PARALLÉLISME EN TEMPS:
Que pouvons-nous prévoir du futur éloigné sans tout connaître du futur proche ?

Martin J. Gander
martin.gander@math.unige.ch

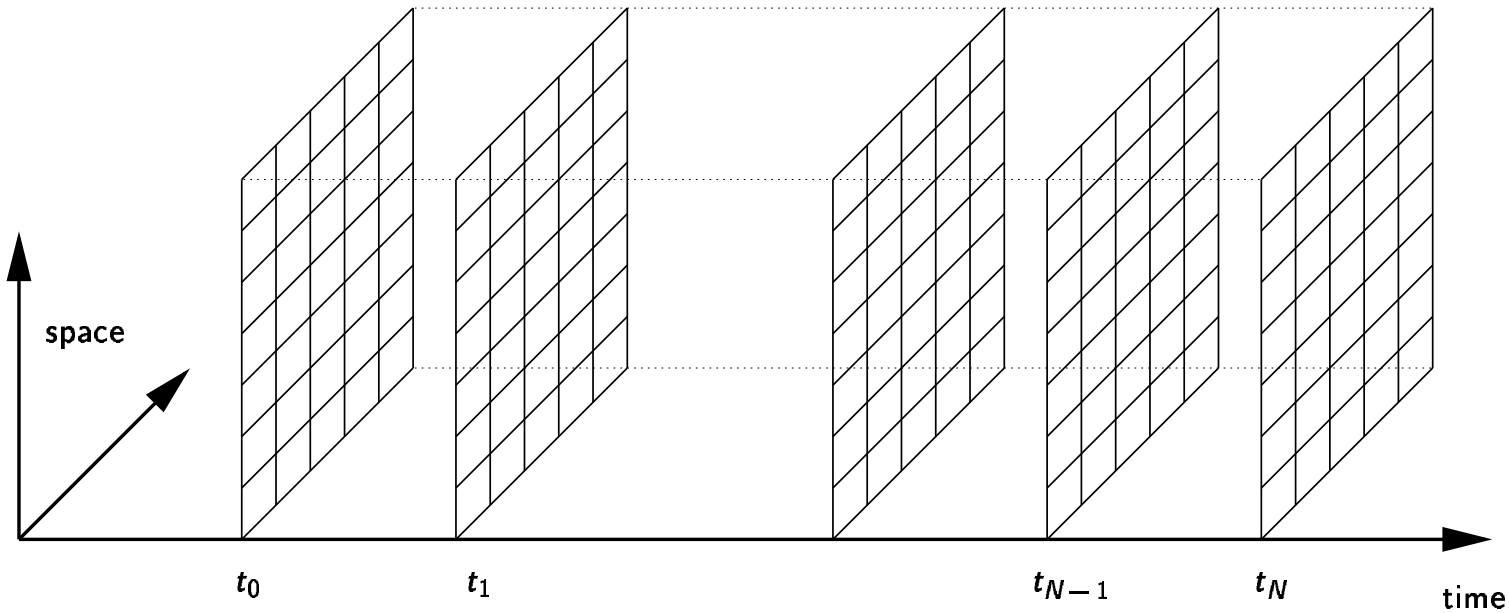
Université de Genève

Juillet 2007

Evolution Problems

Systems of ordinary differential equations $u' = f(u)$,

or partial differential equations $\dfrac{\partial u}{\partial t} = L(u) + f$.



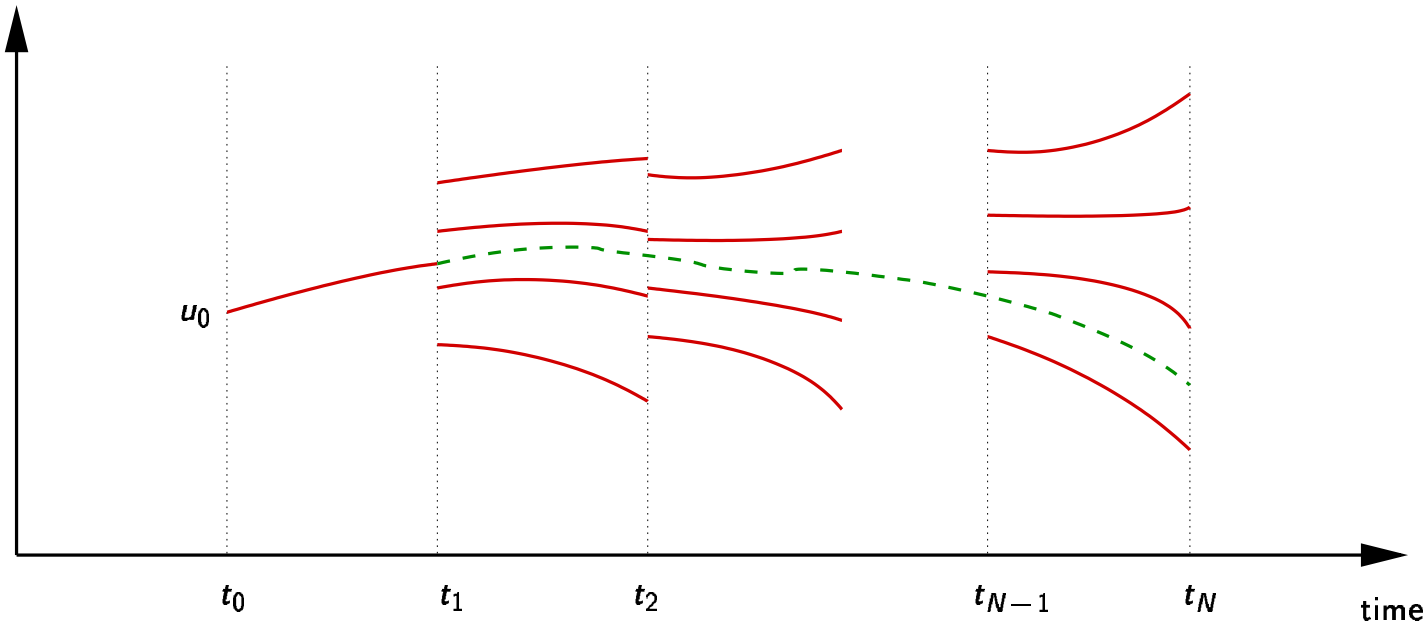
To what degree can the far future be predicted, before the near future is known accurately ?

History of Time Parallel Algorithms

J. Nievergelt, **Parallel Methods for Integrating Ordinary Differential Equations**. Comm. of the ACM, Vol 7(12), 1964.

"For the last 20 years, one has tried to speed up numerical computation mainly by providing ever faster computers. **Today, as it appears that one is getting closer to the maximal speed of electronic components,** *emphasis is put on allowing operations to be performed in parallel. In the near future, much of numerical analysis will have to be recast in a more "parallel" form."*

$$u' = f(u), \quad u(t_0) = u_0$$



Multiple shooting for boundary value problems

H. Keller, **Numerical Methods for Two-Point Boundary Value Problems** 1968.

For the model problem

$$u'' = f(u), \quad u(0) = u^0, \quad u(1) = u^1, \qquad x \in [0,1]$$

one splits the spatial interval into subintervals $[0, \frac{1}{3}]$, $[\frac{1}{3}, \frac{2}{3}]$, $[\frac{2}{3}, 1]$, and then solves on each subinterval

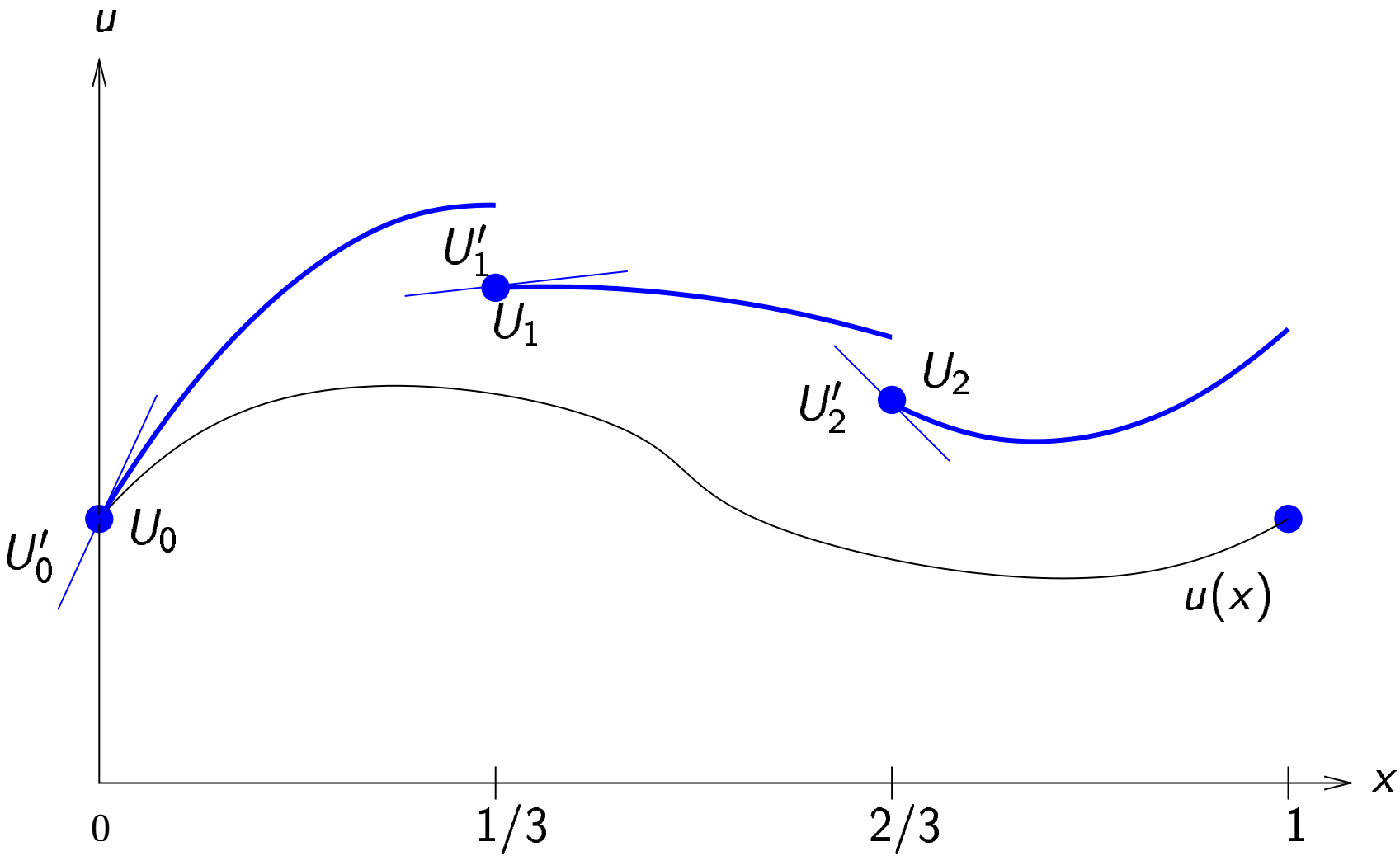
$$\begin{array}{rclrclrcl} u_0'' & = & f(u_0), & u_1'' & = & f(u_1), & u_2'' & = & f(u_2), \\ u_0(0) & = & U_0, & u_1(\frac{1}{3}) & = & U_1, & u_2(\frac{2}{3}) & = & U_2, \\ u_0'(0) & = & U_0', & u_1'(\frac{1}{3}) & = & U_1', & u_2'(\frac{2}{3}) & = & U_2', \end{array}$$

together with the matching conditions

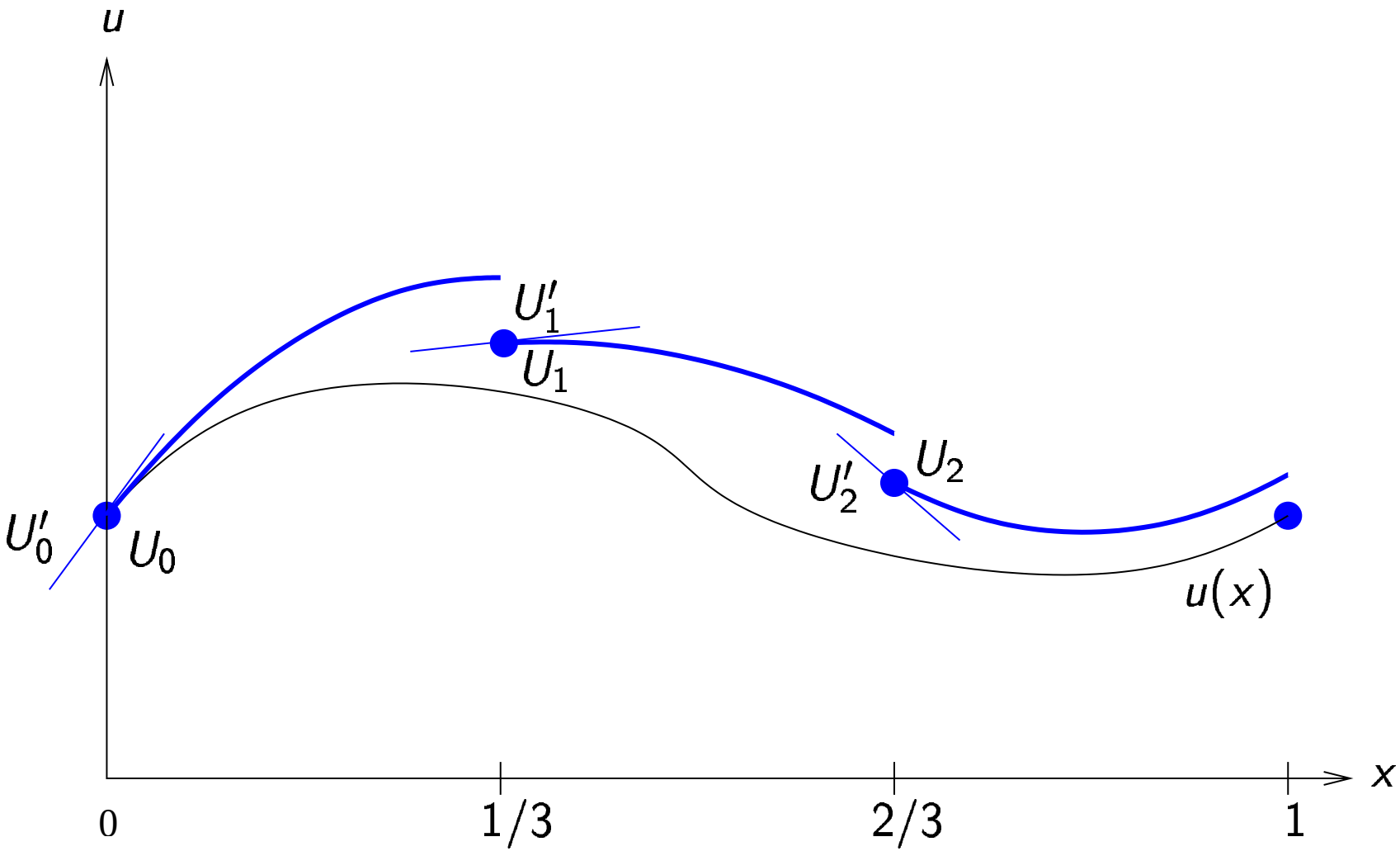
$$\begin{array}{lll} U_0 = u^0, & U_1 = u_0(\frac{1}{3}, U_0, U_0'), & U_2 = u_1(\frac{2}{3}, U_1, U_1'), \\ & U_1' = u_0'(\frac{1}{3}, U_0, U_0'), & U_2' = u_1'(\frac{2}{3}, U_1, U_1'), \quad u^1 = u_2(1, U_2, U_2') \end{array}$$

$$\Longleftrightarrow \quad F(\mathbf{U}) = 0, \quad \mathbf{U} = (U_0, U_1, U_2, U_0', U_1', U_2')^T.$$

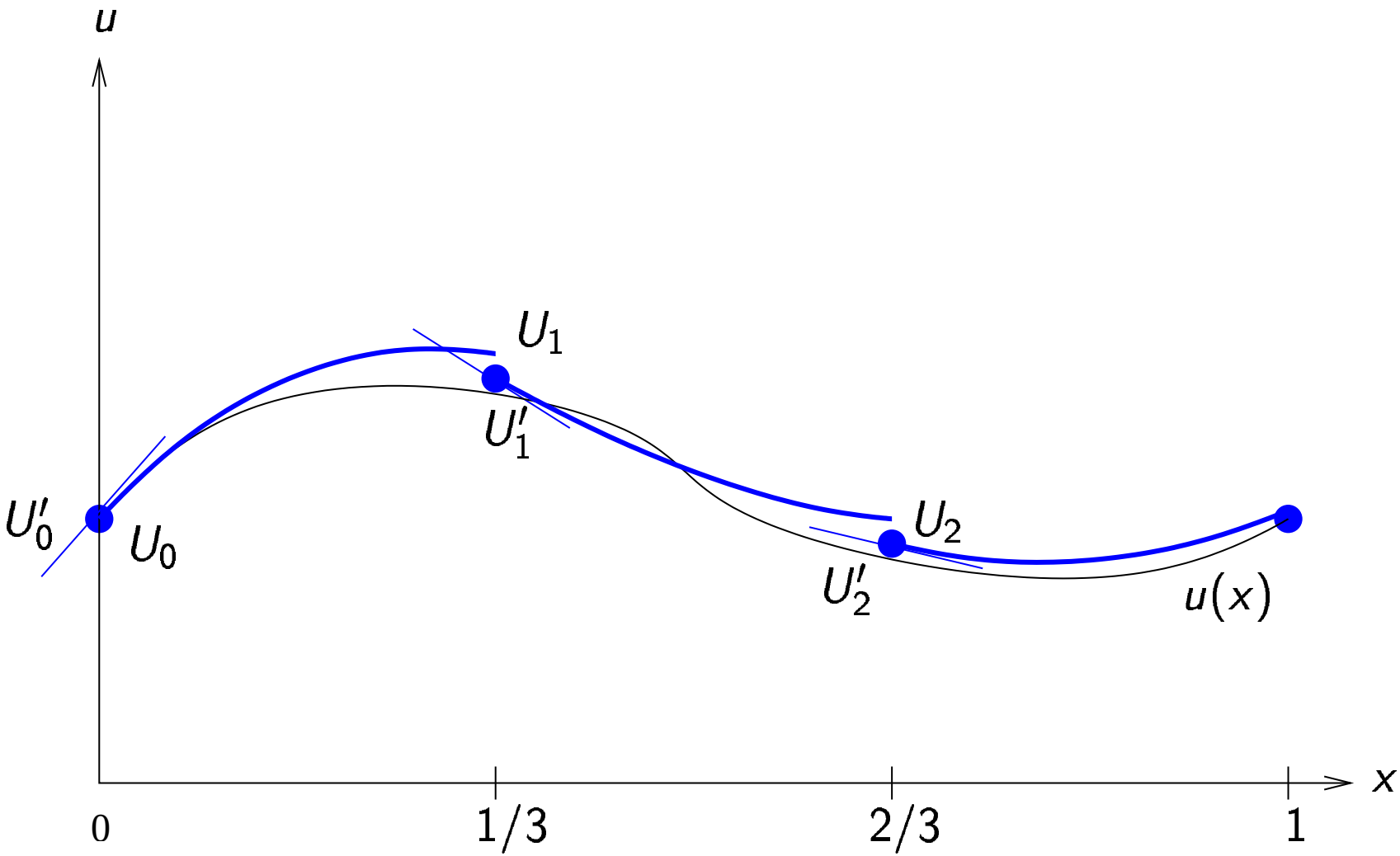
Example: first iteration



Example: second iteration



Example: third iteration



Multiple shooting for initial value problems

For the model problem

$$u' = f(u), \quad u(0) = u^0, \quad t \in [0,1]$$

one splits the time interval into subintervals $[0, \frac{1}{3}]$, $[\frac{1}{3}, \frac{2}{3}]$, $[\frac{2}{3}, 1]$, and then solves on each subinterval

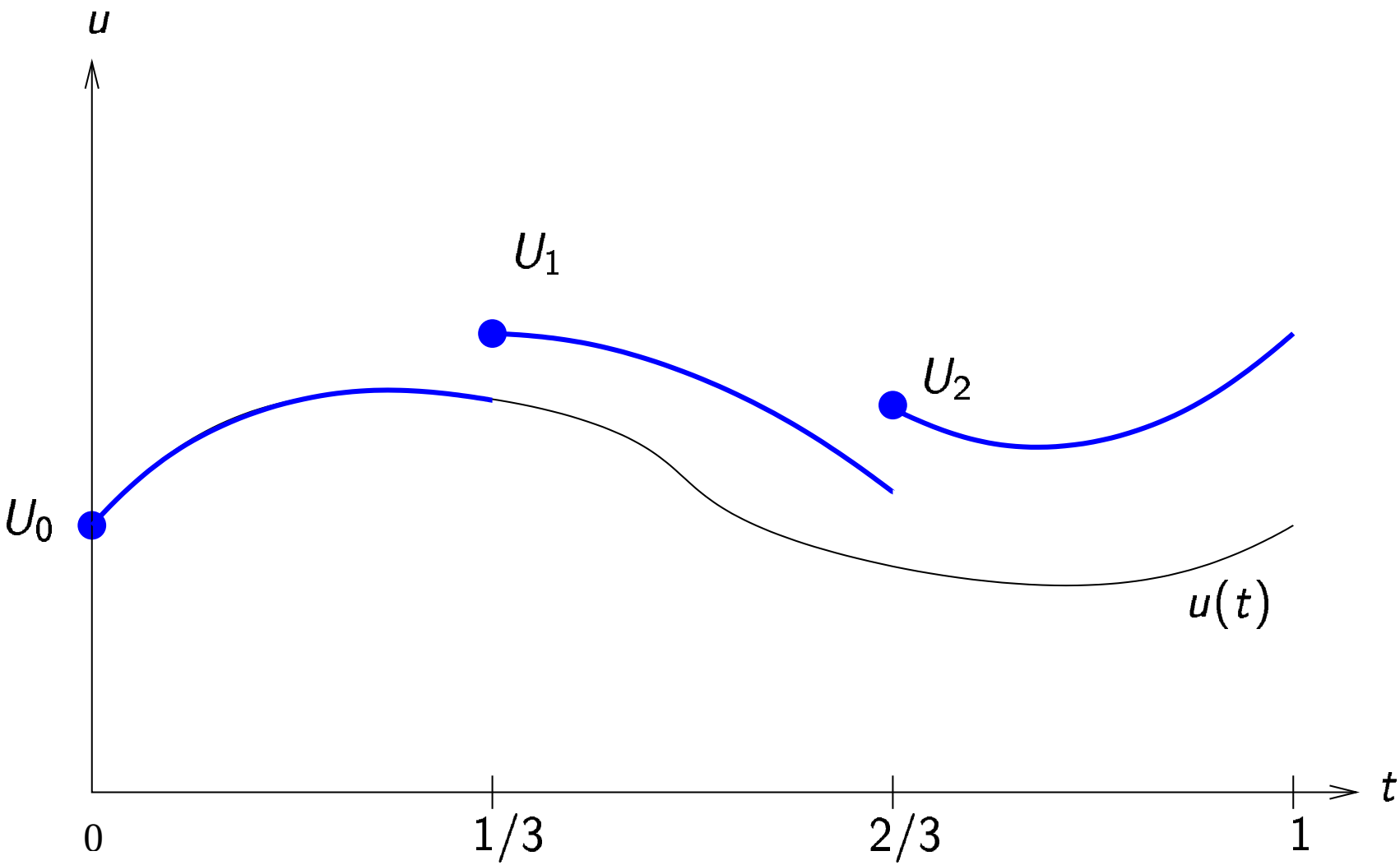
$$\begin{array}{rclrclrcl} u_0' & = & f(u_0), & u_1' & = & f(u_1), & u_2' & = & f(u_2), \\ u_0(0) & = & U_0, & u_1(\frac{1}{3}) & = & U_1, & u_2(\frac{2}{3}) & = & U_2, \end{array}$$

together with the matching conditions

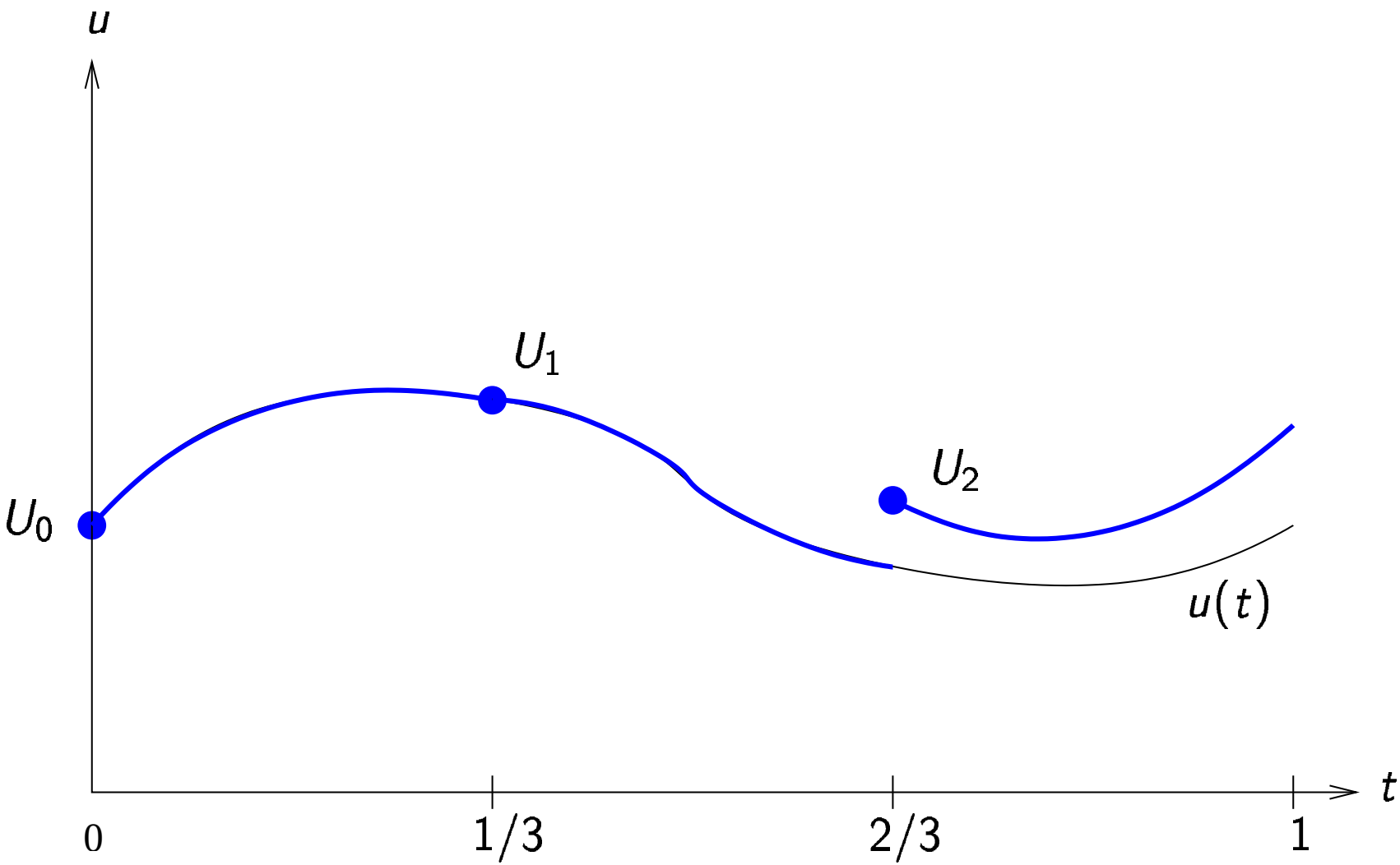
$$U_0 = u^0, \quad U_1 = u_0(\frac{1}{3}, U_0), \quad U_2 = u_1(\frac{2}{3}, U_1)$$

$$\Longleftrightarrow \quad F(\mathbf{U}) = 0, \quad \mathbf{U} = (U_0, U_1, U_2)^T.$$

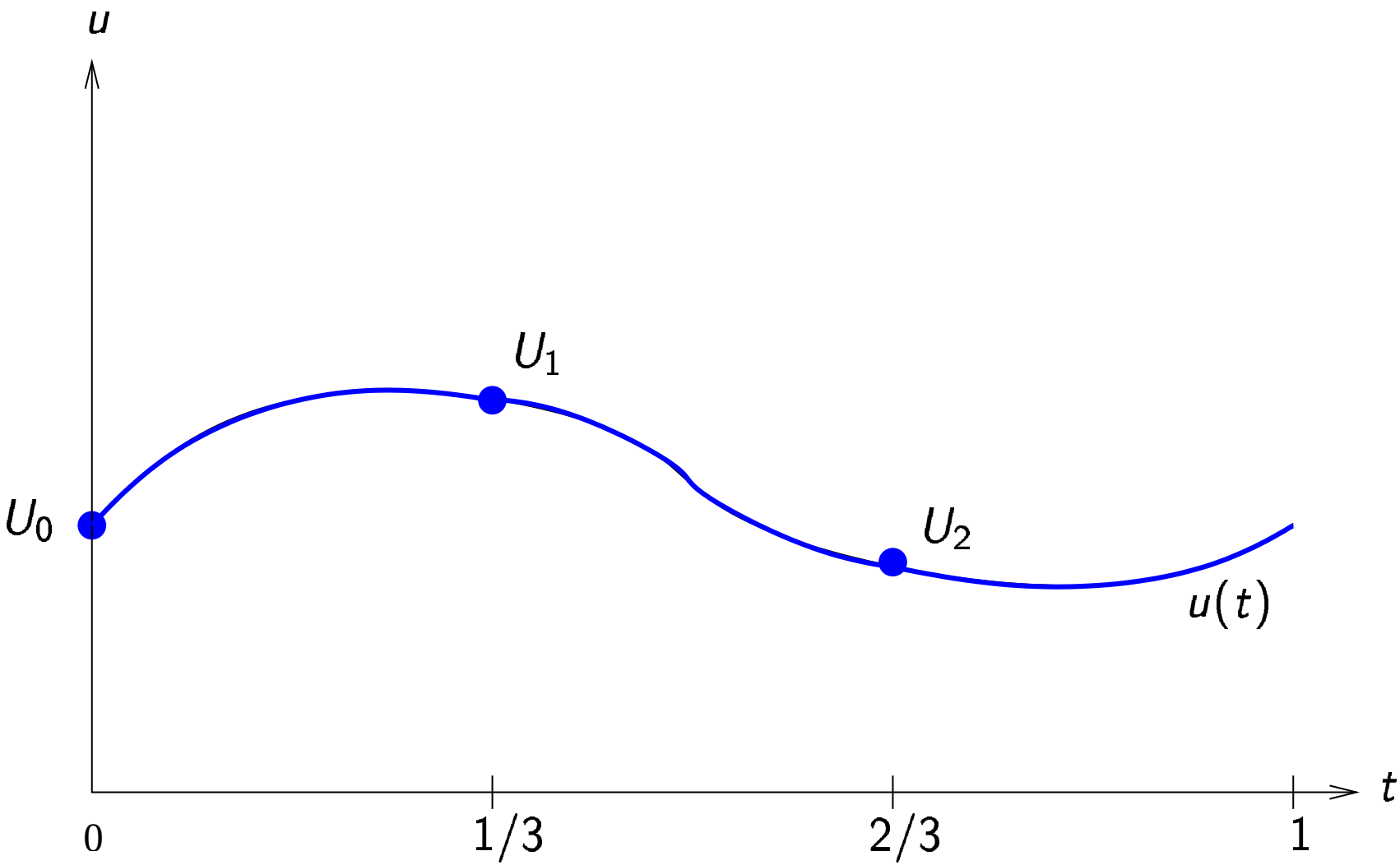
Example: first iteration



Example: second iteration



Example: third iteration



Using Newton's Method

Solving $F(\mathbf{U}) = 0$ with Newton leads to

$$\begin{pmatrix} U_0^{k+1} \\ U_1^{k+1} \\ U_2^{k+1} \end{pmatrix} = \begin{pmatrix} U_0^{k} \\ U_1^{k} \\ U_2^{k} \end{pmatrix} - \begin{bmatrix} 1 & & \\ -\frac{\partial u_0}{\partial U_0}(\frac{1}{3}, U_0^k) & 1 & \\ & -\frac{\partial u_1}{\partial U_1}(\frac{2}{3}, U_1^k) & 1 \end{bmatrix}^{-1} \begin{pmatrix} U_0^k - u^0 \\ U_1^k - u_1(\frac{1}{3}, U_0^k) \\ U_2^k - u_1(\frac{2}{3}, U_1^k) \end{pmatrix}$$

Multiplying through by the matrix, we find the recurrence

$$\begin{aligned} U_0^{k+1} &= u^0, \\ U_1^{k+1} &= u_0(\tfrac{1}{3}, U_0^k) + \tfrac{\partial u_0}{\partial U_0}(\tfrac{1}{3}, U_0^k)(U_0^{k+1} - U_0^k), \\ U_2^{k+1} &= u_1(\tfrac{2}{3}, U_1^k) + \tfrac{\partial u_1}{\partial U_1}(\tfrac{2}{3}, U_1^k)(U_1^{k+1} - U_1^k). \end{aligned}$$

General case with N intervals, $t_n = n\Delta T$, $\Delta T = 1/N$

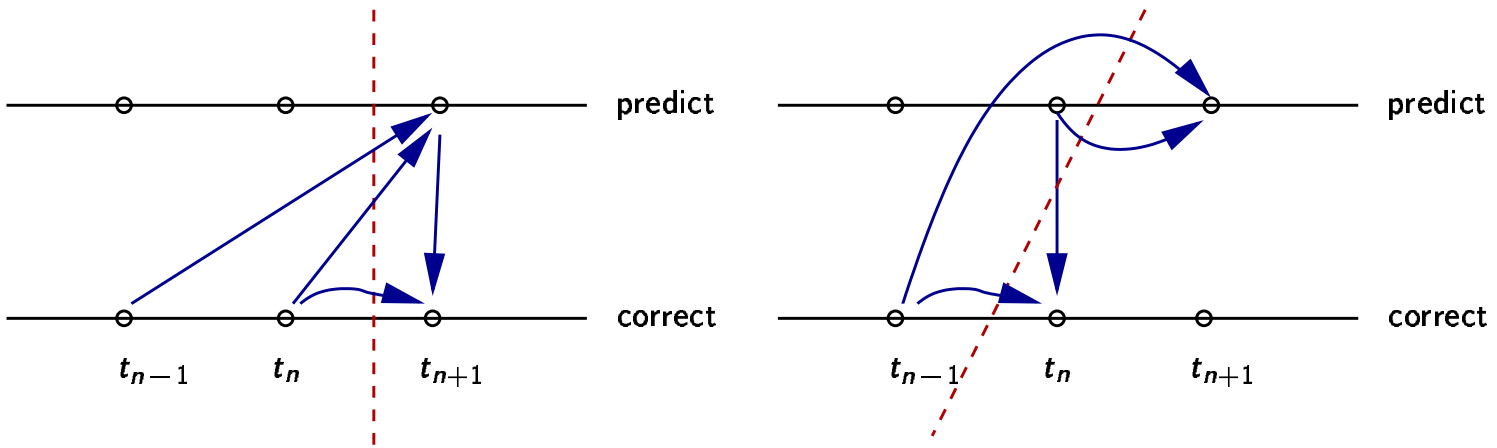
$$U_{n+1}^{k+1} = u_n(t_{n+1}, U_n^k) + \frac{\partial u_n}{\partial U_n}(t_{n+1}, U_n^k)(U_n^{k+1} - U_n^k).$$

Parallel time stepping

W. Miranker and W. Liniger, **Parallel Methods for the Numerical Integration of Ordinary Differential Equations.** Math. Comp., Vol 21, 1967.

"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."*

$$u' = f(u), \quad u(0) = u_0$$



Space-Time Iterative Methods

- **Waveform Relaxation**

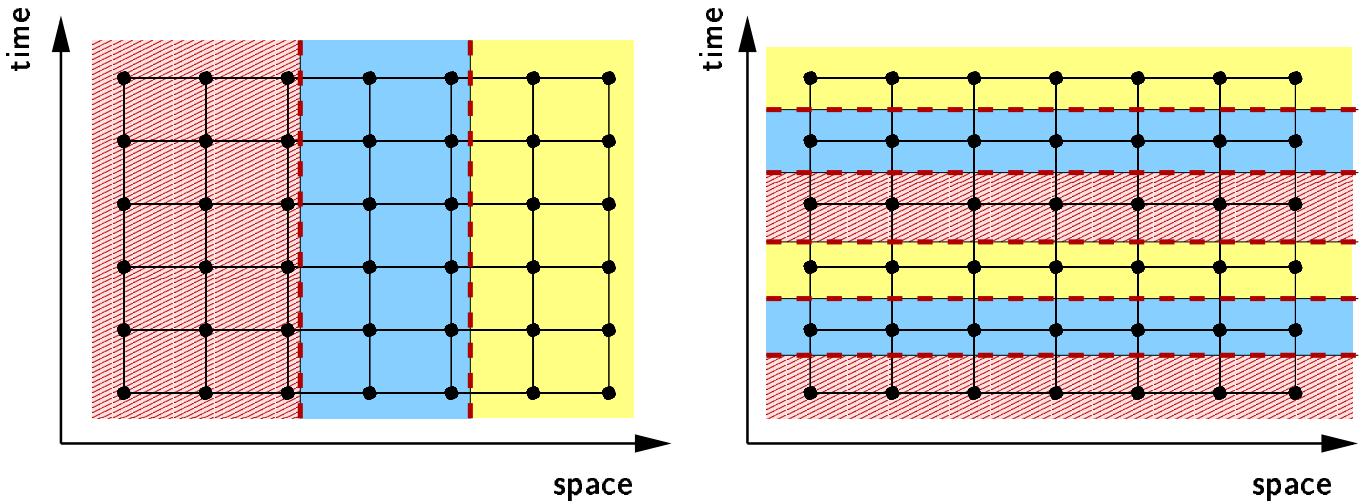
  E. Lelarasmee, A.E. Ruehli, A.L. Sangiovanni-Vincentelli. The Waveform Relaxation Method for Time-Domain Analysis of Large Scale Integrated Circuits, IEEE Trans. on CAD of IC and Syst. 1982.

- **Windowed Relaxation**

  J. Saltz, V. Naik, D. Nicol. Reduction of the effects of communication delays in scientific algorithms on message passing MIMD architectures. SISC 8(1), 1987.

- **Parallel Time Stepping**

  D. Womble. A time-stepping algorithm for parallel computers. SISC 11(5), 1990.



A Negative Result for Parallel Time Stepping

Deshpande, Malhotra, Douglas, Schultz, **Temporal Domain Parallelism: Does it Work ? Tech report 1993, SISC 1995**

Results:

- if a good solver is used on each time step, no parallel speedup is possible.
- if a very slow solver is used on each time step, a small parallel speedup can be achieved.

Quote from the tech report:

"We show that this approach is not normally useful".

Parabolic Multigrid Methods

Applying **a multigrid algorithm** on the **space-time mesh**

- rapid information propagation forward in time, by means of a (set of) coarse meshes
- very few iterations before convergence.

Three **examples:**

- **Time-parallel multigrid** (Hackbusch, 1984; Bastian, Burmeiser, Horton, 1990; Oosterlee, 1992;...)
- **Multigrid waveform relaxation** (Lubich and Ostermann, 1987; Vandevalle and Piessens,1988;...)
- **Space-time multigrid** (Horton and Vandevalle, 1995)

A Time-Multigrid Algorithm

Linear model problem:

$$u' = au, \quad u(0) = u_0.$$

Discretization with Forward Euler:

Fine grid problem:

$$u_{m+1} = u_m + a\Delta t u_m =: \alpha_{\Delta t} u_m,$$

Coarse grid problem:

$$U_{n+1} = U_n + a\Delta T U_n =: \alpha_{\Delta T} U_n.$$

Using Matrix Notation

Fine grid: $M\mathbf{u} = \mathbf{b}$, coarse grid: $M_{\Delta T}\mathbf{U} = \mathbf{b}_{\Delta T}$, where

$$M := \begin{bmatrix} 1 & & \\ -\alpha_{\Delta t} & 1 & \\ & \ddots & \ddots \end{bmatrix}, \quad \mathbf{b} = \begin{pmatrix} u_0 \\ 0 \\ \vdots \end{pmatrix},$$

$$M_{\Delta T} := \begin{bmatrix} 1 & & \\ -\alpha_{\Delta T} & 1 & \\ & \ddots & \ddots \end{bmatrix}, \ \mathbf{b}_{\Delta T} = \begin{pmatrix} u_0 \\ 0 \\ \vdots \end{pmatrix}.$$

Time multigrid algorithm: starting with an initial guess $\mathbf{u}^0$

$$\tilde{\mathbf{u}}^k = \mathbf{u}^k + S(\mathbf{b} - M\mathbf{u}^k), \quad M_{\Delta T}\mathbf{e}_{\Delta T}^{k+1} = I_{\Delta t}^{\Delta T}(\mathbf{b} - M\tilde{\mathbf{u}}^k), \quad \mathbf{u}^{k+1} = \tilde{\mathbf{u}}^k + I_{\Delta T}^{\Delta t}\mathbf{e}_{\Delta T}^{k+1}.$$

Nonlinear problems: Multigrid Full Approximation Scheme

The Parareal Algorithm

J-L. Lions, Y. Maday, G. Turinici, **A "Parareal" in Time Discretization of PDEs, C.R.Acad.Sci. Paris, t.322, 2001.**

The parareal algorithm for the model problem

$$u' = f(u)$$

is defined using two propagation operators:

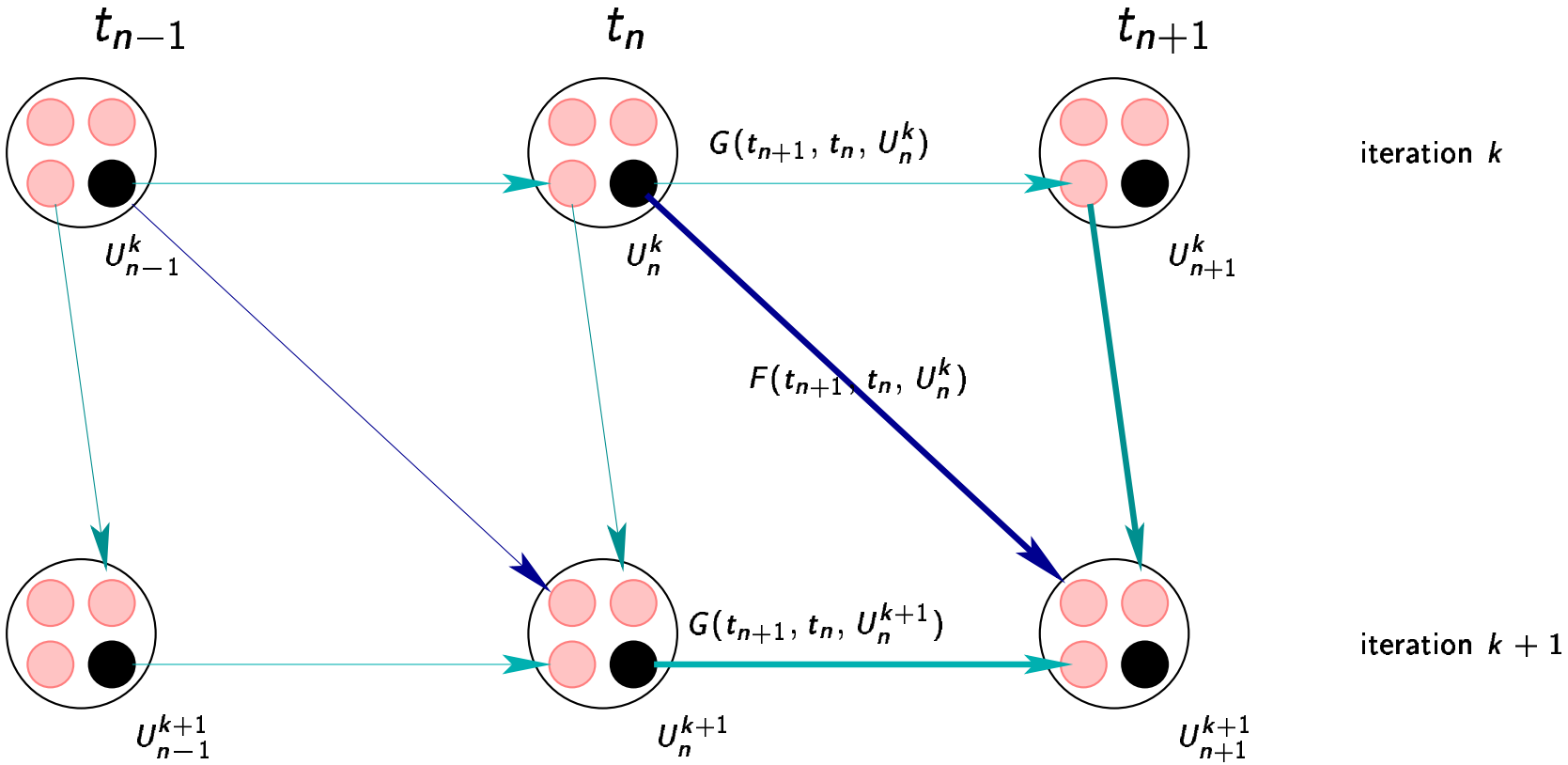
1. $G(t_1, t_2, u_1)$ is a rough approximation to $u(t_2)$ with initial condition $u(t_1) = u_1$,
2. $F(t_1, t_2, u_1)$ is a more accurate approximation of the solution $u(t_2)$ with initial condition $u(t_1) = u_1$.

Starting with a coarse approximation U_n^0 at the time points $t_1, t_2, \ldots, t_N$, parareal performs for $k = 0, 1, \ldots$ the correction iteration

$$U_{n+1}^{k+1} = G(t_{n+1}, t_n, U_n^{k+1}) + F(t_{n+1}, t_n, U_n^k) - G(t_{n+1}, t_n, U_n^k).$$

How does it work?

$$U_{n+1}^{k+1} = G(t_{n+1}, t_n, U_n^{k+1}) + F(t_{n+1}, t_n, U_n^k) - G(t_{n+1}, t_n, U_n^k)$$



Dominant part of the computation (F) is parallel (in time) !

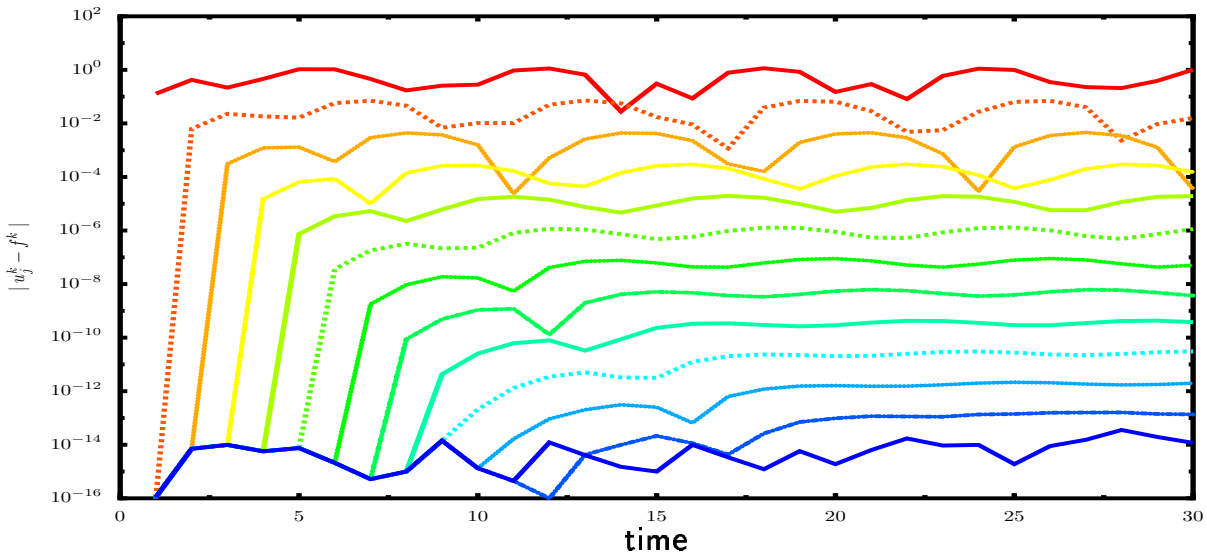
Original Convergence Result for Parareal

Theorem (Lions, Maday, and Turinici, 2001)

If $t_{n+1} - t_n = \Delta T$, G is $O(\Delta T)$ and F is exact, then at iteration k the error for a linear problem is $O(\Delta T^{k+1})$.

Example

$u' = -u + \sin t, \quad u(t_0) = 1.0, \ t \in [0, 30]$, trapezoidal rule, $\Delta T = 1.0$ and $\Delta t = 0.01$



Interpretation of Parareal

In the multiple shooting method

$$U_{n+1}^{k+1} = u_n(t_{n+1}, U_n^k) + \frac{\partial u_n}{\partial U_n}(t_{n+1}, U_n^k)(U_n^{k+1} - U_n^k).$$

one needs to compute or approximate the Jacobian. From the differential equation

$$u_n' = f(u_n), \quad u_n(t_n) = U_n.$$

we obtain the *linear* equation satisfied by the terms in the Jacobian:

$$\left(\frac{\partial u_n}{\partial U_n}\right)' = f'(u_n)\frac{\partial u_n}{\partial U_n}, \quad \frac{\partial u_n}{\partial U_n}(t_n) = I.$$

Multiple Shooting

Theorem (Chartier and Philippe 1993)

If the initial guess $\mathbf{U}^0$ *is close enough to the solution, then under appropriate regularity assumptions, the multiple shooting algorithm converges quadratically.*

Result (G, Vandevalle 2003)

Approximation of the Jacobian on a coarse time grid leads from

$$U_{n+1}^{k+1} = u_n(t_{n+1}, U_n^k) + \frac{\partial u_n}{\partial U_n}(t_{n+1}, U_n^k)(U_n^{k+1} - U_n^k).$$

to

$$U_{n+1}^{k+1} = F(t_{n+1}, t_n, U_n^k) + G(t_{n+1}, t_n, U_n^{k+1}) - G(t_{n+1}, t_n, U_n^k),$$

which is the parareal algorithm.

Parareal is a Time Multigrid Method

Theorem (G, Vandewalle, 2003)

Let F be method ϕ doing $\bar{m}$ steps and G be method Φ, and let $I_{\Delta t}^{\Delta T}$ be the selection operator at $1, \bar{m}+1, 2\bar{m}+1, \ldots$ and $I_{\Delta T}^{\Delta t}$ be the extension operator with 1 and any values in between.
If in the time multigrid algorithm

- *a block Jacobi smoother is used, $S = EM_{jac}^{-1}$, where $M_{jac} + N_{jac} = M$, and E is the identity, except for zeros at positions $(1,1)$, $(\bar{m}+1, \bar{m}+1)$, $(2\bar{m}+1, 2\bar{m}+1)$ $\ldots$*
- *The initial guess $\mathbf{u}^0$ contains U_n^0 from the parareal initial guess at positions 1, $\bar{m}+1$, $2\bar{m}+1$ $\ldots$*

then it coincides with the parareal algorithm.

Convergence Results for Linear Problems

For the linear model problem $u' = -au$, $u(0) = u_0$, $\Re(a) \geq 0$.

Theorem (Superlinear Convergence)

Let $F(t_{n+1}, t_n, U_n^k)$ denote the exact solution at t_{n+1} and $G(t_{n+1}, t_n, U_n^k) = \beta U_n^k$ be a one step method. If the method is in its region of absolute stability, $|\beta| \leq 1$, then at iteration k, we have

$$\max_{1\leq n\leq N} |u(t_n) - U_n^k| \leq \frac{|e^{-a\Delta T} - \beta|^k}{k!} \prod_{j=1}^{k} (N-j) \max_{1\leq n\leq N} |u(t_n) - U_n^0|.$$

If the local truncation error is bounded by $C\Delta T^{p+1}$, then

$$\max_{1\leq n\leq N} |u(t_n) - U_n^k| \leq \frac{(CT)^k}{k!} \Delta T^{pk} \max_{1\leq n\leq N} |u(t_n) - U_n^0|.$$

Convergence Results for Linear Problems

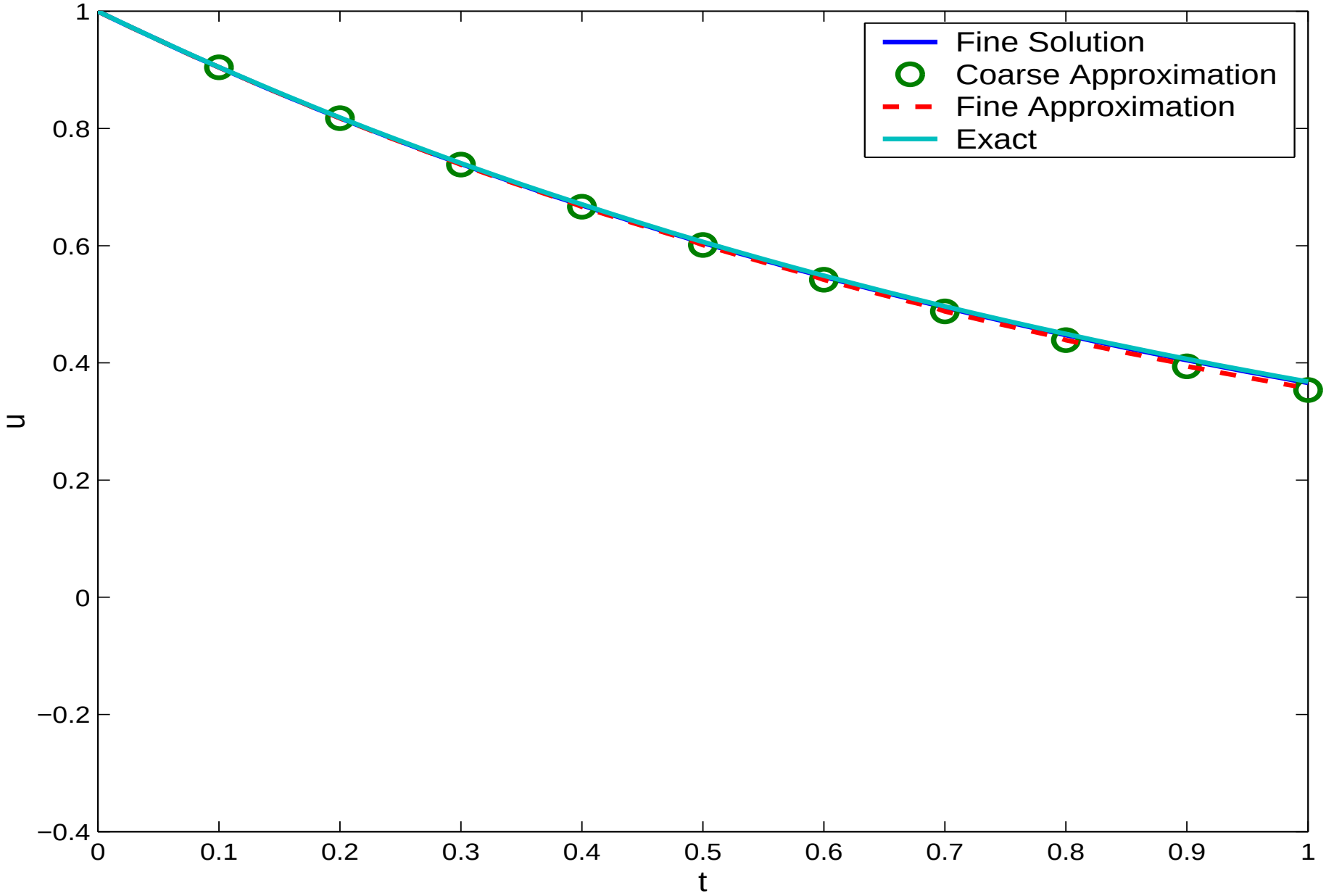
Theorem (Linear Convergence)

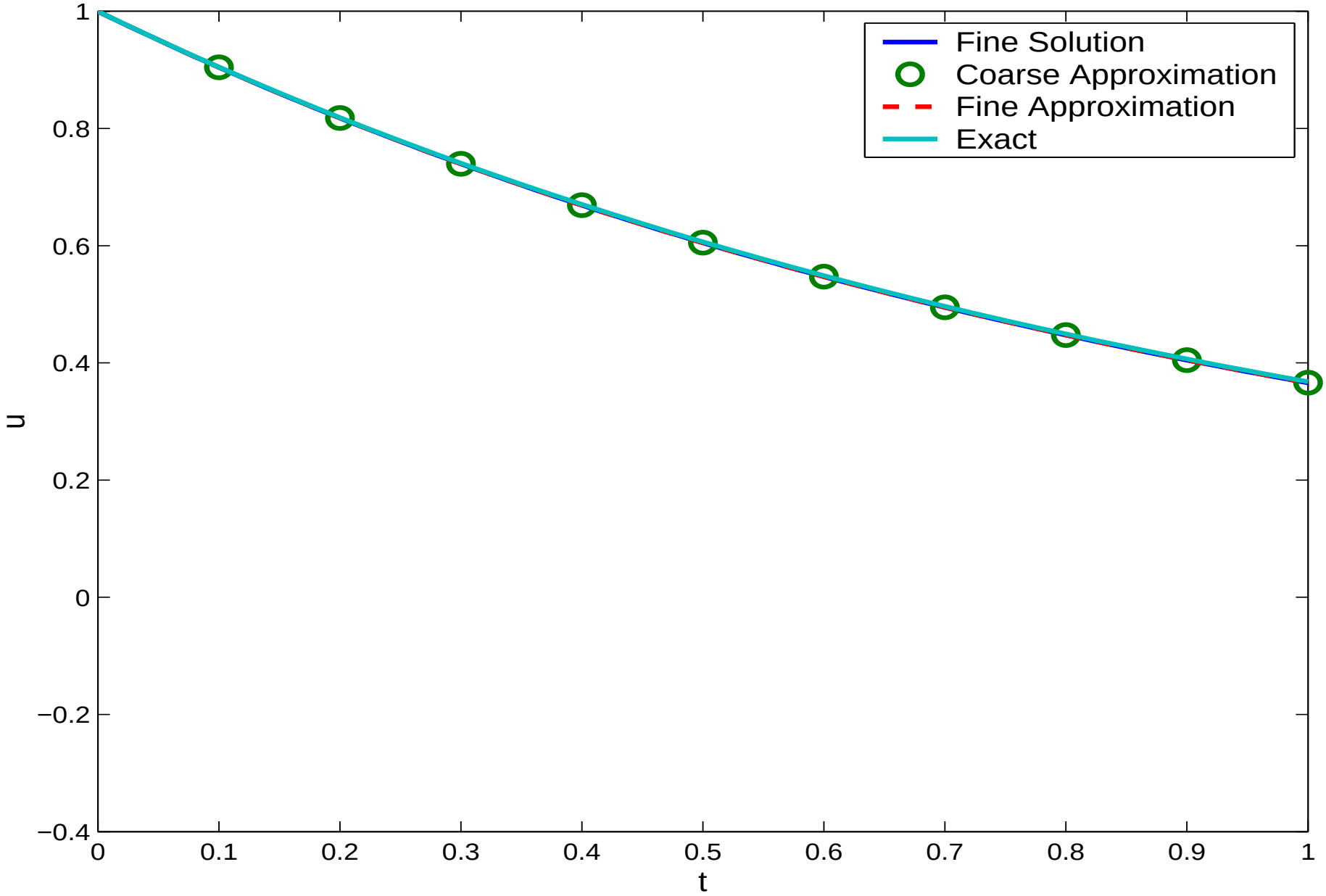
Let $F(t_{n+1}, t_n, U_n^k)$ denote the exact solution at t_{n+1} and $G(t_{n+1}, t_n, U_n^k) = \beta U_n^k$ be a one step method. If ΔT is such that the method is in its region of absolute stability, then at iteration k, we have

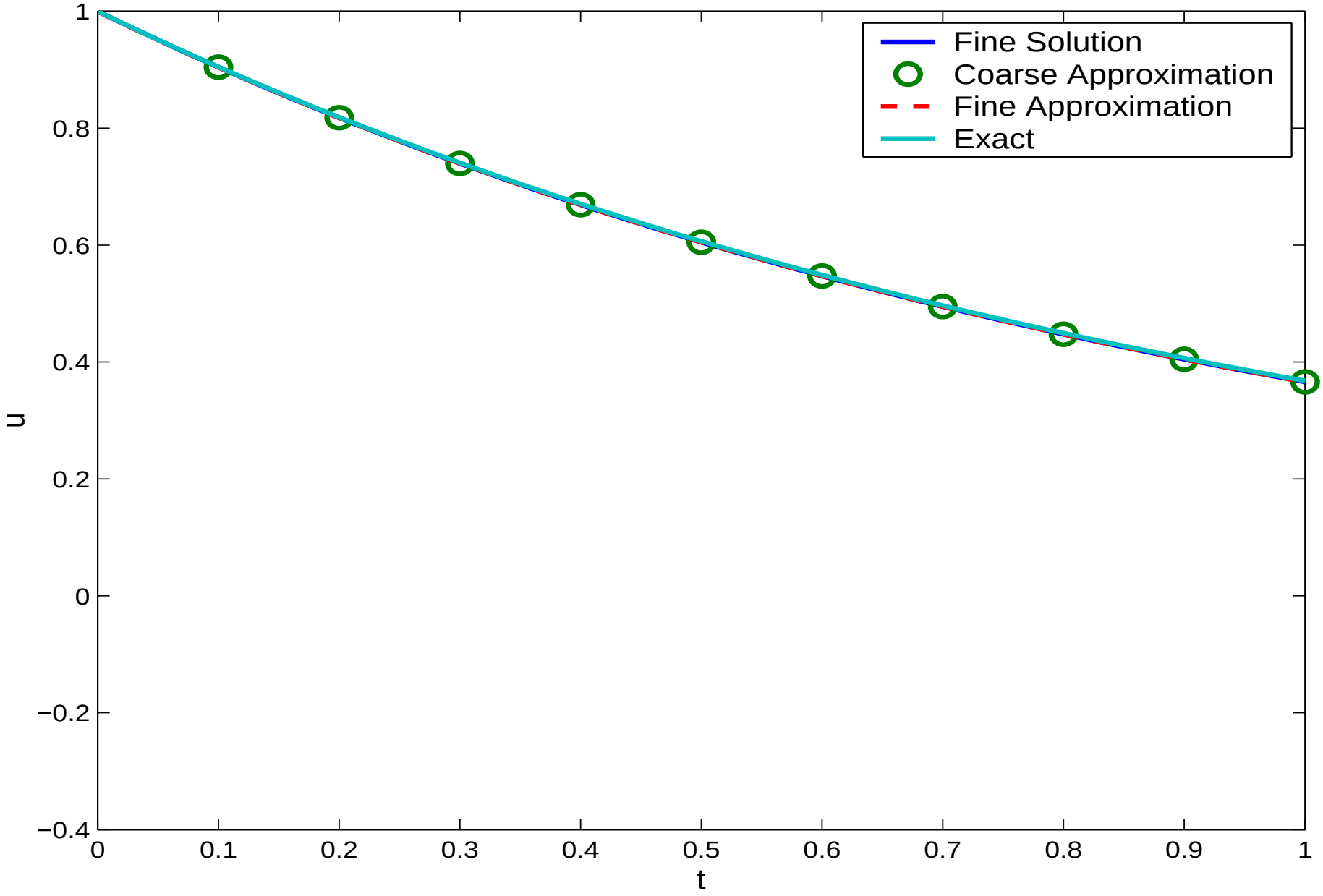
$$\sup_{n>0} |u(t_n) - U_n^k| \leq \left(\frac{|e^{-a\Delta T} - \beta|}{1 - |\beta|} \right)^k \sup_{n>0} |u(t_n) - U_n^0|.$$

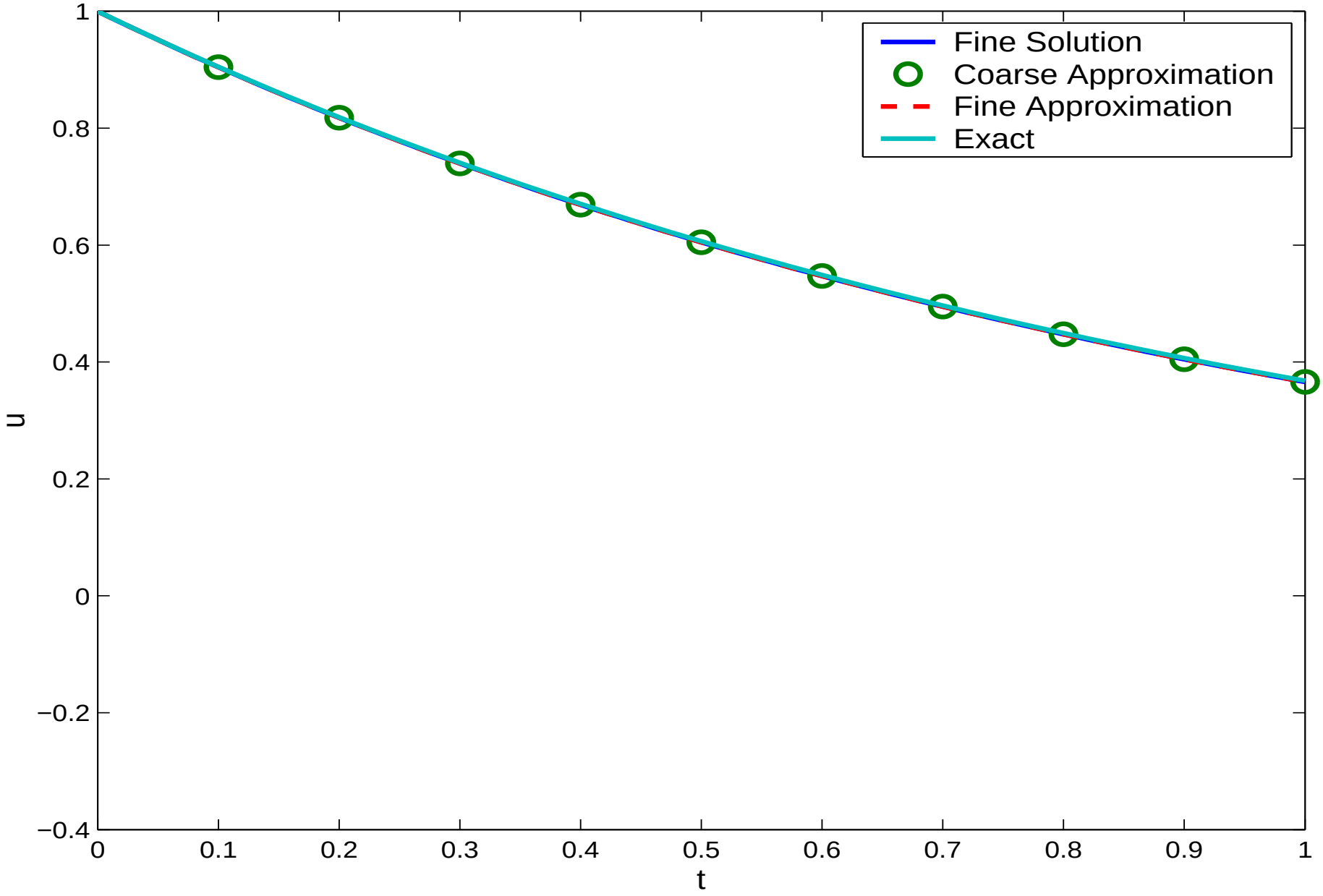
If the local truncation error is bounded by $C\Delta T^{p+1}$, then for ΔT small, we have

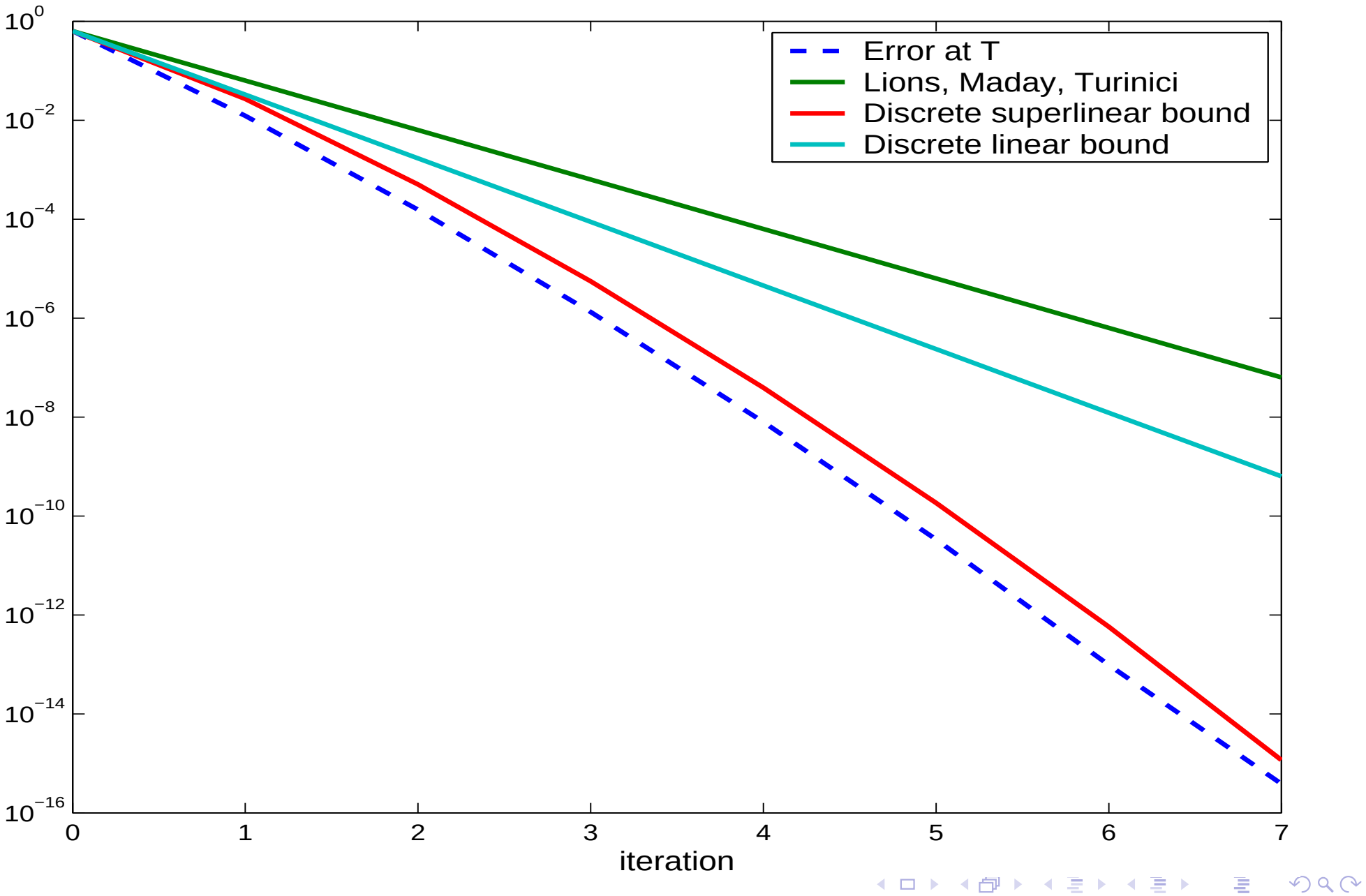
$$\sup_{n>0} |u(t_n) - U_n^k| \leq \left(\frac{C\Delta T^p}{\Re(a) + O(\Delta T)} \right)^k \sup_{n>0} |u(t_n) - U_n^0|.$$

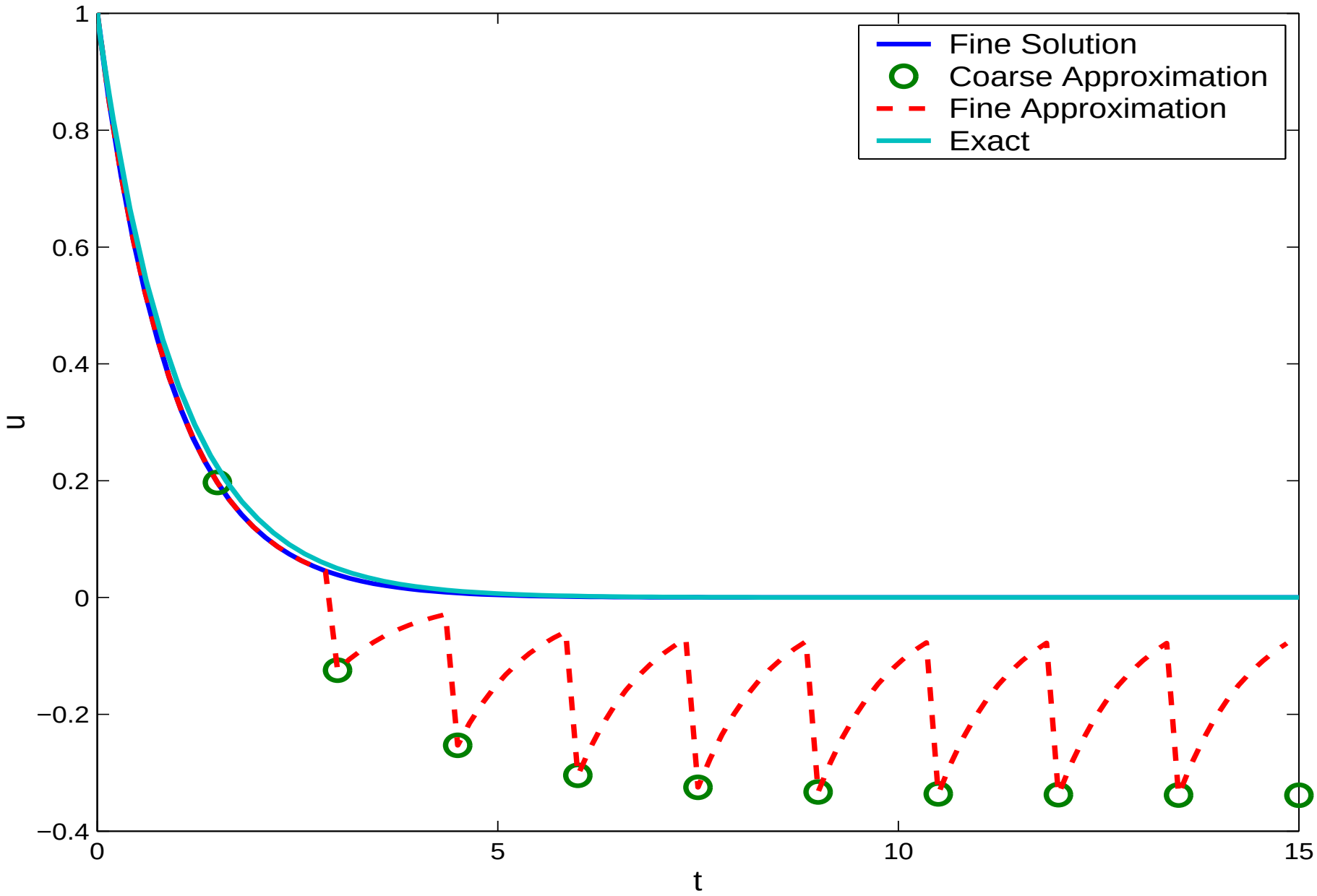

Fine Solution
Coarse Approximation
Fine Approximation
Exact
u
t
1
0.8
0.6
0.4
0.2
0
−0.2
−0.4
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

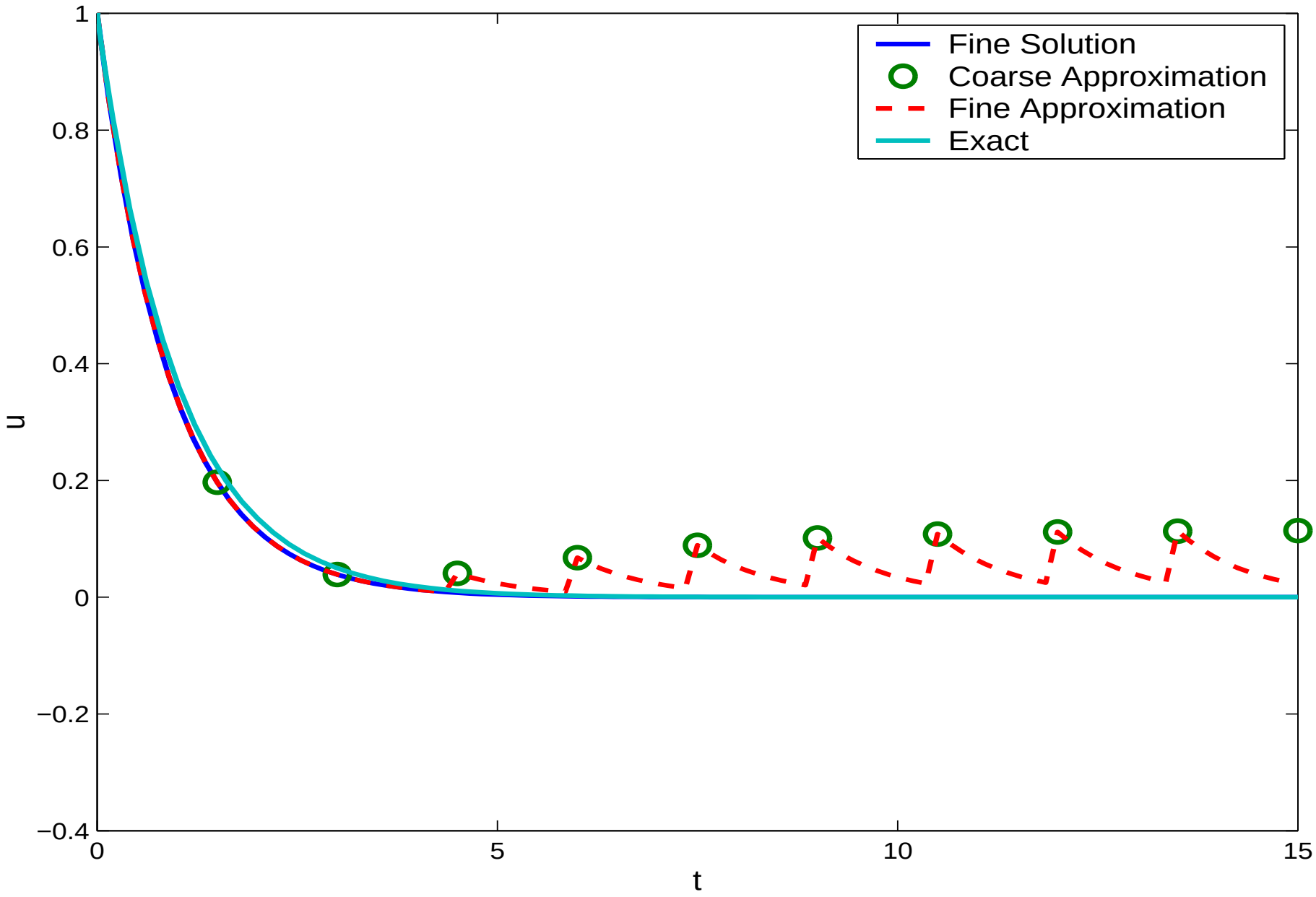

Fine Solution
Coarse Approximation
Fine Approximation
Exact
u
t
1
0.8
0.6
0.4
0.2
0
−0.2
−0.4
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

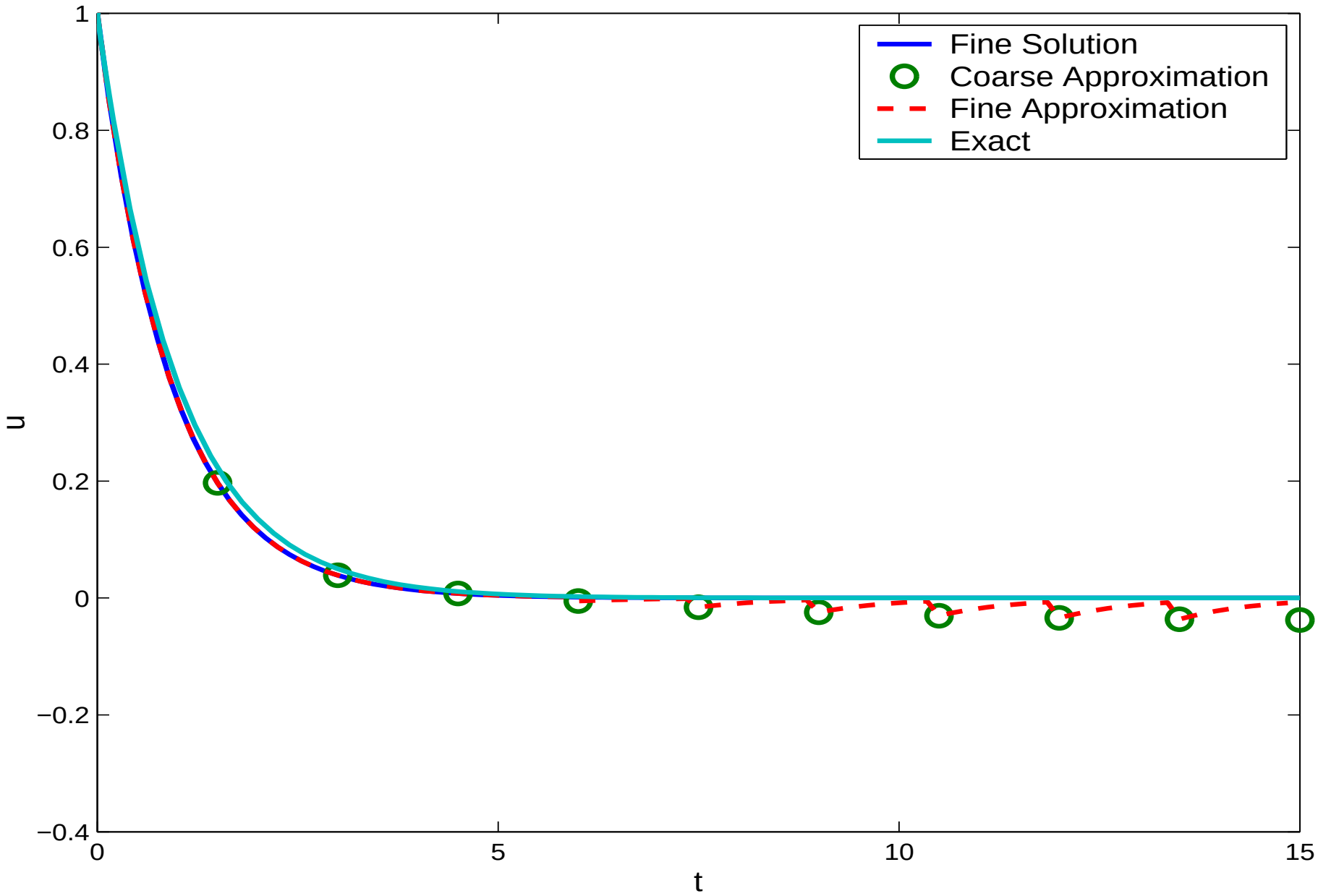

Fine Solution
Coarse Approximation
Fine Approximation
Exact
u
t
1
0.8
0.6
0.4
0.2
0
−0.2
−0.4
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

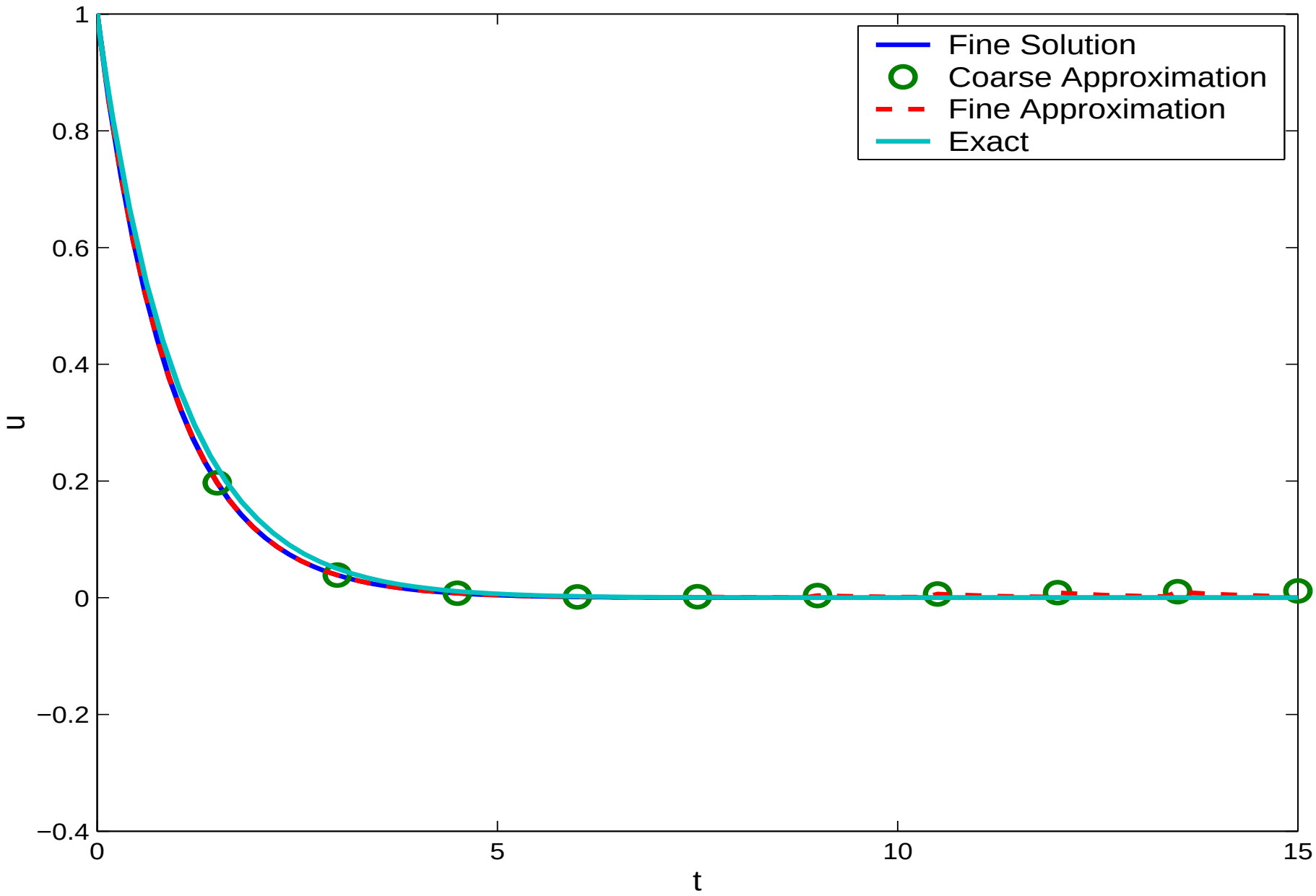

Fine Solution
Coarse Approximation
Fine Approximation
Exact
u
t
1
0.8
0.6
0.4
0.2
0
−0.2
−0.4
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

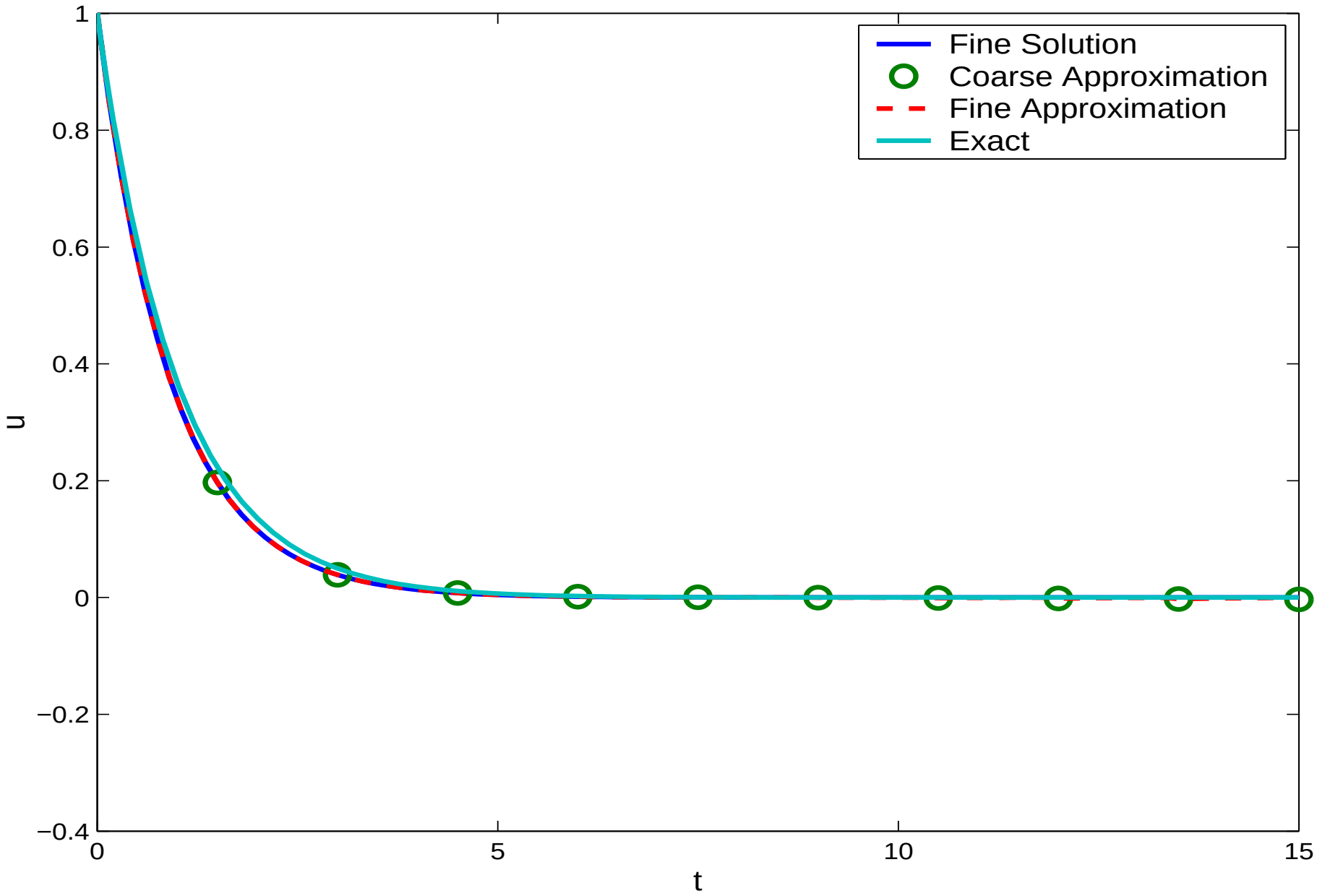

Error at T
Lions, Maday, Turinici
Discrete superlinear bound
Discrete linear bound
10^{0}
10^{-2}
10^{-4}
10^{-6}
10^{-8}
10^{-10}
10^{-12}
10^{-14}
10^{-16}
0
1
2
3
4
5
6
7
iteration

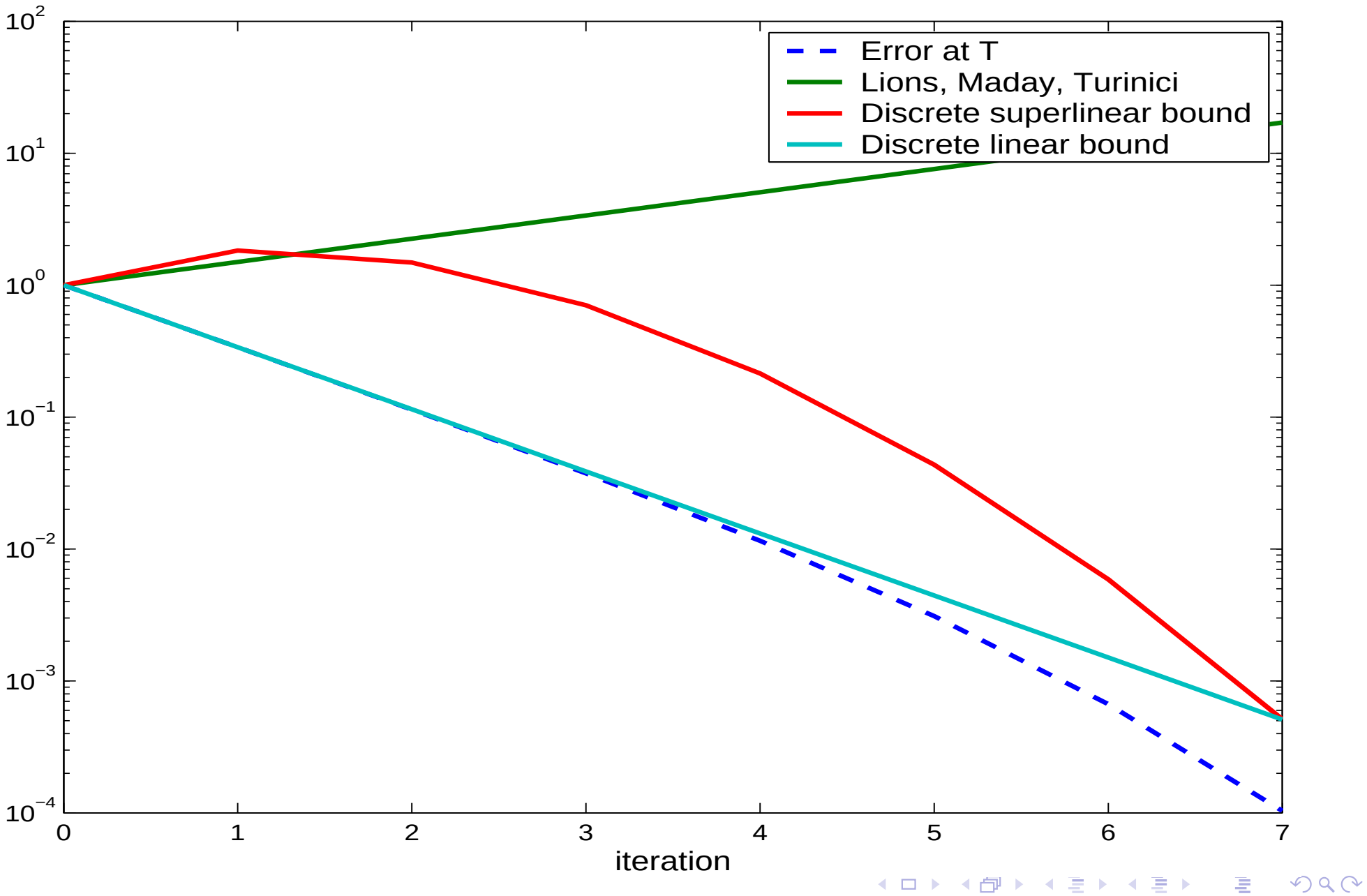

Fine Solution
Coarse Approximation
Fine Approximation
Exact
u
t
1
0.8
0.6
0.4
0.2
0
−0.2
−0.4
0
5
10
15

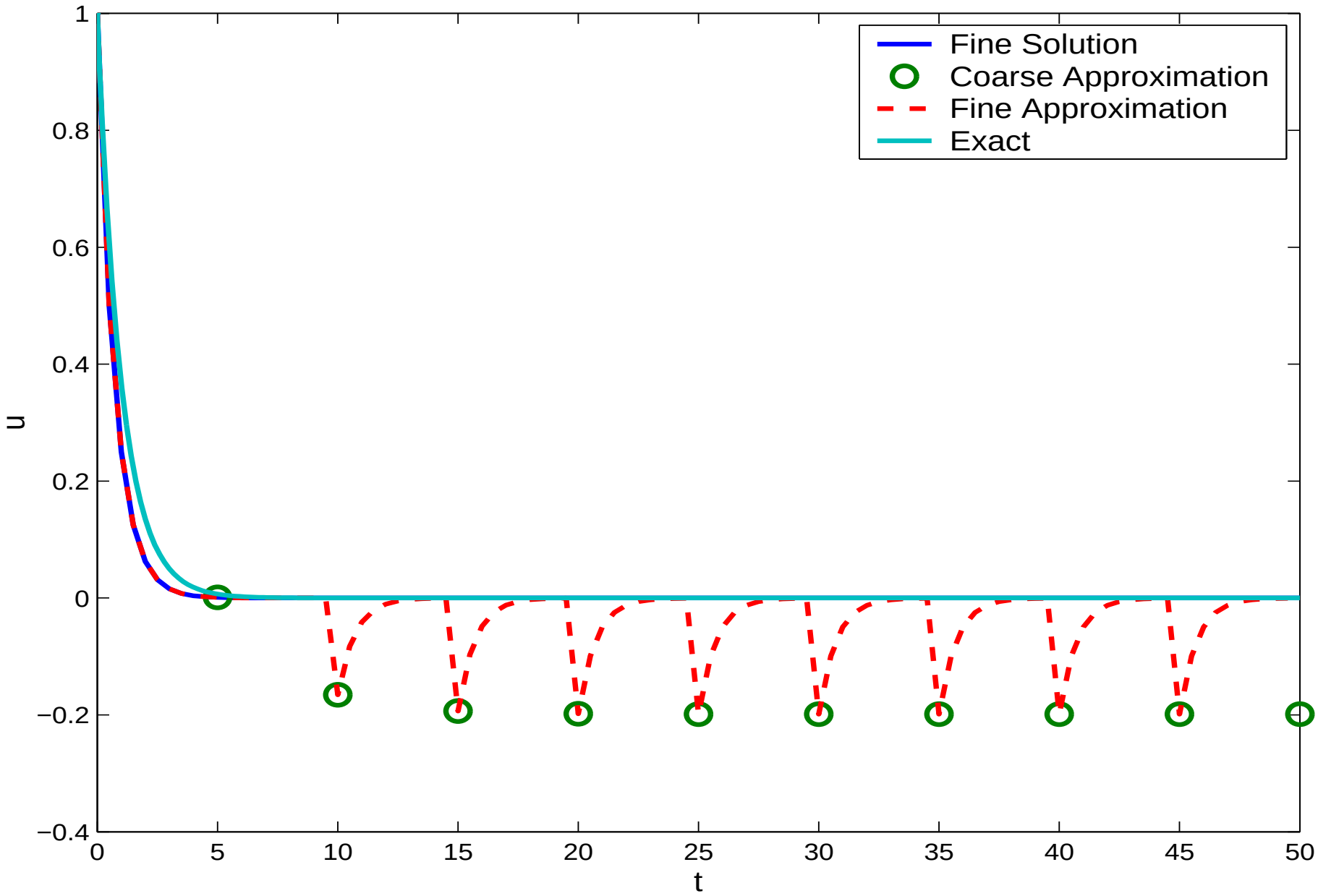

Fine Solution
Coarse Approximation
Fine Approximation
Exact
u
t
1
0.8
0.6
0.4
0.2
0
−0.2
−0.4
0
5
10
15

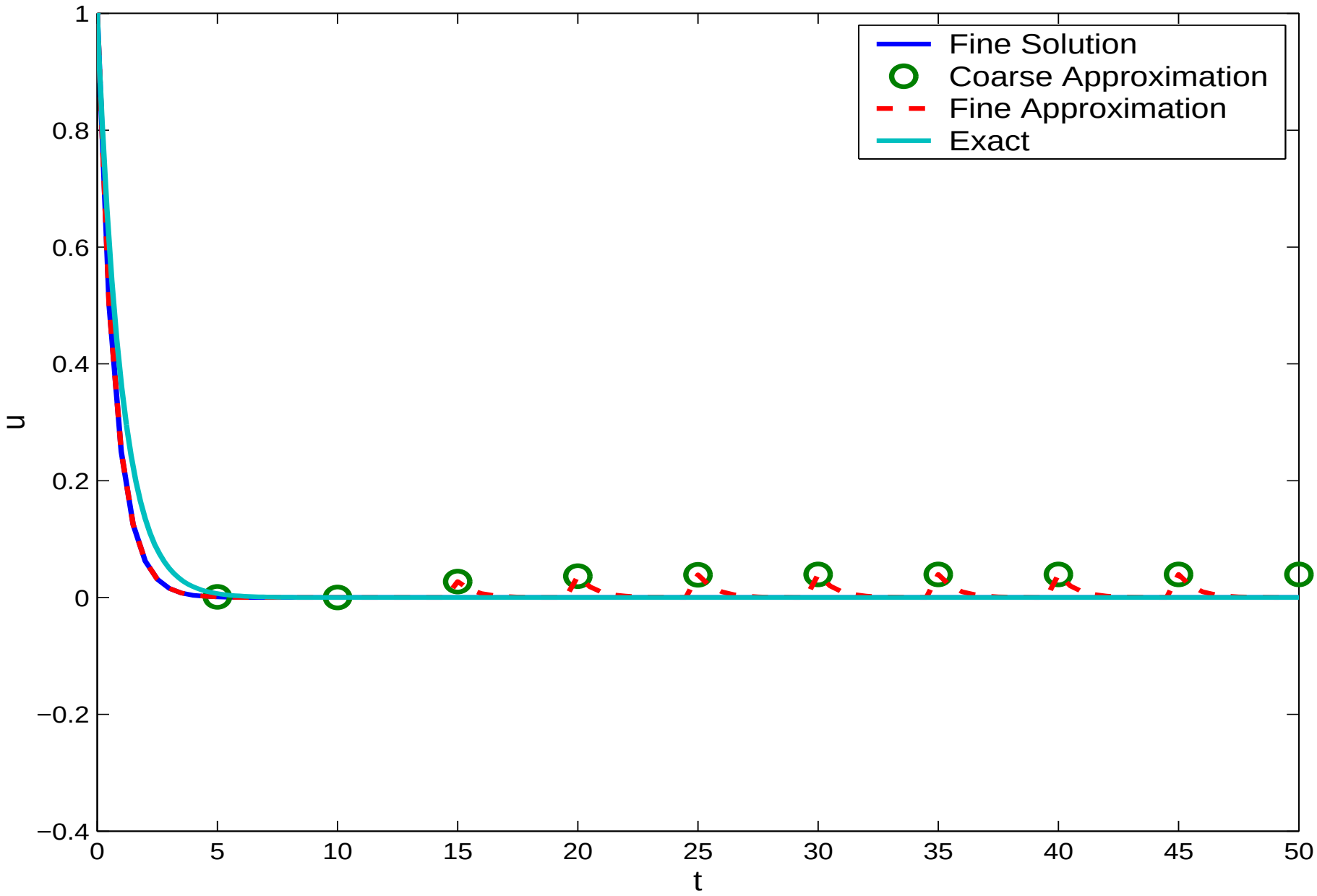

Fine Solution
Coarse Approximation
Fine Approximation
Exact
u
t
1
0.8
0.6
0.4
0.2
0
−0.2
−0.4
0
5
10
15

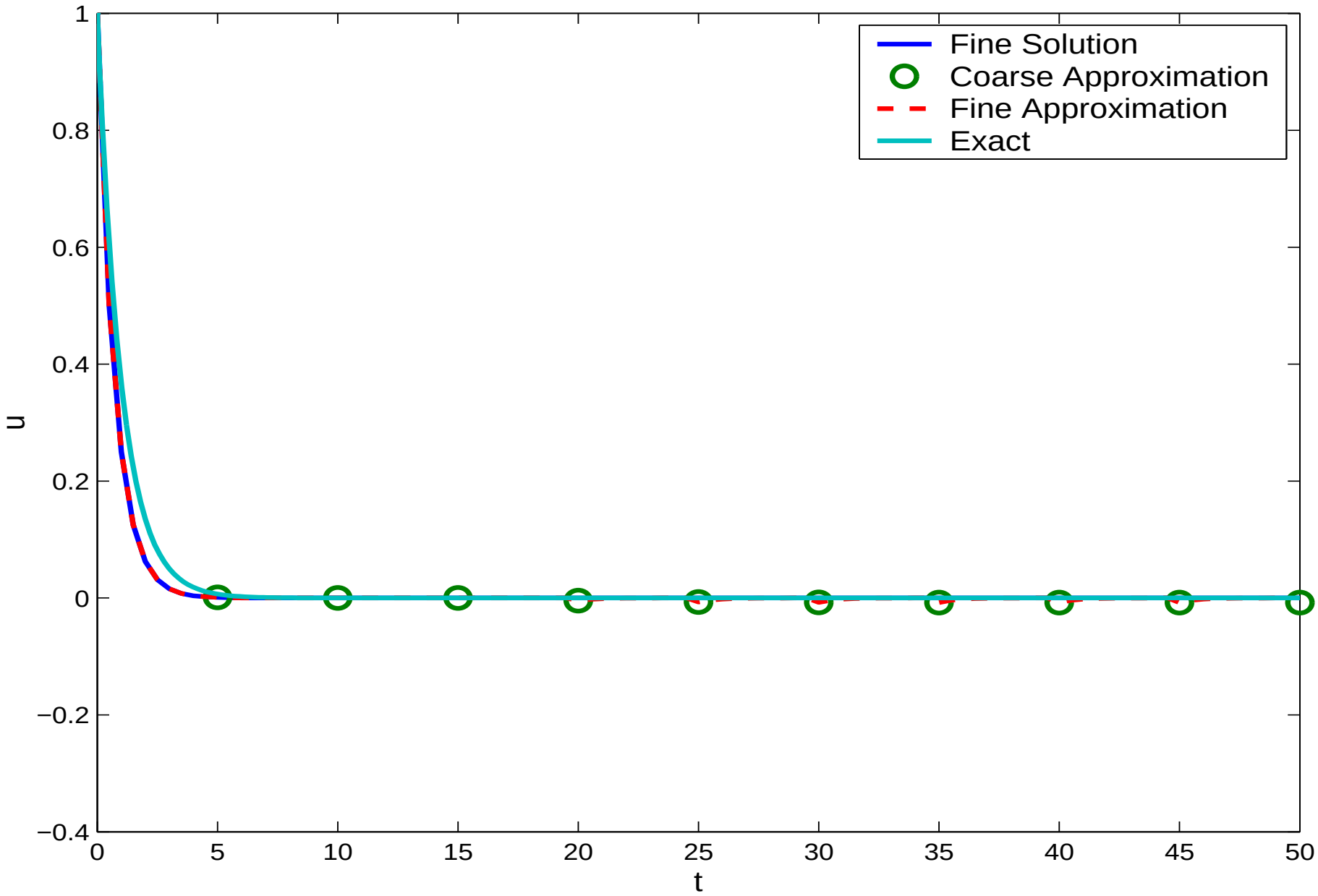

Fine Solution
Coarse Approximation
Fine Approximation
Exact
u
t
1
0.8
0.6
0.4
0.2
0
−0.2
−0.4
0
5
10
15

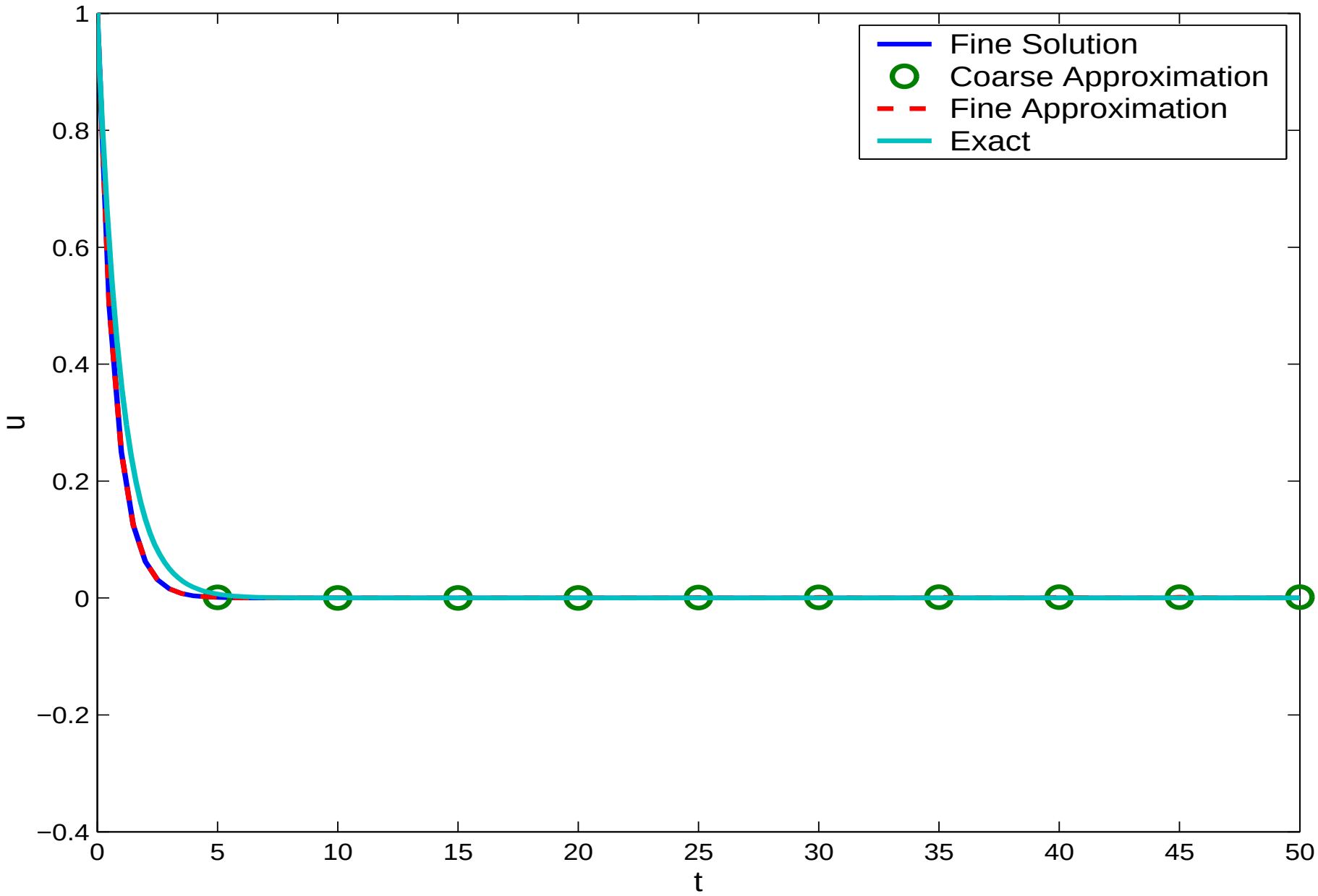

Fine Solution
Coarse Approximation
Fine Approximation
Exact
u
t
1
0.8
0.6
0.4
0.2
0
−0.2
−0.4
0
5
10
15

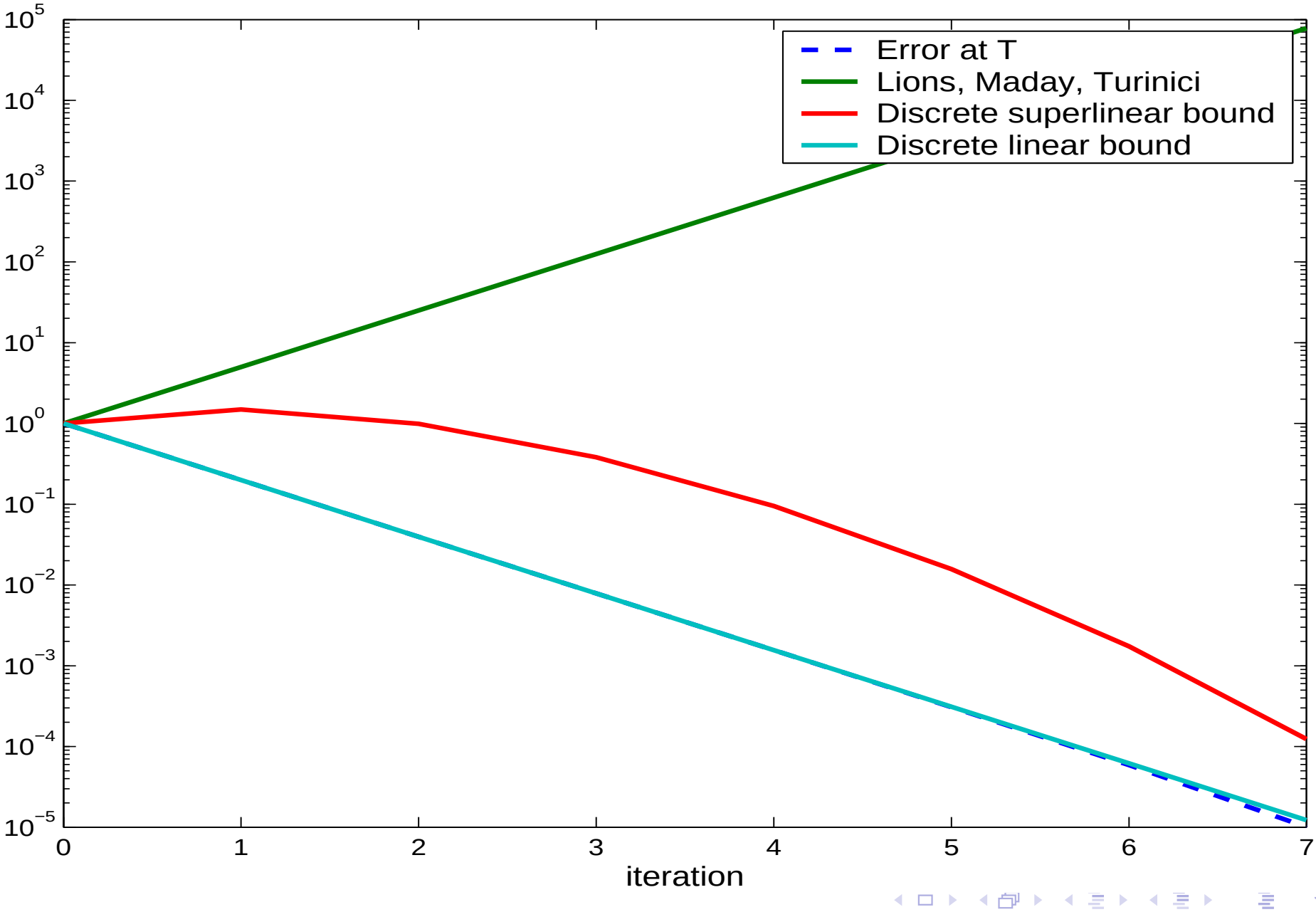

Error at T
Lions, Maday, Turinici
Discrete superlinear bound
Discrete linear bound
10^{2}
10^{1}
10^{0}
10^{-1}
10^{-2}
10^{-3}
10^{-4}
0
1
2
3
4
5
6
7
iteration


Fine Solution
Coarse Approximation
Fine Approximation
Exact
u
t
1
0.8
0.6
0.4
0.2
0
−0.2
−0.4
0
5
10
15
20
25
30
35
40
45
50

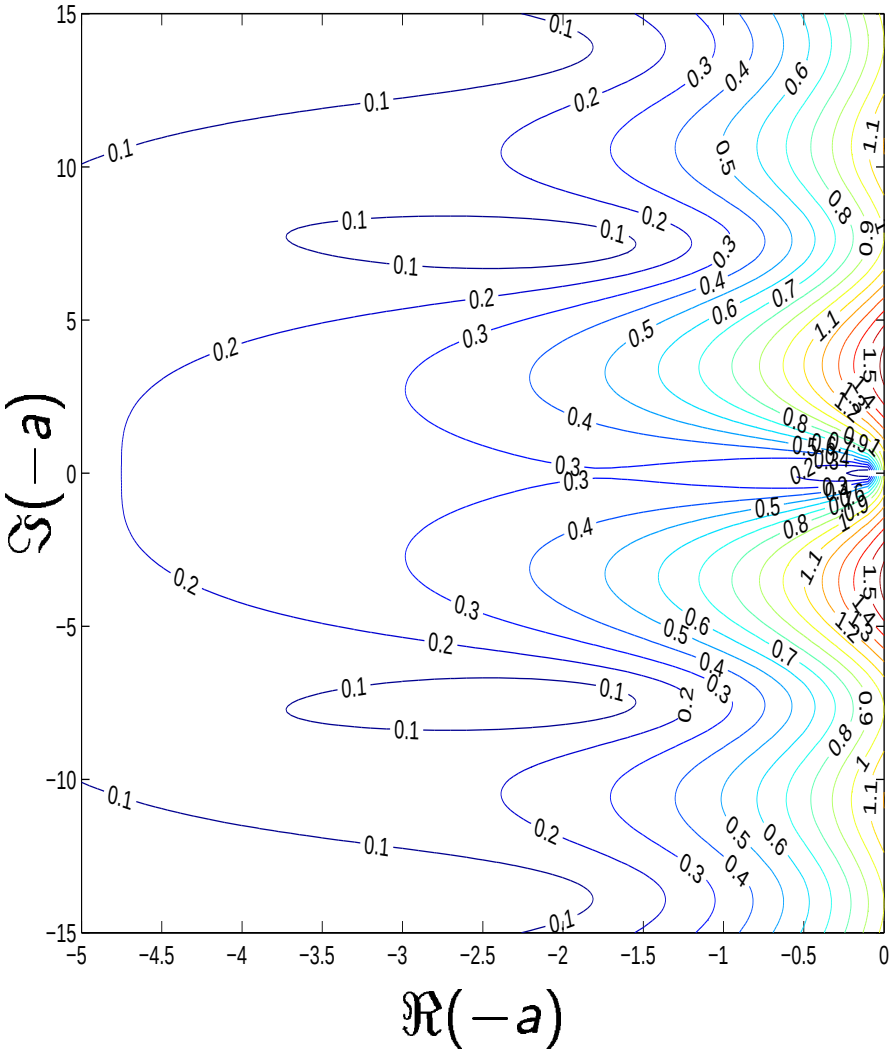

Fine Solution
Coarse Approximation
Fine Approximation
Exact
u
t
1
0.8
0.6
0.4
0.2
0
−0.2
−0.4
0
5
10
15
20
25
30
35
40
45
50


Fine Solution
Coarse Approximation
Fine Approximation
Exact
u
t
1
0.8
0.6
0.4
0.2
0
−0.2
−0.4
0
5
10
15
20
25
30
35
40
45
50

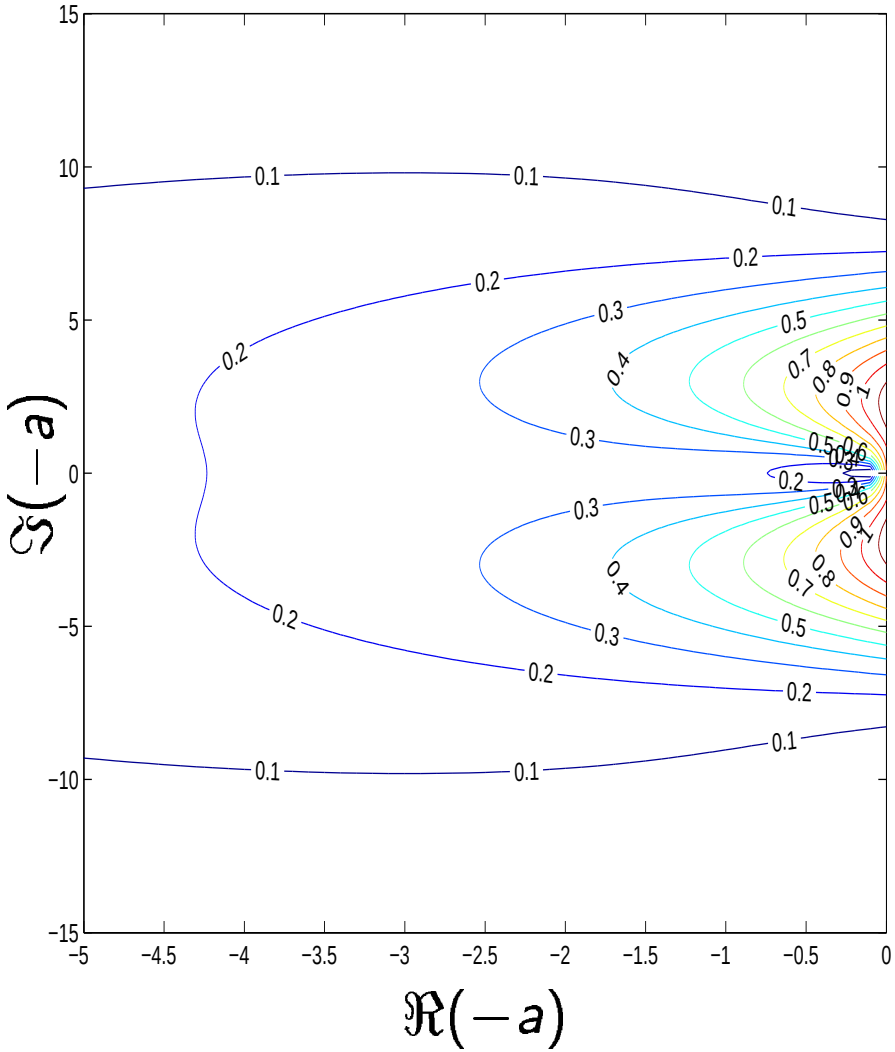

Fine Solution
Coarse Approximation
Fine Approximation
Exact
u
t
1
0.8
0.6
0.4
0.2
0
−0.2
−0.4
0
5
10
15
20
25
30
35
40
45
50


Error at T
Lions, Maday, Turinici
Discrete superlinear bound
Discrete linear bound
10^{5}
10^{4}
10^{3}
10^{2}
10^{1}
10^{0}
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
0
1
2
3
4
5
6
7
iteration

Contraction Factors for the Fully Discrete Case

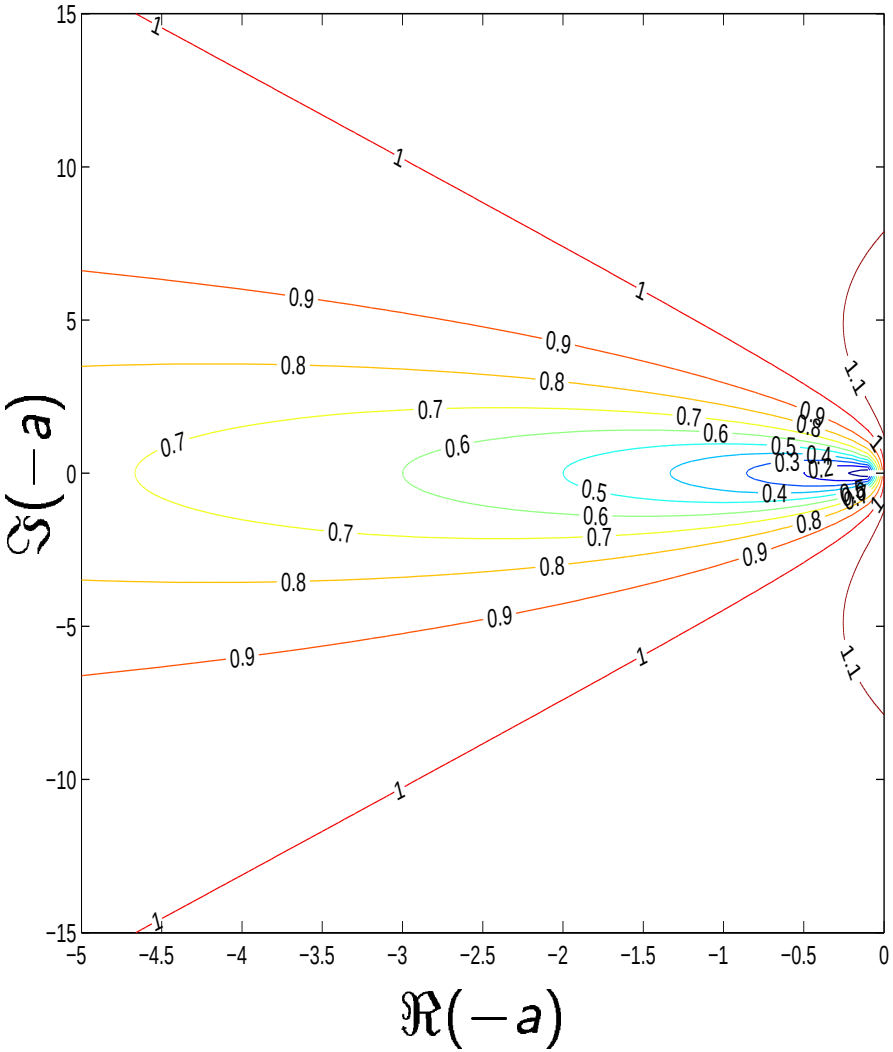


G: Backward Euler F: exact

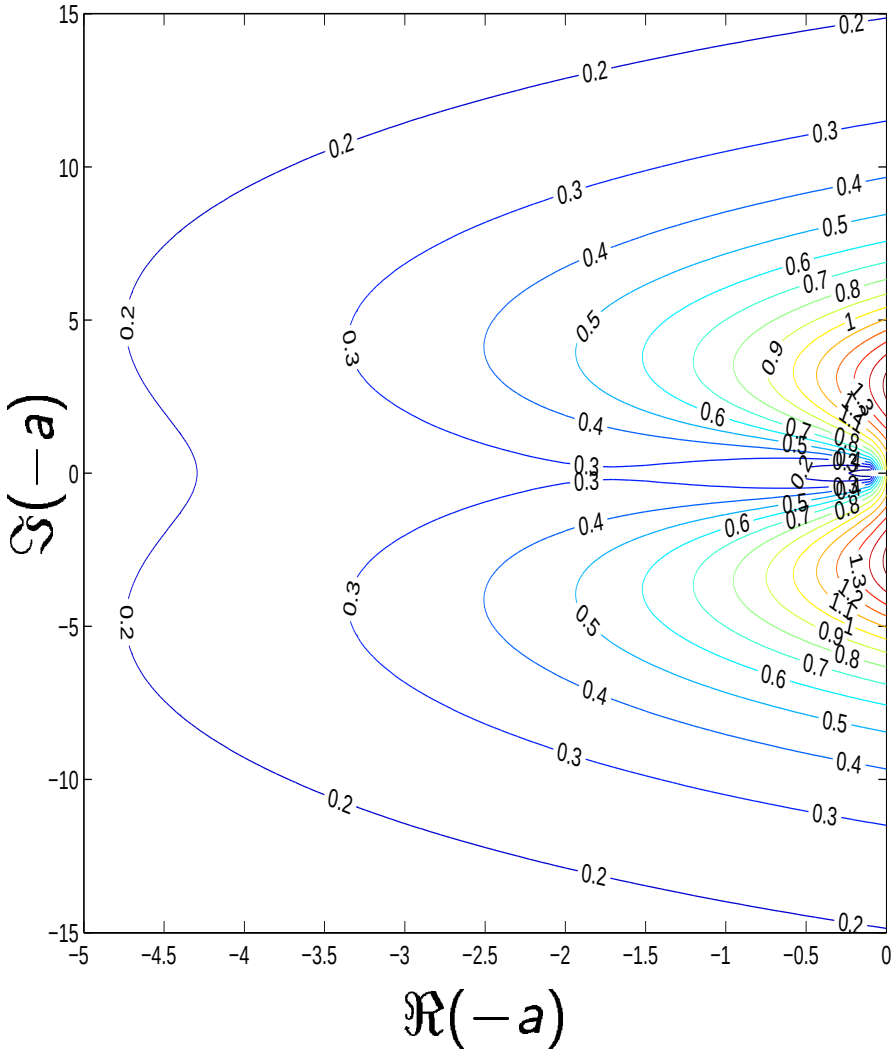


G: Backward Euler F: BE 10 steps

Contraction Factors for the Fully Discrete Case



G: Backward Euler F: Trapezoidal



G: Backward Euler F: Radau