Solving partial differential equations (PDEs)

Hans Fangohr

Engineering and the Environment
University of Southampton
United Kingdom
fangohr@soton.ac.uk

May 3, 2012

Outline I

This lecture

- tries to compress several years of material into 45 minutes
- has lecture notes and code available for download at http://www.soton.ac.uk/∼fangohr/teaching/comp6024

What are partial differential equations (PDEs)

- Ordinary Differential Equations (ODEs)
  - one independent variable, for example t in

$$\frac{\mathrm{d}^2 x}{\mathrm{d}t^2} = -\frac{k}{m} x$$

  - often the indepent variable t is the time
  - solution is function $x(t)$
  - important for dynamical systems, population growth, control, moving particles
- Partial Differential Equations (ODEs)
  - multiple independent variables, for example t, x and y in

$$\frac{\partial u}{\partial t} = D \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right)$$

  - solution is function $u(t, x, y)$
  - important for fluid dynamics, chemistry, electromagnetism, ..., generally problems with spatial resolution

2d Diffusion equation

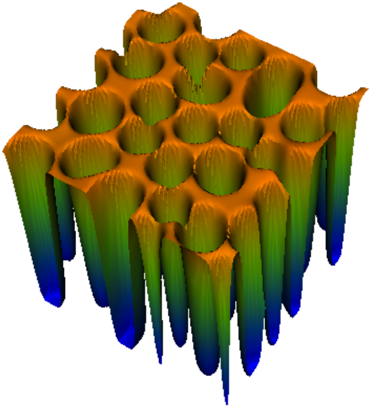
$$\frac{\partial u}{\partial t} = D\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right)$$

- $u(t, x, y)$ is the concentration [mol/m^3]
- t is the time [s]
- x is the x-coordinate [m]
- y is the y-coordinate [m]
- D is the diffusion coefficient [m^2/s]

Also known as Fick's second law. The heat equation has the same structure (and u represents the temperature).
Example:
http://www.youtube.com/watch?v=WC6Kj5ySWkQ

Examples of PDEs

- Cahn Hilliard Equation (phase separation)



- Fluid dynamics (including ocean and atmospheric models, plasma physics, gas turbine and aircraft modelling)
- Structural mechanics and vibrations, superconductivity, micromagnetics, ...

Computing derivatives using finite differences

Overview

- Motivation:
  - We need derivatives of functions for example for optimisation and root finding algorithms
  - Not always is the function analytically known (but we are usually able to compute the function numerically)
  - The material presented here forms the basis of the finite-difference technique that is commonly used to solve ordinary and partial differential equations.
- The following slides show
  - the forward difference technique
  - the backward difference technique and the
  - central difference technique to approximate the derivative of a function.
  - We also derive the accuracy of each of these methods.

The 1st derivative

- (Possible) Definition of the derivative (or "*differential* operator" $\frac{\mathrm{d}}{\mathrm{d}x}$)

$$f'(x) \equiv \frac{\mathrm{d}f}{\mathrm{d}x}(x) = \lim_{h\to 0} \frac{f(x+h) - f(x)}{h}$$

- Use *difference* operator to approximate differential operator

$$f'(x) = \frac{\mathrm{d}f}{\mathrm{d}x}(x) = \lim_{h\to 0} \frac{f(x+h) - f(x)}{h} \approx \frac{f(x+h) - f(x)}{h}$$

- $\Rightarrow$ can now compute *an approximation* of $f'(x)$ simply by evaluating f (twice).
- This is called the *forward difference* because we use $f(x)$ and $f(x+h)$.
- Important questions: How accurate is this approximation?

Accuracy of the forward difference

- Formal derivation using the Taylor series of f around x

$$
\begin{aligned}
f(x+h) &= \sum_{n=0}^{\infty} h^n \frac{f^{(n)}(x)}{n!} \\
&= f(x) + hf'(x) + h^2 \frac{f''(x)}{2!} + h^3 \frac{f'''(x)}{3!} + \dots
\end{aligned}
$$

- Rearranging for $f'(x)$

$$
hf'(x) = f(x+h) - f(x) - h^2 \frac{f''(x)}{2!} - h^3 \frac{f'''(x)}{3!} - \dots
$$

$$
\begin{aligned}
f'(x) &= \frac{1}{h}\left(f(x+h) - f(x) - h^2 \frac{f''(x)}{2!} - h^3 \frac{f'''(x)}{3!} - \dots \right) \\
&= \frac{f(x+h) - f(x)}{h} - \frac{h^2 \frac{f''(x)}{2!} - h^3 \frac{f'''(x)}{3!}}{h} - \dots \\
&= \frac{f(x+h) - f(x)}{h} - h \frac{f''(x)}{2!} - h^2 \frac{f'''(x)}{3!} - \dots
\end{aligned}
$$

Accuracy of the forward difference (2)

$$f'(x) = \frac{f(x+h) - f(x)}{h} \underbrace{- h\frac{f''(x)}{2!} - h^2\frac{f'''(x)}{3!} - \ldots}_{E_{\text{forw}}(h)}$$

$$f'(x) = \frac{f(x+h) - f(x)}{h} + E_{\text{forw}}(h)$$

- Therefore, the error term $E_{\text{forw}}(h)$ is

$$E_{\text{forw}}(h) = -h\frac{f''(x)}{2!} - h^2\frac{f'''(x)}{3!} - \ldots$$

- Can also be expressed as

$$f'(x) = \frac{f(x+h) - f(x)}{h} + \mathcal{O}(h)$$

The 1st derivative using the backward difference

- Another definition of the derivative (or "differential operator" $\frac{\mathrm{d}}{\mathrm{d}x}$)

$$\frac{\mathrm{d}f}{\mathrm{d}x}(x) = \lim_{h\to 0} \frac{f(x) - f(x-h)}{h}$$

- Use difference operator to approximate differential operator

$$\frac{\mathrm{d}f}{\mathrm{d}x}(x) = \lim_{h\to 0} \frac{f(x) - f(x-h)}{h} \approx \frac{f(x) - f(x-h)}{h}$$

- This is called the *backward difference* because we use $f(x)$ and $f(x-h)$.
- How accurate is the backward difference?

Accuracy of the backward difference

- Formal derivation using the Taylor Series of f around x

$$f(x-h) \quad = \quad f(x) - hf'(x) + h^2\frac{f''(x)}{2!} - h^3\frac{f'''(x)}{3!} + \ldots$$

- Rearranging for $f'(x)$

$$hf'(x) = f(x) - f(x-h) + h^2\frac{f''(x)}{2!} - h^3\frac{f'''(x)}{3!} - \ldots$$

$$\begin{aligned}
f'(x) &= \frac{1}{h}\left(f(x) - f(x-h) + h^2\frac{f''(x)}{2!} - h^3\frac{f'''(x)}{3!} - \ldots\right) \\
&= \frac{f(x) - f(x-h)}{h} + \frac{h^2\frac{f''(x)}{2!} - h^3\frac{f'''(x)}{3!}}{h} - \ldots \\
&= \frac{f(x) - f(x-h)}{h} + h\frac{f''(x)}{2!} - h^2\frac{f'''(x)}{3!} - \ldots
\end{aligned}$$

Accuracy of the backward difference (2)

$$f'(x) = \frac{f(x) - f(x-h)}{h} + \underbrace{h\frac{f''(x)}{2!} - h^2\frac{f'''(x)}{3!} - \dots}_{E_{\text{back}}(h)}$$

$$f'(x) = \frac{f(x) - f(x-h)}{h} + E_{\text{back}}(h) \quad (1)$$

- Therefore, the error term $E_{\text{back}}(h)$ is

$$E_{\text{back}}(h) = h\frac{f''(x)}{2!} - h^2\frac{f'''(x)}{3!} - \dots$$

- Can also be expressed as

$$f'(x) = \frac{f(x) - f(x-h)}{h} + \mathcal{O}(h)$$

The approximations are

- forward:

$$f'(x) = \frac{f(x+h) - f(x)}{h} + E_{\text{forw}}(h) \qquad (2)$$

- backward

$$f'(x) = \frac{f(x) - f(x-h)}{h} + E_{\text{back}}(h) \qquad (3)$$

$$\begin{aligned} E_{\text{forw}}(h) &= -h\frac{f''(x)}{2!} - h^2\frac{f'''(x)}{3!} - h^3\frac{f''''(x)}{4!} - h^4\frac{f'''''(x)}{5!} - \ldots \\ E_{\text{back}}(h) &= h\frac{f''(x)}{2!} - h^2\frac{f'''(x)}{3!} + h^3\frac{f''''(x)}{4!} - h^4\frac{f'''''(x)}{5!} + \ldots \end{aligned}$$

$\Rightarrow$ Add equations (2) and (3) together, then the error cancels partly!

Combining backward and forward differences (2)

Add these lines together

$$
\begin{aligned}
f'(x) &= \frac{f(x+h)-f(x)}{h} + E_{\text{forw}}(h) \\
f'(x) &= \frac{f(x)-f(x-h)}{h} + E_{\text{back}}(h)
\end{aligned}
$$

$$
2f'(x) = \frac{f(x+h)-f(x-h)}{h} + E_{\text{forw}}(h) + E_{\text{back}}(h)
$$

Adding the error terms:

$$
E_{\text{forw}}(h) + E_{\text{back}}(h) = -2h^2\frac{f'''(x)}{3!} - 2h^4\frac{f'''''(x)}{5!} - \ldots
$$

The combined (central) difference operator is

$$
f'(x) = \frac{f(x+h)-f(x-h)}{2h} + E_{\text{cent}}(h)
$$

with

$$
E_{\text{cent}}(h) = -h^2\frac{f'''(x)}{3!} - h^4\frac{f'''''(x)}{5!} - \ldots
$$

Central difference

- Can be derived (as on previous slides) by adding forward and backward difference
- Can also be interpreted geometrically by defining the differential operator as

$$\frac{\mathrm{d}f}{\mathrm{d}x}(x) = \lim_{h\to 0} \frac{f(x+h) - f(x-h)}{2h}$$

and taking the finite difference form

$$\frac{\mathrm{d}f}{\mathrm{d}x}(x) \approx \frac{f(x+h) - f(x-h)}{2h}$$

- Error of the central difference is only $\mathcal{O}(h^2)$, *i.e.* better than forward or backward difference

It is generally the case that symmetric differences are more accurate than asymmetric expressions.

Example (1)

Using forward difference to estimate the derivative of $f(x) = \exp(x)$

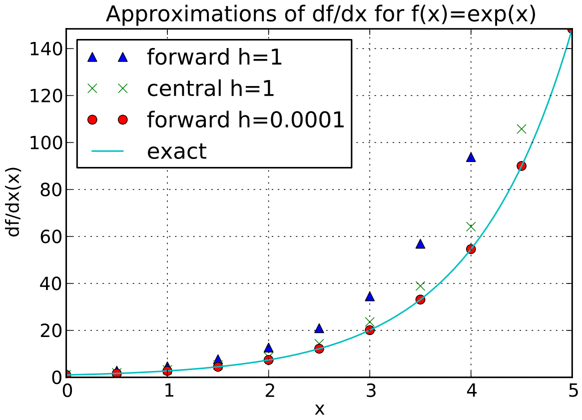
$$f'(x) \approx f'_{\text{forw}} = \frac{f(x+h) - f(x)}{h} = \frac{\exp(x+h) - \exp(x)}{h}$$

Numerical example:

- $h = 0.1$, $x = 1$
- $f'(1) \approx f'_{\text{forw}}(1.0) = \frac{\exp(1.1) - \exp(1)}{0.1} = 2.8588$
- Exact answers is $f'(1.0) = \exp(1) = 2.71828$
- (Central diff: $f'_{\text{cent}}(1.0) = \frac{\exp(1+0.1) - \exp(1-0.1)}{0.2} = 2.72281$)

Example (2)

Comparison: forward difference, central difference and exact derivative of $f(x) = \exp(x)$



Summary

- Can approximate derivatives of f numerically using only function evaluations of f
- size of step h very important
- central differences has smallest error term

name	formula	error
forward	$f'(x) = \frac{f(x+h)-f(x)}{h}$	$\mathcal{O}(h)$
backward	$f'(x) = \frac{f(x)-f(x-h)}{h}$	$\mathcal{O}(h)$
central	$f'(x) = \frac{f(x+h)-f(x-h)}{2h}$	$\mathcal{O}(h^2)$

Appendix: source to compute figure on page 19 I

```
EPS=1 #very large EPS to provoke inaccuracy

def forwarddiff(f,x,h=EPS):
    # df/dx = ( f(x+h)-f(x) )/h + O(h)
    return ( f(x+h)-f(x) )/h

def backwarddiff(f,x,h=EPS):
    # df/dx = ( f(x)-f(x-h) )/h + O(h)
    return ( f(x)-f(x-h) )/h

def centraldiff(f,x,h=EPS):
    # df/dx = (f(x+h) - f(x-h))/h + O(h^2)
    return (f(x+h) - f(x-h))/(2*h)

if __name__ == "__main__":
    #create example plot
    import pylab
    import numpy as np
    a=0        #left and
    b=5        #right limits for x
    N=11       #steps
```

```
def f(x):
    """Our test funtion with
    convenient property that
    df/dx = f"""
    return np.exp(x)

xs=np.linspace(a,b,N)
forward = []
forward_small_h = []
central = []
for x in xs:
    forward.append( forwarddiff(f,x) )
    central.append( centraldiff(f,x) )
    forward_small_h.append(
        forwarddiff(f,x,h=1e-4))

pylab.figure(figsize=(6,4))
pylab.axis([a,b,0,np.exp(b)])
pylab.plot(xs,forward,'^',label='forward h=%g'%EPS)
pylab.plot(xs,central,'x',label='central h=%g'%EPS)
pylab.plot(xs,forward_small_h,'o',
           label='forward h=%g'% 1e-4)
xsfine = np.linspace(a,b,N*100)
```

```
    pylab.plot(xsfine,f(xsfine),'-',label='exact')
    pylab.grid()
    pylab.legend(loc='upper left')
    pylab.xlabel("x")
    pylab.ylabel("df/dx(x)")
    pylab.title("Approximations of df/dx for f(x)=exp(x)")
    pylab.plot()
    pylab.savefig('central-and-forward-difference.pdf')
    pylab.show()
```

Note: Euler's (integration) method — derivation using finite difference operator

- Use forward difference operator to approximate differential operator

$$\frac{\mathrm{d}y}{\mathrm{d}x}(x) = \lim_{h\to 0} \frac{y(x+h)-y(x)}{h} \approx \frac{y(x+h)-y(x)}{h}$$

- Change differential to difference operator in $\frac{\mathrm{d}y}{\mathrm{d}x} = f(x,y)$

$$\begin{aligned} f(x,y) = \frac{\mathrm{d}y}{\mathrm{d}x} &\approx \frac{y(x+h)-y(x)}{h} \\ hf(x,y) &\approx y(x+h)-y(x) \\ \Longrightarrow y_{i+1} &= y_i + hf(x_i,y_i) \end{aligned}$$

- $\Rightarrow$ Euler's method (for ODEs) can be derived from the forward difference operator.

Note: Newton's (root finding) method — derivation from Taylor series

- We are looking for a root, *i.e.* we are looking for a x so that $f(x) = 0$.
- We have an initial guess x_0 which we refine in subsequent iterations:

$$x_{i+1} = x_i - h_i \quad \text{where} \quad h_i = \frac{f(x_i)}{f'(x_i)}. \tag{4}$$

.

- This equation can be derived from the Taylor series of f around x. Suppose we guess the root to be at x and $x + h$ is the actual location of the root (so h is unknown and $f(x + h) = 0$):

$$\begin{aligned} f(x+h) &= f(x) + hf'(x) + \dots \\ 0 &= f(x) + hf'(x) + \dots \\ \Longrightarrow 0 &\approx f(x) + hf'(x) \\ \Longleftrightarrow h &\approx -\frac{f(x)}{f'(x)}. \end{aligned} \tag{5}$$

The diffusion equation

Diffusion equation

- The 2d operator $\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$ is called the Laplace operator Δ, so that we can also write

$$\frac{\partial u}{\partial t} = D\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right) = D\Delta u$$

- The diffusion equation (with constant diffusion coefficient D) reads $\frac{\partial u}{\partial t} = D\Delta u$ where the Laplace operator depends on the number d of spatial dimensions
  - $d = 1$: $\Delta = \frac{\partial^2}{\partial x^2}$
  - $d = 2$: $\Delta = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$
  - $d = 3$: $\Delta = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2}$

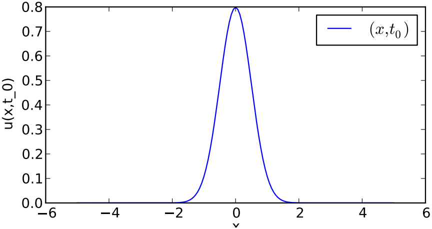
1d Diffusion equation $\frac{\partial u}{\partial t} = D\frac{\partial^2 u}{\partial x^2}$

- In one spatial dimension, the diffusion equation reads

$$\frac{\partial u}{\partial t} = D\frac{\partial^2 u}{\partial x^2}$$

  This is the equation we will use as an example.

- Let's assume an initial concentration $u(x, t_0) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-x_{\mathrm{mean}})^2}{\sigma^2}\right)$ with $x_{\mathrm{mean}} = 0$ and width $\sigma = 0.5$.



1d Diffusion eqn $\frac{\partial u}{\partial t} = D\frac{\partial^2 u}{\partial x^2}$, time integration I

- Let us assume that we have some way of computing $D\frac{\partial^2 u}{\partial x^2}$ at time t_0 and let's call this $g(x, t_0)$, i.e.

$$g(x, t_0) \equiv D\frac{\partial^2 u(x, t_0)}{\partial x^2}$$

- We like to solve

$$\frac{\partial u(x, t)}{\partial t} = g(x, t_0)$$

  to compute $u(x, t_1)$ at some later time t_1.
- Use finite difference time integration scheme:
- Introduce a time step size h so that $t_1 = t_0 + h$.

1d Diffusion eqn $\frac{\partial u}{\partial t} = D\frac{\partial^2 u}{\partial x^2}$, time integration II

- Change differential operator to forward difference operator

$$g(x,t_0) = \frac{\partial u(x,t)}{\partial t} = \lim_{h\to 0} \frac{u(x,t_0+h) - u(x,t_0)}{h} \quad (6)$$

$$\approx \frac{u(x,t_0+h) - u(x,t_0)}{h} \quad (7)$$

- Rearrange to find $u(x,t_1) \equiv u(x,t_0+h)$ gives

$$u(x,t_1) \approx u(x,t_0) + hg(x,t_0)$$

- We can generalise this using $t_i = t_0 + ih$ to read

$$u(x,t_{i+1}) \approx u(x,t_i) + hg(x,t_i) \quad (8)$$

$\rightarrow$ If we can find $g(x,t_i)$, we can compute $u(x,t_{i+1})$

1d Diffusion eqn $\frac{\partial u}{\partial t} = D\frac{\partial^2 u}{\partial x^2}$, spatial part I

$$\frac{\partial u}{\partial t} = D\frac{\partial^2 u}{\partial x^2} = g(x,t)$$

- Need to compute $g(x,t) = D\frac{\partial^2 u(x,t)}{\partial x^2}$ for a given $u(x,t)$.
- Can ignore the time dependence here, and obtain

$$g(x) = D\frac{\partial^2 u(x)}{\partial x^2}.$$

- Recall that

$$\frac{\partial^2 u}{\partial x^2} = \frac{\partial}{\partial x}\frac{\partial u}{\partial x}$$

  and we that know how to compute $\frac{\partial u}{\partial x}$ using central differences.

Second order derivatives from finite differences I

- Recall central difference equation for first order derivative

$$\frac{\mathrm{d}f}{\mathrm{d}x}(x) \approx \frac{f(x+h) - f(x-h)}{2h}$$

- will be more convenient to replace h by $\frac{1}{2}h$:

$$\frac{\mathrm{d}f}{\mathrm{d}x}(x) \approx \frac{f(x+\frac{1}{2}h) - f(x-\frac{1}{2}h)}{h}$$

Second order derivatives from finite differences II

- Apply the central difference equation twice to obtain $\frac{\mathrm{d}^2 f}{\mathrm{d}x^2}$:

$$
\begin{aligned}
\frac{\mathrm{d}^2 f}{\mathrm{d}x^2}(x) &= \frac{\mathrm{d}}{\mathrm{d}x}\frac{\mathrm{d}f}{\mathrm{d}x}(x) \\
&\approx \frac{\mathrm{d}}{\mathrm{d}x}\left(\frac{f(x+\frac{1}{2}h) - f(x-\frac{1}{2}h)}{h}\right) \\
&= \frac{1}{h}\left(\frac{\mathrm{d}}{\mathrm{d}x}f\left(x+\frac{1}{2}h\right) - \frac{\mathrm{d}}{\mathrm{d}x}f\left(x-\frac{1}{2}h\right)\right) \\
&\approx \frac{1}{h}\left(\frac{f(x+h)-f(x)}{h} - \frac{f(x)-f(x-h)}{h}\right) \\
&= \frac{f(x+h)-2f(x)+f(x-h)}{h^2} \qquad (9)
\end{aligned}
$$

Recipe to solve $\frac{\partial u}{\partial t} = D\frac{\partial^2 u}{\partial x^2}$

1. Discretise solution $u(x,t)$ into discrete values
2. $u_j^i \equiv u(x_j, t_i)$ where
   - $x_j \equiv x_0 + j\Delta x$ and
   - $t_i \equiv t_0 + i\Delta t$.
3. Start with time iteration $i = 0$
4. Need to know configuration $u(x, t_i)$.
5. Then compute $g(x, t_i) = D\frac{\partial^2 u}{\partial x^2}$ using finite differences (9).
6. Then compute $u(x, t_{i+1})$ based on $g(x, t_i)$ using (8)
7. increase i to $i+1$, then go back to 5.

A sample solution $\frac{\partial u}{\partial t} = D\frac{\partial^2 u}{\partial x^2}$, I

```python
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.animation as animation

a,b=-5,5                # size of box
N = 51                  # number of subdivisions
x=np.linspace(a,b,N) #positions of subdivisions
h=x[1]-x[0]             #discretisation stepsize in x-direction

def total(u):
    """Computes total number of moles in u."""
    return ((b-a)/float(N)*np.sum(u))

def gaussdistr(mean,sigma,x):
    """Return gauss distribution for given numpy array x"""
    return 1./(sigma*np.sqrt(2*np.pi))*np.exp(
        -0.5*(x-mean)**2/sigma**2)

#starting configuration for u(x,t0)
u = gaussdistr(mean=0.,sigma=0.5,x=x)
```

A sample solution $\frac{\partial u}{\partial t} = D\frac{\partial^2 u}{\partial x^2}$, II

```
def compute_g( u, D, h ):
    """given a u(x,t) in array, compute g(x,t)=D*d^2u/dx^2
    using central differences with spacing h,
    and return g(x,t). """
    d2u_dx2 = np.zeros(u.shape,np.float)
    for i in range(1,len(u)-1):
        d2u_dx2[i] = (u[i+1] - 2*u[i]+u[i-1])/h**2
    #special cases at boundary: assume Neuman boundary
    #conditions, i.e. no change of u over boundary
    #so that u[0]-u[-1]=0 and thus u[-1]=u[0]
    i=0
    d2u_dx2[i] = (u[i+1] - 2*u[i]+u[i])/h**2
    #same at other end so that u[N-1]-u[N]=0
    #and thus u[N]=u[N-1]
    i=len(u)-1
    d2u_dx2[i] = (u[i] - 2*u[i]+u[i-1])/h**2
    return D*d2u_dx2

def advance_time( u, g, dt):
    """Given the array u, the rate of change array g,
    and a timestep dt, compute the solution for u
    after t, using simple Euler method."""
    u = u +dt*g
```

A sample solution $\frac{\partial u}{\partial t} = D\frac{\partial^2 u}{\partial x^2}$, III

```
    return u

#show example, quick and dirtly, lots of global variables
dt = 0.01                         #step size or time
stepsbeforeupdatinggraph = 20 #plotting is slow
D = 1.                            #Diffusion coefficient
stepsdone = 0                     #keep track of iterations

def do_steps(j,nsteps=stepsbeforeupdatinggraph):
    """Function called by FuncAnimation class. Computes
    nsteps iterations, i.e. carries forward solution from
    u(x,t_i) to u(x,t_{i+nsteps}).
    """
    global u,stepsdone
    for i in range(nsteps):
        g = compute_g( u, D, h)
        u = advance_time( u, g, dt)
        stepsdone += 1
        time_passed = stepsdone * dt
    print("stepsdone=%5d, time=%8gs, total(u)=%8g" %
          (stepsdone,time_passed,total(u)))
    l.set_ydata(u)     # update data in plot
    fig1.canvas.draw()# redraw the canvas
```

A sample solution $\frac{\partial u}{\partial t} = D\frac{\partial^2 u}{\partial x^2}$, IV

```
    return l,

fig1 = plt.figure()         #setup animation
l,= plt.plot(x,u,'b-o')     #plot initial u(x,t)
                            #then compute solution and animate
line_ani = animation.FuncAnimation(fig1,
                                   do_steps, range(10000))
plt.show()
```

Boundary conditions I

- For ordinary differential equations (ODEs), we need to know the *initial value(s)* to be able to compute a solution.
- For partial differential equations (PDEs), we need to know the initial values and extra information about the behaviour of the solution $u(x, t)$ at the boundary of the spatial domain (i.e. at $x = a$ and $x = b$ in this example).
- Commonly used boundary conditions are
  - Dirichlet boundary conditions: fix $u(a) = c$ to some constant.
    Would correspond here to some mechanism that keeps the concentration u at position $x = a$ constant.

Boundary conditions II

- Neuman boundary conditions: fix the change of u across the boundary, i.e.

$$\frac{\partial u}{\partial x}(a) = c.$$

  - For positive/negative c this corresponds to an imposed concentration gradient.
  - For $c = 0$, this corresponds to conservation of the atoms in the solution: as the gradient across the boundary cannot change, no atoms can move out of the box. (Used in our program on slide 35)

Numerical issues

- The time integration scheme we use is *explicit* because we have an explicit equation that tells us how to compute $u(x, t_{i+1})$ based on $u(x, t_i)$ (equation (8) on slide 30)
- An implicit scheme would compute $u(x, t_{i+1})$ based on $u(x, t_i)$ *and on* $u(x, t_{i+1})$.
- The implicit scheme is more complicated as it requires solving an additional equation system just to find $u(x, t_{i+1})$ but allows larger step sizes Δt for the time.
- The explicit integration scheme becomes quickly unstable if Δt is too large. Δt depends on the chose spatial discretisation Δx.

Performance issues

- Our sample code is (nearly) as slow as possible
  - interpreted language
  - explicit for loops
  - enforced small step size from explicit scheme
- Solutions:
  - Refactor for-loops into matrix operations and use (compiled) matrix library (`numpy` for small systems, use `scipy.sparse` for larger systems)
  - Use library function to carry out time integration (will use implicit method if required), for example `scipy.integrate.odeint`.

Finite Elements

Another widely spread way of solving PDEs is using so-called finite elements.

- Mathematically, the solution $u(x)$ for a problem like $\frac{\partial^2 u}{\partial x^2} = f(x)$ is written as

$$u(x) = \sum_{i=1}^{N} u_i \phi_i(x) \tag{10}$$

  where each u_i is a number (a coefficient), and each $\phi_i(x)$ a known function of space.
- The ϕ_i are called *basis* or *shape functions*.
- Each ϕ_i is normally chosen to be zero for nearly all x, and to be non-zero close to a particular node in the finite element mesh.
- By substitution (10) into the PDE, a matrix system can be obtained, which – if solved – provides the coefficients u_i, and thus the solution.

Finite Elements vs Finite differences

- Finite differences
  - are mathematically much simpler and
  - for simple geometries (such as cuboids) easier to program
- Finite elements
  - have greater flexibility in the shape of the domain,
  - the specification and implementation of boundary conditions is easier
  - but the basic mathematics and code is more complicated.

Practical observation on time integration

- Usually, we solve the *spatial* part of a PDE using some discretisation scheme such as finite differences and finite elements).
- This results in a set of coupled ordinary differential equations (where time is the independent variable). Can think of this as one ODE for every cube from our discretisation.
- This *temporal* part is then solved using time integration schemes for (systems of) ordinary differential equations.

Summary

- Partial differential equations important in many contexts
- If no analytical solution known, use numerics.
- Discretise the problem through
    - finite differences (replace differential with difference operator, corresponds to chopping space and time in little cuboids)
    - finite elements (project solution on localised basis functions, often used with tetrahedral meshes)
    - related methods (finite volumes, meshless methods).

Finite elements and finite difference calculations are standard tools in many areas of engineering, physics, chemistry, but increasingly in other fields.

```
changeset:   53:a22b7f13329e
tag:         tip
user:        Hans Fangohr [MBP13] <fangohr@soton.ac.uk>
date:        Fri Dec 16 10:57:15 2011 +0000
```