Toward Efficient Parallel in Time Methods for PDEs

M.L. Minion and M.W. Emmett

THE UNIVERSITY of NORTH CAROLINA at CHAPEL HILL

Motivation

The *Parallel Full Approximation Scheme in Space and Time* (PFASST) algorithm is a novel approach for parallelizing PDEs in time. In practice, temporal parallelization is only attractive if the temporal parallelization has greater parallel efficiency than (further) spatial parallelization. Hence, the focus of this research is on constructing time-parallel methods with good parallel efficiency. To efficiently parallelize PDEs in time, the PFASST algorithm decomposes the time domain into several time slices. After a provisional solution is obtained using a relatively inexpensive time integration scheme, the solution is iteratively improved using a deferred correction scheme. To further improve parallel efficiency, the PFASST algorithm uses a hierarchy of discretizations at different spatial and temporal resolutions and employs an analog of the multi-grid full approximation scheme to transfer information between the discretizations.

PFASST

The PFASST algorithm can be described in terms of two propagators, denoted here by $\mathcal{F}$ (fine) and $\mathcal{G}$ (coarse). Both $\mathcal{F}$ and $\mathcal{G}$ propagate an initial value by approximating the solution to

$$u(t) = u_0 + \int_0^t f(\tau, u(\tau))\, d\tau$$

from t_n to t_{n+1}.

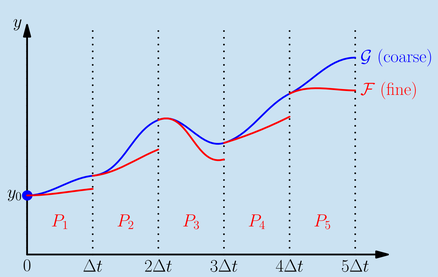
PFASST uses the Spectral Deferred Correction (SDC) scheme for both the $\mathcal{F}$ and $\mathcal{G}$ propagators. A single time step $[t_n, t_{n+1}]$ is divided into a set of intermediate sub-steps by defining intermediate points $t_m \in [t_n, t_{n+1}]$. The intermediate points are typically chosen to correspond to Gauss-Lobatto or Gauss-Radau quadrature points. SDC then constructs higher-order accurate solutions within one time step by iteratively approximating a series of correction equations at the intermediate nodes using lower-order methods. For example, an explicit SDC scheme based on the forward Euler discretization is

$$y_{m+1}^{k+1} = y_m^k + \Delta t_m \left[f(t_m, y_m^{k+1}) - f(t_m, y_m^k) \right] + I_m^{m+1}$$

where

$$I_m^{m+1} \approx \int_{t_m}^{t_{m+1}} f(\tau, y^k(\tau))\, d\tau.$$

The time interval of interest $[0, T]$ is divided into N uniform intervals $[t_n, t_{n+1}]$ which are assigned to the processors P_n where $n = 1 \ldots N$. A provisional solution is obtained using the $\mathcal{G}$ propagator in serial. This solution is then improved by alternately applying the $\mathcal{F}$ propagator in parallel and the $\mathcal{G}$ propagator in serial.



Speed up and efficiency

For PFASST to be efficient, the $\mathcal{G}$ propagator must be less computationally expensive than the $\mathcal{F}$ propagator. To achieve this, the $\mathcal{G}$ propagator is typically defined on coarse temporal and spatial grids. The multi-grid *Full Approximation Scheme* is used to transfer information between the discretizations.

If the PFASST iterations converge to the required accuracy in K_p iterations, the total cost on N processors is

$$C_p = N\Upsilon_G + K_p(\Upsilon_F + \Upsilon_G + \Upsilon_O).$$

where Υ_G is the cost of $\mathcal{G}$, Υ_F is the cost of $\mathcal{F}$, and Υ_O is the overhead associated with communication, interpolation, restriction etc. Compared to a serial SDC method of similar accuracy that requires K_s iterations to converge (so that $C_s = NK_s\Upsilon_F$), the efficiency of the PFASST algorithm is

$$E = \left[\frac{N\alpha}{K_s} + \frac{K_p}{K_s}(1 + \alpha + \beta) \right]^{-1}$$

where $\alpha = \Upsilon_G/\Upsilon_F$ and $\beta = \Upsilon_O/\Upsilon_F$.

Acknowledgments

This work was supported by the Director, DOE Office of Science, Office of Advanced Scientific Computing Research, Office of Mathematics, Information, and Computational Sciences, Applied Mathematical Sciences Program, under contract DE-SC0004011.

This work is currently authored by Michael L. Minion (`minion@unc.edu`), and Matthew Emmett (`memmett@unc.edu`). For more information, please visit our website at

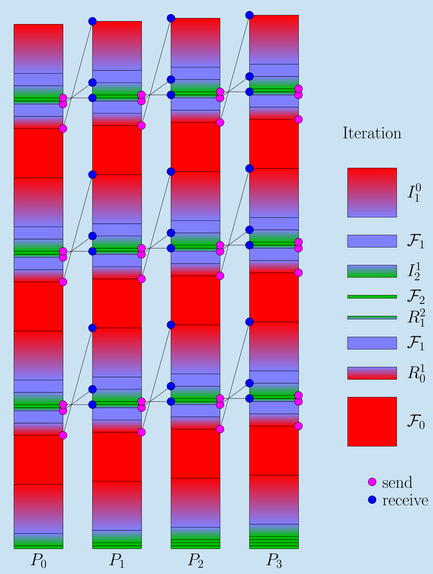
`http://www.unc.edu/~mwemmett/pfasst/`

Parallel SDC sweeps

The steps of the PFASST algorithm are depicted in the diagram below for the first 4 processors of a three-level V-cycle run. Each iteration is comprised of several pieces:

$\mathcal{F}_n$	SDC sweep on level n, with level 0 being the finest
R_n^{n+1}	time/space restriction from level n to level $n+1$
I_{n+1}^n	time/space interpolation from level $n+1$ to level n

The coarse sweeps at the bottom of diagram correspond to the computation of the provisional solution at the beginning of the algorithm. The height of the boxes representing sweeps, interpolations, and restrictions is proportional to how many function evaluations must be performed.



The finest level is red, the middle level is blue, and the coarsest level is green. Gradients depict restriction or interpolation.

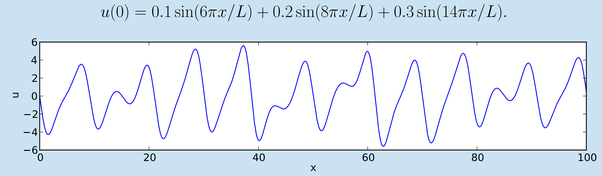
Kuromoto-Silvashinsky

The Kuramoto-Silvashinsky (KS) equation is

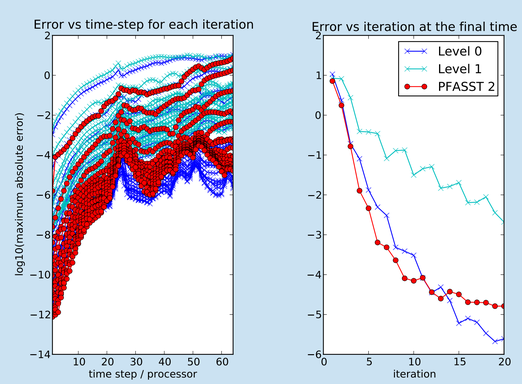
$$u_t + \tfrac{1}{2}|\nabla u|^2 + \nabla^2 u + \nabla^4 u = 0 \quad \text{or in 1d} \quad u_t + uu_x + u_{xx} + u_{xxxx} = 0.$$

It arises as a model for interfacial instabilities in a variety of physical contexts, and has been shown to exhibit nontrivial dynamical behavior, both spatially and temporally, including chaos. We consider a 1d periodic domain with initial conditions

$$u(0) = 0.1\sin(6\pi x/L) + 0.2\sin(8\pi x/L) + 0.3\sin(14\pi x/L).$$



Solution of the 1d KS equation at time $t = 64$.



Convergence of the PFASST algorithm for the 1d KS equation.

Vorticity

The vorticity formulation of the 2D incompressible Navier-Stokes equation is

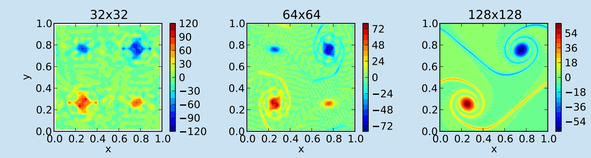
$$\partial_t \omega + \boldsymbol{u} \cdot \nabla \omega = \nu \nabla^2 \omega$$

where ω is the vorticity, $\boldsymbol{u} = (u, v)$ is the velocity, and ν is the viscosity. We consider a doubly periodic domain $[0, 1] \times [0, 1]$ with initial conditions

$$u_0(x, y) = -1.0 + \tanh\big(\rho(0.75 - y)\big) + \tanh\big(\rho(y - 0.25)\big)$$
$$v_0(x, y) = -\delta \sin\big(2\pi(x + 0.25)\big)$$

where $\rho = 100$ and $\delta = 0.05$. These conditions correspond to two very thin horizontal shear layers at $y = 0.5 \pm 0.25$ with a slight disturbance in the vertical velocity.

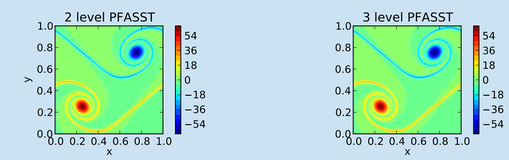
For small grids the solution is severely under-resolved. This is demonstrated by computing the solution to time $t = 1$ using a pseudo-spectral spatial method and an SDC time stepper for various grid sizes.



Serial solutions for various spatial discretizations.

The lower resolution runs exhibit spurious oscillations throughout the domain and have two spurious vorticies. These runs are clearly under-resolved. The high resolution run does not have spurious vorticies and appears to be fairly well behaved.

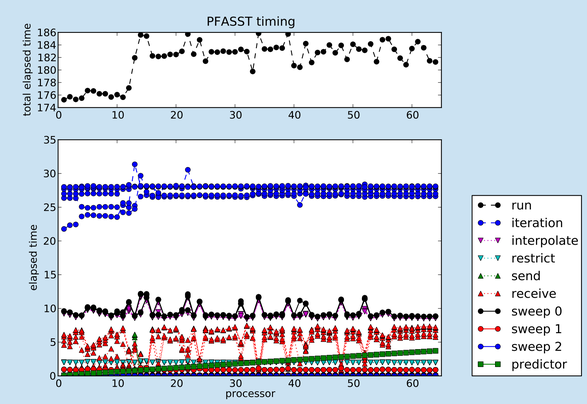
Despite the under-resolution of the coarser levels, the multi-level PFASST solutions are essentially the same as the high-resolution serial run.



PFASST solutions for various number of levels.

Timing

The timing diagram below was generated from MPI timing information collected during a 3-level PFASST run on 64 processors for a three-dimensional PDE. Two SDC sweeps were performed for each coarse (level 2) sweep. Six PFASST iterations are displayed.



The efficiency of the PFASST algorithm has been explored on various machines at UNC-CH. Typical parallel efficiencies range from 40–70% on runs with 4 to 512 processors.

For example, the 2d KS equation was run to a final time of $t \approx 166$ with a time step of $\Delta t = 1.0/256$. The PFASST levels used were

- fine ($\mathcal{F}$): 512x512 spatial grid, 5 Gauss-Lobatto SDC nodes; and
- coarse ($\mathcal{G}$): 256x256 spatial grid, 3 Gauss-Lobatto SDC nodes.

The parallel run was performed using two coarse SDC sweeps per PFASST iteration, and 6 PFASST iterations per time-step. The serial run was performed on the fine level with 10 SDC sweeps per time step.

With 128 processors PFASST acheived a speedup of 72 (56% efficient).