Krylov-enhanced parallel integrators for linear problems

Martin J. Gander Stefan Güttel

Department of Mathematics
University of Geneva

AIMS Dresden, 2010

UNIVERSITÉ DE GENÈVE

Outline

1. Motivation
2. The Parareal Algorithm
3. Linear Problems
   - Exponential Integration
   - Chebyshev Method
   - Load Balancing
4. Numerical Example

Motivation

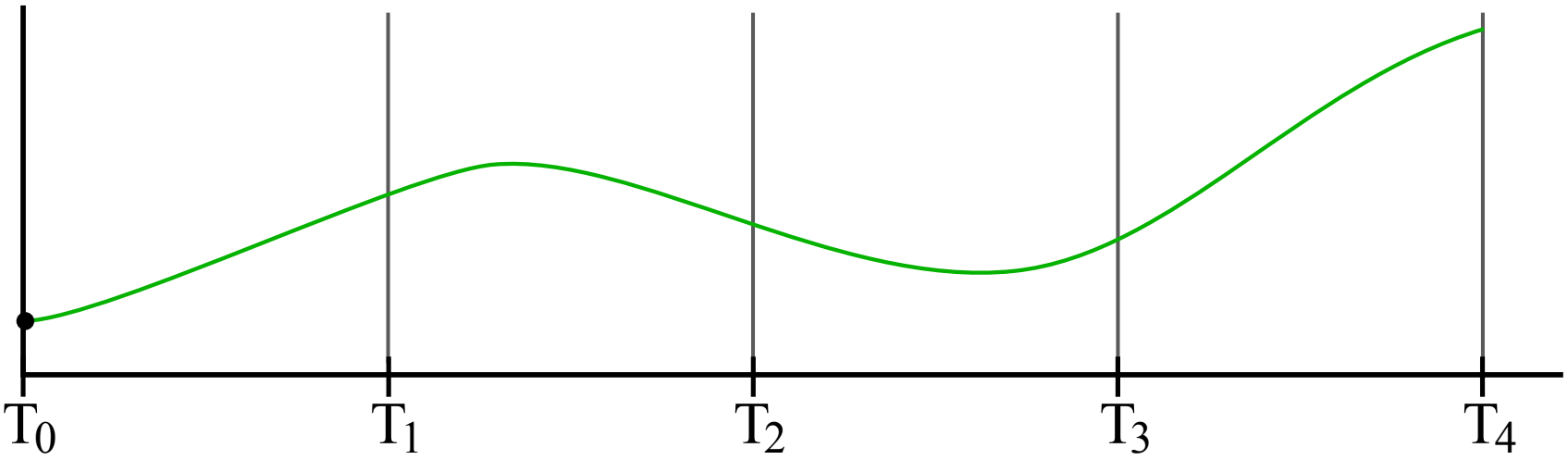
Want to solve the initial value problem

$$u'(t) = f(t, u), \quad t \in [0, T], \quad u(0) = u_0.$$

Usual approach:

Discretize the time interval at $0 = T_0 < T_1 < \cdots < T_N = T$ and compute approximations $U_n \approx u(T_n)$ by some time-stepping scheme

$$U_n = F(t, U_n, U_{n-1}, U_{n-2}, \ldots), \quad n = 1, 2, \ldots, N.$$



Motivation

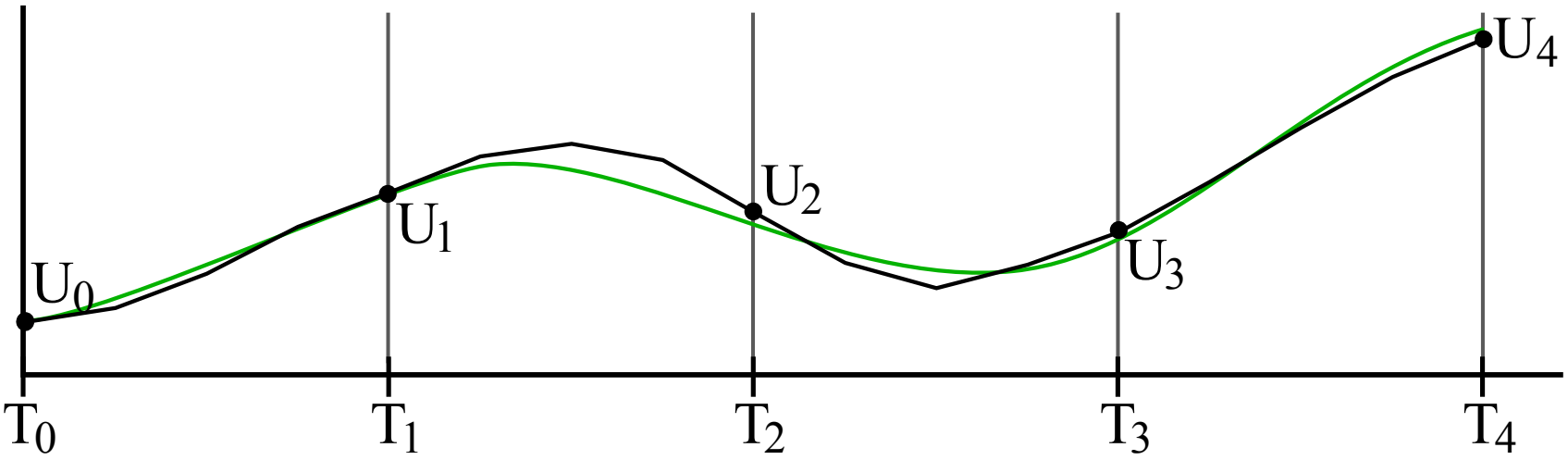
Want to solve the initial value problem

$$u'(t) = f(t, u), \quad t \in [0, T], \quad u(0) = u_0.$$

Usual approach:

Discretize the time interval at $0 = T_0 < T_1 < \cdots < T_N = T$ and compute approximations $U_n \approx u(T_n)$ by some time-stepping scheme

$$U_n = F(t, U_n, U_{n-1}, U_{n-2}, \ldots), \quad n = 1, 2, \ldots, N.$$



Time-parallel algorithms aim at computing approximations U_n^k without having obtained all previous $U_1^k, \ldots, U_{n-1}^k$ to fine accuracy.

Such algorithms are useful

- as another way toward more parallelism (sometimes as the only way),
- in realtime forecast applications (weather prediction),
- in realtime control problems (steering a rocket).

Time-parallel algorithms aim at computing approximations U_n^k without having obtained all previous $U_1^k, \ldots, U_{n-1}^k$ to fine accuracy.

Such algorithms are useful

- as another way toward more parallelism (sometimes as the only way),
- in realtime forecast applications (weather prediction),
- in realtime control problems (steering a rocket).

The Parareal Algorithm (Parallel Realtime)

Introduced by [Lions, Maday & Turincini 01],
further studied by Bal, Chandesris, Farhat, Gander,
Hairer, Petcu, Minion, Vandewalle, Williams, Wu, ...

Idea: Partition the time domain $\Omega = [0, T]$ into N subdomains

$$\Omega_n = [T_{n-1}, T_n], \quad n = 1, 2, \ldots, N.$$

Also define

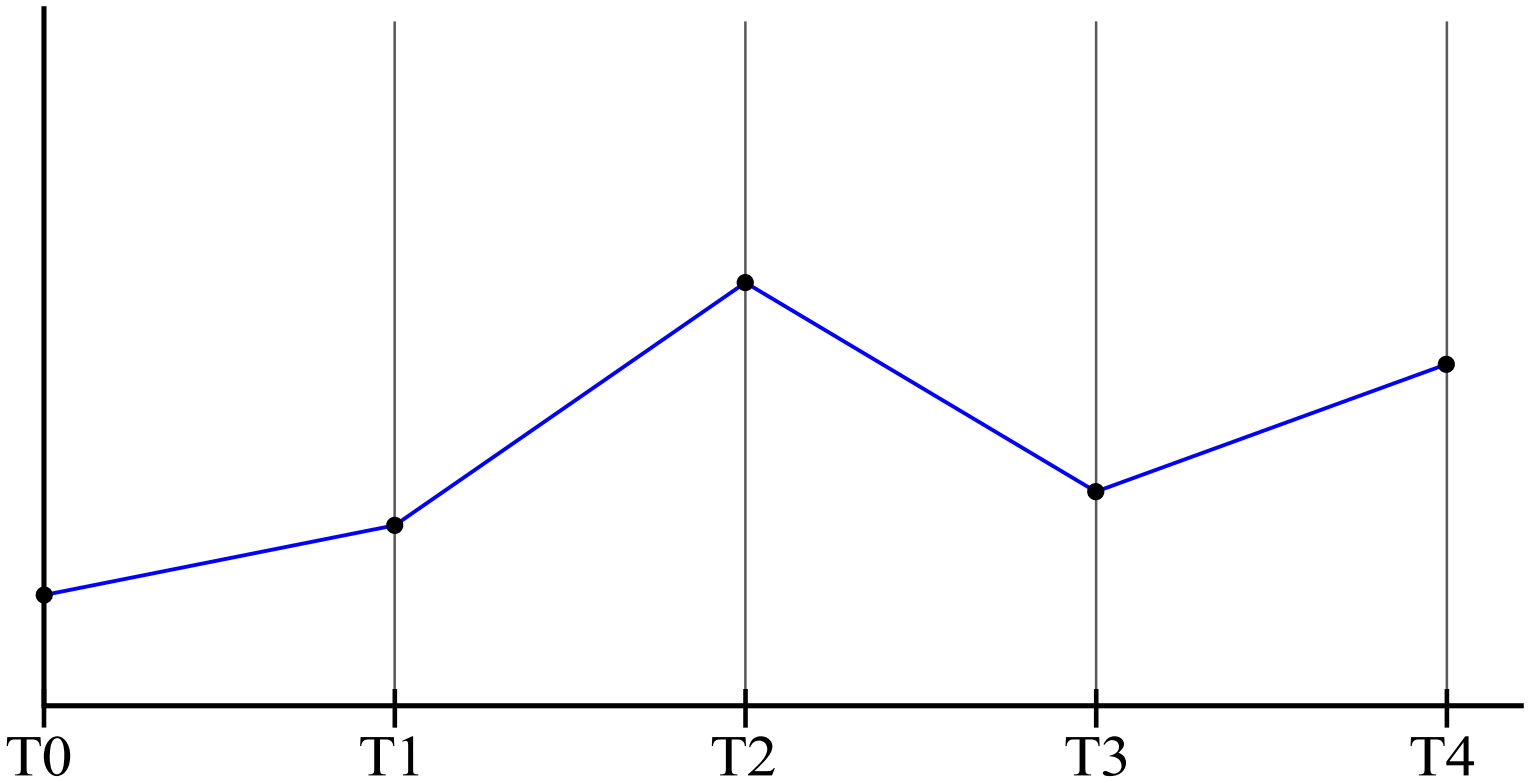
- a coarse propagator $\mathcal{G}(T_n, T_{n-1}, x)$, and
- a fine propagator $\mathcal{F}(T_n, T_{n-1}, x)$,

both of which return approximations U_n of $U(T_n)$ satisfying

$$U'(t) = f(t, U(t)), \quad U(T_{n-1}) = x.$$

In iteration $k = 1$ propagate coarsely

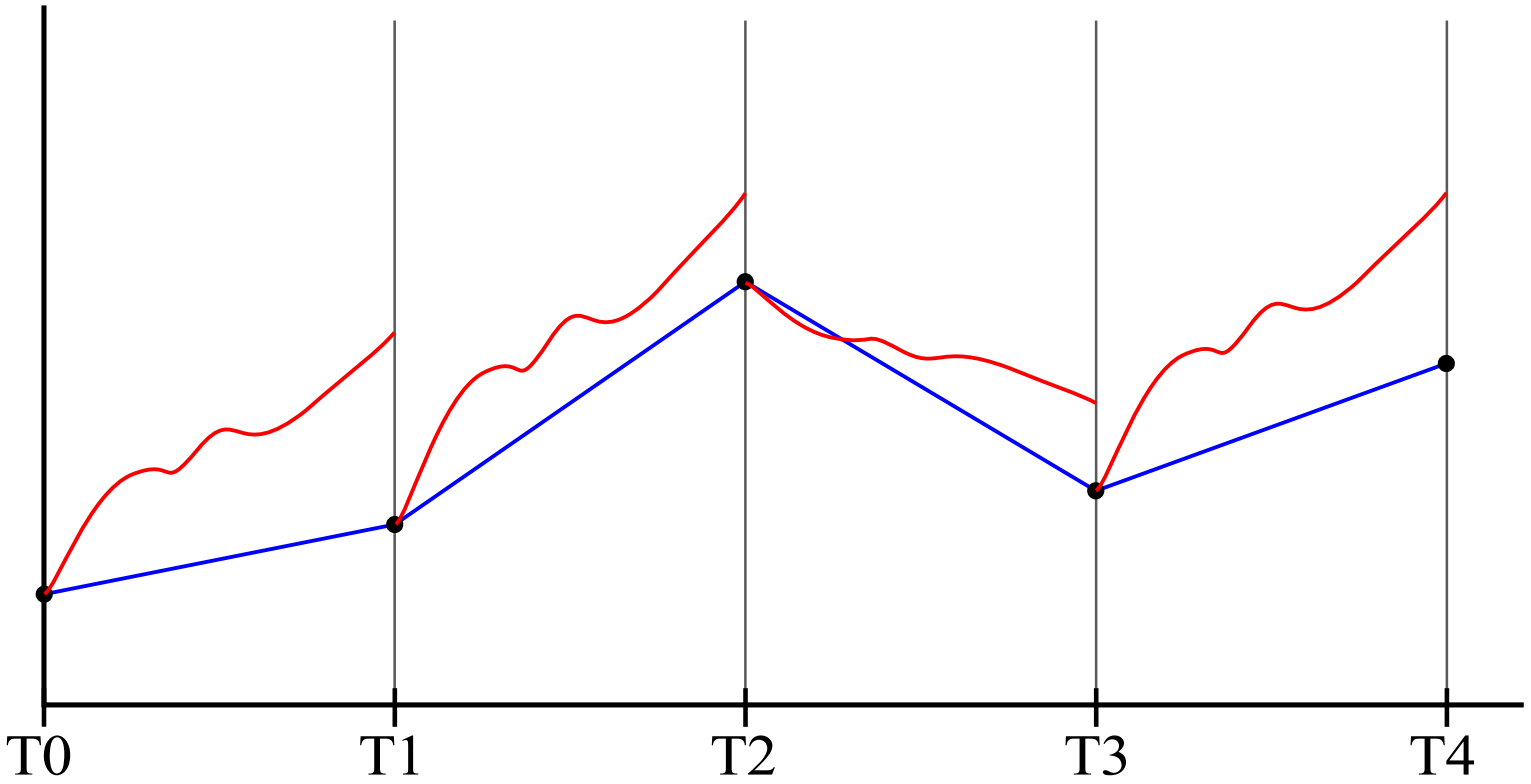
$$U_0^1 = u_0$$
$$U_n^1 = \mathcal{G}(T_n, T_{n-1}, U_{n-1}^1), \quad n = 1, 2, \ldots, N.$$



Now, an initial condition for the fine propagator is available on all subdomains Ω_n, hence

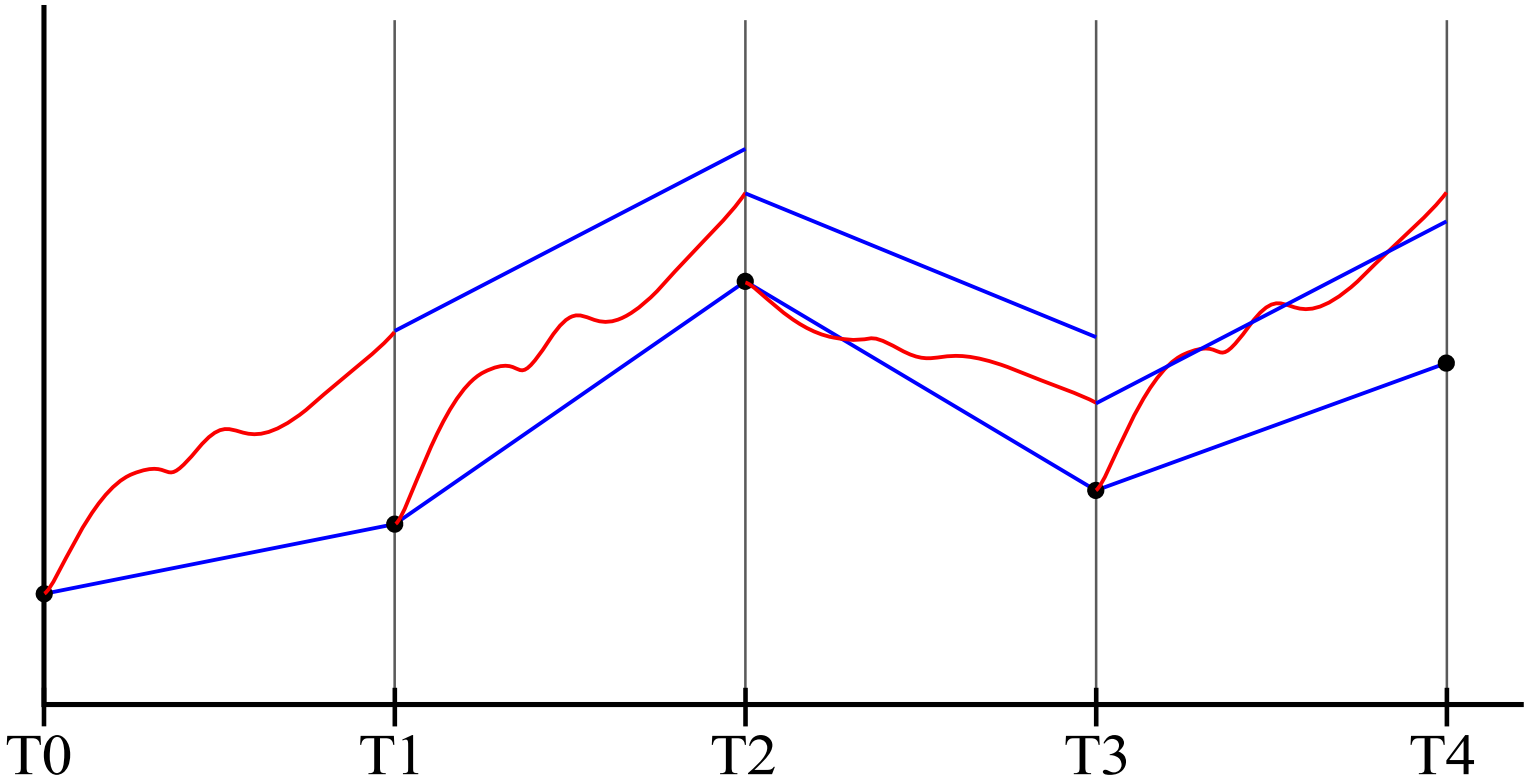
$$\mathcal{F}(T_n, T_{n-1}, U_{n-1}^1), \quad n = 1, 2, \ldots, N,$$

can be computed in parallel.



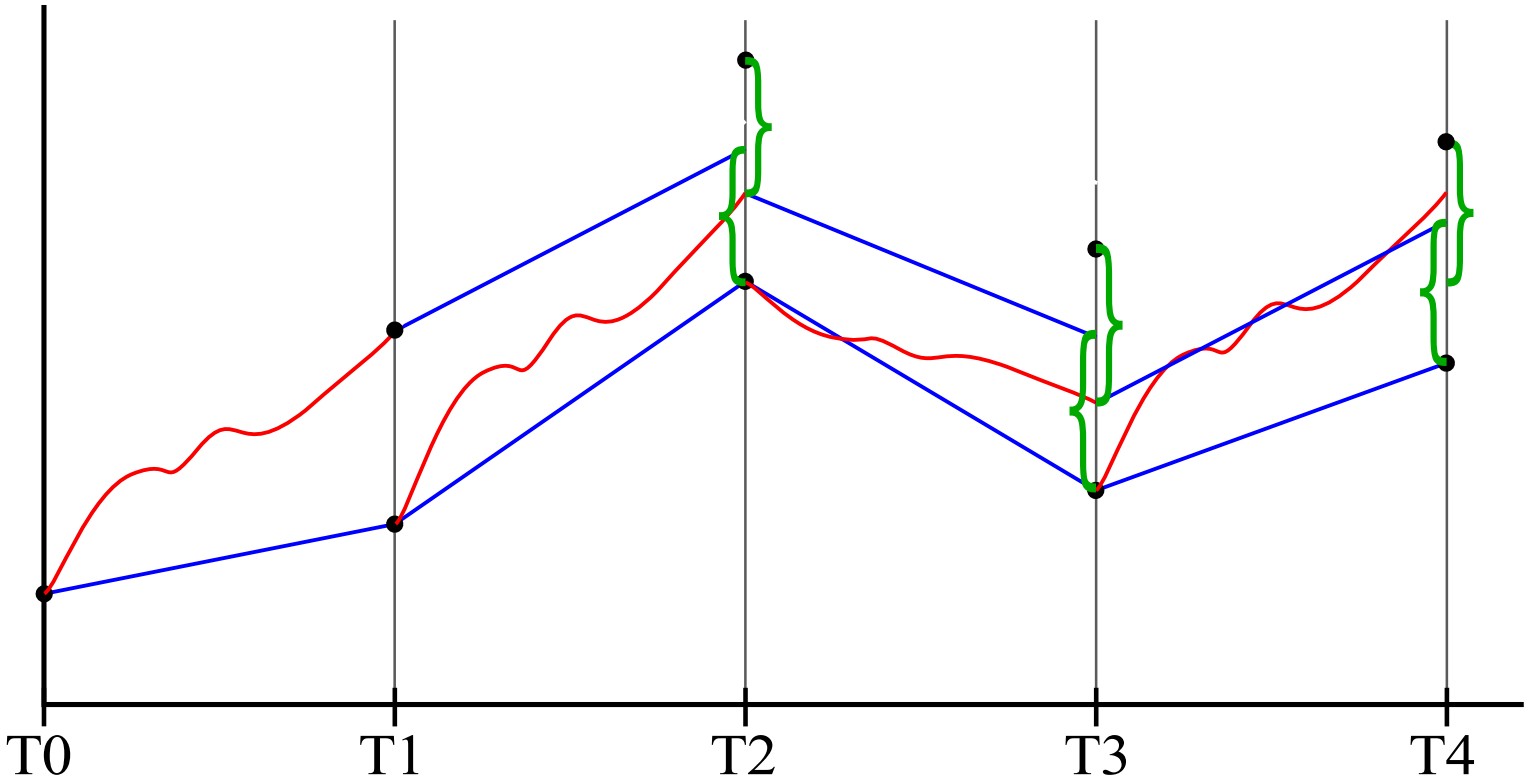
This yields an initial guess better than U_{n-1}^k, which we propagate coarsely, resulting in a correction ($n = 1, 2, \ldots, N$)

$$U_n^{k+1} = \mathcal{F}(T_n, T_{n-1}, U_{n-1}^k) + \mathcal{G}(T_n, T_{n-1}, U_{n-1}^{k+1}) - \mathcal{G}(T_n, T_{n-1}, U_{n-1}^k).$$



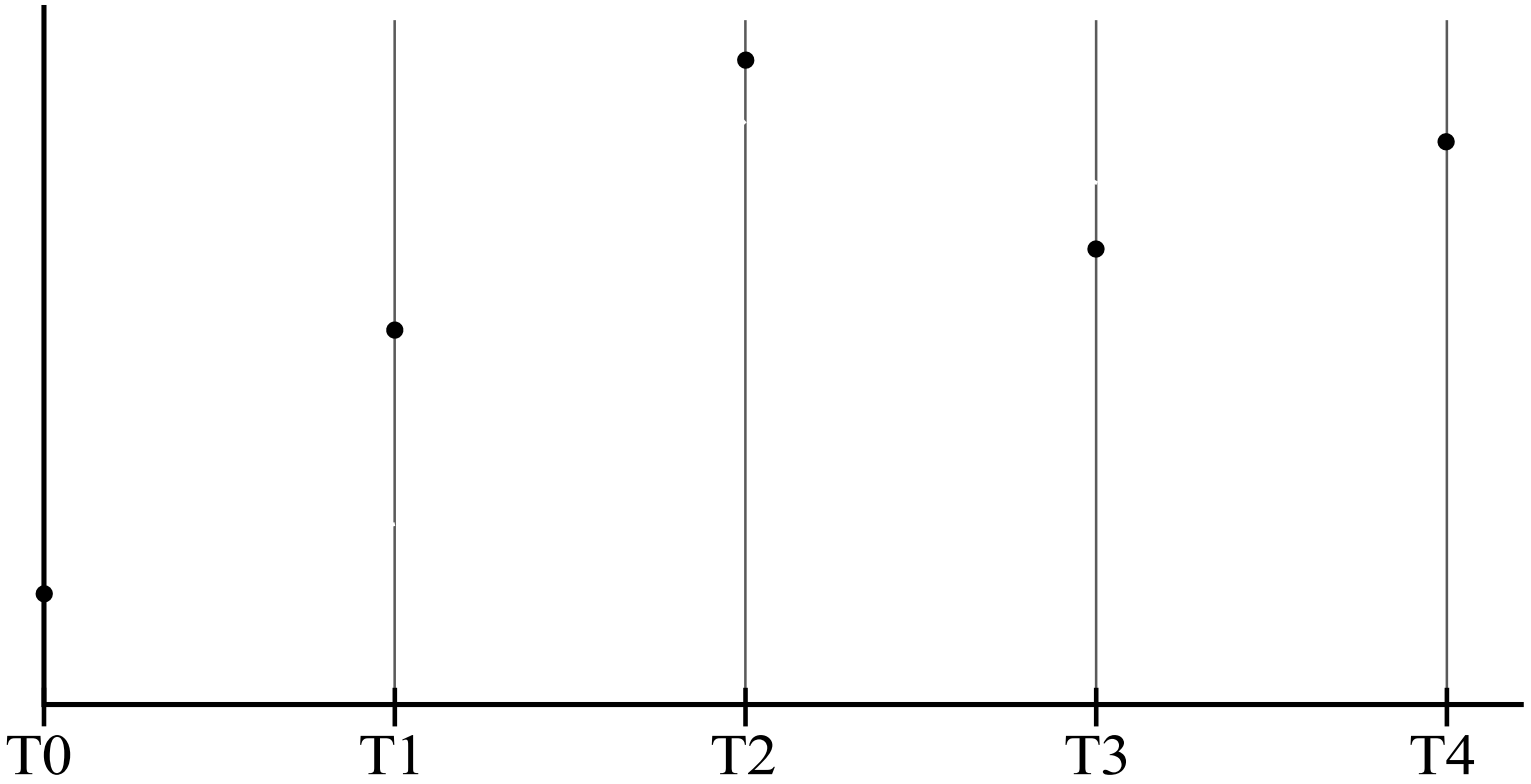
This yields an initial guess better than U_{n-1}^k, which we propagate coarsely, resulting in a correction ($n = 1, 2, \ldots, N$)

$$U_n^{k+1} = \mathcal{F}(T_n, T_{n-1}, U_{n-1}^k) + \mathcal{G}(T_n, T_{n-1}, U_{n-1}^{k+1}) - \mathcal{G}(T_n, T_{n-1}, U_{n-1}^k).$$



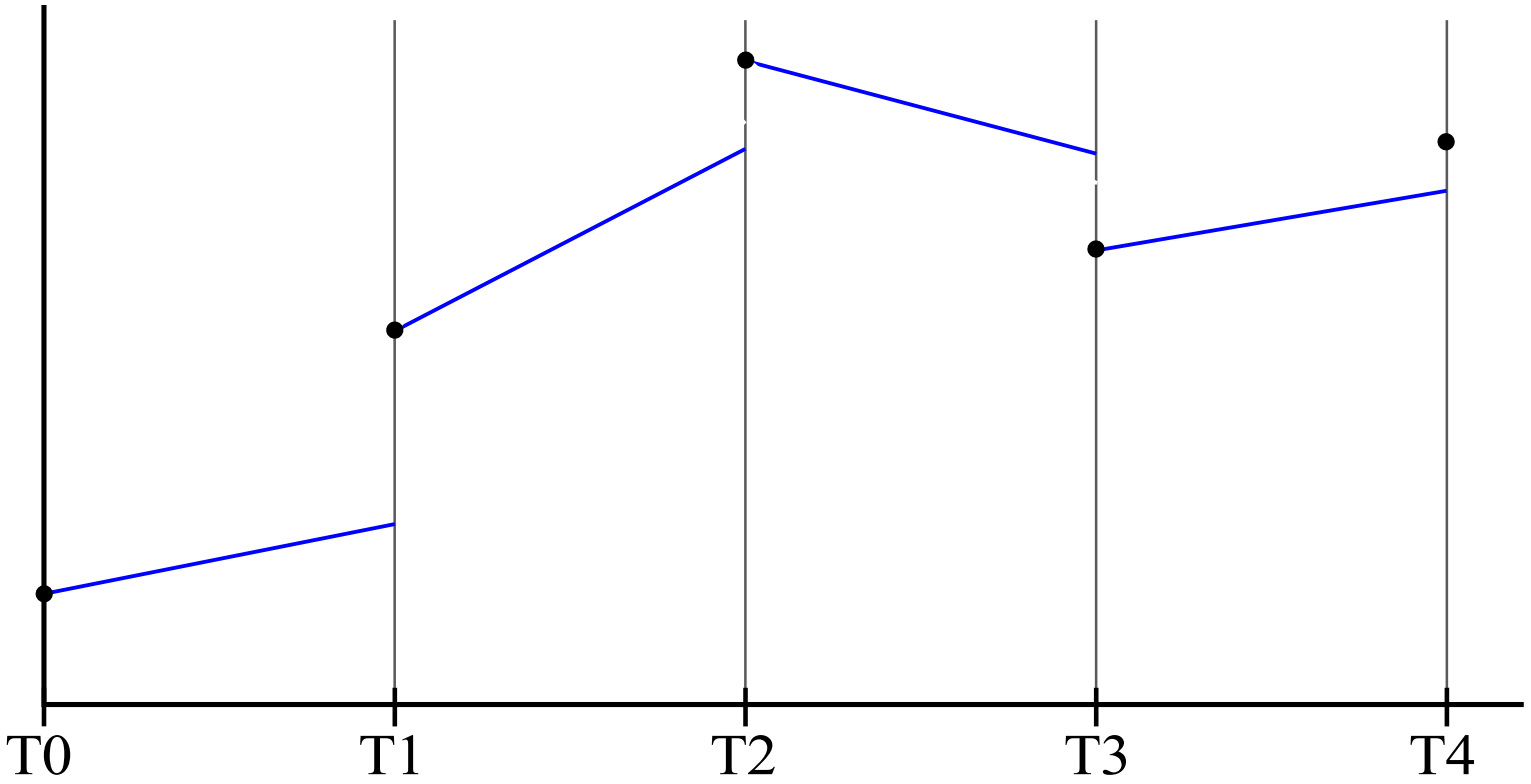
This yields an initial guess better than U_{n-1}^k, which we propagate coarsely, resulting in a correction ($n = 1, 2, \ldots, N$)

$$U_n^{k+1} = \mathcal{F}(T_n, T_{n-1}, U_{n-1}^k) + \mathcal{G}(T_n, T_{n-1}, U_{n-1}^{k+1}) - \mathcal{G}(T_n, T_{n-1}, U_{n-1}^k).$$



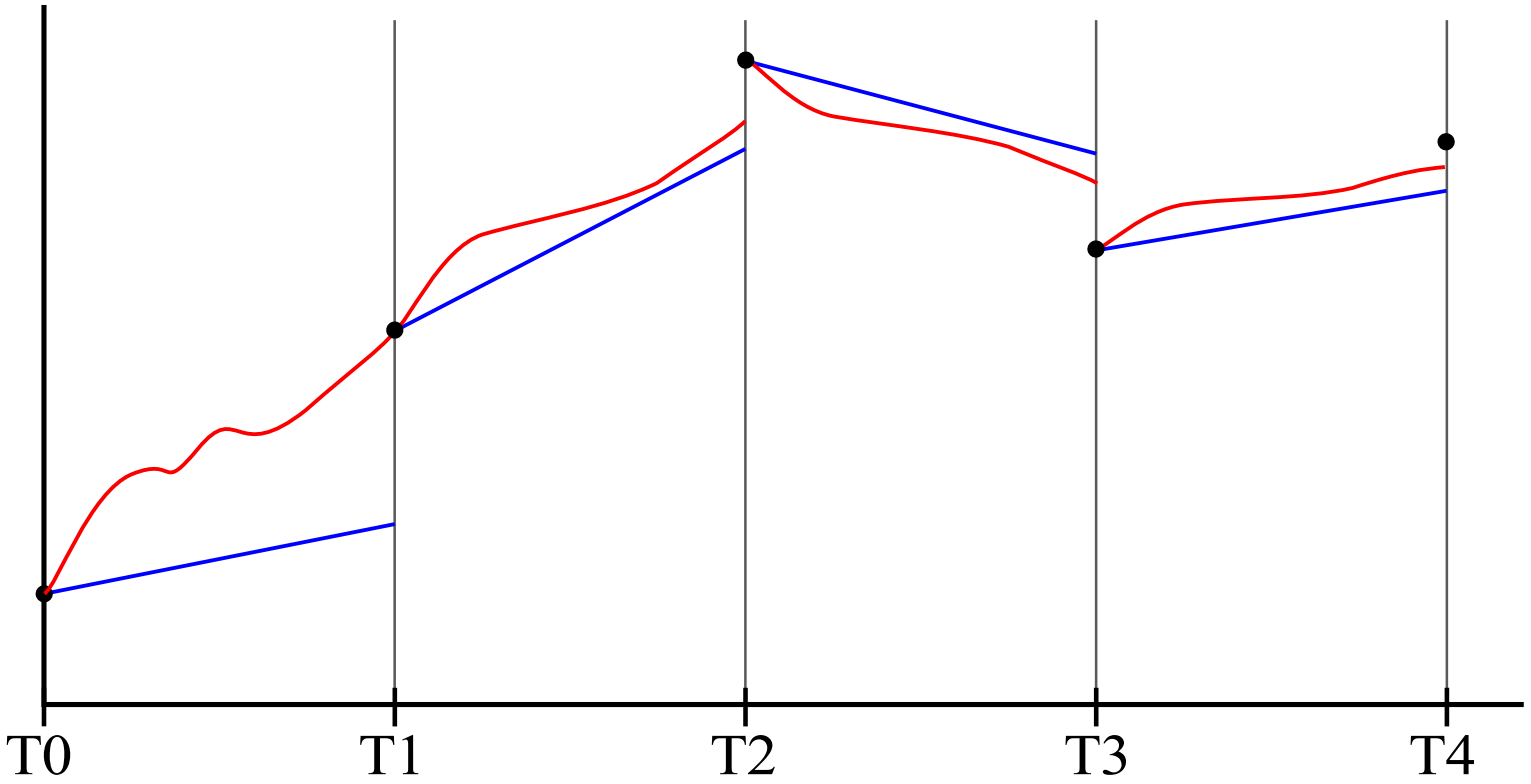
This yields an initial guess better than U_{n-1}^k, which we propagate coarsely, resulting in a correction ($n = 1, 2, \ldots, N$)

$$U_n^{k+1} = \mathcal{F}(T_n, T_{n-1}, U_{n-1}^k) + \mathcal{G}(T_n, T_{n-1}, U_{n-1}^{k+1}) - \mathcal{G}(T_n, T_{n-1}, U_{n-1}^k).$$



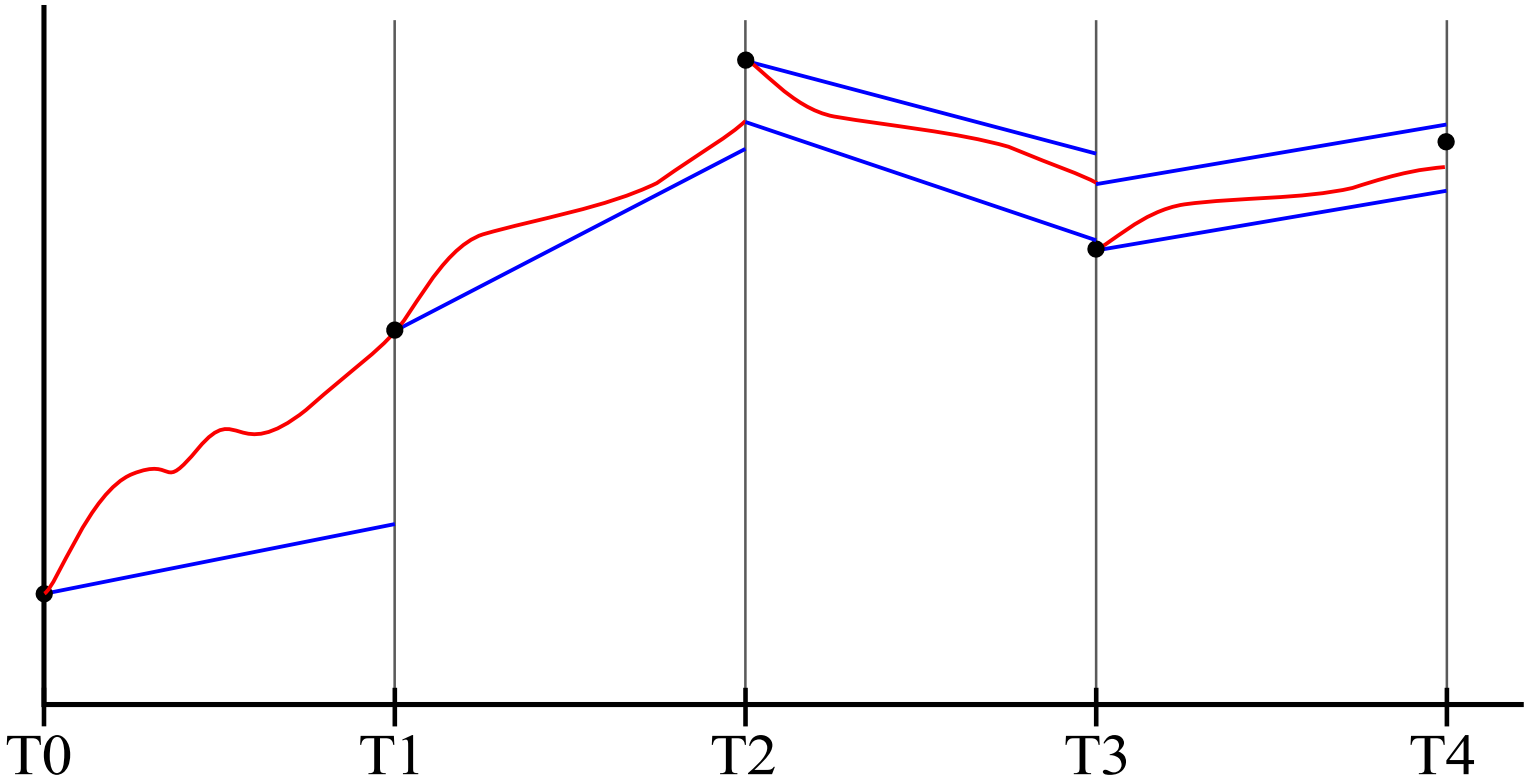
This yields an initial guess better than U_{n-1}^k, which we propagate coarsely, resulting in a correction ($n = 1, 2, \ldots, N$)

$$U_n^{k+1} = \mathcal{F}(T_n, T_{n-1}, U_{n-1}^k) + \mathcal{G}(T_n, T_{n-1}, U_{n-1}^{k+1}) - \mathcal{G}(T_n, T_{n-1}, U_{n-1}^k).$$



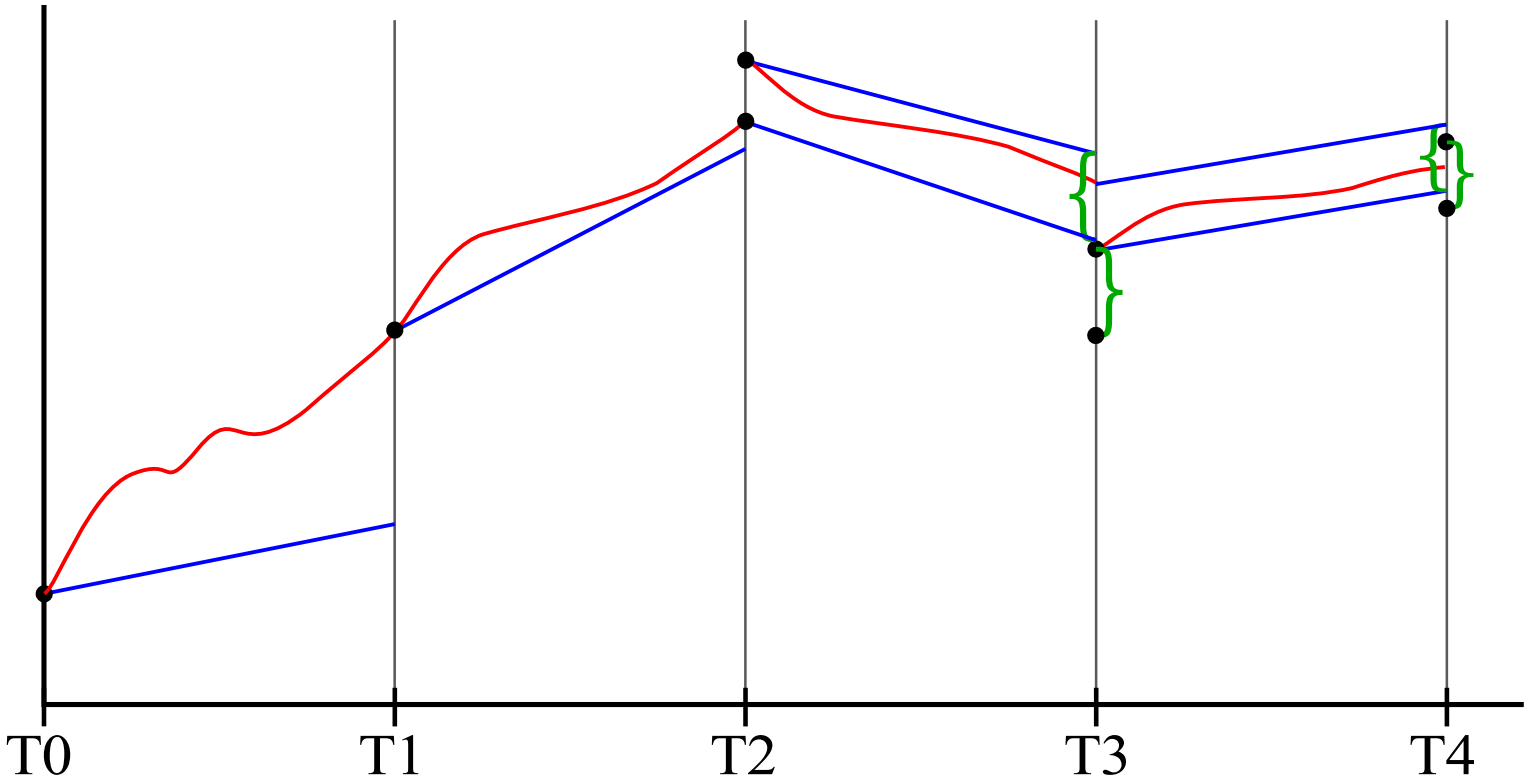
This yields an initial guess better than U_{n-1}^k, which we propagate coarsely, resulting in a correction ($n = 1, 2, \ldots, N$)

$$U_n^{k+1} = \mathcal{F}(T_n, T_{n-1}, U_{n-1}^k) + \mathcal{G}(T_n, T_{n-1}, U_{n-1}^{k+1}) - \mathcal{G}(T_n, T_{n-1}, U_{n-1}^k).$$



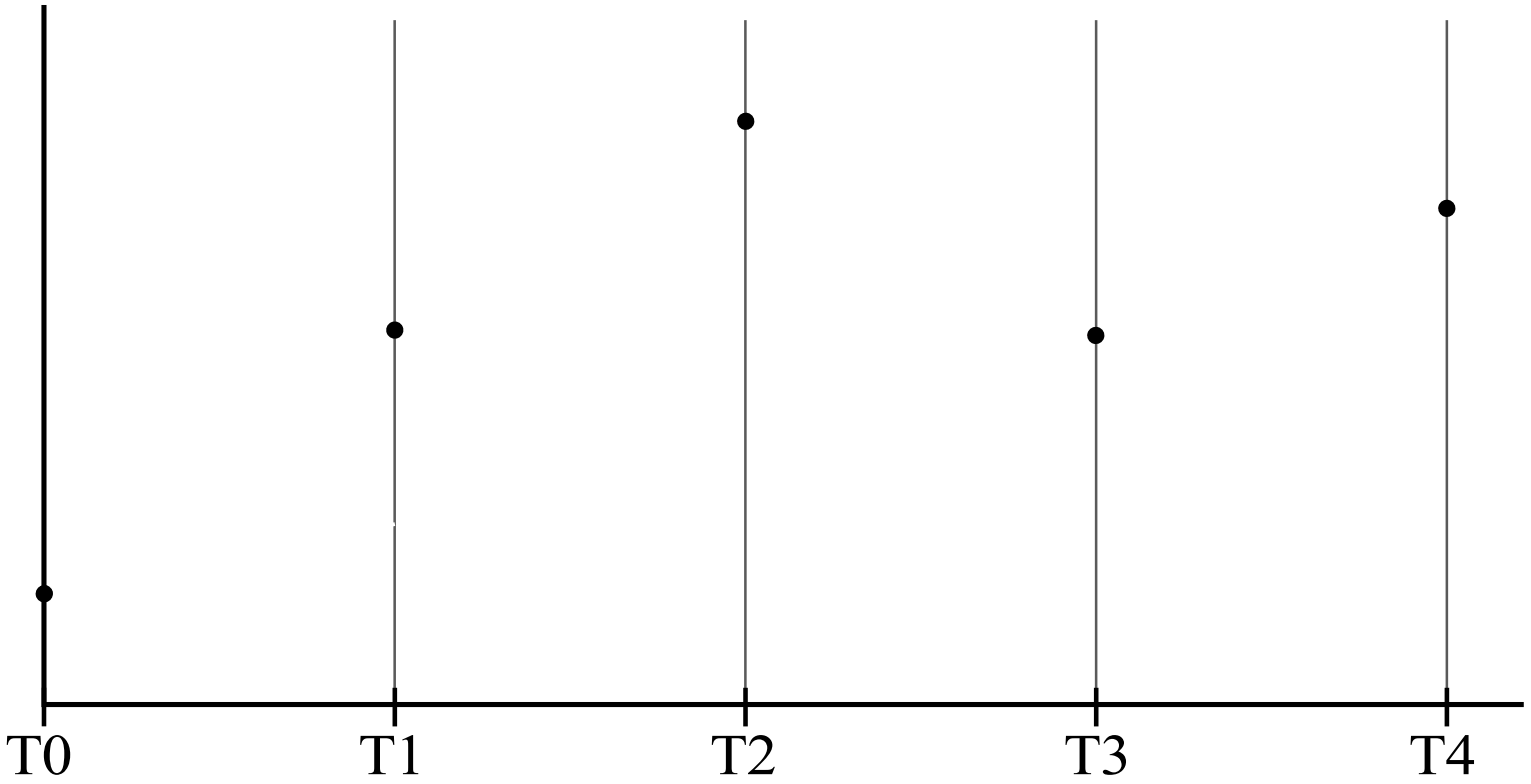
This yields an initial guess better than U_{n-1}^k, which we propagate coarsely, resulting in a correction ($n = 1, 2, \ldots, N$)

$$U_n^{k+1} = \mathcal{F}(T_n, T_{n-1}, U_{n-1}^k) + \mathcal{G}(T_n, T_{n-1}, U_{n-1}^{k+1}) - \mathcal{G}(T_n, T_{n-1}, U_{n-1}^k).$$



This yields an initial guess better than U_{n-1}^k, which we propagate coarsely, resulting in a correction ($n = 1, 2, \ldots, N$)

$$U_n^{k+1} = \mathcal{F}(T_n, T_{n-1}, U_{n-1}^k) + \mathcal{G}(T_n, T_{n-1}, U_{n-1}^{k+1}) - \mathcal{G}(T_n, T_{n-1}, U_{n-1}^k).$$



Convergence of Parareal

Theorem (Gander & Hairer 08)

Assume that all $T_n - T_{n-1} = \Delta T$, $\mathcal{F}$ *is an exact propagator and*

$$\mathcal{F}(T_n, T_{n-1}, x) - \mathcal{G}(T_n, T_{n-1}, x) = c_{p+1}(x)\Delta T^{p+1} + c_{p+2}(x)\Delta T^{p+2} + \cdots$$

with all $c_{p+j}(x)$ *being continuously differentiable. Moreover, assume that*

$$\|\mathcal{G}(t + \Delta T, t, x) - \mathcal{G}(t + \Delta T, t, y)\| \leq (1 + C_2\Delta T)\|x - y\|.$$

Then

$$\|u(T_n) - U_n^k\| \leq \frac{C_3}{C_1} \frac{(C_1 T_n)^{k+1}}{(k+1)!} e^{C_2(T_n - T_{k+1})} \Delta T^{p(k+1)}.$$

Linear Problems

We now consider

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

with $\mathbf{A} \in \mathbb{C}^{d\times d}$ and $g : [0, T] \to \mathbb{C}^d$.

Aim: Exploit linearity to speed up convergence of parareal.

Idea: [Farhat et al. 06] [Gander & Petcu 08] On each interval solve

$$\begin{aligned} V_n'(t) &= \mathbf{A}V_n(t) + g(t), & t \in [T_{n-1}, T_n], \quad V_n(T_{n-1}) &= 0, \\ W_n^{k\prime}(t) &= \mathbf{A}W_n^k(t), & t \in [T_{n-1}, T_n], \quad W_n^k(T_{n-1}) &= U_{n-1}^k, \\ U_n^k(t) &= V_n(t) + W_n^k(t). \end{aligned}$$

The first subproblem needs to be solved only once.
The second subproblem is linear homogeneous, but depends on U_{n-1}^k.

At iteration k of parareal, on interval $[T_{n-1}, T_n]$, we have to solve

$$W_n^{k\prime}(t) = \mathbf{A} W_n^k(t), \quad t \in [T_{n-1}, T_n], \quad W_n^k(T_{n-1}) = U_{n-1}^k$$

with the homogeneous coarse and fine propagator ($\mathcal{G}^H$ and $\mathcal{F}^H$).

At this point, fine evaluations for previous initial conditions U_i^j are available:

$$F_i^j = \mathcal{F}^H(T_i, T_{i-1}, U_i^j).$$

If all U_i^j and F_i^j are collected columnwise in matrices $\mathbf{U}$ and $\mathbf{F}$, then

$$\mathbf{F}\mathbf{U}^\dagger U_{n-1}^k + \mathcal{G}^H(T_i, T_{i-1}, (I - \mathbf{U}\mathbf{U}^\dagger) U_{n-1}^k)$$

is an improvement to the coarse propagator $\mathcal{G}^H(T_i, T_{i-1}, U_{n-1}^k)$.

Remarks

- In a practical implementation, one orthonormalizes the columns of $\mathbf{U}$ such that $\mathbf{U}\mathbf{U}^\dagger = \mathbf{U}\mathbf{U}^*$. This is done by a modified Gram–Schmidt procedure to easily append new initial conditions to $\mathbf{U}$.
- One has to prevent loss of orthogonality by reorthogonalization, otherwise $\mathbf{U}\mathbf{U}^\dagger \neq \mathbf{U}\mathbf{U}^*$ and the method becomes unstable.
- The matrices $\mathbf{U}$ and $\mathbf{F}$ increase in size with every parareal iteration. Orthogonalization, projection, and storage become expensive.

Remarks

- In a practical implementation, one orthonormalizes the columns of $\mathbf{U}$ such that $\mathbf{U}\mathbf{U}^\dagger = \mathbf{U}\mathbf{U}^*$. This is done by a modified Gram–Schmidt procedure to easily append new initial conditions to $\mathbf{U}$.
- One has to prevent loss of orthogonality by reorthogonalization, otherwise $\mathbf{U}\mathbf{U}^\dagger \neq \mathbf{U}\mathbf{U}^*$ and the method becomes unstable.
- The matrices $\mathbf{U}$ and $\mathbf{F}$ increase in size with every parareal iteration. Orthogonalization, projection, and storage become expensive.

Claim: The columns of $\mathbf{U}$ and $\mathbf{F}$ are contained in a Krylov space generated by $\mathbf{A}$ because ...

Virtually every method available to integrate

$$W'(t) = \mathbf{A}W(t), \quad t \in [T_0, T_1], \quad W(T_0) = W^{[0]}$$

can be written as

$$W^{[m]} = p_m(h\mathbf{A})W^{[0]} \approx W(T_1)$$

for a polynomial (or rational function) $p_m(z)$.

Examples

- Forward Euler: $W^{[m]} = W^{[m-1]} + h\mathbf{A}W^{[m-1]} \Rightarrow p_m(z) = (1+z)^m$.
- Backward Euler: $W^{[m]} = W^{[m-1]} + h\mathbf{A}W^{[m]} \Rightarrow p_m(z) = (1-z)^{-m}$.
- Runge–Kutta: $p_m(z) = \left(1 + z + \frac{1}{2}z^2 + \frac{1}{6}z^3 + \frac{1}{24}z^4\right)^m$.
- SDC: interpolating polynomial.

Hence: At any iteration of parareal the degree of $W^{[m]}$ as a function in $\mathbf{A}$ will not exceed the degree we could generate with 1 processor!

Outline

1. Motivation
2. The Parareal Algorithm
3. Linear Problems
   - Exponential Integration
   - Chebyshev Method
   - Load Balancing
4. Numerical Example

Exponential Integration (Paraexp)

We now decompose

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

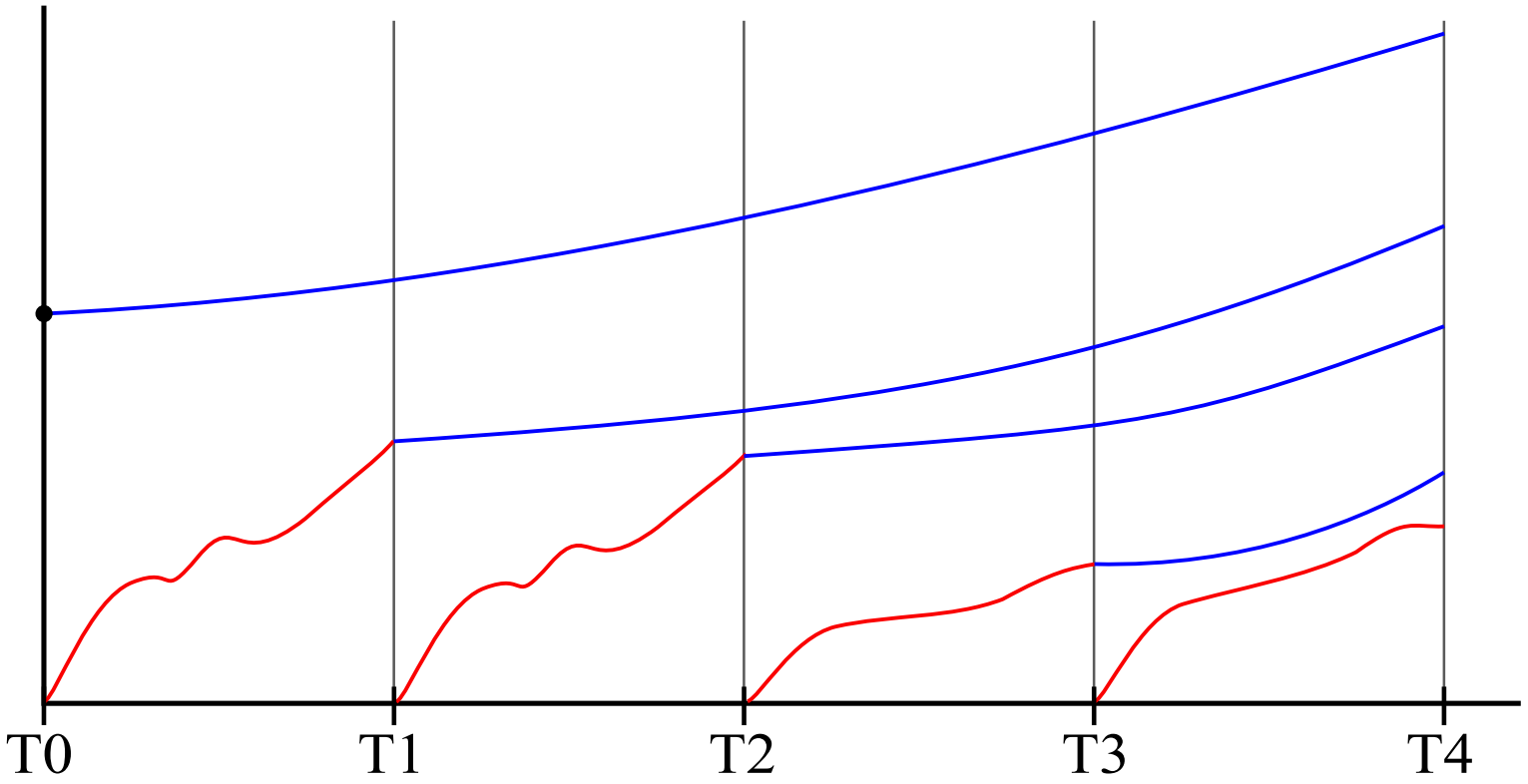
into N decoupled problems

$$\begin{aligned} V_n'(t) &= \mathbf{A}V_n(t) + g(t), & t \in [T_{n-1}, T_n], \quad V_n(T_{n-1}) &= 0, \\ W_n'(t) &= \mathbf{A}W_n(t), & t \in [T_{n-1}, T_N], \quad W_n(T_{n-1}) &= V_{n-1}(T_{n-1}), \end{aligned}$$

where $V_0(T_0) = u_0$, such that

$$u(t) = \sum_{n=1}^{N} V_n(t) + W_n(t).$$

If processor $P(n)$ computes $V_{n-1}(t)$ and $W_n(t)$, this summation is the only communication point.



1. On processor $P(n)$ compute $V_{n-1}(T_{n-1}) = \mathcal{F}(T_{n-1}, T_{n-2}, 0)$.
2. On processor $P(n)$ compute $W_n(t) = e^{(t-T_{n-1})\mathbf{A}} V_{n-1}(T_{n-1})$.
3. Summation $U(T_j) = \sum_{n=1}^{N} V_n(T_j) + W_n(T_j)$ at desired T_j.

To compute the matrix exponential in

$$W_n(t) = e^{(t-T_{n-1})\mathbf{A}} v, \quad v = V_{n-1}(T_{n-1}),$$

we can directly use a polynomial Krylov method to select approximations from an m-dimensional polynomial Krylov space

$$W_n(t) \approx W_n^m(t) \in \mathcal{K}_m(\mathbf{A}, v) := \text{span}\{v, \mathbf{A}v, \mathbf{A}^2 v, \ldots, \mathbf{A}^{m-1} v\}.$$

In what follows: **Chebyshev method**

[Druskin & Knizhnerman 89] [Schaefer 90]

Outline

1. Motivation
2. The Parareal Algorithm
3. Linear Problems
   - Exponential Integration
   - Chebyshev Method
   - Load Balancing
4. Numerical Example

Chebyshev Method

Aim: Approximation of $e^{\tau \mathbf{A}} v$ for symmetric $\mathbf{A} \in \mathbb{C}^{d \times d}$, $\Lambda(\mathbf{A}) \in [-1, 1]$, without computing $e^{\tau \mathbf{A}}$.

The Chebyshev method is based on the polynomial expansion

$$e^{\tau x} = \sum_{j=0}^{\infty} \gamma_j C_j(x), \quad x \in [-1, 1],$$

where $C_0 \equiv 1$, $C_1(x) = x$, $C_j(x) = 2xC_{j-1}(x) - C_{j-2}(x)$.

Chebyshev polynomials $C_j(x)$ are orthogonal w.r.t. the inner product

$$\langle f(x), g(x) \rangle := \int_{-1}^{1} \frac{f(x)g(x)}{\sqrt{1-x^2}} \, \mathrm{d}x$$

and the γ_j are the coordinates of $e^{\tau x}$, i.e.,

$$\gamma_j := \langle e^{\tau x}, C_j(x) \rangle.$$

The truncated Chebyshev sum

$$S_m(x) := \sum_{j=0}^{m-1} \gamma_j C_j(x), \quad x \in [-1,1],$$

is the best approximation of $f(x) = e^{\tau x}$ from $\mathcal{P}_{m-1} \subset L^2_w([-1,1])$, and $\|e^{\tau x} - S_m(x)\|_{L^2_w([-1,1])} \to 0$ superlinearly as $m \to \infty$.

Note that $\|C_j\|_\infty := \max\{|C_j(x)| : x \in [-1,1]\} = 1$ and $C_j(1) = 1$.

Moreover, we have $\gamma_j > 0$ for all j. Hence

$$\|e^{\tau x} - S_m(x)\|_\infty = \left\| \sum_{j=m}^{\infty} \gamma_j C_j(x) \right\|_\infty = \sum_{j=m}^{\infty} \gamma_j C_j(1) = e - S_m(1).$$

A practical method for computing $S_m(\mathbf{A})v \approx e^{\tau \mathbf{A}} v$ is

- Initialize $C_0 = v$, $C_1 = \mathbf{A}v$.
- Set $S_2 = \gamma_0 C_0 + \gamma_1 C_1$.
- For $j = 2, \ldots, m-1$ compute $C_j = 2\mathbf{A}C_{j-1} - C_{j-2}$ and set $S_{j+1} = S_j + \gamma_j C_j$.

A practical method for computing $S_m(\mathbf{A})v \approx e^{\tau \mathbf{A}} v$ is

- Initialize $C_0 = v$, $C_1 = \mathbf{A}v$.
- Set $S_2 = \gamma_0 C_0 + \gamma_1 C_1$.
- For $j = 2, \ldots, m-1$ compute $C_j = 2\mathbf{A}C_{j-1} - C_{j-2}$ and set $S_{j+1} = S_j + \gamma_j C_j$.

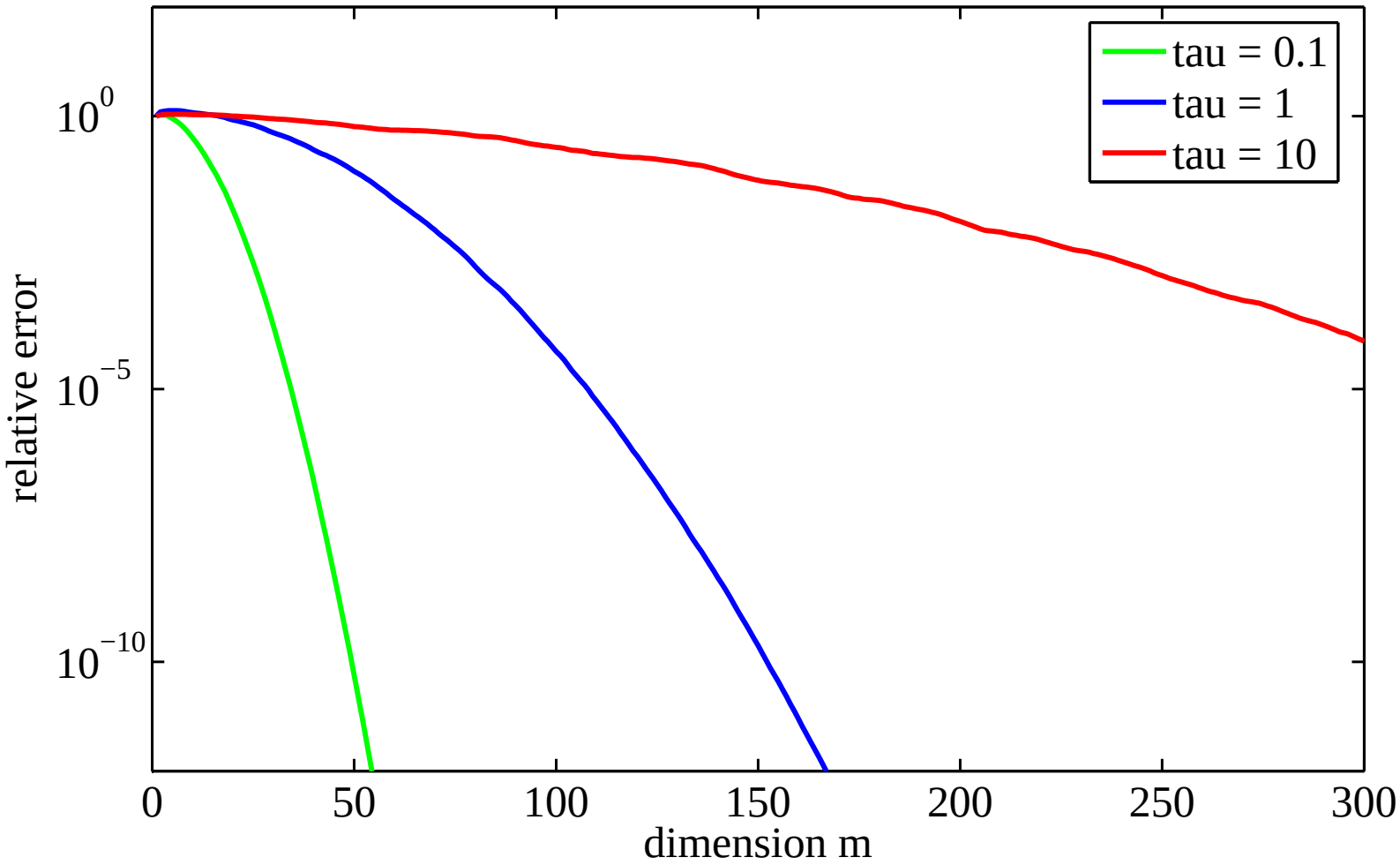
Remarks: Storage for only 3 vectors needed. We have the bound

$$\|e^{\tau \mathbf{A}} v - S_m(\mathbf{A})v\|_2 \leq \|e^{\tau x} - S_m(x)\|_\infty \|v\|_2 = (e - S_m(1))\|v\|_2.$$

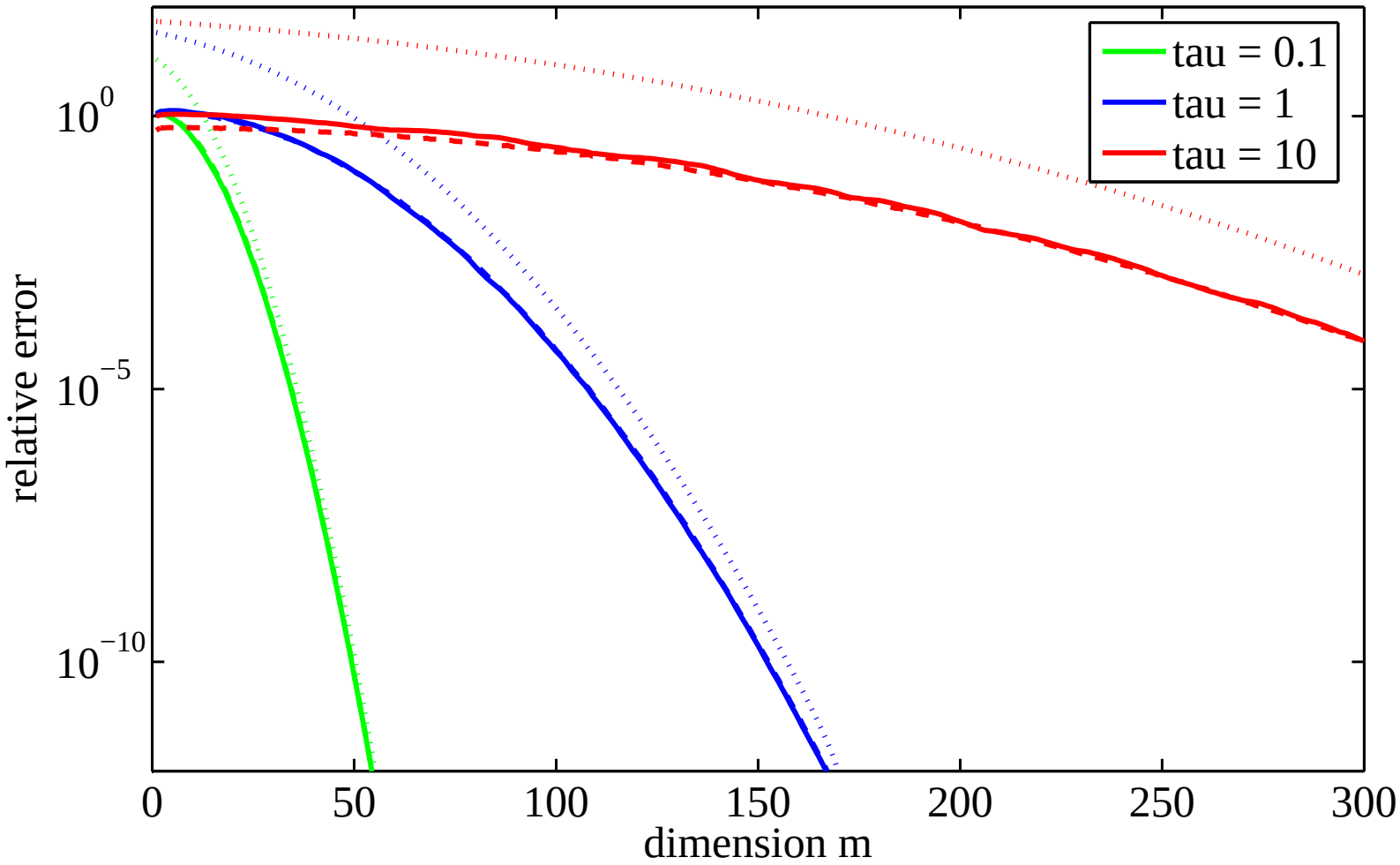
A practical error estimate is

$$\|e^{\tau \mathbf{A}} v - S_m(\mathbf{A})v\|_2 \approx \|\gamma_m C_m + \gamma_{m+1} C_{m+1}\|_2.$$

Example: $\mathbf{A} = \mathrm{diag}(-1000, -999, \ldots, 0)$, $v = \mathrm{randn}$, compute $e^{\tau \mathbf{A}} v$.



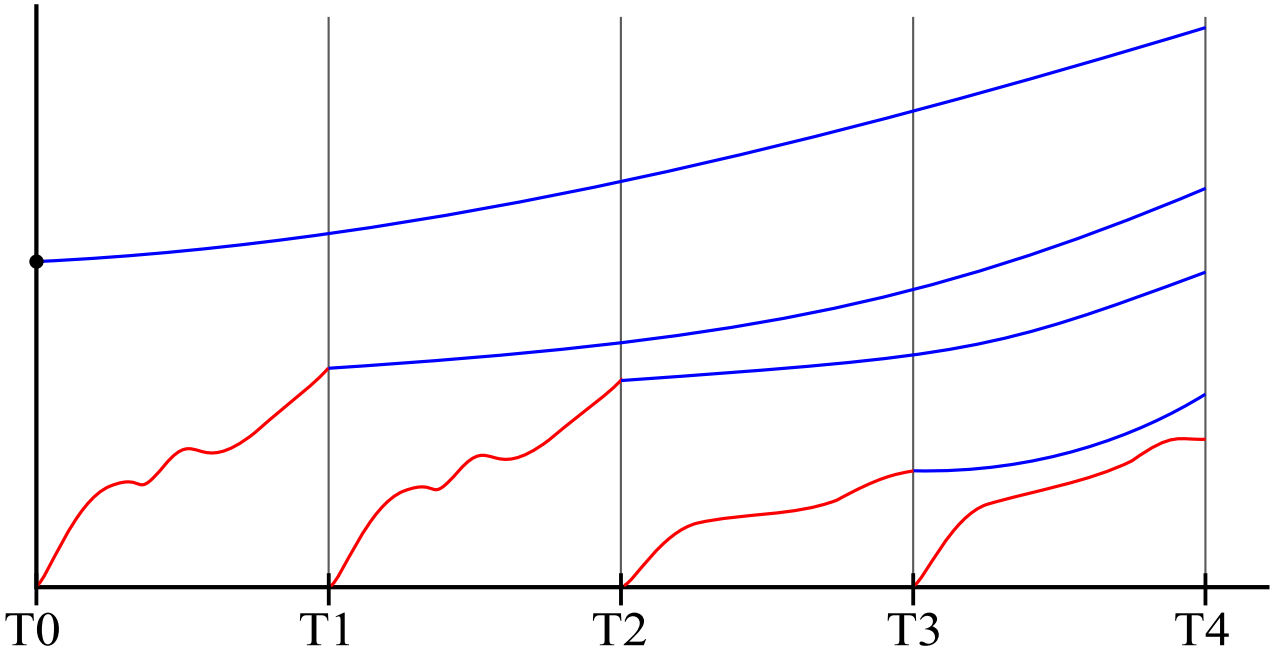
Example: $\mathbf{A} = \mathrm{diag}(-1000, -999, \ldots, 0)$, $v = \mathrm{randn}$, compute $e^{\tau \mathbf{A}} v$.



Outline

1. Motivation
2. The Parareal Algorithm
3. Linear Problems
   - Exponential Integration
   - Chebyshev Method
   - Load Balancing
4. Numerical Example

Load Balancing



1. On processor $P(n)$ compute $V_{n-1}(T_{n-1}) = \mathcal{F}(T_{n-1}, T_{n-2}, 0)$.
2. On processor $P(n)$ compute $W_n(t) = e^{(t-T_{n-1})\mathbf{A}} V_{n-1}(T_{n-1})$.
3. Summation $U(T_j) = \sum_{n=1}^{N} V_n(T_j) + W_n(T_j)$ at desired T_j.

Idea: Choose $T_1, \ldots, T_{N-1}$ such that steps 1 & 2 end synchronously.

Numerical Example

1D heat equation with homogeneous Dirichlet boundary condition and "oscillating hat" source (half-width $w = 0.05$, height $h = 50$, frequency $f = 23$):

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

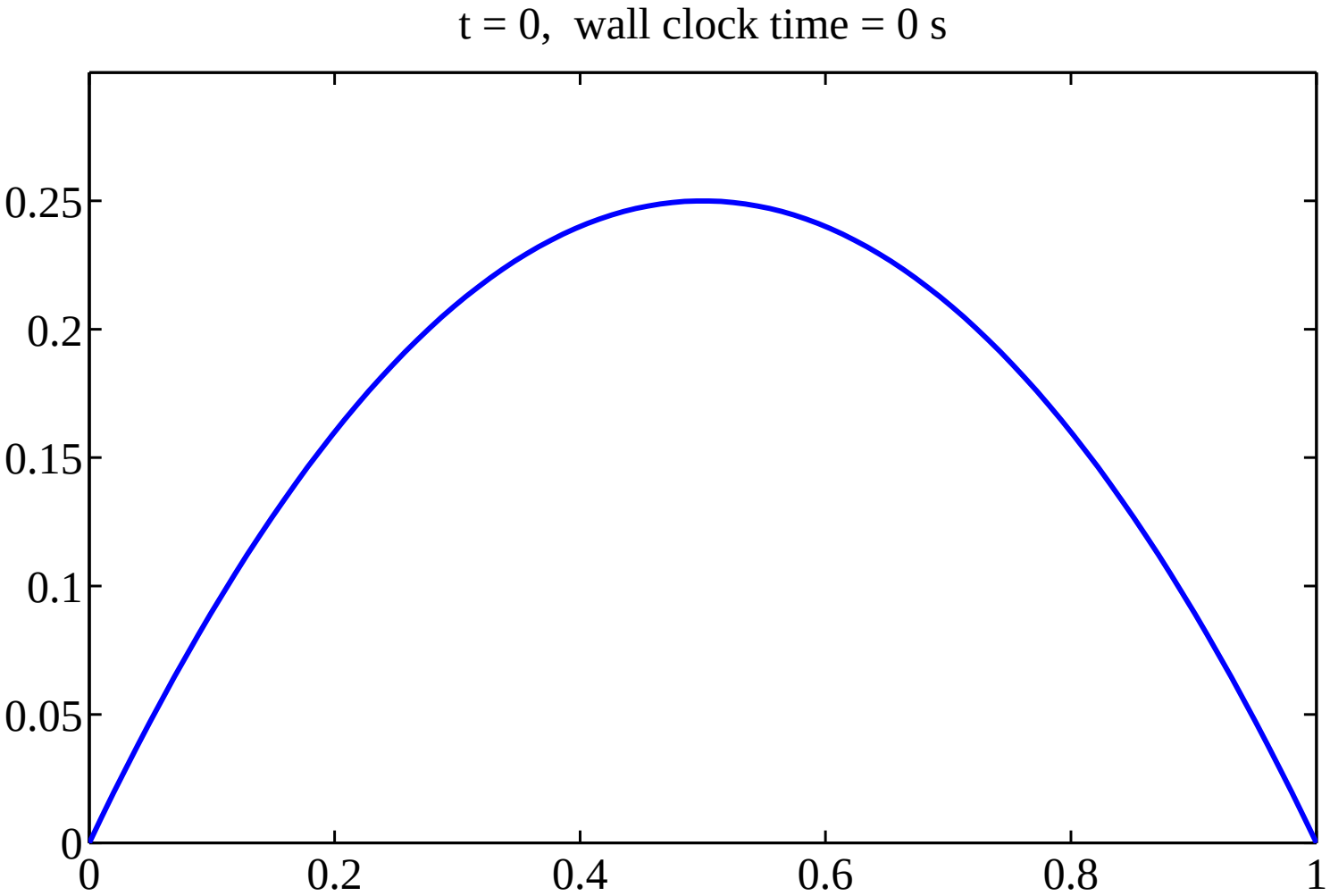
Finite-difference discretization in space yields

$$u'(t) = \mathbf{A}u(t) + q(t), \quad u(0) = u_0,$$

where $\mathbf{A} = (d+1)^2 \text{ tridiag}(1, -2, 1) \in \mathbb{R}^{d \times d}$, $d = 100$.

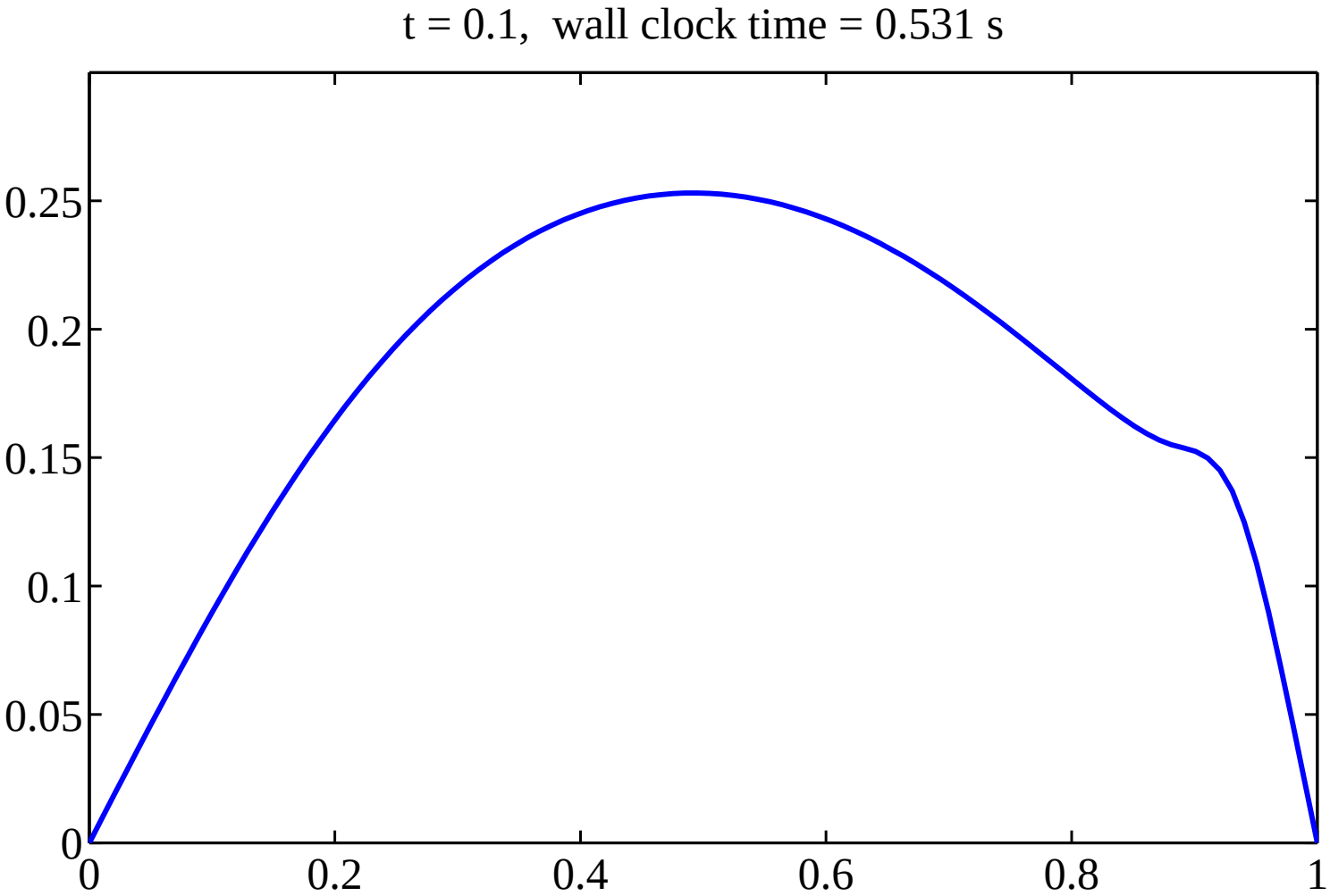
Serial integration

We integrate for $t \in [0, 1]$ using `ode15s` with error tolerance 10^{-3}.
Overall computation time: 5.2 s



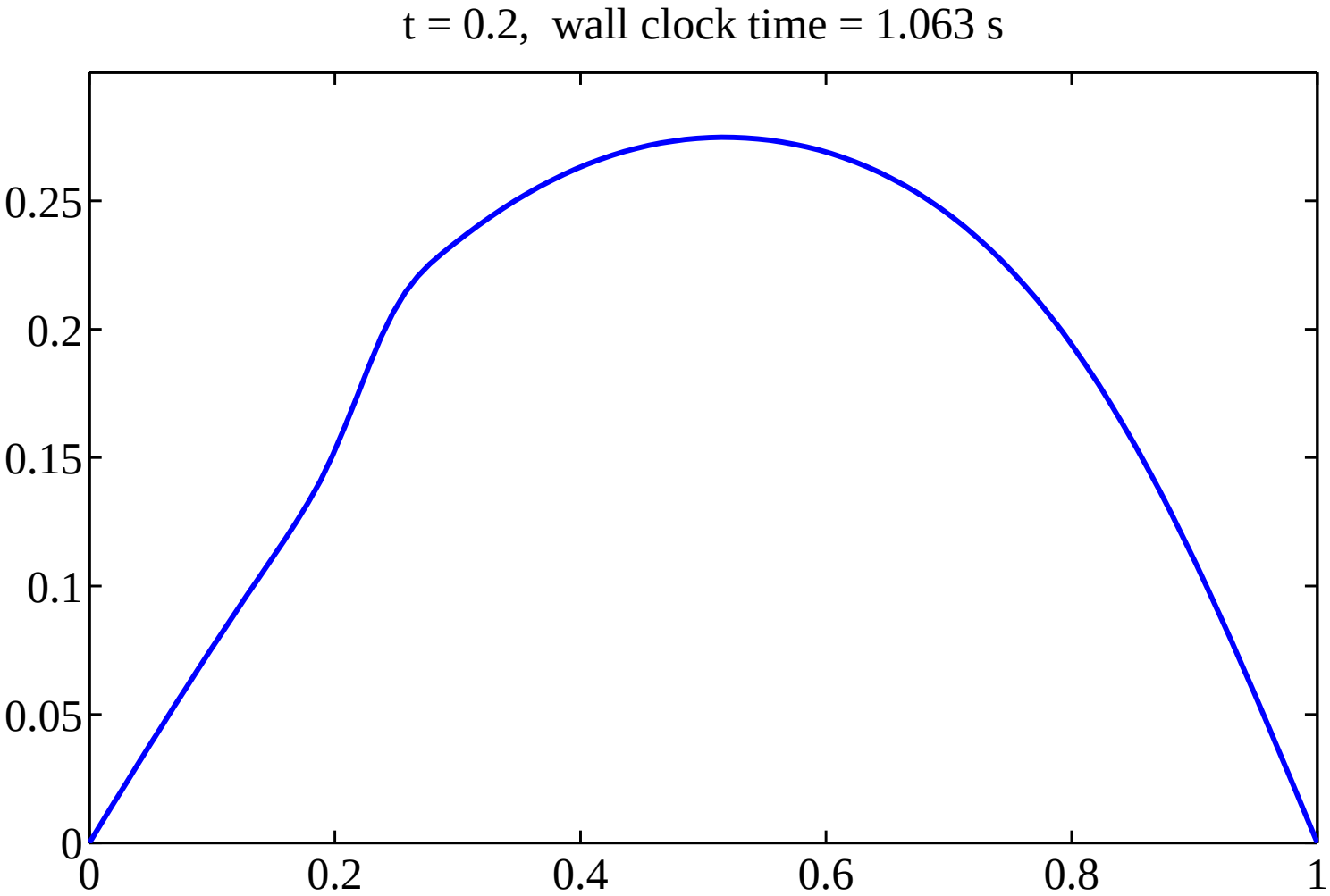
Serial integration

We integrate for $t \in [0, 1]$ using `ode15s` with error tolerance 10^{-3}.
Overall computation time: 5.2 s



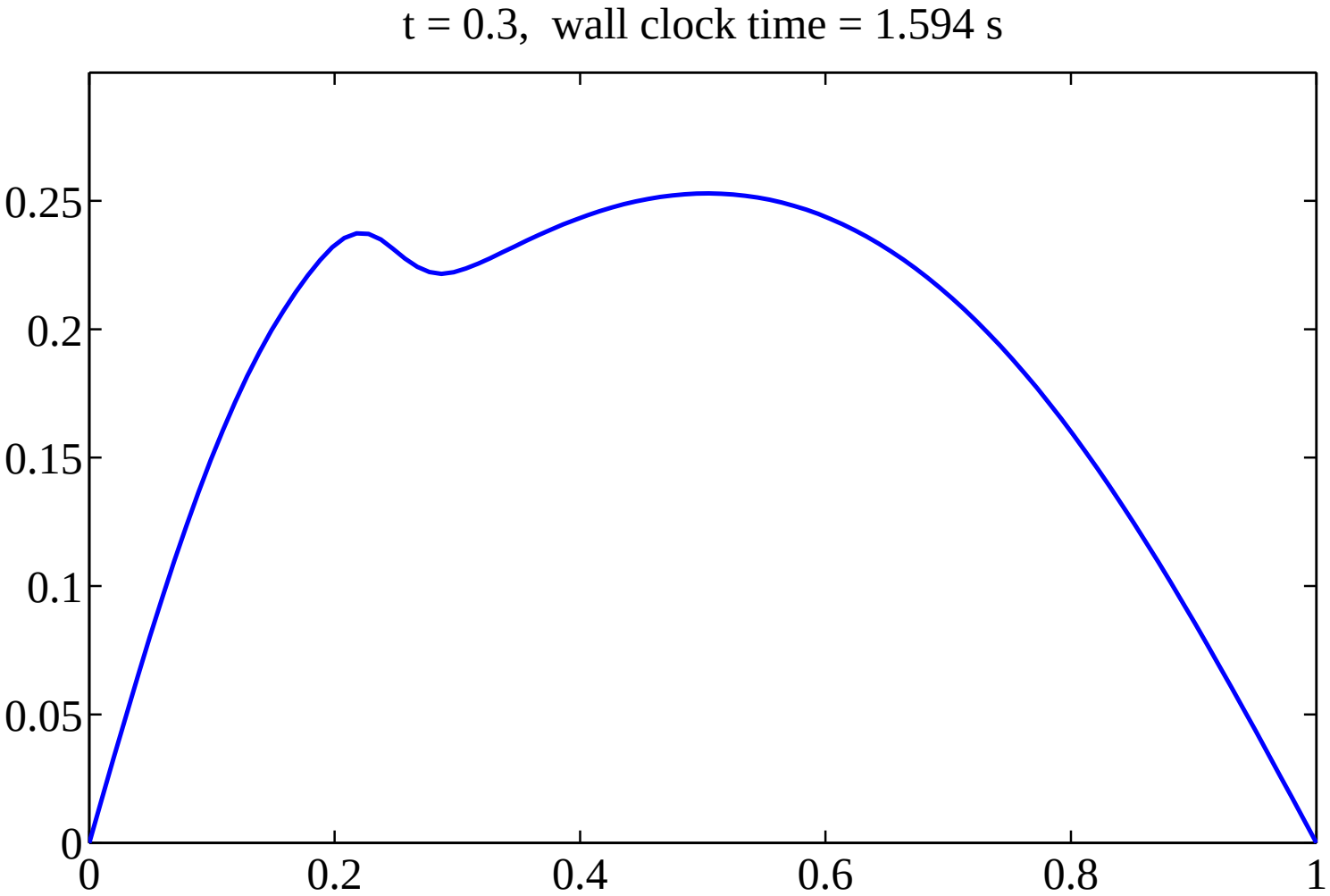
Serial integration

We integrate for $t \in [0, 1]$ using `ode15s` with error tolerance 10^{-3}.
Overall computation time: 5.2 s



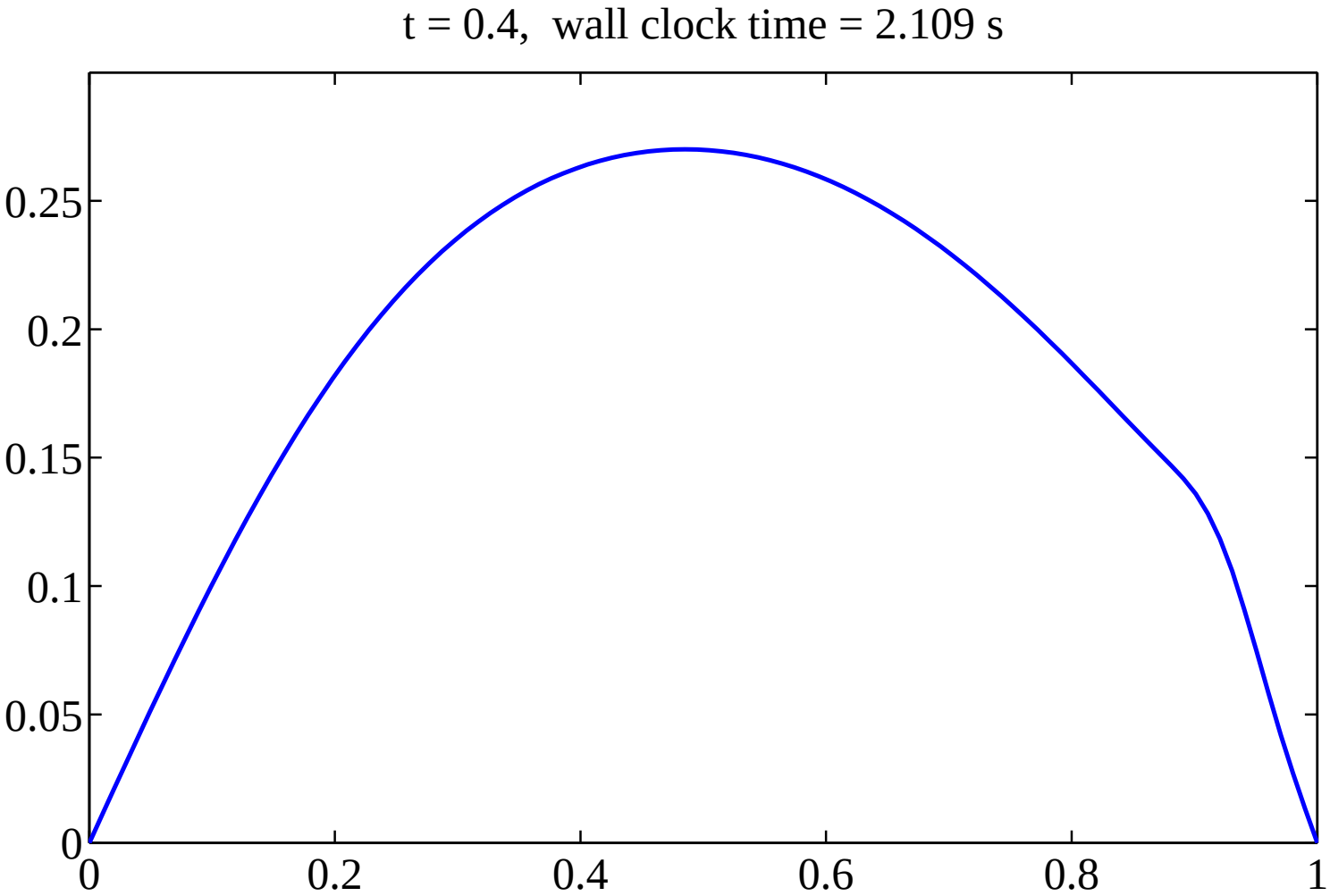
Serial integration

We integrate for $t \in [0, 1]$ using `ode15s` with error tolerance 10^{-3}.
Overall computation time: 5.2 s



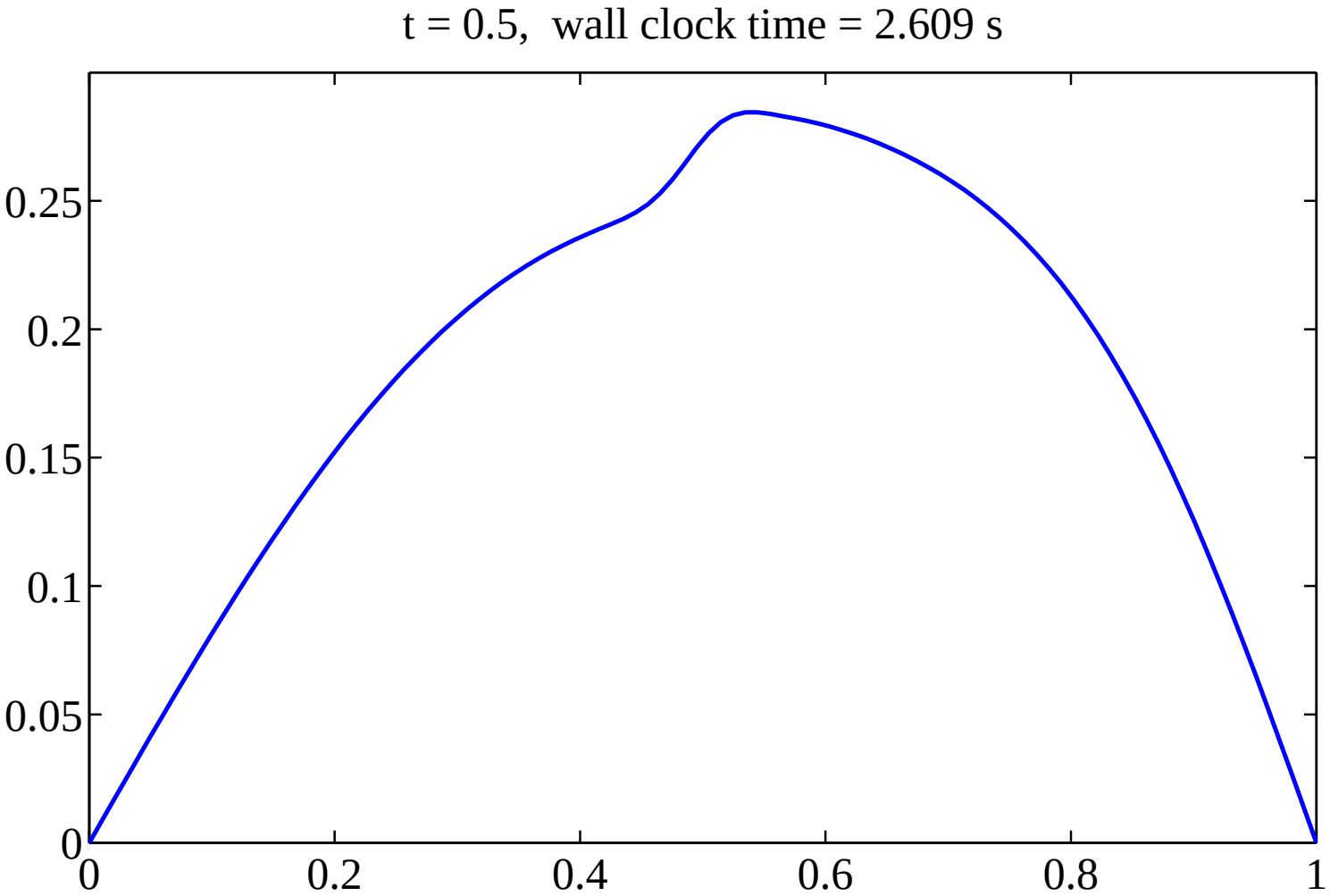
Serial integration

We integrate for $t \in [0, 1]$ using `ode15s` with error tolerance 10^{-3}.
Overall computation time: 5.2 s



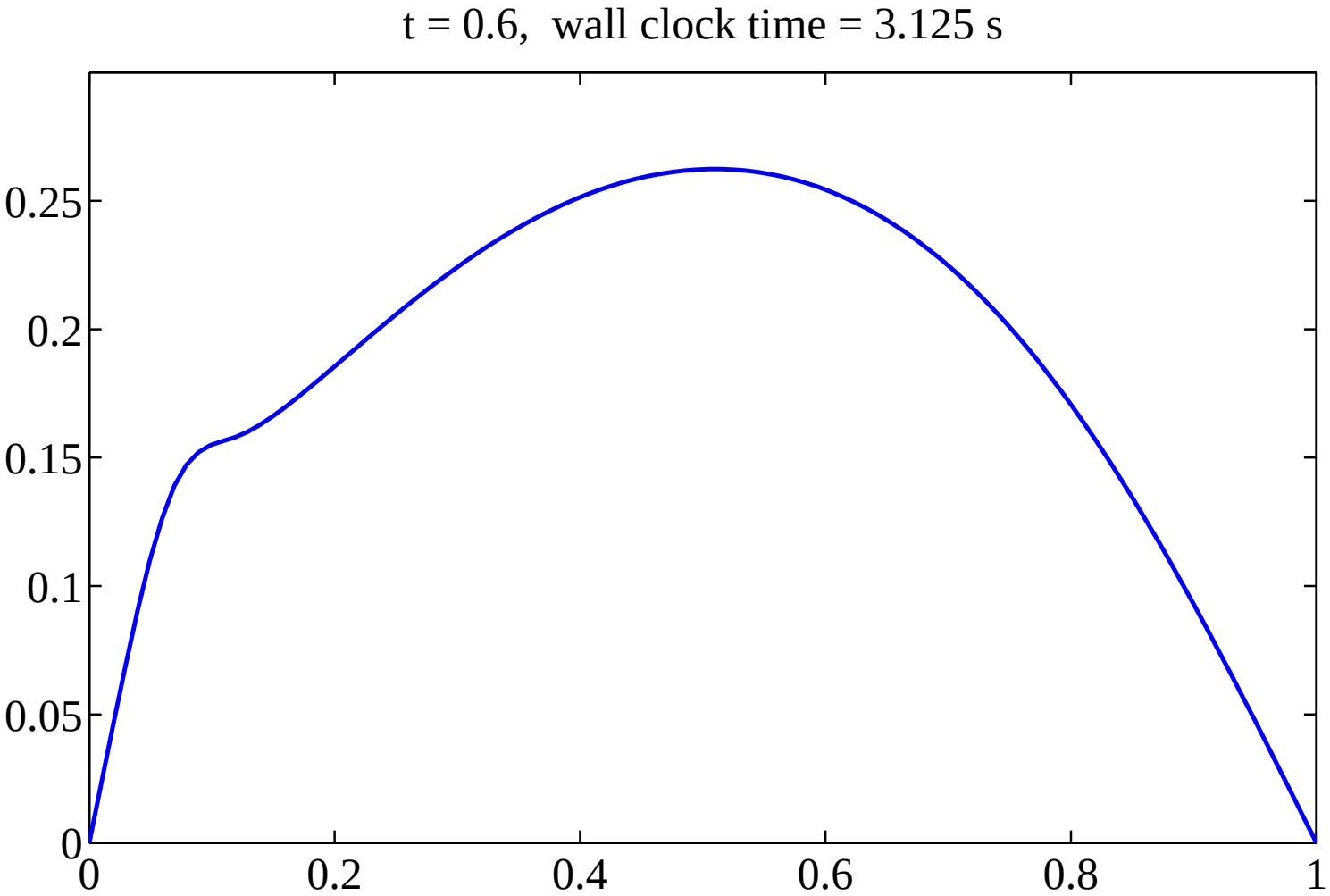
Serial integration

We integrate for $t \in [0, 1]$ using `ode15s` with error tolerance 10^{-3}.
Overall computation time: 5.2 s



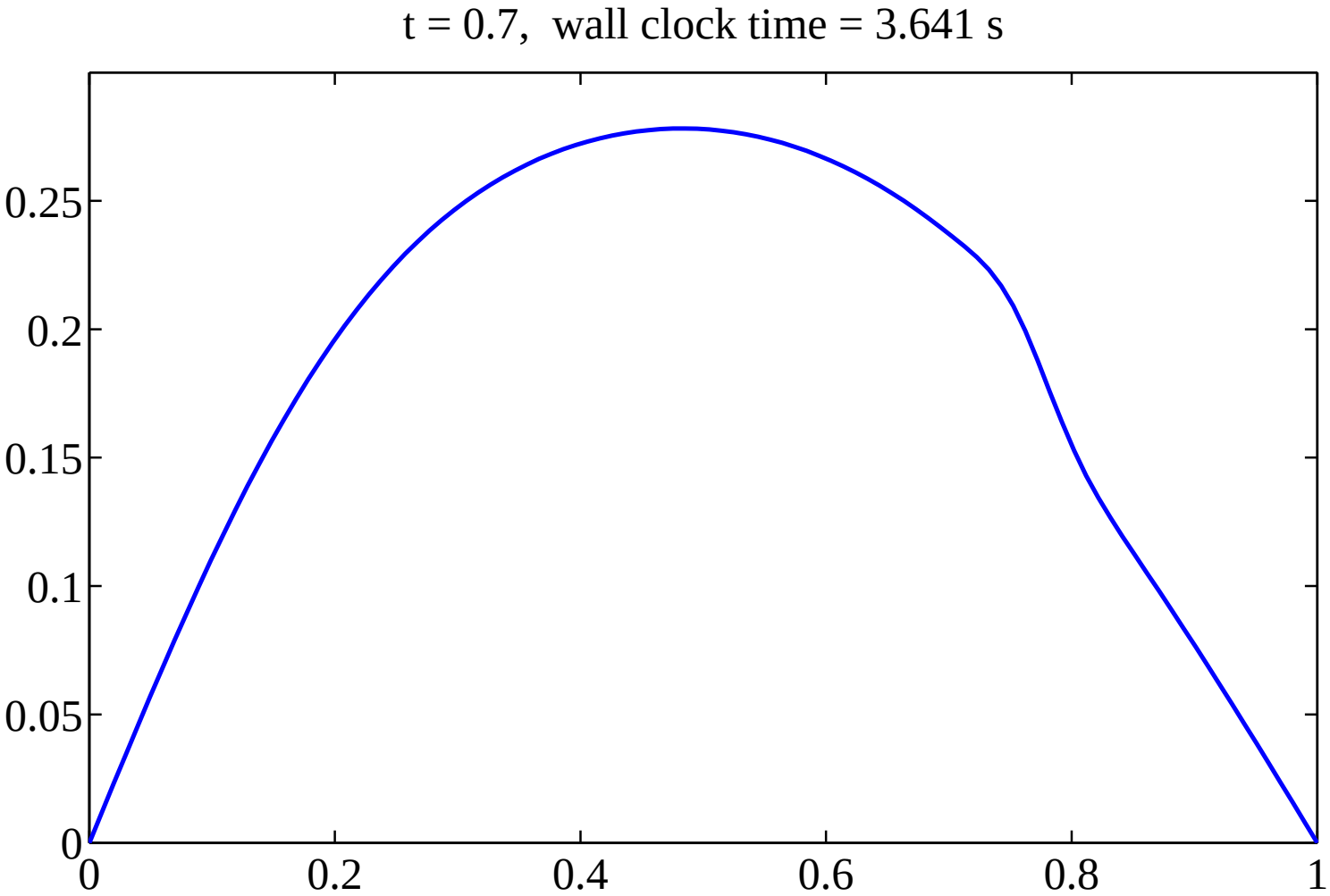
Serial integration

We integrate for $t \in [0, 1]$ using `ode15s` with error tolerance 10^{-3}.
Overall computation time: 5.2 s



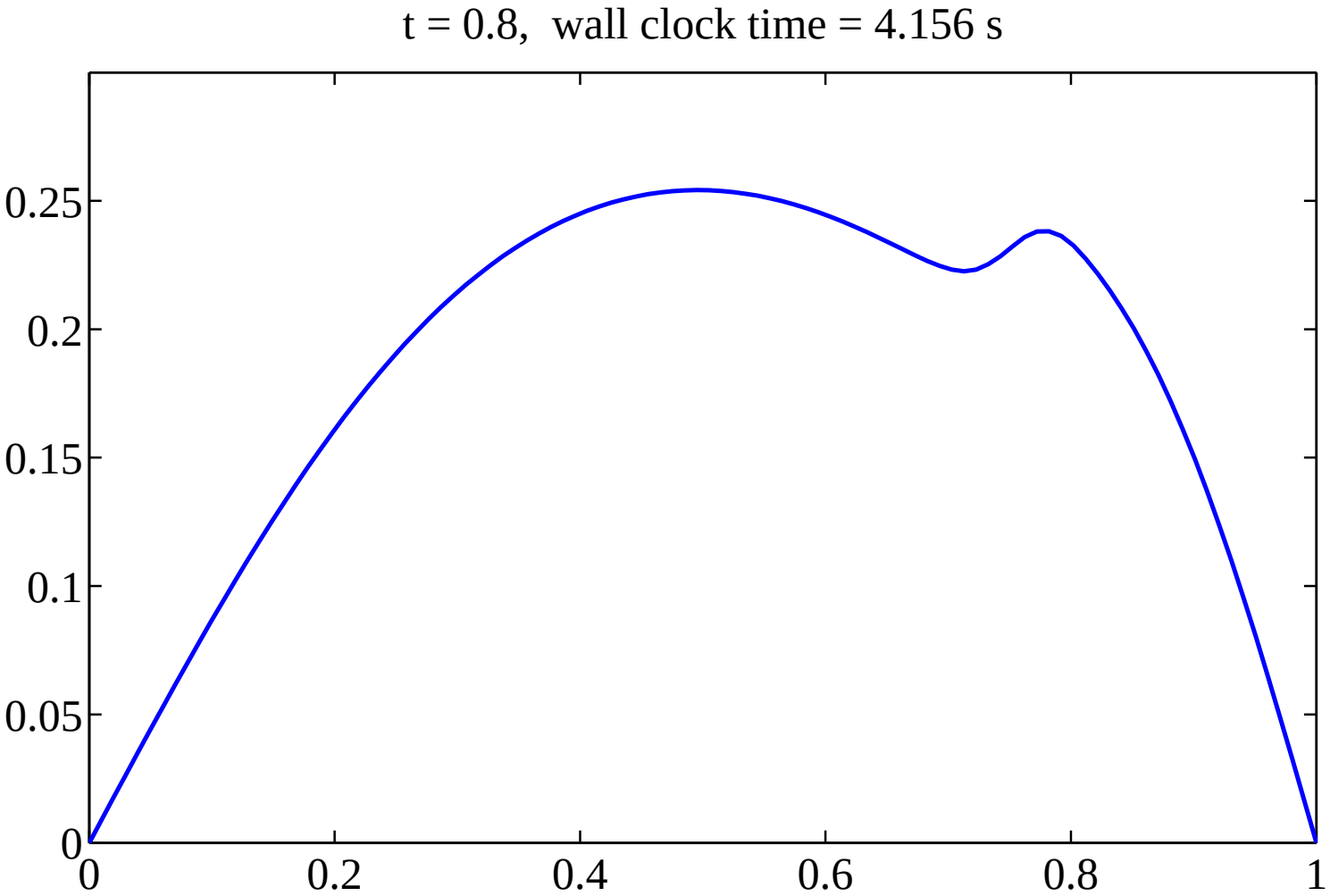
Serial integration

We integrate for $t \in [0, 1]$ using `ode15s` with error tolerance 10^{-3}.
Overall computation time: 5.2 s



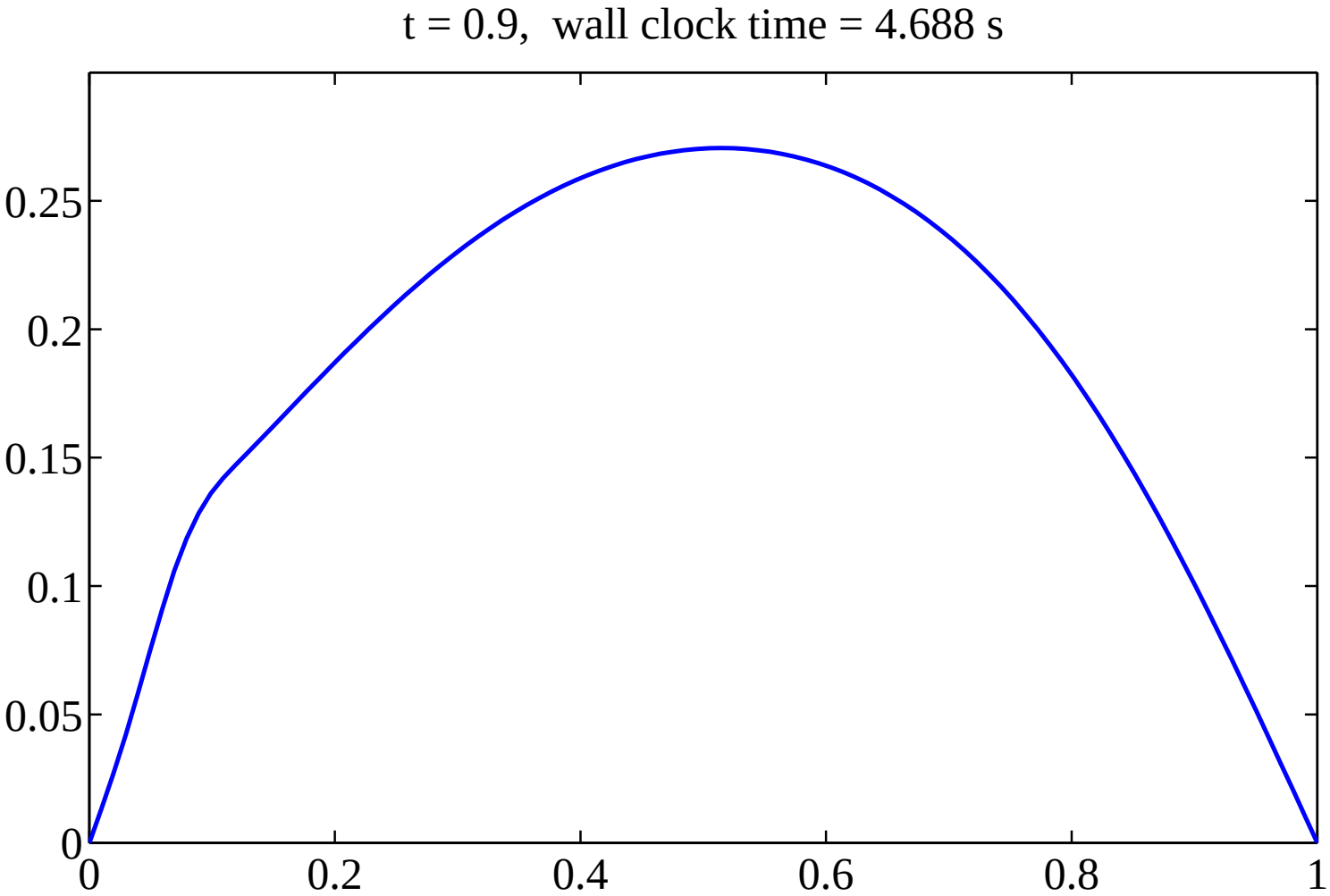
Serial integration

We integrate for $t \in [0, 1]$ using `ode15s` with error tolerance 10^{-3}.
Overall computation time: 5.2 s



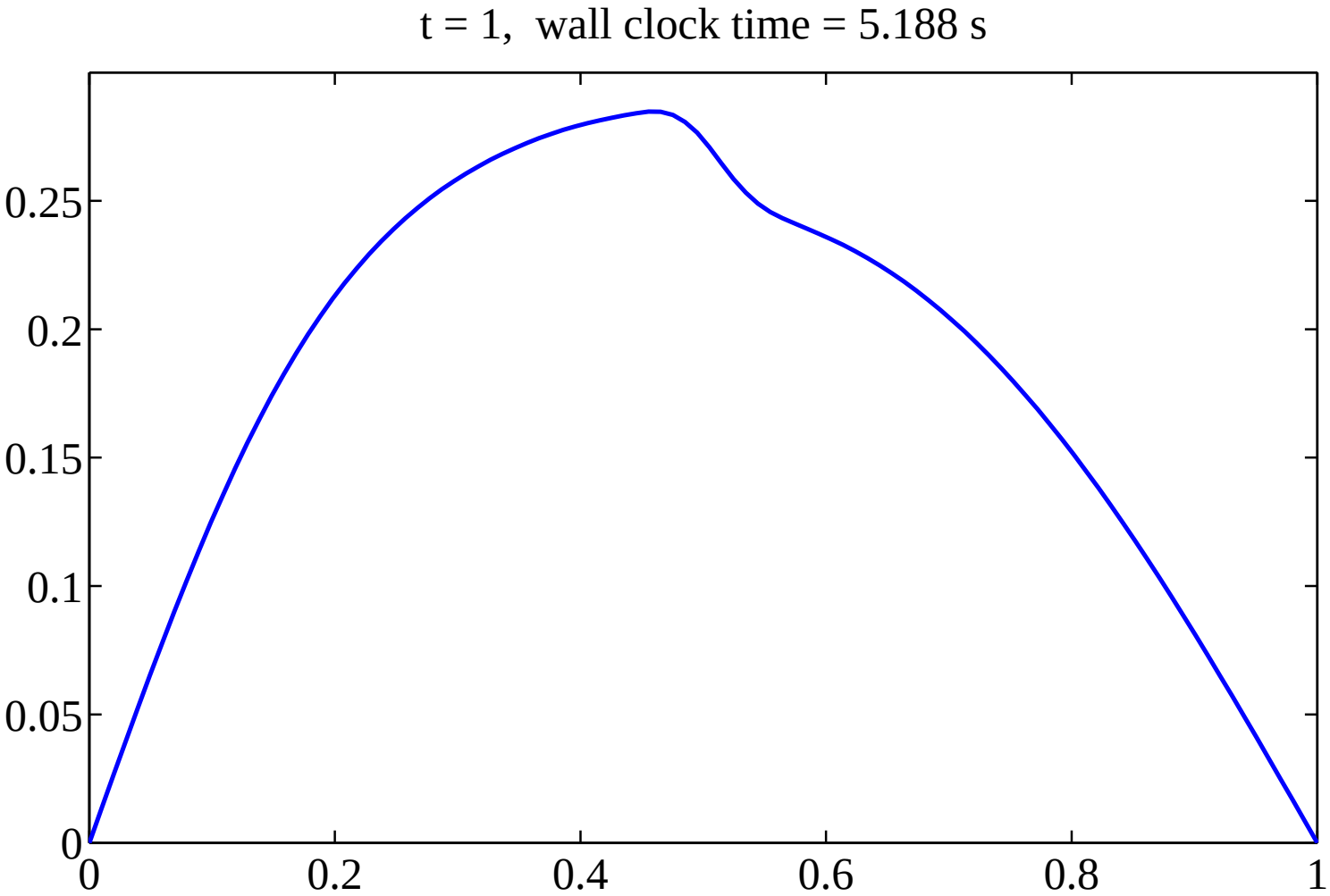
Serial integration

We integrate for $t \in [0, 1]$ using `ode15s` with error tolerance 10^{-3}.
Overall computation time: 5.2 s

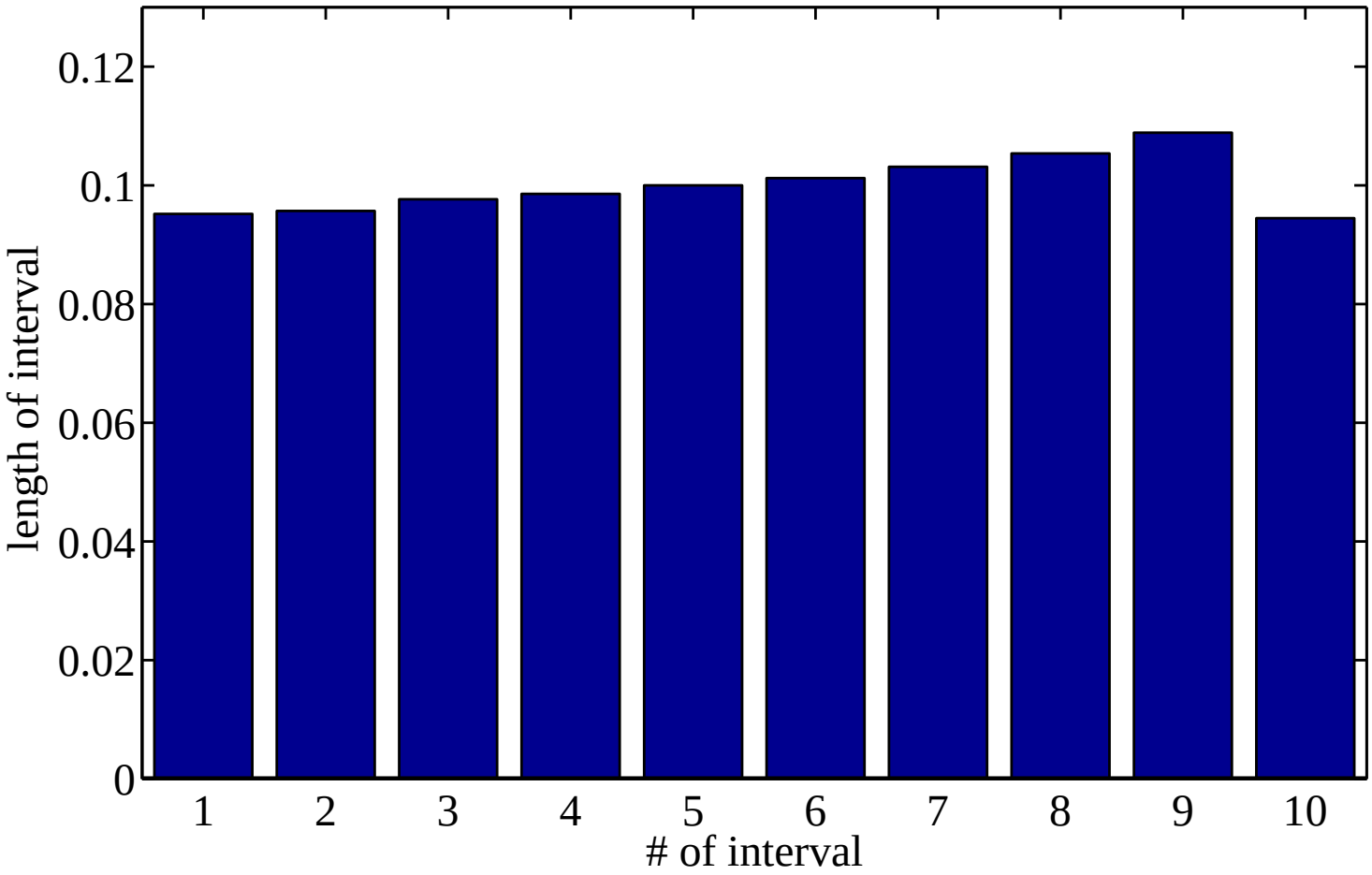


Serial integration

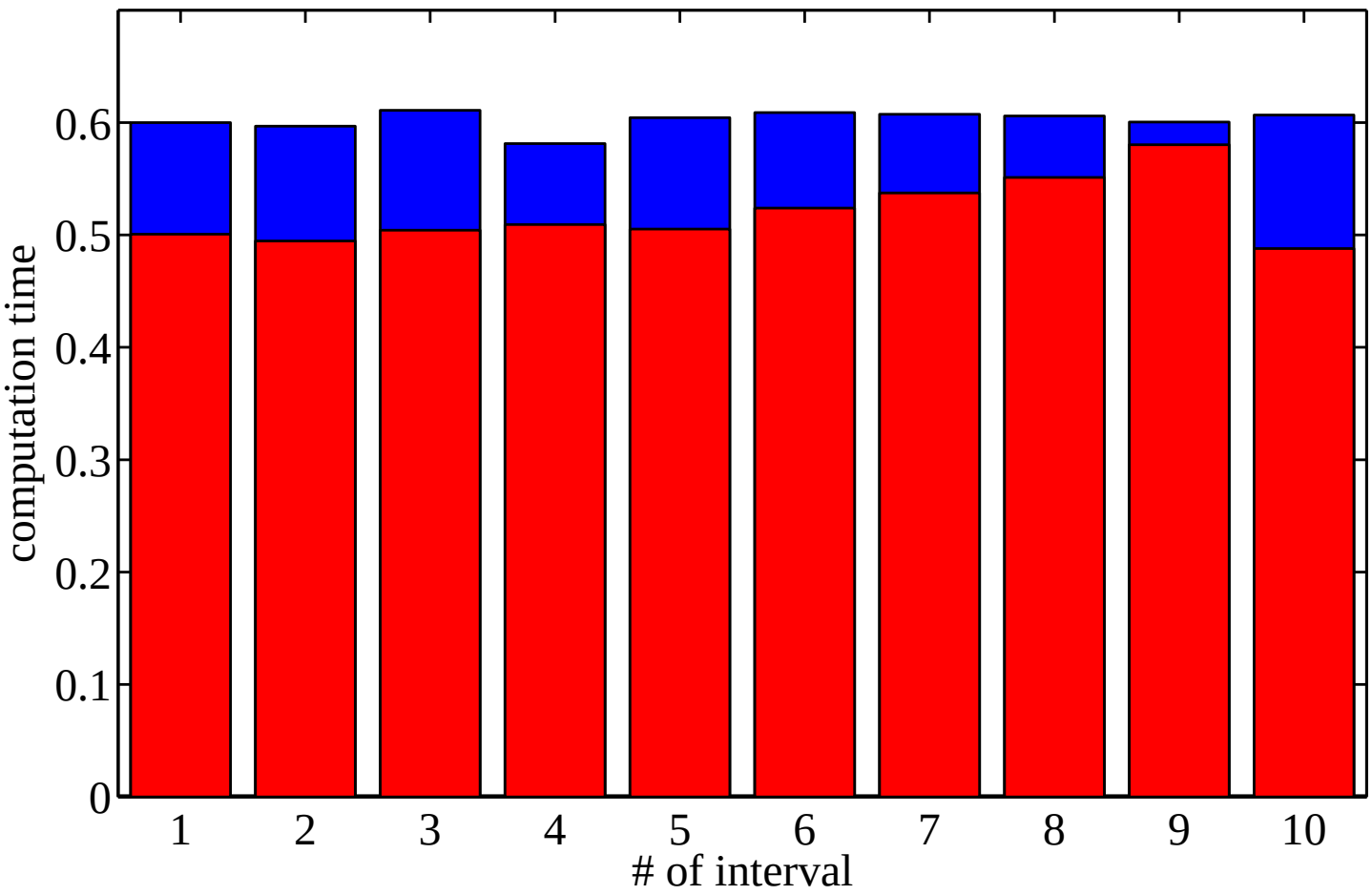
We integrate for $t \in [0, 1]$ using `ode15s` with error tolerance 10^{-3}.
Overall computation time: 5.2 s



Optimized length of 10 integration intervals



Parallel computation time of paraexp (ode15s / cheby)



Speed up = 5.2 / 0.6 ≈ 8.7.

Summary

- For linear problems almost perfectly scaling methods can be derived by appropriate decomposition into homogeneous and inhomogeneous subproblems.
- Efficient (in fact, near-optimal) polynomial and rational Krylov methods are available to integrate the homogeneous subproblem.
- The proposed *paraexp method* requires almost no communication and converges superlinearly.
- For linear problems it is questionable whether a parareal iteration is actually necessary.
- For semilinear problems $u'(t) = \mathbf{A}u(t) + g(t, u(t))$ similar ideas can be applied. However, in this case a parareal-like iteration is indeed necessary.