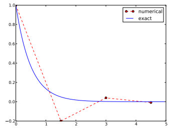
# On Schemes for Exponential Decay

Hans Petter Langtangen[1,2]

Center for Biomedical Computing, Simula Research Laboratory[1]

Department of Informatics, University of Oslo[2]

Aug 15, 2014




## Goal

The primary goal of this demo talk is to demonstrate how to write talks with DocOnce and get them rendered in numerous HTML formats.

**Layout.**

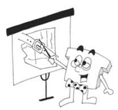
This version utilizes beamer slides with the theme `red_shadow`.

## Mathematical problem

$$u'(t) = -au(t), \qquad (1)$$

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

- $t \in (0, T]$
- $a$, $I$, and $T$ are prescribed parameters
- $u(t)$ is the unknown function



## Numerical solution method

- Mesh in time: $0 = t_0 < t_1 \cdots < t_N = T$
- Assume constant $\Delta t = t_n - t_{n-1}$
- $u^n$: numerical approx to the exact solution at $t_n$

Numerical scheme:

$$u^{n+1} = \frac{1 - (1-\theta)a\Delta t}{1 + \theta a\Delta t}u^n, \quad n = 0, 1, \ldots, N-1$$

## Forward Euler explained

http://youtube.com/PtJrPEIHNJw

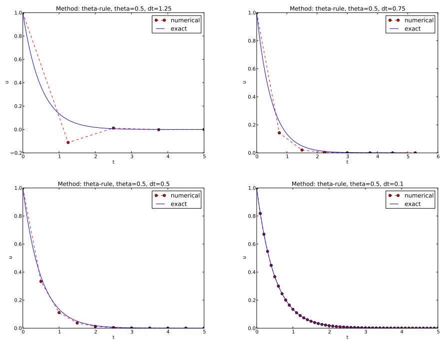
## Implementation

The numerical method is implemented in a Python function:

```python
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
    N = int(round(T/dt))         # no of time intervals
    T = N*dt                     # adjust T to fit time step dt
    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,...,N-1
        u[n+1] = (1 - (1-theta)*a*dt)/(1 + theta*dt*a)*u[n]
    return u, t
```

## The Crank-Nicolson method




## The artifacts can be explained by some theory

Exact solution of the scheme:

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

- Stability: $|A| < 1$
- No oscillations: $A > 0$
- Always for Backward Euler ($\theta = 1$)
- $\Delta t < 1/a$ for Forward Euler ($\theta = 0$)
- $\Delta t < 2/a$ for Crank-Nicolson ($\theta = 1/2$)

**Concluding remarks:**

Only the Backward Euler scheme is guaranteed to always give qualitatively correct results.