**Numerical methods – ODE**

*Slide 1: Overview*

How to solve with : accurately and efficiently?

*Slide 2: What is an ODE?*

Algebraic equation: , e.g.

First order ODE: , e.g.

Second order ODE: , e.g.

Very often a second order ODE can be rewritten into a system of first order ODEs (whether it is handy depends on the boundary conditions!)

Define then , thus solve system: and

Or more general second order ODE:

⇒ and where *z* is a new variable

*Slide 3: Importance of boundary conditions*

Nature of boundary conditions determines appropriate numerical method.

Classification in two main categories:

1. *Initial value problems (IVP):*   
   We know the values of all *yi* at some starting position *xs* and it is desired to find the values of *yi* at some final point *xf*.
2. *Boundary value problems (BVP):*

Boundary conditions are specified at more than one *x*. Typically, some of the BC are specified at *xs* and the remainder at *xf*.

In this course, only IVP are considered

Consider general system of ODE’s:

*i* = 1, 2, …, *n*

with

or in vector notation: with

*Slide 4: Euler’s method*

Consider single initial value problem:

with : (initial value problem)

Easiest solution algorithm: Euler’s method

Taylor series expansion:

Neglect terms with higher order than two:

Substitution: ⇒

*Slide 5: Euler’s method - solution method*

start with *t* = *t*0, *c*=*c*0, then calculate at discrete points in time:

Pseudo-code Euler’s method: and

Initialize variables, functions; Set

Set *x* = *x*0, *y* = *y*0

While *x*<*xend* do

*xi+1* = *xi* + *h*; *yi+1* = *yi* + *h*\**f*(*xi*,*yi*)

*Slide 6: Euler’s method - example*

Example: First order reaction in a Batch reactor

with *c*(*t*=0) = 1 mol/m3, *k* = 1 (s-1), *tend* = 2 (s)

Table:

|  |  |
| --- | --- |
| Time [s] | Concentration [mol/m3] |
| *t*0 = 0 | *c*0 = 1 |
|  |  |
|  |  |
|  |  |
| … | … |
|  |  |
| … | … |
|  |  |

Use Δt = 0.1 s (100 steps) -> Fig. 7.2 (p.59)

Show Fig. conv vs. N.

*Slide 7: Problems with Euler’s method:*

1. *accuracy*

Show results for *N* = 20, 40, 80, 160, 320… or for Δt = 0.1 s, 0.01 s, 0.001 s for *k* = 1 s-1

What step size/How many steps to use? ⇒ need information on numerical error!

1. *stability*

Show results for *N* = 20, 40, 80, 160, 320… or for *k* = 50 s-1

What step size/How many steps to use? ⇒ need information on stability limits!

*Slide 8: Problems with Euler’s method: accuracy*

Comparison with analytical solution: ⇒ ⇒ ζ*analytical* = 0.864665

|  |  |  |
| --- | --- | --- |
| N | ζ |  |
| 20 | 0.878423 | 0.015912 |
| 40 | 0.871488 | 0.007891 |
| 80 | 0.868062 | 0.003929 |
| 160 | 0.866360 | 0.001961 |
| 320 | 0.865511 | 0.000979 |

⇒ conclusion: error halves when the number of grid points is doubled, i.e. error is proportional to Δ*t*.

First order method.

Error estimate:

⇒ Error in the order of Δ*t*, i.e. first order method

*Slide 9: Errors and convergence rate*

Definitions of norm:

L2 norm (Euclidean norm):

L∞ norm (maximum norm):

Definitions of error:

(L2 or L∞ norm for absolute difference)

(L2 or L∞ norm for relative difference)

Convergence rate/order of convergence *r*:

i.e.

first order method reduces error by factor 2 when increasing the number of steps by a factor 2

second order method reduces error by factor 4 when increasing the number of steps by a factor 2

What to do when there is no analytical solution (easily) available?

*Slide 10: Errors and convergence rate*

Compare to calculations with different number of steps:

and and solve for *r*:

⇒

⇒ in the limit of Δ*x* → 0 or *N* → ∞.

*Slide 11: Example: Euler’s method - order of convergence*

|  |  |  |  |
| --- | --- | --- | --- |
| N | ζ |  |  |
| 20 | 0.878423 | 0.015912 | - |
| 40 | 0.871488 | 0.007891 | 1.011831775 |
| 80 | 0.868062 | 0.003929 | 1.005968855 |
| 160 | 0.866360 | 0.001961 | 1.002995636 |
| 320 | 0.865511 | 0.000979 | 1.001500387 |

⇒ Euler’s method is a first order method   
(as we already knew from the truncation error analysis)

Wouldn’t it be great to have a method that can give the answer using much less steps?

⇒ higher order methods.

*Slide 12: Runge-Kutta methods*

Propagate a solution by combining the information of several Euler-style steps (each involving one function evaluation) to match a Taylor series expansion up to some higher order.

Euler: (with ), i.e. slope =

Show picture similar to Fig. 17.1.2 Num. Rec. p. 906

Classical second order Runge-Kutta (RK2) method (= Heun’s method, or improved Euler method):

1. Approximate the slope at :
2. Approximate the slope at :   
   where we use Euler’s method to approximate *yi*+1:
3. Perform an Euler step with the average of the slopes:

Pseudo code classical RK2 (Heun’s method): and

Initialize variables, functions; Set

Set *x* = *x*0, *y* = *y*0

While *x*<*xend* do

*xi+1* = *xi* + *h*;

*Slide 13: Runge-Kutta methods – derivation*

Taylor:

(chain rule)

Substitution:

Note multivariate Taylor expansion:

⇒

Concluding:

Rewriting: , ,

*Slide 14: Runge-Kutta methods – derivation*Generalization: *,*

Note that classical RK2: and

Bivariate Taylor expansion:

Comparison with Taylor:

, , ⇒ 3 eqns & 4 unknowns ⇒ multiple possibilities

Alternative second order Runge-Kutta (RK2) method (= midpoint rule, or modified Euler method):

*, ,*

*Slide 15: Second order Runge-Kutta methods:*Show picture demonstrating the differences

|  |  |
| --- | --- |
| **Classical RK2**  ( = Heun’s method, improved Euler method) | **(Explicit) midpoint rule**  (modified Euler method) |
|  |  |

*Slide 16: Second order Runge-Kutta method - Example*

Example: First order reaction in a Batch reactor

with *c*(*t*=0) = 1 mol/m3, *k* = 1 (s-1), *tend* = 2 (s)

Show figure with comparison Euler/midpoint rule vs. analytical:

|  |  |  |  |
| --- | --- | --- | --- |
| *t* | *c* |  |  |
| 0 | 1 | 0.1·(–1·1) = –0.1 | 0.1·(–1·(1 – 0.5·0.1)) = –0.095 |
| 0.1 | 1 – 0.095 = 0.905 | 0.1·(–1·0.905) = –0.0905 | 0.1·(–1·(0.905 – 0.5·0.0905)) = -0.085975 |
| … | … | … | … |
| 2 | 0.1358225 | –0.0135822 | –0.0129031 |

*Slide 17: RK2 method - order of convergence*

|  |  |  |  |
| --- | --- | --- | --- |
| N | ζ |  |  |
| 20 | 0.864177542 | 5.634·10-4 | - |
| 40 | 0.864547573 | 1.355·10-4 | 2.056 |
| 80 | 0.864635985 | 3.323·10-5 | 2.028 |
| 160 | 0.864657601 | 8.229·10-6 | 2.014 |
| 320 | 0.864662946 | 2.048·10-6 | 2.007 |

⇒ Euler’s method is a second order method:   
Doubling the number of cells, reduces the error by a factor 4!

Can we do even better?

*Slide 18: RK4 method (classical fourth order Runge-Kutta method)*

Show Fig. 17.1.3 from Num. Rec. p 909

*Slide 19: RK4 method – order of convergence*

|  |  |  |  |
| --- | --- | --- | --- |
| N | ζ |  |  |
| 20 | 0.864664472 | 2.836·10-7 | - |
| 40 | 0.864664702 | 1.700·10-8 | 4.060 |
| 80 | 0.864664716 | 1.040·10-9 | 4.030 |
| 160 | 0.864664717 | 6.435·10-11 | 4.015 |
| 320 | 0.864664717 | 4.001·10-12 | 4.007 |

⇒ RK4 is a fourth order method: Doubling the number of cells, reduces the error by a factor 16!

*Slide 20: Adaptive step size control:*

Globally two different approaches can be used:

1. **Step doubling:** compare solutions when taking one full step or two consecutive halve steps

Picture as Fig. 17.2.1 Num Rec. p.912

RK4 with one large step of *h*:

RK4 with two steps of

Estimation of the truncation error by comparing and :

If Δ too large ⇒ reduce step size for accuracy, if Δ too small ⇒ increase step size for efficiency

Ignoring higher order terms and solving for *C*:

⇒ ⇒ ⇒ (local Richardson extrapolation)

Note that we specify a tolerance *tol* we can estimate the maximum allowable step size as:

⇒ with *S* a safety factor (typically *S* = 0.9)

1. **Embedded methods:** Compare solutions when using two approximations of different order

Use a special fourth and a fifth order Runge Kutta method to approximate , special because we want to use the same positions for the evaluation for computational efficiency.

RK45 preferred method (minimum number of function evaluations)   
(this is built in Matlab as ode45).

*Slide 21: Problems with Euler’s method: instability*

with

First order approximation of derivative:

Where to evaluate the function *f*?

1. (Explicit) Euler method (forward Euler):

use values at *xi*: ⇒ )   
= explicit equation for *yi*+1 in terms of *yi*

⇒ Can give instabilities with large function values, e.g.

First order Batch reactor: ⇒ ⇒

Thus if unphysical results are obtained!!

Stability requirement: (but probably accuracy requirement more stringent here!)

1. Implicit Euler method (Backward Euler):   
   use values at *xi*+1: ⇒ )

Consider again first order Batch reactor: ⇒ ⇒

Never unphysical results!! ⇒ Unconditionally stable

Example: Batch reactor with first order reaction:

Show two figures with on the left Explicit Euler and on the right implicit Euler for high value of *k*.

⇒ Implicit Euler always stable (but maybe not very accurate/efficient).

*Slide 22: Semi-implicit Euler method*

Usually *f* is a non-linear function of *y* ⇒ linearization required (Newton’s method):

⇒ using

⇒ ⇒

⇒

For the case that we could add the variable *x* as an additional variable

Or add one fully implicit Euler step (which avoids the computation of ):

⇒ ⇒

*Slide 23: Semi-implicit Euler method – example*

Second order reaction in a batch reactor

with *c*(*t*=0) = 1 mol/m3, *k* = 1 (m3·mol-1·s-1), *tend* = 2 (s)

*Define* , then ⇒ (note analytical: )

|  |  |  |  |
| --- | --- | --- | --- |
| N | ζ |  |  |
| 20 | 0.654066262 | 0.018900607 | - |
| 40 | 0.660462687 | 0.009305969 | 1.022204214 |
| 80 | 0.663589561 | 0.004615658 | 1.01162005 |
| 160 | 0.665134433 | 0.002298350 | 1.00593795 |
| 320 | 0.665902142 | 0.001146787 | 1.003000697 |

*Slide 24: Second order implicit method: Implicit midpoint method*

|  |  |
| --- | --- |
| **Implicit midpoint rule** (second order) | **Explicit midpoint rule**  (modified Euler method) |
|  |  |

In case then:

Implicit midpoint rule reduces to:

⇒ ⇒

*Slide 25: Second order implicit method: Implicit midpoint method*

Second order reaction in a batch reactor

with *c*(*t*=0) = 1 mol/m3, *k* = 1 (m3·mol-1·s-1), *tend* = 2 (s)

*Define* , then ⇒ (note analytical: )

Substitution:

⇒

You will find that this method is exact for all step sizes *h* because of the quadratic source term!!

For example: third order reaction:

⇒ (analytical solution: )

|  |  |  |  |
| --- | --- | --- | --- |
| N | ζ |  |  |
| 20 | 0.552691617 | 0.000171472 | - |
| 40 | 0.552763373 | 4.16641·10-5 | 2.041091195 |
| 80 | 0.552780730 | 1.02645·10-5 | 2.021148588 |
| 160 | 0.552784996 | 2.54715·10-6 | 2.010704561 |
| 320 | 0.552786054 | 6.34415·10-7 | 2.005382122 |

*Slide 26: Systems of ODE’s:*

Overview of formula’s

Examples: (stiffness)

*Slide 28: Other methods*

Other explicit methods:

Bulirsch-Stoer method (Richardson extrapolation + modified midpoint method)

Other implicit methods:

Rosenbrock methods (higher order implicit Runge Kutta methods)

Predictor-corrector methods