Testing HTML Slide Environments

Hans Petter Langtangen

Center for Biomedical Computing, Simula Research Laboratory

Feb 23, 2016

DocOnce: example on slide code with cells

One can introduce a table-like layout with MxN cells and put slide elements in various cell. A cell with position MN is surrounded by `!bslidecell MN` and `!eslidecell` tags. Below is an example with a bullet list to the left and a figure to the right (two cells, numbered 00 and 01).

```
!split
======= Headline =======

!bslidecell 00
 * Key point 1
 * Key point 2
 * Key point 3 takes very much more text to explain because
   this point is really comprehensive, and although long
   bullet points are not recommended in general, we need
   it here for demonstration purposes

!bt
\[ -\nabla^2 u = f \quad\hbox{in }\Omega \]
!et

!eslidecell

!bslidecell 01
FIGURE: [fig/broken_pen_and_paper, width=400]
!eslidecell

!split
======= Next slide... =======
```

DocOnce: example on slide code

Last page gets rendered to

Headline

- Key point 1
- Key point 2
- Key point 3 takes very much more text to explain because this point is really comprehensive, and although long bullet points are not recommended in general, we need it here for demonstration purposes

$$-\nabla^2 u = f \quad \text{in } \Omega$$

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:

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

$\theta = 0$: Forward Euler, $\theta = 1$: Backward Euler, $\theta = 1/2$: Crank-Nicolson

Implementation: Python program is embedded in a web service, Python Online Tutor, which enables stepwise execution and examination of variables

```
def solver(I, a, T, dt, theta):
    dt = float(dt)
    N = int(round(T/dt))
    T = N*dt
    u = [0.0]*(N+1)
    t = [i*dt for i in range(N+1)]

    u[0] = I
    for n in range(0, N):
        u[n+1] = (1 - (1-theta)*a*dt)/(1 + theta*dt*a)*u[n]
    return u, t

u, t = solver(I=1, a=1, T=3, dt=1., theta=0.5)
print u
```

(Visualize execution)