Solving PDEs with PGI CUDA Fortran
Part 5: Explicit methods for evolutionary partial differential equations

Outline

Heat equation in one, two and three dimensions. Discretization stencils. Block and tiling implementations. Method of lines.

Heat equation

temporal evolution (physically, diffusion) of heat (temperature) in a domain
a partial differential equation (1-st order in time t, 2-nd order in spatial variables X)
for a function u(t, X)
1D (one-dimensional) case: X = x, 2D case: X = x,y, 3D case: X = x,y,z

General form: $$\partial_t u(t,X) = \Delta u(t,X)$$

in 3D: $$\partial_t u(t,x,y,z) = (\partial_x^2 + \partial_y^2 + \partial_z^2) u(t,x,y,z)$$

Initial condition: $$u(t_0,X) = u_0(X)$$

Boundary conditions: $u(t,X_B) = u_B(t,X_B)$ on the boundary

i.e., the initial value problem (IVP) for the parabolic partial differential equation

Heat equation

Discretization grids and schemes

the equidistant grid on a rectangular domain, constant time steps

$$
\begin{aligned}
t_n &= t_0 + n\,dt, & dt &= (t_N - t_0)/N \\
x_j &= x_0 + j\,dx, & dx &= (x_J - x_0)/J \\
y_k &= z_0 + k\,dy, & dy &= (y_K - y_0)/K \\
z_l &= z_0 + l\,dz, & dz &= (z_L - z_0)/L \\
u^n_{jkl} &\approx u(t_n, x_j, y_k, z_l)
\end{aligned}
$$

moreover,

$$J = K = L, \quad dx = dy = dz$$

Heat equation

Explicit FTCS scheme (forward-in-time, centered-in-space)

FD1 for time:

$$\partial_t u^n_{jkl} \approx (u^{n+1}_{jkl} - u^n_{jkl})/dt$$

(cf. Euler method for ODEs)

FD2 for space:

$$\partial^2_x u^n_{jkl} \approx (u^n_{j-1,k,l} - 2u^n_{jkl} + u^n_{j+1,k,l})/dx^2$$
$$\partial^2_y u^n_{jkl} \approx (u^n_{j,k-1,l} - 2u^n_{jkl} + u^n_{j,k+1,l})/dy^2$$
$$\partial^2_z u^n_{jkl} \approx (u^n_{j,k,l-1} - 2u^n_{jkl} + u^n_{j,k,l+1})/dz^2$$

More spatial stencils:

FD4 $$\partial^2_x u^n_j \approx \tfrac{1}{12}(-u^n_{j-2} + 16u^n_{j-1} - 30u^n_j + 16u^n_{j+1} - u^n_{j+2})/dx^2$$

FD6 $$\partial^2_x u^n_j \approx \tfrac{1}{180}(2u^n_{j-3} - 27u^n_{j-2} + 270u^n_{j-1} - 490u^n_j + 270u^n_{j+1} - 27u^n_{j+2} + 2u^n_{j+3})/dx^2$$

Discretized heat equation in 1D

1D heat equation $u_j^{n+1} = (1-2\beta)u_j^n + \beta\left(u_{j-1}^n + u_{j+1}^n\right), \qquad \beta = dt/dx^2$

accuracy: 1st-order in time, 2-nd order in space

stability condition: $\beta \le 1/2, \quad dt \le dx^2/2, \quad N \ge 2J^2(t_N - t_0)/(x_J - x_0)^2$

The sinus example

domain $t_0 = 0,\ 0 \le x \le 1$

initial condition $u_0(x) = \sin(\pi x)$

boundary conditions constant and consistent with the initial condition

analytical solution $u(t,x) = e^{-\pi^2 t}\sin(\pi x)$

minimal number of timesteps to reach t = 1, according to the stability condition,
is $N = 2\ J^2$

Discretized heat equation in 1D

Equilibrium solution of the heat equation

In the equilibrium limit, $\partial_t u = 0$, the heat equation takes form of the Laplace's equation, i.e., long-time solutions of the heat equation converge to the solutions of the Laplace's equation.

Iterations

$$u_j^{n+1} = (1-2\beta)u_j^n + \beta\left(u_{j-1}^n + u_{j+1}^n\right)$$

are called the Jacobi iterations, as they, in the stability limit of $\beta = 1/2$, take form of

$$u_j^{n+1} = \left(u_{j-1}^n + u_{j+1}^n\right)/2,$$

that we have already called the Jacobi iterations for the 1D Laplace's equation.

Discretized heat equation in 2D

2D heat equation

$$u_{jk}^{n+1} = (1-4\beta)u_{jk}^{n} + \beta\left(u_{j-1,k}^{n} + u_{j+1,k}^{n} + u_{j,k-1}^{n} + u_{j,k+1}^{n}\right), \qquad \beta = dt/dx^2$$

the stability condition

$$\beta \le 1/4, \quad dt \le dx^2/4, \quad N \ge 4J^2(t_N - t_0)/(x_J - x_0)^2$$

The 2D sinus example

domain $t_0 = 0,\ 0 \le x \le 1,\ 0 \le y \le 1$

initial condition $u_0(x,y) = \sin(\pi x)\sin(\pi y)$

boundary conditions constant and consistent with the initial condition

analytical solution

$$u(t,x,y) = e^{-2\pi^2 t}\sin(\pi x)\sin(\pi y)$$

minimal number of timesteps to reach t = 1, according to the stability condition, is $N = 4\,J^2$

GPU implementations of Jacobi iterations in 2D

Block approach

– the spatial domain is split into rectangular blocks (not necessarily squares)
– each block of grid points (with halo or ghost points on block boundaries)
  is assigned to 1 CUDA block
– each thread updates one grid point

Notes:
CUDA blocksize limit of 1024 threads/block corresponds to the number of grid points,
  i.e., max. 32x32 (32x16, 32x8, 64x8, ...)
smem limit of 48 KB/multiprocessor: 4+ KB for a SP array of 32x32 grid points
more work in a kernel:
  merging (e.g., 4) grid points for 1 thread
  using higher-order spatial discretization (FD4 etc.)
keeping CUDA blocks smaller makes better multiprocessor occupancy
  (up to 8 blocks/multiprocessor)
allows for implementation of wildly asynchronous kernels

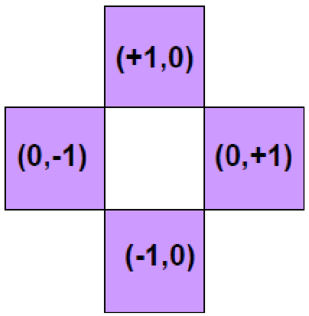
GPU implementations of Jacobi iterations in 2D

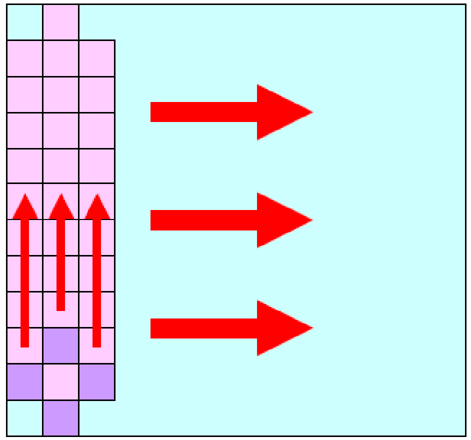
Tiling approach

– the spatial domain is split into rectangular strips
– each strip of grid points (with halos on strip boundaries) is assigned to 1 CUDA block
– each thread updates one line of grid points
– a 1D temporary shared-memory array (a tile, degenerated in 2D to an abscissa) moves along these lines together with two abscissas made from registers

Notes:
– CUDA blocksize ~ 64, 128, 256, e.g., for 1024^2 grid points and CUDA block size of 128, there is 8 CUDA blocks
– smem limit high enough
– well suited for FD4 etc.

$B(I \pm 1, J \pm 1)$





Discretized heat equation in 3D

3D heat equation

$$u^{n+1}_{jkl} = (1-6\beta)u^n_{jkl} + \beta\left(u^n_{j-1,kl} + u^n_{j+1,kl} + u^n_{j,k-1,l} + u^n_{j,k+1,l} + u^n_{j,k,l-1} + u^n_{j,k,l+1}\right),$$
$$\beta = dt/dx^2$$

the stability condition $\beta \le 1/6, \quad dt \le dx^2/6, \quad N \ge 6J^2(t_N - t_0)/(x_J - x_0)^2$

The 3D sinus example

domain $t_0 = 0,\ 0 \le x \le 1,\ 0 \le y \le 1,\ 0 \le z \le 1$

initial condition $u_0(x,y,z) = \sin(\pi x)\sin(\pi y)\sin(\pi z)$

boundary conditions constant and consistent with the initial condition

analytical solution $u(t,x,y,z) = e^{-3\pi^2 t}\sin(\pi x)\sin(\pi y)\sin(\pi z)$

minimal number of timesteps to reach t = 1, according to the stability condition, is $N = 6\,J^2$

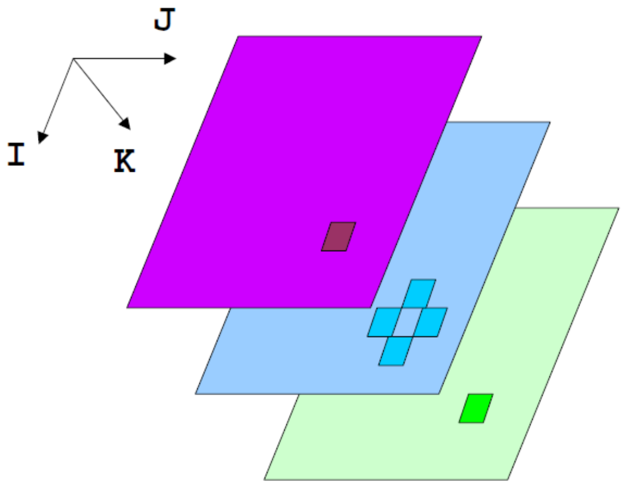
GPU implementations of Jacobi iterations in 3D

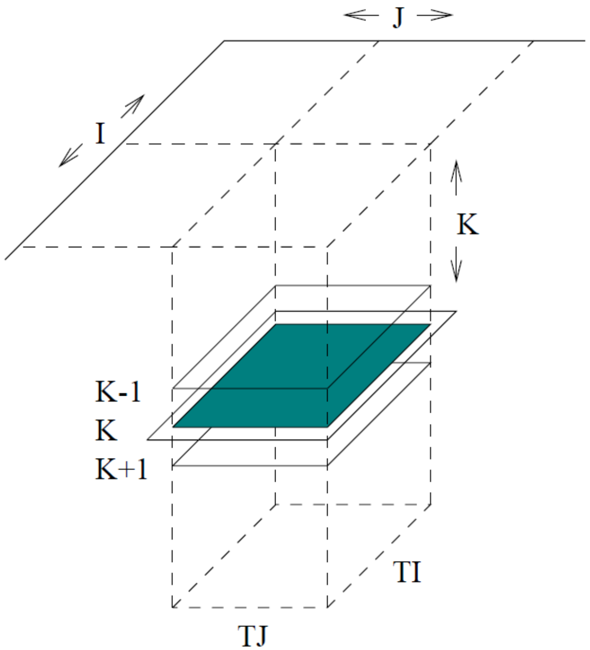
Block approach

size 3D blocks of grid points substantially limited by the CUDA blocksize limit of 1024 threads/block (e.tg., 16x8x8)

Tiling approach

– the spatial domain is split into rectangular columns
– each column (with halos on column boundaries) is assigned to 1 CUDA block
– each thread updates one line of grid points
– a 2D temporary shared-memory array (the tile) moves along these lines together with two tiles made from registers





Method of lines (MOL)

motivation: use ODEs techniques for time integration instead of explicit Euler method in the FTCS scheme

procedure: discretization of spatial variables but not the time variable, i.e., from PDEs to ODEs, and solving the ODEs with advanced solvers

Heat equation with Dirichlet boundary conditions

1D: $$\partial_t u_j(t) = \beta\,(u_{j-1} - 2u_j + u_{j+1})\,, \quad \beta = 1/dx^2, \quad j = 1,\ldots,J$$

$$\partial_t \begin{pmatrix} u_1(t) \\ u_2(t) \\ \cdot \\ u_{J-1}(t) \\ u_J(t) \end{pmatrix} = \beta \begin{pmatrix} -2 & 1 & & & \\ 1 & -2 & 1 & & \\ & \cdot & \cdot & \cdot & \\ & & 1 & -2 & 1 \\ & & & 1 & -2 \end{pmatrix} \begin{pmatrix} u_1(t) \\ u_2(t) \\ \cdot \\ u_{J-1}(t) \\ u_J(t) \end{pmatrix} + \beta \begin{pmatrix} u_0 \\ 0 \\ \cdot \\ 0 \\ u_{J+1} \end{pmatrix}, \quad \beta = 1/dx^2$$

2D:

$$\partial_t u_{jk}(t) = \beta\,(u_{j-1,k} + u_{j+1,k} + u_{j,k-1} + u_{j,k+1} - 4u_{jk})\,, \quad \beta = 1/dx^2, \quad j,k = 1,\ldots,J$$

etc.

Method of lines (MOL)

On GPU, the Jacobi iterations are required, both block or tiling approaches are possible.

The GPU/CPU speedup is the same as the speedup for Jacobi iterations in the FTCS case but we received the chance to converge faster than with the Euler method.

However, using implicit ODEs solvers should be considered.

Links and references

Numerical methods

Koev P., Numerical Methods for Partial Differential Equations, 2005
http://dspace.mit.edu/bitstream/handle/1721.1/56567/18-336Spring-2005/OcwWeb/Mathematics/18-336Spring-2005/CourseHome/index.htm

Press W. H. et al., Numerical Recipes in Fortran 77: The Art of Scientific Computing, Second Edition, Cambridge, 1992
Chapter 19.0: Introduction
Chapter 19.2: Diffusive initial value problems
Chapter 19.3: Initial value problems in multidimensions
Chapter 19.5: Relaxation methods for boundary value problems
http://www.nr.com, PDFs available at http://www.nrbook.com/a/bookfpdf.php

Spiegelman M., Myths and Methods in Modelling, 2000
http://www.ldeo.columbia.edu/~mspieg/mmm/

Links and references

CUDA techniques

Micikevicius P., 3D finite difference computation on GPUs using CUDA, 2009
Rivera G. and Tseng Ch.-W., Tiling optimizations for 3D scientific computations, 2000
Venkatasubramanian S. and Vuduc R. W., Tuned and wildly asynchronous stencil kernels for hybrid CPU/GPU systems, 2009
Xu Ch. et al., Tiling for performance tuning on different models of GPUs, 2009