Università della Svizzera italiana
Faculty of Informatics
Institute of Computational Science ICS

Parallel-in-time integration with PFASST

Swiss Numerical Analysis Day 2015

Daniel Ruprecht, Robert Speck, Rolf Krause

in collaboration with M. Emmett, M. Minion, M. Bolten and others

Institute of Computational Science
Università della Svizzera italiana
Lugano, Switzerland

17 April 2015

Joint work with

Rolf Krause

Matthew Emmett
Michael Minion

Matthias Bolten

Robert Speck

A fundamental turn toward concurrency

- In about 10 years, **Exascale** systems will require 100-million way parallelism
- Multitude of challenges for numerical methods:
  1. massive concurrency
  2. resilience against faults
  3. energy consumption
  4. ...

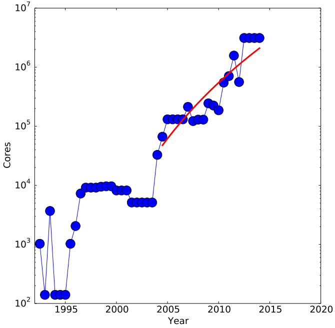


Figure: Number of cores in the top system in `www.top500.org`.

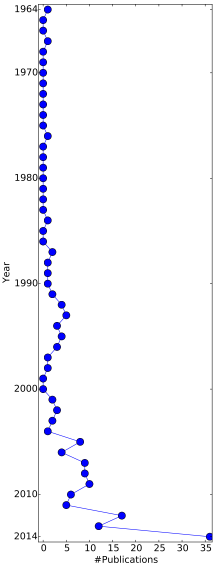
#8: Don't rethink your algorithms.
(K. Yelick, "Ten ways to waste a parallel computer", 2009)

51 years of parallel-in-time integration[1]

Parallelization in time can

1. extend strong scaling limits of space parallelization
2. help with the "trap of weak scaling"

- Interpolation-based approach (Nievergelt 1964)
- Parabolic or time multi-grid (Hackbusch 1984) and (Horton 1992)
- Parallel Runge-Kutta methods (e.g. Butcher 1997) and extrapolation (Richardson 1910)
- Parareal (Lions, Maday, Turinici 2001) and PITA (Farhat, Chandesris 2003)
- PFASST (Emmett, Minion 2012)
- MGRIT (Falgout et al. 2014)
- DG time multi-grid (Gander, Neumueller 2014)



[1] For a recent overview see M. Gander's paper.

Collocation

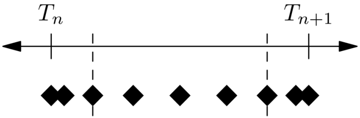


Figure: A time-step $[T_n, T_{n+1}]$ with $M = 9$ Gauss-Lobatto collocation nodes t_j.

Integrate initial value problem $u'(t) = f(u(t), t)$ from T_n to T_{n+1}:

- Consider Picard formulation of IVP

$$u(T_{n+1}) = u(T_n) + \int_{T_n}^{T_{n+1}} f(\tau, u(\tau))\ d\tau$$

- The integral is approximated by quadrature

$$\int_{T_n}^{T_{n+1}} f(\tau, u(\tau))\ d\tau \approx \Delta t \sum_{j=1}^{M} q_{n,j} f(t_j, u_j), \quad \text{with stages } u_j \in \mathbb{R}^N$$

- Collocation problem can compactly be written as

$$\mathbf{U} = \mathbf{U}_0 + \Delta t \mathbf{QF}(\mathbf{U}),\ \mathbf{U} = (u_1, \dots, u_M)^{\mathrm{T}} \in \mathbb{R}^{NM \times NM}$$

Richardson or discrete Picard iteration

- Rewrite the **collocation** equation as linear[1] equation

$$\mathbf{MU} = (\mathbf{I} - \Delta t\mathbf{QF})\,\mathbf{U} = \mathbf{U}_0$$

- Richardson iteration corresponds to (discrete) **Picard iteration**

$$\mathbf{U}^{k+1} = \mathbf{U}^k + \left(\mathbf{U}_0 - \mathbf{MU}^k\right) = \mathbf{U}_0 + \Delta t\mathbf{QFU}^k \approx u_0 + \int_{T^n}^{T^{n+1}} f(\tau, u^k(\tau))\ d\tau$$

$\rightarrow$ Typically not very good convergence properties

- Need a **preconditioner** $\mathbf{Q}_\Delta$ with

$$\Delta t\mathbf{Q}_\Delta\mathbf{FU} \approx \Delta t\mathbf{QFU} \approx \int_{T_n}^{T_{n+1}} f(u(s))\ ds$$

which is easy to invert

[1] For the sake of simplicity

Spectral deferred corrections (SDC)

Dutt, Greengard, Rokhlin
BIT Numerical Mathematics 2000

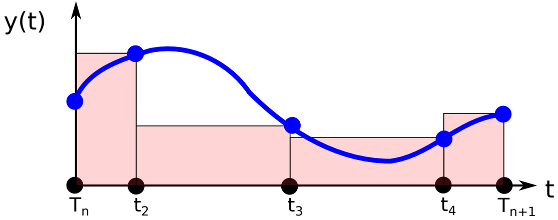


Figure: Use composite rectangular rule $\mathbf{Q}_\Delta$ to precondition high-order collocation rule $\mathbf{Q}$

- Consider preconditioned Richardson iteration

$$\mathbf{Y}^{k+1} = \mathbf{Y}^k + (\mathbf{I} - \Delta t \mathbf{Q}_\Delta \mathbf{F})^{-1} \left[\mathbf{Y}_0 - (\mathbf{I} - \Delta t \mathbf{Q}\mathbf{F}) \mathbf{Y}^k \right], \quad \mathbf{Q}_\Delta \approx \mathbf{Q}, \quad \mathbf{Y} \in \mathbb{R}^{NM}$$

- Can computes this node-by-node which gives "classical" SDC sweep

$$y_m^{k+1} = y_{m-1}^{k+1} + \Delta t f(y_m^{k+1}) - \Delta t f(y_m^k) + \sum_{j=1}^{M} s_{m,j} f(y_j^k), \quad y_m^k \in \mathbb{R}^N$$

- Classically SDC is derived from error/correction equations (Dutt et al. 2000)

Multi-level spectral deferred corrections (MLSDC)

Speck, Ruprecht, Emmett, Minion, Bolten, Krause

BIT Numerical Mathematics 2014

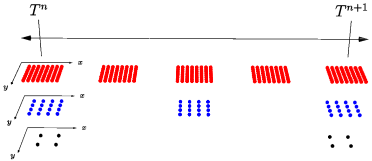


Figure: Hierarchy of three levels with 5 (red), 3 (blue) and 2 (black) collocation nodes and a coarsened spatial mesh for MLSDC applied to a semi-discrete time-dependent PDE.

- SDC sweeps on a **hierarchy** of levels
  $\rightarrow$ SDC sweep analogue to relaxation in space MG
- Employ **FAS correction** to accurately represent solution on $l > 1$
- Reduce overhead from f evaluations on higher levels by **coarsening**:
  $\rightarrow$ reduced number of points, reduced order, reduced implicit solves, reduced physics, ...

PFASST

Emmett, Minion

CAMCOS 2012

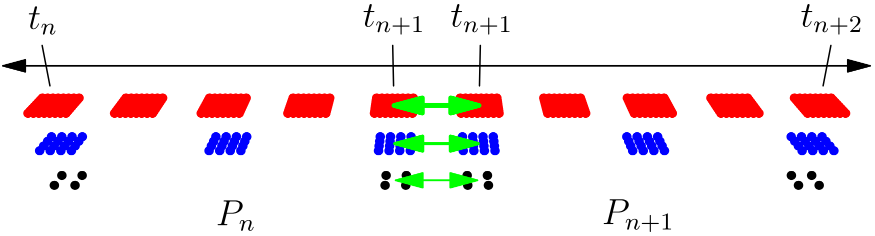


Figure: PFASST performs MLSDC cycles concurrently on multiple time steps, sending updated initial values after every sweep.

PFASST in massively parallel simulations

- Particle based Navier-Stokes solver on $262,144$ cores[1]
- Mesh-based combination with parallel multi-grid on $458,752$ cores[2]

[1] Speck et al., Supercomputing 2012

[2] Ruprecht et al., Supercomputing 2013

Inexact SDC

Speck, Ruprecht, Emmett, Minion, Krause
LNCSE 2015

Algorithm 1: Standard SDC sweep

```
while (sdc_residual > time_tolerance) do
    for (m = 1 ... #substeps) do
        while (pmg_residual > space_tolerance) do
            Perform PMG V-cycle
        end
        Perform backward Euler step: y^k_{m-1} → y^k_m
    end
end
```

Algorithm 2: Sweep with inexact SDC

```
while (sdc_residual > time_tolerance) do
    for (m = 1 ... #substeps) do
        // Incomplete implicit solve
        for (k = 1 ... NumV) do
            Perform PMG V-cycle
        end
        Perform approximate backward Euler step: y^k_{m-1} → y^k_m
    end
end
```

IPFASST convergence

Minion et al.
SISC 2015

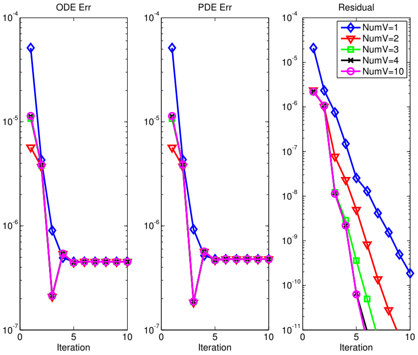


Figure: Convergence of IPFASST for simple $u_t = u_{xx}$ problem depending on number of V-cycles.

- Stage y_m^k becomes increasingly accurate initial guess when solving for y_m^{k+1}:

$$y_m^{k+1} = y_{m-1}^{k+1} + \Delta t f(y_m^{k+1}) - \Delta t f(y_m^k) + \sum_{j=1}^{M} s_{m,j} f(y_j^k), \quad y_m^k \in \mathbb{R}^N$$

IPFASST strong scaling

Minion et al.

SISC 2015

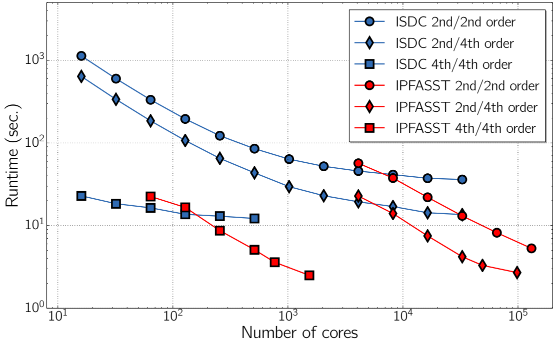


Figure: Strong scaling of IPFASST with parallel multi-grid solver (PMG) on the IBM Blue Gene/Q JUQUEEN at JSC. All shown runs provide the same accuracy.

- Space parallelization through parallel multigrid (code by M. Bolten)
- IPFASST can provide speedup beyond saturation of spatial parallelization alone

Summary

- SDC is a preconditioned fixed point iteration to solve collocation equation
- MLSDC is multi-level extension with sweeps in a sense analogue to relaxation
- PFASST parallelizes MLSDC in time by iterating concurrently on multiple time-steps
- Iterative structure of SDC provides increasingly accurate starting values for PMG
  $\rightarrow$ can use approximate solves in implicit sub steps
  $\rightarrow$ corresponds to different order of time and space sweeps/cycles
- Results in ISDC, IMLSDC, IPFASST
- IPFASST shows good parallel performance on $\mathcal{O}(10k - 100k)$ cores in first benchmarks

The End – for now.