Durham University
Department of Computer Science

Numerical Algorithms (COMP 3371)

Session VII: Truncation errors, stability and consistency

Dr. Weinzierl

Michaelmas term 2019

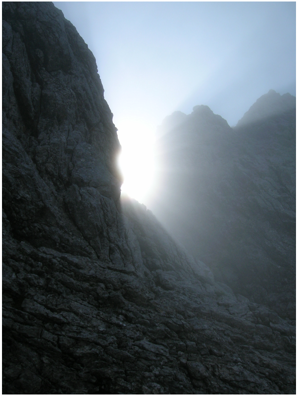
Recapitulation

Durham University
Department of Computer Science

- Can you explain the idea of explicit Euler at hands of a Taylor series?
- Is the explicit Euler stable w.r.t. rounding errors?
- What is the definition of machine precision?
- What is the difference between the local round-off error and the global round-off error?

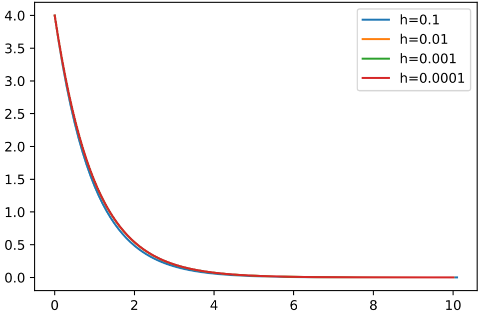
Outline

Durham University
Department of Computer Science



Consistency
Stability
Consistency and convergence

Playing around with explicit Euler



Consistency of explicit Euler (1/2)

Durham University
Department of Computer Science

$$y'(t) = F(y, t)$$

- The smaller h, the better our scheme approximates the real solution
- Intuitively clear, as we used Taylor and did truncate it
- Can we show it?

Consistency of explicit Euler (1/2)

Durham University
Department of Computer Science

$$y'(t) = F(y,t)$$

- The smaller h, the better our scheme approximates the real solution
- Intuitively clear, as we used Taylor and did truncate it
- Can we show it?
- Exact solution:

$$y(t+\Delta t) = \sum_{i=0}^{\infty} \frac{\Delta t^i}{i!} y^{(i)}(t)$$

- Our approximation:

$$y_h(t+\Delta t) = y_h(t) + \underbrace{\Delta t \cdot y_h'(t)}_{plug\ F\ into} + \underbrace{\sum_{i=2}^{\infty} \frac{\Delta t^i}{i!} y^{(i)}(t)}_{cut\ off\,/\,truncate}$$

- Subscript h used as $h = \Delta t$ very often denotes discretisation of continuum (time)

Consistency of explicit Euler (2/2)

Durham University
Department of Computer Science

- Error equals truncated terms:

$$y_h(t+\Delta t) - y(t+\Delta t) = -\sum_{i=2}^{\infty} \frac{\Delta t^{(i)}}{i!} y^{(i)}(t)$$

- Reiterate definition:

> Consistency: A scheme is consistent if
>
> $$\lim_{h \mapsto 0} y_h(t) = y(t)$$
>
> and $y(t)$ is the analytical solution.

- Study error for $\Delta t \mapsto 0$ (from the right side):

$$\lim_{\Delta t \mapsto 0^+} y_h(t+\Delta t) - y(t+\Delta t) = \lim_{\Delta t \mapsto 0^+} -\sum_{i=2}^{\infty} \frac{\Delta t^{i}}{i!} y^{(i)}(t)$$

- Trivially goes to zero as long as derivatives are well-behaved

Consistency and truncation error

Durham University
Department of Computer Science

$$y'(t) = F(y,t)$$

Consistency: A scheme is consistent if

$$\lim_{h \mapsto 0} y_h(t) = y(t)$$

and $y(t)$ is the analytical solution.

- Explicit Euler is trivially consistent, as we simple cut the Taylor at $\mathcal{O}(h^2)$
- Consistency means we solve the *right* problem (in the limit)
- The $\mathcal{O}(h^2)$ is the *truncation error*
- Consistent scheme = truncation error goes to zero with $h \mapsto 0$

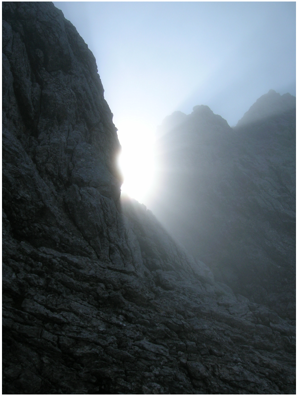
We now switch from Δt to h forth and back!

Concept of building block

Durham University
Department of Computer Science

- Content
  - Introduce term consistency
  - Define truncation error
- Expected learning outcomes
  - The student *knows* definition of introduced terms
  - The student can *explain* what consistency means (“solve right problem”)

Outline



Consistency
Stability
Consistency and convergence

Recap

Durham University
Department of Computer Science

- We know that the round-off error does not make our algorithm explode
  (if F is well-behaved)
- We know that our system is consistent, i.e. solved (in the limit) the right problem

$\Rightarrow$ what could possibly go wrong?

Explicit Euler with error measurements

for the test equation

Durham University
Department of Computer Science

Explicit Euler: For the test equation, we know the analytical solution and thus can directly compute the error. This does not work for more complex problems.

$$
\begin{aligned}
y`(t) &= \lambda y(t) \qquad \text{with } y(t=0) = y_0 \\
y_h(t+h) &= y_h(t) + h \cdot y_h`(t) = y_h(t) + h\lambda y_h(t) = (1+h\lambda)y_h(t) \\
y(t) &= y_0 \cdot e^{\lambda t}
\end{aligned}
$$

```
const double y0 = 4.0;

double t      =   0.0;
double y      =    y0;
double lambda =  -1.0;
double h      = 0.001;

while (t<2.0) {                        // simulate until t=2.0
  t = t + h;
  y = (1.0+lambda*h) * y;
  double error = y - y0 * exp(lambda * t);
  std::cout << "t=" << t << ",y=" << y << ",e=" << error << std::endl;
}
```

Writing down the (pseudo-) code

Results for $\lambda = -1.0$:

- `h=0.001`: `t=2.001,y=0.540259,e=-0.000541161`
- `h=0.002`: `t=2,y=0.540258,e=-0.00108304`
- `h=0.004`: `t=2,y=0.539174,e=-0.00216681`
- `h=0.032`: `t=2.016,y=0.515475,e=-0.0172737`
- `h=0.256`: `t=2.048,y=0.375529,e=-0.140442`

$\Rightarrow$ Please validate yourself (bottom-up)!

Observations:

- Some round-off error in `t` for `h=0.001`
- Halving *h* roughly reduces error by factor of two, i.e.
- error decreases if we reduce *h* (consistency)
- Every run here gives us some meaningful result (subject to errors)

$\Rightarrow$ Confirms our assumption that everything is fine

Another “material” parameter

Durham University
Department of Computer Science

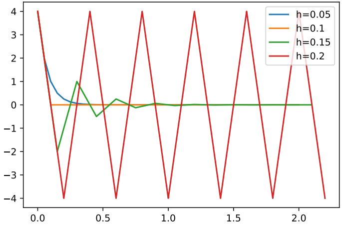
With $\lambda = -10.0$:

- `dt=0.001`: `h=2.001,y=7.38048e-09,e=-7.82103e-10`
- `h=0.002`: `t=2,y=6.73187e-09,e=-1.51275e-09`
- `h=0.256`:

```
t=0.256,y=−6.24,e=−6.54922
t=0.512,y=9.7344,e=9.7105
t=0.768,y=−15.1857,e=−15.1875
t=1.024,y=23.6896,e=23.6895
t=1.28,y=−36.9558,e=−36.9558
t=1.536,y=57.6511,e=57.6511
t=1.792,y=−89.9357,e=−89.9357
t=2.048,y=140.3,e=140.3
```

Overshooting of explicit Euler

Durham University
Department of Computer Science



Stability (1/2)

Durham University
Department of Computer Science

- Stability typically is studied at hands of Dahlquist test equation $\partial_t y = \lambda y$ with fixed time step size $\Delta t = h$
- We know analytical solution, and can construct stable and unstable solutions due to the choice of λ
- Explicit Euler (on step):

$$y_h(nh) = (1 + h\lambda)y_h((n-1)h)$$

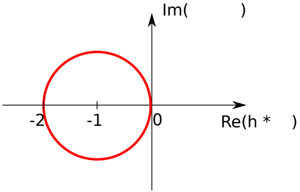
- Apply recursion/induction to derive global scheme for test equation
- Global scheme:

$$y_h(nh) = (1 + h\lambda)^n y_h(0)$$

$\Rightarrow$ We can now determine the nth approximation directly without any recursion.
$\Rightarrow$ We know for $\lambda < 0$ that it should go towards 0 (attractive/stable critical point).
$\Rightarrow$ Does it do so?

Stability (2/2)

Durham University
Department of Computer Science



$$y_h(nh) = (1 + h\lambda)^n y_h(0)$$

$$\lim_{n \mapsto \infty} y_h(nh) = 0 \quad \Leftrightarrow \quad |1 + h\lambda| < 1$$

- We see stability for $|(1 + h\lambda)| < 1$, i.e. $0 < h < -2/\lambda$
  $\Rightarrow$ consistency is trivial but stability not
- We now have a formal criterion for oscillations of error in the test equation
- For another ODE, we have to redo these steps

Flavours of stability

Durham University
Department of Computer Science

Round-off stability: . . . (covered that one)

Zero stability: Small perturbations of input data do not make the truncation error explode as $h \mapsto 0$.

- Clear/trivial for explicit Euler
- Necessary for a consistent scheme
- Kind of generic for explicit Euler

Absolute (A) stability: Small perturbations of input data do not make the truncation error explode.

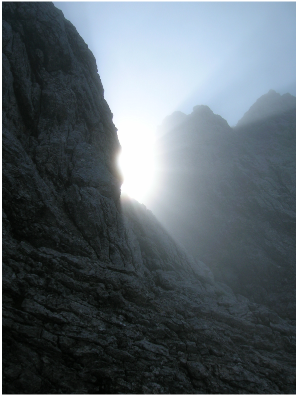
- For every λ, we can construct h such that explicit Euler breaks
- Necessary for a consistent scheme
- Depends on problem

Concept of building block

- Content
  - Discuss when explicit Euler fails and how
  - Write down global iteration scheme
  - Derive stability criteria
  - Sketch proper time step choice for explicit Euler
- Expected learning outcomes
  - Student knows formalisation of convergence analysis
  - Student can explain formally and at hands of sketches why and where explicit Euler becomes unstable
  - Student can analyse any time stepping scheme w.r.t. stability
- Revision for exam
  - Experiment with different time step sizes for the two-body problem from the coursework (collision)

Outline

Durham University
Department of Computer Science



Consistency
Stability
Consistency and convergence

Convergence

Durham University
Department of Computer Science

> Lax Equivalence Theorem: An algorithm converges iff it is consistent and stable.

- These are the schemes we are searching for
- We can “rely” our computer simulation’s outcome

Error analysis with Taylor for Euler's method (1/2)

Durham University
Department of Computer Science

$$y' = F(t, y)$$

Local error: The local error is the error we make in one step.

- Taylor series: $y(t+h) = y(t) + hy`(t,y) + h^2/2y``(t,y) + h^3/6y```(t,y) + \dots$
- Explicit Euler: $y_h(t+h) = y_h(t) + hy`(t, y_h(t))$
- Error: $e(t) = y_h(t) - y(t)$

First time step:

$$e(h) \quad = \quad y_h(h) - y(h)$$

Error analysis with Taylor for Euler's method (1/2)

Durham University
Department of Computer Science

$$y' = F(t,y)$$

Local error: The local error is the error we make in one step.

- Taylor series: $y(t+h) = y(t) + hy`(t,y) + h^2/2y``(t,y) + h^3/6y```(t,y) + \dots$
- Explicit Euler: $y_h(t+h) = y_h(t) + hy`(t, y_h(t))$
- Error: $e(t) = y_h(t) - y(t)$

First time step:

$$\begin{aligned} e(h) &= y_h(h) - y(h) \\ &= \underbrace{y_h(0) - y(0)}_{=0} + h \cdot \underbrace{F(0, y_h(0)) - F(0, y(0))}_{=0} + \dots \end{aligned}$$

- Error is $h \cdot (f`(t,y) - f`(t, y_h(t))) + \mathcal{O}(h^2) = \mathcal{O}(h^2)$ as the y' are exact as well.
- Scheme is second order.

Error analysis with Taylor for Euler's method (2/2)

> Global error: The global error is the error between the numerical solution and the true solution (with multiple local steps in-between) at a given point.

$\Rightarrow$ Is the global error just the sum of all the local errors?

First time step:

- Error is $h \cdot (y'(t,y) - y'(t,y_h(t))) + \mathcal{O}(h^2) = \mathcal{O}(h^2)$ as y'_h is exact.
- Scheme is second order.

Second time step:

- There is an additional error that is propagated.
- It results from the fact that $y'(t,y) - y'(t,y_h(t)) \neq 0$.
- This reduces the scheme to first order, i.e. the inaccurate derivatives pollute the solution.

Convergence order

Durham University
Department of Computer Science

Convergence order p: An algorithm converges with order p if

$$|y_h(t) - y(t)| \leq C \cdot h^p.$$

- Global property (not only single time step)
- $p = 1$ for explicit Euler (as we do not write $t = h$)

Convergence (rephrased): An algorithm converges if

$$\forall t: \lim_{h \mapsto 0} \|y_h(t) - y(t)\| = 0.$$

Remark: We “only” need zero-stability though in practice A-stability might be more important.

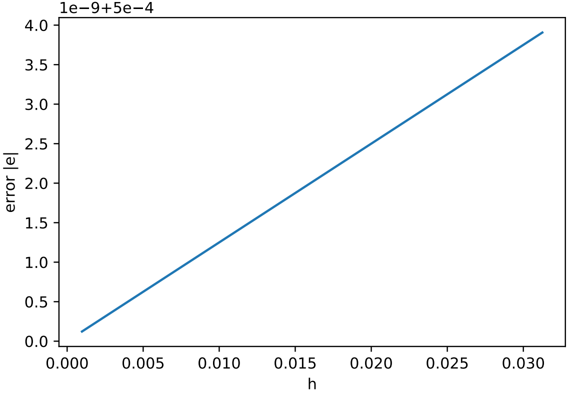
Plotting convergence

Nice. But the motivation behind the whole course is that we don’t know the answer and thus can’t compute e!

Measuring convergence

- Error for one h:
$$e_h(t) = (y_h - y)(t)$$
Let's drop the t parameter. We always measure things at one point in time
- Error for more accurate simulation:
$$e_{h/2} = y_{h/2} - y$$
- Plug into each other:
$$e_h - e_{h/2} = y_h - h - y_{h/2} + y = y_h - y_{h/2}$$
- We still do not know the error, but we know how the error has changed:
$$e \in \mathcal{O}(h^p) \Leftrightarrow e = Ce^p$$
$$e_h - e_{h/2} = \underbrace{y_h - y_{h/2}}_{measure} \quad = \quad Ch^p - C\,(h/2)^p = C(1 - \frac{1}{2^p})h^p$$
- Now continue with $y_h - y_{h/4}$, and $y_h - y_{h/8}$, and ...

Example


1e−9+5e−4
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
error |e|
0.000
0.005
0.010
0.015
0.020
0.025
0.030
h

Concept of building block

Durham University
Department of Computer Science

- Content
  - Define convergence
  - Study local and global error
  - Convergence order
  - Discuss measurements
- Expected learning outcomes
  - The student *knows* definition of introduced terms
  - The student can *compute* convergence order for model problems
  - The student can *experimentally determine* convergence orders
- Revision for exam
  - Run some numerical experiments (for test equation and with coursework, e.g.) and determine convergence order; this also works for other pieces of coursework

Summary, outlook & homework

Durham University
Department of Computer Science

Concepts discussed:

- We have done all of our explicit Euler homework

Next:

- We study ways how to choose proper time step sizes
- We continue our search for a better scheme which is more stable all the time
- We start to construct more accurate schemes

Preparation:

- Solve the problem **Stiff problems** from the worksheet