Experiments with Schemes for Exponential Decay

Hans Petter Langtangen[1,2] (hpl@simula.no)

[1]Center for Biomedical Computing, Simula Research Laboratory
[2]Department of Informatics, University of Oslo.

Aug 28, 2014

Abstract

This report investigates the accuracy of three finite difference schemes for the ordinary differential equation $u' = -au$ with the aid of numerical experiments. Numerical artifacts are in particular demonstrated.

Contents

1 Mathematical problem

We address the initial-value problem

$$u'(t) = -au(t), \quad t \in (0, T], \tag{1}$$
$$u(0) = I, \tag{2}$$

where a, I, and T are prescribed parameters, and $u(t)$ is the unknown function to be estimated. This mathematical model is relevant for physical phenomena featuring exponential decay in time, e.g., vertical pressure variation in the atmosphere, cooling of an object, and radioactive decay.

2 Numerical solution method

We introduce a mesh in time with points $0 = t_0 < t_1 \cdots < t_{N_t} = T$. For simplicity, we assume constant spacing Δt between the mesh points: $\Delta t = t_n - t_{n-1}$, $n = 1, \ldots, N_t$. Let u^n be the numerical approximation to the exact solution at t_n.

The θ-rule [1] is used to solve (1) numerically:

$$u^{n+1} = \frac{1-(1-\theta)a\Delta t}{1+\theta a\Delta t}u^n,$$

for $n = 0, 1, \ldots, N_t - 1$. This scheme corresponds to

- The Forward Euler[1] scheme when $\theta = 0$
- The Backward Euler[2] scheme when $\theta = 1$
- The Crank-Nicolson[3] scheme when $\theta = 1/2$

3 Implementation

The numerical method is implemented in a Python function [2] `solver` (found in the `decay_mod`[4] module):

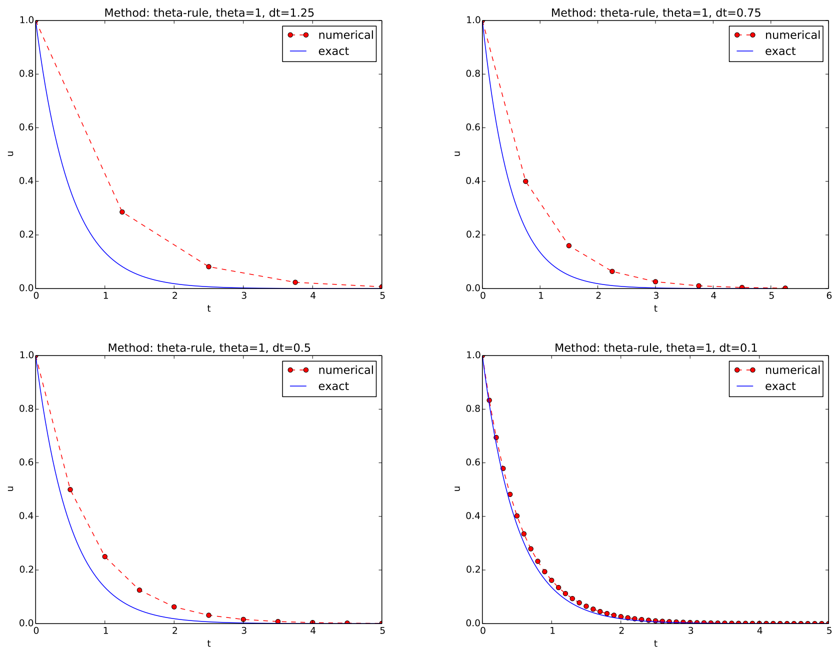
```
def solver(I, a, T, dt, theta):
    """Solve u'=-a*u, u(0)=I, for t in (0,T] with steps of dt."""
    dt = float(dt)            # avoid integer division
    Nt = int(round(T/dt))     # no of time intervals
    T = Nt*dt                 # adjust T to fit time step dt
    u = zeros(Nt+1)           # array of u[n] values
    t = linspace(0, T, Nt+1)  # time mesh

    u[0] = I                  # assign initial condition
    for n in range(0, Nt):    # n=0,1,...,Nt-1
        u[n+1] = (1 - (1-theta)*a*dt)/(1 + theta*dt*a)*u[n]
    return u, t
```

4 Numerical experiments

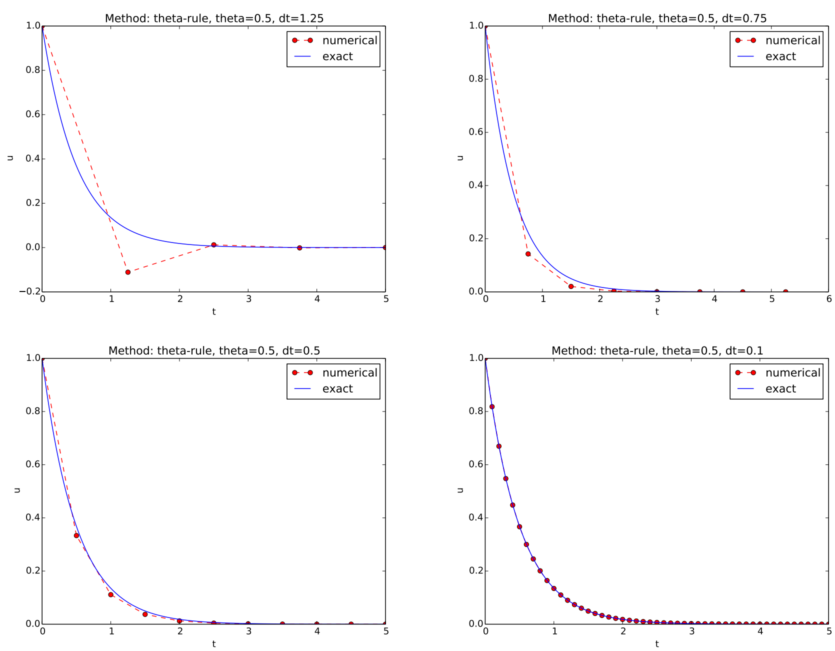
We define a set of numerical experiments where I, a, and T are fixed, while Δt and θ are varied. In particular, $I = 1$, $a = 2$, $\Delta t = 1.25, 0.75, 0.5, 0.1$.

[1] http://en.wikipedia.org/wiki/Forward_Euler_method
[2] http://en.wikipedia.org/wiki/Backward_Euler_method
[3] http://en.wikipedia.org/wiki/Crank-Nicolson
[4] https://github.com/hplgit/INF5620/blob/gh-pages/src/decay/experiments/dc_mod.py

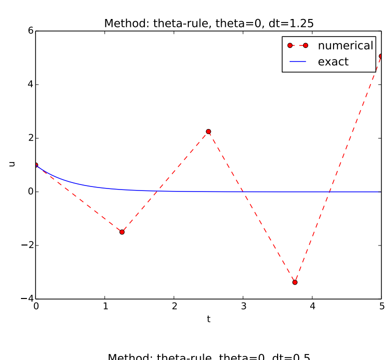
4.1 The Backward Euler method

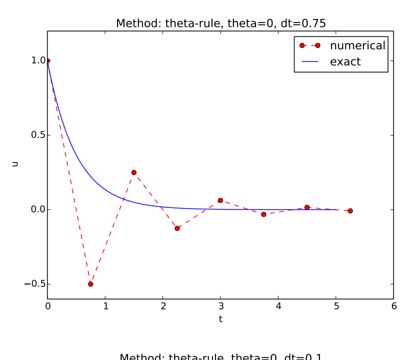


4.2 The Crank-Nicolson method



4.3 The Forward Euler method





Observe:
The data points for the three largest Δt values in the Forward Euler method are not relevant as the solution behaves non-physically.

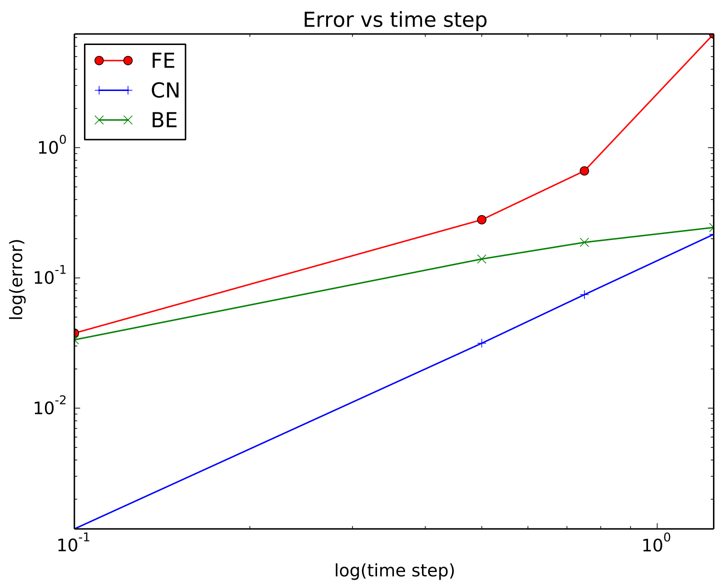


Figure 1: Variation of the error with the time step.

The numbers corresponding to the figure above are given in the table below.

Δt	$\theta = 0$	$\theta = 0.5$	$\theta = 1$
1.25	7.4630	0.2161	0.2440
0.75	0.6632	0.0744	0.1875
0.50	0.2797	0.0315	0.1397
0.10	0.0377	0.0012	0.0335

Summary

1. $\theta = 1$: $E \sim \Delta t$ (first-order convergence).
2. $\theta = 0.5$: $E \sim \Delta t^2$ (second-order convergence).
3. $\theta = 1$ is always stable and gives qualitatively corrects results.

4. $\theta = 0.5$ never blows up, but may give oscillating solutions if Δt is not sufficiently small.

5. $\theta = 0$ suffers from fast-growing solution if Δt is not small enough, but even below this limit one can have oscillating solutions that disappear if Δt is sufficiently small.

References

[1] A. Iserles. *A First Course in the Numerical Analysis of Differential Equations*. Cambridge Texts in Applied Mathematics. Cambridge University Press, second edition, 2009.

[2] H. P. Langtangen. *A Primer on Scientific Programming With Python*. Texts in Computational Science and Engineering. Springer, third edition, 2012.