TDT4127 Programming and Numerics
Week 44
Solving ordinary differential equations
- More dimensions, stability and accuracy
Some facts about the exam

- **November 30, 09:00-13:00.**
  - Check location at studentweb (may not be available yet)

- The exam will be **digital**
  - Same as the auditorium exercises
    - [https://innsida.ntnu.no/wiki/-/wiki/norsk/digital+eksamen](https://innsida.ntnu.no/wiki/-/wiki/norsk/digital+eksamen)

- Theory questions will have **multiple choice** answers
  - Mostly numerics, possibly some programming related

- Theory questions and numerical exercise(s) will be similar to those in **Auditorium exercise 2**

- You do **not** need to remember formulas for the numerics
  - But it will of course help if you are familiar with them!
Learning goals

• Goals
  – Solving ordinary differential equations
  – Analysis of algorithms:
    • Explicit Euler method
    • Implicit Euler method
    • Heun’s method
  – Stability and accuracy

• Curriculum
  – Exercise set 9
    • But only in the interpretation of results
Numerical methods for ODEs

- Last week: explicit Euler, implicit Euler, Heun’s method
- These schemes numerically solve the ODE
  \[ \dot{x}(t) = f(x(t), t) \]
- The explicit Euler method:
  \[ x^{j+1} = x^j + hf(x^j, t_j) \]
- The implicit Euler method:
  \[ x^{j+1} = x^j + hf(x^{j+1}, t_{j+1}) \]
- Heun’s method:
  \[ s^{j+1} = x^j + hf(x^j, t_j) \]
  \[ x^{j+1} = x^j + \frac{h}{2} \left( f(x^j, t_j) + f(s^{j+1}, t_{j+1}) \right) \]
Treating ODEs in many dimensions

• In more than one dimensions, equations are vectorized
  \[ \dot{x}(t) = f(x(t), t) \]
• The methods are exactly the same, but with vectors
• The explicit Euler method:
  \[ x^{j+1} = x^j + hf(x^j, t_j) \]
• The implicit Euler method:
  \[ x^{j+1} = x^j + hf(x^{j+1}, t_{j+1}) \]
• Heun’s method:
  \[ s^{j+1} = x^j + hf(x^j, t_j) \]
  \[ x^{j+1} = x^j + \frac{h}{2} \left( f(x^j, t_j) + f(s^{j+1}, t_{j+1}) \right) \]
• Implementation difference: vector addition. Numerical solver in several dimensions for the implicit Euler method
Stability

• When solving an ODE numerically, we need to choose an appropriate time step size $h$ (i.e. number of steps $N$).
  – Too small $h \rightarrow$ takes long time to compute solution
  – Too large $h \rightarrow$ inaccurate (bad) or unstable (worse) solutions

• What is meant by stability and instability?
  – Instability is when the solution «blows up» when it’s not supposed to
  – Stability is when it doesn’t

• Test example: Apply the numerical method to the ODE
  \[ \dot{x}(t) = -\lambda x(t), \quad \lambda > 0, \quad x(0) = x_0 \]

• This equation has the strictly decreasing solution
  \[ x(t) = x_0 e^{-\lambda t} \]
Numerical instability example

- Below: *explicit Euler* applied with various $h$ to the ODE
  $$\dot{x}(t) = -8x(t), \quad \lambda = 8, \quad x(0) = 1$$

Left: *instability*. Right: *stability*
Numerical stability example

- Below: implicit Euler applied with various $h$ to the ODE
  \[ \dot{x}(t) = -8x(t), \quad \lambda = 8, \quad x(0) = 1 \]

Left: stability.
Right: stability
Numerical stability explained

• So what happens? Compare solutions for this particular ODE:
  \[ \dot{x}(t) = -\lambda x, \quad \lambda > 0, \quad x(0) = x_0 \]

• Explicit Euler:
  
  \[ x_{n+1} = (1 - \lambda h)x_n = (1 - \lambda h)^2 x_{n-1} = \cdots = (1 - \lambda h)^n x_0 \]

• Blows up if \(|1 - \lambda h| > 1\), i.e. if \(\lambda h > 2\).
  – Must take \(h < 2/\lambda\)! This can be restrictive.

• Implicit Euler:

  \[ x_{n+1} = x_n - \lambda hx_{n+1} \]

  \[ x_{n+1} = \frac{1}{1 + \lambda h} x_n = \left(\frac{1}{1 + \lambda h}\right)^2 x_{n-1} = \cdots = \left(\frac{1}{1 + \lambda h}\right)^n x_0 \]

• Always decreasing, no matter the value of \(\lambda\) or \(h\)
Stability summarized

• Intuition: *different methods have different stability properties.*
  – Some methods are stable for larger or even all $h$
  – There is a large literature on this for *Runge-Kutta* methods
  – **Implicit** methods are typically **more stable** than explicit ones
  – No **explicit** method is stable for all $h$
  – All methods work with **small enough** $h$
    • But it may mean a restrictively very long computation time

• What about *Heun’s method*?
  – **More stable** than explicit *Euler*, but it is **explicit** and has restrictions

• **Practical tips:** If you see unwanted blow-up or oscillations, try a smaller $h$ first, before switching to another solver.
Accuracy

• Assuming our solutions don’t blow up, the next question is: how **accurate** are they?
• Accuracy is a function of $h$
Accuracy

• Assuming our solutions don’t blow up, the next question is: how **accurate** are they?
• Accuracy is a function of $h$
Accuracy

• Assuming our solutions don’t blow up, the next question is: how accurate are they?
• Accuracy is a function of $h$
Accuracy

- Accuracy is measured in terms of the global error
  \[ e_j = |x(t_j) - x_j|, \quad t_j = jh \]
- A method is said to be of order \( p \) if for some constant \( C_j \),
  \[ e_j \leq h^p C_j \]
- Explicit and implicit Euler are both of order 1:
  \[ e_j^{\text{Euler}} \leq h C_j, \quad C_j = \frac{e^{L(t_j-t_0)} - 1}{L} \]
- Heun’s method is of order 2:
  \[ e_j^{\text{Heun}} \leq h^2 C_j, \quad C_j = \frac{e^{L(t_j-t_0)} - 1}{L} \]
- Note: These \( C_j \) estimates are very conservative and should not be used in practice. **The order is what’s important.**
Accuracy

• Practical consequence: order $p$ methods improve errors by a factor $2^p$ when halving the step size.
• The table below demonstrates the orders by considering the global error at $T = 1$ for the test problem from before.
• explicit/implicit Euler: $p = 1$, Heun’s method: $p = 2$

<table>
<thead>
<tr>
<th>h</th>
<th>Explicit Euler</th>
<th>Implicit Euler</th>
<th>Heun’s method</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.01</td>
<td>$9.62 \times 10^{-5}$</td>
<td>$11.91 \times 10^{-5}$</td>
<td>$30.54 \times 10^{-7}$</td>
</tr>
<tr>
<td>0.005</td>
<td>$5.09 \times 10^{-5}$</td>
<td>$5.66 \times 10^{-5}$</td>
<td>$7.38 \times 10^{-7}$</td>
</tr>
<tr>
<td>0.0025</td>
<td>$2.61 \times 10^{-5}$</td>
<td>$2.76 \times 10^{-5}$</td>
<td>$1.82 \times 10^{-7}$</td>
</tr>
</tbody>
</table>

• High-order methods gain a lot from smaller time steps!
  – Methods with order 4 are often used!
Summary of ODE solvers

• **Explicit/implicit** methods
  – **Explicit**: Can calculate directly. Explicit Euler and Heun’s method
  – **Implicit**: Need to solve an equation per step. Implicit Euler

• **Multi-stage**
  – Methods can have more than one **stage**. Heun’s method

• **Stability**
  – Does the method blow up when applied to $\dot{x}(t) = -\lambda x$?  
  – **Explicit** methods are **unstable** for too **large step sizes** $h$  
  – **Implicit** methods are generally **more stable**

• **Accuracy**
  – A method is **order** $p$ accurate if $e_j \leq h^p C_j$.  
  – implicit/explicit Euler methods are **order 1**  
  – Heun’s method is **order 2**  
  – Can construct integration methods of even **higher order**
Next weeks

• Three lectures left
  – **Adaptive Simpson** next week (November 9)
    • Last regular lecture
  – **Repetition** and **exam prep** on November 16 and November 23!
  – I will go through the numerics from **auditorium exercise 2** in detail
  – **Suggest other topics** you want me to cover
    • Otherwise, I’ll pick them myself
Questions?