occupancy calculation from vehicle relative positions

Lane occupancy is defined as the time that loop detector is on divided by the measured time and multiplied by 100. The value of lane occupancy is avai lable from the loop detector installed under the freeway pavement. If we assume that the speed of the vehicle is constant during measurement time and each vehicle's length is the same, we can derive the relationship between density and occupancy as follows.

$$k = 52.8 \times \frac{\phi}{L_e},$$

where effective length $L_e = \frac{\sum_{i=1}^{N}(L_i+L_d)}{N}$, ϕ=lane occupancy, L_i=vehicle length, L_d=intervehicle distance.

6 Freeway Model with Multiple Entries/Exits

6.1 Freeway testsites

We considered two multiple entry/exit freeway sections in the Minneapolis, Minnesota freeway network.

- A section of I-35W Northbound for the workstation computer implementation
- A section of I-494 Eastbound for the parallel computer implementation

The I-35W roadway geometry is presented in Figure 20. The upstream and downstream boundaries were set at the location of the 86th street and the 63th street. It has two weaving sections at the first two entry/exit zones. There are a total of 5 entry/exit ramps in this 2.65 miles road section. The road geometry of I-494 is similar. The I-494 Eastbound section extends from the Carlson Pwy to Portland Avenue. It is 18 miles long and it has 21 entry and 18 exit ramps. The I-35W section is short but it has interestingly complicated topology and we used it to test our discrete model accuracy on a workstation computer. We then used the (longer) I-494 section for the simulation on a parallel computer.

6.2 Entry/Exit Ramp Modeling

We have used two schemes to add/subtract entry/exit (ramp) traffic volumes to the mainlane traffic volume in IHOCM.

1. Point Entry/Exit Scheme: Ramp volumes are assumed to merge into (diverge from) the freeway mainlane at a single node. This treatment is necessary to simplify the modeling and reduce computation time at such mainlane nodes. (2)

2. Average Entry/Exit Scheme: Ramp volumes are assumed to merge evenly into (diverge from) the freeway mainlane at the freeway nodes in the vicinity of the entry/exit node.

3. Weaving Entry/Exit Scheme: This is used when the ramp is directly connected to another freeway and explained below.

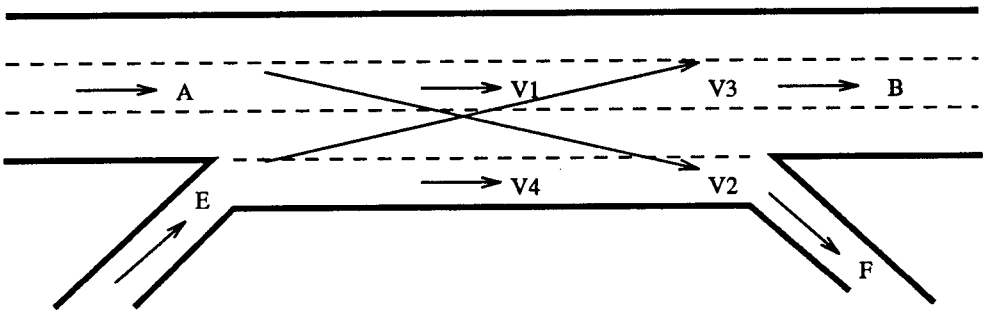


Figure 3. Weaving flows in a freeway

The weaving scheme is outlined as follows. In Figure 3, volume v_1 represents the through traffic stream flow in from link A to link B and volume v_2 represents the diverging stream from link A to link F, where $q_A = v_1 + v_2$; v_3 is the merging stream from link E to link B and volume v_4 is the through stream from link E to link F, and $q_E = v_3 + v_4$. It is obvious that $q_F = v_2 + v_4$ and $q_B = v_1 + v_3$. Because there are interchang es of v_2 and v_3, traffic friction at link B and link E in this case is greater than the case of a single

entrance ramp or exit ramp. Likewise, mergin g dynamics at an entrance ramp should be employed if $v_2 = 0$.

When L is less than 600ft, merging and diverging movements must be completed wit hin a short distance. In such a case a net value of the merging and exiting volumes is sought for volume conservation. However, since both q_{in}^n and q_{out}^n require lane changing at the same limited length of roadway at the same time, the sum of q_{in}^n and q_{out}^n should be included in the generation term. If $g > 0$, the short weaving section is treated as a single on-ramp, if $g < 0$, it is treated as a single off-ramp. The generation term in IHOCM becomes

$$g = q_{net}/\Delta x = (q_{in}^n - q_{out}^n)/\Delta x,$$

where q_{in}^n and q_{out}^n are the merging and exiting volumes.

7 Parallel implementation on nCUBE

7.1 Parallel Euler Implicit

We have implemented the Euler-Momentum method on the nCUBE2 parallel computer at the Computer Science Department of the University of Minnesota. The nCUBE2 has 16 processors(**CPUs**) connected in a hypercube network and to a host (Sun 3/50) computer for interaction with the user. The number of processors to be active is chosen by the user, but must be a power of 2. The host computer allocates a number of processors arranged in a hypercube of dimension *ndim*. Each processor is directly connected to *ndim* other processors. In table 1 we show a summary of inter processor communication times for neighbor processors and the basic floating point operation times [9]. We see that communication even between neighbor processors is several times slower than floating point operations. Programs run most efficiently when inter processor communication is minimized and when all communication occurs between neighbor processors.

Operation	Time	Comm/Comp
8 Byte transfer	111 μ sec	-
8 Byte Add	1.23 μsec	90 times
8 Byte Multiply	1.28 μsec	86 times

Table 1. Computation and Communication times on the nCUBE2

In section 3, we derived the linear system of banded matrix by the high-order Euler implicit method applied to the IHOCM. In the parallel implementation of Orthomin iterative method to solve the linear system, we have three main components to be parallelized.

Assume we have p processors. For simplicity, suppose that $n = 2pq + s$, where $s = 2(p-1)$. We distribute the matrix and **rhs** data to p processors as follows. The first processor has the data of matrix from the first row up to (2q+2)th row, also it has the data of **rhs** vector from the first componeent up to the (2q+2)th component, the second processor has the data of matrix from the (2q+3)th row up to (4q+4)th row and the data of **rhs** vector from the (2q+3)th componeent up to the (4q+4)th component, and so on. The last processor p has the matrix data from row n-2q up to row n and the **rhs** vector data from component n-2q up to n.

A **linear combination** (steps 5, 6, 9 and 10 in Algorithm 1) is mapped to each processor to do their corresponding part. Since elements of the same indices in vector arrays are on the same processor, this whole operation are done locally and no interprocessor communication are required.

A **dot product** (steps 4 and 8 in Algorithm 1) to compute the norms of vectors is also mapped to each processor to do their corresponding sum. Then a broadcast of their respective sum is done on each processor, then they are added together. So finally each processor has the dot product.

A **matrix vector multiplication** (step 7 in Algorithm 1) is also mapped to each processor to do their corresponding computation. But for the boundary data between processors, each processor needs to get the three vector data from its neighbor processors so that it can compute to get the result.

Figure 4 shows an example of matrix A and right hand side vector x for $maxj = 12$(space nodes), or $n = 22$(dimension of matrix and right hand side vector), where

$$a = 1 + \Delta t \frac{k_{jam} - rk}{t_0 kjam},$$

$$b = -\Delta t[\frac{u_f}{t_0 k_{jam}}(k_{jam} - 2rk) + \frac{r}{t_0 k_{jam}} q],$$

$$c = \frac{\Delta t}{2\Delta x}[-u^2 + \nu],$$

$$d = \frac{\Delta t}{\Delta x}\frac{q}{k},$$

$$e = -\frac{\Delta t}{\Delta x}\frac{q}{k},$$

$$f = -\frac{\Delta t}{2\Delta x}[-u^2 + \nu],$$

$$g = \frac{\Delta t}{2\Delta x}.$$

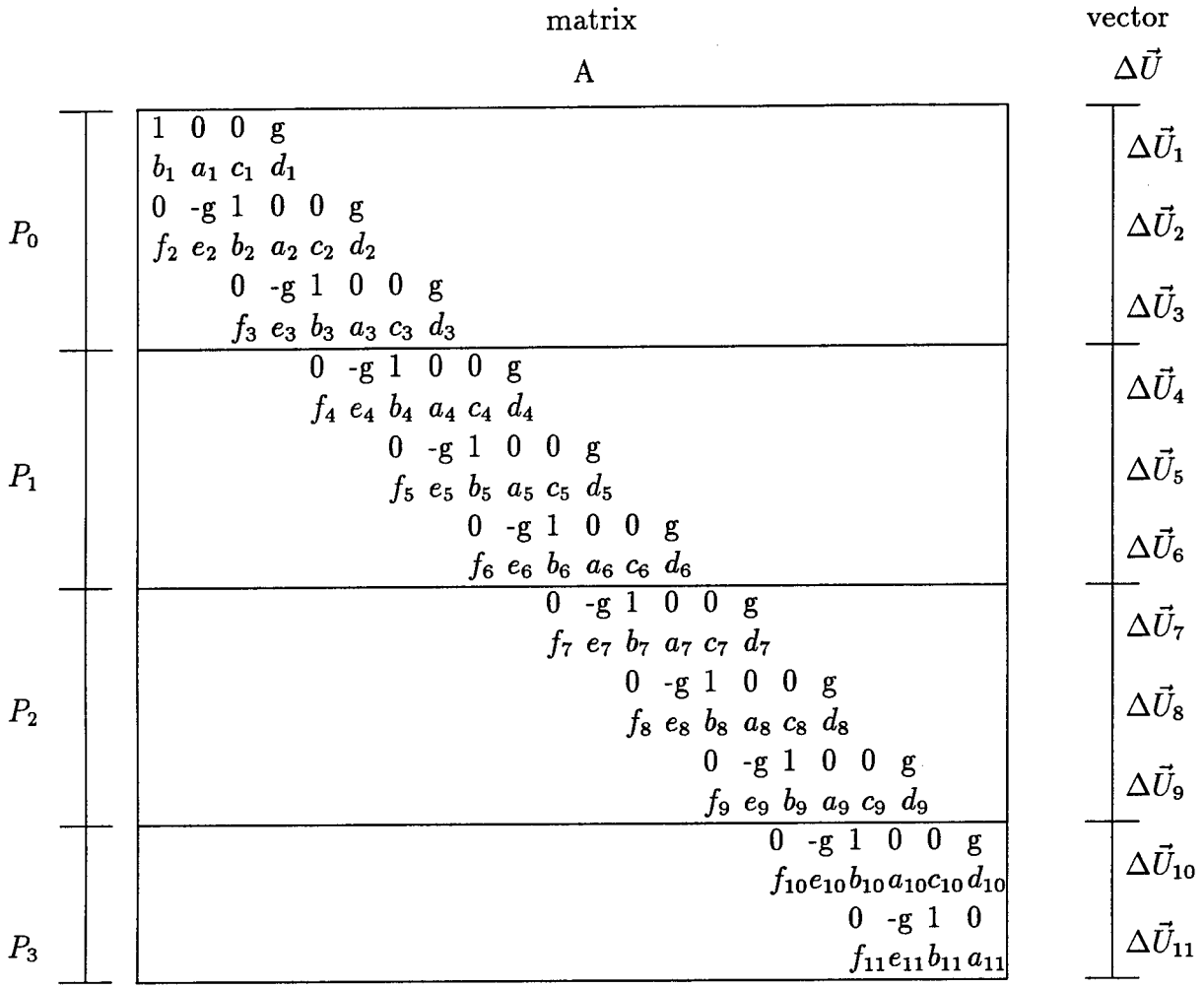


Figure 4. original structure of matrix A and vector $\Delta\vec{U}$.

7.2 ILU Preconditioning

In determining a *preconditioner* (matrix) M, we look for a matrix that is a close approximation to the inverse of A, so that

$$M \approx A^{-1} \tag{8}$$

or AM has clustered eigenvalues. The preconditioner M must be easily invertible, so that the system $M^{-1}x = b$ is easy to solve.

The basis for the ILU factorization method is that the matrix A is decomposed into upper and lower triangular matrices U and L such that $A = LU + \Delta$, where $M^{-1} = LU$ and Δ is an error matrix. Also, we assume that if $A_{i_1,i_2} = 0$, then both U_{i_1,i_2} and $L_{i_1,i_2} = 0$. In other words, L and U have the same sparsity patterns as A. This is the ILU(0) method.

Although ILU preconditioning can improve the convergence rate of the iterative solvers considerably, the preconditioner itself may have very slow execution rates if not implemented properly on a vector-parallel computer. This is due to the following fact. Let L and U be the incomplete LU factors of A. Then solution of the two triangular systems $Ly = b$ and $Ux = y$ requires back-solving, which is a serial operation.

A method for parallelizing the ILU preconditioner was introduced by Radicati di Brozolo and Robert in 1988 [4]. It was proposed to partition the preconditioned matrix into a number (m) of overlapping *submatrix regions.* Each region consists (of a continguous index sequence) of submatrix blocks of the type $[\ C_{j-1}, D_j, B_j\ ]$. The loss of connection between the regions is partially compensated for by introducing smaller overlapping *region segments.* Each submatrix region is then executed in parallel on m processors. After the back solution step is carried out (independently in each submatrix region) the overlapped values between the separate regions are set equal to the average of that determined in each region. It is found ([4]) that this overlapping strategy gives better performance than the nonoverlapping one. Here, we use an overlapping of a single submatrix block of the type $[\ C_{j-1}, D_j, B_j\ ]$, between two successive parallel regions. The parallel implementation of this technique, on m processors (**CPUs**), is illustrated in Figure 5:

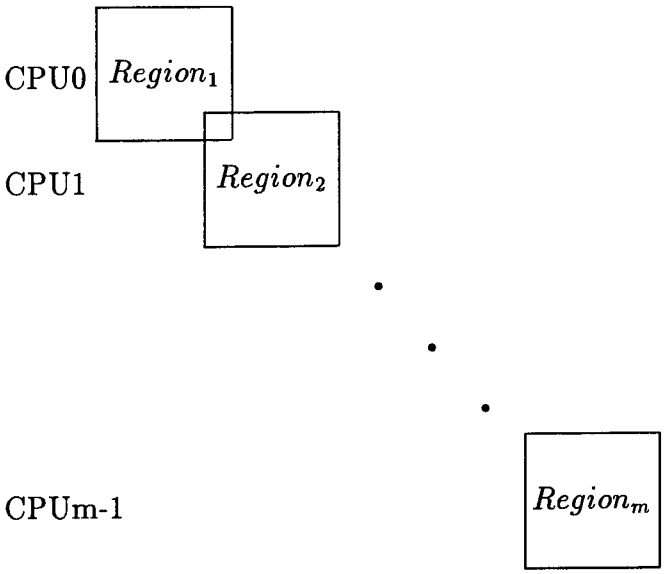


Figure 5. Parallel execution of ILU

If m=1, this preconditioner is exactly the same as the vectorized ILU preconditioned method. If $m > 1$, this type of preconditioning is slightly less effective at reducing the condition number because of the loss of connection between the submatrix regions. In general, the effectiveness of the preconditioner is reduced as m becomes larger, but the performance on parallel processors improves.

7.3 Domain decomposition preconditioning

We first describe LU factorization with domain decomposition reordering of the unknowns. The domain decomposition method [17] partitions the unknowns $\Delta\vec{U}_i$ as follows. We call $\bar{D}_1$ the unknowns $\Delta\vec{U}_1$, ..., $\Delta\vec{U}_q$, we define $\bar{x}_1 = [\Delta\vec{U}_1, ..., \Delta\vec{U}_q]^T$, $\bar{D}_2$ the unknowns $\Delta\vec{U}_{q+2}$, ..., $\Delta\vec{U}_{2q+1}$, $\bar{x}_2 = [\Delta\vec{U}_{q+2}, ..., \Delta\vec{U}_{2q+1}]^T$, and so on. For simplicity, we assume that $n/2 = pq + (p-1)$. Then there are p sets $\bar{D}_1$, ..., $\bar{D}_p$ each containing q unknowns and p-1 unknowns $\Delta\vec{U}_{q+1}$, $\Delta\vec{U}_{2q+2}$, ..., which are between the sets $\bar{D}_i$. These p-1 unknowns constitute the separator set S. We then order the unknowns $\Delta\vec{U}_j$

by numbering those unknowns in the separator set last.

Suppose we want to parallelize it on 4 processors of nCUBE2. First we distribute the matrix and **rhs** vector data to 4 processors as shown in figure 4, then we reorder them as described above. Figure 6 shows the matrix and vector after reordering and how they are mapped to 4 processors. In Figure 4 and Figure 6, only the nonzero entries are shown. The matrix and right hand side are partitioned into blocks and assigned to CPUs (P_0, P_1, P_2, P_3). The separator block is assigned to all CPUs. In the following we will describe how the new system is solved in parallel on nCUBE2.

Suppose $\bar{A}_1$, ..., $\bar{A}_p$ are of size $2q \times 2q$ and $\bar{A}_s$ is of size $s \times s$, where $s = 2(p-1)$ is the number of unknowns in the separator set $S = \cup S_i$. Then the $\bar{B}_i$ are of size $2q \times s$ and the $\bar{C}_i$ are of size $s \times 2q$. We have assumed , for simplicity, that $n = 2pq + s$.

The following are the parallel domain decomposition algorithms [17].

Algorithm (Block LU Factorization)

1. on every processor, do LU decompsition $\bar{A}_i = L_i U_i$, i = 1, ..., p.
2. on every processor, solve the system $\bar{A}_i \bar{Z}_i = \bar{B}_i$, i = 1. ..., p.
3. on every processor, form $\bar{C}_i \bar{Z}_i$, i = 1, ..., p.
4. on every processor, broadcast $\bar{C}_i \bar{Z}_i$ so that every processor has the data $\bar{C}_i \bar{Z}_i$, i = 1, ..., p.
5. on all processors, form $\hat{A} = \bar{A}_s - \sum_{i=1}^{p} \bar{C}_i \bar{Z}_i$ and compute the LU decomposition of $\hat{A}$.

Algorithm (Block LU Backwards Substitution)

1. on every processor, solve the system $\bar{A}_i \bar{z}_i = \bar{b}_i$, i = 1. ..., p.
2. on every processor, form $\bar{C}_i \bar{z}_i$, i = 1, ..., p.
3. on every processor, broadcast $\bar{C}_i \bar{z}_i$ so that every processor has the data $\bar{C}_i \bar{z}_i$, i = 1, ..., p.
4. on all processors, form $\hat{b} = \bar{b}_s - \sum_{i=1}^{p} \bar{C}_i \bar{z}_i$.
5. on all processors, solve the system $\hat{A} \bar{x}_s = \hat{b}$.
6. on every processor, form $\bar{c}_i = \bar{b}_i - \bar{B}_i \bar{x}_s$, i = 1, ..., p.
7. on every processor, solve the system $\bar{A}_i \bar{x}_i = \bar{c}_i$, i = 1, ..., p.

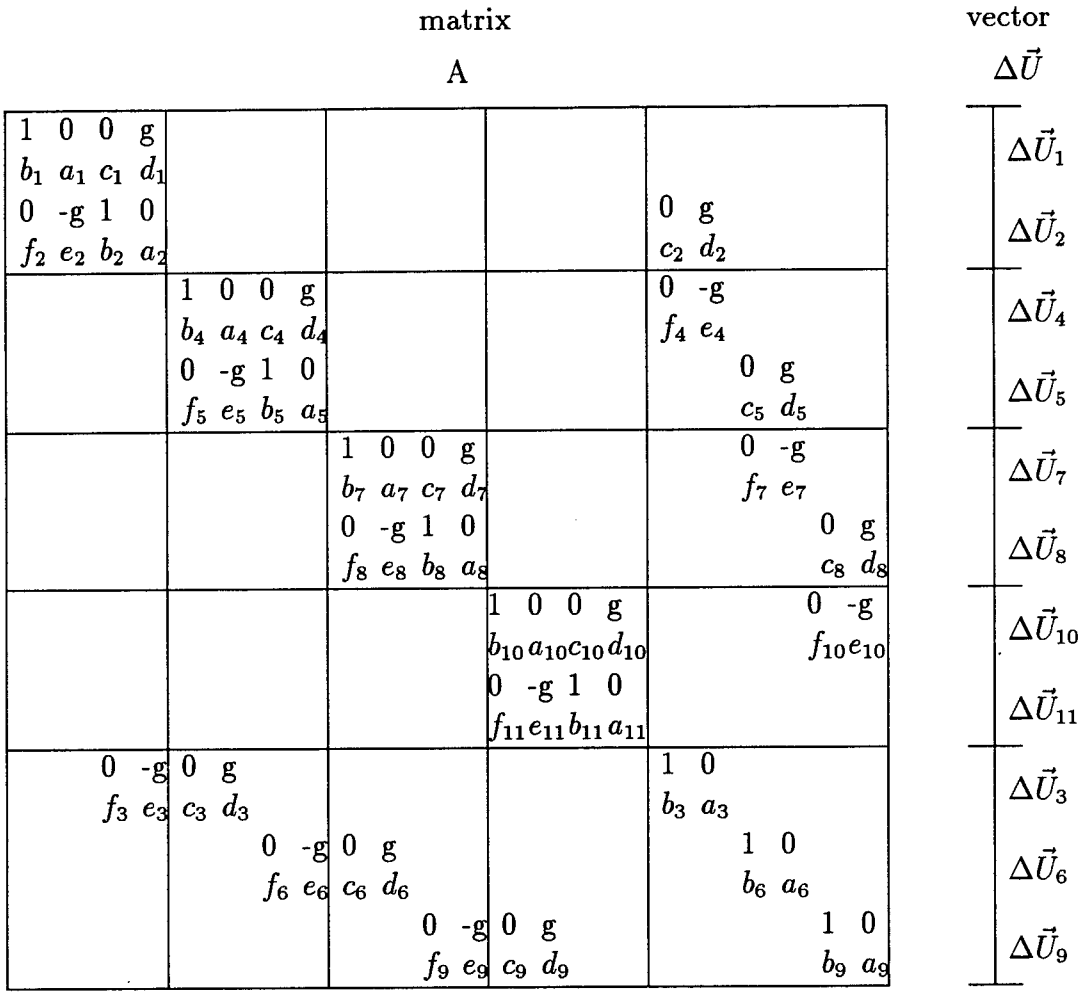


Figure 6. partition of linear system and mapping to CPUs.

In the Euler-Momentum method, we obtained the domain decomposition LU and use it as a preconditioning with Orthomin. Thus, the LU factorization is obtained once and the backward substitution is applied at each iteration of Orthomin.

8 Results

To test the program, the space mesh and time step size selection was made as follows. We set $\Delta t = 10sec$ and $\Delta x = 200$ ft. Our results are distinguished into three units: The q-k model curves testing, the comparisons with real data, the parallel computer implementation.

8.1 q-k Model curves tests

We used the I-35W data and the Greenshields', Least Squares, and Occupancy methods. The 3-D graphs of volume, density and speed are shown in Figures 11-19. The occupancy curves seem to provide the smoothest results and probably the Greenshields' model are the least accurate.

8.2 Comparisons with real data

The following 6 moduli are used to measure the effectiveness of the simulation in comparison with actual data collected at checkstation nodes of the freeway at the same time as the upstream/downstream data are being gathered.

$$Maximum\ Absolute\ Error = |Observed - Simulated| \tag{9}$$

$$Maximum\ Absolute\ Error = \frac{|Observed - Simulated|}{Observed} \tag{10}$$

$$Mean\ Absolute\ Error = \frac{1}{N}\sum_{i=1}^{N} |Observed - Simulated| \tag{11}$$

$$Mean\ Relative\ Error = \frac{1}{N}\sum_{i=1}^{N}\frac{|Observed - Simulated|}{Observed} \qquad (12)$$

$$Maximum\ Relative\ Error\ with\ 2-Norm = [\frac{\sum_{i=1}^{N}(Observed - Simulated)^2}{\sum_{i=1}^{N} Observed^2}]^{1/2} \qquad (13)$$

$$Standard\ Deviation = [\frac{1}{(N-1)}\sum_{i=1}^{N}(Observed - Simulated)^2]^{1/2} \qquad (14)$$

where N is the number of observations.

For I-35W and I-494 we used point entry/exit schemes. The error statistics are summarized in Tables 2-7. The relative errors are about more than 10% for the volume but are lower for the speed measurements.

Point Entry/Exit scheme. Lax $dt = 10sec$, Volume error $(veh/5min)$						
Sites	*Maximum*	*Max. Rel.*	*2 - Norm*	*Average*	*Average Rel.*	Std. Dev.
1	29.6	0.15	0.06	12.5	0.06	14.9
2	57.1	0.35	0.11	19.7	0.10	24.5
3	43.9	0.17	0.08	15.7	0.07	19.3
4	68.6	0.23	0.10	20.3	0.08	25.1
5	64.6	0.22	0.11	21.6	0.08	26.5

Table 2. Error statistics for Traffic flow (I-35W)

Point Entry/Exit scheme. Lax $dt = 10sec$, Speed error $(mile/hour)$						
Sites	*Maximum*	*Max. Rel.*	*2 - Norm*	*Average*	*Average Rel.*	Std. Dev.
1	2.5	0.06	0.03	1.3	0.03	1.4
2	5.4	0.13	0.07	3.3	0.07	3.5
3	6.2	0.15	0.08	3.4	0.08	3.7
4	7.0	0.17	0.08	3.3	0.07	3.7
5	11.9	0.32	0.13	5.6	0.12	6.0

Table 3. Error statistics for Traffic speed (I-35W)

Point Entry/Exit scheme, ILU version. Lax $dt = 10sec$, Volume error $(veh/5min)$						
Sites	*Maximum*	*Max. Rel.*	*2 - Norm*	*Average*	*Average Rel.*	Std. Dev.
1	56.2	0.24	0.12	29.0	0.12	31.0
2	59.8	0.19	0.11	32.1	0.11	35.1
3	71.4	0.18	0.11	27.8	0.09	33.1
4	41.3	0.13	0.07	16.1	0.05	21.0
5	94.6	0.24	0.13	35.2	0.11	42.8

Table 4. Error statistics for Traffic flow (I-494)

Point Entry/Exit scheme, ILU version. Lax $dt = 10sec$, Speed error $(mile/hour)$						
Sites	*Maximum*	*Max. Rel.*	*2 - Norm*	*Average*	*Average Rel.*	Std. Dev.
1	2.5	0.06	0.03	1.3	0.03	1.4
2	5.4	0.13	0.07	3.3	0.07	3.5
3	6.2	0.15	0.08	3.4	0.08	3.7
4	7.0	0.17	0.08	3.3	0.07	3.7
5	11.9	0.32	0.13	5.6	0.13	6.0

Table 5. Error statistics for Traffic speed (I-494)

Point Entry/Exit scheme, domain decomposition version. Lax $dt = 10sec$, Volume error $(veh/5min)$						
Sites	*Maximum*	*Max. Rel.*	*2 - Norm*	*Average*	*Average Rel.*	Std. Dev.
1	57.2	0.25	0.12	28.9	0.12	31.4
2	93.3	0.23	0.16	42.6	0.13	51.2
3	106.5	0.27	0.16	41.6	0.13	50.4
4	99.3	0.40	0.18	44.6	0.16	52.6
5	142.6	0.34	0.22	57.1	0.17	70.0

Table 6. Error statistics for Traffic flow (I-494)

Point Entry/Exit scheme, domain decomposition version. Lax $dt = 10sec$, Speed error $(mile/hour)$						
Sites	*Maximum*	*Max. Rel.*	*2 - Norm*	*Average*	*Average Rel.*	Std. Dev.
1	2.5	0.06	0.03	1.3	0.03	1.4
2	8.3	0.21	0.09	3.6	0.08	4.3
3	8.6	0.22	0.09	3.5	0.08	4.3
4	8.5	0.21	0.08	3.2	0.07	4.0
5	12.9	0.35	0.12	4.6	0.11	5.7

Table 7. Error statistics of for Traffic speed (I-494)

8.3 Parallel Implementation

time in ILU version for I-494						
No. of Processors	*Matrix & RHS*	*Solver*	*Dotproduct*	*Matvec*	*Lin. Com*	Total Ti
2	32.4	112.8	91.6	74.7	165.0	483.5
16	6.3	29.9	51.1	17.6	28.2	133.8

Table 8. Time in ILU version for I-494

time in the domain decomposition version for I-494						
No. of Processors	*Matrix & RHS*	*Solver*	*Dotproduct*	*Matvec*	*Lin. Com*	Total Ti
2	33.4	202.5	81.6	67.9	147.1	538.7
16	27.7	76.3	23.5	8.4	11.5	169.9

Table 9. Time in domain decomposition version for I-494

Percentage in ILU version for I-494					
No. of Processors	*Matrix & RHS*	*Solver*	*Dotproduct*	*Matvec*	Lin. Com
2	6.7	23.3	20.0	15.5	34.1
16	4.7	22.3	38.2	13.2	21.2

Table 10. Percentage in ILU version for I-494

Percentage in domain decomposition version for I-494					
No. of Processors	*Matrix & RHS*	*Solver*	*Dotproduct*	*Matvec*	Lin. Com
2	6.2	37.6	15.2	12.6	27.3
16	16.3	45.0	13.8	4.9	6.8

Table 11. Percentage in domain decomposition version for I-494

Parallel execution time from the eighteen mile stretch of I-494 case on both 2 and 16 processors on nCUBE for ILU and domain decomposition preconditioning are summarized in Table 8, 9, 10 and 11.

To test the program, the time stepsize selection was made as follows. For the Euler implicit method the time stepsize was increased such that the maximum error did not exceed that of the Lax method [2]. For the uncongested and entry/exit flow cases a single time stepsize of $dt = 10sec$ was selected.

We use the following moduli to measure the parallel processor performance.

$$\text{Speedup} = \frac{T_1}{T_P},$$

where T_1 is the execution time on one processor and T_P is the execution time on P processors.

$$\text{Efficiency} = \frac{\text{Speedup}}{P}.$$

The speedup and efficiency on nCUBE are summarized in Figure 7, 8, 9 and 10. The efficiency of the parallel processing drops as the number of processors increase.

9 Conclusions

We studied a high-order continuum model using an implicit (Euler) method. Implicit methods allow much larger time step size than explicit methods, for the same accuracy. However, at each time step a nonlinear system must be solved, we use the Newton method coupled with a linear iterative method (Orthomin). We accelerate the convergence of Orthomin with parallel incomplete LU factorization preconditioning. We wrote a code in C simulating a freeway uncongested pipeline and freeway entry/exit traffic flow. Tests

with real data collected from the I-35W and I-494 freeway in Minneapolis were conducted on a Sun workstation computer. We have also implemented efficiently Euler-Momentum method on the (16 processor) nCUBE2 parallel computer located at the Department of Computer Science. Each processor of the nCUBE2 is as powerful as a SUN 3/50 workstation. On the nCUBE2, the parallel Euler-Momentum method on the 16 processors runs 4 times faster than on the one processor.

References

[1] Chronopoulos, A. T., Michalopoulos, P. and Donohoe, J. (1992), "Efficient Traffic Flow Simulation Computations", In: *Mathematical and Computer Modelling, Vol. 16, No. 5, pp. 107-120*;

[2] Chronopoulos, A.T., Michalopoulos, P. and Rhee, C. (1993), "Traffic Flow Simulation Through High Order Traffic Modelling", In: *Mathematical Computing Modelling, Vol. 17, No. 8, pp. 11-22*

[3] Chronopoulos, A. T., Wang, G. and Rhee, C. (1994), "Real Time Freeway Traffic Flow Simulation Through Parallel Processing", to be published.

[4] Di Brozolo, R. and Robert, Y. (1989) "Parallel Conjugate Gradient-like Algorithms for Sparse Nonsymmmetric Systems on a Vector Multiprocessors", In: *Parallel Computing 11, (1989), pp. 223-239.*

[5] Dongarra, J.J. et al.(1979), "LINPACK user's guide. " *SIAM publications.*

[6] Gerlough, D. L. and Huber, M. J. (1975), "Traffic Flow Theory", *Transportation Research Board, special report 165, pp11.*

[7] Greenshields, B. D. (1934), "A study of traffic capacity ", In: *Proc. Highway Res. Board, Vol. 14, pp. 448-477.*

[8] Hirsch, C. (1988), "Numerical Computation of Internal and External Flows ", *Vol.2, John Wiley and Sons.*

[9] Kim, S. K. and Chronopoulos, A. T. " A Class of Lanczos-like Algorithms Implemented on Parallel Computers", Parallel Computing 17, pp. 763-778, 1991.

[10] Leo, C. J. and Pretty, R. L. (1990), "Some Comments and Numerical Tests on Upwind Finite Difference Schemes for Macroscopic Traffic Modeling", *Dept. of Civil Eng., Univ. of Queensland, Australia.*

[11] Lighthill, M. H. and Witham, G. B. (1955), "On Kinematic waves: II A theory of traffic flow on long crowded roads", In: *Proceed. R. Soc. Ser. A 229, No. 1178, pp. 317-345.*

[12] Ma, S. and Chronopoulos, A. T. (1990), "Implementation of Iterative Methods for Large Sparse Nonsymmetric Systems on Parallel Vector Computers", In: *Int. J. on Supercomputing, 4, (1990), pp. 9-24.*

[13] McShane, W. and Roess, R. (1990), "Traffic Engineering ", *Prentice Hall.*

[14] Michalopoulos, P. G., Kwon, E. and Kang, J. G. (1991), "Enhancements and Field Testing of a Dynamic Freeway Simulation Program " *Transp. Res. Rec. 1320, pp. 203-215.*

[15] Michalopoulos, P. G. and Lin, J. (1985), " A freeway simulation program for microcomputers ", In: *Proceed., 1st National Conf. on Microcomputers in Urban Transp., ASCE, California, pp. 330-341.*

[16] Michalopoulos, P. G., Yi, Ping, Beskos, D.E. and Lyrintzis, A. S. (1991), " Continuum Modeling of Traffic Dynamics ", In: *Proc. of the 2nd Int. Conf. on Appl. of Advanced Tech. in Transportation Eng., Aug. 18-21, Minneapolis, Minnesota, pp. 36-40*

[17] Ortega, J. M. (1988), "Introduction to Parallel and Vector Solution of Linear Systems", *Plenum Press, pp. 120-124.*

[18] Payne, H. J. (1971), "Models of freeway traffic and control", In: *Proceed. Math. of Publ. Syst. published by Simul. Council, Vol. 1, No. 1, pp. 51-61.*

[19] Payne, H. J. (1979), "FREEFLO: A macroscopic simulation model of freeway traffic" *TRR, 772, pp. 68-75*

ACKNOWLEDGMENTS

This project was funded through the Center for Transportation Studies, ITS Institute Program.

Appendix A

Figure 7. I-494 speedup for parallel ILU version on NCUBE

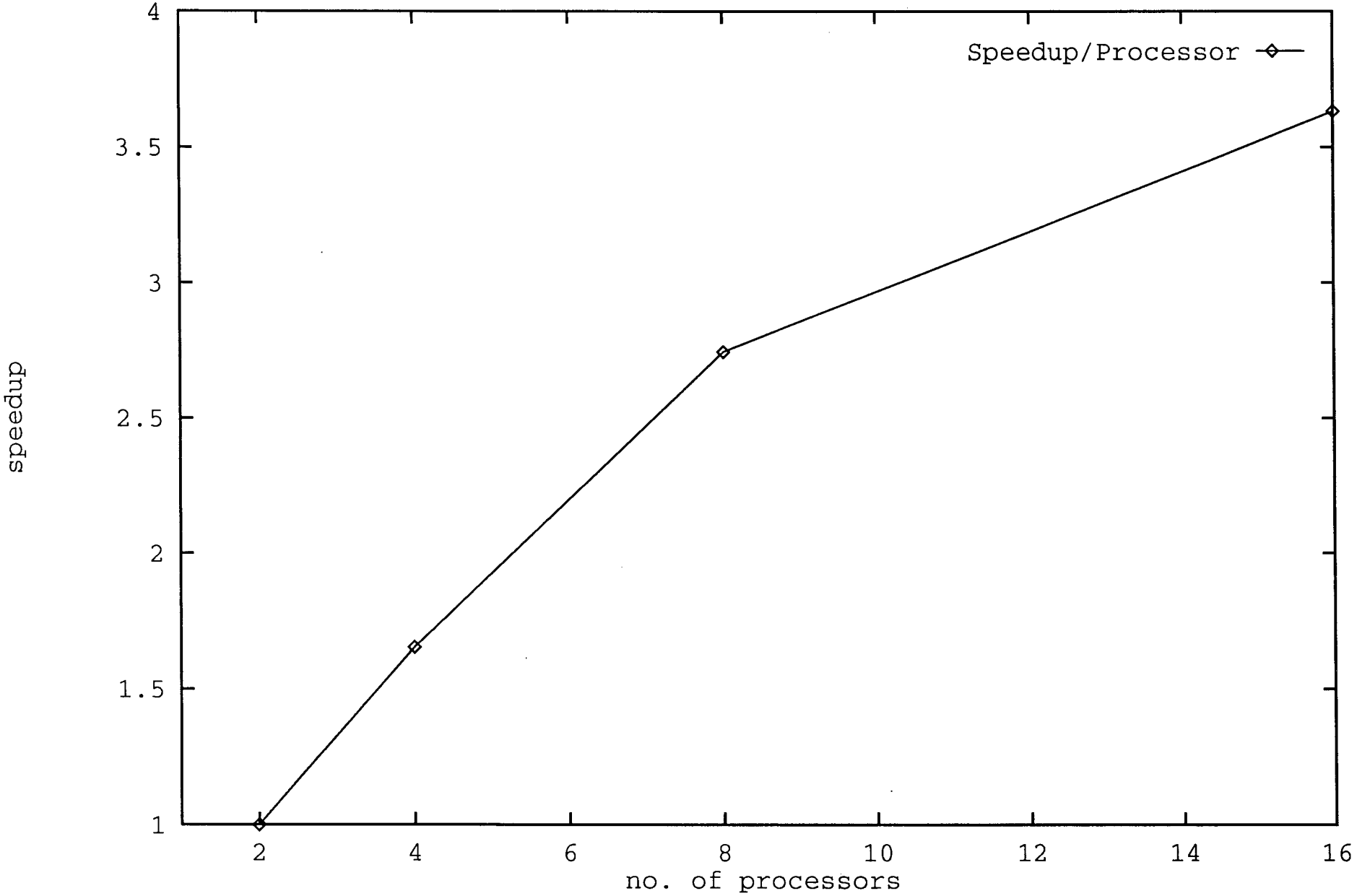


Figure 8. I-494 efficiency for parallel ILU version on NCUBE

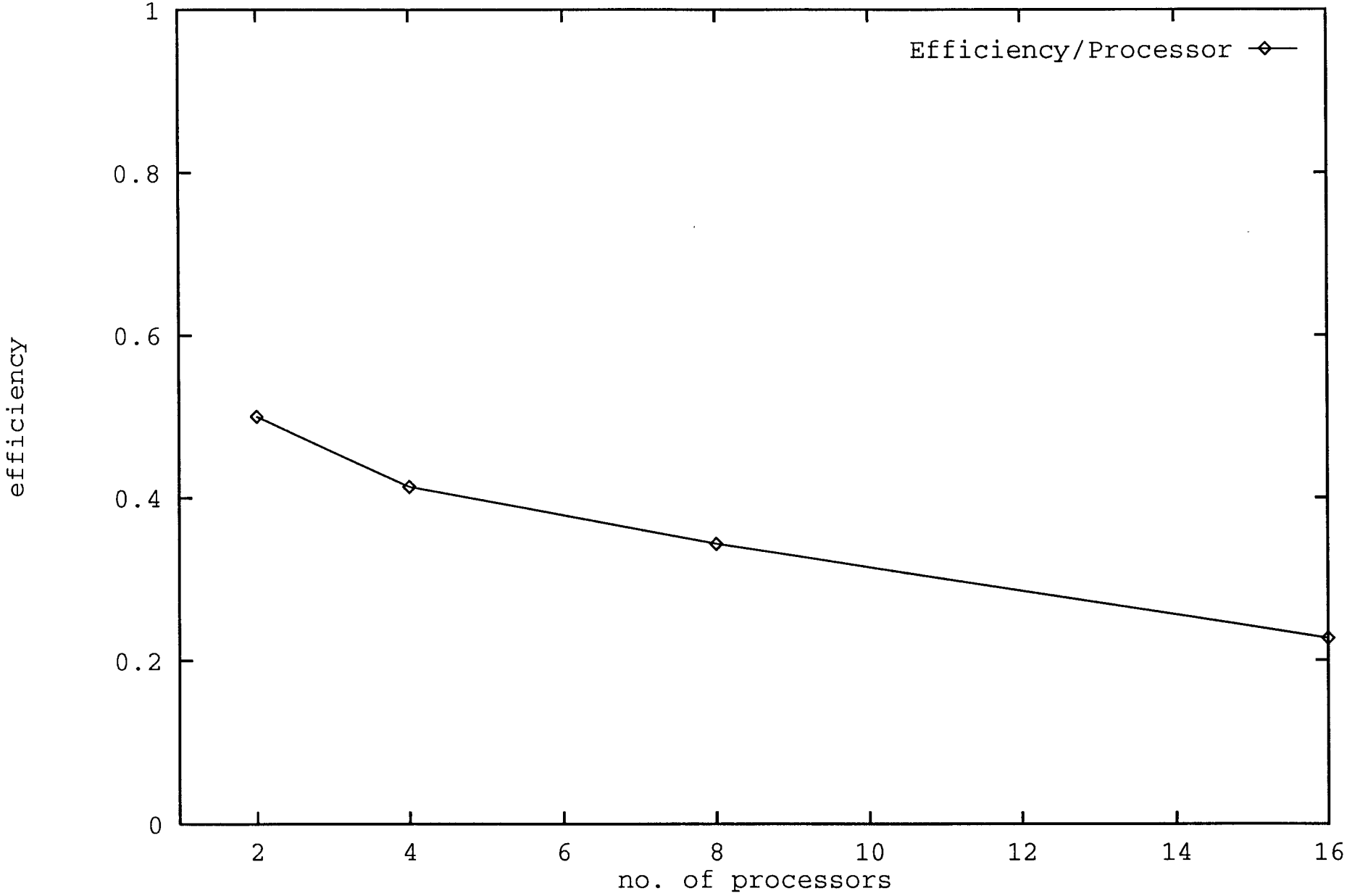


Figure 9. I-494 speedup for domain decomposition version on NCUBE

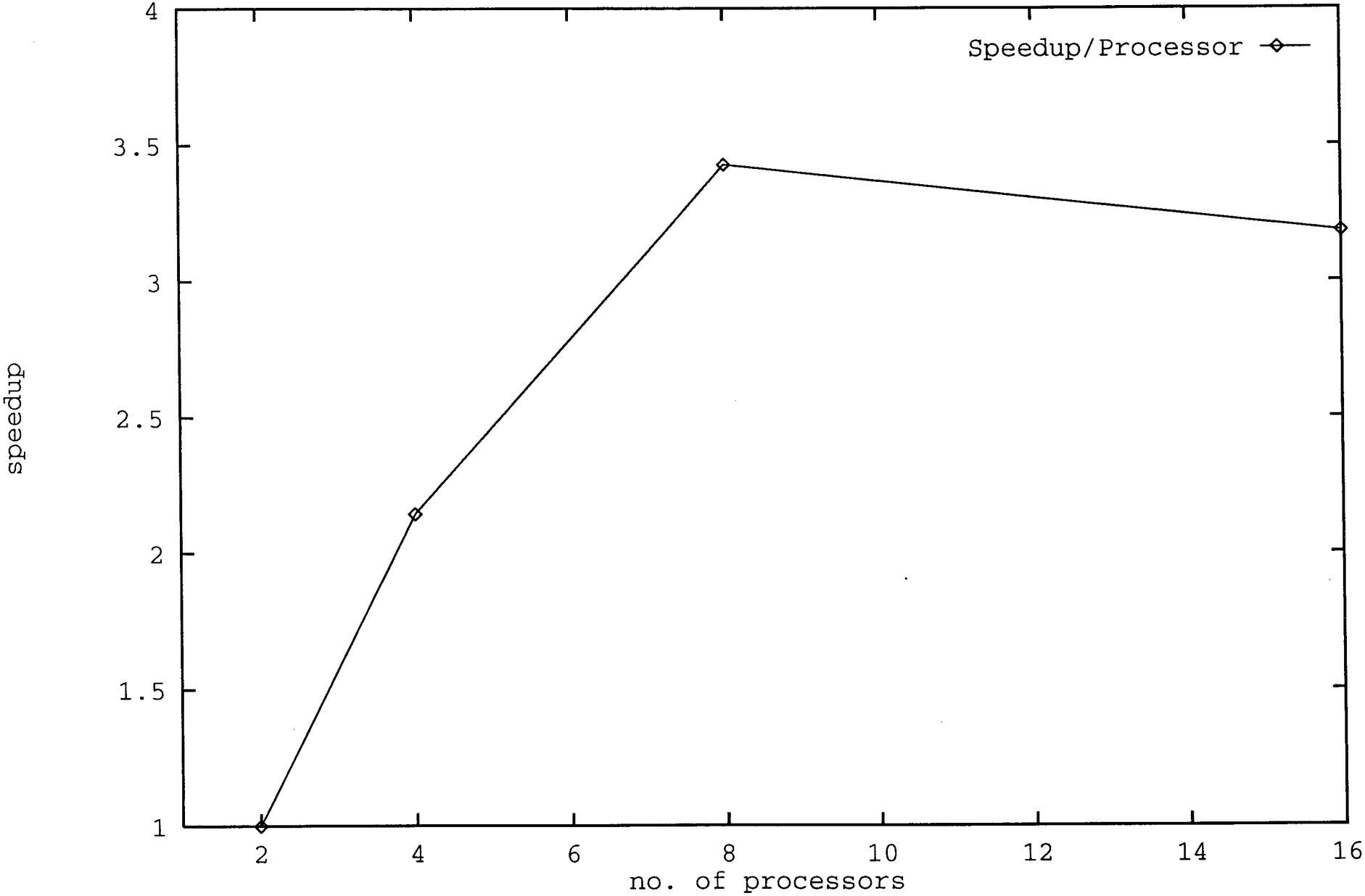
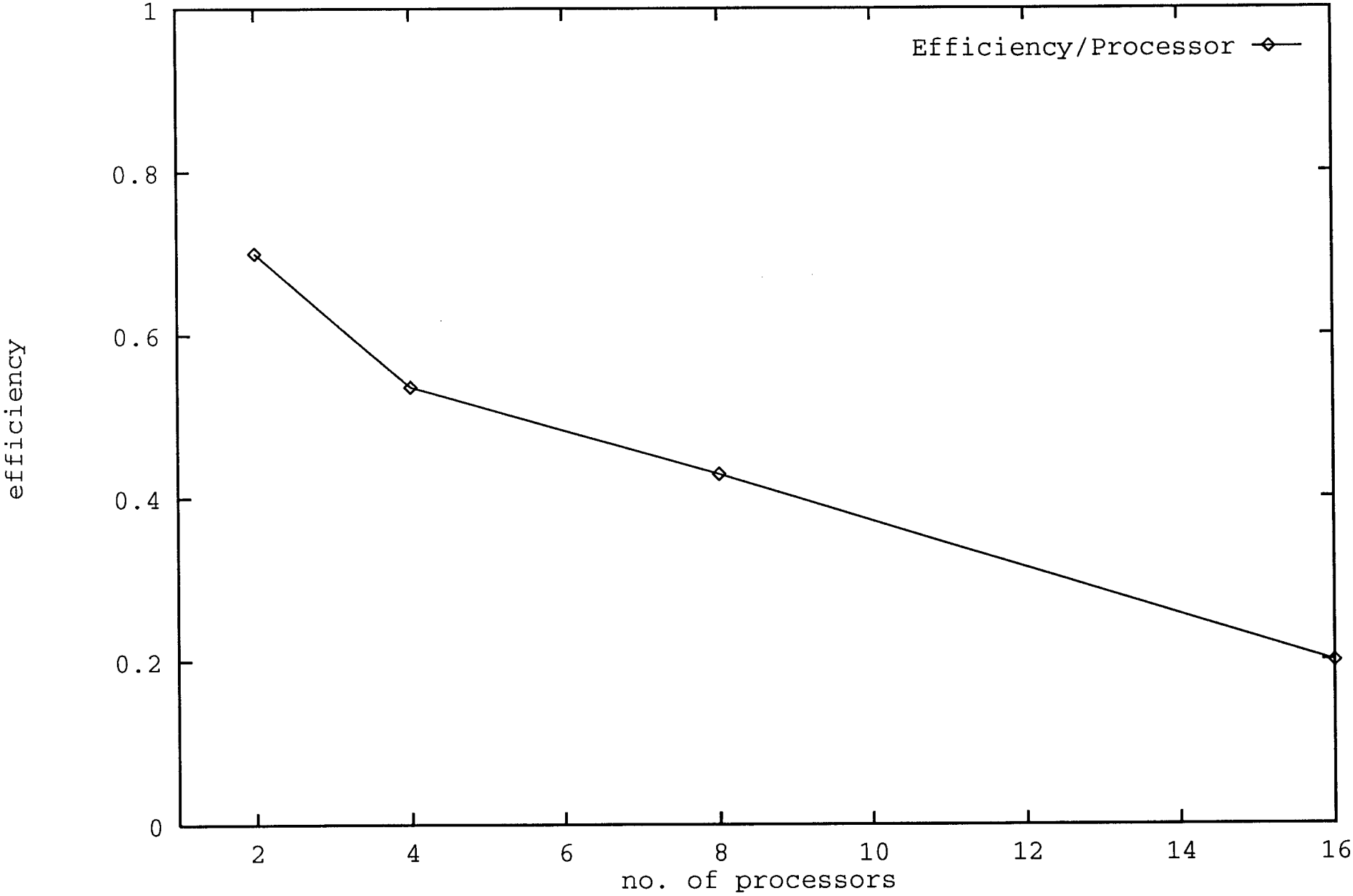


Figure 10. I-494 efficiency for domain decomposition version on NCUBE



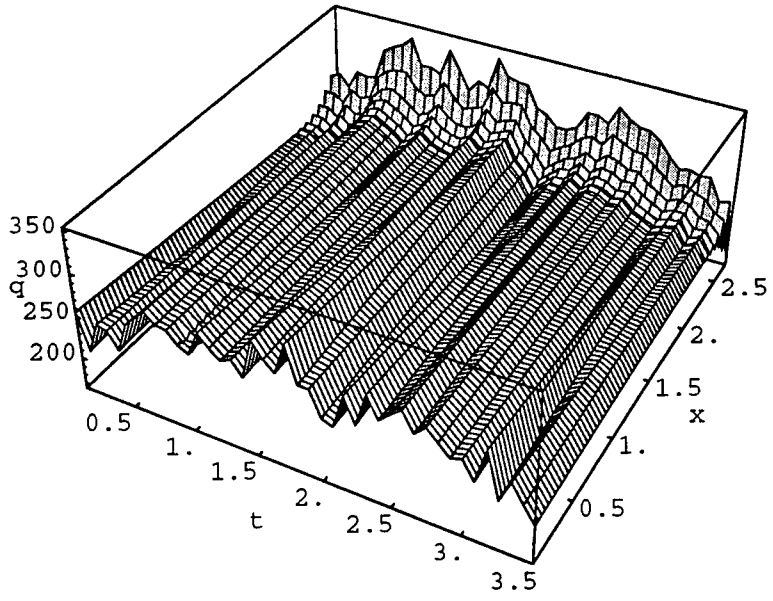


Fig. 11. Volume (q: cars/5min/lane) vs. (t: hr,x: Mi)
Euler-Greenshield's method (I-35W)

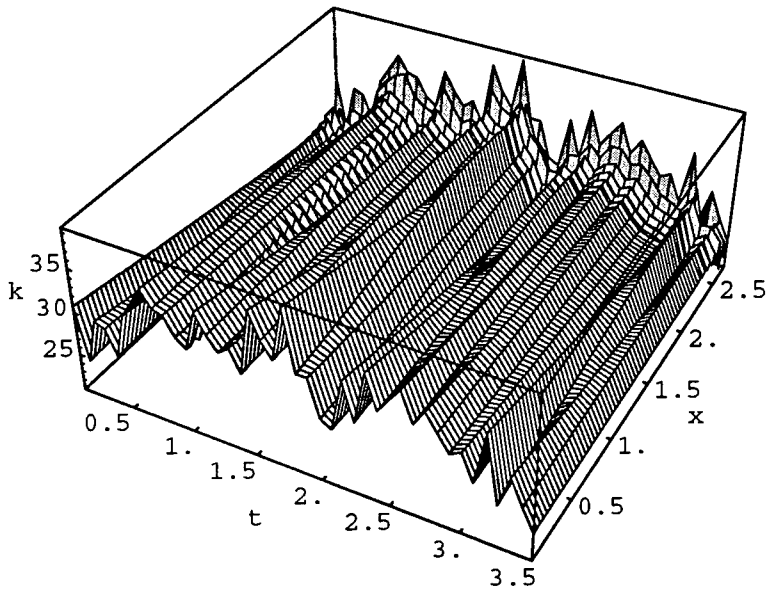


Fig. 12. Density (k: cars/Mi) vs. (t: hr,x: Mi)
Euler-Greenshield's method (I-35W)

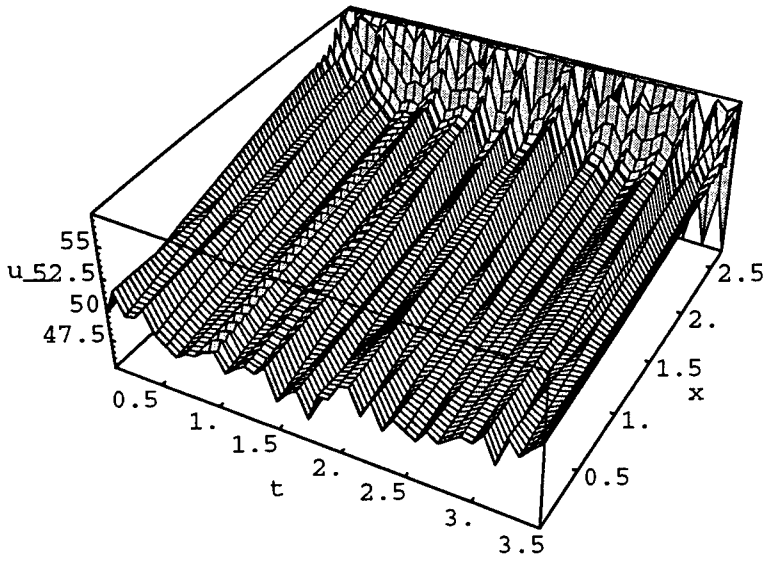


Fig. 13. Speed (u: Mi/hr) vs. (t: hr,x: Mi)
Euler-Greenshield's method (I-35W)

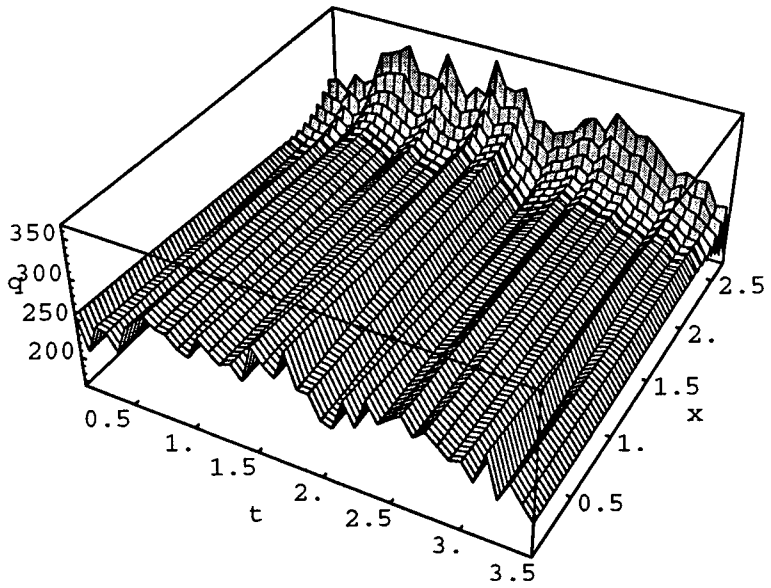


Fig. 14. Volume (q: cars/5min/lane) vs. (t: hr,x: Mi)
Euler-Least Squares method (I-35W)

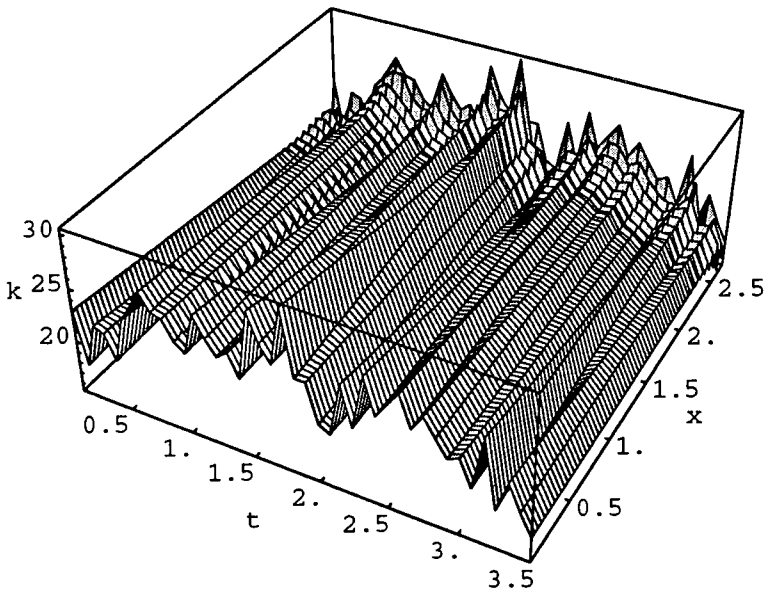


Fig. 15. Density (k: cars/Mi) vs. (t: hr,x: Mi)
Euler-Least Squares method (I-35W)

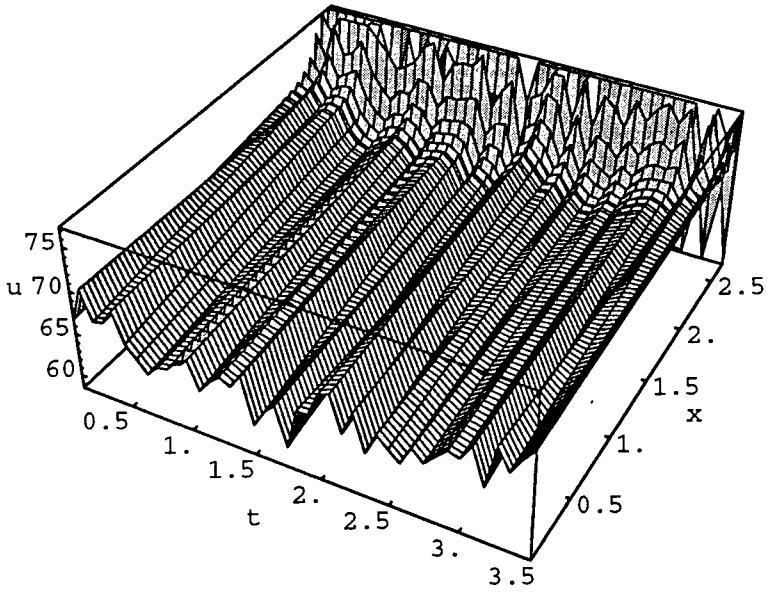


Fig. 16. Speed (u: Mi/hr) vs. (t: hr,x: Mi)
Euler-Least Squares method (I-35W)

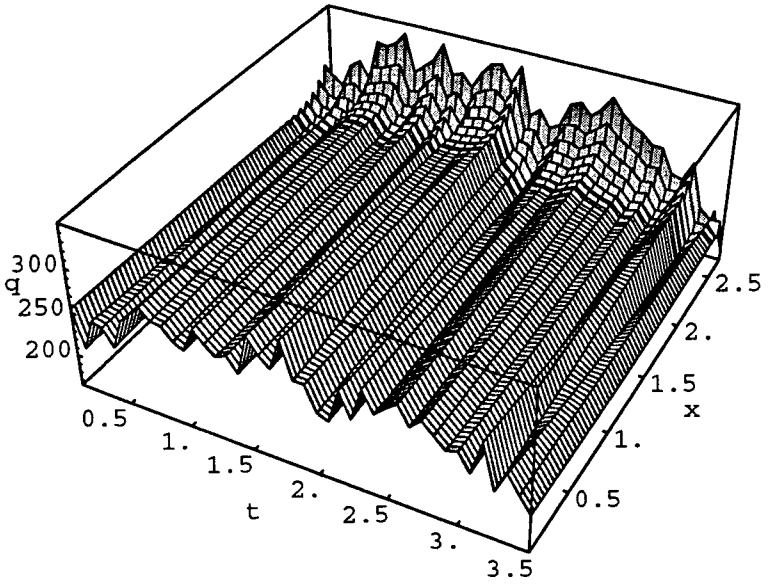


Fig. 17. Volume (q: cars/5min/lane) vs. (t: hr,x: Mi) Euler-Occupancy method (I-35W)

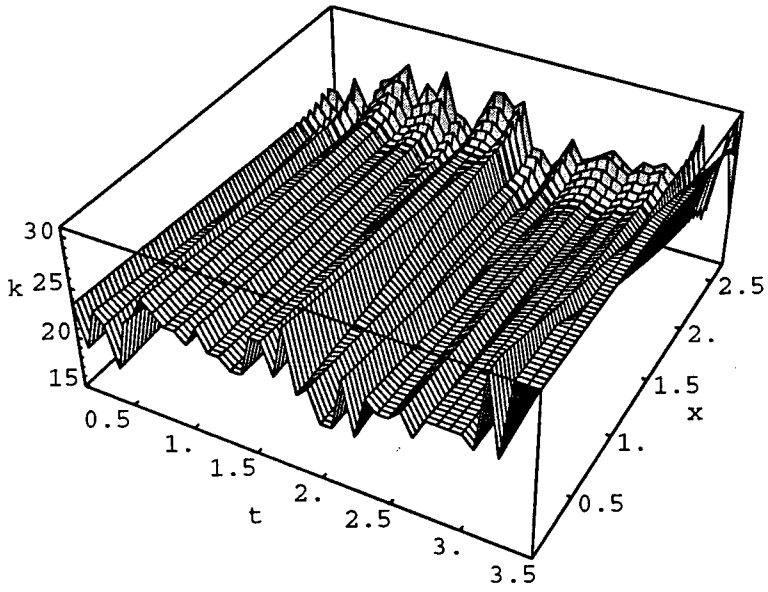


Fig. 18. Density (k: cars/Mi) vs. (t: hr,x: Mi) Euler-Occupancy method (I-35W)

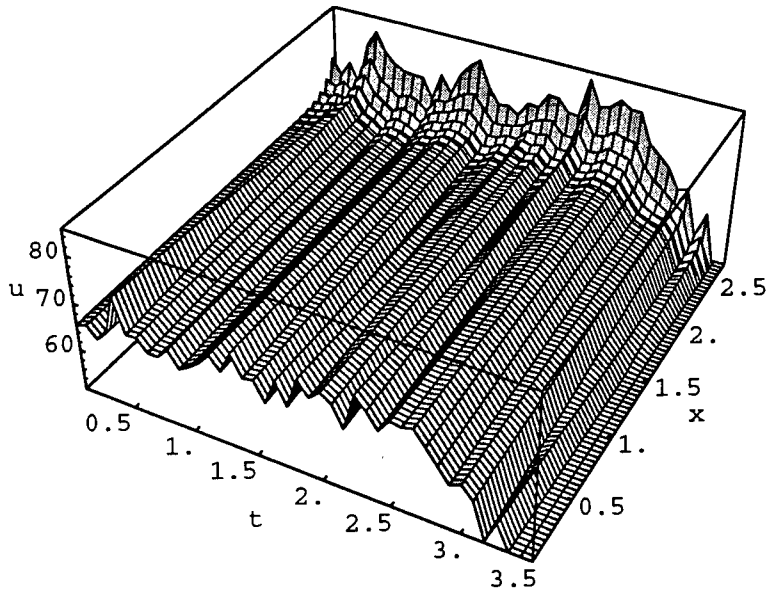


Fig. 19. Speed (u: Mi/hr) vs. (t: hr,x: Mi)
Euler-Occupancy method (I-35W)

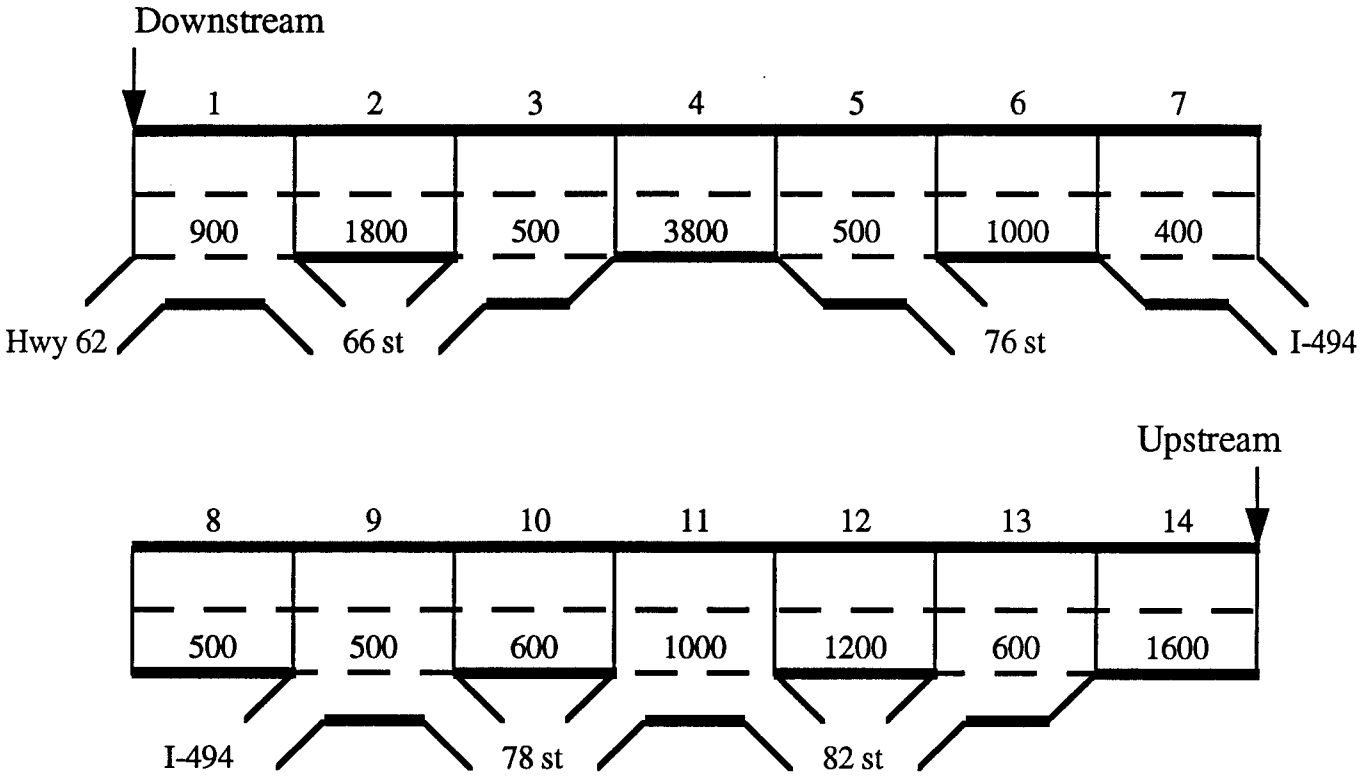


Figure 20. Freeway Geometries for I-35W from 46th to 86th Street (Southbound)