

# Experiments with Schemes for Exponential Decay

Hans Petter Langtangen<sup>1,2</sup> (hpl@simula.no)

<sup>1</sup>Center for Biomedical Computing, Simula Research Laboratory

<sup>2</sup>Department of Informatics, University of Oslo.

Feb 23, 2013

## 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	2
2	Numerical solution method	2
3	Implementation	3

<b>4</b>	<b>Numerical experiments</b>	<b>4</b>
4.1	The Backward Euler method . . . . .	4
4.2	The Crank-Nicolson method . . . . .	5
4.3	The Forward Euler method . . . . .	6
4.4	Error vs $\Delta t$ . . . . .	6

# 1 Mathematical problem

We address the initial-value problem

$$u'(t) = -au(t), \quad t \in (0, T], \quad (1)$$

$$u(0) = I, \quad (2)$$

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.

# 2 Numerical solution method

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

The  $\theta$ -rule is used to solve (1) numerically:

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

for  $n = 0, 1, \dots, N - 1$ . This scheme corresponds to

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

### 3 Implementation

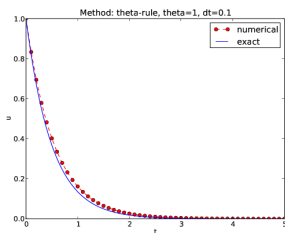
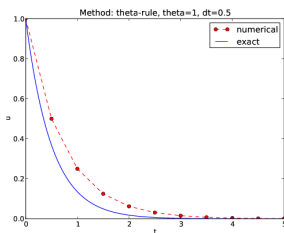
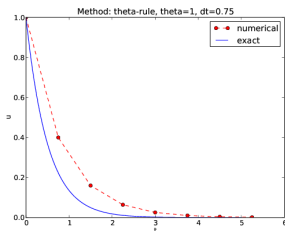
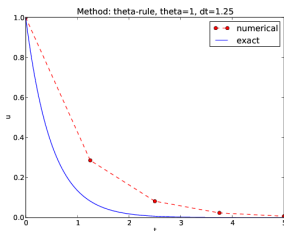
The numerical method is implemented in a Python function solver (found in the `decay_mod` module):

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

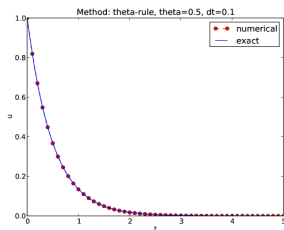
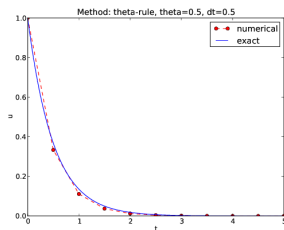
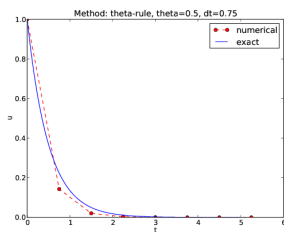
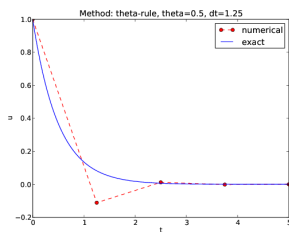
## 4 Numerical experiments

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

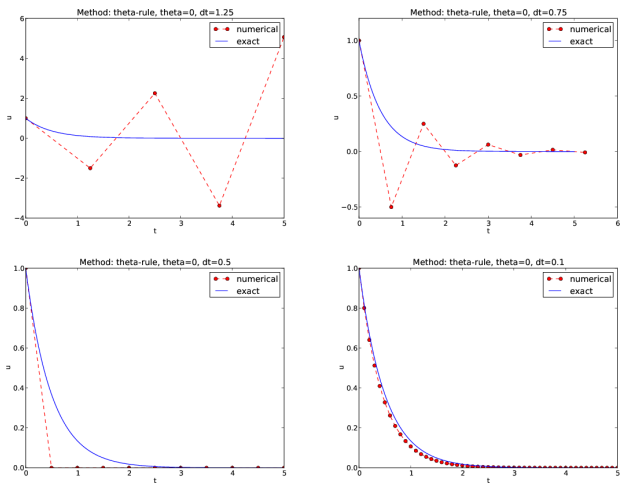
### 4.1 The Backward Euler method



## 4.2 The Crank-Nicolson method



## 4.3 The Forward Euler method



## 4.4 Error vs $\Delta t$

How  $E$  varies with  $\Delta t$  for  $\theta = 0, 0.5, 1$  is shown in Figure 1.

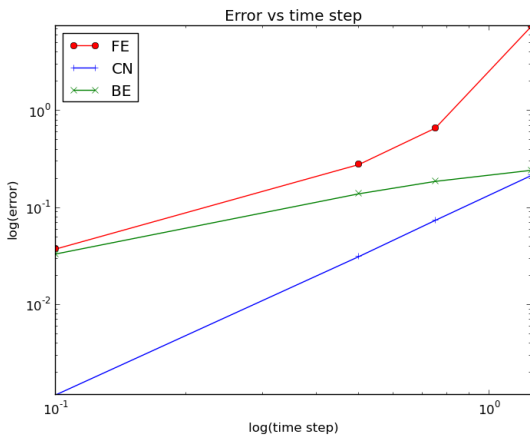


Figure 1: Error versus time step.