Parallel Solution of Burger's Equation in Two Dimensions

J. Koppelman*
joshua@mit.edu
MIT Laboratory for Computer Science

A. Caola†
caola@mit.edu
MIT Department of Chemical Engineering

R. Lippert‡
ripper@mit.edu
MIT Department of Mathematics

May 18,1995

In this study, a time and space second order algorithm is used to evolve Burger's equation in two dimensions. The parallel programming package LPARX [1] was employed on an IBM SP2 to simplify the parallelization of the algorithm. The resulting algorithm solves Burger's equation on any two dimensional geometry with arbitrary initial conditions that may be decomposed into rectangles. The instantaneous state of the flow field can be written to files during the solution period, so that MATLAB [2] may be used to provide animations of shockwaves traveling in the flow field. This allows us to determine which states or configurations are likely for the given geometry and boundary conditions.

Burger's equation may be thought of as a simplified version of the Navier-Stokes equations [3]. The one dimensional form of Burger's equation is given by

$u_t + uu_x = \frac{1}{Re} u_{xx}$

Turbulent features are still present because both dissapative and inertial terms appear. Physically, the equation may be thought of as refering to an infinitely compressible fluid with no pressure gradients or shear forces: u refers to the velocity and Re is the Reynolds number. The higher Re becomes, the more inviscid the character of the flow becomes. At the limit $Re \rightarrow \infty$, Burger's equation reduces to a wave equation – nonzero values of Re cause diffusion of the shockwaves.

Our two-dimensional Burger's equation is:

$u_t + uu_x = \frac{1}{Re}(u_{xx} + u_{yy})$

The numerical algorithm employed uses centered differences to approximate the spacial

*Joshua Koppelman is advised by Dr. Stephanie Seneff. Address: Room NE43-607, 545 Technology Square, Cambridge MA 02139

†Anthony Caola is advised by Prof. R. Brown. Address: Room 66-342, 77 Massachusetts Avenue, Cambridge MA 02139

‡Ross Lippert is advised by Prof. G-C Rota. Address: Room 26-342, 77 Massachusetts Avenue, Cambridge MA 02139

derivatives and evolves the system in time using a second order Runge-Kutta scheme.

Specifically, we approximate the time derivative as:

$$\frac{\partial u(x,y)}{\partial t} = -u(x,y)\left(\frac{u(x+\Delta x,y)-u(x-\Delta x,y)}{\Delta x}\right)$$
$$+ \frac{1}{R}\left(\frac{u(x+\Delta x,y)+u(x-\Delta x,y)-2u(x,y)}{(\Delta x)^2}\right.$$
$$\left.+ \frac{u(x,y+\Delta y)+u(x,y-\Delta y)-2u(x,y)}{(\Delta y)^2}\right).$$

We then evolve the system by:

$$w(t,x,y) = u(t,x,y) + \Delta t \frac{\partial u(t,x,y)}{\partial t}$$
$$u(t+\Delta t,x,y) = \frac{1}{2}\left(u(t,x,y) + w(t,x,y) + \Delta t \frac{\partial w(t,x,y)}{\partial t}\right).$$

A linear stability analysis shows that $\Delta t \leq \frac{1}{8}R(\Delta x)^2$ and that $|u| \leq 4R\Delta x$ for the evolution to be stable.

Our goal was to develop an environment in which it would be easy to simulate the behavior of Burger's equation on various initial conditions and topologies. To that end, we have developed a simple descriptive language which allows a user to easily specify the physical characteristics of any envisioned scenario.

Because of the large amount of computation required, we decided to run the simulation in parallel on an IBM SP2 with 12 processors. This parallelism is completely transparent to the descriptive language; our code automatically partitions the specified space into reasonable sections (with an eye toward maintaining good load balancing properties). Each processor moves its assigned space through time, having to communicate with its nearest neighbors only once every iteration.

Figure 1 represents the output of a sample scenario after two seconds have elapsed. The region has a constant source of flow at the left edge, and contains four "obstacles" – square regions permanantly set to zero flow. The simulation runs for two seconds, during which twenty snapshots are taken so that an animation can be created depicting the changes in flow field over time.

We have taken timing statistics as we vary the number of processors and the area (i.e. the number of nodes) of the grid. For small grids (less than $20,000$ nodes), it is not worthwhile to parallelize the computation, since the communication time between processors is much larger than the computation time on one processor. However, as we increase the size of the grid, the computation time for a single processor becomes prohibitively large, making the need for multiple processors apparent. For medium sized grids ($60,000$ to $90,000$ nodes), three of four processors is optimal. For large grids with upwards of $150,000$ nodes, all twelve are necessary.

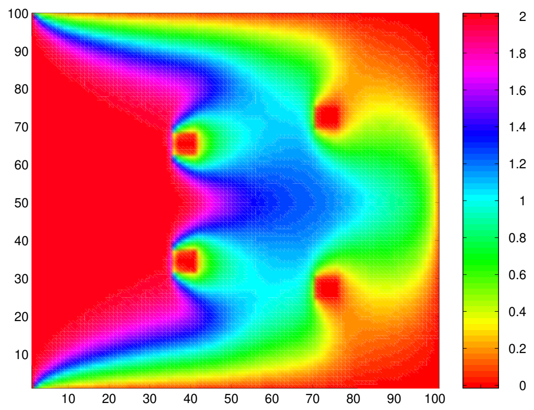

Figure 1: Fluid Velocity at $t = 2.0$ seconds

References

[1] S. Baden, S. Kohn, S. Figueira, and S. Fink, *LPARX User's Guide, v2.0.* University of California, San Diego, through ftp to cs.ucsd.edu, November 1994.

[2] The MathWorks, Inc., Natick, MA, *MATLAB Reference Guide*, August 1992.

[3] J. M. Burgers, *The Nonlinear Diffusion Equation.* Boston, MA, USA: D. Reidel Publishing Company, 1973.