Solving parabolic PDEs in parallel via time discretization using Laplace transforms

Kyle Geyser and Dylan Denning

December 3, 2012

The Problem

$$u_t + Au = f(t), \quad \text{for } t > 0, \quad \text{with } u(0) = u_0$$

where A is the second-order differential operator (∇^2) and u_0 and $f(t)$ are given.

i.e. The heat equation...

The Transformed Problem

$$(zI + A)w(z) = u_0 + \hat{f}(z)$$

where z is the transform variable, I is the identity matrix, $\hat{f}(z) = \mathcal{L}\{f\}$, and $w(z) = \mathcal{L}\{u\}$.

Note: all functions above have implied spatial terms based on the dimensionality of the problem

We find an approximation for $w(z)$ using the finite difference method with $m-1$ interior points, m a chosen parameter. In one spacial dimension, A is the $m-1 \times m-1$ discrete Laplacian given by:

$$A = \begin{bmatrix} 2 & -1 & 0 & 0 & \cdots & \cdots & \cdots \\ -1 & 2 & -1 & 0 & 0 & \cdots & \cdots \\ 0 & -1 & 2 & -1 & 0 & 0 & \cdots \\ \vdots & & & \ddots & & & \vdots \\ \vdots & & & & \ddots & & \vdots \\ 0 & 0 & \cdots & \cdots & -1 & 2 & -1 \\ 0 & 0 & \cdots & \cdots & \cdots & -1 & 2 \end{bmatrix}$$

The tridiagonal system we solve becomes

$$\frac{1}{h^2}(zI + A)w(z) = g(t)$$

where

$$g(t) = \vec{u_0} + \vec{\hat{f}}(z) + \frac{1}{h^2}\begin{bmatrix}\alpha \\ 0 \\ 0 \\ \vdots \\ \beta\end{bmatrix}$$

with

$$h = \frac{b-a}{m},\ a \text{ and } b \text{ the endpoints of our domain, } \alpha = u(a),\ \beta = u(b),$$

and $\vec{u_0} + \vec{\hat{f}}(z)$ vectors of size $m-1$ containing u_0 and $\hat{f}(z)$ evaluated at $x_i = a + ih, \quad i = 1, 2, ..., m-1.$

The Solution: time

Using a trapezoidal rule:

$$u(t) \approx U_{N,\tau}(t) = 2\mathsf{Re}\left(\frac{1}{N\tau}\sum_{j=0}^{N-1}{}'\tilde{\mu}_j \mathsf{e}^{z_j t} w(z_j)\right)$$

where z_j are the qudrature points on the transformed contour,
$\tilde{\mu_j}$ are the weights associated with each z_j,
τ is a time scaling parameter,
and the $'$ denotes halving the first term in the sum.

The first term is halved and the sum from 0 to $N-1$ is doubled because of the even symmetry of the contour.

Parallel Approach

We parallelized the sum from the previous slide using MPI.

Using this approach, each core, except the first and last cores, computes a local sum corresponding to $\frac{N-1}{P}$ quadrature points.

The first core computes the first term in the full sum, and adds to that a local sum corresponding to $\frac{N-1}{P}$ quadrature points.

The last core computes a local sum corresponding to $\frac{N-1}{P} + \text{Mod}(N-1, P)$ quadrature points.

We then use `MPI_Reduce` with the `master` core to reduce the local sums into a full sum.

Test PDE

$$u_t - u_{xx} = f(x,t), \quad \text{for } 0 < x < \pi,\ t > 0,$$

$$u(x,t) = 0 \quad \text{for } x = 0 \text{ and } \pi,\ t > 0,$$

with

$$u(x,0) = u_0(x), \quad \text{for } 0 < x < \pi$$

and

$$f(x,t) = \mathrm{e}^{-t}\sin(x) + \mathrm{e}^{-2t}(2cos(t) - \sin(t))\sin(2x),$$

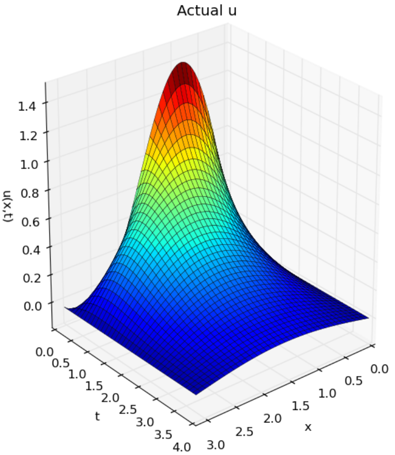
$$u_0(x) = \sin(x) + \sin(2x),$$

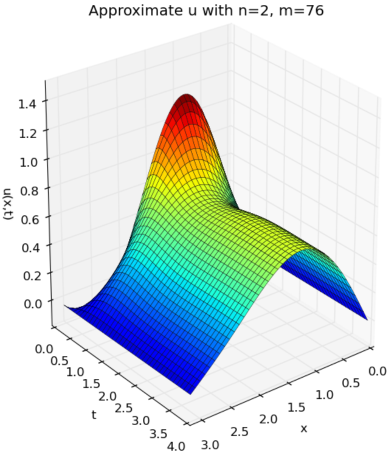
$$\hat{f}(x,z) = \frac{1}{1+z}\sin(x) + \frac{2z+3}{(z+2)^2+1}\sin(2x)$$

Actual Solution

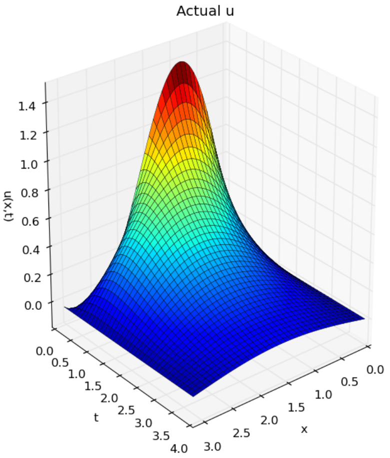
$$u(x,t) = (1+t)\mathsf{e}^{-t}\sin(x) + \cos(t)\mathsf{e}^{-2t}\sin(2x)$$

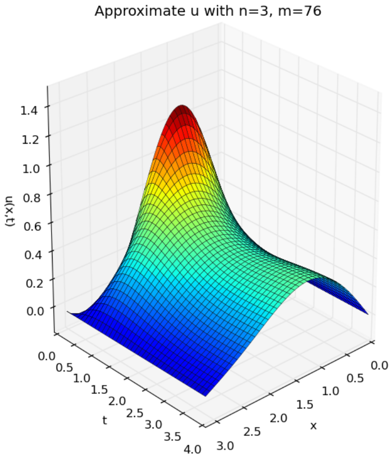
Results


Actual u
u(x,t)
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
t
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
x
0.0
0.5
1.0
1.5
2.0
2.5
3.0

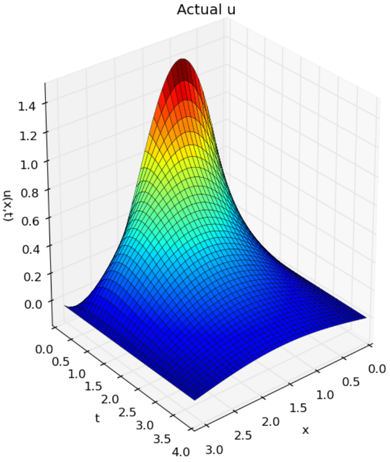

Approximate u with n=2, m=76
u(x,t)
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
t
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
x
0.0
0.5
1.0
1.5
2.0
2.5
3.0

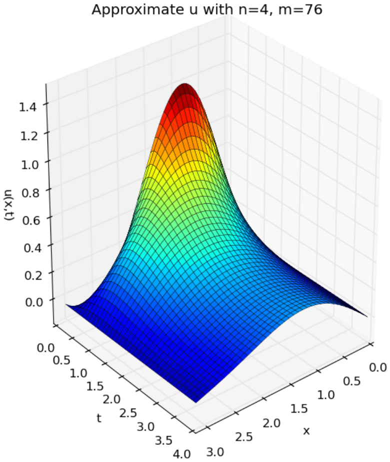
Results


Actual u
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
u(x,t)
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
t
0.0
0.5
1.0
1.5
2.0
2.5
3.0
x

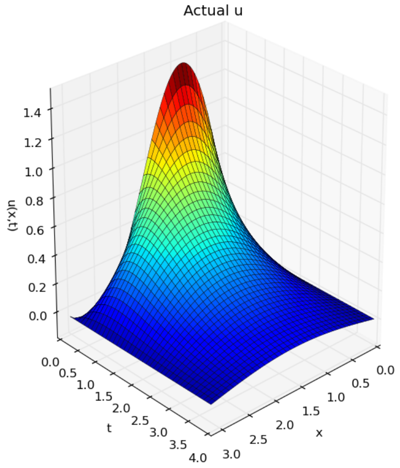

Approximate u with n=3, m=76
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
u(x,t)
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
t
0.0
0.5
1.0
1.5
2.0
2.5
3.0
x

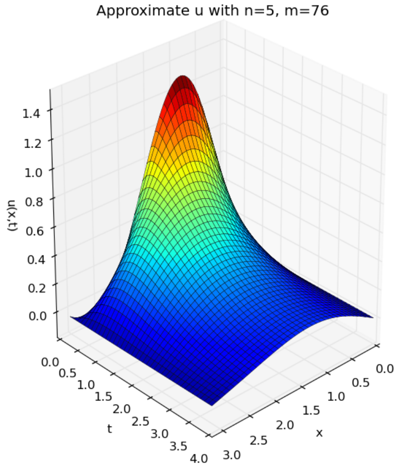
Results


Actual u
u(x,t)
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
t
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
x
0.0
0.5
1.0
1.5
2.0
2.5
3.0

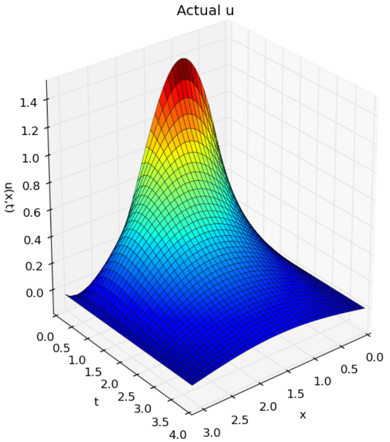

Approximate u with n=4, m=76
u(x,t)
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
t
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
x
0.0
0.5
1.0
1.5
2.0
2.5
3.0

Results

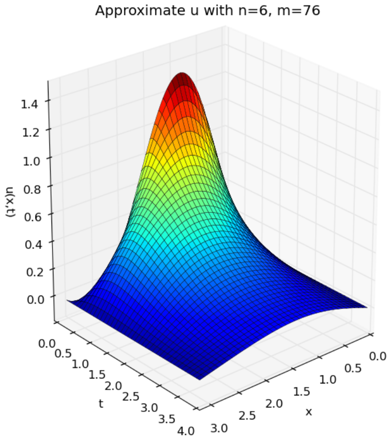

Actual u
u(x,t)
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
t
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
x
0.0
0.5
1.0
1.5
2.0
2.5
3.0


Approximate u with n=5, m=76
u(x,t)
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
t
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
x
0.0
0.5
1.0
1.5
2.0
2.5
3.0

Results

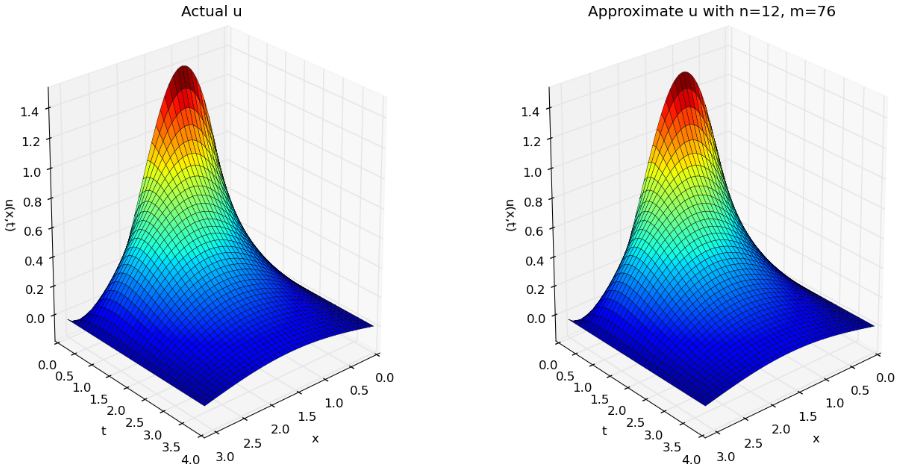


Actual u
u(x,t)
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
t
0.0 0.5 1.0 1.5 2.0 2.5 3.0 3.5 4.0
x
3.0 2.5 2.0 1.5 1.0 0.5 0.0



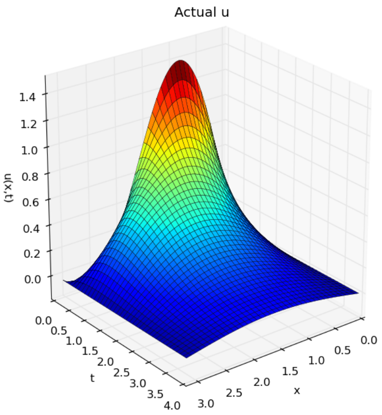
Approximate u with n=6, m=76
u(x,t)
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
t
0.0 0.5 1.0 1.5 2.0 2.5 3.0 3.5 4.0
x
3.0 2.5 2.0 1.5 1.0 0.5 0.0

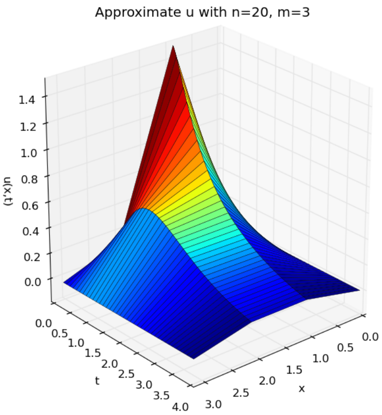
Results



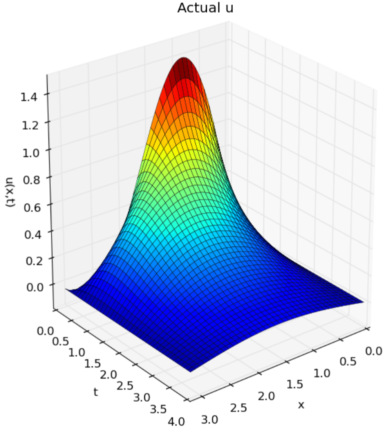
Actual u
u(x,t)
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
t
0.0 0.5 1.0 1.5 2.0 2.5 3.0 3.5 4.0
x
0.0 0.5 1.0 1.5 2.0 2.5 3.0
Approximate u with n=12, m=76
u(x,t)
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
t
0.0 0.5 1.0 1.5 2.0 2.5 3.0 3.5 4.0
x
0.0 0.5 1.0 1.5 2.0 2.5 3.0

Results

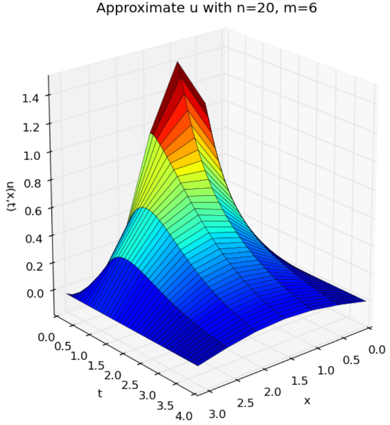

Actual u
u(x,t)
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
t
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
x
3.0
2.5
2.0
1.5
1.0
0.5
0.0


Approximate u with n=20, m=3
u(x,t)
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
t
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
x
3.0
2.5
2.0
1.5
1.0
0.5
0.0

Results

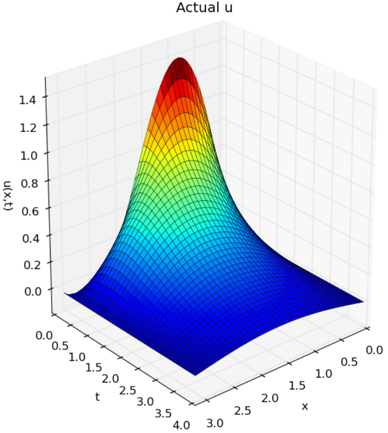


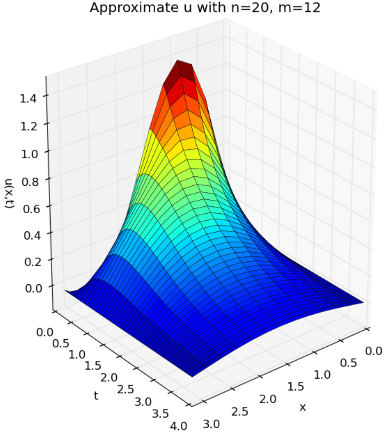
Actual u
u(x,t)
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
t
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
x
3.0
2.5
2.0
1.5
1.0
0.5
0.0



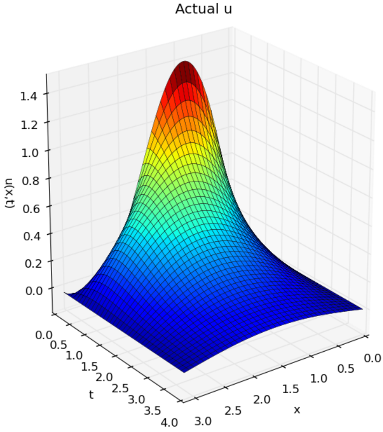
Approximate u with n=20, m=6
u(x,t)
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
t
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
x
3.0
2.5
2.0
1.5
1.0
0.5
0.0

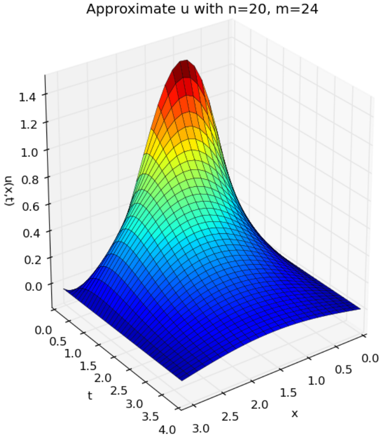
Results


Actual u
u(x,t)
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
t
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
x
3.0
2.5
2.0
1.5
1.0
0.5
0.0

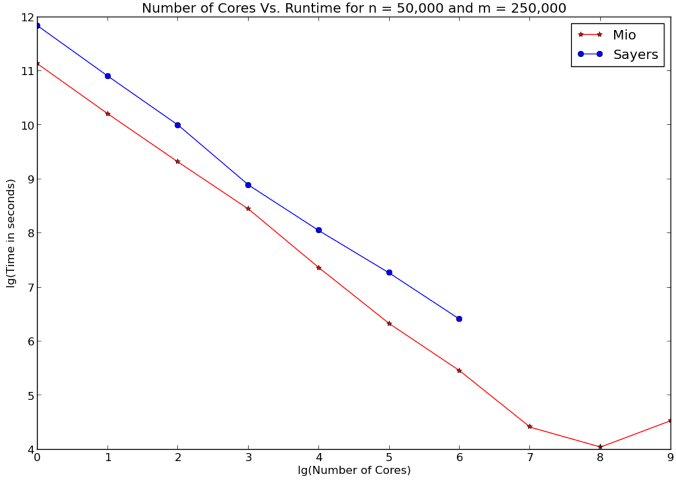

Approximate u with n=20, m=12
u(x,t)
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
t
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
x
3.0
2.5
2.0
1.5
1.0
0.5
0.0

Results


Actual u
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
u(x,t)
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
t
3.0
2.5
2.0
1.5
1.0
0.5
0.0
x

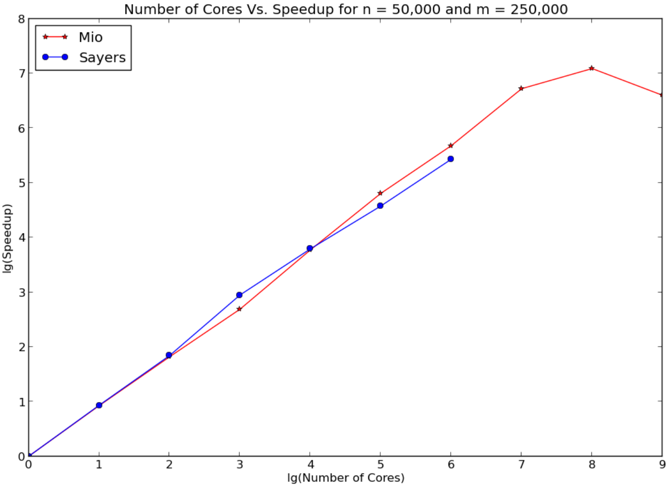

Approximate u with n=20, m=24
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
u(x,t)
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
t
3.0
2.5
2.0
1.5
1.0
0.5
0.0
x

Results



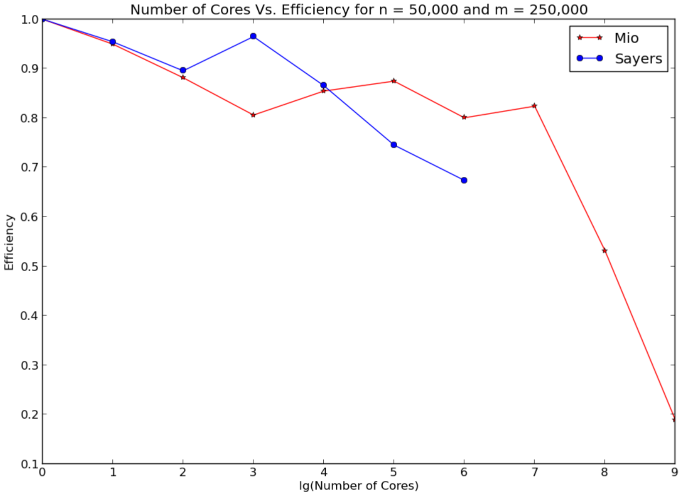
Number of Cores Vs. Runtime for n = 50,000 and m = 250,000
Mio
Sayers
lg(Time in seconds)
lg(Number of Cores)
12
11
10
9
8
7
6
5
4
0
1
2
3
4
5
6
7
8
9

Results



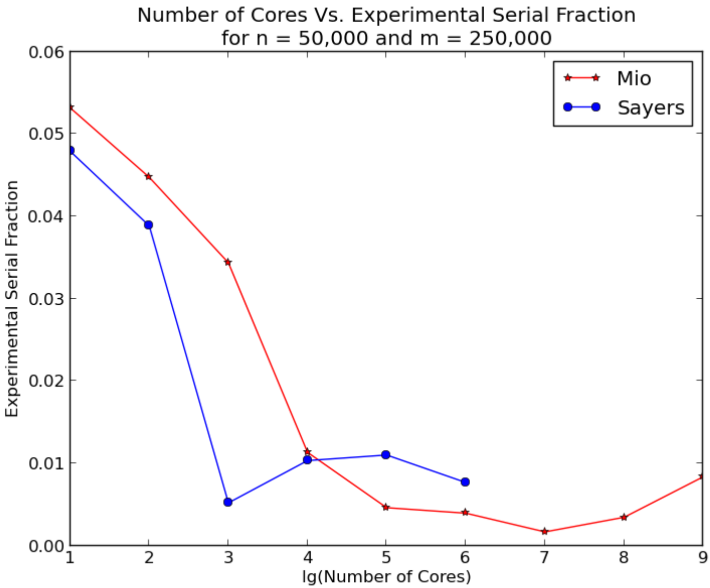
Number of Cores Vs. Speedup for n = 50,000 and m = 250,000
Mio
Sayers
lg(Speedup)
lg(Number of Cores)
0
1
2
3
4
5
6
7
8
9

Results



Number of Cores Vs. Efficiency for n = 50,000 and m = 250,000
Mio
Sayers
Efficiency
lg(Number of Cores)
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
1
2
3
4
5
6
7
8
9

Results



Number of Cores Vs. Experimental Serial Fraction
for n = 50,000 and m = 250,000
Mio
Sayers
Experimental Serial Fraction
lg(Number of Cores)
0.06
0.05
0.04
0.03
0.02
0.01
0.00
1
2
3
4
5
6
7
8
9

References

Dongwoo Sheen, Ian H. Sloan, and Vidar Thomée, *A parallel method for time discretization of parabolic equations based on Laplace transformation and quadrature*. IMA Journal of Numerical Analysis (2003) 23, 269-299

Mahadevan Ganesh, PhD, Colorado School of Mines