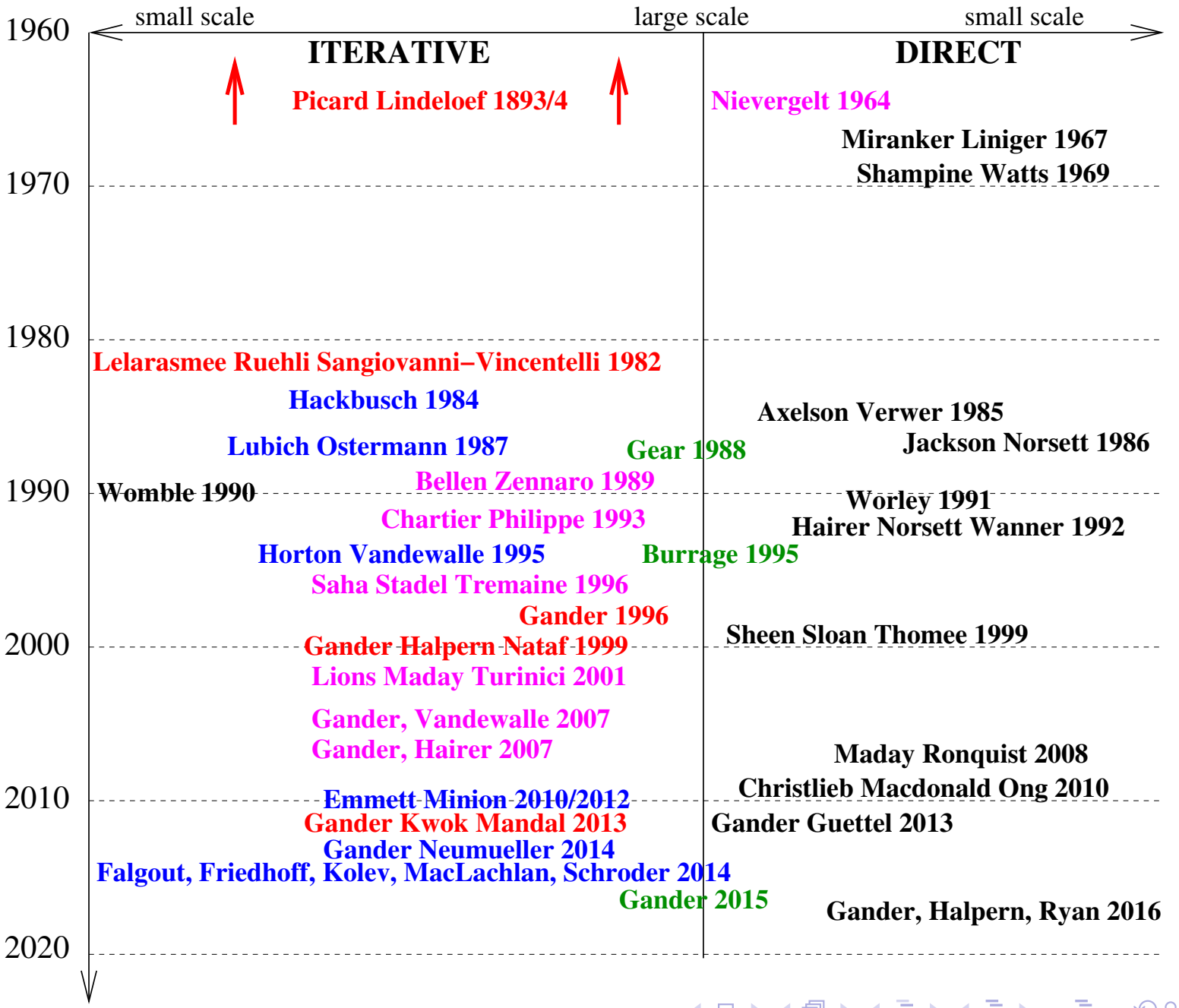
Time Parallel Time Integration Chapter 5: Direct Space-Time Parallel Methods

Martin J. Gander
martin.gander@unige.ch

University of Geneva

Michigan, August 5th, 2022

Direct Space-Time Parallel Methods



Parallel Predictor Corrector Methods

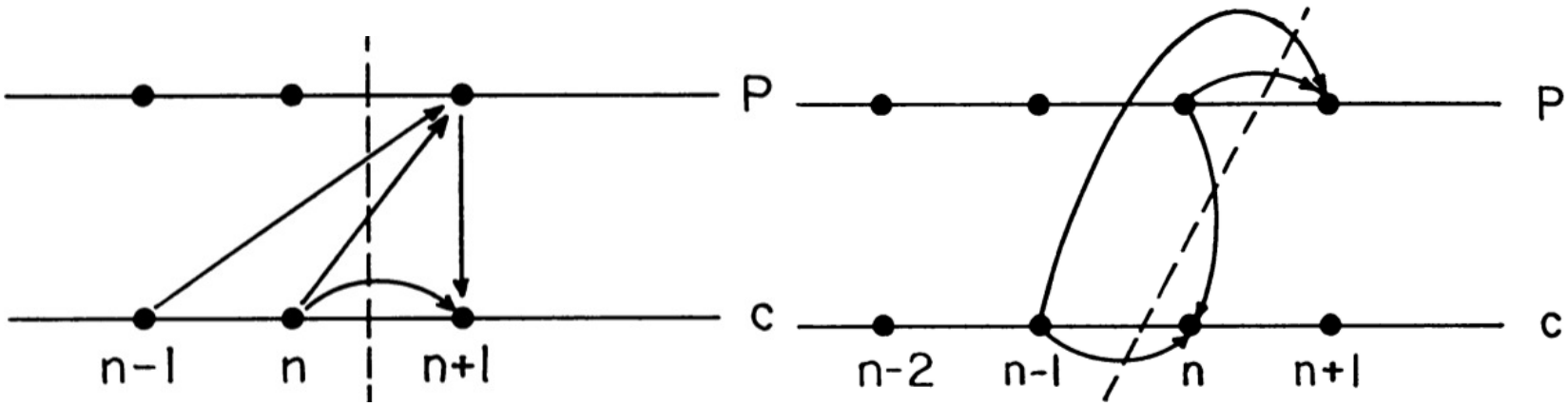
Miranker, Liniger (1967): Parallel Methods for the Numerical Integration of Ordinary Differential Equations

> *"It appears at first sight that the sequential nature of the numerical methods do not permit a parallel computation on all of the processors to be performed. We say that* the front of computation *is too narrow to take advantage of more than one processor... Let us consider how we might widen the computation front."*

Miranker and Liniger consider predictor corrector formulas for ODEs of the form

$$y' = f(x, y), \quad y(0) = y_0.$$

Two examples of Miranker and Liniger



$$y^p_{n+1}=y^c_n+\frac{h}{2}(f(y^c_n)-f(y^c_{n-1})), \quad y^p_{n+1}=y^c_{n-1}+2hf(y^p_n),$$
$$y^c_{n+1}=y^c_n+\frac{h}{2}(f(y^p_{n+1})+f(y^c_n)), \quad y^c_n=y^c_{n-1}+\frac{h}{2}(f(y^p_n)+f(y^c_{n-1}))$$

The left method is sequential, the right is parallel!

Miranker and Liniger derive general numerical integration methods which can be executed on $2s$ processors in parallel, and study their stability and convergence.

Shampine and Watts (1969): Similar parallelism with the block implicit one-step methods.

Boundary Value Methods

Axelson and Verwer (1985):

> *"Hereby we concentrate on explaining the fundamentals of the method because for initial value problems the boundary value method seems to be fairly unknown [...] In the forward-step approach, the numerical solution is obtained by stepping through the grid [...] In this paper, we will tackle the numerical solution in a completely different way [...]* **We will consider $\dot{y} = f(x, y)$ as a two point boundary value problem with a given value at the left endpoint and an implicitly defined value, by the equation $\dot{y} = f(x, y)$, at the right endpoint**."

Simple example (already proposed by Fox in 1954). Suppose we discretize $\dot{y} = f(y)$ with the explicit midpoint rule

$$y_{n+1} - y_{n-1} - 2hf(y_n) = 0, \quad y_0 = y(0).$$

Boundary Value Methods

For such two step methods, we also need an initial approximation for y_1. Usually, one defines y_1 from y_0 using a one step method, e.g. Backward Euler,

$$y_1 - y_0 - hf(y_1) = 0.$$

In boundary value methods, one leaves y_1 as an unknown, and uses Backward Euler at the endpoint y_N to close the system, imposing

$$y_N - y_{N-1} - hf(y_N) = 0.$$

This transforms the normally triangular system into a tridiagonal system.

Example of a Boundary Value Method

For the Dahlquist problem $\dot{y} = \lambda y$, we get with Backward Euler to define y_1 the triangular system

$$\begin{pmatrix} 1-\lambda h & & & & \\ -2\lambda h & 1 & & & \\ -1 & -2\lambda h & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & -2\lambda h & 1 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_N \end{pmatrix} = \begin{pmatrix} y_0 \\ y_0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}.$$

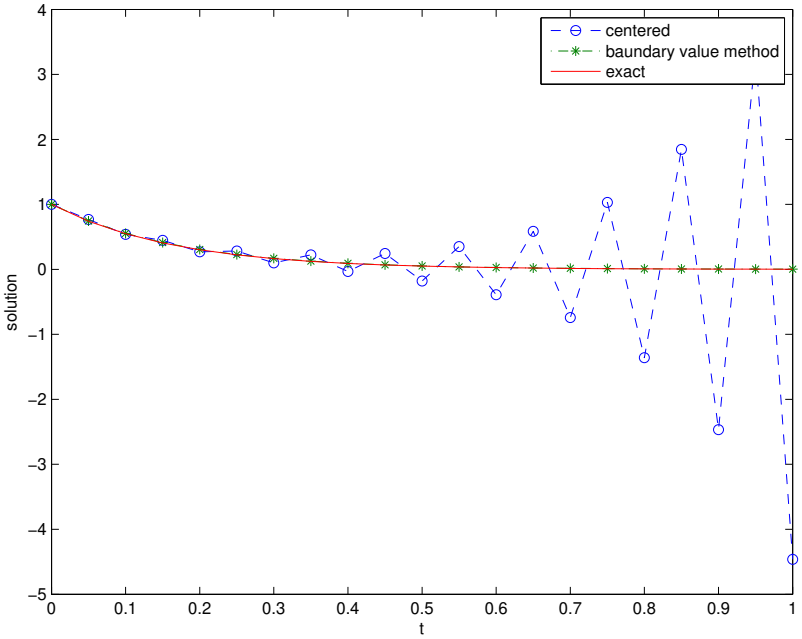
For the boundary value method, leaving y_1 free and using Backward Euler on the right gives the tridiagonal system

$$\begin{pmatrix} -2\lambda h & 1 & & & \\ -1 & -2\lambda h & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & -2\lambda h & 1 \\ & & & -1 & 1-\lambda h \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_N \end{pmatrix} = \begin{pmatrix} y_0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

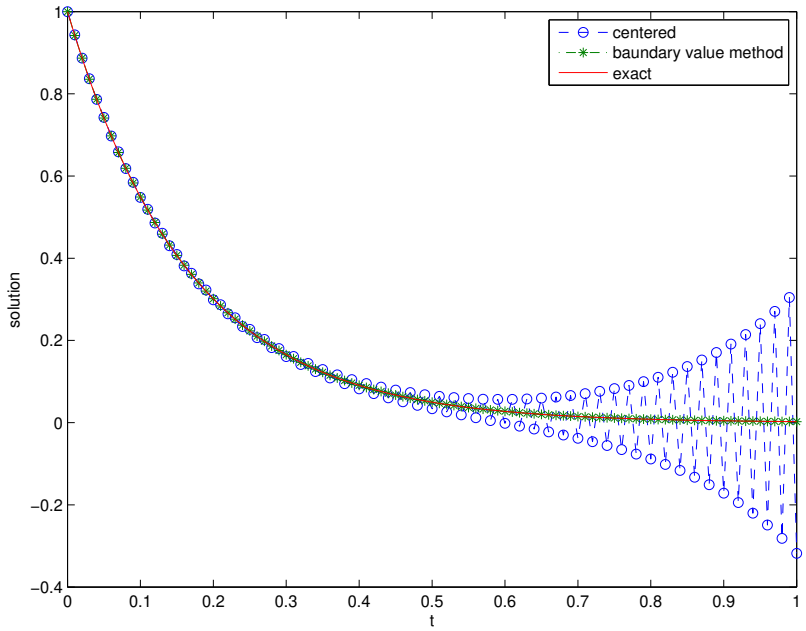
This can be solved working on all time levels in parallel.

Numerical Properties

Boundary value methods are completely different discretizations from initial value methods.



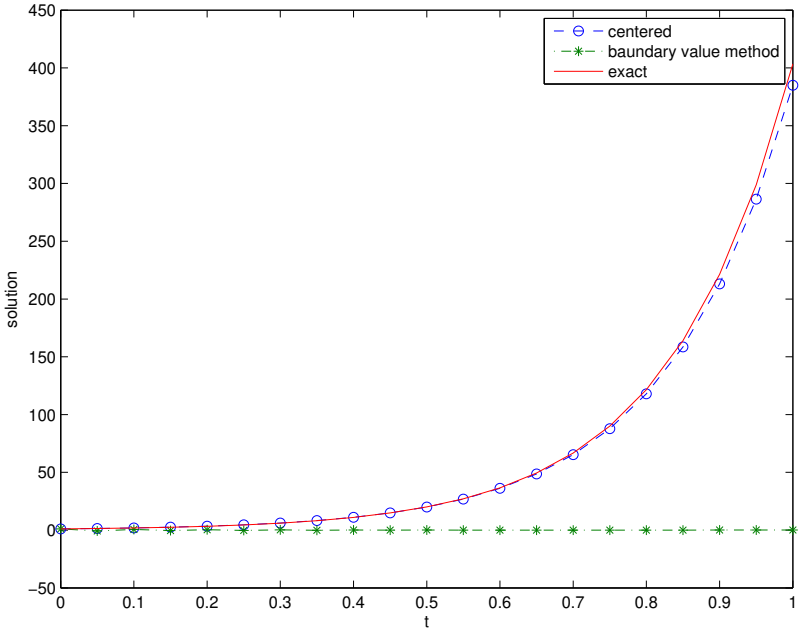
$\lambda = -6,\ h = 1/20$



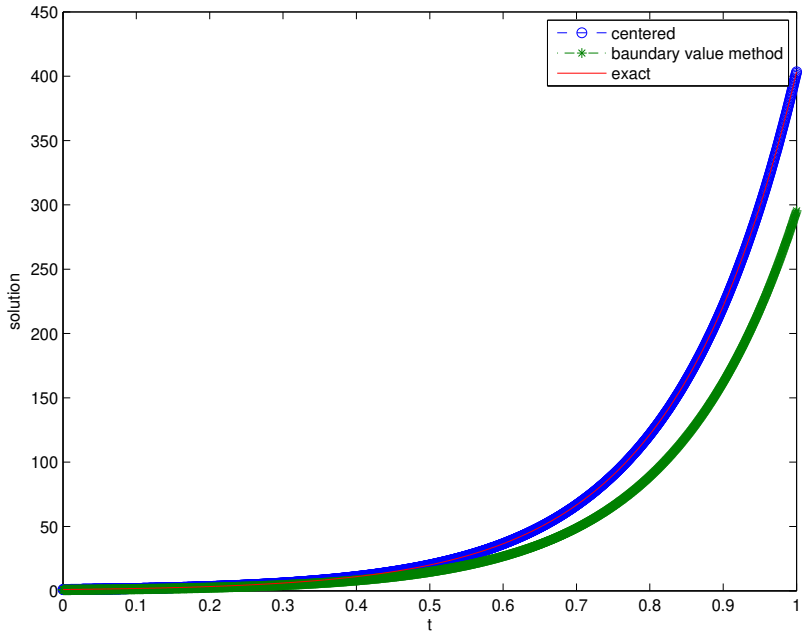
$\lambda = -6,\ h = 1/100$

For a decaying solution, $\lambda < 0$, the initial value method is exhibiting stability problems, while the boundary value method is perfectly stable.

Numerical Properties



$\lambda = 6,\ h = 1/20$



$\lambda = 6,\ h = 1/2000$

For growing solutions, $\lambda > 0$ it is the opposite, the initial value method gives very good approximations, while the boundary value method needs extremely fine time steps to converge.

One can therefore not just transform an initial value method into a boundary value method in order to obtain a parallel solver, see Brugnano (1993, 1996, 1998) and references therein.

Time Parallel Time Stepping

Womble (1990): A Time-Stepping Algorithm for Parallel Computers

> *"Parabolic and hyperbolic differential equations are often solved numerically by time stepping algorithms. These algorithms have been regarded as sequential in time; that is, the solution on a time level must be known before the computation for the solution at subsequent time levels can start. While this remains true in principle, we demonstrate that* it is possible for processors to perform useful work on many time levels simultaneously."

(see also the earlier work by Saltz and Naik (1988))

Idea: Solve time stepping equations by iteration on several time levels simultaneously.

Time Parallel Time Stepping

To explain the method, we discretize the parabolic problem

$$u_t = \mathcal{L}u + f$$

by an implicit time discretization to get at each time step

$$A_n \boldsymbol{u}_n = \boldsymbol{f}_n + B_n \boldsymbol{u}_{n-1}.$$

Using Gauss-Seidel to solve iteratively, with $A_n = L_n + D_n + U_n$, one solves for $k = 1, 2, \ldots, K$

$$(L_n + D_n)\boldsymbol{u}_n^k = -U_n \boldsymbol{u}_n^{k-1} + \boldsymbol{f}_n + B_n \boldsymbol{u}_{n-1}^K.$$

Idea: break the sequential nature by the slight modification

$$(L_n + D_n)\boldsymbol{u}_n^k = -U_n \boldsymbol{u}_n^{k-1} + \boldsymbol{f}_n + B_n \boldsymbol{u}_{n-1}^{k-1},$$

Womble was the first to demonstrate practical speedup on a 1024-processor machine, even though it was later shown that only limited speedups are possible with this relaxation alone (Deshpande, Malhotra, Schultz, Douglas 1995)

Time Parallel Cyclic Reduction

Worley (1991): Parallelizing across time when solving time-dependent partial differential equations

> *"The waveform relaxation multigrid algorithm is normally implemented in a fashion* **that is still intrinsically sequential in the time direction**. *But computation in the time direction only involves solving linear scalar ODEs. If the ODEs are solved using a linear multistep method with a statically determined time step, then each ODE solution corresponds to the solution of a banded lower triangular matrix equation, or, equivalently, a linear recurrence. Parallelizing linear recurrence equations has been studied extensively. In particular,* **if a cyclic reduction approach is used to parallelize the linear recurrence, then parallelism is introduced without increasing the order of the serial complexity**. "

See also Kogge and Stone (1973) for the parallel evaluation of recurrence relations and and Sameh and Brent (1977) for the parallel inversion of triangular matrices.

Time Parallel Cyclic Reduction

Suppose we want to solve the bidiagonal matrix equation

$$\begin{pmatrix} a_{11} & & & \\ a_{21} & a_{22} & & \\ & a_{32} & a_{33} & \\ & & a_{43} & a_{44} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \end{pmatrix}.$$

Then one step of the cyclic reduction algorithm leads to a new matrix equation of half the size,

$$\begin{pmatrix} a_{22} & \\ -\frac{a_{43}}{a_{33}} a_{32} & a_{44} \end{pmatrix} \begin{pmatrix} x_2 \\ x_4 \end{pmatrix} = \begin{pmatrix} f_2 - \frac{a_{21}}{a_{11}} f_1 \\ f_4 - \frac{a_{43}}{a_{33}} f_3 \end{pmatrix},$$

This Schur complement process can be repeated to always get a bidiagonal matrix of half the size.
Once a two by two system is obtained, one can solve directly, and then back-substitute.
Each step of the cyclic reduction is parallel, since each combination of two equations is independent of the others, and also the back-substitution process is parallel.

Generalizations and Complexity

The idea can be generalized to larger bandwidth using block elimination.

Serial complexity in the above example:

- forward substitution $3n$
- cyclic reduction $7n$ (or $5n$ if certain quantities are precomputed)

Parallel complexity becomes a logarithm in n, substantially faster in parallel than just by forward substitution.

Horton, Vandewalle (1995): further results in combination with multigrid waveform relaxation.

Simoens, Vandewalle (2000): a truly optimal time-parallel algorithm, based on a preconditioner in a waveform relaxation setting using a fast Fourier transform in space to decouple the unknowns, and cyclic reduction in time.

Time Parallel Methods with Laplace Transforms

Sheen, Sloan, Thomée (2000): A parallel method for time-discretization of parabolic problems based on contour integral representation and quadrature

> "**These problems are completely independent, and can therefore be computed on separate processors**, *with no need for shared memory. In contrast, the normal step-by-step time-marching methods for parabolic problems are not easily parallelizable.*",

see also Crann, Davies, Lai, Leong (1998).

The idea is to Laplace transform the problem, and then to solve a sequence of steady problems at quadrature nodes used for the numerical evaluation of the inverse Laplace transform, and goes back to the solution in the frequency domain of hyperbolic problems (Douglas Jr, Santos, Sheen, Bennethum 1993)

Example

Suppose we have the initial value problem

$$u_t + Au = 0, \quad u(0) = u_0,$$

where A represents a linear operator. A Laplace transform with parameter s gives

$$s\hat{u} + A\hat{u} = u_0,$$

and to obtain the solution in the time domain, one needs the inverse Laplace transform

$$u(t) = \frac{1}{2\pi i} \int_\Gamma e^{st} \hat{u}(s) ds,$$

where Γ is a suitably chosen contour in the complex plane. If the integral is approximated by a quadrature rule with quadrature nodes s_j, one can compute $\hat{u}(s)$ at $s = s_j$ in parallel.

Remark: restricted to problems where Laplace transform can be applied, i.e. linear problems with constant coefficients in the time direction

Time Parallelization Based on Diagonalization

Maday, Rønquist (2008): Parallelization in time through tensor-product space–time solvers

> *"Pour briser la nature intrinsèquement séquentielle de cette résolution, on utilise* **l'algorithme de produit tensoriel rapide**.*"*

To explain the idea, we discretize the linear evolution problem $u_t = Lu + f$ using Backward Euler,

$$\begin{pmatrix} \frac{1}{\Delta t_1} - L & & & \\ -\frac{1}{\Delta t_2} & \frac{1}{\Delta t_2} - L & & \\ & \ddots & \ddots & \\ & & -\frac{1}{\Delta t_N} & \frac{1}{\Delta t_N} - L \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_N \end{pmatrix} = \begin{pmatrix} f_1 + \frac{1}{\Delta t_1} u_0 \\ f_2 \\ \vdots \\ f_N \end{pmatrix}$$

Using the Kronecker symbol, this linear system can be written in compact form as

$$(B \otimes I_x - I_t \otimes L)\boldsymbol{u} = \boldsymbol{f},$$

I_x and I_t identity matrices and B is the time stepping matrix

Time Parallelization Based on Diagonalization

$$B := \begin{pmatrix} \frac{1}{\Delta t_1} & & & \\ -\frac{1}{\Delta t_2} & \frac{1}{\Delta t_2} & & \\ & \ddots & \ddots & \\ & & -\frac{1}{\Delta t_N} & \frac{1}{\Delta t_N} \end{pmatrix}.$$

If B is diagonalizable, $B = SDS^{-1}$, one can rewrite the system in factored form, namely

$$(S \otimes I_x)(\mathrm{diag}(D - L))(S^{-1} \otimes I_x)\boldsymbol{u} = \boldsymbol{f},$$

and we can hence solve it in 3 steps:

$$\begin{array}{lrcll} (a) & (S \otimes I_x)\boldsymbol{g} & = & \boldsymbol{f}, & \\ (b) & (\frac{1}{\Delta t_n} - L)\boldsymbol{w}^n & = & \boldsymbol{g}^n, & 1 \le n \le N, \\ (c) & (S^{-1} \otimes I_x)\boldsymbol{u} & = & \boldsymbol{w}. & \end{array}$$

Note that the expensive step (b) requiring a solve with the system matrix L can now be done entirely in parallel for all time levels t_n.

Time Parallelization Based on Diagonalization

- Maday and Ronquist obtain with this algorithm for the 1d heat equation close to perfect speedup.
- Use a geometric time mesh $\Delta t_k = \rho^{k-1}\Delta t_1$, $\rho = 1.2$, since "choosing ρ much closer to 1 may lead to instabilities".
- **ParaDiag** is not defined for equal time steps, since it is not possible to diagonalize a Jordan block !
- Truncation and round-off error analysis for wave equation (G, Halpern, Rannou, Ryan 2019).

New ParaDiag: use quasi-circular approximation as preconditioner (Wu et al, Wathen et al,...)

$$B := \begin{pmatrix} \frac{1}{\Delta t} & & & \frac{\alpha}{\Delta t} \\ -\frac{1}{\Delta t} & \frac{1}{\Delta t} & & \\ & \ddots & \ddots & \\ & & -\frac{1}{\Delta t} & \frac{1}{\Delta t} \end{pmatrix}.$$

Revisonist Integral Deferred Correction (RIDC)

Christlieb, Macdonald, Ong 2010: Parallel high-order integrators

> *"...we discuss a class of defect correction methods which is easily adapted to create* **parallel time integrators for multicore architectures**."

Recall Integral Deferred Correction:

1. For an approximation $\tilde{u}_m$ for $\dot{u} = f(u)$ compute the residual

$$r(t) := \tilde{u}(0) + \int_0^t f(\tilde{u}(\tau))d\tau - \tilde{u}(t),$$

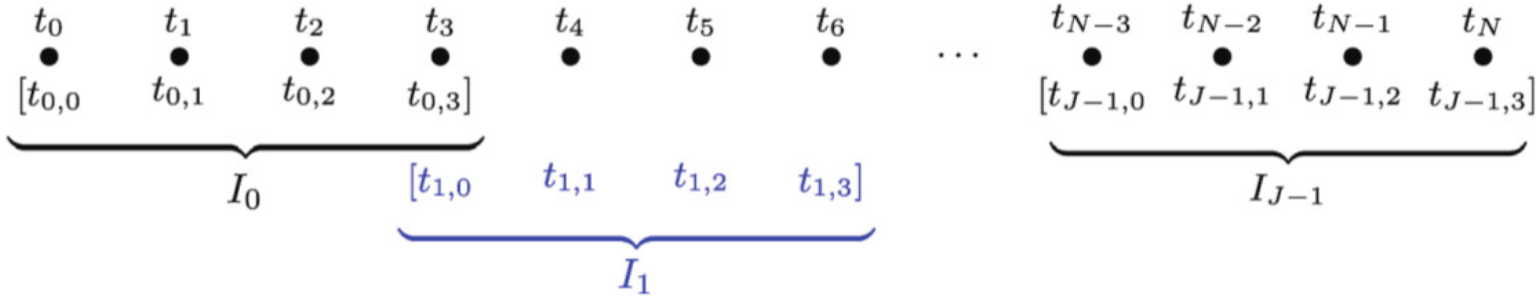
   at t_m, $m = 0, 1, \ldots, M$ using high order quadrature.
2. Solve $e'(t) = r'(t) + f(\tilde{u}(t) + e(t)) - f(\tilde{u}(t))$ with e.g. FE,

$$e_{m+1} = e_m + r_{m+1} - r_m + \Delta t(f(\tilde{u}_m + e_m) - f(\tilde{u}_m)).$$

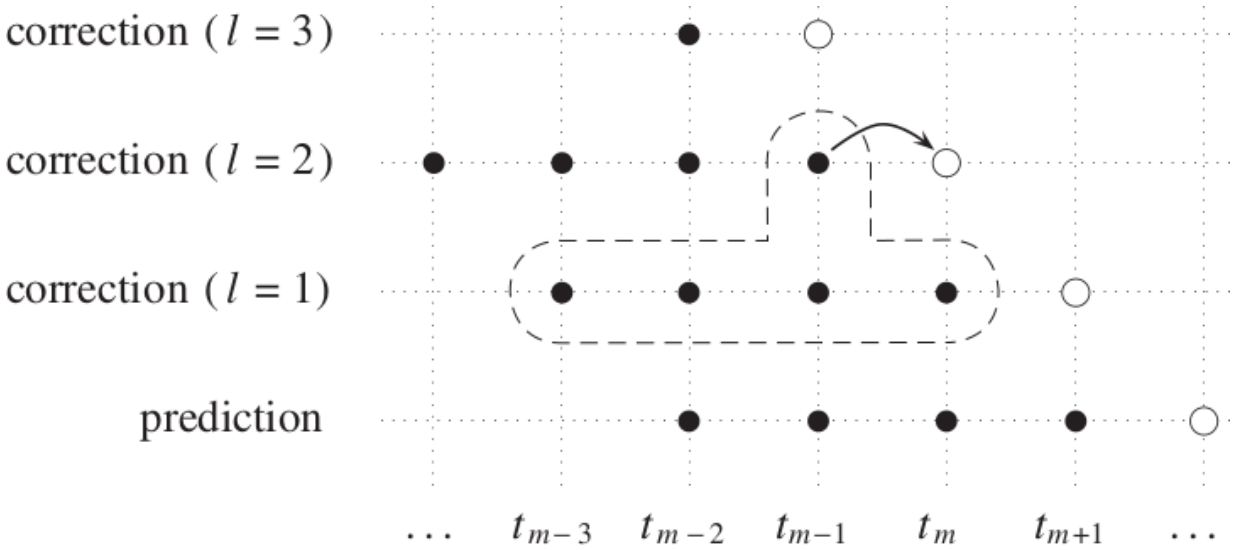
3. Add this correction to obtain a new approximation

$$\tilde{u}_m + e_m, \quad m = 1, 2, \ldots, M.$$

RIDC



In RIDC one increases M to contain more points than the quadrature formula needs and pipelines the computation:



- the Euler prediction step and the correction steps of the integral deferred correction can be executed in parallel
- very good for multicore architectures

ParaExp

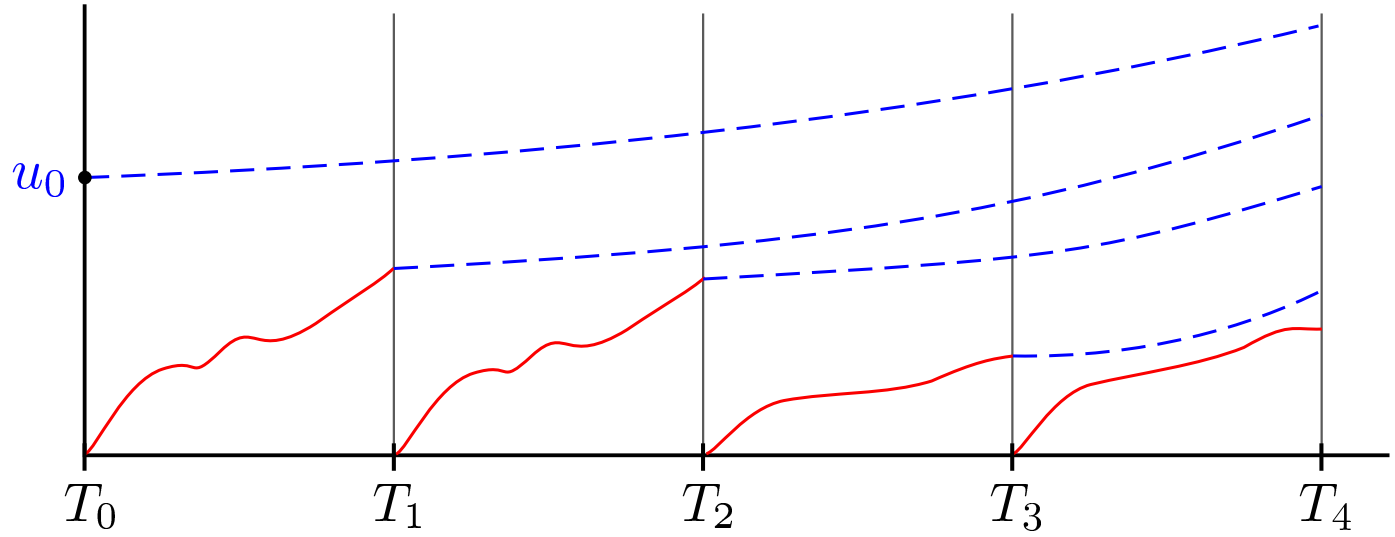
G., Güttel (2013): ParaExp: a Parallel Integrator for Linear Initial-Value Problems

> *"We introduce an* **overlapping time-domain decomposition method** *for linear initial-value problems which gives rise to an efficient solution method for parallel computers without resorting to the frequency domain. This parallel method exploits the fact that* **homogeneous initial-value problems can be integrated much faster than inhomogeneous problems** *by using an efficient Arnoldi approximation for the matrix exponential function."*

Consider the linear system of evolution equations

$$\boldsymbol{u}'(t) = A\boldsymbol{u}(t) + \boldsymbol{g}(t), \quad t \in [0, T], \quad \boldsymbol{u}(0) = \boldsymbol{u}_0.$$

ParaExp is based on a completely overlapping decomposition of the time interval $[0, T]$ into subintervals, e.g. $[0, T_4 := T]$, $[T_1, T_4]$, $[T_2, T_4]$, and $[T_3, T_4]$.



ParaExp is a direct solver, consisting of two steps:

1. solve the non-overlapping inhomogeneous red problems

$$\boldsymbol{v}_j'(t) = A\boldsymbol{v}_j(t) + \boldsymbol{g}(t), \quad \boldsymbol{v}_j(T_{j-1}) = 0, \quad t \in [T_{j-1}, T_j],$$

2. solve the overlapping homogeneous blue problems

$$\boldsymbol{w}_j'(t) = A\boldsymbol{w}_j(t), \quad \boldsymbol{w}_j(T_{j-1}) = \boldsymbol{v}_{j-1}(T_{j-1}),\ t \in [T_{j-1}, T]$$

By linearity, the solution is then obtained by summation,

$$\boldsymbol{u}(t) = \boldsymbol{v}_k(t) + \sum_{j=1}^{k} \boldsymbol{w}_j(t) \quad \text{with } k \text{ s.t.} \quad t \in [T_{k-1}, T_k].$$

Why ParaExp Works

A first sight this seems to be absurd, since the overlapping propagation of the linear homogeneous problems is redundant, and the blue dashed solution needs to be integrated over the entire time interval $[0, T]$!

However near-optimal approximations of the matrix exponential are known, and so the homogeneous problems in dashed blue become very cheap:

1. *projection based methods*, where one approximates $\boldsymbol{a}_n(t) \approx exp(tA)\boldsymbol{v}$ from a Krylov space built with $S := (I - A/\sigma)^{-1}A$

2. *expansion based methods*, which approximate $\exp(tA)\boldsymbol{v} \approx \sum_{j=0}^{n-1} \beta_j(t) p_j(A)\boldsymbol{v}$, where p_j are polynomials or rational functions.

ParaExp for the Wave Equation

$$\begin{aligned} \partial_{tt} u(t,x) &= \alpha^2 \partial_{xx} u(t,x) + \text{hat}(x)\sin(2\pi f t), \quad x,t \in (0,1), \\ u(t,0) &= u(t,1) = u(0,x) = u'(0,x) = 0, \end{aligned}$$

α^2	f	serial		parallel			effi-
		τ_0	error	$\max(\tau_1)$	$\max(\tau_2)$	error	ciency
0.1	1	2.54e−01	3.64e−04	4.04e−02	1.48e−02	2.64e−04	58 %
0.1	5	1.20e+00	1.31e−04	1.99e−01	1.39e−02	1.47e−04	71 %
0.1	25	6.03e+00	4.70e−05	9.83e−01	1.38e−02	7.61e−05	76 %
1	1	7.30e−01	1.56e−04	1.19e−01	2.70e−02	1.02e−04	63 %
1	5	1.21e+00	4.09e−04	1.97e−01	2.70e−02	3.33e−04	68 %
1	25	6.08e+00	1.76e−04	9.85e−01	2.68e−02	1.15e−04	75 %
10	1	2.34e+00	6.12e−05	3.75e−01	6.31e−02	2.57e−05	67 %
10	5	2.31e+00	4.27e−04	3.73e−01	6.29e−02	2.40e−04	66 %
10	25	6.09e+00	4.98e−04	9.82e−01	6.22e−02	3.01e−04	73 %

Finite differences and RK45 for the red problems, Chebyshev exponential integrator for the blue ones, 8 processors

ParaExp for the Heat Equation

$$\begin{aligned} \partial_t u(t,x) &= \alpha \partial_{xx} u(t,x) + \text{hat}(x)\sin(2\pi f t), \qquad x,t \in (0,1), \\ u(t,0) &= u(t,1) = 0, \quad u(0,x) = 4x(1-x), \end{aligned}$$

α	f	serial		parallel			effi-
		τ_0	error	$\max(\tau_1)$	$\max(\tau_2)$	error	ciency
0.01	1	4.97e−02	3.01e−04	1.58e−02	9.30e−03	2.17e−04	50 %
0.01	10	2.43e−01	4.14e−04	7.27e−02	9.28e−03	1.94e−04	74 %
0.01	100	2.43e+00	1.73e−04	7.19e−01	9.26e−03	5.68e−05	83 %
0.1	1	4.85e−01	2.24e−05	1.45e−01	9.31e−03	5.34e−06	79 %
0.1	10	4.86e−01	1.03e−04	1.45e−01	9.32e−03	9.68e−05	79 %
0.1	100	2.42e+00	1.29e−04	7.21e−01	9.24e−03	7.66e−05	83 %
1	1	4.86e+00	7.65e−08	1.45e+00	9.34e−03	1.78e−08	83 %
1	10	4.85e+00	8.15e−06	1.45e+00	9.33e−03	5.40e−07	83 %
1	100	4.85e+00	3.26e−05	1.44e+00	9.34e−03	2.02e−05	84 %

Finite differences and RK45 for the red problems, Chebyshev exponential integrator for the blue problems, 4 processors.

Conclusion

1. Methods based on multiple shooting
   - Nievergelt
   - True multiple shooting
   - Parareal using an approximate Jacobian
2. Methods based on domain decomposition and waveform relaxation
   - Waveform relaxation for circuits
   - Schwarz waveform relaxation
3. Space-time multigrid methods
   - Parabolic multigrid
   - Block Jacobi smoother for highly scalable Space-Time MG
4. Direct time parallel methods:
   - Many techniques (Predictor-Corrector, Boundary Value Methods, Parallel Time Stepping, Cyclic Reduction, Laplace Transforms, RIDC, ParaExp)
   - New ParaDiag family

Strong Research Community, Annual PinT Workshop, Many New Results!