CSC 7700: Scientific Computing

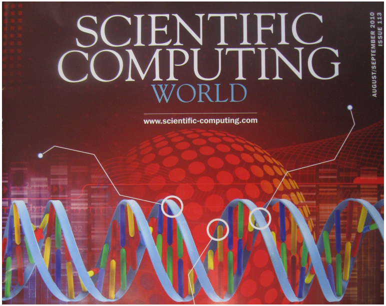
Module A: Basic Skills

Lecture 2: Introduction to numerical methods

Dr Frank Löffler

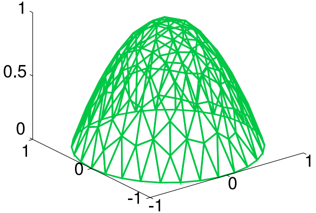
September 7 2010

Introduction



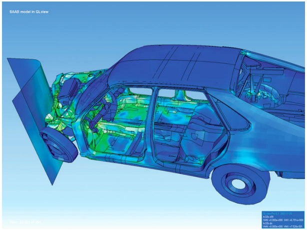
Numerical methods

- Problems very often expressed as *equations*
- Often those equations are not solvable analytically
- Equations often continuous $\longrightarrow$ infinite degrees of freedom
- Computer memory and computing power limited
- Equations have to be approximated to be solved $\longrightarrow$ Discretization



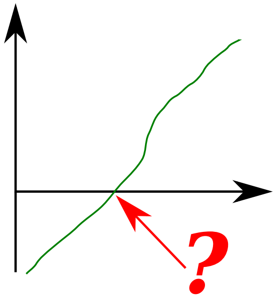
Discretization

- Different methods for discretization exist, but all have in common:
  - Specify finite number of points covering the region of interest
  - Assign function values to those points (grid points)
  - Prescribe how to obtain function values and derivatives at and between grid points (approximation)
- Choice of discretization is problem dependent
- Given discretization, a number of common problems/tasks remain



Outline

Root finding



Root finding

Task:

- Given (usually complicated) function $f = f(x)$
- Find x such that $f(x) = 0$
- No analytical solution can be found

Numerical, iterative solution:

- Choose initial guess
- Improve guess to obtain new guess
- Iterate until consecutive guesses don't change by much anymore

Root finding

Important: to know something about the function first:

- Which root are you looking for?
- Number and approximate position of roots
- Pathological behaviour
- $f'(x) = \frac{\partial f(x)}{\partial x}$ known?

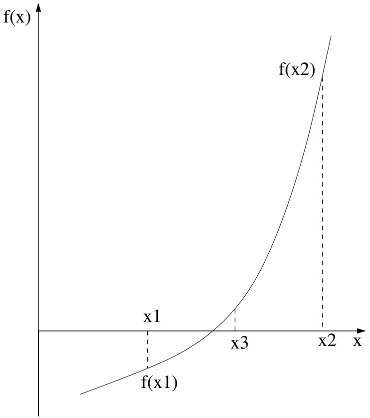
Good idea: make a plot before starting and try to make the initial guess as good as possible.

We will consider four methods here:

- Bisection method
- Regula falsi method
- Newton Rathson method
- Secant method

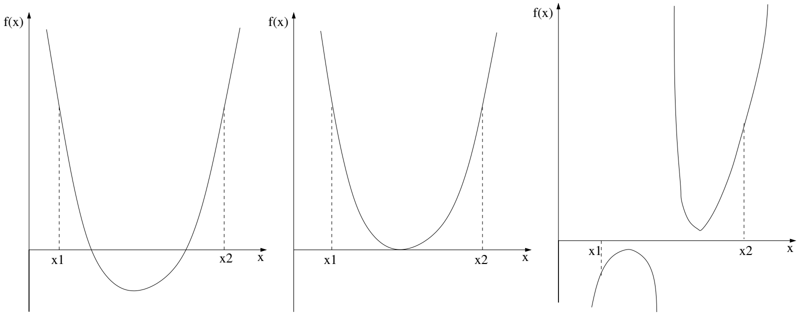
Root finding - Bisection

- Bracket the root: find x_1 and x_2 with $x_1 < x < x_2$, such that $f(x_1) < 0$ and $f(x_2) > 0$ or vice versa.
- Halve (bisect) the interval between x_1 and x_2: $x_3 = \frac{x_1+x_2}{2}$
- Find the new bracket for the root (either $[x_1, x_3]$ or $[x_3, x_2]$)
- Repeat until the required accuracy is reached (beware of the machine accuracy)



Root finding - Bisection

However, this can go wrong:



Merits of this method:

- Always works if properly backeted
- Does not need special knowledge of function, like $f'(x)$
- Automatically keeps the root bracketed

Drawback:

- not very fast

Root finding - Bisection

If ϵ_n is the tolerance at the n-th step:

$$\epsilon_{n+1} = \frac{\epsilon_n}{2}$$

Definition: Order of convergence m:

$$\epsilon_{n+1} = (\text{constant} < 1) \times (\epsilon_n)^m$$

$\rightarrow$ Bisection method gives first order (linear) of convergence:

$$\epsilon_n = \frac{1}{2}(\epsilon_{n-1})^1 = 2^{-n}\epsilon_0$$

Root finding - Regula falsi

Works like bisection, but puts new x at intersection of line connecting $(x_1, f(x_1))$ and $(x_2, f(x_2))$ with x axis.

$$\frac{x_3 - x_1}{-f(x_1)} = \frac{x_2 - x_3}{f(x_2)}$$

$$x_3 = x_2 - \frac{(x_1 - x_2)f(x_2)}{f(x_1) - f(x_2)}$$

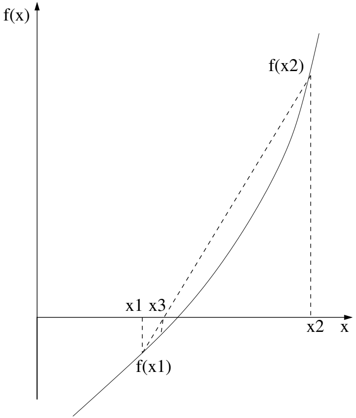
For the next step: again find out which interval brackets the root.

Merits

- Can be faster than bisection

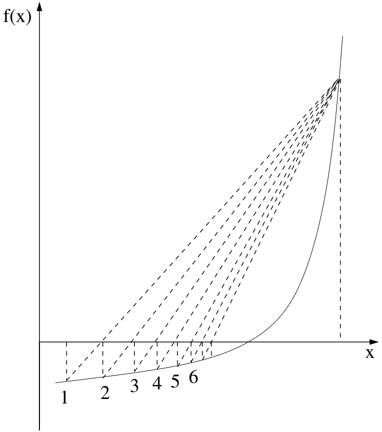
Drawbacks

- Can sometimes be very slow



Root finding - Regula falsi

Unfavourable case for Regula falsi:



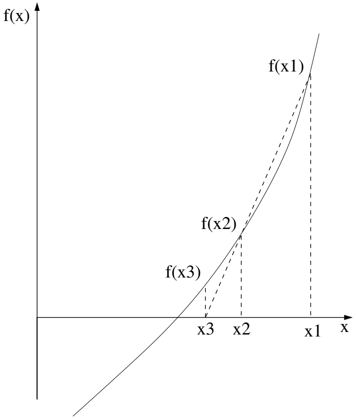
Convergence: $1 <= m <= 1.618$

Root finding - Secant method

- Uses last two x-values to compute derivative

$$x_3 = x_2 - \frac{f(x_2)}{\left[\frac{f(x_1)-f(x_2)}{x_1-x_2}\right]}$$

- Converges more quickly: $m \sim 1.618$, whereas $1 <= m <= 1.618$ for regular falsi
- However: Regular falsi keeps root bracketed, secant method does not



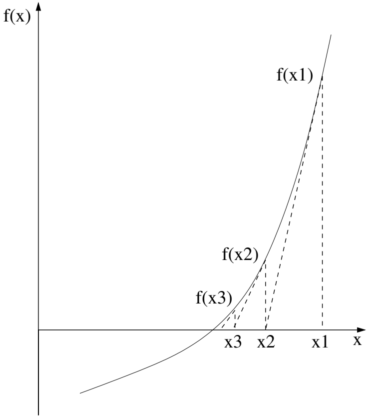
Root finding - Newton-Raphson

Knowledge about $f'(x)$ required.
Root is not kept bracketed.

$$x_2 = x_1 - \frac{f(x_1)}{f'(x_1)}$$

Merits

- Fast: second order convergence (m=2)
- Can be used to "polish the result" of other methods

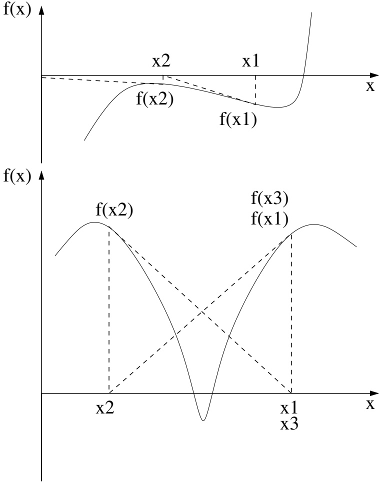


Drawbacks:

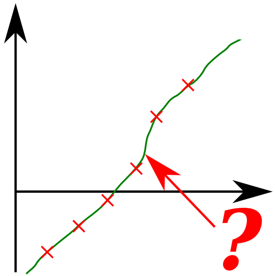
- Fast only for well-behaved function and good initial guess
- First derivative needs to be known

Root finding - Newton-Raphson

Examples for unfavourable functions for Newton-Raphson method:

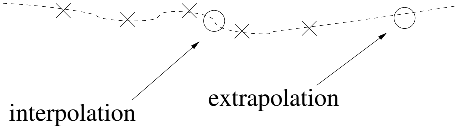


Interpolation and Extrapolation



Interpolation and Extrapolation

- Suppose values of a function a some finite number of points are known
- Task: provide function values at some other point
- Interpolation: Values known on both "sides"
- Extrapolation: Values only known on one "side"



- Extrapolation usually more "dangerous"

Interpolation and Extrapolation

Questions before starting:

- What is the nature of the given data?
  - if precise, finding an exact fit is what is required
  - if not, we might want to get a 'best fit'
- What will be done with the answer?
  - One value at a particular point - might require accuracy
  - Many values - might require continuity of derivatives

Conceptionally, there are two stages:

- Find a function which approximates the given points
- Evaluate this function at the required point(s)

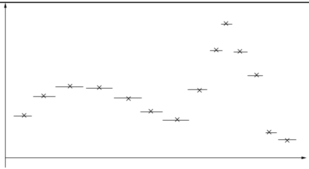
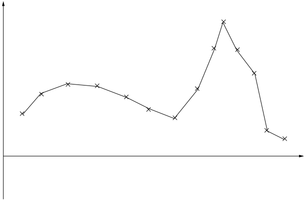
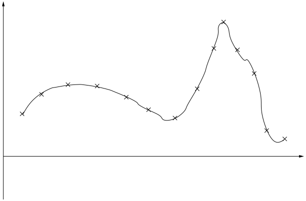
However, it is usually more efficient to combine these in one step.

Interpolation and Extrapolation

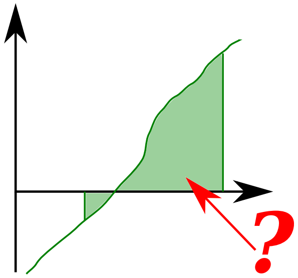
Guidelines:

- Try to use points 'close' and centered on both sides of requested coordinate
- Using more known points usually results in higher accuracy, e.g. using polynomial functions of different orders
- Higher order schemes are dangerous because they tend to introduce oscillations.
- Any sort of interpolation or extrapolation assumes that the function is reasonably well behaved.

Interpolation and Extrapolation

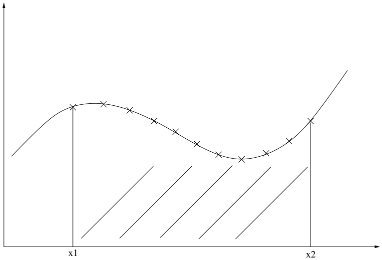
order	example
0 (constant)	
1 (linear)	
>= 2	

Integration



Integration

- Task: Find integral of given function within interval
- Function may be only given at finite number of points
- In other words: find area “below” function in given interval



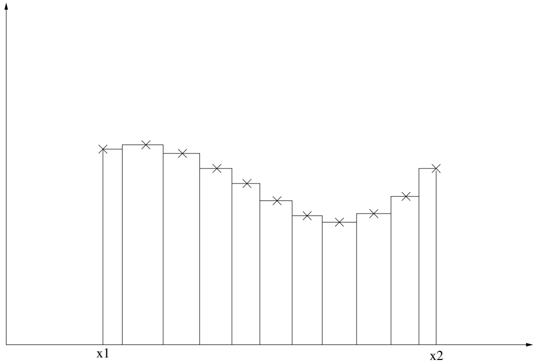
General procedure:

- Approximate (pointwise) function by analytic function (see interpolation)
- Integrate approximation traditionally

Integration

Approximation by constant values:

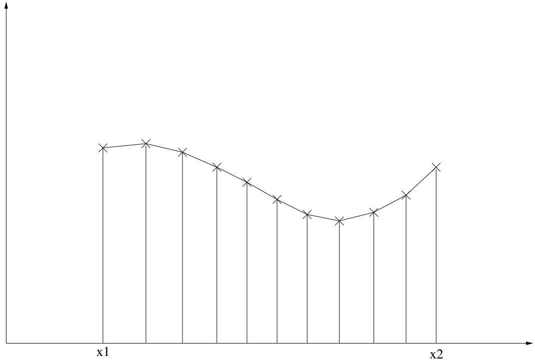
$$A = \sum_{x_1}^{x_2} A_{\text{rectangles}}$$



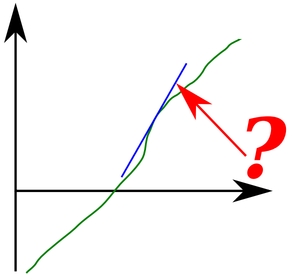
Integration

Approximation by linear functions:

$$A = \sum_{x_1}^{x_2} A_{\text{trapezoids}}$$



Differentiation

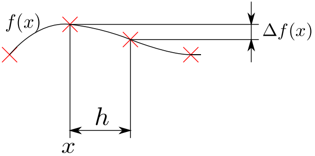


Finite Differences

A finite difference is a mathematical expression of the form

$$\Delta f(x) = f(x+h) - f(x)$$

- x: position of grid point
- $h \equiv \Delta x$: distance from x to next grid point
- x: coordinate of point typically between $x+h$ and x



If this is divided by h, one gets the difference quotient:

$$\frac{\Delta f(x)}{h} = \frac{f(x+h) - f(x)}{h} = \frac{\Delta f(x+h)}{h}$$

Finite Differences

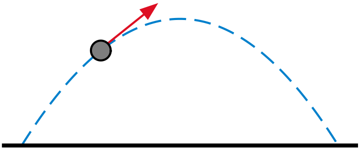
The limit: the derivative of f:

$$\lim_{h\to 0}\left(\frac{\Delta f(x)}{h}\right) = \frac{\mathrm{d}f(x)}{\mathrm{d}x} = f'(x)$$

Different orders of approximations:

order	approximation
1	$\mathrm{D}_{+}\left(f(x)\right) = \frac{f(x+h)-f(x)}{h}$
1	$\mathrm{D}_{-}\left(f(x)\right) = \frac{f(x)-f(x-h)}{h}$
2	$\mathrm{D}_{\pm}\left(f(x)\right) = \frac{f(x+h)-f(x-h)}{2h}$

Ordinary differential equations



Differential equations

Differential equation:

- A mathematical equation for an unknown function of one or several variables that relates the values of the function itself and its derivatives
- Arise in many areas of science and technology
- Examples:
  - Determination of the position and velocity of a ball falling through the air, considering only gravity and air resistance
  - Determination of orbits of planets

Differential equations

- Two classes:
  - Ordinary differential equations (ODEs): unknown function (also known as the dependent variable) only a function of a *single* independent variable
  - Partially differential equations (PDEs): unknown function is a function of *multiple* independent variables and the equation involves its partial derivatives

Numerical differential equations:

- Sometimes also called *numerical integration*
- Used if differential equations cannot be solved analytically

Numerical ordinary differential equations

Problem:

$$y'(t) = f(t, y(t)),\ y(t_0) = y_0$$

Examples of different solution methods:

- Euler
- Crank-Nicholson
- Runge-Kutta

Euler method

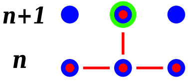
$$y'(t) = f(t, y(t)),\ y(t_0) = y_0$$

$$\downarrow$$

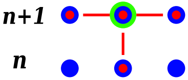
$$\text{forward Euler: } y_{n+1} = y_n + hf(t_n, y_n)$$

$$\text{backward Euler: } y_{n+1} = y_n + hf(t_{n+1}, y_{n+1})$$

- Forward Euler:
  - Explicit method
  - Typically requires small time steps



- Backward Euler:
  - Implicit method (requires to solve equation)
  - Allows bigger time steps



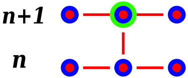
- Error relates to h as $\frac{1}{h}$: first order
- Can be unstable
- Thus: simple, but not very often used in practice

Crank–Nicolson method

$$y'(t) = f(t, y(t)),\ y(t_0) = y_0$$

$$\downarrow$$

$$y_{n+1} = y_n + \frac{1}{2}h\left(f(t_n, y_n) + y_n + f(t_{n+1}, y_{n+1})\right)$$



- Average of forward and backward Euler methods
- *Not* average of solutions of forward and backward Euler methods
- Implicit method
- Error relates to h as $\frac{1}{h^2}$: second order

Runge-Kutta method(s)

$$y'(t) = f(t, y(t)),\ y(t_0) = y_0$$

$$\downarrow$$

$$\begin{aligned}
y_{n+1} &= y_n + \frac{1}{6}h\left(k_1 + 2k + 2 + 2k_3 + k_4\right) \\
k_1 &= f(t_n, y_n) \\
k_2 &= f(t_n + \frac{1}{2}h, y_n + \frac{1}{2}hk_1) \\
k_3 &= f(t_n + \frac{1}{2}h, y_n + \frac{1}{2}hk_2) \\
k_4 &= f(t_n + h, y_n + hk_3)
\end{aligned}$$

- Explicit method
- Error relates to h as $\frac{1}{h^4}$: forth order

Partial differential equations

Partial differential equations

- Partially differential equations (PDEs): unknown function is a function of *multiple* independent variables and the equation involves its partial derivatives
- Used to formulate, and thus aid the solution of, problems involving functions of several variables; such as
  - the propagation of sound or heat
  - electrostatics
  - electrodynamics
  - fluid flow
  - elasticity
- In general no unique solution, additional conditions needed, e.g. boundary conditions
- Example: $\frac{\partial}{\partial x} u(x, y) = u_x = 0$

Partial differential equations

Examples:

Mathematical formula	describes
$u_t = \alpha u_{xx}$	heat equation
$u_{tt} = c^2 u_{xx}$	wave equation
$\Psi_t + (u\Psi)_x + (v\Psi)_y + (w\Psi)_z = 0$	advection equation
$i\hbar\frac{\partial}{\partial t}\Psi = \hat{H}\Psi$	Schrödinger equation
$G_{\mu\nu} + \Lambda g_{\mu\nu} = \frac{8\pi G}{c^4} T_{\mu\nu}$	Einstein field equations

Numerical methods to solve PDEs:

- Finite element methods
- Finite volume methods
- Finite difference methods

Partial differential equations

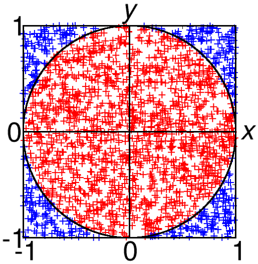
Classification of *some* linear, second-order partial differential equations:

$$Au_{xx} + Bu_{xy} + Cu_{yy} + \text{(lower order terms)} = 0$$

- $B^2 - 4AC = 0$: parabolic
  Examples: heat diffusion, stock option pricing
- $B^2 - 4AC > 0$: hyperbolic
  Example: wave equation
- $B^2 - 4AC < 0$:elliptic
  Example: Electrostatic potential solutions

Parabolic and hyperbolic problems are 'evolution problems', while elliptic problems are 'static' problems.

Random Numbers and Monte-Carlo Techniques



Monte Carlo methods

- Some problems cannot be described (well, efficiently) by grid-based approaches
- Alternative approach: Monte Carlo method
- Relies on repeated random sampling
- Especially useful in studying systems with a large number of coupled degrees of freedom, and/or phenomena with significant uncertainty in inputs
  - fluids
  - disordered materials
  - strongly coupled solids
  - cellular structures
  - calculation of risk in business

Monte Carlo methods

No single Monte Carlo method, but similar pattern:

1. Define domain of possible inputs
2. Generate inputs randomly from the domain using a certain specified probability distribution
3. Perform deterministic computation using the inputs
4. Aggregate results of individual computations into final result

- Monte Carlo method relies on random numbers
- Computers are deterministic: no true random numbers (without special hardware)

Pseudo Random Number Generators

Pseudo Random Number Generator (PRNG): algorithm for generating a sequence of numbers that approximates the properties of random numbers.

Problems of deterministic generators:

- Sequences not truly random: completely determined by a relatively small set of initial values (state)
- Sequences necessarily periodic, sometimes much shorter than expected for some initial values
- Lack of uniformity of distribution
- Correlation of successive values

Advantages of PRNGs over true RNGs:

- Faster
- Cheaper
- Reproducable

Pseudo Random Number Generators

Some commonly used algorithms:

- Linear congruential generators
- Lagged Fibonacci generators
- Linear feedback shift registers
- Feedback with carry shift registers
- Generalised feedback shift registers
- Blum Blum Shub
- Fortuna
- Mersenne twister

Pseudo Random Number Generators

General procedure:

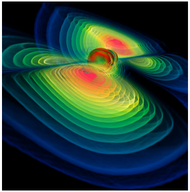
1. Choose initial PRNG state (seed)
2. Perform calculation on state
    - changing state
    - computing result of current step
3. Repeat step 2 for every (pseudo) random number

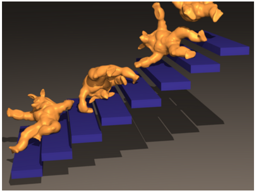
Criteria for Monte-Carlo simulations:

- Speed
- Large period
- Small correlation

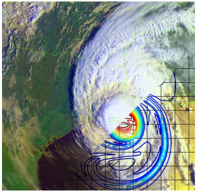
Summary

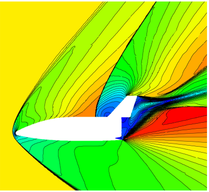
Summary





- Numerical methods important for large number of scientific fields
- Often very common problems $\rightarrow$ common solution methods
- Numerical simulations: model, discretize, solve, analyse





Coursework

Choose (at least) one of the following tasks and solve using one of the TeraGrid machines:

- Linear Interpolation
  - Consider function $y(x) = 3 + 20x - \frac{5x^2}{x-4.5}$
  - Evaluate it at N=100 points equally spaced in interval $I : x \in [0 : 10]$
  - Interpolate the value at $x = \sqrt{2}$ using linear interpolation
  - Plot error (difference of result to true value) over different N
  - Explain resulting plot
- Calculate π using Monte-Carlo method
  - Calculate PRNs in interval $(x, y) \in [-1 : 1, -1 : 1]$
  - Count points inside/outside circle with $r = 1$
  - Calculate π from ratio of points inside/outside
  - Plot result for π over number of points
  - Explain resulting plot