A Parallel Overlapping Time-Domain Decomposition Method for ODEs

Stefan Güttel[1]

University of Geneva, Department of Mathematics, 1–2 rue du Lievre, CH-1202 Geneva,
stefan.guettel@unige.ch

We introduce an overlapping time-domain decomposition 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.

1 Introduction

We are interested in the parallel solution of a linear initial-value problem

$$u'(t) = Au(t) + g(t), \quad t \in [0,T], \quad u(0) = u_0, \tag{1}$$

where $A \in \mathbb{R}^{N\times N}$ is a possibly large (and sparse) matrix and $u, g : t \mapsto \mathbb{R}^N$. Throughout this paper we assume that the function $g(t)$ is a source term which is difficult to integrate numerically (e.g., highly oscillating or given by a slow computer subroutine). For example, if (1) arises from the space discretization of a heat-diffusion problem, then A represents a diffusion operator and $g(t)$ is a time-dependent heat source.

Problems of the above form arise often in scientific computing, and various solution methods for parallel computers have been proposed in the literature. A popular approach (see, e.g., [8, 1]) is based on the Laplace-transformed equation

$$s\widehat{u}(s) - u_0 = A\widehat{u}(s) + \widehat{g}(s)$$

and the contour integral representation of the inverse transformation

$$u(t) = \frac{1}{2\pi i}\int_\Gamma e^{ts}\widehat{u}(s)\,\mathrm{d}s,$$

with a suitable contour Γ surrounding the singularities of $\widehat{u}(s)$ (which are the eigenvalues of A and all singularities of $\widehat{g}(s)$). Discretization of this integral by a quadrature formula with complex nodes s_j and weights w_j yields

$$u(t) \approx \sum_{j=1}^{p} w_j \widehat{u}(s_j) = \sum_{j=1}^{p} w_j (s_j I - A)^{-1}(u_0 + \widehat{g}(s_j)).$$

This method is suitable for parallel computation because the p complex shifted linear systems are decoupled. On the other hand, there are obvious drawbacks such as the introduction of complex arithmetic into a real problem and the need for calculating $\widehat{g}(s_j)$. Moreover, many nodes s_j may be required to represent a stiff source $g(t)$ to prescribed accuracy.

Another approach, perhaps closest in spirit to the method described here, is known as exponential quadrature. It is based on the variation-of-constants formula

$$u(t) = e^{tA}u_0 + \int_0^t e^{(t-\tau)A} g(\tau)\,\mathrm{d}\tau$$

and the approximation of the integrand by a quadrature rule in nodes $\tau_1, \ldots, \tau_p$. This yields $p+1$ independent matrix exponentials

$$e^{tA}u_0 \quad \text{and} \quad e^{(t-\tau_j)A} g(\tau_j) \quad \text{for} \quad j = 1, \ldots, p,$$

each of which may be approximated efficiently by a Krylov method (see the discussion in Section 3). However, exponential quadrature is impractical if the source term $g(t)$ is stiff enough so that too many quadrature nodes are needed.

To overcome the problems mentioned above, we propose in Section 2 a decomposition of (1) into subproblems on overlapping time intervals. These subproblems are decoupled and can be assigned to independent processors. Our method requires almost no communication or synchronization between the processors, except a summation step at the end of the algorithm. Another advantage of our method is its ease of implementation; any available serial integrator for (1) can be used in black-box fashion. Because the efficiency of our method relies on the fast integration of homogeneous linear initial-value problems, Section 3 contains a brief discussion of the Arnoldi method for computing the matrix exponential function. In Section 4 we discuss the error control and parallel efficiency of our method. In Section 5 we present results of a numerical experiment.

2 Overlapping time-domain decomposition

On a time grid $\{T_j = jT/p : j = 0, \ldots, p\}$ we decompose (1) into the following subproblems of two types.

Type 1 : For $j = 1, \ldots, p$ solve

$$v_j'(t) = Av_j(t) + g(t), \quad v_j(T_{j-1}) = 0, \quad t \in [T_{j-1}, T_j],$$

using some serial integrator.

Type 2 : For $j = 1, \ldots, p$ solve

$$w_j'(t) = Aw_j(t), \quad w_j(T_{j-1}) = v_{j-1}(T_{j-1}), \quad t \in [T_{j-1}, T],$$

using exponential propagation (we set $v_0(T_0) := u_0$).

Note that the p subproblems of Type 1 are completely decoupled due to the homogeneous initial values. The same is true for each subproblem of Type 2, the exact solution of which can be computed as

$$w_j(t) = e^{(t-T_{j-1})A} v_{j-1}(T_{j-1}) \tag{2}$$

as soon as the initial value $v_{j-1}(T_{j-1})$ is available. Therefore it is natural to assign the integrations for v_{j-1} and w_j to the same processor so that there is no need for communication and synchronization between the two types of subproblems. Note that the time intervals $[T_{j-1}, T]$ for the w_j are overlapping (see also Figure 1). By superposition, the solution of (1) is

$$u(t) = v_k(t) + \sum_{j=1}^{k} w_j(t) \quad \text{with } k \text{ such that} \quad t \in [T_{k-1}, T_k].$$

Only the computation of this sum requires communication between the processors. Our parallel algorithm is given by simultaneously integrating the subproblems of Type 1 and Type 2, and finally forming the sum for $u(t)$ at the required time points t.

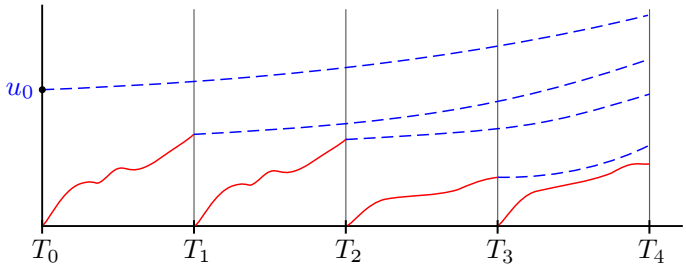

Fig. 1. Time-domain decomposition of an initial-value problem into inhomogeneous subproblems with zero initial value (Type 1, solid red curves) and overlapping homogeneous subproblems (Type 2, dashed blue curves). The solution is obtained as the sum of all curves.

3 Computing the matrix exponential

The overlapping propagation of the linear homogeneous subproblems of Type 2 is clearly redundant. To obtain an efficient parallel method, we require that the computation of the matrix exponentials in (2) is fast compared to the integration of the subproblems of Type 1.

For scalar problems ($N = 1$) the computation of the exponential is a trivial task. For computing the exponential of small to medium-sized dense matrices ($N \lessapprox 500$) there are various methods available, see the review [5] and the monograph [4].

The computations become more challenging when the problem size N gets large, in which case the matrix A should be sparse. Then one has to make use of the fact that not the matrix exponential $\exp(tA)$ itself is required, but only the product $\exp(tA)v_0$ with a vector v_0, by using a polynomial or rational Krylov method (see [3] and the references therein). For brevity we will only describe a variant of the restricted-denominator Arnoldi method described in [6] (see also [9]), which extracts an approximation $f_n(t) \approx \exp(tA)v_0$ from a Krylov space built with the matrix $S = (I - A/\sigma)^{-1}A$,

$$\mathscr{K}_n(S, v_0) = \operatorname{span}\{v_0, Sv_0, \ldots, S^{n-1}v_0\},$$

the choice of the parameter $\sigma \in (\mathbb{R} \cup \{\infty\}) \setminus (\Lambda(A) \cup \{0\})$ being dependent on the spectral properties of A. For $\sigma = \infty$ we obtain a standard Krylov space with the matrix A, i.e., $\mathscr{K}_n(S, v_0) = \mathscr{K}_n(A, v_0)$. If $\mathscr{K}_n(S, v_0)$ is of full dimension n, as we assume in the following, we can compute an orthonormal basis $V_n = [v_1, v_2, \ldots, v_n]$ by using the well-known Arnoldi orthogonalization process (see, e.g., [2, §9.3.5]). The Arnoldi approximation of $\exp(tA)v_0$ is then defined as

$$f_n(t) := V_n \exp(t\,(S_n^{-1} + I_n/\sigma)^{-1})V_n^* v_0, \quad S_n := V_n^* S V_n.$$

Provided that n is small, the computation of $f_n(t)$ requires the evaluation of a $n \times n$ matrix function which is small compared to the original $N \times N$ matrix exponential. Moreover, the matrix S_n can be constructed without explicit projection from quantities computed in the Arnoldi process.

In Figure 2 we show the error norm $\|\exp(A)v_0 - f_n(1)\|_2$ of the Arnoldi approximations with parameters $\sigma = \infty$ and $\sigma = 40$ (a rather arbitrary choice) as a function of n, for the matrices

$$A_1 = \operatorname{tridiag}(30, -40, 10) \in \mathbb{R}^{199\times 199},\ A_2 = \operatorname{tridiag}(60, -90, 30) \in \mathbb{R}^{299\times 299}$$

arising from the finite-difference discretization of the same 1D advection–diffusion problem, and a random vector v_0. We have also plotted the error of orthogonal projection of the exact solution onto the space $\mathscr{K}_n(S, v_0)$, namely $V_n V_n^* e^A v_0$, and observe that the Arnoldi method is capable of extracting an approximation nearby this projection. For comparison we show the error of the result produced by n steps of various explicit and implicit integrators for the initial-value problem $v' = Av$, $v(0) = v_0$, integrated to $t = 1$. For this linear homogeneous problem all integrators actually compute approximations from some Krylov space $\mathscr{K}_n(S, v_0)$ (for the explicit integrators with shift $\sigma = \infty$ and for implicit Euler with $\sigma = n$), but the Arnoldi methods extract much better approximations in the same number of iterations. Note also that the Arnoldi method with finite shift $\sigma = 40$ converges almost independently of the problem size N, a property often referred to as *mesh-independence*.

Because the error of Arnoldi approximations decays usually very fast (i.e., $\|e^{tA}v_0 - f_{n+1}(t)\|$ is considerably smaller than $\|e^{tA}v_0 - f_n(t)\|$), it is often sufficient

to use the difference of two consecutive iterates as an estimate for the approximation error:

$$\begin{aligned}\|e^{tA}v_0 - f_n(t)\| &\le \|e^{tA}v_0 - f_{n+1}(t)\| + \|f_{n+1}(t) - f_n(t)\| \\ &\approx \|f_{n+1}(t) - f_n(t)\|. \end{aligned} \tag{3}$$

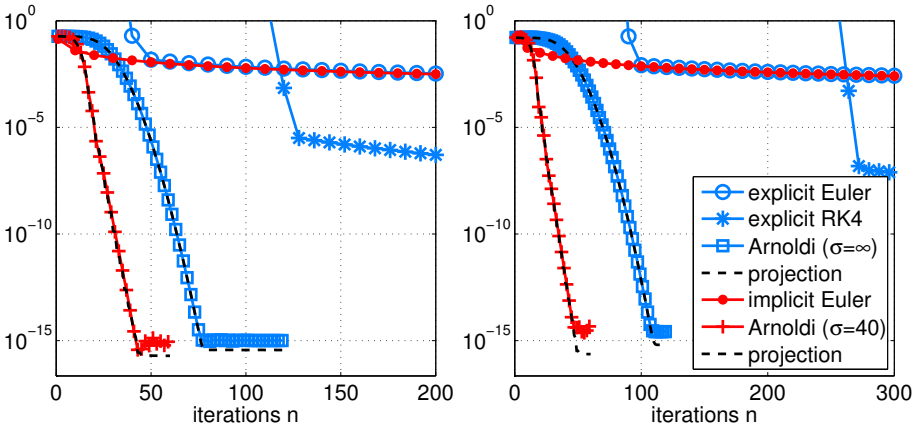

Fig. 2. Error (2-norm) of various time-stepping methods and Krylov methods for a linear homogeneous advection–diffusion problem $v' = Av$, $v(0) = v_0$, of size $N = 199$ (left) and $N = 299$ (right) as a function of time steps or Krylov space dimension n, respectively.

4 Error control and parallel efficiency

Many ODE solvers, for example those of MATLAB, use an error control criterion like

$$\|e(t)\|_\infty \le \max\{\texttt{reltol} \cdot \|\widetilde{u}(t)\|_\infty, \texttt{abstol}\}, \quad t \in [0,T],$$

where $e(t) = u(t) - \widetilde{u}(t)$ is the (estimated) error of the computed solution $\widetilde{u}(t)$. Because the inhomogeneous subproblems of Type 1 for $v_j(t)$ are solved with zero initial guess, it is not advisable to use an error criterion which is relative to the norm of the solution. Hence we assume that all of these subproblems are solved with an absolute error $\|e_j(t)\|_\infty \le \texttt{abstol}/p$ over the time interval $[T_{j-1}, T_j]$. This error is then propagated exponentially over the remaining interval $[T_j, T]$, hence we have to study the transient behavior of

$$\|e^{tA}e_j(T_j)\|_\infty \le \|e^{tA}\|_\infty \texttt{abstol}/p \tag{4}$$

for $t \in [0, T - T_j]$. It is well known that for a *stable* matrix A (i.e., all eigenvalues lie in the left complex half-plane) the limit $\lim_{t\to\infty} \|e^{tA}\|_\infty$ is finite. Unfortunately, the

norm may initially grow arbitrarily large before convergence sets in, a phenomenon usually referred to as *hump* (see [5]). However, for a diagonally dominant matrix $A = (a_{ij})$ with $a_{ii} \leq 0$ this cannot happen, as one can show as follows (cf. [7]): Define $\rho = \max_i\{a_{ii} + \sum_{j\neq i}|a_{ij}|\} \leq 0$. By the formula $\exp(tA) = \lim_{k\to\infty}(I + tA/k)^k$ we have $\|e^{tA}\|_\infty \leq \lim_{k\to\infty}\|I + tA/k\|_\infty^k$. For k sufficiently large we have

$$\|I + tA/k\|_\infty = \max_i \left\{1 + t\Big(a_{ii} + \sum_{j\neq i}|a_{ij}|\Big)/k\right\} = 1 + t\rho/k,$$

hence

$$\|e^{tA}\|_\infty \leq \lim_{k\to\infty}(1 + t\rho/k)^k = e^{t\rho} \leq 1 \quad \text{for all } t \geq 0.$$

Of course, it is possible to estimate the behavior of $\|e^{tA}\|$ for general matrices and in other norms (see, e.g., [10]), but for brevity we will only consider a diagonally dominant A. In this case the errors $e_j(t)$ of the subproblem solutions $v_j(t)$ $(j = 1,\ldots,p)$ are non-increasing when being exponentially propagated, and if we assume that the subproblems of Type 2 are solved exactly (or with sufficiently high accuracy), then the overall error $e(t)$ is bounded[1] by the sum of subproblem errors (4), hence $\|e(t)\|_\infty \leq$ `abstol`. If the integrator is a time-stepping method of order q, it is reasonable to assume that the computation time for one subproblem of Type 1 is at most $\tau_1(p) = (\tau_0 \cdot p^{1/q})/p$, where τ_0 is the computation time for serial integration over $[0,T]$. If each subproblem of Type 2 takes at most τ_2 units of computation time, the expected efficiency of our parallel algorithm is at least

$$\text{efficiency} = \frac{\text{speedup}}{p} = \frac{1}{p}\cdot\frac{\tau_0}{\tau_1(p)+\tau_2} = \left(p^{1/q} + \frac{p\cdot\tau_2}{\tau_0}\right)^{-1}. \tag{5}$$

The efficiency becomes large if the serial computation time τ_0 is long compared to $p\cdot\tau_2$, and if the integration order q is high.

5 Numerical example

As a simple model problem we consider the 1D heat equation

$$\begin{aligned}
\partial_t u(t,x) &= \alpha\,\partial_{xx}u(t,x) + g(t,x) && \text{on } x \in (0,1),\\
u(t,0) &= u(t,1) = 0,\\
u(0,x) &= u_0(x) = 4x(1-x),\\
g(t,x) &= e\max\{1-|c-x|/d,0\}, && \text{where } c = .5 + (.5-d)\sin(2\pi f t).
\end{aligned}$$

The source term $g(t,x)$ is a hat function centered at c with half-width $d = 0.05$ and height $e = 100\cdot\alpha^{1/2}$, oscillating with frequency f. Finite-difference discretization

[1] This worst-case bound is sharp only if all errors e_j are collinear, which is rather unlikely. Probabilistic error estimation would give $\|e(t)\|_\infty \lessapprox$ `abstol`$/\sqrt{p}$. This explains why the observed parallel efficiency of our algorithm is usually better than predicted by (5). We plan to investigate this in a sequel.

at $N = 100$ points $x_j = j/(N+1)$ $(j = 1,\ldots,N)$ yields an initial-value problem (1), where $A = \alpha(N+1)^2\,\mathrm{tridiag}(1,-2,1) \in \mathbb{R}^{N\times N}$. This problem is integrated over the time interval $[0, T = 1]$. For the serial integration we have used the classical Runge–Kutta method of order $q = 4$ (implemented in MATLAB) with constant step size

$$h_0 = \min\{5\cdot 10^{-5}/\alpha, 10^{-2}/f\},$$

chosen to avoid instability of the time-stepping method caused by the stiff linear term $Au(t)$ and to capture the oscillations of $g(t)$. As shown in Table 1, the absolute error (∞-norm) is at most $5\cdot 10^{-4}$ for all diffusion coefficients $\alpha = 0.01, 0.1, 1$ and frequencies $f = 1, 10, 100$. These parameters determine the stiffness of $Au(t)$ and $g(t)$, respectively. We have also tabulated the serial integration times τ_0. As expected, these are roughly proportional to h_0^{-1}.

For our parallel algorithm we have partitioned the interval $[0, T]$ in $p = 4$ subintervals, and computed the solution $u(t)$ at all time points $T_j = jT/p$ $(j = 1,\ldots,p)$. The subproblems of Type 1 are integrated with step size $h_1 = h_0/\sqrt{p}^{1/q}$ (based on a probabilistic error assumption, see the footnote on page 206). In Table 1 we list the maximal computation time τ_1 for all subproblems of Type 1 among all processors.

For the subproblems of Type 2 we have used the Arnoldi method described in Section 3 with shift $\sigma = 5.3$, in combination with the ∞-norm error estimate (3) for an accuracy of 10^{-4} (for more details on the selection of σ we refer to [9]). In Table 1 we list the maximal computation time τ_2 for all subproblems of Type 2 among all processors.

The errors of the final solutions computed with our parallel algorithm are shown in the second-last column, and they are all below the errors obtained by sequential integration. This indicates that our choice for the step size h_1 is reasonable. The parallel efficiency of our algorithm is above 50 % for all nine tests, and it increases with frequency f because smaller time steps are required to integrate the inhomogeneity accurately. We finally note that for large-scale computations our algorithm could also be used to further speed up a saturated space parallelization (e.g., by domain decomposition).

Acknowledgment: I am grateful to Martin J. Gander for many helpful discussions and valuable comments.

References

[1] I. P. Gavrilyuk and V. L. Makarov. Exponentially convergent algorithms for the operator exponential with applications to inhomogeneous problems in Banach spaces. *SIAM J. Numer. Anal.*, 43:2144–2171, 2005.

[2] G. H. Golub and C. F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, Baltimore, MD, 3rd edition, 1996.

[3] S. Güttel. *Rational Krylov Methods for Operator Functions*. PhD thesis, Institut für Numerische Mathematik und Optimierung der Technischen Universität Bergakademie Freiberg, 2010.

Table 1. Serial and parallel performance with $p = 4$ processors for a heat equation with diffusion coefficient α and source-term frequency f.

α	f	serial		parallel			efficiency
		τ_0	error	τ_1	τ_2	error	
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 %

[4] N. J. Higham. *Functions of Matrices. Theory and Computation*. SIAM, Philadelphia, PA, 2008.

[5] C. Moler and C. F. Van Loan. Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later. *SIAM Rev.*, 45:3–39, 2003.

[6] I. Moret and P. Novati. RD-rational approximations of the matrix exponential. *BIT*, 44:595–615, 2004.

[7] D. L. Powers and R. Jeltsch. Problem 74-5: On the norm of a matrix exponential. *SIAM Rev.*, 17:174–176, 1975.

[8] D. Sheen, I. H. Sloan, and V. Thomée. A parallel method for time discretization of parabolic equations based on Laplace transformation and quadrature. *IMA Journal of Numerical Analysis*, 23:269–299, 2003.

[9] J. van den Eshof and M. Hochbruck. Preconditioning Lanczos approximations to the matrix exponential. *SIAM J. Sci. Comput.*, 27:1438–1457, 2006.

[10] C. F. Van Loan. The sensitivity of the matrix exponential. *SIAM J. Numer. Anal.*, 14:971–981, 1977.