A GPU/Multi-core Accelerated Multigrid Preconditioned Conjugate Gradient Method for Adaptive Mesh Refinement

Austen Duffy - Florida State University

SIAM Conference on Computational Science and Engineering

March 2, 2011

Acknowledgements

This work is with

- Mark Sussman - FSU Mathematics

And supported by the National Science Foundation through grant DMS-1016381

Introduction

In this talk We will describe

- A new method for computing the pressure projection step of our coupled level set and volume of fluid (CLSVOF) code on adaptive meshes using the multigrid preconditioned conjugate gradient method.
- Steps involving the GPU acceleration of the code.

Outline

Governing Equations for two-phase flow

$$\rho_L \frac{D\mathbf{u}_L}{Dt} = -\nabla p_L + 2\mu_L \nabla \cdot \mathcal{D} + \rho_L \mathbf{g}, \quad \nabla \cdot \mathbf{u}_L = 0, \quad \mathbf{x} \in \text{liquid},$$

$$\rho_g \frac{D\mathbf{u}_g}{Dt} = -\nabla p_g + 2\mu_g \nabla \cdot \mathcal{D} + \rho_g \mathbf{g}, \quad \nabla \cdot \mathbf{u}_g = 0, \quad \mathbf{x} \in \text{gas},$$

$$D = \frac{\nabla \mathbf{u} + (\nabla \mathbf{u})^T}{2}$$

Governing Equations for two-phase flow (cont)

Interfacial Conditions

$$(2\mu_L \mathcal{D} - 2\mu_g \mathcal{D}) \cdot \mathbf{n} = (p_L - p_g + \sigma\kappa)\,\mathbf{n} \quad \text{and} \quad \mathbf{u}_L = \mathbf{u}_g, \quad \mathbf{x} \in \Gamma,$$

$$\mathbf{u}_L = \mathbf{u}_g, \quad \mathbf{x} \in \Gamma$$

$$\frac{d\mathbf{x}}{dt} = \mathbf{u}(\mathbf{x}(t), t), \quad \mathbf{x} \in \Gamma$$

$$\kappa = \nabla \cdot \mathbf{n}, \quad \mathbf{x} \in \Gamma$$

Level set equations for multiphase flow.

Weak solutions of the following equations satisfy the interfacial boundary conditions:

$$\rho \frac{D\mathbf{u}}{Dt} = \nabla \cdot (-pI + 2\mu D) + \rho g \mathbf{z} - \sigma \kappa \nabla H$$

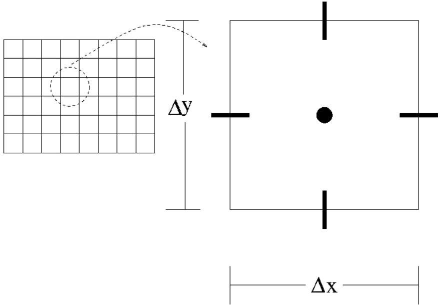
$$\nabla \cdot \mathbf{u} = 0 \quad \frac{D\phi}{Dt} = 0 \quad \kappa(\phi) = \nabla \cdot \frac{\nabla \phi}{|\nabla \phi|}$$

$$H(\phi) = \begin{cases} 1 & \phi \geq 0 \\ 0 & \phi < 0 \end{cases}$$

$$\rho = \rho_L H(\phi) + \rho_G (1 - H(\phi)) \quad \mu = \mu_L H(\phi) + \mu_G (1 - H(\phi))$$

Y.C. Chang, T.Y. Hou, B. Merriman, and S. Osher, A Level Set Formulation of Eulerian Interface Capturing Methods for Incompressible Fluid Flows, J.Comput.Phys.,124 (1996), pp. 449-464.

Staggered grid discretization



The cell centered variables $\phi_{i,j}$ and $p_{i,j}$ are approximations to $\phi(x_i, y_j)$ and $p(x_i, y_j)$ respectively where $x_i = (i + 1/2)\Delta x$ and $y_j = (j + 1/2)\Delta y$. The horizontal face centered velocity, $u^{fc}_{i+1/2,j}$, is an approximation to $u(x_{i+1/2}, y_j)$ where $x_{i+1/2} = (i + 1)\Delta x$ and $y_j = (j + 1/2)\Delta y$. The vertical face centered velocity, $v^{fc}_{i,j+1/2}$, is an approximation to $v(x_i, y_{j+1/2})$ where $x_i = (i + 1/2)\Delta x$ and $y_{j+1/2} = (j + 1)\Delta y$.

Projection method

1. Coupled level set and volume of fluid interface advection: $\frac{D\phi}{Dt} = 0$, $\frac{DF}{Dt} = 0$.
2. Velocity advection: $\frac{D\mathbf{u}}{Dt} = 0$, $\mathbf{F}^{fc,advect} = \left(\frac{\mathbf{u}^{cc,*} - \mathbf{u}^{cc,n}}{\Delta t}\right)^{c \to f}$
3. Viscous force term (sub-cycling algorithm):
   1. $\mathbf{u}^{cc,(0)} = \mathbf{u}^{cc,*}$
   2. For $k = 1, \ldots K$, $\mathbf{u}^{cc,(k)} = \mathbf{u}^{cc,(k-1)} + \frac{\Delta t}{K} \frac{1}{\rho^{cc}(\phi^{n+1})} \nabla \cdot (2\mu(\phi^{n+1}) D^{(k-1)})$
   3. $\mathbf{F}^{fc,visc} = \left(\frac{\mathbf{u}^{cc,(K)} - \mathbf{u}^{cc,(0)}}{\Delta t}\right)^{c \to f}$

5. Project velocity:

$$\mathbf{V}^{fc} = \mathbf{u}^{fc,n} + \Delta t \left(\mathbf{F}^{fc,advect} + \mathbf{F}^{fc,visc} + g\mathbf{z} - \sigma\kappa\nabla H/\rho\right)$$

$$\nabla \cdot \frac{1}{\rho}\nabla p = \frac{1}{\Delta t}\nabla \cdot \mathbf{V}^{fc}$$

$$\mathbf{u}^{fc,n+1} = \mathbf{V}^{fc} - \Delta t \frac{\nabla p}{\rho}$$

Pressure Poisson Equation

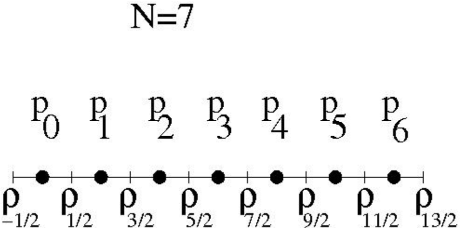
We will be concerned with the solution of the pressure Poisson equation

$$\nabla \cdot \frac{1}{\rho} \nabla p = F \tag{1}$$

The phase density ρ is represented in liquid and gas phases by $\rho = \rho_L H(\phi) + \rho_G (1 - H(\phi))$, where $H(\phi)$ is a Heaviside function equal to 1 in liquid and 0 in gas.

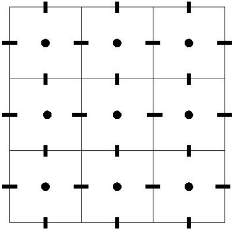
1D Discretization of Pressure Equation

$$\frac{1}{\rho_{i+\frac{1}{2}}}\frac{p_{i+1}-p_i}{\Delta x}-\frac{1}{\rho_{i-\frac{1}{2}}}\frac{p_i-p_{i-1}}{\Delta x}=\Delta x F_i, \quad (2)$$



MGPCG-AMR, 2D discretization of pressure equation

$$\beta_{i+1/2,j}(p_{i+1,j} - p_{i,j}) - \beta_{i-1/2,j}(p_{i,j} - p_{i-1,j}) +$$
$$\beta_{i,j+1/2}(p_{i,j+1} - p_{i,j}) - \beta_{i,j-1/2}(p_{i,j} - p_{i,j-1}) = \quad f_{i,j}\Delta x^2$$



Matrix for Pressure Projection Step

Solve $\nabla \cdot \beta \nabla p = f \quad \rightarrow \quad Ax = b$

$$
A = \left|\begin{array}{ccccccccc}
a & b & & & d & & & & \\
c & a & b & & & d & & & \\
 & c & a & b & & & . & & \\
 & & . & . & . & & & . & \\
e & & & . & . & . & & & d \\
 & e & & & . & . & . & & \\
 & & . & & & c & a & b & \\
 & & & . & & & c & a & b \\
 & & & & e & & & c & a
\end{array}\right|
$$

$$a = -(\beta_{i+1/2,j} + \beta_{i-1/2,j}) - (\beta_{i,j+1/2} + \beta_{i,j-1/2})$$

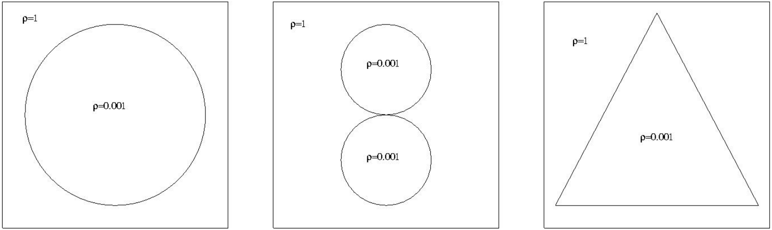
$$b = \beta_{i+1/2,j} \quad c = \beta_{i-1/2,j} \quad d = \beta_{i,j+1/2} \quad e = \beta_{i,j-1/2}$$

Condition Number and Density Ratio

In multiphase flows, the condition number of the discretization matrix for eqn. 1 grows with the density ratio. In this table, the condition number is calculated for the discretization matrix of a 1D two phase flow in the domain $[0, 1]$ following eqn. 2, and with the phase interface occurring at $x = 0.25$. The example flow has a density of 1 in the first phase on the interval $[0, 0.25)$ and α in a second phase on the interval $(0.25, 1]$. Values represent a discretization with 256 grid points and were calculated in MATLAB using the built in condition number function cond().

α	1	10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-5}
Density Ratio	1	10	10^2	10^3	10^4	10^5
Condition #	205	1.2×10^3	1.2×10^4	1.2×10^5	1.2×10^6	1.2×10^7

Introduction - Condition Number and Problem Geometry



The condition number of the discretization matrix is not as sensitive to the problem geometry as it is to the density ratio. The corresponding condition numbers for these figures are 6,132,300 (left), 1,861,000 (middle) and 2,548,900 (right) using a 2D version of discretization eqn. 2 on a 64 x 64 grid.

Other possible MG Approaches for Inc. Two Phase Flow

- Klaus Stuben, Patrick Delaney, Serguei Chmakov, Algebraic Multigrid (AMG) for Ground Water Flow and Oil Reservoir Simulation
- Ruge, J.W., Stuben, K., 1986. Algebraic Multigrid (AMG), in .Multigrid Methods. (S. McCormick, ed.), Frontiers in Applied Mathematics, Vol 5, SIAM, Philadelphia.
- The Black Box Multigrid Numerical Homogenization Algorithm J. David Moulton, Joel E. Dendy Jr., and James M. Hyman JCP 142, (1998)
- Wan and Liu, "A boundary condition capturing multigrid approach to irregular boundary problems," SIAM J. Sci. Comput., 2004.

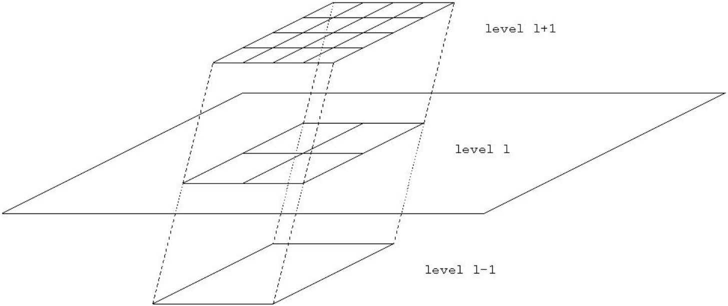
Other possible MG Approaches for Inc. Two Phase Flow (cont)

- Mayo, "The fast solution of poisson's and the biharmonic equations in irregular domains," SIAM J. Num. Anal., 1984.
- Howell and Bell, "An adaptive-mesh projection method for viscous incomopressible flow," SIAM J. Sci. Comp., 1997.
- Schaffer, A Semicoarsening Multigrid Method for Elliptic PDE'S with Highly Discontinuous and Anisotropic Coefficients. SIAM J. SCI. COMP., 1998.

PCG on Adaptive Grids

- F. Losasso, R. Fedkiw, and S. Osher. Spatially adaptive techniques for level set methods and incompressible flow. Comput. Fluids, 35(10):9951010, 2006.
- F. Lossaso, F. Gibou, and R. Fedkiw. Simulating water and smoke with an octree data structure. ACM Trans. Graph., 23:457462, 2004.
- S. Popinet. Gerris: a tree-based adaptive solver for the incompressible Euler equations in complex geometries. J. Comput. Phys., 190(2):572600, 2003.
- S. Popinet. An accurate adaptive solver for surface-tension-driven interfacial flows. J. Comput. Phys., 228:5838-5866, 2009.

MG-MGPCG Algorithm for Adaptive Meshes



MG-MGPCG on level $\ell + 1$ requires calculations at levels ℓ and $\ell - 1$

MG-MGPCG AMR Algorithm

Given x^0, $r = b - Ax^0$, $x = x^0$, $\delta x = 0$
Repeat until $||r|| < \epsilon$
1. Call relax!$(\delta x, r, \ell^{max})$ on finest level
2. Let $x = x + \delta x$, $r = r - A(\delta x)$

```
Recursive Routine relax!(sol, rhs, ℓ)
if Coarsest Level then
   Solve exactly using MGPCG
else
   (a) Presmoothing Step
   for i = 1 to presmooth do
      Smooth using MGPCG on level ℓ until ||r_ℓ|| < ε/10
   end for
   (b) Restriction Step
         (i) restrict!(r) to covered level ℓ − 1 cells and exposed level ℓ − 1
   cells neighboring a covered cell.
         (ii) cor = 0
   (c) Relaxation on Next Coarser Level
         Call relax!(cor^coarse, rhs^coarse, ℓ − 1)
   (d) Prolongate the Correction to the present level l cells covering coarse
   level ℓ − 1 cells and one layer of ``virtual" level ℓ cells.
         sol = sol + I(cor)
   (e) Postsmoothing Step
   for i = 1 to postsmooth do
      Smooth using MGPCG on level ℓ
   end for
end if
```

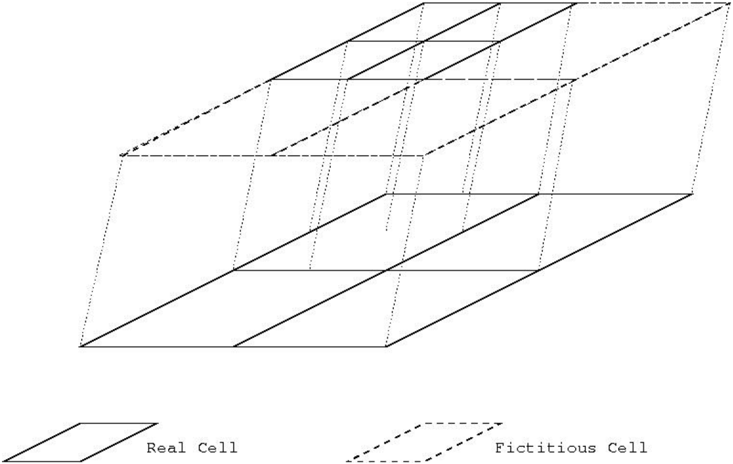
Improved MGPCG AMR Algorithm

Given x^0, $r = b - Ax^0$, $x = x^0$, $\delta x = 0$
$z = 0$
Call relaxAMR(z, r, ℓ^{max})
$\rho = z \cdot r$
if $n = 1$ **then**
  $p = z$
else
  $\beta = \frac{\rho}{\rho_{old}}$
  $p = z + \beta p$
end if
$\alpha = \frac{\rho}{p \cdot (Ap)}$
$\rho_{old} = \rho$
$x = x + \alpha p$
$r = r - \alpha Ap$

Improved MGPCG AMR Algorithm (cont.)

Recursive Routine relaxAMR(sol, rhs, ℓ)
if Coarsest Level **then**
  Solve exactly using MGPCG
else
  (a) Presmoothing Step
  for $i = 1$ to presmooth **do**
    Smooth using ILU on level ℓ
  end for
  (b) Restriction Step
    (i) restrict(r) to covered level $\ell - 1$ cells and recalculate r on exposed level $\ell - 1$ cells neighboring a covered cell.
    (ii) $cor = 0$
  (c) Relaxation on Next Coarser Level
    Call `relaxAMR!`(cor^{coarse}, rhs^{coarse}, $\ell - 1$)
  (d) Prolongate the Correction to the present level ℓ cells covering coarse level $\ell - 1$ cells and one layer of ''virtual'' level ℓ cells.
    $sol = sol + I(cor)$
  (e) Postsmoothing Step
  for $i = 1$ to postsmooth do
    Smooth using ILU on level ℓ
  end for
end if

MGPCG-AMR: Real and Fictitious Cells



Coarse and fine grid levels depicting real and fictitious cells

MGPCG-AMR: Restriction, Prolongation, Smoother

Restriction (real cells):

$$r^{\ell}_{i_c,j_c} = r^{ell+1}_{i_f,j_f} + r^{ell+1}_{i_f+1,j_f} + r^{ell+1}_{i_f,j_f+1} + r^{ell+1}_{i_f+1,j_f+1}$$

Restriction (fictitious cells):

$$r^{\ell}_{i_c,j_c} = r^{ell+1}_{i_f,j_f}$$

Prolongation (real cells):

$$p^{\ell+1}_{i_f,j_f} = p^{\ell}_{i_c,j_c} \qquad p^{\ell+1}_{i_f+1,j_f} = p^{\ell}_{i_c,j_c} \qquad p^{\ell+1}_{i_f,j_f+1} = p^{\ell}_{i_c,j_c} \qquad p^{\ell+1}_{i_f+1,j_f+1} = p^{\ell}_{i_c,j_c}$$

Prolongation (fictitious cells):

$$p^{\ell+1}_{i_f,j_f} = p^{\ell}_{i_c,j_c}$$

Smoother (real cells):

$$p^{k+1} = p^k + M(b - Ap^k)$$

Smoother (fictitious cells):

$$p^{k+1} = p^k$$

Restriction Operator - Matrix Representation

$$\begin{bmatrix} 1 & 0 & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & 0 \\ 0 & 1 & & & & & & & \\ \vdots & & \ddots & & & & & & \\ \vdots & & & 1 & 1 & 1 & 1 & & \\ \vdots & & & & & & & \ddots & \\ 0 & & & & & & & & 1 \end{bmatrix} \begin{bmatrix} p_1^{\ell+1} \\ p_2^{\ell+1} \\ \vdots \\ p_{i1}^{\ell+1} \\ p_{i2}^{\ell+1} \\ p_{i3}^{\ell+1} \\ p_{i4}^{\ell+1} \\ \vdots \\ p_N^{\ell+1} \end{bmatrix} = \begin{bmatrix} p_1^{\ell} \\ p_2^{\ell} \\ \vdots \\ p_i^{\ell} \\ \vdots \\ p_N^{\ell} \end{bmatrix} \quad (3)$$

Prolongation Operator - Matrix representation

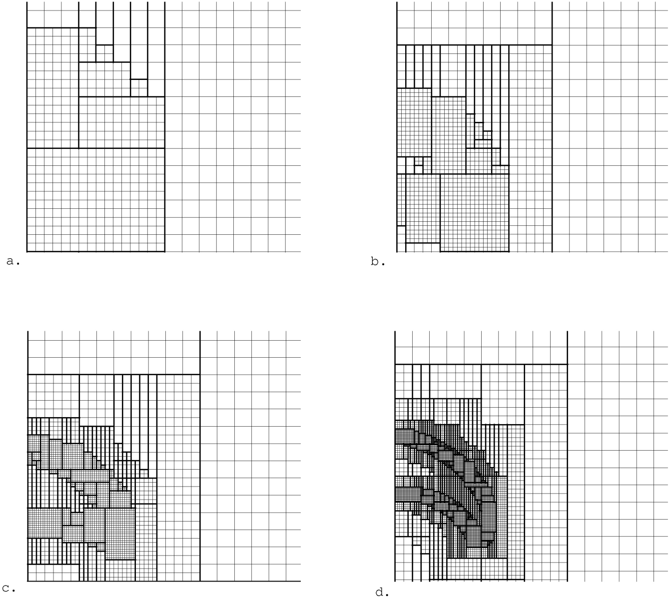
$$
\begin{bmatrix}
1 & 0 & \cdots & \cdots & \cdots & 0 \\
0 & 1 & & & & \\
\vdots & & \ddots & & & \\
\vdots & & & 1 & & \\
\vdots & & & 1 & & \\
\vdots & & & 1 & & \\
\vdots & & & 1 & & \\
\vdots & & & & \ddots & \\
0 & & & & & 1
\end{bmatrix}
\begin{bmatrix}
p_1^\ell \\ p_2^\ell \\ \vdots \\ p_i^\ell \\ \vdots \\ p_N^\ell
\end{bmatrix}
=
\begin{bmatrix}
p_1^{\ell+1} \\ p_2^{\ell+1} \\ \vdots \\ p_{i1}^{\ell+1} \\ p_{i2}^{\ell+1} \\ p_{i3}^{\ell+1} \\ p_{i4}^{\ell+1} \\ \vdots \\ p_N^{\ell+1}
\end{bmatrix}
\quad (4)
$$

Convergence

The convergence conditions are the same as those for MGPCG

- The MG smoother is symmetric
- The restriction operator is the transpose of the prolongation operator
- The matrix A in the smoothing step is symmetric

2D Test Problem



2D Test Problem Results

Blocking Factor	2			4			8		
Adaptive Levels	1	3	5	1	3	5	1	3	5
ILU Smoother									
PCG	0.659	5.424	63.49	0.563	3.359	35.54	0.365	2.875	26.78
MG	0.270	2.181	13.93	0.252	1.349	10.05	0.098	0.751	6.060
MGPCG	0.142	0.636	4.109	0.127	0.439	2.493	0.096	0.382	2.156
ICRB Smoother									
PCG	0.653	5.366	69.49	0.567	3.561	39.02	0.377	3.012	25.53
MG	0.281	2.177	16.54	0.278	1.435	10.96	0.112	0.885	6.634
MGPCG	0.157	0.655	4.511	0.152	0.498	2.732	0.123	0.415	2.402
GSRB Smoother									
PCG	0.641	5.706	65.99	0.567	4.037	37.72	0.364	3.014	29.26
MG	0.284	2.165	13.91	0.266	1.367	10.38	0.108	0.845	5.957
MGPCG	0.153	0.678	4.426	0.145	0.464	2.743	0.118	0.408	2.176

2D Test Problem Speedup

Blocking Factor	2			4			8		
Adaptive Levels	1	3	6	1	3	6	1	3	6
ILU	1.91X	3.43X	3.39X	1.99X	3.07X	4.03X	1.01X	1.97X	2.81X
ICRB	1.79X	3.32X	3.67X	1.83X	2.88X	4.01X	0.91X	2.13X	2.76X
GSRB	1.86X	3.19X	3.14X	1.84X	2.95X	3.78X	0.92X	2.07X	2.74X

3D Test Problem Results

Blocking Factor	2			4			8		
Adaptive Levels	1	2	3	1	2	3	1	2	3
ILU Smoother									
PCG	3.839	18.45	71.11	3.326	14.24	49.79	4.154	22.22	69.70
MG	2.156	9.432	40.46	1.704	6.140	24.02	2.146	7.410	21.68
MGPCG	1.884	6.295	19.05	1.747	5.220	14.63	1.999	6.007	17.53
ICRB Smoother									
PCG	4.158	19.67	75.61	3.682	15.28	56.82	4.770	27.19	84.94
MG	2.227	9.916	41.31	1.910	6.958	30.35	2.490	10.10	28.47
MGPCG	2.354	7.083	19.36	2.263	5.950	16.96	2.686	7.559	20.33
GSRB Smoother									
PCG	4.023	19.99	78.66	3.735	15.43	58.33	4.801	26.19	86.51
MG	2.145	9.708	41.06	1.860	7.086	31.98	2.485	10.10	29.68
MGPCG	2.331	7.149	20.20	2.241	6.191	17.67	2.571	8.001	20.19

3D Test Problem Speedup

Blocking Factor	2			4			8		
Adaptive Levels	1	2	3	1	2	3	1	2	3
ILU	1.14X	1.50X	2.12X	0.98X	1.18X	1.64X	1.07X	1.23X	1.24X
ICRB	0.95X	1.40X	2.13X	0.84X	1.17X	1.79X	0.93X	1.34X	1.40X
GSRB	0.92X	1.36X	2.03X	0.83X	1.14X	1.81X	0.97X	1.26X	1.47X

3D Whale Problem Results

Blocking Factor	2			4			8		
Adaptive Levels	1	2	3	1	2	3	1	2	3
ILU Smoother									
PCG	32.33	126.5	468.1	33.69	122.0	528.6	47.32	198.3	759.9
MG	10.52	41.69	145.6	9.874	35.71	133.4	13.91	57.37	202.2
MGPCG	5.445	15.06	43.66	5.796	14.84	48.74	7.113	21.28	70.07
ICRB Smoother									
PCG	40.15	171.9	636.1	42.80	160.6	705.6	68.93	269.2	1056
MG	14.93	64.01	226.1	14.61	61.08	226.1	24.02	102.5	371.9
MGPCG	7.606	19.93	56.83	8.794	19.45	62.45	10.49	28.93	94.08
GSRB Smoother									
PCG	41.07	179.2	648.8	43.11	167.0	710.2	70.00	276.3	1078
MG	15.99	67.29	230.4	15.42	63.24	234.9	26.03	106.8	395.9
MGPCG	8.063	22.38	57.47	8.451	21.61	65.76	10.39	31.77	94.70

3D Whale Problem Speedup

Blocking Factor	2			4			8		
Adaptive Levels	1	2	3	1	2	3	1	2	3
ILU	1.93X	2.77X	3.33X	1.70X	2.41X	2.74X	1.96X	2.70X	2.89X
ICRB	1.96X	3.21X	3.98X	1.66X	3.14X	3.62X	2.29X	3.54X	3.95X
GSRB	1.98X	3.01X	4.01X	1.82X	2.93X	3.57X	2.51X	3.36X	4.18X

With recent advances in GPU technology, parallel CFD codes are able to be accelerated on distributed hybrid architectures with multiple cores sharing a single GPU. We have done this for the NASA FUN3D code.

A.C. Duffy, D.P. Hammond, E.J. Nielsen. Production level CFD code acceleration for hybrid many-core architectures. *Submitted to Parallel Computing*, Under Revision.

GPU Advancements

Architecture	Cores	L1 Cache	L2 Cache	Memory Access Speed
G80	112	16 KB [1]	128 KB [2]	57.6 GB/s GDDR3
GT200	240	24 KB [1]	256 KB [2]	102 GB/s GDDR3
Fermi	448	48/16 KB [3]	768 KB	144 GB/s GDDR5

GPU architecture evolution from G80, which approximately coincided with the release of Intel's quad core CPUs, to Fermi which coincided with the release of Intel's six core processors. GPU advancements over the last few years have noticeably outpaced those of CPUs. Representative GPUs are: G80-GeForce 8800 GT, GT200-Tesla C1060, Fermi-Tesla C2050.
[1]shared memory, [2] texture memory, [3]Configurable L1/shared memory

GPU Accleration of the MG Smoothers

We have previously developed acclerated GSRB and ICRB smoothers using the PGI Fortran compiler with accelerator directives.

- The PGI compiler allows for simple code porting for GPUs
- Directives are similar to those of OpenMP, the accelerator code will be ignored if the compiler option is not used (e.g. when no accelerators or when not using the PGI compiler)
- Acts as a wrapper, code is converted to CUDA C

Jacobi method

- Simplest smoother, easy to port to GPU.
- Slow converging

Given x^0, $r = b - Ax^0$, $x = x^0$
$x = x + D^{-1}r$
$r = b - Ax$

Jacobi Code

PGI Fortran Code for Simple 1-D problem

```
!$acc region
do iterates=1,maxiterates
    do i=is,ie
       x(i)=(b(i)-L(i-1)x_old(i-1)-U(i+1)x_old(i+1))/DE(i)
    end do
    do i=is,ie
       x_old(i)=x(i)
     end do
end do
!$acc end region
```

Symmetric-Gauss-Seidel

- Symmetric Gauss Seidel smoothers lead to faster convergence compared to Jacobi smoothers, but are not vectorizable in their natural form.
- A Red-Black ordering allows Gauss Seidel to be vectorized, simple ex.

$$\left|\begin{array}{cccc} 2 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 2 \end{array}\right| \left|\begin{array}{c} x_1 \\ x_2 \\ x_3 \\ x_4 \end{array}\right| \rightarrow \left|\begin{array}{cccc} 2 & 0 & -1 & 0 \\ 0 & 2 & -1 & -1 \\ -1 & -1 & 2 & 0 \\ 0 & -1 & 0 & 2 \end{array}\right| \left|\begin{array}{c} x_1 \\ x_3 \\ x_2 \\ x_4 \end{array}\right|$$

Symmetric Red-Black GS

$$\begin{vmatrix} D^R & C^T \\ C & D^B \end{vmatrix} \begin{vmatrix} x_R \\ x_B \end{vmatrix} = \begin{vmatrix} r_R \\ r_B \end{vmatrix}$$

$$x_R^* = (D^R)^{-1} r_R$$

$$x_B = (D^B)^{-1}(r_B - C x_R^*)$$

$$x_R = (D^R)^{-1}(r_R - C^T x_B)$$

Symmetric Red-Black GS (cont)

$$M = \begin{vmatrix} (D^R)^{-1} + (D^R)^{-1} C^T (D^B)^{-1} C (D^R)^{-1} & -(D^R)^{-1} C^T (D^B)^{-1} \\ -(D^B)^{-1} C (D^R)^{-1} & (D^B)^{-1} \end{vmatrix}$$

$$x^{n+1} = x^n + M(r - Ax^n)$$

1. $r^* = r - Ax^n$
2. $x^{n+1} = x^n + Mr^*$

MG Smoothers - Incomplete Cholesky (IC)

IC factorizations are often used as the preconditioner themselves, but here we use IC as a smoother for multigrid.

- Fastest MG smoother
- Factorization ensures M maintains same sparse structure as A
- The standard IC preconditioner cannot be vectorized.

MG Smoothers - ICRB

We can again use a Red-Black ordering just as in the GSRB case here to vectorize the IC algorithm following the method of Ortega*, then the algorithm is the same as for the GSRB case with D^B replaced by $(D^B)^*$.

$$\begin{vmatrix} D^R & C^T \\ C & D^B \end{vmatrix} \rightarrow \begin{vmatrix} I & 0 \\ C(D^R)^{-1} & I \end{vmatrix} \begin{Vmatrix} D^R & 0 \\ 0 & (D^B)^* \end{Vmatrix} \begin{vmatrix} I & (D^R)^{-1}C^T \\ 0 & I \end{vmatrix}$$

$$(D^B)^* = diagonal(D^B - C(D^R)^{-1}C^T)$$

*James Ortega, "Introduction to Parallel and Vector Solution of Linear Systems", Springer, 1988.

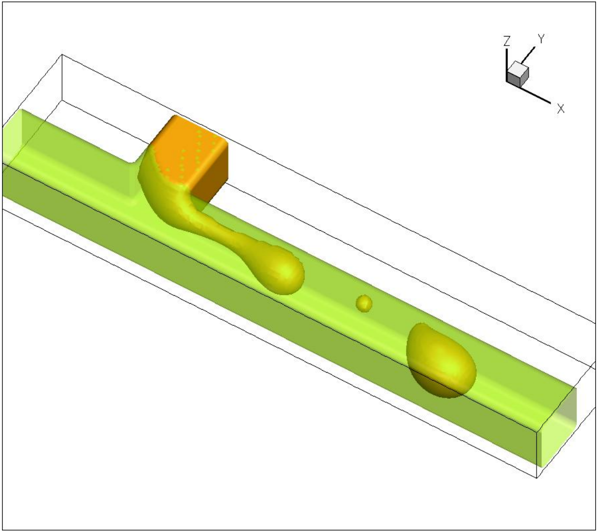
GPU Acceleration Results and Discussion

- Using the PGI Fortran compiler, the smoothers are limited to a 2X speedup due to costly data transfer overhead. Switching to a CUDA C implementation will allow us to store the A matrix coefficients permanently on the GPU, and should provide for more substantial speedup.

- An iterative refinement technique has been employed to reduce the residual error by 18 orders of magnitude in single precision, which is optimal for GPU acceleration.

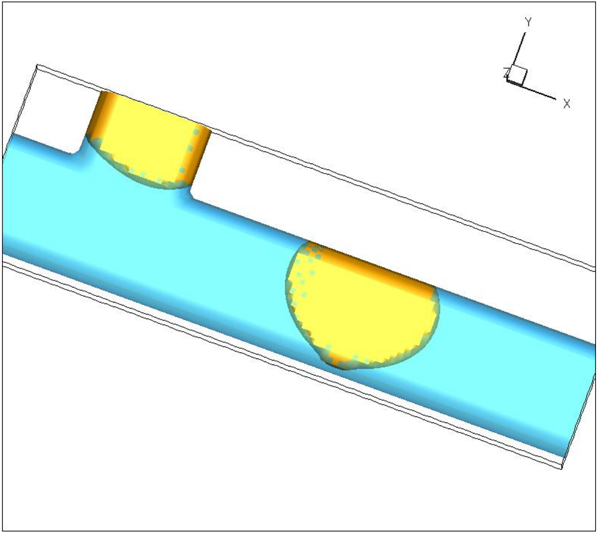
- We plan to develop the new CUDA C implementation using a GPU sharing model such as was done on the NASA FUN3D code. This will allow the code to be accelerated by a multicore processor (current capability) and a GPU simultaneously

Microfluidic T-Junction



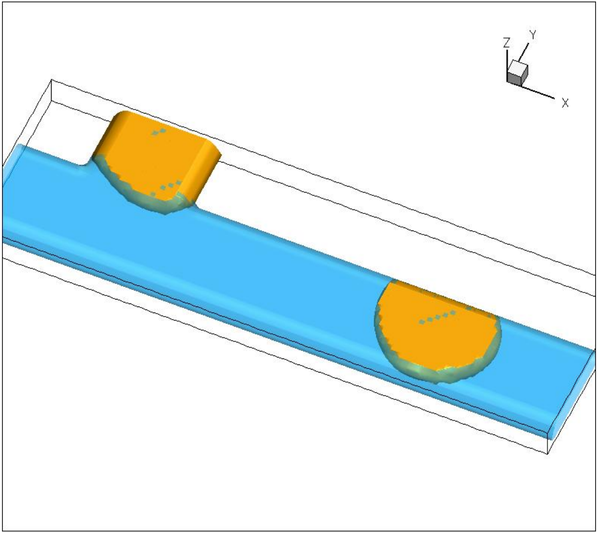
Microfluidic T-Junction - COMSOL/4 Cores

Microfluidic T-Junction



Simulation using data from Roper's Lab

Microfluidic T-Junction



Simulation using data from Roper's Lab

QUESTIONS?