Contents

Lecture 0 Module logistics & Coursework guidelines

Module title

Numerical solution of Ordinary Differential equations

Brief description

The module is an introductory course in numerical methods for ordinary differential equations. The purpose of this course is to learn how to develop and analyse your own numerical methods as well as to provide you with theoretical knowledge and practical skills to lay solid foundations necessary to advance in scientific computing. In this course we will be mostly focused on the numerical solution of initial value problems for ordinary differential equations. In addition we will consider boundary value problems for both ordinary and partial differential equations.

Learning outcomes

On the successful completion of the module you will be able to

- use classical numerical methods for ordinary differential equations
- analyse different properties of numerical methods (accuracy, stability, etc.)
- develop your own methods with prescribed properties
- compare different methods with respect to accuracy, stability, computational and space complexity
- develop efficient numerical algorithms
- apply numerical methods to solve boundary value problems for partial differential equations

Brief syllabus

- Taylor series methods
- Linear multi-step methods
- Improved approximations
- Runge-Kutta methods
- Adaptive step size control
- Boundary value problems for ordinary differential equations
- Introduction to Finite Difference Method
- Introduction to boundary value problems for partial differential equations

Pattern of teaching and teaching approach

Two (four) hours of lectures followed by a one-hour tutorial (a two-week coursework) the goal of which is to ensure that students can use their knowledge and skills in analysis of numerical methods and solving practical problems.

Strategy for assessment and feedback

Your performance is assessed through tutorials and courseworks. Feedback is given in written form and in person.

Prerequisites

Basic knowledge of MATLAB (or Python) and theory of ordinary differential equations is expected.

Recommended texts

- E. Coddington and N. Levinson, Theory of ordinary differential equations.
- E. Ince Ordinary differential equations.
- E. Hairer et al., Solving ordinary differential equations I. Nonstiff problems.

- J. Butcher, Numerical methods for ordinary differential equations.
- J. Lambert, Numerical methods for ordinary differential systems.
- H. Keller, Numerical solution of two point boundary value problems.

I would recommend to take this course if you want to progress into numerical methods for PDEs, numerical analysis, and simulations of real-life problems.

Note that all material below is lecture notes *not lecture slides*, and therefore it can solely be used without lecture recordings to master the course. Feel free to contact me at i.shevchenko@imperial.ac.uk if you think the lecture notes are missing or lacking something, or there is a way to make them better. I can do it on the fly so that you can benefit from it during the course.

Registration for the module

By submitting Coursework 1 you confirm to take the module for credit.

Coursework Guidelines

Below is a set of guidelines to help you understand what coursework is and how to improve it.

Coursework

- The coursework requires more than just following what has been done in the lectures, some amount of individual work is expected.
- The coursework report should describe in a concise, clear, and coherent way of what you did, how you did it, and what results you have.
- The report should be understandable to the reader with the mathematical background, but unfamiliar with your current work.
- Do not bloat the report by paraphrasing or presenting the results in different forms.
- Use high-quality and carefully constructed figures with captions and annotated axis, put figures where they belong.
- All numerical solutions should be presented as graphs.
- Use tables only if they are more explanatory than figures. The maximum table length is a half page.
- All figures and tables should be embedded in the report. The report should contain all discussions and explanations of the methods and algorithms, and interpretations of your results and further conclusions.
- The report should be typeset in LaTeX or Word Editor and submitted as a single pdf-file.
- The maximum length of the report is ten A4-pages (additional 3 pages is allowed for Year 4 students); the problem sheet is not included in these ten pages.
- Do not include any codes in the report.
- Marks are not based solely on correctness. The results must be described and interpreted. The presentation and discussion is as important as the correctness of the results.

Codes

- You cannot use third party numerical software in the coursework.
- The code you developed should be well-structured and organised, as well as properly commented to allow the reader to understand what the code does and how it works.
- All codes should run out of the box and require no modification to generate the results presented in the report.

Submission

- The coursework submission must be made via Turnitin on your Blackboard page. You must complete and submit the coursework anonymously, **the deadline is 1pm on the date of submission** (unless stated otherwise). The coursework should be submitted via two separate Turnitin drop boxes as a pdf-file of the report and a zip-file containing MATLAB (m-files only) or Python (py-files only) code. The code should be in the directory named `CID_Coursework#`. The report and the zip-file should be named as `CID_Coursework#.pdf` and `CID_Coursework#.zip` , respectively. The executable MATLAB (or Python) scripts for the exercises should be named as follows: exercise1.m, exercise2.m, etc.

Lecture 1 The Euler method (Forward Euler method)

Let us consider an initial value problem (IVP) of the form

$$\mathbf{x}' = \mathbf{f}(t, \mathbf{x}), \quad \mathbf{x}(t_0) = \boldsymbol{\alpha}, \quad t \in [t_0, t_N], \quad \mathbf{x}' := \frac{d\mathbf{x}}{dt}, \quad \mathbf{x} = \mathbf{x}(t), \tag{1}$$

with $\mathbf{f} : \mathbb{R} \times \mathbb{R}^d \to \mathbb{R}^d$ being a given function, $\boldsymbol{\alpha}$ is a given initial condition, and $\mathbf{x} \in \mathbb{R}^d$, where $d \in \mathbb{Z}^+$ (positive integers). Throughout the course we assume that all functions used are continuous functions differentiable as many times as needed.

We start from the simplest numerical method proposed by Leonard Euler in 1770-ish, and named after his name afterwards. The method is based on a simple idea. Let us suppose that a particle $\boldsymbol{x}$ is moving in such a way that at time t_0 its position,$\boldsymbol{x}_0$, and velocity, $\boldsymbol{v}_0$ are known. The simple idea is that within an extremely short period of time, $\Delta t = t_1 - t_0$, the velocity at time t_1, $\boldsymbol{v}_1$, has no time to change significantly from the initial velocity $\boldsymbol{v}_0$ (i.e. $\boldsymbol{v}_1 \approx \boldsymbol{v}_0$), and thus the change in the position of the particle, $\Delta \boldsymbol{x} = \boldsymbol{x}_1 - \boldsymbol{x}_0$, is approximately equal to the change in time, Δt, multiplied by the initial velocity $\boldsymbol{v}_0$, i.e.

$$\Delta \boldsymbol{x} = \Delta t \boldsymbol{v}_0. \tag{2}$$

Let us rewrite (2) as

$$\boldsymbol{x}_1 - \boldsymbol{x}_0 = (t_1 - t_0)\boldsymbol{v}_0 \tag{3}$$

to find

$$\boldsymbol{x}_1 = \boldsymbol{x}_0 + (t_1 - t_0)\boldsymbol{v}_0. \tag{4}$$

We assume that $\boldsymbol{v}_1$ can be found accurately enough from $\boldsymbol{x}_1$ and t_1 using the differential equation $\boldsymbol{x}' = \boldsymbol{v}(t)$ which describes the motion of particle $\boldsymbol{x}$. If we know $\boldsymbol{v}_1$ then we can find the position of the particle at time t_2:

$$\boldsymbol{x}_2 = \boldsymbol{x}_1 + (t_2 - t_1)\boldsymbol{v}_1. \tag{5}$$

Finally, for t_n the position of the particle is given by

$$\boldsymbol{x}_n = \boldsymbol{x}_{n-1} + (t_n - t_{n-1})\boldsymbol{v}_{n-1}, \tag{6}$$

or

$$\boldsymbol{x}_n = \boldsymbol{x}_{n-1} + \Delta t \boldsymbol{v}_{n-1}. \tag{7}$$

Thus, for IVP (1) we have

$$\text{Euler method (Forward Euler method):} \qquad \boldsymbol{x}_n = \boldsymbol{x}_{n-1} + h\boldsymbol{f}_{n-1}, \quad \boldsymbol{x}_0 = \boldsymbol{\alpha}, \tag{8}$$

or

$$\text{Euler method (Forward Euler method):} \qquad \boldsymbol{x}_{n+1} = \boldsymbol{x}_n + h\boldsymbol{f}_n, \quad \boldsymbol{x}_0 = \boldsymbol{\alpha}, \tag{9}$$

where $h := \Delta t$, $\boldsymbol{f}_n := \boldsymbol{f}(t_n, \boldsymbol{x}_n)$. Equation (9) is known as the Euler method or the Forward Euler method.

Example 1.1 (Euler's method for a scalar equation) Let's consider the initial value problem for the scalar equation

$$x' = (1 - 2t)x, \quad x(0) = 1, \quad t \in [0, 1]. \tag{10}$$

In order to compute the solution of (10) with the Euler method, we have to define the time step h. Let it be $h = 0.1$. Our next step is to apply the Euler method (9) to (10). For this, we write

$$x_{n+1} = x_n + hf_n, \quad f_n := (1 - 2t_n)x_n, \quad x_0 = 1. \tag{11}$$

Since the initial condition and the right hand side are known, we can compute x_1 (this is the position of the particle at time $t_1 = h = 0.1$):

$$x(0.1) = x(0) + 0.1(1 - 2 \cdot 0)x(0) = 1.10\,. \tag{12}$$

Having computed the position of the particle at time $t_1 = 0.1$, we can now compute its position at time $t_2 = 0.2$:

$$x(0.2) = x(0.1) + 0.1(1 - 2 \cdot 0.1)x(0.1) \approx 1.19\,. \tag{13}$$

For $t_3 = 0.3$ we have

$$x(0.3) = x(0.2) + 0.1(1 - 2 \cdot 0.2)x(0.2) \approx 1.26\,. \tag{14}$$

For $t_{10} = 1.0$ we have

$$x(1.0) = x(0.9) + 0.1(1 - 2 \cdot 0.9)x(0.9) \approx 1.1\,. \tag{15}$$

Acting in the same vein, we can compute the particle position at any given moment in time. ▲

Example 1.2 (Euler's method for higher order ODEs and systems of ODEs) Consider the second order ODE

$$x'' + x = 7, \quad x(t_0) = 10, \quad x'(t_0) = 20, \quad t \in [t_0, t_N]. \tag{16}$$

In order to apply Euler's method to (16), we turn this equation into a second order system of first order ODEs by substituting $x = u$ and $x' = v$ into (16)

$$\begin{cases} u' = v, \\ v' = 7 - u. \end{cases} \tag{17}$$

and into the initial conditions:

$$u(t_0) = 10, \quad v(t_0) = 20. \tag{18}$$

Then, the Euler method for IVP (17)-(18) reads

$$\boldsymbol{y}_{n+1} = \boldsymbol{y}_n + h\boldsymbol{f}_n, \quad \boldsymbol{y}_n = \begin{pmatrix} u_n \\ v_n \end{pmatrix}, \quad \boldsymbol{f}_n = \begin{pmatrix} v_n \\ 7 - u_n \end{pmatrix}, \quad \boldsymbol{y}_0 = \begin{pmatrix} 10 \\ 20 \end{pmatrix}. \tag{19}$$

▲

Although we have found the numerical solution with the Euler method, this solution is of no value to us if the difference between it and the exact solution is too large. It brings us to a series of important definitions.

1.1 Local truncation error

Let us consider the Taylor expansion of the exact solution to IVP (9):

$$x(t + h) = x(t) + hx'(t) + \frac{h^2}{2}x''(t) + \mathcal{O}(h^3). \tag{20}$$

Setting $t = t_n$ in (20) and taking into account that $t_{n+1} = t_n + h$ we have

$$x(t_{n+1}) = x(t_n) + hx'(t_n) + \frac{h^2}{2}x''(t_n) + \mathcal{O}(h^3). \tag{21}$$

Definition 1.1 (Local truncation error (LTE) and error constant) *The local truncation error* *of order $p > 0$ is the difference between the numerical, x_{n+1}, and exact, $x(t_{n+1})$, solutions*

$$x(t_{n+1}) - x_{n+1} = h^p C_p \frac{d^p x}{dt^p} = \mathcal{O}(h^p). \tag{22}$$

The constant C_p is known as the error constant.

Definition 1.2 (Global error) *The difference*

$$e_n = x(t_n) - x_n \tag{23}$$

between the exact solution, $x(t_n)$, and the numerical solution, x_n, is called the global error at time $t = t_n$.

Let us get back to the question: how accurate is the numerical solution we have found? In order to answer it, we first compute the exact solution of equation (10), which is $x(t) = e^{-t(t-1)}$, and than find the global error, as well as another useful quantity called the constant of proportionality $|e_n|/h$.

h	$x(t_n)$	x_n	$\lvert e_n\rvert$	$\lvert e_n\rvert/h$
0.1	1.0	1.1	0.1	1.0
0.05	1.0	1.05	0.05	1.0
0.025	1.0	1.025	0.025	1.0
0.0125	1.0	1.0125	0.0125	1.0

Table 1: Dependence of the global error, $|e_n|$, and the constant of proportionality, $|e_n|/h$, on the time step, h, for Example 1.1.

As can be seen from Table 1, the global error decreases as $h \to 0$ that suggests that the method converges to the exact solution.

Definition 1.3 (Convergence) *A numerical method is said to converge to the exact solution $x(t)$ of a given problem at time $t = t_n$ if the global error*

$$|e_n| = |x(t_n) - x_n| \to 0 \text{ as } h \to 0. \tag{24}$$

In other words, the numerical solution tends to the exact solution as the time step becomes increasingly small. *It is important to note that a numerical method is useless if it does not converge.*

Definition 1.4 (Order of convergence) *A numerical method is said to converge with order p if*

$$e_n = \mathcal{O}(h^p), \quad p > 0. \tag{25}$$

Getting back to Table 1, we notice that the constant of proportionality is 1 thus showing that the global error is proportional to h. In other words, in order to reduce the error by a factor of 10 one should reduce the time step by a factor of 10. If the constant of proportionality is known than we can compute the number of time steps needed to achieve a given accuracy. For example, in order for the Euler method to get $|e_n| \approx 0.001$ at time $t_n = 1$, the method will take around 1000 time steps. Namely, since $|e_n| \approx 0.001$ and $|e_n|/h = 1.0$ then $1.0h \approx 0.001 \Rightarrow h \approx 0.001$. On the other hand, $t_n = nh = 1 \Rightarrow n = 1/h \approx 1000$, where n is the number of time steps.

Theorem 1.1 (Convergence of the Euler method for the linear IVP)
The Euler method for the linear IVP

$$x' = \lambda x + g(t), \quad x(t_0) = \alpha, \quad t \in [0, t_N], \tag{26}$$

where $\lambda \in \mathbb{C}$, $g(t) \in C^1$, and $g(t_n) = g_n$ for all $t_n \in [0, t_N]$ converges, and the global error is bounded from above:

$$|e_N| \leq Bh, \quad B = const > 0. \tag{27}$$

Proof. The Euler method for the linear IVP (26) is given by

$$x_{n+1} = x_n + h(\lambda x_n + g(t_n)) = (1+\hat{h})x_n + hg(t_n), \quad x_0 = \alpha, \quad t_i := ih, \quad i = 0, 1, 2, \ldots, N, \tag{28}$$

where $\hat{h} := h\lambda$. In order to study the convergence of the Euler method, we have to study the behavior of the global error. For doing so, we Taylor expand the exact solution of (26) up to the second order

$$\begin{aligned} x(t_{n+1}) =& x(t_n) + hx'(t_n) & +R \\ =& x(t_n) + h(\lambda x(t_n) + g(t_n)) & +R, \end{aligned} \tag{29}$$

where $R = \mathcal{O}(h^2)$. In accordance with the definition of the global error above, the global error is given by the difference between the exact solution, $x(t_{n+1})$, and the numerical solution, x_{n+1}, namely

$$\begin{aligned} e_{n+1} =& x(t_{n+1}) - x_{n+1} \\ =& (1+\hat{h})(x(t_n) - x_n) & +R \\ =& (1+\hat{h})e_n & +R. \end{aligned} \tag{30}$$

Equation (30) tells us how the global error at time t_{n+1} depends on *the propagated error*

$$(1+\hat{h})e_n$$

and the local truncation error given by the reminder R.

For $n = 0, 1, 2, \ldots, N-1$ we find

$$\begin{aligned} e_0 =& x(t_0) - x_0 = 0, \\ e_1 =& R_1, \\ e_2 =& (1+\hat{h})e_1 + R_2 = (1+\hat{h})R_1 + R_2, \\ e_3 =& (1+\hat{h})e_2 + R_3 = (1+\hat{h})^2 R_1 + (1+\hat{h})R_2 + R_3, \\ &\ldots \\ e_N =& \sum_{i=1}^{N} (1+\hat{h})^{N-i} R_i \,. \end{aligned} \tag{31}$$

The error in the initial condition is zero, since the initial condition is known exactly.

The next step is to estimate $|e_N|$. In order to do that we use the following inequality

$$|1+\hat{h}| \leq 1 + |\hat{h}| \leq e^{|\hat{h}|}. \tag{32}$$

Exponentiating (32) to the power of $(N-i)$ gives

$$|1+\hat{h}|^{N-i} \leq e^{|\hat{h}|(N-i)}. \tag{33}$$

On the other hand, we have

$$e^{|\hat{h}|(N-i)} \leq e^{|\lambda| t_{N-i}} \leq e^{|\lambda| t_N}, \text{ because } t_{N-i} \leq t_N \text{ for } i \in [1, N]. \tag{34}$$

Hence,

$$\begin{aligned} |e_N| \leq & \sum_{i=1}^{N} e^{|\lambda| t_N} |R_i| \\ \leq & N e^{|\lambda| t_N} C h^2, \text{ since}^1 \ |R_i| \leq C h^2, \ C = const > 0 \\ \leq & e^{|\lambda| t_N} C h t_N, \text{ since } t_N = N h. \end{aligned} \tag{35}$$

Finally, we have

$$|e_N| \leq B h, \quad B := e^{|\lambda| t_N} C t_N. \tag{36}$$

Thus, we have proved that the Euler method converges to the exact solution with order $p = 1$. ■

Having proved the convergence of the Euler method for the linear IVP, let us try to do the same for the general nonlinear case.

Theorem 1.2 (Convergence of the Euler method for the general IVP)
The Euler method for the IVP

$$x' = f(t, x), \quad x(t_0) = \alpha, \quad t \in [0, t_N], \tag{37}$$

with a Lipschitz continuous function $f(t, x)$ and $f \in C^1$ converges, and the global error satisfies:

$$|e_N| \leq B h, \quad B = const > 0. \tag{38}$$

Proof. The Euler method for IVP (37) is given by

$$x_{n+1} = x_n + h f_n, \quad x_0 = \alpha, \quad t_i := i h, \quad i = 0, 1, 2, \ldots, N. \tag{39}$$

The proof follows the same steps as that of Theorem 1.1. Namely, we Taylor expand the exact solution of (37) up to the second order

$$\begin{aligned} x(t_{n+1}) = & x(t_n) + h x'(t_n) & + R \\ = & x(t_n) + h f(t_n, x(t_n)) & + R, \end{aligned} \tag{40}$$

where $R = \mathcal{O}(h^2)$. The global error is given by the difference between the exact solution, $x(t_{n+1})$, and the numerical solution x_{n+1}:

$$\begin{aligned} e_{n+1} = & x(t_{n+1}) - x_{n+1} & \\ = & x(t_n) - x_n & + h(f(t_n, x(t_n)) - f_n) + R \\ = & e_n & + h(f(t_n, x(t_n)) - f_n) + R. \end{aligned} \tag{41}$$

As in Theorem 1.1, equation (41) shows us how the global error at time t_{n+1} depends on the propagated error

$$e_n + h(f(t_n, x(t_n)) - f_n)$$

and the local truncation error given by the reminder R. The only difference with the analogous equation in Theorem 1.1 is the term $h(f(t_n, x(t_n)) - f_n)$. To proceed with this equation in the same manner as we did in Theorem 1.1, we take the absolute value of (41), and estimate $|e_{n+1}|$.

$$
\begin{aligned}
|e_{n+1}| =& |e_n + h(f(t_n, x(t_n)) - f_n) + R| \\
\le& |e_n| + h|f(t_n, x(t_n)) - f_n| + |R| \\
& \text{ using the Lipschitz continuity of } f \text{ we have } |f(t_n, x(t_n)) - f_n| \le L|x(t_n) - x_n|,\ L = const \ge 0 \\
\le& (1+\hat{h})|e_n| + |R|, \quad \hat{h} := hL.
\end{aligned}
\tag{42}
$$

For $n = 0, 1, 2, \ldots, N-1$ we find

$$
\begin{aligned}
|e_0| =& x(t_0) - x_0 = 0, \\
|e_1| \le& |R_1|, \\
|e_2| \le& (1+\hat{h})|e_1| + |R_2| = (1+\hat{h})|R_1| + |R_2|, \\
|e_3| \le& (1+\hat{h})|e_2| + |R_3| = (1+\hat{h})^2|R_1| + (1+\hat{h})|R_2| + |R_3|, \\
& \ldots \\
|e_N| \le& \sum_{i=1}^{N} (1+\hat{h})^{N-i} |R_i|\,.
\end{aligned}
\tag{43}
$$

The error in the initial condition is zero, since the initial condition is known exactly.

The next step is to estimate $|e_N|$. Acting in the same way as in Theorem 1.1, we find that

$$
\begin{aligned}
|e_N| \le& \sum_{i=1}^{N} e^{Lt_N} |R_i| \\
\le& N e^{Lt_N} C h^2, \ \text{since}[2]\ |R_i| \le Ch^2, \quad C = const > 0 \\
\le& e^{Lt_N} C h t_N, \ \text{since } t_N = Nh.
\end{aligned}
\tag{44}
$$

Finally, we have

$$
|e_N| \le Bh, \quad B := e^{Lt_N} C t_N. \tag{45}
$$

Thus, we have proved that the Euler method converges to the exact solution with order $p = 1$. ■

Remark 1.1 *The error estimation in Theorem 1.2 is of no use in practice, due to the presence of the exponential term. However, the importance of this proof is that it shows that the error decays globally as $\mathcal{O}(h)$, i.e. the Euler method converges to the exact solution with order $p = 1$.*

The Euler method via Taylor series

Let me draw your attention to another way of deriving the Euler method. I believe you have already noticed the similarity between Euler's method and the first two terms in the Taylor expansion. And this is how one can derive the Euler method using the Taylor series. Just truncate the $\mathcal{O}(h^2)$ terms in Taylor series of $x(t_{n+1})$ to get the Euler method:

$$
x(t_{n+1}) = x(t_n) + hx'(t_n) + \mathcal{O}(h^2) \Rightarrow x(t_{n+1}) = x(t_n) + hx'(t_n). \tag{46}
$$

Using the differential equation $x' = f(t, x)$ in (46) leads to

$$
x(t_{n+1}) = x(t_n) + hf(t_n, x(t_n)), \tag{47}
$$

and finally we have the Euler method:

$$x_{n+1} = x_n + hf_n. \tag{48}$$

The idea of truncating the Taylor series brings us to a new class of numerical methods called Taylor Series Methods which will be studied in the next lecture.

Lecture 2 Taylor series methods

As we have seen, the Euler method accuracy is controlled by the time step – the smaller the time step is, the higher the accuracy becomes. However, following this route is not always practical, since it can take a lot of time steps to get high accuracy. As an alternative, one can use more accurate methods at each time step to get higher accuracy with the same time step or the same accuracy with a larger time step. There are a plethora of methods to achieve this goal. Now, we investigate the possibility of improving the accuracy by retaining more term in the Taylor series. The methods based on this idea form the class of Taylor series (TS) methods.

TS methods are methods of general applicability which can be derived to have a given degree of accuracy, and therefore it is the standard to which we compare the accuracy of numerical methods studied in this course. We start from

Theorem 2.1 (Taylor's theorem) *Assume that $x(t) \in C^{n+1}_{[t_0,t_N]}$ then $x(t)$ has the following expansion of order n about the fixed value $t = t_k \in [t_0, t_N]$:*

$$x(t_k+h) = x(t_k)+hx'(t_k, x(t_k))+\frac{h^2}{2!}x''(t_k, x(t_k))+\frac{h^3}{3!}x'''(t_k, x(t_k))+\ldots+\frac{h^n}{n!}x^{(n)}(t_k, x(t_k))+\mathcal{O}(h^{n+1}), \tag{49}$$

where $x^{(n)}$ is the n-th derivative with respect to t.

Using the ODE $x' = f(t, x)$ we replace the derivative of x in the Taylor series (49):

$$x(t_k+h) = x(t_k)+hf(t_k, x(t_k))+\frac{h^2}{2!}f'(t_k, x(t_k))+\frac{h^3}{3!}f''(t_k, x(t_k))+\ldots+\frac{h^n}{n!}f^{(n-1)}(t_k, x(t_k))+\mathcal{O}(h^{n+1}). \tag{50}$$

A Taylor series method of order n (TS(n) method for short) is derived by retaining the terms up to order n in the expansion. In particular,

$$\begin{aligned}
&\text{TS(1) method (Euler's method):} && x_{n+1} = x_n + hf_n.\\
&\text{TS(2) method:} && x_{n+1} = x_n + hf_n + \frac{h^2}{2!}f'_n.\\
&\text{TS(3) method:} && x_{n+1} = x_n + hf_n + \frac{h^2}{2!}f'_n + \frac{h^3}{3!}f''_n.\\
& && \ldots\\
&\text{TS(n) method:} && x_{n+1} = x_n + hf_n + \frac{h^2}{2!}f'_n + \frac{h^3}{3!}f''_n + \ldots + \frac{h^n}{n!}f^{(n-1)}_n.
\end{aligned} \tag{51}$$

Note that the TS(n) method has the local truncation error and global error of order $\mathcal{O}(h^{n+1})$ and $\mathcal{O}(h^n)$ (to be proven below), respectively. Hence, n can be chosen as large as necessary to make the error as small as desired.

Let us consider some example on how to apply TS methods to scalar equations and systems of equations.

Example 2.1 (TS(2) for a scalar ODE) Let's consider the initial value problem for the scalar equation

$$x' = (1 - 2t)x, \quad x(0) = 1, \quad t \in [0, 1]. \tag{52}$$

Taking into account that $x'' = (1 - 2t)x' - 2x$, the TS(2) method for (52) is given by

$$x_{n+1} = x_n + hf_n + \frac{h^2}{2!}f'_n, \quad f_n := (1 - 2t_n)x_n, \quad f'_n := (1 - 2t_n)f_n - 2x_n. \tag{53}$$

▲

Example 2.2 (TS(3) for a scalar ODE) Let's consider the initial value problem for the scalar equation

$$x' = (1-2t)x, \quad x(0) = 1, \quad t \in [0,1]. \tag{54}$$

Taking into account that $x'' = (1-2t)x' - 2x$ and $x''' = (1-2t)x'' - 4x'$, the TS(3) method for (54) is

$$x_{n+1} = x_n + hf_n + \frac{h^2}{2!}f_n' + \frac{h^3}{3!}f_n'', \quad f_n := (1-2t_n)x_n, \quad f_n' := (1-2t_n)f_n - 2x_n, \quad f_n'' := (1-2t_n)f_n' - 4f_n. \tag{55}$$

▲

Example 2.3 (TS(2) for a system of ODEs) Let's consider the initial value problem for the system of ODEs

$$\begin{cases} u' = v, \\ v' = t^2 - u. \end{cases} \tag{56}$$

with the initial conditions:

$$u(t_0) = 2, \quad v(t_0) = 3. \tag{57}$$

Then, the TS(2) method for IVP (56)-(57) reads

$$\boldsymbol{x}_{n+1} = \boldsymbol{x}_n + h\boldsymbol{f}_n + \frac{h^2}{2!}\boldsymbol{f}_n', \quad \boldsymbol{x}_n = \begin{pmatrix} u_n \\ v_n \end{pmatrix}, \quad \boldsymbol{f}_n = \begin{pmatrix} v_n \\ t_n^2 - u_n \end{pmatrix}, \quad \boldsymbol{f}_n' = \begin{pmatrix} v_n' \\ 2t_n - u_n' \end{pmatrix}. \tag{58}$$

▲

Example 2.4 (TS(3) for a system of ODEs) Let's consider the initial value problem for the system of ODEs

$$\begin{cases} u' = v, \\ v' = t^2 - u. \end{cases} \tag{59}$$

with the initial conditions:

$$u(t_0) = 2, \quad v(t_0) = 3. \tag{60}$$

Then, the TS(3) method for IVP (59)-(60) reads

$$\boldsymbol{x}_{n+1} = \boldsymbol{x}_n + h\boldsymbol{f}_n + \frac{h^2}{2!}\boldsymbol{f}_n' + \frac{h^3}{3!}\boldsymbol{f}_n'', \quad \boldsymbol{x}_n = \begin{pmatrix} u_n \\ v_n \end{pmatrix}, \quad \boldsymbol{f}_n = \begin{pmatrix} v_n \\ t_n^2 - u_n \end{pmatrix}, \quad \boldsymbol{f}_n' = \begin{pmatrix} v_n' \\ 2t_n - u_n' \end{pmatrix}, \quad \boldsymbol{f}_n'' = \begin{pmatrix} v_n'' \\ 2 - u_n'' \end{pmatrix} \tag{61}$$

where $v_n'' = 2t_n - v_n$, $u_n'' = t_n^2 - u_n$. ▲

In order to prove the convergence of a TS method, we will take the same steps as we did for the Euler method. For example, let us prove the convergence of the TS(2) method.

Theorem 2.2 (Convergence of the TS(2) method for the general IVP) *The TS(2) method for the IVP*

$$x' = f(t,x), \quad x(t_0) = \alpha, \quad t \in [0, t_N], \tag{62}$$

with a Lipschitz continuous functions $f(t,x)$ and $f'(t,x)$, and $f \in C^2$ converges, and the global error satisfies:

$$|e_N| \le Bh^2, \quad B = const > 0. \tag{63}$$

Proof. The TS(2) method for IVP (62) is given by

$$x_{n+1} = x_n + hf_n + \frac{h^2}{2} f'_n, \quad x_0 = \alpha, \quad t_i := ih, \quad i = 0, 1, 2, \ldots, N. \tag{64}$$

Taylor expand the exact solution of (62) up to the third order

$$\begin{aligned} x(t_{n+1}) =& x(t_n) + hx'(t_n) + \frac{h^2}{2} x'''(t_n) \qquad &+R \\ =& x(t_n) + hf(t_n, x(t_n)) + \frac{h^2}{2} f''(t_n) \quad &+R, \end{aligned} \tag{65}$$

where $R = \mathcal{O}(h^3)$. The global error is given:

$$\begin{aligned} e_{n+1} =& x(t_{n+1}) - x_{n+1} \\ =& x(t_n) - x_n \qquad + h(f(t_n, x(t_n)) - f_n) + \frac{h^2}{2}(f'(t_n, x(t_n)) - f'_n) + R \\ =& e_n \qquad\qquad + h(f(t_n, x(t_n)) - f_n) + \frac{h^2}{2}(f'(t_n, x(t_n)) - f'_n) + R. \end{aligned} \tag{66}$$

As in Theorem 1.2, equation (66) shows us how the global error at time t_{n+1} depends on the propagated error

$$e_n + h(f(t_n, x(t_n)) - f_n) + \frac{h^2}{2}(f'(t_n, x(t_n)) - f'_n)$$

and the local truncation error given by the reminder R. The only difference with the analogous equation in Theorem 1.2 is the term $\frac{h^2}{2}(f'(t_n, x(t_n)) - f'_n)$. To proceed with this equation in the same manner as we did in Theorem 1.2, we take the absolute value of (66), and estimate $|e_{n+1}|$.

$$\begin{aligned} |e_{n+1}| =& |e_n + h(f(t_n, x(t_n)) - f_n) + R| \\ \leq& |e_n| + h|f(t_n, x(t_n)) - f_n| + \frac{h^2}{2}|f'(t_n, x(t_n)) - f'_n| + |R| \\ &\text{using the Lipschitz continuity of } f \text{ we have } |f(t_n, x(t_n)) - f_n| \leq L|x(t_n) - x_n|,\ L = const \geq 0, \\ &\text{and the Lipschitz continuity of } f' \text{ we have } |f'(t_n, x(t_n)) - f'_n| \leq L^2|x(t_n) - x_n|, \\ \leq& (1 + \hat{h} + \frac{\hat{h}^2}{2})|e_n| + |R|, \quad \hat{h} := hL. \end{aligned} \tag{67}$$

For $n = 0, 1, 2, \ldots, N-1$ we find that

$$\begin{aligned} |e_0| =& x(t_0) - x_0 = 0, \\ |e_1| \leq& |R_1|, \\ |e_2| \leq& (1 + \hat{h} + \frac{\hat{h}^2}{2})|e_1| + |R_2| = (1 + \hat{h} + \frac{\hat{h}^2}{2})|R_1| + |R_2|, \\ |e_3| \leq& (1 + \hat{h} + \frac{\hat{h}^2}{2})|e_2| + |R_3| = (1 + \hat{h} + \frac{\hat{h}^2}{2})^2|R_1| + (1 + \hat{h} + \frac{\hat{h}^2}{2})|R_2| + |R_3|, \\ &\ldots \\ |e_N| \leq& \sum_{i=1}^{N} (1 + \hat{h} + \frac{\hat{h}^2}{2})^{N-i}|R_i|\,. \end{aligned} \tag{68}$$

The error in the initial condition is zero, since the initial condition is known exactly.

In order to estimate $|e_N|$, we use the following inequality

$$|1+\hat{h}+\frac{\hat{h}^2}{2}| \leq e^{|\hat{h}|}. \tag{69}$$

Exponentiating (69) to the power of $(N-i)$ gives

$$|1+\hat{h}+\frac{\hat{h}^2}{2}|^{N-i} \leq e^{|\hat{h}|(N-i)}. \tag{70}$$

Acting in the same manner as in Theorem 1.2, we find that

$$\begin{aligned} |e_N| \leq & \sum_{i=1}^{N} e^{Lt_N}|R_i| \\ \leq & Ne^{Lt_N}Ch^3, \text{ since}^1\ |R_i| \leq Ch^3, \quad C = const > 0 \\ \leq & e^{Lt_N}Ch^2 t_N, \text{ since } t_N = Nh. \end{aligned} \tag{71}$$

Finally, we have

$$|e_N| \leq Bh^2, \quad B := e^{Lt_N}Ct_N. \tag{72}$$

Thus, we have proved that the TS(2) method converges to the exact solution with order $p = 2$. ■

Using the same approach, one can prove the convergence of a TS(n) method for any given n. From what we have seen so far, we can easily come to the conclusion that TS methods can dramatically increase the accuracy of the solution and, more importantly, can do it in a systematic way. Indeed, this is the case! However, these methods are not widely used in practice, because computing higher-order derivatives of the right hand side can result in very complicated expressions (especially for systems of ODEs) that are very difficult to manage. In the following lectures, we will study methods which allow to compute the numerical solution with high-accuracy while avoiding computing higher-order derivatives.

Lecture 3 Linear multistep methods

We will consider the family of linear multistep methods (LMMs), which give the same or higher accuracy as TS methods but at lower computational complexity (the number of elementary operations required to solve a given problem) by using available values of $x(t)$ and $x'(t)$ computed at the previous s time steps.

For simplicity, let us recall the derivation of the TS(2) method for the IVP

$$x' = f(t,x), \quad x(t_0) = \alpha, \quad t \in [t_0, t_N]. \tag{73}$$

We start from the Taylor expansion of $x(t)$ up to order 2:

$$x(t+h) = x(t) + hx'(t) + \frac{h^2}{2}x''(t) + \mathcal{O}(h^3). \tag{74}$$

If we truncate $\mathcal{O}(h^3)$-terms and substitute $x'(t)$ and $x''(t)$ with $f(t,x)$ and $f'(t,x)$, respectively, we will get the TS(2) method. As you can see, this substitution requires the differentiation of the right hand side. This is the very moment when LMMs come into play. The idea behind LMMs is to avoid exact calculations of higher-order derivatives (more than one) and to use approximations to higher-order derivatives instead.

3.1 The general form of linear multistep methods

The general form of a linear s-step method is given by

$$\sum_{m=0}^{s} \alpha_m x_{n+m} = h \sum_{m=0}^{s} \beta_m f(t_{n+m}, x_{n+m}), \quad n = 0, 1, 2, \ldots \tag{75}$$

where $\alpha_m, \beta_m = const \in \mathbb{R}$, and we take $\alpha_s = 1$ as a normalizing condition.

Definition 3.1 (Explicit/Implicit LMM) *An LMM is said to be explicit if $\beta_s = 0$ and implicit if $\beta_s \neq 0$.*

We are not going to discuss how to derive this general form, since it will become clear when we will have gain more experience with LMM. I would like to draw your attention to the fact that LMMs with $s > 1$ are not self-starting, since they require $s-1$ starting values.

How would you compute the starting values for these LMMs?

3.2 Convergence of linear multistep methods

The definition of convergence (1.3) should be adapted to be used with LMMs to accommodate the starting values of the LMM. Let us consider the IVP

$$x' = f(t,x), \quad x(t_0) = \alpha, \quad t \in [t_0, t_N]. \tag{76}$$

Then convergence for LMMs can be defined as follows.

Definition 3.2 (Convergence for LMMs) *An LMM with starting values $\{x_m\}_{m\in[0,s-1]}$ satisfying $|x(t_0+mh) - x_m| \to 0$ for $m \in [0, s-1]$ is said to converge to the exact solution $x(t)$ of IVP* (76) *at time $t = t_n$ if the global error*

$$|e_n| = |x(t_n) - x_n| \to 0 \text{ as } h \to 0. \tag{77}$$

Now, let us switch to more practical questions and consider LMMs based on different approximations of $x''(t)$. Our first method is the Trapezoidal rule.

3.3 The Trapezoidal rule

In order to approximate $x''(t)$ we differentiate the Taylor expansion (74)

$$x'(t+h) = x'(t) + hx''(t) + \frac{h^2}{2}x'''(t) + \mathcal{O}(h^3), \tag{78}$$

and isolate $hx''(t)$

$$hx''(t) = x'(t+h) - x'(t) + \mathcal{O}(h^2). \tag{79}$$

Note that the sign of $\mathcal{O}(h^2)$-term is insignificant. Also note that the ratio

$$x''(t) \approx \frac{x'(t+h) - x'(t)}{h}. \tag{80}$$

is called the forward difference, and approximate the function x'' with the first order of accuracy. Substituting $hx''(t)$ into (74) gives

$$\begin{aligned} x(t+h) =& x(t) + hx'(t) + \frac{h}{2}(x'(t+h) - x'(t)) + \mathcal{O}(h^3) \\ =& x(t) + \frac{h}{2}(x'(t+h) + x'(t)) + \mathcal{O}(h^3). \end{aligned} \tag{81}$$

Now, we can use the differential equation (73) to replace $x'(t)$ with $f(t,x)$:

$$x(t+h) = x(t) + \frac{h}{2}(f(t+h, x(t+h)) + f(t, x(t))) + \mathcal{O}(h^3). \tag{82}$$

Evaluating (82) at $t = t_n$ and truncating $\mathcal{O}(h^3)$-terms yields

$$\text{The Trapezoidal rule:} \qquad x_{n+1} = x_n + \frac{h}{2}(f_{n+1} + f_n), \tag{83}$$

where $f_n := f(t_n, x_n)$.
Unlike the TS methods, we do not have an explicit expression for x_{n+1} in terms of data from the previous time step, therefore the Trapezoidal rule is an implicit method.

Example 3.1 (The Trapezoidal rule for a scalar ODE) Let's consider the initial value problem for the scalar equation

$$x' = (1-2t)x, \quad x(0) = 1, \quad t \in [0,1]. \tag{84}$$

The Trapezoidal rule for (84) is

$$x_{n+1} = x_n + \frac{h}{2}((1-2t_{n+1})x_{n+1} + (1-2t_n)x_n). \tag{85}$$

▲

Example 3.2 (The Trapezoidal rule for a system of ODEs) Let's consider the initial value problem for the system of ODEs

$$\begin{cases} u' = v, \\ v' = t^2 - u. \end{cases} \tag{86}$$

with the initial conditions:
$$u(t_0) = 2, \quad v(t_0) = 3. \tag{87}$$
Then, the Trapezoidal rule for IVP (86)-(87) reads
$$\boldsymbol{x}_{n+1} = \boldsymbol{x}_n + \frac{h}{2}(\boldsymbol{f}_{n+1} + \boldsymbol{f}_n), \quad \boldsymbol{x}_n = \begin{pmatrix} u_n \\ v_n \end{pmatrix}, \quad \boldsymbol{f}_{n+1} = \begin{pmatrix} v_{n+1} \\ t_{n+1}^2 - u_{n+1} \end{pmatrix}, \quad \boldsymbol{f}_n = \begin{pmatrix} v_n \\ t_n^2 - u_n \end{pmatrix}, \tag{88}$$

▲

3.4 The local truncation error of the Trapezoidal rule

To calculate the local truncation error of the Trapezoidal rule, we act as before with the only exception. Namely, we roll back to the equation
$$x(t+h) = x(t) + \frac{h}{2}(x'(t+h) + x'(t))) \tag{89}$$
and Taylor expand $x'(t+h)$. Here, the idea is to get rid off the implicit dependence on the right hand side. For this, we Taylor expand $x(t+h)$ and then differentiate the Taylor series with respect to t as in (78):
$$x'(t+h) = x'(t) + hx''(t) + \frac{h^2}{2}x'''(t) + \mathcal{O}(h^3). \tag{90}$$
It is repetitive, but it is better to get this expression right at hand. Substitution of (90) into (89) results in
$$\begin{aligned} x(t+h) =& x(t) + \frac{h}{2}(x'(t) + hx''(t) + \frac{h^2}{2}x'''(t) + x'(t)) + \mathcal{O}(h^4) \\ =& x(t) + hx'(t) + \frac{h^2}{2}x''(t) + \frac{h^3}{4}x'''(t) + \mathcal{O}(h^4), \end{aligned} \tag{91}$$
or
$$x_{n+1} = x_n + hx'_n + \frac{h^2}{2}x''_n + \frac{h^3}{4}x'''_n + \mathcal{O}(h^4). \tag{92}$$
The Taylor expansion of $x(t)$ is given by
$$x(t+h) = x(t) + hx'(t) + \frac{h^2}{2}x''(t) + \frac{h^3}{6}x'''(t) + \mathcal{O}(h^4). \tag{93}$$
To compute the local truncation error of the Trapezoidal rule, we just subtract x_{n+1} from $x(t+h)$:
$$x(t+h) - x_{n+1} = \mathcal{O}(h^3). \tag{94}$$

3.5 The global error of the Trapezoidal rule

Theorem 3.1 (Convergence of the Trapezoidal rule for the general IVP) *The Trapezoidal rule for the IVP*
$$x' = f(t,x), \quad x(t_0) = \alpha, \quad t \in [0, t_N], \tag{95}$$
with a Lipschitz continuous functions $f(t,x)$, and $f \in C^2$ converges, and the global error satisfies:
$$|e_N| \leq Bh^2, \quad B = const > 0. \tag{96}$$

Proof. The Trapezoidal rule for IVP (95) is given by

$$x_{n+1} = x_n + \frac{h}{2}(f_{n+1} + f_n), \quad x_0 = \alpha, \quad t_i := ih, \quad i = 0, 1, 2, \ldots, N. \tag{97}$$

Taylor expand the exact solution of (95) up to the third order using the approximation of the second order derivative (as we did for the derivation of the Trapezoidal rule):

$$x(t + h) = x(t) + \frac{h}{2}(f(t + h, x(t + h)) + f(t, x(t))) + R. \tag{98}$$

where $R = \mathcal{O}(h^3)$. The global error is given:

$$\begin{aligned}
e_{n+1} =& x(t_{n+1}) - x_{n+1} \\
=& x(t_n) - x_n \qquad + \frac{h}{2}(f(t_{n+1}, x(t_{n+1})) - f_{n+1}) + \frac{h}{2}(f(t_n, x(t_n)) - f_n) + R \\
=& e_n \qquad + \frac{h}{2}(f(t_{n+1}, x(t_{n+1})) - f_{n+1}) + \frac{h}{2}(f(t_n, x(t_n)) - f_n) + R.
\end{aligned} \tag{99}$$

As in Theorem 2.2, equation (99) shows us how the global error at time t_{n+1} depends on the propagated error

$$e_n + \frac{h}{2}(f(t_{n+1}, x(t_{n+1})) - f_{n+1}) + \frac{h}{2}(f(t_n, x(t_n)) - f_n)$$

and the local truncation error given by the reminder R. To proceed, we take the absolute value of (99), and estimate $|e_{n+1}|$.

$$\begin{aligned}
|e_{n+1}| =& |e_n + \frac{h}{2}(f(t_{n+1}, x(t_{n+1})) - f_{n+1}) + \frac{h}{2}(f(t_n, x(t_n)) - f_n) + R| \\
\le& |e_n| + \frac{h}{2}|f(t_{n+1}, x(t_{n+1})) - f_{n+1}| + \frac{h}{2}|f(t_n, x(t_n)) - f_n| + |R| \\
& \text{using the Lipschitz continuity of } f \text{ we have } |f(t_n, x(t_n)) - f_n| \le L|x(t_n) - x_n|,\ L = const \ge 0, \\
\le& |e_n| + \frac{\hat{h}}{2}|x(t_{n+1}) - x_{n+1}| + \frac{\hat{h}}{2}|x(t_n) - x_n| + |R|, \quad \hat{h} := hL, \\
\le& (1 + \frac{\hat{h}}{2})|e_n| + \frac{\hat{h}}{2}|e_{n+1}| + |R| \\
\le& \frac{1 + \frac{\hat{h}}{2}}{1 - \frac{\hat{h}}{2}}|e_n| + \frac{|R|}{1 - \frac{\hat{h}}{2}} \\
=& A|e_n| + |R|S^{-1}, \quad A := (1 + \frac{\hat{h}}{2})S^{-1}, \quad S := 1 - \frac{\hat{h}}{2}.
\end{aligned} \tag{100}$$

For $n = 0, 1, 2, \ldots, N-1$ we find that

$$
\begin{aligned}
|e_0| =& x(t_0) - x_0 = 0, \\
|e_1| \leq& |R_1|S^{-1}, \\
|e_2| \leq& A|e_1| + |R_2|S^{-1} = (A|R_1| + |R_2|)S^{-1}, \\
|e_3| \leq& A|e_2| + |R_3|S^{-1} = (A^2|R_1| + A|R_2| + |R_3|)S^{-1}, \\
&\ldots \\
|e_N| \leq& \sum_{i=1}^{N} A^{N-i}|R_i|S^{-1}.
\end{aligned}
\tag{101}
$$

The error in the initial condition is zero, since the initial condition is known exactly.
Since[1] $|R_i| \leq Ch^3$, $C = const > 0$, we can estimate (101) as

$$
\begin{aligned}
|e_N| &\leq \sum_{i=1}^{N} A^{N-i}|R_i|S^{-1} \\
&\leq \sum_{i=1}^{N} A^{N-i}Ch^3S^{-1}.
\end{aligned}
\tag{102}
$$

Using the geometric series

$$1 + A + A^2 + \ldots + A^{N-1} = \frac{A^N - 1}{A - 1} \tag{103}$$

in (102) we find

$$
\begin{aligned}
|e_N| \leq& \frac{A^N - 1}{A - 1} Ch^3 S^{-1} \\
=& \frac{A^N - 1}{L} Ch^2.
\end{aligned}
\tag{104}
$$

Then we rewrite A as

$$A := (1 + \frac{\hat{h}}{2})S^{-1} = 1 + \hat{h}S^{-1}, \tag{105}$$

and use the following inequality

$$1 + \hat{h}S^{-1} \leq e^{\hat{h}S^{-1}}. \tag{106}$$

Exponentiating (106) to the power of N gives

$$(1 + \hat{h}S^{-1})^N \leq e^{N\hat{h}S^{-1}}. \tag{107}$$

Using (107) in (104) leads to

$$
\begin{aligned}
|e_N| \leq& \frac{e^{N\hat{h}S^{-1}} - 1}{L} Ch^2 \\
\leq& Bh^2,
\end{aligned}
\tag{108}
$$

where $B := C(e^{Lt_N S^{-1}} - 1)/L$.
Thus, we have proved that the Trapezoidal rule converges to the exact solution with order $p = 2$.
■

3.6 How to develop the most accurate one-step method

Before reading further, give yourself a minute to think about how would you approach the task of developing the most accurate one-step method.

The general roadmap of the development can look as follows.

1. **Fix the general form of the one-step method.**
   For this, just use $s = 1$ in (75). It gives

$$\alpha_0 x_n + \alpha_1 x_{n+1} = h(\beta_1 f_{n+1} + \beta_0 f_n). \tag{109}$$

   By definition, $\alpha_1 = 1$, and therefore we have

$$x_{n+1} = \alpha_0 x_n + h(\beta_1 f_{n+1} + \beta_0 f_n). \tag{110}$$

   The sign in front of α_0 is immaterial, since α_0 has to be defined from the conditions described below.

   We have the method! The only snag is that it has three unknown coefficients α_0, β_0, and β_1. How to find them? Well, since we want the most accurate method then we have to choose these coefficients so that they minimize the local truncation. For doing so, we first eliminate the implicit dependence on the solution in the right hand side of (110) and then find the local truncation error in the usual way. Let us write (110) in the continuous form, namely

$$x(t_n + h) = \alpha_0 x(t_n) + h(\beta_1 f(t_n + h, x(t_n + h)) + \beta_0 f(t_n, x(t_n))), \tag{111}$$

   Using the ODE $x' = f(t, x)$ in (111) gives

$$x(t_n + h) = \alpha_0 x(t_n) + h(\beta_1 x'(t_n + h) + \beta_0 x'(t_n)). \tag{112}$$

   Now, we Taylor expand $x'(t_n + h)$

$$x'(t_n + h) = x'(t_n) + hx''(t_n) + \frac{h^2}{2}x'''(t_n) + \mathcal{O}(h^3). \tag{113}$$

   and plug it back into (112):

$$x(t_n + h) = \alpha_0 x(t_n) + h(\beta_1 (x'(t_n) + hx''(t_n) + \frac{h^2}{2}x'''(t_n)) + \beta_0 x'(t_n)) + \mathcal{O}(h^4). \tag{114}$$

2. **Find the minimum local truncation error between the exact solution and its approximation.**
   The Taylor expansion of the exact solution is given by

$$x(t + h) = x(t) + hx'(t) + \frac{h^2}{2}x''(t) + \frac{h^3}{6}x'''(t) + \mathcal{O}(h^4). \tag{115}$$

   Since $x(t) \equiv x(t_n)$, the difference is

$$x(t+h) - x(t_n+h) = (1-\alpha_0)x(t) + h(1-\beta_1-\beta_0)x'(t) + h^2(\frac{1}{2}-\beta_1)x''(t) + h^3(\frac{1}{6}-\frac{\beta_1}{2})x''' + \mathcal{O}(h^4). \tag{116}$$

   To get the most accurate one-step method, we have to find α_0, β_0, and β_1 which deliver the minimum to (116). In this case, it happens to be $\alpha_0 = 1$, $\beta_0 = \beta_1 = \frac{1}{2}$. Thus, **the most**

accurate one-step method is

$$x_{n+1} = x_n + \frac{h}{2}(f_{n+1} + f_n). \tag{117}$$

This is the Trapezoidal rule!

If you want to find the most accurate s-step method you can use the same approach:
(1) fix the general form of the method;
(2) find unknown coefficients $\{\alpha_m\}_{m\in[0,s-1]}, \alpha_s = 1$ and $\{\beta_m\}_{m\in[0,s]}$ which minimize the local truncation.

This simple idea gives you a powerful tool for development of your own LMMs with a given order of accuracy. We will return to this discussion later in the context of linear multi-step methods.

Lecture 4 The 2-step Adams-Bashforth method, AB(2)

Now, let us shift our focus on 2-step methods. We start from the 2-step Adams-Bashforth method. The derivation of the 2-step Adams-Bashforth method is similar to the derivation of the Trapezoidal. First, we Taylor expand

$$x(t+h) = x(t) + hx'(t) + \frac{h^2}{2}x''(t) + \mathcal{O}(h^3), \tag{118}$$

and instead of using the exact value of $x''(t)$, we approximate it. For the Trapezoidal rule we isolated $hx''(t)$ from

$$x'(t+h) = x'(t) + hx''(t) + \frac{h^2}{2}x'''(t) + \mathcal{O}(h^3), \tag{119}$$

For the AB(2) method, we find $hx''(t)$ from

$$x'(t-h) = x'(t) - hx''(t) + \frac{h^2}{2}x'''(t) + \mathcal{O}(h^3). \tag{120}$$

This gives

$$hx''(t) = x'(t) - x'(t-h) + \mathcal{O}(h^2). \tag{121}$$

Note that the ratio

$$x''(t) \approx \frac{x'(t) - x'(t-h)}{h}. \tag{122}$$

is called the backward difference, and approximate the function x'' with the first order of accuracy. Substituting $hx''(t)$ into (118) gives

$$\begin{aligned} x(t+h) =& x(t) + hx'(t) + \frac{h}{2}(x'(t) - x'(t-h)) + \mathcal{O}(h^3) \\ =& x(t) + \frac{h}{2}(3x'(t) - x'(t-h)) + \mathcal{O}(h^3). \end{aligned} \tag{123}$$

Using the differential equation $x' = f(t,x)$ in (123) to replace $x'(t)$ with $f(t,x)$ results in

$$x(t+h) = x(t) + \frac{h}{2}(3f(t,x(t)) - f(t-h,x(t-h))) + \mathcal{O}(h^3). \tag{124}$$

Evaluating (124) at $t = t_n$ and truncating $\mathcal{O}(h^3)$-terms yields

$$\text{The 2-step Adams-Bashforth method:} \qquad x_{n+1} = x_n + \frac{h}{2}(3f_n - f_{n-1}), \tag{125}$$

where $f_n := f(t_n, x_n)$. Note that (125) is a 2-step method, since it uses the solution at two previous time steps (x_{n-1} and x_n) to compute x_{n+1}. Therefore, in order to start the AB(2) method one can use a 1-step method. Let us consider how to use the AB(2) method together with the Euler method.

Example 4.1 (The AB(2) method and Euler's method) *Let's consider the initial value problem for the system of equations*

$$\boldsymbol{x}' = \boldsymbol{f}(t,\boldsymbol{x}), \quad \boldsymbol{x}(0) = \boldsymbol{\alpha}, \quad t \in [t_0, t_N]. \tag{126}$$

The AB(2) method for IVP (126) *reads*

$$\boldsymbol{x}_{n+1} = \boldsymbol{x}_n + \frac{h}{2}(3\boldsymbol{f}_n - \boldsymbol{f}_{n-1}). \tag{127}$$

In order to use the Euler method together with the AB(2) method we shift the indices in (127) *to the right:*

$$\boldsymbol{x}_{n+2} = \boldsymbol{x}_{n+1} + \frac{h}{2}(3\boldsymbol{f}_{n+1} - \boldsymbol{f}_n). \tag{128}$$

Now, we can use the initial condition to compute $\boldsymbol{f}_0(t, \boldsymbol{x}_0)$ and apply the Euler method only once at time t_0 to find $\boldsymbol{x}_1$ and then use it to compute $\boldsymbol{f}_1(t, \boldsymbol{x}_1)$ on the right hand side of (128)*. Having computed $\boldsymbol{f}_0$, $\boldsymbol{f}_1$, and $\boldsymbol{x}_1$, we then compute $\boldsymbol{x}_2$ using the AB(2) method. To compute the next value, $\boldsymbol{x}_3$, we do not need Euler's method anymore, since previous values $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ have already been computed.* ▲

Example 4.2 (The AB(2) method for a scalar ODE) Let's consider the initial value problem for the scalar equation

$$x' = (1-2t)x, \quad x(0) = 1, \quad t \in [0, 1]. \tag{129}$$

The AB(2) method for (129) is

$$x_{n+1} = x_n + \frac{h}{2}(3(1-2t_n)x_n - (1-2t_{n-1})x_{n-1}). \tag{130}$$

▲

Example 4.3 (The AB(2) method for a system of ODEs) Let's consider the initial value problem for the system of ODEs

$$\begin{cases} u' = v, \\ v' = t^2 - u. \end{cases} \tag{131}$$

with the initial conditions:

$$u(t_0) = 2, \quad v(t_0) = 3. \tag{132}$$

Then, the AB(2) method for IVP (131)-(132) reads

$$\boldsymbol{x}_{n+1} = \boldsymbol{x}_n + \frac{h}{2}(3\boldsymbol{f}_n - \boldsymbol{f}_{n-1}), \quad \boldsymbol{x}_n = \begin{pmatrix} u_n \\ v_n \end{pmatrix}, \quad \boldsymbol{f}_n = \begin{pmatrix} v_n \\ t_n^2 - u_n \end{pmatrix}, \quad \boldsymbol{f}_{n-1} = \begin{pmatrix} v_{n-1} \\ t_{n-1}^2 - u_{n-1} \end{pmatrix}, \tag{133}$$

▲

Using Taylor's series is not the only way of derivation of the 2-step Adams-Bashforth method. It also concerns other methods considered in this course. Let us study an alternative derivation of the AB(2) method.

4.1 Derivation of the AB(2) method via Lagrange polynomials

Consider the following IVP

$$x' = f(t, x), \quad x(t_0) = \alpha, \quad t \in [t_0, t_N]. \tag{134}$$

The integration of (143) over the interval $[t_n, t_{n+1}]$ gives

$$\int_{t_n}^{t_{n+1}} x' dt = \int_{t_n}^{t_{n+1}} f(t,x)dt \quad \Rightarrow \quad x(t_{n+1}) = x(t_n) + \int_{t_n}^{t_{n+1}} f(t,x)dt\,. \tag{135}$$

To proceed with the numerical solution, we have to compute the integral of $f(t,x)$. For this, we can approximate $f(t,x)$ with Lagrange polynomials and then do the integration. Let us take the Lagrange polynomial

$$\mathcal{L}(t) := \sum_{j=0}^{k} f_j \ell_j(t), \quad \ell_j(t) := \prod_{\substack{0\le m\le k\\ m\neq j}} \frac{t-t_m}{t_j-t_m}. \tag{136}$$

To derive the AB(2) method, we take the Lagrange polynomial through two points t_{n-1} and t_n (in this case $k=1$):

$$\mathcal{L}(t) := \frac{t-t_n}{t_{n-1}-t_n} f_{n-1} + \frac{t-t_{n-1}}{t_n-t_{n-1}} f_n. \tag{137}$$

Thus, we can approximate the integral in (135) as follows:

$$\begin{aligned}
\int_{t_n}^{t_{n+1}} f(t,x)dt &\approx \int_{t_n}^{t_{n+1}} \mathcal{L}(t)dt \\
&= \int_{t_n}^{t_{n+1}} \left(\frac{t-t_n}{t_{n-1}-t_n} f_{n-1} + \frac{t-t_{n-1}}{t_n-t_{n-1}} f_n \right) dt \\
&= \left[\frac{t(t-2t_n)}{2(t_{n-1}-t_n)} f_{n-1} + \frac{t(t-2t_{n-1})}{2(t_n-t_{n-1})} f_n \right]_{t_n}^{t_{n+1}} \\
&\quad \text{taking into account that } h = t_n - t_{n-1} \text{ we have} \\
&= \left[\frac{t(t-2t_{n-1})}{2h} f_n - \frac{t(t-2t_n)}{2h} f_{n-1} \right]_{t_n}^{t_{n+1}} \\
&= \frac{t_{n+1}(t_{n+1}-2t_{n-1}) - t_n(t_n-2t_{n-1})}{2h} f_n - \frac{t_{n+1}(t_{n+1}-2t_n) - t_n(t_n-2t_n)}{2h} f_{n-1} \\
&\quad \text{since } t_{n-1} = t_n - h, \quad t_{n+1} = t_n + h \text{ we have} \\
&= \frac{(t_n+h)(3h-t_n) - t_n(2h-t_n)}{2h} f_n - \frac{(t_n+h)(h-t_n) + t_n^2}{2h} f_{n-1} \\
&= \frac{h}{2}(3f_n - f_{n-1}).
\end{aligned} \tag{138}$$

Using (138) in (135) gives us the 2-step Adams-Bashforth method.

4.2 The local truncation error of the AB(2) method

To compute the local truncation error of the AB(2) method we first rewrite (125) as

$$x(t_n+h) = x(t_n) + \frac{h}{2}(3x'(t_n) - x'(t_n-h)), \tag{139}$$

and then plug the Taylor expansion of x'

$$x'(t_n - h) = x'(t_n) - hx''(t_n) + \frac{h^2}{2}x'''(t_n) + \mathcal{O}(h^3) \tag{140}$$

into (139)

$$\begin{aligned} x(t_n + h) =& x(t_n) + \frac{h}{2}(3x'(t_n) - (x'(t_n) - hx''(t_n) + \frac{h^2}{2}x'''(t_n))) + \mathcal{O}(h^4) \\ =& x(t_n) + hx'(t_n) + \frac{h^2}{2}x''(t_n) - \frac{h^3}{4}x'''(t_n) + \mathcal{O}(h^4). \end{aligned} \tag{141}$$

Then, we find the local truncation error, i.e. the difference between the Taylor expansion of the exact solution (118) and its approximation (141):

$$x(t + h) - x(t_n + h) = \mathcal{O}(h^3). \tag{142}$$

4.3 The global error of the AB(2) method

Essentially, the proof of the next theorem follows the same idea as the proof of all theorems considered so far.

Theorem 4.1 (Convergence of the AB(2) method for the general IVP) *The 2-step Adams-Bashforth method for the IVP*

$$x' = f(t, x), \quad x(t_0) = \alpha, \quad t \in [0, t_N], \tag{143}$$

with a Lipschitz continuous functions $f(t, x)$, and $f \in C^2$ converges, and the global error satisfies:

$$\max_{n \in [0,N]} |x(t_n) - x_n| \leq Bh^2, \quad B = const > 0. \tag{144}$$

Proof. The AB(2) method for IVP (143) is given by

$$x_{n+1} = x_n + \frac{h}{2}(3f_n - f_{n-1}), \quad x_0 = \alpha, \quad t_i := ih, \quad i = 0, 1, 2, \ldots, N. \tag{145}$$

We have already seen in the analysis of the local truncation error of the AB(2) method that

$$\begin{aligned} e_{n+1} =& x(t_{n+1}) - x_{n+1} \\ =& x(t_n) - x_n \qquad + \frac{3h}{2}(f(t_n, x(t_n)) - f_n) - \frac{h}{2}(f(t_{n-1}, x(t_{n-1})) - f_{n-1}) + R \\ =& e_n \qquad + \frac{3h}{2}(f(t_n, x(t_n)) - f_n) - \frac{h}{2}(f(t_{n-1}, x(t_{n-1})) - f_{n-1}) + R, \end{aligned} \tag{146}$$

where $R = \mathcal{O}(h^3)$. As in Theorem 3.1, equation (146) shows us how the global error at time t_{n+1} depends on the propagated error

$$e_n + \frac{3h}{2}(f(t_n, x(t_n)) - f_n) - \frac{h}{2}(f(t_{n-1}, x(t_{n-1})) - f_{n-1})$$

568 and the local truncation error given by the reminder R. Taking the absolute value of (146) yields

$$\begin{aligned}|e_{n+1}| =&|e_n + \frac{3h}{2}(f(t_n, x(t_n)) - f_n) - \frac{h}{2}(f(t_{n-1}, x(t_{n-1})) - f_{n-1}) + R| \\ \le&|e_n| + \frac{3h}{2}|f(t_{n+1}, x(t_{n+1})) - f_{n+1}| + \frac{h}{2}|f(t_{n-1}, x(t_{n-1})) - f_{n-1}| + |R| \\ &\text{using the Lipschitz continuity of } f \text{ we have } |f(t_n, x(t_n)) - f_n| \le L|x(t_n) - x_n|,\ L = const \ge 0, \\ \le&|e_n| + \frac{3\hat{h}}{2}|x(t_n) - x_n| + \frac{\hat{h}}{2}|x(t_{n-1}) - x_{n-1}| + |R|, \quad \hat{h} := hL, \\ \le&(1 + \frac{3\hat{h}}{2})|e_n| + \frac{\hat{h}}{2}|e_{n-1}| + |R|.\end{aligned} \tag{147}$$

569 Introducing the error bounding function $\delta_n = \max\limits_{\substack{0\le i\le n\\ n\in[0,N]}} |e_i|$ and rewriting (147) in terms of δ_n gives

$$\begin{aligned}\delta_{n+1} \le&(1 + \frac{3\hat{h}}{2})\delta_n + \frac{\hat{h}}{2}\delta_n + |R| \\ \le&(1 + 2\hat{h})\delta_n + |R|.\end{aligned} \tag{148}$$

570 For $n = 0, 1, 2, \dots, N-1$ we find that

$$\begin{aligned}\delta_0 =&x(t_0) - x_0 = 0, \\ \delta_1 \le&|R_1|, \\ \delta_2 \le&(1 + 2\hat{h})\delta_1 + |R_2| = (1 + 2\hat{h})|R_1| + |R_2|, \\ \delta_3 \le&(1 + 2\hat{h})\delta_2 + |R_3| = (1 + 2\hat{h})^2|R_1| + (1 + 2\hat{h})|R_2| + |R_3|, \\ \dots& \\ \delta_N \le& \sum_{i=1}^{N}(1 + 2\hat{h})^{N-i}|R_i|\,.\end{aligned} \tag{149}$$

571 In order to estimate $|\delta_N|$, we use the following inequality

$$(1 + 2\hat{h}) \le e^{2\hat{h}}. \tag{150}$$

572 Exponentiating (150) to the power of $(N - i)$ gives

$$(1 + 2\hat{h})^{N-i} \le e^{|2\hat{h}|(N-i)}. \tag{151}$$

573 Using (151) in (149) leads to

$$\begin{aligned}\delta_N \le& \sum_{i=1}^{N} e^{2Lt_N}|R_i| \\ \le&Ne^{2Lt_N}Ch^3, \text{ since[1] } |R_i| \le Ch^3, \quad C = const > 0 \\ \le&e^{2Lt_N}Ch^2t_N, \text{ since } t_N = Nh.\end{aligned} \tag{152}$$

574 Finally, we have

$$\max_{n\in[0,N]} |x(t_n) - x_n| \le Bh^2, \quad B := e^{2Lt_N}Ct_N. \tag{153}$$

575 Thus, we have proved that the AB(2) method converges to the exact solution with order $p = 2$. ■

Lecture 5 The 2-step Adams-Moulton method, AM(2)

As we have already seen, the use of different approximations to the second order derivative leads to different methods. For example, the forward difference gives the Trapezoidal rule, while the backward difference results in the AB(2) method. If we use the central difference for $x''(t)$

$$x''(t) \approx \frac{x'(t+h) - x'(t-h)}{2h} \tag{154}$$

and $x'''(t)$

$$x'''(t) \approx \frac{x'(t+h) - 2x'(t) + x'(t-h)}{h^2} \tag{155}$$

then we will end up with the 2-step Adams-Moulton method. Pay attention that $x'''(t)$ is approximated in terms of $x'(t)$, i.e. we do not need the derivatives higher than one in the AM(2) method. Also note that the central difference is a second order accurate finite difference scheme.

In order to derive the central difference formula, we find the difference between the derivative of the forward Taylor expansion

$$x'(t+h) = x'(t) + hx''(t) + \frac{h^2}{2}x'''(t) + \mathcal{O}(h^3), \tag{156}$$

and backward Taylor expansion

$$x'(t-h) = x'(t) - hx''(t) + \frac{h^2}{2}x'''(t) + \mathcal{O}(h^3). \tag{157}$$

Namely, we have

$$x''(t) = \frac{x'(t+h) - x'(t-h)}{2h} + \mathcal{O}(h^2). \tag{158}$$

The central difference formula for $x'''(t)$ can be computed as a sum of the derivative of the forward and backward Taylor expansions:

$$x'''(t) = \frac{x'(t+h) - 2x'(t) + x'(t-h)}{h^2} + \mathcal{O}(h^2). \tag{159}$$

We can now take the Taylor expansion

$$x(t+h) = x(t) + hx'(t) + \frac{h^2}{2}x''(t) + \frac{h^3}{6}x'''(t) + \mathcal{O}(h^4), \tag{160}$$

and use the central difference approximation for $x''(t)$ and $x'''(t)$:

$$\begin{aligned} x(t+h) =& x(t) + hx'(t) + \frac{h}{4}(x'(t+h) - x'(t-h)) + \frac{h}{6}(x'(t+h) - 2x'(t) + x'(t-h)) + \mathcal{O}(h^4) \\ =& x(t) + \frac{h}{12}(8x'(t) + 5x'(t+h) - x'(t-h)) + \mathcal{O}(h^4). \end{aligned} \tag{161}$$

Using the ODE $x' = f(t, x)$ in (161) we arrive at

The 2-step Adams-Moulton method: $$x_{n+1} = x_n + \frac{h}{12}(8f_n + 5f_{n+1} - f_{n-1}), \tag{162}$$

where $f_n := f(t_n, x_n)$. The AM(2) method is a 2-step implicit method.

5.1 Derivation of the AM(2) method via Lagrange polynomials

As with the AB(2) method, the AM(2) method can also be derived via Lagrange polynomials. Consider the following IVP

$$x' = f(t,x), \quad x(t_0) = \alpha, \quad t \in [t_0, t_N]. \tag{163}$$

The integration of (180) over the interval $[t_n, t_{n+1}]$ gives

$$\int_{t_n}^{t_{n+1}} x' dt = \int_{t_n}^{t_{n+1}} f(t,x) dt \quad \Rightarrow \quad x(t_{n+1}) = x(t_n) + \int_{t_n}^{t_{n+1}} f(t,x) dt\,. \tag{164}$$

We approximate $f(t,x)$ with Lagrange polynomials and then do the integration. Let us take the Lagrange polynomial

$$\mathcal{L}(t) := \sum_{j=0}^{k} f_j \ell_j(t), \quad \ell_j(t) := \prod_{\substack{0 \le m \le k \\ m \ne j}} \frac{t - t_m}{t_j - t_m}. \tag{165}$$

To derive the AM(2) method, we take the Lagrange polynomial through three points t_{n-1}, t_n, and t_{n+1} (in this case $k = 2$):

$$\mathcal{L}(t) := \frac{t - t_n}{t_{n-1} - t_n} \frac{t - t_{n+1}}{t_{n-1} - t_{n+1}} f_{n-1} + \frac{t - t_{n+1}}{t_n - t_{n+1}} \frac{t - t_{n-1}}{t_n - t_{n-1}} f_n + \frac{t - t_{n-1}}{t_{n+1} - t_{n-1}} \frac{t - t_n}{t_{n+1} - t_n} f_{n+1}. \tag{166}$$

Thus, we can approximate the integral in (164) as follows:

$$\int_{t_n}^{t_{n+1}} f(t,x) dt \approx \int_{t_n}^{t_{n+1}} \mathcal{L}(t) dt = \text{ after some calculations } = \frac{h}{12}(8f_n + 5f_{n+1} - f_{n-1}). \tag{167}$$

Using (167) in (164) gives us the 2-step Adams-Moulton method.
Be sure you know how to do the calculations in (167).

Analogously to the AB(2) method, the AM(2) method requires a 1-step method to start the integration. Let us consider several examples showing how to apply the AM(2) method.

Example 5.1 (The AM(2) method and Euler's method) *Let's consider the initial value problem for the system of equations*

$$\boldsymbol{x}' = \boldsymbol{f}(t, \boldsymbol{x}), \quad \boldsymbol{x}(0) = \boldsymbol{\alpha}, \quad t \in [t_0, t_N]. \tag{168}$$

The AM(2) method for IVP (168) *reads*

$$\boldsymbol{x}_{n+1} = \boldsymbol{x}_n + \frac{h}{12}(8\boldsymbol{f}_n + 5\boldsymbol{f}_{n+1} - \boldsymbol{f}_{n-1}). \tag{169}$$

In order to use the Euler method together with the AM(2) method we shift the indices in (169) *to the right as we did it for the AB(2) method:*

$$\boldsymbol{x}_{n+2} = \boldsymbol{x}_{n+1} + \frac{h}{12}(8\boldsymbol{f}_{n+1} + 5\boldsymbol{f}_{n+2} - \boldsymbol{f}_n). \tag{170}$$

Now, we can use the initial condition to compute $\boldsymbol{f}_0(t, \boldsymbol{x}_0)$ and apply the Euler method only once at time t_0 to find $\boldsymbol{x}_1$ and then use it to compute $\boldsymbol{f}_1(t, \boldsymbol{x}_1)$ on the right hand side of (128)*. Having* *computed $\boldsymbol{f}_0$, $\boldsymbol{f}_1$, and $\boldsymbol{x}_1$, we then compute $\boldsymbol{x}_2$ using the AM(2) method. Pay attention that the computation of $\boldsymbol{x}_2$ (as well as $\boldsymbol{x}_3$, $\boldsymbol{x}_4$, etc.) in general case requires to find the solution of a nonlinear system of equations, since the method is implicit. To compute the next value, $\boldsymbol{x}_3$, we do not need Euler's method anymore, since previous values $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ have already been computed.* ▲

Example 5.2 (The AM(2) method for a scalar ODE) Let's consider the initial value problem for the scalar equation

$$x' = (1-2t)x, \quad x(0) = 1, \quad t \in [0,1]. \tag{171}$$

The AM(2) method for (171) is

$$x_{n+1} = x_n + \frac{h}{12}(8(1-2t_n)x_n + 5(1-2t_{n+1})x_{n+1} - (1-2t_{n-1})x_{n-1}). \tag{172}$$

Note that for linear equations as the one considered here, you can explicitly compute x_{n+1}:

$$x_{n+1} = \frac{1}{1-\frac{5h}{12}(1-2t_{n+1})}(x_n + \frac{h}{12}(8(1-2t_n)x_n - (1-2t_{n-1})x_{n-1})). \tag{173}$$

▲

Example 5.3 (The AM(2) method for a system of ODEs) Let's consider the initial value problem for the system of ODEs

$$\begin{cases} u' = v, \\ v' = t^2 - u. \end{cases} \tag{174}$$

with the initial conditions:

$$u(t_0) = 2, \quad v(t_0) = 3. \tag{175}$$

Then, the AM(2) method for IVP (174)-(175) reads

$$\boldsymbol{x}_{n+1} = \boldsymbol{x}_n + \frac{h}{12}(8\boldsymbol{f}_n + 5\boldsymbol{f}_{n+1} - \boldsymbol{f}_{n-1}), \tag{176}$$

where

$$\boldsymbol{x}_n = \begin{pmatrix} u_n \\ v_n \end{pmatrix}, \quad \boldsymbol{f}_n = \begin{pmatrix} v_n \\ t_n^2 - u_n \end{pmatrix}, \quad \boldsymbol{f}_{n+1} = \begin{pmatrix} v_{n+1} \\ t_{n+1}^2 - u_{n+1} \end{pmatrix}, \quad \boldsymbol{f}_{n-1} = \begin{pmatrix} v_{n-1} \\ t_{n-1}^2 - u_{n-1} \end{pmatrix}. \tag{177}$$

▲

5.2 The local truncation error of the AM(2) method

I believe you can easily see that the local truncation error of the AM(2) method is of order 4. To show it rigorously, one should plug (156) and (157) into (161), and subtract the result from the Taylor expansion of the exact solution. The substitution of (156) and (157) into (161) gives

$$\begin{aligned} x(t_n+h) &= x(t_n) + \frac{h}{12}(8x'(t_n) + 5(x'(t) + hx''(t) + \frac{h^2}{2}x'''(t) + \frac{h^3}{6}x''''(t)) \\ &\qquad - (x'(t) - hx''(t) + \frac{h^2}{2}x'''(t) - \frac{h^3}{6}x''''(t))) + \mathcal{O}(h^5) \\ &= x(t_n) + \frac{h}{12}(12x'(t_n) + 6hx''(t_n) + 2h^2x'''(t_n) + h^3x''''(t_n)) + \mathcal{O}(h^5) \\ &= x(t_n) + hx'(t_n) + \frac{h^2}{2}x''(t_n) + \frac{h^3}{6}x'''(t_n) + \frac{h^4}{12}x''''(t_n) + \mathcal{O}(h^5) \end{aligned} \tag{178}$$

Here, we use $x(t_n + h)$ notation to distinguish the approximated solution from the Taylor expansion of the exact solution denoted as $x(t+h)$. Now, we find the local truncation error:

$$x(t+h) - x(t_n + h) = \mathcal{O}(h^4). \tag{179}$$

5.3 The global error of the AM(2) method

The proof of convergence of the AM(2) method is very similar to that one of the AB(2) method.

Theorem 5.1 (Convergence of the AM(2) method for the general IVP) *The 2-step Adams-Moulton method for the IVP*

$$x' = f(t,x), \quad x(t_0) = \alpha, \quad t \in [0, t_N], \tag{180}$$

with a Lipschitz continuous functions $f(t,x)$, and $f \in C^3$ converges, and the global error satisfies:

$$\max_{n \in [0,N]} |x(t_n) - x_n| \leq Bh^3, \quad B = const > 0. \tag{181}$$

Proof. The AM(2) method for IVP (180) is given by

$$x_{n+1} = x_n + \frac{h}{12}(8f_n + 5f_{n+1} - f_{n-1}), \quad x_0 = \alpha, \quad t_i := ih, \quad i = 0, 1, 2, \ldots, N. \tag{182}$$

As it follows from the analysis of the local truncation error of the AM(2)

$$\begin{aligned} e_{n+1} =& x(t_{n+1}) - x_{n+1} \\ =& x(t_n) - x_n + \frac{2h}{3}(f(t_n, x(t_n)) - f_n) + \frac{5h}{12}(f(t_{n+1}, x(t_{n+1})) - f_{n+1}) - \frac{h}{12}(f(t_{n-1}, x(t_{n-1})) - f_{n-1}) + R \\ =& e_n + \frac{2h}{3}(f(t_n, x(t_n)) - f_n) + \frac{5h}{12}(f(t_{n+1}, x(t_{n+1})) - f_{n+1}) - \frac{h}{12}(f(t_{n-1}, x(t_{n-1})) - f_{n-1}) + R, \end{aligned} \tag{183}$$

where $R = \mathcal{O}(h^4)$. As in Theorem 4.1, equation (183) shows us how the global error at time t_{n+1} depends on the propagated error

$$e_n + \frac{2h}{3}(f(t_n, x(t_n)) - f_n) + \frac{5h}{12}(f(t_{n+1}, x(t_{n+1})) - f_{n+1}) - \frac{h}{12}(f(t_{n-1}, x(t_{n-1})) - f_{n-1})$$

and the local truncation error given by the reminder R. Taking the absolute value of (183) yields

$$\begin{aligned} |e_{n+1}| \leq& |e_n| + \frac{2h}{3}|f(t_n, x(t_n)) - f_n| + \frac{5h}{12}|f(t_{n+1}, x(t_{n+1})) - f_{n+1}| + \frac{h}{12}|f(t_{n-1}, x(t_{n-1})) - f_{n-1}| + |R| \\ & \text{using the Lipschitz continuity of } f \text{ we have } |f(t_n, x(t_n)) - f_n| \leq L|x(t_n) - x_n|,\ L = const \geq 0, \\ \leq& |e_n| + \frac{2\hat{h}}{3}|x(t_n) - x_n| + \frac{5\hat{h}}{12}|x(t_{n+1}) - x_{n+1}| + \frac{\hat{h}}{12}|x(t_{n-1}) - x_{n-1}| + |R|, \quad \hat{h} := hL, \\ \leq& |e_n| + \frac{2\hat{h}}{3}|e_n| + \frac{5\hat{h}}{12}|e_{n+1}| + \frac{\hat{h}}{12}|e_{n-1}| + |R| \\ \leq& (1 + \frac{2\hat{h}}{3})\widetilde{A}^{-1}|e_n| + \frac{\hat{h}}{12}\widetilde{A}^{-1}|e_{n-1}| + \widetilde{A}^{-1}|R|, \quad \widetilde{A} := 1 - \frac{5\hat{h}}{12}. \end{aligned} \tag{184}$$

Introducing the error bounding function $\delta_n = \max\limits_{\substack{0\le i\le n\\ n\in[0,N]}} |e_i|$ and rewriting (184) in terms of δ_n gives

$$\delta_{n+1} \le A\delta_n + \widetilde{A}^{-1}|R|, \quad A := (1+\frac{9\hat{h}}{12})\widetilde{A}^{-1}. \tag{185}$$

For $n = 0,1,2,\dots,N-1$ we find that

$$\begin{aligned}
\delta_0 =& x(t_0) - x_0 = 0,\\
\delta_1 \le& A\delta_0 + \widetilde{A}^{-1}|R_1|,\\
\delta_2 \le& A\delta_1 + \widetilde{A}^{-1}|R_2| = A^2\delta_0 + (A|R_1| + |R_2|)\widetilde{A}^{-1},\\
\delta_3 \le& A\delta_2 + \widetilde{A}^{-1}|R_3| = A^3\delta_0 + (A^2|R_1| + A|R_2| + |R_3|)\widetilde{A}^{-1},\\
&\dots\\
\delta_N \le& \sum_{i=1}^{N} A^{N-i}|R_i|\widetilde{A}^{-1}\,.
\end{aligned} \tag{186}$$

The error in the initial condition is zero, since the initial condition is known exactly.
Since[1] $|R_i| \le Ch^4$, $C = const > 0$, we can estimate (186) as

$$\delta_N \le \sum_{i=1}^{N} A^{N-i} Ch^4 \widetilde{A}^{-1}. \tag{187}$$

As in Theorem 3.1, we use the geometric series

$$1 + A + A^2 + \dots + A^{N-1} = \frac{A^N - 1}{A - 1} \tag{188}$$

in (187) which gives

$$\delta_N \le \frac{A^N - 1}{A - 1} Ch^4 \widetilde{A}^{-1}. \tag{189}$$

Then we rewrite A as

$$A := 1 + \frac{14\hat{h}}{12}\widetilde{A}^{-1}, \tag{190}$$

and use the following inequality

$$A \le e^{\frac{14\hat{h}}{12}\widetilde{A}^{-1}}. \tag{191}$$

Exponentiating (191) to the power of N gives

$$\begin{aligned}
A^N \le& e^{\frac{14\hat{h}}{12}\widetilde{A}^{-1}N}\\
\le& e^{t_N \frac{14L}{12}\widetilde{A}^{-1}}.
\end{aligned} \tag{192}$$

Using (192) in (189) leads to

$$\delta_N \le Bh^3 \tag{193}$$

where $B := 12C(e^{t_N \frac{14L}{12}\widetilde{A}^{-1}} - 1)/(14L)$.
Thus, we have proved that the 2-step Adams-Moulton method converges to the exact solution with order $p = 3$. ■

Lecture 6 Consistency of linear multistep methods

In one of the lectures before, we have already seen how to derive the most accurate 1-step method. Now, let us talk about the development of new LMMs in general.

Definition 6.1 (Linear Difference Operator of the LMM) *The linear difference operator $\mathcal{L}_h$ associated with the LMM*

$$\sum_{m=0}^{s} \alpha_m x_{n+m} = h \sum_{m=0}^{s} \beta_m f(t_{n+m}, x_{n+m}), \quad n = 0, 1, 2, \ldots \qquad , \text{ where } \alpha_m, \beta_m = const \in \mathbb{R}, \alpha_s = 1$$

is defined for an arbitrary continuously differentiable function $z(t)$ by

$$\mathcal{L}_h z(t) = \sum_{m=0}^{s} \alpha_m z(t+mh) - h \sum_{m=0}^{s} \beta_m z'(t+mh). \tag{194}$$

The development of new LMMs amounts to finding coefficients $\{\alpha_m, \beta_m\}$ to ensure that the resulting LMM is convergent.

Definition 6.2 (Consistency of the linear difference operator) *The linear difference operator $\mathcal{L}_h$ is said to be consistent of order p if*

$$\mathcal{L}_h z(t) = \mathcal{O}(h^{p+1}),\ p > 0.$$

If $p = 0$ then the linear difference operator is inconsistent.

Definition 6.3 (Consistency of the LMM) *An LMM which linear difference operator is consistent of order $p > 0$ is said to be consistent with the ODE $\boldsymbol{x}' = \boldsymbol{f}(t, \boldsymbol{x})$, otherwise the LMM is called inconsistent.*

Note that the method is of no use if it is inconsistent!

6.1 Is the Euler method consistent?

The Euler method for the ODE $x' = f(t, x)$ is

$$x_{n+1} = x_n + hf_n. \tag{195}$$

The linear difference operator (LDO) for the Euler method is then given by

$$\mathcal{L}_h z(t) = z(t+h) - z(t) - hz'(t). \tag{196}$$

In order to compute the order of consistent of the LDO associated with the Euler method we plug the Taylor expansion of $z(t+h)$ into (196) and get

$$\mathcal{L}_h z(t) = \frac{h^2}{2} z''(t) + \mathcal{O}(h^3) = \mathcal{O}(h^2). \tag{197}$$

Thus the Euler method is consistent of order $p = 1$.

Remark 6.1 *Another way to figure out whether the numerical method is consistent or inconsistent is to look at the local truncation error. If the local truncation error is of order $p > 1$ then the method is consistent of order $p - 1$, otherwise it is inconsistent. For example, the local truncation* *error of Euler's method is of order $p = 2$ therefore the method is consistent of order $p = 1$ as shown above.*

All the methods we have studied so far are consistent!

6.2 What are the conditions for the 1-step LMM to be consistent?

As we know, in order for the LMM to be consistent of order p it must have the associated LDO of order $p > 1$ or, equivalently, the same order of the local truncation error. This gives rise to the question: what are the conditions which make the LMM consistent, or what are the conditions which give the associated LDO of order $p > 1$? In order to answer the question we will take the LDO of the 1-step LMM in the general form, namely

$$\mathcal{L}_h z(t) = z(t+h) + \alpha_0 z(t) - h(\beta_0 z'(t) + \beta_1 z'(t+h)), \tag{198}$$

Taylor expand $z(t+h)$ and $z'(t+h)$, and plug the expansions back into (198)

$$\begin{aligned}\mathcal{L}_h z(t) =& (z(t) + hz'(t) + \frac{h^2}{2}z''(t) + \mathcal{O}(h^3)) + \alpha_0 z(t) - h(\beta_0 z'(t) + \beta_1(z'(t) + hz''(t) + \frac{h^2}{2}z'''(t) + \mathcal{O}(h^3)))\\ =& (z(t) + hz'(t) + \frac{h^2}{2}z''(t)) + \alpha_0 z(t) - h(\beta_0 z'(t) + \beta_1(z'(t) + hz''(t)) + \mathcal{O}(h^3)\\ =& (1+\alpha_0)z(t) + h(1-\beta_0-\beta_1)z'(t) + \frac{h^2}{2}(1-2\beta_1)z''(t) + \mathcal{O}(h^3).\end{aligned} \tag{199}$$

Then, in order to make the LMM consistent, we have to choose the coefficients α_0, β_0, β_1 such that LDO (199) is of order at least $p = 2$. Hence, these conditions are

$$\begin{cases} 1+\alpha_0 = 0,\\ 1-\beta_0-\beta_1 = 0.\end{cases} \tag{200}$$

System (200) has to be solved to find a particular set of coefficients for the method.

6.2.1 The θ-method

The general solution of the system is given by $\alpha_0 = -1$ and $\beta_0 = 1 - \beta_1$, where $\beta_1 = \theta$. The parameter θ gives rise to a one-parameter family of solutions, which in turn leads to a one-parameter family of LMMs known as the θ-method:

$$\text{The } \theta\text{-method:} \qquad x_{n+1} = x_n + h(\theta f_{n+1} + (1-\theta)f_n) \tag{201}$$

You can easily see that different values of θ give different method. In particular,

$$\begin{cases} \theta = 0 \Rightarrow & \text{The Euler method,}\\ \theta = \frac{1}{2} \Rightarrow & \text{The Trapezoidal rule,}\\ \theta = 1 \Rightarrow & \text{The Backward Euler method.}\end{cases} \tag{202}$$

6.3 What are the conditions for the 2-step LMM to be consistent?

We can derive the consistency condition for the 2-step LMM in the same manner as we did it for the 1-step LMM. First, we take the LDO corresponding to the 2-step LMM

$$\mathcal{L}_h z(t) = z(t+2h) + \alpha_1 z(t+h) + \alpha_0 z(t) - h(\beta_0 z'(t) + \beta_1 z'(t+h) + \beta_2 z'(t+2h)), \tag{203}$$

Taylor expand $z(t+h)$, $z(t+2h)$, $z'(t+h)$, and $z'(t+2h)$, and plug the expansions back into (203)

$$\begin{aligned}\mathcal{L}_h z(t) =& z(t) + 2hz'(t) + 2h^2 z''(t) + \frac{4h^3}{3} z'''(t) + \mathcal{O}(h^4)\\ &+\alpha_1(z(t) + hz'(t) + \frac{h^2}{2} z''(t) + \frac{h^3}{6} z''(t) + \mathcal{O}(h^4)) + \alpha_0 z(t)\\ &-h(\beta_0 z'(t)\\ &+\beta_1(z'(t) + hz''(t) + \frac{h^2}{2} z'''(t) + \frac{h^3}{6} z''''(t) + \mathcal{O}(h^4))\\ &+\beta_2(z'(t) + 2hz''(t) + 2h^2 z'''(t) + \frac{4h^3}{3} z''''(t) + \mathcal{O}(h^4)))\\ =&(1 + \alpha_0 + \alpha_1)z(t) + h(2 + \alpha_1 - (\beta_0 + \beta_1 + \beta_2)z'(t) + \mathcal{O}(h^2).\end{aligned} \tag{204}$$

Thus, the 2-step LMM is consistent of order $p = 1$ if

$$\begin{cases} 1 + \alpha_0 + \alpha_1 =0,\\ \qquad 2 + \alpha_1 =\beta_0 + \beta_1 + \beta_2.\end{cases} \tag{205}$$

Note that the consistency conditions (205) are the minimum requirement for the method to be consistent. If one wants to derive an LMM with the highest order of consistency (i.e., the most accurate one) then the technique used for the derivation of the most accurate one-step method can be applied in this case too.

6.4 What are the conditions for the 3-step LMM to be consistent?

Acting in the same manner as before, we write out the LDO corresponding to the 3-step LMM

$$\mathcal{L}_h z(t) = z(t+3h)+\alpha_2 z(t+2h)+\alpha_1 z(t+h)+\alpha_0 z(t)-h(\beta_0 z'(t)+\beta_1 z'(t+h)+\beta_2 z'(t+2h)+\beta_3 z'(t+3h)), \tag{206}$$

Taylor expand $z(t+h)$, $z(t+2h)$, $z(t+3h)$, $z'(t+2)$, $z'(t+2h)$, $z'(t+3h)$ up to order 2, and plug these expansions back into (206)

$$\begin{aligned}\mathcal{L}_h z(t) =& z(t) + 3hz'(t) + \alpha_2(z(t) + 2hz'(t)) + \alpha_1(z(t) + hz'(t)) + \alpha_0 z(t)\\ &-h(\beta_0 z'(t) + \beta_1 z'(t) + \beta_2 z'(t) + \beta_3 z'(t)) + \mathcal{O}(h^2).\end{aligned} \tag{207}$$

Note that we do not really need terms of order higher than 2 in (207), since we only want to show that the method is consistent. Thus, the 3-step LMM is consistent of order $p = 1$ if

$$\begin{cases} 1 + \alpha_0 + \alpha_1 + \alpha_2 =0,\\ \qquad 3 + 2\alpha_2 + \alpha_1 =\beta_0 + \beta_1 + \beta_2 + \beta_3.\end{cases} \tag{208}$$

6.5 How to find a 2-step LMM with the highest order of consistency

As with the derivation of the 1-step LMM, it is very advisable to take some time to think about how would you approach this problem.
The general roadmap of the development looks exactly the same as for the 1-step method.

1. **Fix the form of a two-step method.**
   Let it be
   $$x_{n+2} + \alpha_0 x_n = h(\beta_1 f_{n+1} + \beta_0 f_n). \tag{209}$$
   In this particular form, we take $\alpha_1 = \beta_2 = 0$ for simplicity of exhibition.

2. **Find the maximum order of consistency of the LDO.**
The LDO associated with LMM (209) is

$$\mathcal{L}_h z(t) = z(t+2h) + \alpha_0 z(t) - h(\beta_1 z'(t+h) + \beta_0 z'(t)) \tag{210}$$

To get the maximum order of consistency, we Taylor expand $z(t+2h)$ and $z'(t+h)$, plug them into (210):

$$\begin{aligned}\mathcal{L}_h z(t) =& z(t+2h) + \alpha_0 z(t) - h(\beta_1 z'(t+h) + \beta_0 z'(t)) \\ =& z(t) + 2hz'(t) + 2h^2 z''(t) + \frac{4h^3}{3} z'''(t) + \mathcal{O}(h^4) + \alpha_0 z(t) \\ &- h(\beta_1(z'(t) + hz''(t) + \frac{h^2}{2} z'''(t) + \frac{h^3}{6} z''''(t) + \mathcal{O}(h^4)) + \beta_0 z'(t)) \\ =& (1+\alpha_0)z(t) + h(2-(\beta_1+\beta_0))z'(t) + h^2(2-\beta_1)z''(t) + h^3(\frac{4}{3} - \frac{\beta_1}{2})z'''(t) + \mathcal{O}(h^4).\end{aligned} \tag{211}$$

Now, we have to find the unknown coefficients α_0, β_0, β_1 which maximize the order of consistency of (211). These coefficients can be found as the solution to the following system:

$$\begin{cases} 1+\alpha_0 =0, \\ 2-(\beta_1+\beta_0) =0, \\ 2-\beta_1 =0. \end{cases} \tag{212}$$

The solution is given by

$$\begin{cases} \alpha_0 =-1, \\ \beta_0 =0, \\ \beta_1 =2. \end{cases} \tag{213}$$

Upon substitution of (213) into (209) gives the 2-step LMM with the highest order of consistency:

$$\text{The Leapfrog method:} \qquad x_{n+2} = x_n + 2hf_{n+1}. \tag{214}$$

Can we have even higher order of consistency with (209)**?**

Getting back to the derivation of consistency conditions for higher-step LMMs it is worth noting that the same steps as for the derivation of lower-step LMMs should be taken. However, instead of doing it we will shift our attention to characteristic polynomials, which will allow us to reformulate the consistency conditions and look at it at a different angle.

Definition 6.4 (Characteristic polynomials of the LMM) *The first and second characteristic polynomials of the s-step LMM*

$$\sum_{m=0}^{s} \alpha_m x_{n+m} = h \sum_{m=0}^{s} \beta_m f(t_{n+m}, x_{n+m}), \quad n = 0,1,2,\ldots \qquad , \text{ where } \alpha_m, \beta_m = const \in \mathbb{R}, \alpha_s = 1$$

are defined to be

$$\begin{aligned} \textit{First characteristic polynomial: } \rho(r) &= \sum_{m=0}^{s} \alpha_m r^m, \\ \textit{Second characteristic polynomial: } \sigma(r) &= \sum_{m=0}^{s} \beta_m r^m. \end{aligned} \tag{215}$$

6.6 Consistency in terms of characteristic polynomials

Now we can reformulate the consistency conditions in term of characteristic polynomials.

Theorem 6.1 *The s-step method*

$$\sum_{m=0}^{s} \alpha_m x_{n+m} = h \sum_{m=0}^{s} \beta_m f(t_{n+m}, x_{n+m}), \quad n = 0, 1, 2, \dots \quad , \alpha_m, \beta_m = const \in \mathbb{R}, \quad \alpha_s = 1$$

is consistent with the ODE $x' = f(t, x)$ if and only if

$$\rho(1) = 0 \text{ and } \rho'(1) = \sigma(1). \tag{216}$$

You can easily check that (216) is the same as (200) or (205).

6.7 Consistency and convergence

What can convergence tell us about consistency?

Theorem 6.2 *A convergent LMM is consistent.*

Proof. Suppose that the s-step LMM

$$\sum_{m=0}^{s} \alpha_m x_{n+m} = h \sum_{m=0}^{s} \beta_m f(t_{n+m}, x_{n+m}), \quad n = 0, 1, 2, \dots \qquad , \text{ where } \alpha_m, \beta_m = const \in \mathbb{R}, \alpha_s = 1$$

is convergent, namely $x_{n+m} \to x(t^* + mh)$, $m = 0, 1, \dots, s$, as $h \to 0$ when $t^* = t_n$. Here, x_{n+m} is a numerical solution, and $x(t^* + mh)$ is an exact solution.

Taking the limit of the LDO corresponding to the the LMM gives

$$\lim_{h \to 0} \sum_{m=0}^{s} \alpha_m x(t^* + mh) = \lim_{h \to 0} h \sum_{m=0}^{s} \beta_m f(t^* + mh, x(t^* + mh)). \tag{217}$$

Thus, we have

$$\sum_{m=0}^{s} \alpha_m x(t^*) = 0 \Rightarrow \rho(1) x(t^*) = 0. \tag{218}$$

Since $x(t^*) \neq 0$ then $\rho(1) = 0$ to satisfy $\rho(1)x(t^*) = 0$. Thus, we have proved that in order for the LMM to be convergent it must satisfy the first consistency condition $\rho(1) = 0$.

The next step is to prove that the convergent LMM satisfies the second consistency condition $\rho'(1) = \sigma(1)$. For doing so, we rewrite (217) as

$$\lim_{h \to 0} \frac{1}{h} \sum_{m=0}^{s} \alpha_m x(t^* + mh) = \lim_{h \to 0} \sum_{m=0}^{s} \beta_m f(t^* + mh, x(t^* + mh)). \tag{219}$$

The right hand side of (219) becomes

$$\sum_{m=0}^{s} \beta_m f(t^*, x(t^*)) = \sigma(1) f(t^*, x(t^*)). \tag{220}$$

To proceed with the left hand side, we apply the L'Hopital's rule, and get

$$\sum_{m=0}^{s} m\alpha_m x'(t^*) = \rho'(1)x'(t^*). \tag{221}$$

Equating (220) and (220) gives

$$\rho'(1)x'(t^*) = \sigma(1)f(t^*, x(t^*)). \tag{222}$$

Using the differential equation $x'(t^*) = f(t^*, x(t^*))$ in (223) results in the second consistency condition:

$$\rho'(1) = \sigma(1). \tag{223}$$

Thus we have proved that the s-step convergent LMM is consistent. ■

Up to now, all methods we studied happened to be convergent and consistent. It might seem like consistency can imply convergence. However, it is not always true. Let us consider a method which is consistent but not convergent (divergent).

Example 6.1 (Consistent but divergent LMM) *Consider the following 2-step LMM*

$$x_{n+2} + 4x_{n+1} - 5x_n = h(4f_{n+1} + 2f_n). \tag{224}$$

To show the method is consistent we can simply check the consistency conditions for the 2-step LMM, which are

$$\begin{cases} 1 + \alpha_0 + \alpha_1 = 0, \\ \quad 2 + \alpha_1 = \beta_0 + \beta_1 + \beta_2. \end{cases} \tag{225}$$

Method (224) *is consistent, since it satisfies the consistency conditions, namely* $\alpha_1 = 4$, $\alpha_0 = -5$, $\beta_1 = 4$, *and* β_0=*2.*

To show that the method is convergent or divergent we have to analyse the behaviour of the global error. For this, we first have to find the local truncation error. The LDO corresponding to the method is given by

$$\mathcal{L}_h z(t) = z(t+2h) + 4z(t+h) - 5z(t) - h(4z'(t+h) + 2z'(t)), \tag{226}$$

Taylor expansion of $z(t+h)$, $z(t+2h)$, *and* $z'(t+h)$, *and their substitution into* (226)

$$\begin{aligned} \mathcal{L}_h z(t) =& z(t) + 2hz'(t) + 2h^2 z''(t) + \frac{4h^3}{3} z'''(t) + \mathcal{O}(h^4) \\ &+4(z(t) + hz'(t) + \frac{h^2}{2} z''(t) + \frac{h^3}{6} z''(t) + \mathcal{O}(h^4)) - 5z(t) \\ &-h(4(z'(t) + hz''(t) + \frac{h^2}{2} z'''(t) + \frac{h^3}{6} z''''(t) + \mathcal{O}(h^4)) + 2z'(t)) \\ =& \mathcal{O}(h^4). \end{aligned} \tag{227}$$

Hence, the method is consistent of order $p = 3$. **Can you check this is correct?**

Theorem 6.3 *The 2-step LMM* (224) *for the general IVP*

$$x' = f(t, x), \quad x(t_0) = \alpha, \quad t \in [t_0, t_N] \tag{228}$$

diverges.

Can you prove this theorem? Or, can you show by example that the method diverges? ▲

In order to understand why consistent methods can diverge we will study finite difference equations and how they are related to the divergence of numerical methods.

Lecture 7 Linear difference equations (LDEs)

A detailed introduction into the theory of LDEs is beyond the scope of this course, therefore we limit ourselves only to discussing the material necessary for the purpose of this course. Our goal here is to know how solve an equation of the form

$$x_{n+2} + ax_{n+1} + bx_n = g_n, \tag{229}$$

with a, b being constants, and g_n being a function.

7.1 First-order inhomogeneous LDE $x_{n+1} = ax_n + b$

Let us start from a first order inhomogeneous LDE

$$x_{n+1} = ax_n + b,\ n = 0, 1, 2, \dots \tag{230}$$

with unknowns $\{x_n\}_{n=1,2,\dots}$ and constants a, b.

Since the initial condition x_0 is supposed to be known, we can solve this equation by computing its solution via the following sequence

$$\begin{aligned} x_1 &= ax_0 + b, \\ x_2 &= ax_1 + b = a^2x_0 + (a+1)b, \\ x_3 &= ax_2 + b = a^3x_0 + (a^2 + a + 1)b, \\ &\dots \end{aligned} \tag{231}$$

We find the general solution to (230), as a sum of the general solution to the homogeneous equation and a particular solution to the inhomogeneous equation (230). First, we consider the homogeneous equation by setting $b = 0$ in (230):

$$x_{n+1} = ax_n,\ n = 0, 1, 2, \dots \tag{232}$$

and set $x_0 = A$ ($A = const$). Thus, the general solution to the homogeneous equation (232) is

$$x_n = a^nA,\ n = 0, 1, 2, \dots \tag{233}$$

We can now seek for a particular solution of the inhomogeneous equation (230) in the form of a constant sequence

$$x_{n+1} = x_n = C,\ C = const. \tag{234}$$

Substitution of (234) into (230) gives

$$C = aC + b \Rightarrow C = \frac{b}{1-a},\ a \neq 1. \tag{235}$$

The sum of the particular solution (235) and the general solution (233) gives the general solution of the inhomogeneous equation (229) gives the general solution of the inhomogeneous equation:

$$x_n = Aa^n + \frac{b}{1-a},\ a \neq 1. \tag{236}$$

In fact, this is only one part of the solution which corresponds to the case $a \neq 1$. If $a = 1$ then equation (230)

$$x_{n+1} = x_n + b,\ n = 0, 1, 2, \dots \tag{237}$$

or

$$x_{n+1} - x_n = b,\ n = 0, 1, 2, \dots \tag{238}$$

Using the telescoping series property

$$x_n = (x_n - x_{n-1}) + (x_{n-1} - x_{n-2}) + \dots + (x_1 - x_0) + x_0 \tag{239}$$

together with (238) we find

$$x_n = nb + x_0,\ n = 0, 1, 2, \dots \tag{240}$$

which upon Substitution of $x_0 = A$ gives

$$x_n = A + nb,\ n = 0, 1, 2, \dots \tag{241}$$

Thus, the general solution to (230) is

$$x_n = \begin{cases} Aa^n + \dfrac{b}{1-a}, & a \neq 1, \\ A + nb, & a = 1. \end{cases} \tag{242}$$

To get the solution of equation (229), we need to consider another inhomogeneous equation.

7.2 First-order inhomogeneous LDE $x_{n+1} = ax_n + bk^n$

Let us consider an equation of the form

$$x_{n+1} = ax_n + bk^n,\ n = 0, 1, 2, \dots \tag{243}$$

with a, b, k being constants.

To get the solution of (243), we use the substitution $x_n = k^n y_n$ that yields

$$k^{n+1} y_{n+1} = ak^n y_n + bk^n, \tag{244}$$

or

$$y_{n+1} = \frac{a}{k} y_n + \frac{b}{k}. \tag{245}$$

Equation (245) is in the form of (230). Hence, the solution to (245) is given by

$$y_n = \begin{cases} A(\dfrac{a}{k})^n + \dfrac{\frac{b}{k}}{1-\frac{a}{k}}, & \dfrac{a}{k} \neq 1, \\ A + n\dfrac{b}{k}, & \dfrac{a}{k} = 1. \end{cases} \tag{246}$$

Or, in terms of x_n, it is

$$x_n = \begin{cases} Aa^n + \dfrac{b}{k-a} k^n, & \dfrac{a}{k} \neq 1, \\ Ak^n + nbk^{n-1}, & \dfrac{a}{k} = 1. \end{cases} \tag{247}$$

Now, let us switch to second-order LDEs.

7.3 Second-order homogeneous LDE $x_{n+2} + ax_{n+1} + bx_n = 0$

We start by solving a homogeneous second-order

$$x_{n+2} + ax_{n+1} + bx_n = 0, \; n = 0, 1, 2, \ldots \tag{248}$$

with a, b being constants.

This equation has the solution of the form $x_n = Ar^n$ ($A = const$) with r being a root of the quadratic equation

$$r^2 + ar + b = 0 \tag{249}$$

called **the auxiliary equation**. Suppose equation (249) has two roots r_1 and r_2. Then, we can recast (249) in the form

$$(r - r_1)(r - r_2) = r^2 - (r_1 + r_2)r + r_1 r_2 = r^2 + ar + b = 0, \quad a := -(r_1 + r_2), \quad b := r_1 r_2. \tag{250}$$

Thus, the homogeneous equation (248) becomes

$$x_{n+2} - (r_1 + r_2)x_{n+1} + r_1 r_2 x_n = 0, \tag{251}$$

or

$$(x_{n+2} - r_1 x_{n+1}) - r_2(x_{n+1} - r_1 x_n) = 0, \tag{252}$$

or

$$y_{n+1} - r_2 y_n = 0, \text{ where } y_n = x_{n+1} - r_1 x_n. \tag{253}$$

We already know that equation (253) has the general solution $y_n = Cr_2^n$, $C = const$. Then,

$$x_{n+1} - r_1 x_n = Cr_2^n. \tag{254}$$

We also know the general solution of (254):

$$x_n = \begin{cases} Ar_1^n + Br_2^n, & r_1 \neq r_2, \\ (A + nD)r_2^n, & r_1 = r_2, \end{cases} \tag{255}$$

where $B := C/(r_2 - r_1)$ and $D := C/r_2$.

For the third-order homogeneous LDE the solution is given by

$$x_n = \begin{cases} Ar_1^n + Br_2^n + Cr_3^n, & r_1 \neq r_2 \neq r_3, \\ Ar_1^n + (B + nC)r_2^n, & r_1 \neq r_2, \, r_2 = r_3, \\ (A + nB)r_1^n + Cr_3^n, & r_1 = r_2, \, r_2 \neq r_3, \\ (A + nB + n^2C)r_1^n, & r_1 = r_2 = r_3. \end{cases} \tag{256}$$

As can be seen from (255), the solution of the second-order homogeneous LDE is bounded if all the roots of its auxiliary equation $|r_i| \leq 1$ and any roots $|r_k| = 1$ are simple, or if all roots $|r_i| < 1$. The same is true for any k-order homogeneous or inhomogeneous LDE, but we will not derive this result in the course (for example, see ???).

Lecture 8 Convergence and zero stability

Our study of linear difference equations brings us to the series of important results.

Definition 8.1 (The root condition) *A polynomial of degree n is said to satisfy the root condition if all its roots $|r_i|_{i\in[1,n]} \le 1$ and any roots that satisfy $|r_k| = 1$ are simple.*

Definition 8.2 (The strict root condition) *A polynomial of degree n is said to satisfy the strict root condition if all its roots $|r_i|_{i\in[1,n]} < 1$.*

Definition 8.3 (Zero stability) *An LMM is said to be zero-stable if its first characteristic polynomial, $\rho(r)$, satisfies the root condition.*

Theorem 8.1 (Dahlquist equivalence theorem) *An LMM is convergent $\iff$ it is both consistent and zero-stable.*

Theorem 8.2 *The global error of a convergent LMM equals to its order of consistency.*

Being armed with new knowledge of why a numerical method can diverge let us get back to the consistent but divergent LMM, and study how our new findings can help us to understand the reason of its divergence.

8.1 Back to consistent but divergent LMM $x_{n+2} + 4x_{n+1} - 5x_n = h(4f_{n+1} + 2f_n)$

Application of the LMM

$$x_{n+2} + 4x_{n+1} - 5x_n = h(4f_{n+1} + 2f_n)$$

to the model problem

$$x' = 0, \quad x(t_0) = 1, \quad t \in [0,1] \tag{257}$$

gives the second-order homogeneous LDE

$$x_{n+2} + 4x_{n+1} - 5x_n = 0. \tag{258}$$

The characteristic equation for (258) is

$$r^2 + 4r - 5 = 0, \tag{259}$$

which has the roots $r_1 = 1$ and $r_2 = -5$. It is already clear (see the Dahlquist equivalence theorem) that the method diverges, since it is not zero-stable. However, we proceed with the example to figure out the exact reason of its divergence. We know that the solution of the second-order homogeneous LDE is given by (255). In this particular case, we use the solution for which $r_1 \neq r_2$, namely

$$x_n = Ar_1^n + Br_2^n. \tag{260}$$

Constants A and B are not known, and can be found from the initial conditions; we need two initial conditions for the LMM to start because it is a 2-step LMM. The first initial condition is known from the problem formulation ($x_0 = 1$). We can take $x_1 = 1 + h$, since the only extra requirement for the LMM to converge is that all initial conditions tend to x_0 as the time step $h \to 0$; $x_1 \to 0$ as $h \to 0$. Based on this, we write a system of equations to be solved for A and B:

$$\begin{cases} x_0 = Ar_1^0 + Br_2^0, \\ x_1 = Ar_1^1 + Br_2^1. \end{cases} \tag{261}$$

Substitution of x_0, x_1, and r_1, r_2 into (261) leads to

$$\begin{cases} 1 =& A + B, \\ 1 + h =& A - 5B. \end{cases} \tag{262}$$

The solution of (262) is

$$\begin{cases} A =& 1 + \dfrac{h}{6}, \\ B =& -\dfrac{h}{6}. \end{cases} \tag{263}$$

Upon substitution of this solution into (260) we get

$$x_n = (1 + \frac{h}{6}) - \frac{h}{6}(-5)^n. \tag{264}$$

Since (257) is integrated over $[0, t^*]$, $t^* = 1$, and $t^* = nh = 1 \Rightarrow h = 1/n$, we can rewrite (264) as

$$x_n = 1 + \frac{1}{6n}(1 - (-5)^n). \tag{265}$$

Clearly, x_n grows rapidly as $n \to \infty$ thus showing that the method diverges.

8.2 Convergence of the 3-step LMM

Let us consider a 3-step LMM

$$x_{n+3} + x_{n+2} - x_{n+1} - x_n = 10hf_n. \tag{266}$$

Does this LMM converge or diverge? To answer the question, we apply the LMM to the same problem (257)

$$x_{n+3} + x_{n+2} - x_{n+1} - x_n = 0. \tag{267}$$

This is a third-order homogeneous LDE with the characteristic polynomial

$$r^3 + r^2 - r - 1 = 0 \tag{268}$$

with the roots $r_1 = 1$, $r_2 = r_3 = -1$, which rule the LMM out from being convergent. Obviously, one can use, for example, the initial conditions $x_1 = 1 + h$ and $x_2 = 1 + 2h$ which converge to x_0 as $h \to 0$ and find the solution x_n to see that the method is divergent. For this, one should use the second solution in (256), i.e.

$$Ar_1^n + (B + nC)r_2^n, \quad r_1 \neq r_2,\, r_2 = r_3. \tag{269}$$

Constants A, B, and C can be found from the following system of linear equations

$$\begin{cases} x_0 =& Ar_1^0 + (B + 0C)r_2^0, \\ x_1 =& Ar_1^1 + (B + 1C)r_2^1, \\ x_2 =& Ar_1^2 + (B + 2C)r_2^2. \end{cases} \tag{270}$$

The solution of (270) is given by

$$\begin{cases} A =& 1 + h, \\ B =& -h, \\ C =& h. \end{cases} \tag{271}$$

Upon substitution of (271) into (260) we get

$$x_n = (1 + h) + (-h + nh)(-1)^n. \tag{272}$$

As can be seen from the global error

$$|x(1) - x_1| \to 1 \text{ as } h \to 0, \tag{273}$$

thus showing that the method diverges; here, $x(1)$ is the exact solution of (257), and x_1 is the numerical solution.

When developing new LMMs it is very instructive to know the following theorem

Theorem 8.3 (The first Dahlquist barrier) *The order p of a stable s-step LMM satisfies*

$$\begin{cases} p \le s+2, & \textit{if s is even}, \\ p \le s+1, & \textit{if s is odd}, \\ p \le s, & \textit{if } \beta_s \le 0. \end{cases} \tag{274}$$

8.3 Consistency and zero-stability in error analysis

In order to study how consistency and zero-stability contribute in the global error we consider a linear ODE

$$x' = \lambda x + g(t), \quad x(t_0) = \alpha, \quad t \in [t_0, t_N] \tag{275}$$

and the general s-step method

$$\sum_{m=0}^{s} \alpha_m x_{n+m} = h \sum_{m=0}^{s} \beta_m f(t_{n+m}, x_{n+m}), \quad n = 0, 1, 2, \ldots \qquad , \tag{276}$$

where $\alpha_m, \beta_m = const \in \mathbb{R}$, and $\alpha_s = 1$. The application of (276) to (275) leads to

$$\sum_{m=0}^{s} \alpha_m x_{n+m} = \hat{h} \sum_{m=0}^{s} \beta_m x_{n+m} + h \sum_{m=0}^{s} \beta_m g(t_{n+m}), \quad \hat{h} := \lambda h. \tag{277}$$

Using the linear difference operator $\mathcal{L}_h z(t)$ we can see that the exact solution satisfies the same equation with a reminder R_n:

$$\sum_{m=0}^{s} \alpha_m z(t + mh) = \hat{h} \sum_{m=0}^{s} \beta_m z(t + mh) + h \sum_{m=0}^{s} \beta_m g(t_{n+m}) + R_n. \tag{278}$$

Then the global error $e_n = z(t_n) - x_n$ satisfies the linear difference equation

$$\sum_{m=0}^{s} (\alpha_m - \hat{h}\beta_m) e_{n+m} = R_n, \tag{279}$$

with the starting values $e_0 = 0, e_1 = x(t_1) - x_1, \ldots, e_{s-1} = x(t_{s-1}) - x_{s-1}$; $e_0 = 0$ since $x(t_0)$ is known exactly.

Equation (279) describes how local errors R_n accumulate into the global error e_n. To simplify further calculations and make use of our knowledge about linear difference equations we assume that R_n is constant R. This gives us an s-order inhomogeneous linear difference equation. The general solution of this equation can be found as a sum of a particular solution of the inhomogeneous equation and the general solution of the homogeneous equation.

First, we seek for a particular solution of the inhomogeneous equation in the form of a constant sequence $e_n = C$, $C = const$. Substitution of $e_n = C$ into (279) and taking into account $R_n = R = const$ as well as the consistency condition $\rho(1) = 0$ (the first characteristic polynomial $\rho(r) = \sum\limits_{m=0}^{s} r^m \alpha_m$) we have

$$\sum_{m=0}^{s} (\alpha_m - \hat{h}\beta_m) C = R, \tag{280}$$

$$C = \frac{R}{\hat{h}\sigma(1)}, \quad \sigma(r) = \sum_{m=0}^{s} r^m \beta_m, \tag{281}$$

with $\sigma(r)$ being the second characteristic polynomial.

Second, we have to find the general solution of the homogeneous equation

$$\sum_{m=0}^{s} (\alpha_m - \hat{h}\beta_m) e_{n+m} = 0. \tag{282}$$

If the characteristic polynomial

$$\sum_{m=0}^{s} (\alpha_m - \hat{h}\beta_m) r^n = 0. \tag{283}$$

has distinct roots $r_i \neq r_j$ for $i \neq j$ then the solution is given by

$$e_n = \sum_{m=1}^{s} A_m r_m^n,\ A_m = const,\ m = 1, 2, \ldots, s. \tag{284}$$

Hence, the general solution of the inhomogeneous equation is

$$e_n = \sum_{m=1}^{s} A_m r_m^n + C. \tag{285}$$

The constants A_m are determined from the initial conditions for the LMM.

If the root condition is violated then the first term grows as $h \to 0$ and it leads to divergence, therefore we need zero-stability to keep it under control. On the other hand, the local truncation error contributes to the global error through the second term which is of order $\mathcal{O}(h^p)$ if $R = \mathcal{O}(h^{p+1})$. Thus, consistency of the LMM ensures that the local errors tend to zero as $h \to 0$.

Lecture 9 Absolute stability

What we have learned for now tells us that a convergent method gives a numerical solution which is arbitrary close to the exact solution provided that the time step, h, is sufficiently small. However, even a convergent method may not be that much useful in practice if the number of time steps to obtain results of even low accuracy is too high. Let us consider several examples to have a better grasp of this situation.

Example 9.1 (Forward Euler method) *Consider the IVP*

$$x' = -8x + 40(3e^{-t/8} + 1), \quad x(0) = 100, \quad t \in [0, 10]. \tag{286}$$

The exact solution to (286) *is given by*

$$x = \frac{1675}{21} e^{-8t} + \frac{320}{21} e^{-t/8} + 5. \tag{287}$$

Note that the first term (287) *decays very rapidly, while the second one decays slowly. If you draw this solution you will see that it decays quite rapidly up to approximately* $\frac{320}{21} + 5$ *and then tends slowly to 5. Application of the Forward Euler method (*$x_{n+1} = x_n + hf_n$*) to* (286) *gives*

$$x_{n+1} = x_n + h(-8x_n + 40(3e^{-t_n/8} + 1)). \tag{288}$$

The numerical solution for different time steps is presented in Figure 1.

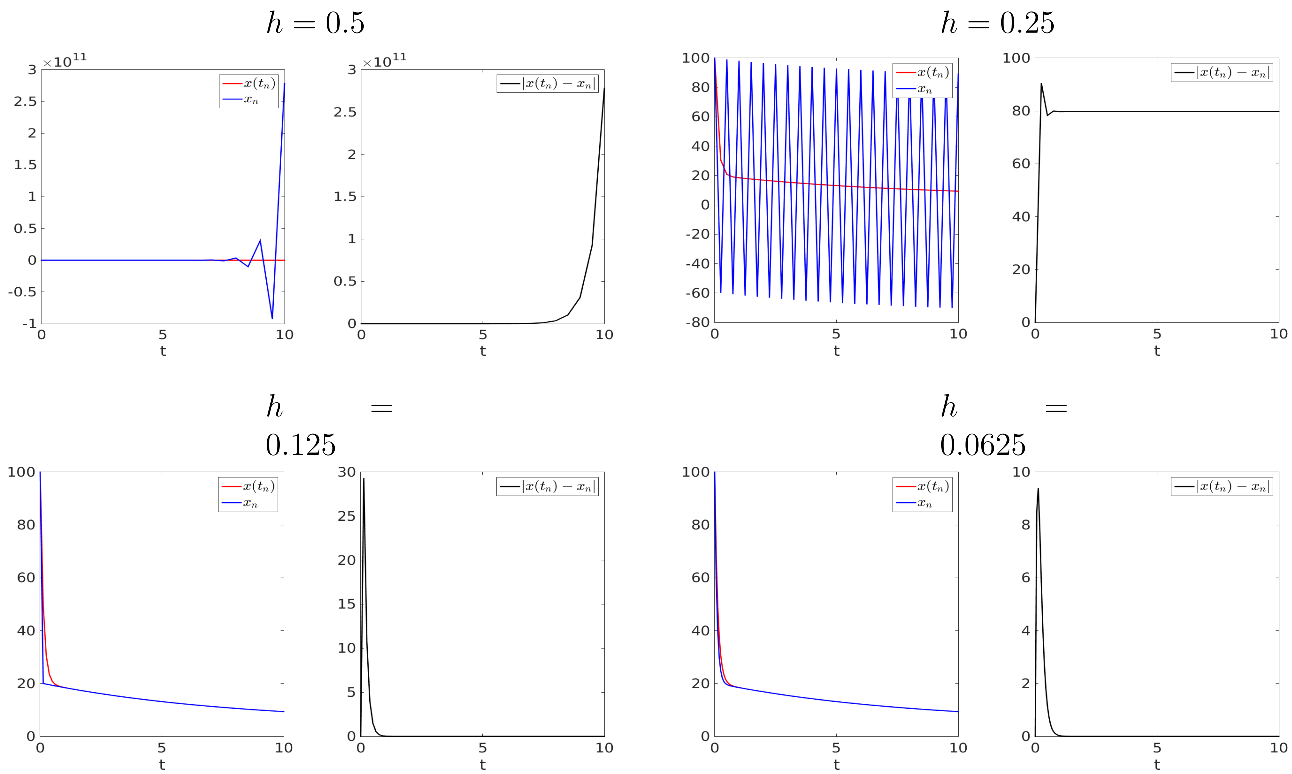

Figure 1: Shown is the exact, $x(t_n)$, and numerical, x_n, solutions to (286) computed with the Forward Euler method with different time steps, and the evolution of the global error $|x(t_n) - x_n|$.

As can be seen from Figure 1, the Euler method exhibits an instability when the time step is not small enough. This behaviour rules the Euler method out from the list of methods which can treat problems with rapidly decaying solutions unless the time step is small. On the other hand, when h

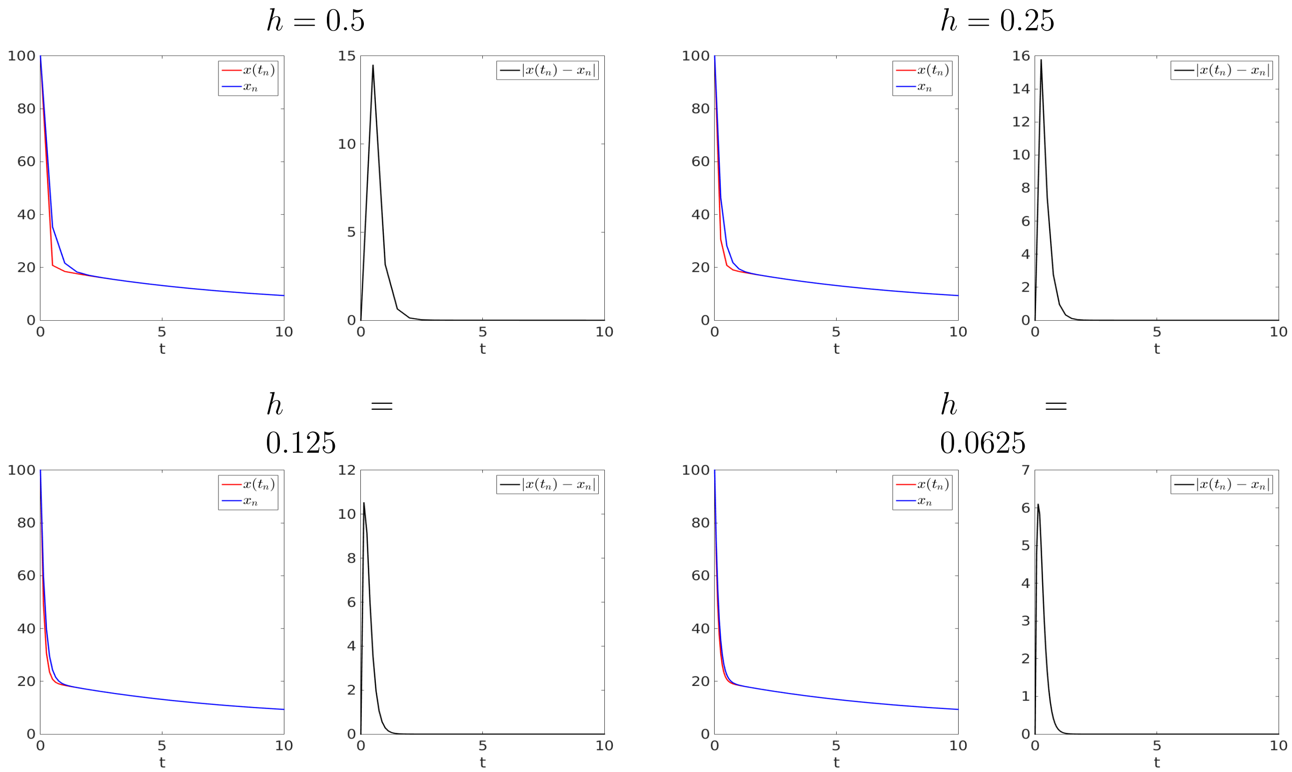

Figure 2: Shown is the exact, $x(t_n)$, and numerical, x_n, solutions to (286) computed with the Backward Euler method with different time steps, and the evolution of the global error $|x(t_n) - x_n|$.

is small enough to avoid the instability, the accuracy (especially for the long-term solution) may be way higher than is required. ▲

Example 9.2 (Backward Euler method) *In this example, we consider the same IVP* (286)*, but will use the Backward Euler method ($x_{n+1} = x_n + hf_{n+1}$):*

$$x_{n+1} = x_n + h(-8x_{n+1} + 40(3e^{-t_{n+1}/8} + 1)). \tag{289}$$

The numerical solution for different time steps is presented in Figure 2.

Although the accuracy of the Forward and Backward Euler methods is the same, the way the errors propagate is different. The Backward Euler method does not show any instability depending on the time step. Therefore, we can choose the time step on grounds of accuracy. ▲

Example 9.3 (Forward Euler method) *Let us consider another IVP*

$$x' = \frac{1}{8}(5 - x - 5025e^{-8t}), \quad x(0) = 100, \quad t \in [0, 10]. \tag{290}$$

This equation looks different compared to equation (286)*, but it has the same solution. Application of the Forward Euler method ($x_{n+1} = x_n + hf_n$) to* (290) *gives*

$$x_{n+1} = x_n + \frac{h}{8}(5 - x_n - 5025e^{-8t_n}). \tag{291}$$

The numerical solution for different time steps is presented in Figure 3.

The Forward Euler method shows no sign of instability and reacts to the time step size as expected - the smaller the time step is, the smaller the error becomes. ▲

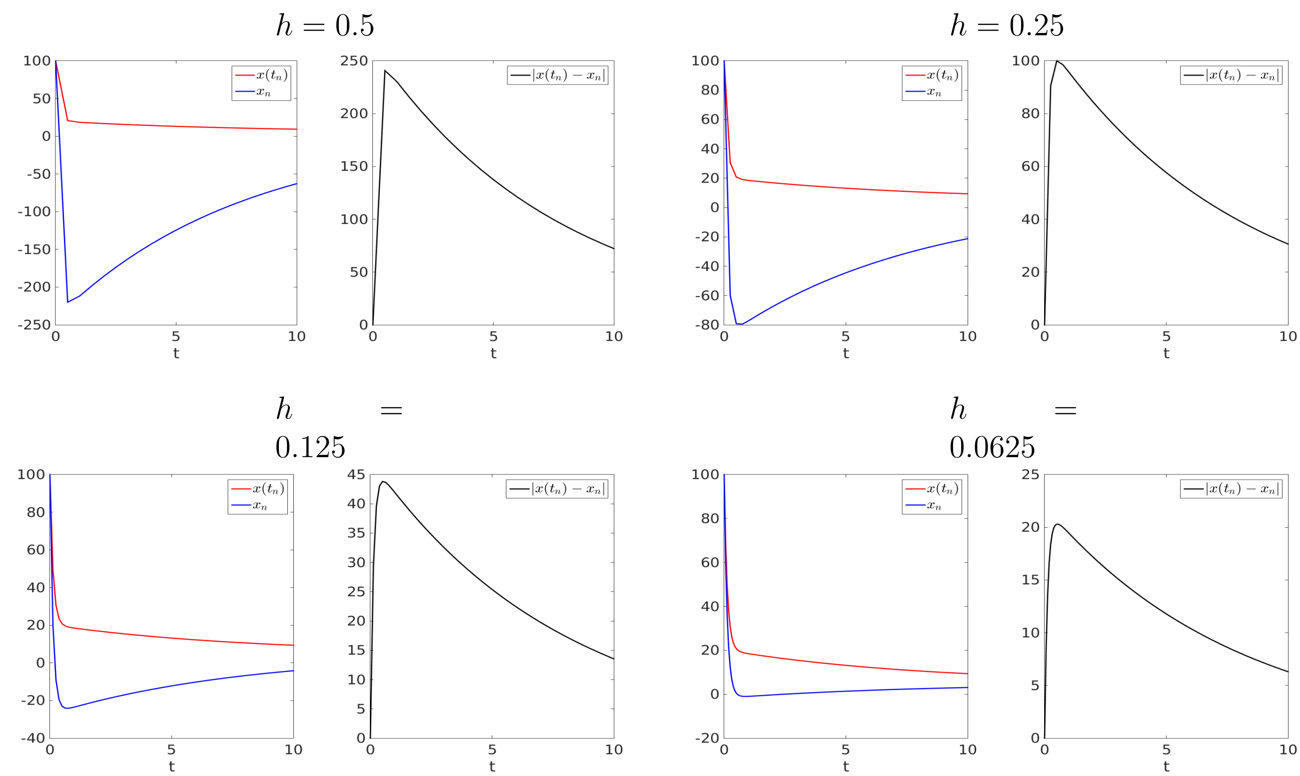

Figure 3: Shown is the exact, $x(t_n)$, and numerical, x_n, solutions to (290) computed with the Forward Euler method with different time steps, and the evolution of the global error $|x(t_n) - x_n|$.

Example 9.4 (Backward Euler method) *Now, we apply the Backward Euler method to* (290)*:*

$$x_{n+1} = x_n + \frac{h}{8}(5 - x_{n+1} - 5025e^{-8t_{n+1}}). \tag{292}$$

and show the numerical solution for different time steps in Figure 4.

As with the Forward Euler method, the Backward Euler method works as expected. ▲

If we take a closer look at (286) and (290) we can see that the forcing functions $e^{-t/8}$ and e^{-8t} are different. In particular,

1. The function $e^{-t/8}$ decays slowly compared to the solution of the homogeneous equation $x' = -8x$ ($x = e^{-8t}$). This suggests that the oscillations in the numerical solution of (286) computed with the Forward Euler method can be associated with the rapidly decaying solution of the homogeneous equation $x' = -8x$.

2. The function e^{-8t} decays rapidly relative to the solution of the homogeneous equation $x' = -x/8$ ($x = e^{-t/8}$). This suggests that the absence of oscillations in the numerical solution of (290) computed with the Forward Euler method can be associated with the slowly decaying solution of the homogeneous equation $x' = -x/8$.

Thus, since we have two inhomogeneous equations with the same solutions but different homogeneous equations, and these inhomogeneous equations demonstrate different behaviour we conclude that this behaviour is due to the difference in the homogeneous equations, and this motivates our study of homogeneous equations.

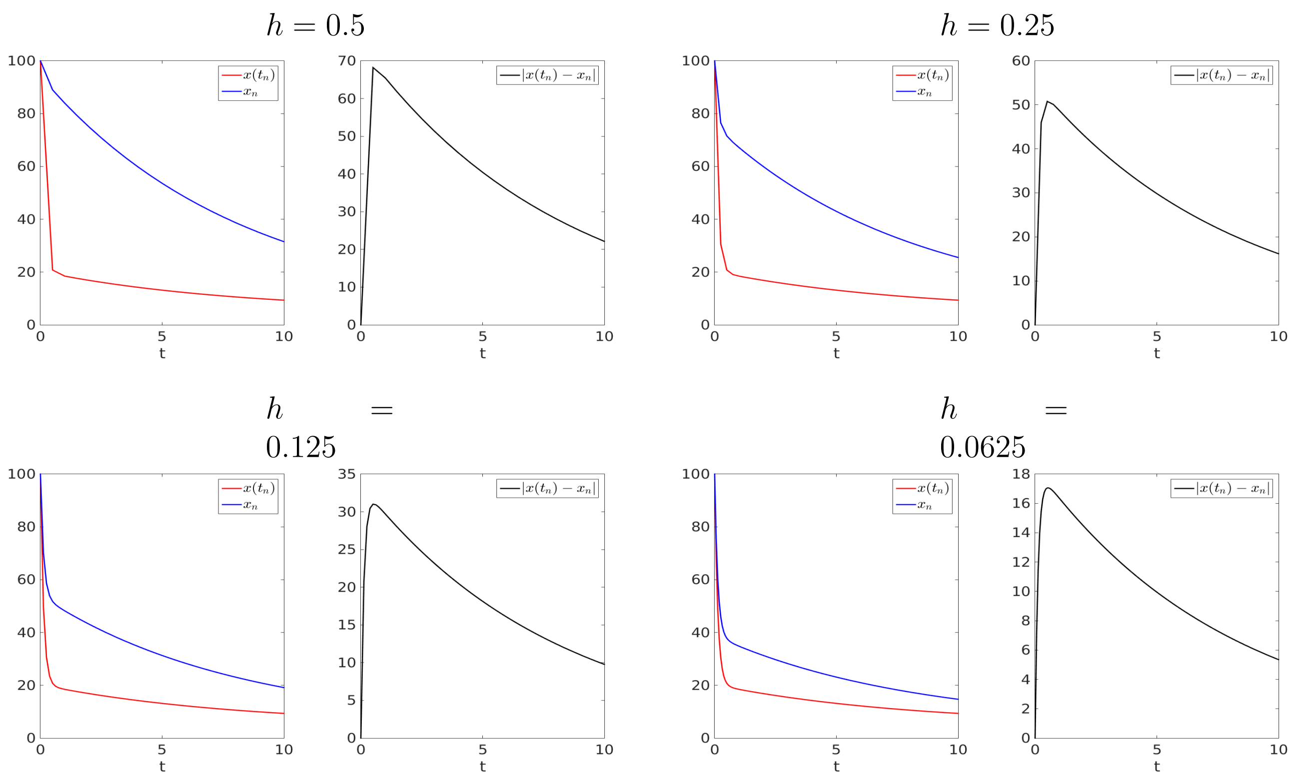

Figure 4: Shown is the exact, $x(t_n)$, and numerical, x_n, solutions to (290) computed with the Backward Euler method with different time steps, and the evolution of the global error $|x(t_n) - x_n|$.

9.1 Homogeneous equation $x' = \lambda x$

Let us consider a homogeneous equation

$$x' = \lambda x, \quad Re(\lambda) < 0, \quad x(t_0) = \alpha, \quad t \in [t_0, t_N]. \tag{293}$$

The solution of (293) is given by $x = Ce^{\lambda t}$, $C = const$, and $x(t) \to 0$ as $t \to \infty$ for any C. Note that λ can be complex.

Our goal is to find those LMMs which (when applied to (293)) give solutions $x_n \to 0$ as $n \to \infty$ with a given fixed time step h. In other words, we will seek for LMMs which can reproduce the long-time behaviour.

Definition 9.1 (Absolute stability) *An LMM is said to be absolutely stable (when applied to $x' = \lambda x$, $Re(\lambda) < 0$ with a given h), if the numerical solution $x_n \to 0$ as $n \to \infty$ for any choice of starting values.*

The requirement ($x_n \to 0$ as $n \to \infty$) amounts to the global error being damped as time increases.

It is very helpful and instructive to find a link between absolute stability and roots of the auxiliary equation, since it will give a better understanding of absolute stability. For this, we consider the general s-step LMM

$$\sum_{m=0}^{s} \alpha_m x_{n+m} = h \sum_{m=0}^{s} \beta_m f(t_{n+m}, x_{n+m}), \quad n = 0, 1, 2, \ldots \qquad , \text{ where } \alpha_m, \beta_m = const \in \mathbb{R}, \text{ and } \alpha_s = 1$$

and apply it to (293):

$$\sum_{m=0}^{s} (\alpha_m - \hat{h}\beta_m) x_{n+m} = 0. \tag{294}$$

This is a homogeneous linear difference equation with constant coefficient α_m and β_m. It has solutions of the form $x_n = Cr^n$, where $C = const$ and r is a root of the auxiliary equation

$$p(r) := \sum_{m=0}^{s} (\alpha_m - \hat{h}\beta_m) r^m = 0. \tag{295}$$

The polynomial

$$p(r) = \rho(r) - \hat{h}\sigma(r)$$

is called **the stability polynomial of the LMM**.

If the roots of the stability polynomial are all distinct then the solution of LDE (294) is given by

$$x_n = \sum_{m=1}^{s} A_m r_m^n, \; A_m = const. \tag{296}$$

In order for $x_n \to 0$ as $n \to \infty$ (for any constant A_m), all roots of the stability polynomial must be $|r_m| < 1$, i.e. the stability polynomial must satisfy the strict root condition.

9.2 Region and interval of absolute stability

In general, a given LMM is not supposed to be absolutely stable for any choice of h, therefore we define the region of absolute stability.

Definition 9.2 (Region of absolute stability) *The set of values in the complex $\hat{h}$-plane for which the LMM is absolutely stable is called the region of absolute stability of the LMM.*

Definition 9.3 (Interval of absolute stability) *The interval* $(Re(\hat{h}), 0)$, $Re(\hat{h}) < 0$ *for which the LMM is absolutely stable is called the interval of absolute stability of the LMM.*

The interval of absolute stability is given by the intersection of the region of absolute stability with the negative real $\hat{h}$-axis.

Example 9.5 (Region and interval of absolute stability of the Euler method) *What is the region and interval of absolute stability of the Euler method? To answer the question we apply the Euler method to the homogeneous equation* (293)*:*

$$x_{n+1} = x_n + \hat{h}x_n, \quad \hat{h} := \lambda h. \tag{297}$$

The stability polynomial for (297) *is given by*

$$p(r) = r - (1 + \hat{h}). \tag{298}$$

It has one root $r = 1 + \hat{h}$*. Hence, the region of absolute stability is an open disk* $|1 + \hat{h}| < 1$ *of radius 1 centered at* $\hat{h} = -1$*.*
To find the interval of absolute stability we have to solve the inequality $|1 + \hat{h}| < 1$*. The solution is given by* $\hat{h} \in (-2, 0)$*. Pay attention that this is the interval of absolute stability for any* λ*. For a given equation, one has to compute* h *which satisfies the interval of absolute stability. As an illustration, let us consider the equation* $x' = -8x$ *and find* h *which satisfies the interval of absolute stability* $-2 < \hat{h} < 0$*. Since* $\lambda = -8$ *and* $\hat{h} = \lambda h$ *we have* $-2 < -8h < 0 \Rightarrow h < 1/4$*. This means that for* $h < 1/4$ *the numerical solution given by the Euler method tends to zero as* $t \to \infty$*. And this is exactly the reason for why the Forward Euler method for problem* (286) *with* $h = 0.25$ *diverges (see the example above). If* $h < 1/4$ *(let it be* $h = 0.24$*) the method starts to converge to the exact solution (Figure 5).* ▲

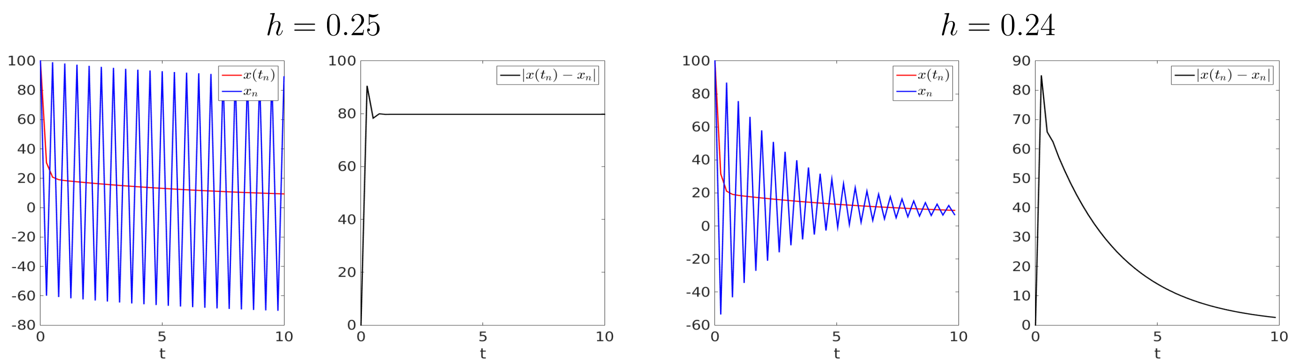

Figure 5: Shown is the exact, $x(t_n)$, and numerical, x_n, solutions to (286) computed with the Forward Euler method with different time steps, and the evolution of the global error $|x(t_n) - x_n|$.

Example 9.6 (Region and interval of absolute stability of the Trapezoidal rule) *In order to find the region and interval of absolute stability of the Trapezoidal rule*

$$x_{n+1} = x_n + \frac{h}{2}(f_{n+1} + f_n)$$

we have to apply it to the equation $x' = \lambda x$*:*

$$x_{n+1} = x_n + \frac{\hat{h}}{2}(x_{n+1} + x_n), \quad \hat{h} := \lambda h. \tag{299}$$

The stability polynomial for (299) *is given by*

$$p(r) = r - 1 - \frac{\hat{h}}{2}(r+1). \tag{300}$$

It has single root $r = \dfrac{1+\frac{\hat{h}}{2}}{1-\frac{\hat{h}}{2}}$. *Hence, the region of absolute stability is the entire left-half complex* $\hat{h}$*-plane, and therefore the interval of absolute stability is* $\hat{h} \in (-\infty, 0)$. *It means that the time step can be chosen on grounds of accuracy with no regard to stability. Take a look at how the Trapezoidal rule works for* (286)*:*

$$x_{n+1} = x_n + \frac{h}{2}(-8x_{n+1} + 40(3e^{-t_{n+1}/8} + 1) - 8x_n + 40(3e^{-t_n/8} + 1)). \tag{301}$$

The results are shown in Figure 6. Even for a large time step $h = 1$ *the error decreases very quickly.* ▲

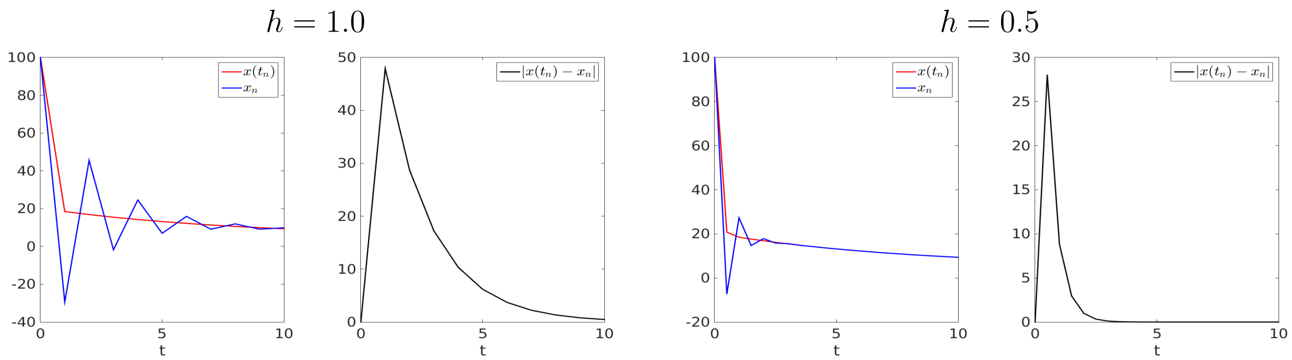

Figure 6: Shown is the exact, $x(t_n)$, and numerical, x_n, solutions to (286) computed with the Trapezoidal rule with different time steps, and the evolution of the global error $|x(t_n) - x_n|$.

Definition 9.4 (A-stability) *A numerical method is said to be A-stable if its region of absolute stability includes the entire left-half complex* $\hat{h}$*-plane.*

For example, the Trapezoidal rule is an A-stable method.

It is instructive to know the second Dahlquist barrier theorem when developing new LMMs.

Theorem 9.1 (The second Dahlquist barrier)

- *There is no A-stable explicit LMM.*
- *An A-stable implicit LMM cannot be of order* $p > 2$.
- *The order-two A-stable LMM with the scaled error constant of smallest magnitude is the Trapezoidal rule.*

Lecture 10 The Boundary Locus Method and Absolute Stability for systems

10.1 The Boundary Locus Method

In general, determining the region of absolute stability of an s-step LMM is not a trivial task, since one has to find the set of all $\hat{h}$ for which the stability polynomial satisfies the strict root condition ($|r_m|_{m\in[1,s]} < 1$). Instead, it is much more helpful to find the boundary of the region of absolute stability and then check whether $\hat{h}$ from a particular domain within the region of absolute stability satisfies the strict root condition. The Boundary Locus Method is the method for determining the stability region of s-step LMMs.

How to find this boundary? The strict root condition is a set of all roots of the stability polynomial which $|r| < 1$. The roots of the stability polynomial on the boundary satisfy the equation $|r| = 1$. The solution of this equation is $r = e^{is}$, where $i = \sqrt{-1}$ and $s \in [0, 2\pi)$. Thus, to find the boundary of the region of absolute stability one has to plug $r = e^{is}$ into the stability polynomial and solve it for $\hat{h} = \hat{h}(s)$. By varying s, we can plot a closed curve in the complex $\hat{h}$-plane. This curve divides the plane into different subregions some of which (where $\hat{h}$ such that $|r| < 1$) form the region of absolute stability.

Example 10.1 (Region and interval of absolute stability) *In this example, we are aiming at using the Boundary Locus Method to find the region and interval of absolute stability for the 2-step LMM*

$$x_{n+2} = x_{n+1} + hf_n. \tag{302}$$

First, we have to find the stability polynomial. It is given by

$$p(r) = r^2 - r - \hat{h}, \tag{303}$$

with the roots being $r_{1,2} = \dfrac{1 \pm \sqrt{1+4\hat{h}}}{2}$. *Substitution of* $r = e^{is}$ *into* (303) *leads to*

$$e^{2is} - e^{is} - \hat{h} = 0 \Rightarrow \hat{h} = e^{2is} - e^{is}, \quad s \in [0, 2\pi). \tag{304}$$

The plot of $\hat{h}(s)$ *is presented in Figure 7*

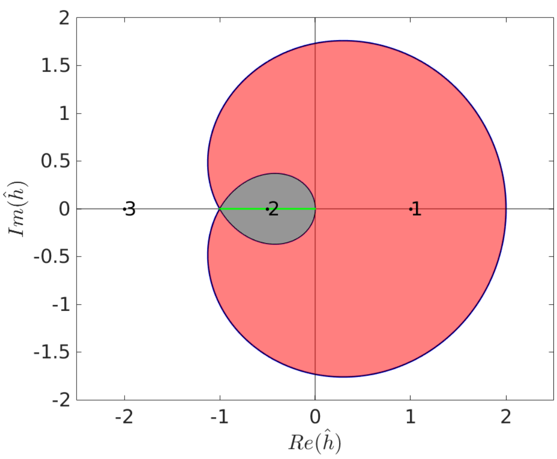

Figure 7: Shown is the locus of points for which the stability polynomial has the root $|r| = 1$. The region of absolute stability is shown in grey, and the interval of absolute stability is given by the solid green line.

The locus of points for which $|r| = 1$ *divides the plane into three subdomains: grey, red, and white. The question is which of these regions is the region of absolute stability. In order to figure that out, we take one value of* $\hat{h}$ *in each subregion marked by 1, 2, and 3 in Figure 7, plug them into the* *stability polynomial and compute the roots. If the roots for a particular value of $\hat{h}$ satisfy the strict root condition then this subregion is the region of absolute stability. For $\hat{h}_1 = 1$ (red region) and $\hat{h}_3 = -2$ (white region), the roots of stability polynomial $|r_{1,2}| > 1$, i.e. the strict root condition is violated. For $\hat{h}_2 = -1/2$ the roots are $|r_{1,2}| = 1/\sqrt{2}$, and therefore satisfy the strict root condition. Thus, the grey region is the region of absolute stability, and its intersection with the left-half plane gives the interval of absolute stability $(-1, 0)$ marked by the solid green line in Figure 7.*

In this particular example, the roots of the stability polynomial are not that complicated, and the interval of absolute stability can be computed by solving the inequality $|r_{1,2}| < 1$ for $\hat{h}$. However, in more complicated cases, like in the next example, the Boundary Locus Method is the method of choice. ▲

Example 10.2 (Region and interval of absolute stability) *Let us consider a 3-step LMM given by*

$$x_{n+3} = x_{n+2} + \frac{h}{12}(5f(t_{n+3}, x_{n+2} + \frac{h}{12}(23f_{n+2} - 16f_{n+1} + 5f_n)) + 8f_{n+2} - f_{n+1}). \tag{305}$$

When applied to the ODE $x' = \lambda x$, it gives

$$x_{n+3} = (1 + \frac{13}{12}\hat{h} + \frac{115}{144}\hat{h}^2)x_{n+2} - (\frac{\hat{h}}{12} + \frac{5}{9}\hat{h}^2)x_{n+1} + \frac{25}{144}\hat{h}^2 x_n. \tag{306}$$

The stability polynomial is then

$$p(r) = r^3 - (1 + \frac{13}{12}\hat{h} + \frac{115}{144}\hat{h}^2)r^2 + (\frac{\hat{h}}{12} + \frac{5}{9}\hat{h}^2)r - \frac{25}{144}\hat{h}^2. \tag{307}$$

One can find those $\hat{h}$ that satisfy the root condition for $p(r)$ by solving three inequalities $|r_i| < 1$, $i = 1, 2, 3$. It is not a trivial task. An easier and faster way is to apply the Boundary Locus Method. For doing so, we solve (307) *for $\hat{h}$:*

$$\hat{h}_{1,2} = \frac{(156r^2 - 12r) \pm 12r\sqrt{460r^3 - 611r^2 + 394r - 99}}{160r - 230r^2 - 50} \tag{308}$$

and substitute $r = e^{is}$ into (308)*:*

$$\hat{h}_{1,2} = \frac{(156e^{2is} - 12e^{is}) \pm 12e^{is}\sqrt{460e^{3is} - 611e^{2is} + 394e^{is} - 99}}{160e^{is} - 230e^{2is} - 50}. \tag{309}$$

Now, we can plot the roots $\hat{h}_1$ and $\hat{h}_2$:
It is worth noting that in order to find the region of absolute stability one should find the roots of the stability polynomial, $p(r)$, substitute $\hat{h}$'s from different regions of the domain into each root and check whether the roots satisfy the strict root condition. ▲

10.2 Absolute Stability for systems

Let us consider a system of ODEs

$$\mathbf{x}' = \mathbf{A}\mathbf{x}, \quad \mathbf{x}(t_0) = \boldsymbol{\alpha}, \quad t \in [t_0, t_N], \tag{310}$$

where $\mathbf{x} \in \mathbb{R}^m$, $\mathbf{A} \in \mathbb{R}^{m\times m}$, with $\mathbf{A}$ being a matrix of constant coefficients. In order to study absolute stability for systems of ODEs the matrix $\mathbf{A}$ has to be diagonalized. It is diagonalizable if it has m linearly independent eigenvectors $\{\mathbf{v}_i\}_{i\in[1,m]}$ with the corresponding eigenvalues $\{\lambda_i\}_{i\in[1,m]}$.

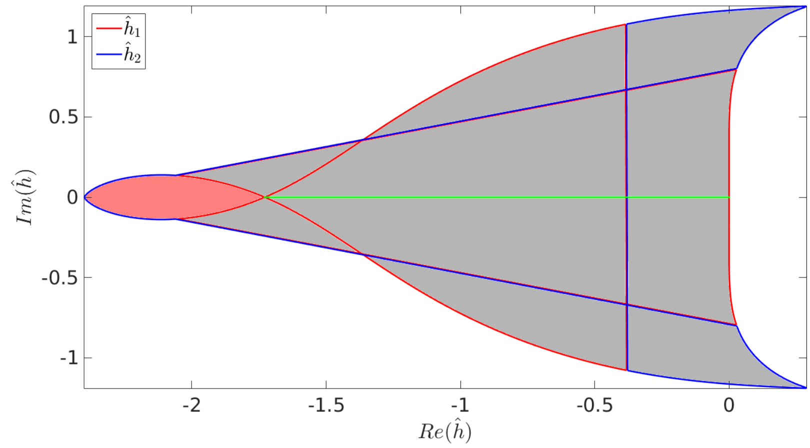

Figure 8: Shown is the locus of points for which the stability polynomial has the root $|r| = 1$. The region of absolute stability is shaded in grey, and the interval of absolute stability is given by the solid green line.

In this case, there exists a nonsingular matrix $\mathbf{V}$ with columns $\{\mathbf{v}_i\}_{i\in[1,m]}$ such that

$$\mathbf{V}^{-1}\mathbf{A}\mathbf{V} = \boldsymbol{\Lambda}, \tag{311}$$

where $\boldsymbol{\Lambda}$ is a diagonal matrix of eigenvalues.

Let us define $\mathbf{x} = \mathbf{V}\mathbf{y}$ and substitute it into (310)

$$\mathbf{V}\mathbf{y}' = \mathbf{A}\mathbf{V}\mathbf{x}. \tag{312}$$

Multiplying it by $\mathbf{V}^{-1}$ from the left we have

$$\mathbf{y}' = \mathbf{V}^{-1}\mathbf{A}\mathbf{V}\mathbf{y}, \tag{313}$$

or

$$\mathbf{y}' = \boldsymbol{\Lambda}\mathbf{y}. \tag{314}$$

Thus, the solution of system (310) reduced to the solution of a set of uncoupled scalar ODEs (314) of the type ($y_i' = \lambda_i y_i$, $i \in [1, m]$) which we have studied before. As long as $\mathbf{x}$ and $\mathbf{y}$ are connected through a linear transformation, they have the same long-time behaviour.

Definition 10.1 (Absolute stability for systems of ODEs) *An LMM is said to be absolutely stable for a diagonalizable system of ODEs $\mathbf{x}' = \mathbf{A}\mathbf{x}$ ($\mathbf{x} \in \mathbb{R}^m$, $\mathbf{A} \in \mathbb{R}^{m\times m}$, and all eigenvalues $Re(\lambda_i) < 0$, $i \in [1, m]$) if the numerical solution $\mathbf{x}_n \to 0$ as $n \to \infty$ for any choice of starting values.*

Example 10.3 (The interval of absolute stability for a system of ODEs) *Consider the system of ODEe*

$$\mathbf{x}' = \begin{pmatrix} 1 & 3 \\ -2 & -4 \end{pmatrix}\mathbf{x}, \quad \mathbf{x}(t_0) = \boldsymbol{\alpha}, \quad t \in [t_0, t_N]. \tag{315}$$

The eigenvalues of $\mathbf{A}$ are given by $\Lambda_{1,2} = \{-1, -2\}$. To find the interval of absolute stability, we have to choose a method to solve the system. Let it be the Euler method:

$$\mathbf{x}_{n+1} = \mathbf{x}_n + h\begin{pmatrix} -1 & 0 \\ 0 & -2 \end{pmatrix}\mathbf{x}_n. \tag{316}$$

We know that the interval of absolute stability for the Euler method is $\hat{h} \in (-2, 0)$. For the system of equations, we have to find the smallest interval of absolute stability to ensure the method is *absolutely stable. In this particular case, we have two intervals (since the system is of order 2): the one for $\lambda_1 = -1$ is $h \in (0, 2)$ and the other for $\lambda_2 = -2$ is $h \in (0, 1)$. Hence, the smallest one is $h \in (0, 1)$. Only if the time step is taken from the smallest interval the method will be absolutely stable, otherwise the method will suffer from an instability like the one we observed when studied scalar ODEs.*

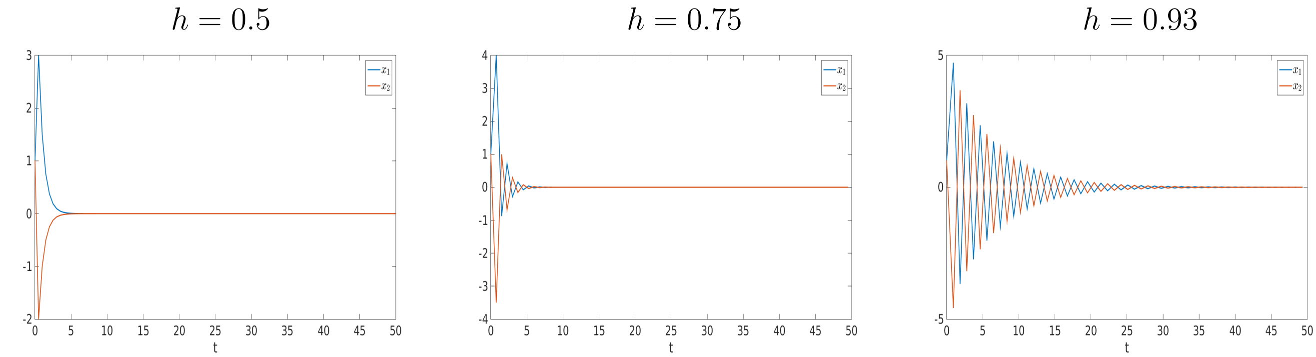

Figure 9: Shown is the numerical solution $\mathbf{x} = (x_1, x_2)$ computed with the Euler method with different time steps vs time.

Note that the closer the time step to the border of the region of absolute stability is, the more time it takes for the solution to get to zero. For example, the time step $h = 0.5$ is in the middle of the interval of absolute stability and therefore the method converges to zero quite rapidly, while for the time step much closer to the border, $h = 0.93$, it takes more time for the method to converge.
▲

Lecture 11 Implicit LMMs and nonlinear equations

Our previous study and experience with developing numerical methods shows that implicit methods usually allow to chose the time step based on grounds of accuracy rather than stability. However, it does not come for free, especially for nonlinear systems of ODEs, since, in this case a system of nonlinear algebraic equations has to be solved at each time step to find the numerical solution.

Let us consider the general s-step implicit LMM

$$\sum_{m=0}^{s} \alpha_m x_{n+m} = h \sum_{m=0}^{s} \beta_m f(t_{n+m}, x_{n+m}), \quad n = 0, 1, 2, \dots \quad , \text{ where } \alpha_m, \beta_m = const \in \mathbb{R}, \text{ and } \alpha_s = 1 \tag{317}$$

with $\beta_s \neq 0$. The condition $\beta_s \neq 0$ makes the method implicit (the right hand side depends on the solution x_{n+s}). We can rewrite (317) in the following form

$$x_{n+s} = h\beta_s f(t_{n+s}, x_{n+s}) + g_n, \quad g_n := h \sum_{m=0}^{s-1} \beta_m f(t_{n+m}, x_{n+m}) - \sum_{m=0}^{s-1} \alpha_m x_{n+m}, \quad n = 0, 1, 2, \dots \tag{318}$$

There are many numerical methods which can be used to solve nonlinear systems. In this course we will consider the following three:

- Fixed Point Iteration method
- The Newton method (also called Newton-Raphson method)
- Predictor-Corrector methods

11.1 Fixed Point Iteration method

The idea behind the Fixed Point Iteration method is to take a nonlinear equation, or a system of nonlinear equations, (in our case it is equation (318)), choose an initial guess x^0_{n+s}, and iterate the equation

$$x^{i+1}_{n+s} = h\beta_s f(t_{n+s}, x^i_{n+s}) + g_n. \tag{319}$$

until convergence or $i = M$; i is the iteration number, and M is the maximum number of iterations. As a stopping criterion, one can take, say $\|x^{i+1}_{n+s} - x^i_{n+s}\|_2 / \|x^{i+1}_{n+s}\|_2 < \epsilon$, where ϵ is a given tolerance. Typically, the Fixed Point Iteration method requires very small time steps to converge and convergence is slow. This may be seen as a drawback of the method. On the other hand, it is easy and straightforward to implement.

Example 11.1 (Backward Euler method and Fixed Point Iteration) *As an example of use of the Fixed Point Iteration method, let us consider the nonlinear initial value problem*

$$x' = 2x(1-x)t, \quad x(t_0) = \alpha, \quad t \in [t_0, t_N]. \tag{320}$$

As a numerical method, let us take the Backward Euler method ($x_{n+1} = x_n + hf_{n+1}$):

$$x_{n+1} = x_n + h2x_{n+1}(1 - x_{n+1})t_{n+1}. \tag{321}$$

Then, the Fixed Point Iteration for (321) *is given by*

$$x^{i+1}_{n+1} = x_n + h2x^i_{n+1}(1 - x^i_{n+1})t_{n+1}, \quad i = 0, 1, 2, \dots, M. \tag{322}$$

Note that equation (322) *has to be solved at each time step.* ▲

11.2 The Newton method

The Newton method for a nonlinear equation

$$\mathbf{F}(\mathbf{x}) = 0$$

is given by

$$\mathbf{x}^{i+1} = \mathbf{x}^i - (\mathbf{F}'(\mathbf{x}^i))^{-1}\mathbf{F}(\mathbf{x}^i), \quad i = 0, 1, 2, \ldots, M\,, \tag{323}$$

where $\mathbf{F}'$ is the Jacobian of $\mathbf{F}$, and i is the iteration number, and M is the maximum number of iterations. The Newton method has a quadratic converge rate provided that the initial guess x^0 is sufficiently close to the root. However, it requires a Jacobian of the function and a system of linear equation to be solved for each iteration step.

The Newton method for the nonlinear equation (318) reads

$$x^{i+1}_{n+s} = x^i_{n+s} - (\mathbf{F}'(x^i_{n+s}))^{-1}\mathbf{F}(x^i_{n+s}), \quad i = 1, 2, \ldots \quad , \tag{324}$$

where

$$\mathbf{F}(x^i_{n+s}) = x^i_{n+s} - (h\beta_s f(t_{n+s}, x^i_{n+s}) + g_n)$$

and

$$g_n := h\sum_{m=0}^{s-1}\beta_m f(t_{n+m}, x_{n+m}) - \sum_{m=0}^{s-1}\alpha_m x_{n+m}, \quad n = 0, 1, 2, \ldots$$

The stopping criterion for the Newton method is the same as for the Fixed Point Iteration.

Example 11.2 (Backward-Euler and Newton's methods for a system of ODEs) *Let us consider the following nonlinear system of ODEs:*

$$\begin{cases} x' = -\,2y^3, \\ y' = 2x - y^4, \quad x(0) = y(0) = 1. \end{cases} \tag{325}$$

The Backward-Euler method is given by

$$\begin{cases} x_{n+1} = x_n - h2y^3_{n+1}, \\ y_{n+1} = y_n + h(2x_{n+1} - y^4_{n+1}), \end{cases} \tag{326}$$

$$\mathbf{F}\begin{pmatrix} x_{n+1} \\ y_{n+1} \end{pmatrix} = \begin{pmatrix} x_{n+1} - (x_n - h2y^3_{n+1}) \\ y_{n+1} - (y_n + h(2x_{n+1} - y^4_{n+1})) \end{pmatrix} \tag{327}$$

with the Jacobian

$$\mathbf{F}'\begin{pmatrix} x_{n+1} \\ y_{n+1} \end{pmatrix} = \begin{pmatrix} 1 & 6hy^2_{n+1} \\ -2h & 1 + 4hy^3_{n+1} \end{pmatrix}. \tag{328}$$

Hence, the Newton method for (325) *is:*

$$\begin{pmatrix} x^{i+1}_{n+1} \\ y^{i+1}_{n+1} \end{pmatrix} = \begin{pmatrix} x^i_{n+1} \\ y^i_{n+1} \end{pmatrix} - \mathbf{F}'\begin{pmatrix} x^i_{n+1} \\ y^i_{n+1} \end{pmatrix}^{-1}\mathbf{F}\begin{pmatrix} x^i_{n+1} \\ y^i_{n+1} \end{pmatrix}, \quad i = 0, 1, 2, \ldots, M. \tag{329}$$

Notice that the system of equations (329) *has to be solved at each time step.* ▲

11.3 Predictor-corrector methods

The basic principle underlaying predictor-corrector methods is to predict the numerical solution with the predictor, and then correct it with the corrector. There different types of predictor-corrector method, in this course we consider two types: $P(EC)^kE$ (Predict Evaluate Correct Evaluate) and $P(EC)^k$ (Predict Evaluate Correct); k is an non-negative integer. As a predictor and corrector we will use the method studied in the course.

11.3.1 $P(EC)^kE$ (Predict Evaluate Correct Evaluate)

For the $P(EC)^kE$ method, we use an s-step explicit LMM as a predictor:

$$\sum_{m=0}^{s} \alpha_m x_{n+m} = h \sum_{m=0}^{s-1} \beta_m f(t_{n+m}, x_{n+m}), \tag{330}$$

and an s-step implicit LMM as a corrector

$$\sum_{m=0}^{s} \gamma_m x_{n+m} = h \sum_{m=0}^{s} \delta_m f(t_{n+m}, x_{n+m}), \quad \delta_s \neq 0. \tag{331}$$

Combining (330) and (331) into the $P(EC)^kE$ method gives

$$\begin{aligned}
&\text{Predict} \quad x^0_{n+s} = -\sum_{m=0}^{s-1} \alpha_m x^k_{n+m} + h \sum_{m=0}^{s-1} \beta_m f(t_{n+m}, x^k_{n+m}) \\
&\left(\begin{aligned} &\text{Evaluate} \quad f^i(t_{n+s}, x^i_{n+s}) \\ &\text{Correct} \quad x^{i+1}_{n+s} = -\sum_{m=0}^{s-1} \gamma_m x^k_{n+m} + h(\delta_s f^i(t_{n+s}, x^i_{n+s}) + \sum_{m=0}^{s-1} \delta_m f(t_{n+m}, x^k_{n+m})) \end{aligned} \right)_{i=0,1,2,\dots,k-1} \\
&\text{Evaluate} \quad f(t_{n+s}, x^k_{n+s})
\end{aligned} \tag{332}$$

where $n = 0, 1, 2, \dots$.

11.3.2 $P(EC)^k$ (Predict Evaluate Correct)

For the $P(EC)^k$ method, we use the same predictor and corrector as for $P(EC)^kE$.

$$\begin{aligned}
&\text{Predict} \quad x^0_{n+s} = -\sum_{m=0}^{s-1} \alpha_m x^k_{n+m} + h \sum_{m=0}^{s-1} \beta_m f(t_{n+m}, x^{k-1}_{n+m}) \\
&\left(\begin{aligned} &\text{Evaluate} \quad f^i(t_{n+s}, x^i_{n+s}) \\ &\text{Correct} \quad x^{i+1}_{n+s} = -\sum_{m=0}^{s-1} \gamma_m x^k_{n+m} + h(\delta_s f^i(t_{n+s}, x^i_{n+s}) + \sum_{m=0}^{s-1} \delta_m f(t_{n+m}, x^{k-1}_{n+m})) \end{aligned} \right)_{i=0,1,2,\dots,k-1}
\end{aligned} \tag{333}$$

where $n = 0, 1, 2, \dots$.

The local truncation error, the global error, the region and interval of absolute stability for predictor-corrector methods can be calculated in the same way as for the methods considered in the course. To make it clearer we will address all this in the series of examples below. As an example of predictor-corrector methods, we consider two methods based on explicit/implicit methods of the same order of accuracy.

Example 11.3 (Forward-Backward Euler method as PECE) *In this example, we consider the*

Forward-Backward Euler method, with the Forward Euler method

$$x_{n+1} = x_n + hf_n$$

being the predictor, and the Backward Euler method

$$x_{n+1} = x_n + hf_{n+1}$$

being the corrector. Then, the PECE method based on the Forward-Backward Euler pair is given by

$$\begin{array}{ll} \textbf{P}\textit{redict} & \hat{x}_{n+1} = x_n + hf(t_n, x_n) \\ \textbf{E}\textit{valuate} & f(t_{n+1}, \hat{x}_{n+1}) \\ \textbf{C}\textit{orrect} & x_{n+1} = x_n + hf(t_{n+1}, \hat{x}_{n+1}) \\ \textbf{E}\textit{valuate} & f(t_{n+1}, x_{n+1}) \end{array} \tag{334}$$

where $n = 0, 1, 2, \ldots$

The local truncation error

Another question we will study in this example is how to compute the local truncation error of the predictor-corrector method. For this, we take the continuous form of the predictor

$$\hat{x}(t_{n+1}) = x(t_n) + hf(t_n, x(t_n))$$

and use the difference equation $x' = f(t, x)$ *to replace the term* $f(t_n, x(t_n))$*:*

$$\hat{x}(t_{n+1}) = x(t_n) + hx'(t_n). \tag{335}$$

Then, we take the continuous form of the corrector

$$x(t_{n+1}) = x(t_n) + hf(t_{n+1}, \hat{x}(t_{n+1}))$$

and replace $f(t_{n+1}, \hat{x}(t_{n+1}))$ *with its derivative* $\hat{x}'(t_{n+1})$*:*

$$x(t_{n+1}) = x(t_n) + h\hat{x}'(t_{n+1}). \tag{336}$$

Finally, we plug (335) *into* (336) *to get*

$$\begin{aligned} x(t_{n+1}) =& x(t_n) + h(x'(t_n) + hx''(t_n)) \\ =& x(t_n) + hx'(t_n) + h^2x''(t_n). \end{aligned} \tag{337}$$

The last step is to calculate the difference between the Taylor expansion of the exact solution:

$$x(t+h) = x(t) + hx'(t) + \frac{h^2}{2}x''(t) + \mathcal{O}(h^3)$$

and its approximation given by (337)*:*

$$x(t+h) - x(t_{n+1}) = \mathcal{O}(h^2).$$

This gives the local truncation error of order 2.

The global error and convergence

We can prove that the method is convergent in the same way as we did for the Euler method. **Can**

you do it?

The region and interval of absolute stability

The region and interval of absolute stability can be computed by applying the PECE method to the equation $x' = \lambda x$.

$$\begin{aligned}\hat{x}_{n+1} =& x_n + \hat{h}x_n,\\ x_{n+1} =& x_n + hf(t_{n+1}, \hat{x}_{n+1}) = x_n + \hat{h}x_n + \hat{h}^2 x_n.\end{aligned} \tag{338}$$

The stability polynomial for the PECE is

$$p(r) = r - (1 + \hat{h} + \hat{h}^2). \tag{339}$$

Using the Boundary Locus method we compute the region of absolute stability and present the results in Figure 10.

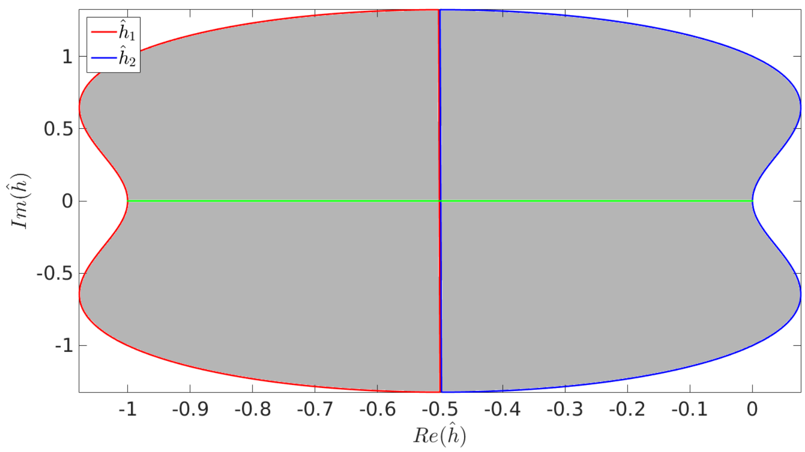

Figure 10: The region of absolute stability is shown in grey, and the interval of absolute stability is given by the solid green line.

The interval of absolute stability is $\hat{h} \in (-1, 0)$. ▲

Example 11.4 (AB(2)-Trapezoidal rule as PECE) *In this example, we use the AB(2) method*

$$x_{n+1} = x_n + \frac{h}{2}(3f_n - f_{n-1})$$

as a predictor and the Trapezoidal rule

$$x_{n+1} = x_n + \frac{h}{2}(f_{n+1} + f_n)$$

as a corrector, which gives the following PECE method:

$$\begin{aligned}&\textbf{P}\textit{redict} && \hat{x}_{n+1} = x_n + \frac{h}{2}(3f(t_n, x_n) - f(t_{n-1}, x_{n-1}))\\ &\textbf{E}\textit{valuate} && f(t_{n+1}, \hat{x}_{n+1})\\ &\textbf{C}\textit{orrect} && x_{n+1} = x_n + \frac{h}{2}(f(t_{n+1}, \hat{x}_{n+1}) + f(t_n, x_n))\\ &\textbf{E}\textit{valuate} && f(t_{n+1}, x_{n+1})\end{aligned} \tag{340}$$

where $n = 0, 1, 2, \ldots$

The local truncation error
In order to compute the local truncation error of the PECE method, we follow the same steps as in the previous example. Namely, we rewrite the predictor as

$$\hat{x}(t_{n+1}) = x(t_n) + \frac{h}{2}(3f(t_n, x(t_n)) - f(t_{n-1}, x(t_{n-1})))$$

and use the difference equation $x' = f(t, x)$ to replace f:

$$\hat{x}(t_{n+1}) = x(t_n) + \frac{h}{2}(3x'(t_n) - x'(t_{n-1})). \tag{341}$$

Then, we take the continuous form of the corrector

$$x(t_{n+1}) = x(t_n) + \frac{h}{2}(f(t_{n+1}, \hat{x}(t_{n+1})) + f(t_n, x(t_n)))$$

and replace $f(t_{n+1}, \hat{x}(t_{n+1}))$ with $\hat{x}'(t_{n+1})$ and $f(t_n, x(t_n))$ with $x'(t_n)$:

$$x(t_{n+1}) = x(t_n) + \frac{h}{2}(\hat{x}'(t_{n+1}) + x'(t_n)). \tag{342}$$

Substitution of (341) *into* (342) *leads to*

$$\begin{aligned} x(t_{n+1}) =& x(t_n) + \frac{h}{2}(x'(t_n) + \frac{h}{2}(3x''(t_n) - x''(t_{n-1})) + x'(t_n)) \\ & \textit{using the Taylor expansion of } x''(t_{n-1}) = x''(t_n) - hx'''(t_n) + \frac{h^2}{2}x'''' + \mathcal{O}(h^3) \textit{ gives:} \\ =& x(t_n) + hx'(t_n) + \frac{h^2}{2}x''(t_n) + \frac{h^3}{4}x'''(t_n) + \mathcal{O}(h^4). \end{aligned} \tag{343}$$

We can now calculate the difference between the Taylor expansion of the exact solution and its approximation given by (343)*:*

$$x(t+h) - x(t_{n+1}) = \mathcal{O}(h^3).$$

Thus, the local truncation error of order 3.

The global error and convergence
We can prove that the method is convergent in the same way as we did for the Euler method. **Can you do it?**

The region and interval of absolute stability
The region and interval of absolute stability can be computed by applying the PECE method to the equation $x' = \lambda x$.

$$\begin{aligned} \hat{x}_{n+1} =& x_n + \frac{\hat{h}}{2}(3x_n - x_{n-1}), \\ x_{n+1} =& x_n + \frac{h}{2}(f(t_{n+1}, \hat{x}_{n+1}) + f(t_n, x_n)) \\ =& x_n + \hat{h}x_n + \frac{\hat{h}^2}{4}(3x_n - x_{n-1}). \end{aligned} \tag{344}$$

Shifting the indices to the right gives

$$x_{n+2} = x_{n+1} + \hat{h}x_{n+1} + \frac{\hat{h}^2}{4}(3x_{n+1} - x_n). \tag{345}$$

Thus, the stability polynomial for the PECE is

$$p(r) = r^2 - (1 + \hat{h} + \frac{3\hat{h}^2}{4})r + \frac{\hat{h}^2}{4}. \tag{346}$$

We use the Boundary Locus method to compute the region and interval of absolute stability (Figure 11).

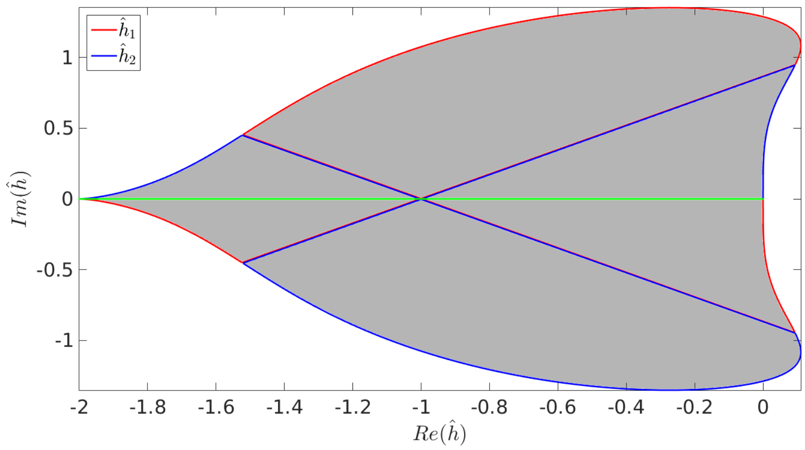

Figure 11: The region of absolute stability is shown in grey, and the interval of absolute stability is given by the solid green line.

The interval of absolute stability is $\hat{h} \in (-2, 0)$. ▲

As you probably have noticed from the examples, the interval of absolute stability of predictor-corrector methods is closer to that of the predictor. Sometimes, like in the examples above, the interval of absolute stability of the corrector can be significantly shrunk. For instance, the Trapezoidal rule is an A-stable method (i.e., the time steps can be chosen based on accuracy not stability), while the interval of absolute stability of the AB(2) method is $\hat{h} \in (-1, 0)$. However, the interval of absolute stability of the PECE method is only $\hat{h} \in (-2, 0)$.

Lecture 12 Improved approximations

Let us think about how to improve the accuracy of predictor-corrector methods. An obvious idea is to decrease the time step. But, in this case the method will take more time to compute the solution. Another idea is to use higher-order methods for the predictor-corrector. This will take some extra work to implement new higher-order methods, but it will definitely pay off in the long run. A less trivial approach is to manipulate the local truncation error of both the predictor and corrector in such a way that the accuracy of the predictor-corrector pair is higher. This approach is known as the Milne estimator or Milne's device.

12.1 The Milne estimator

Let us assume we have to develop a predictor-corrector method using a pair of s-step LMMs of order $\mathcal{O}(h^{p+1})$ as a predictor and corrector. Then, the difference

$$x(t_{n+s}) - \hat{x}_{n+s} = h^{p+1}\hat{C}_{p+1}\frac{d^{p+1}x}{dt^{p+1}} + \mathcal{O}(h^{p+2}) \tag{347}$$

is the error of the predictor, and the difference

$$x(t_{n+s}) - x_{n+s} = h^{p+1}C_{p+1}\frac{d^{p+1}x}{dt^{p+1}} + \mathcal{O}(h^{p+2}) \tag{348}$$

is the error of the corrector. Here, $x(t_{n+s})$ is the exact solution, $\hat{x}_{n+s}$ and x_{n+s} are the numerical solutions computed with the predictor and the corrector, respectively, and $\hat{C}_{p+1}$ and C_{p+1} are error constants.

Ignoring $\mathcal{O}(h^{p+2})$ and subtracting (348) from (347) we obtain

$$x_{n+s} - \hat{x}_{n+s} = h^{p+1}(\hat{C}_{p+1} - C_{p+1})\frac{d^{p+1}x}{dt^{p+1}}. \tag{349}$$

Multiplication of (349) by $\dfrac{C_{p+1}}{\hat{C}_{p+1} - C_{p+1}}$ gives

$$\frac{C_{p+1}}{\hat{C}_{p+1} - C_{p+1}}(x_{n+s} - \hat{x}_{n+s}) = h^{p+1}C_{p+1}\frac{d^{p+1}x}{dt^{p+1}}. \tag{350}$$

The right hand side of (350) is the local truncation error of the corrector. Thus, we have an estimation of the leading order error in the corrector. This is known as **the Milne estimator**.

Using (350) in (348) results in

$$x(t_{n+s}) - x_{n+s} = \frac{C_{p+1}}{\hat{C}_{p+1} - C_{p+1}}(x_{n+s} - \hat{x}_{n+s}) + \mathcal{O}(h^{p+2}) \tag{351}$$

or

$$x(t_{n+s}) - \widetilde{x}_{n+s} = \mathcal{O}(h^{p+2}), \quad \widetilde{x}_{n+s} := x_{n+s} + \frac{C_{p+1}}{\hat{C}_{p+1} - C_{p+1}}(x_{n+s} - \hat{x}_{n+s}). \tag{352}$$

The new solution of the corrector, $\widetilde{x}_{n+s}$, is one order more accurate than the old one x_{n+s}. Thus, to increase the accuracy by an order of magnitude, one has to only know the error constants of the predictor and corrector. Obviously, it is way easier than developing higher-order methods. Note that the Milne estimator can be used for methods with the same order of the local truncation error.

Example 12.1 (The Milne estimator for the Forward-Backward Euler PECE method) *Let* *us consider the PECE method based on the Forward-Backward Euler pair.*

$$
\begin{array}{ll}
\textbf{P}\textit{redict} & \hat{x}_{n+1} = x_n + hf(t_n, x_n) \\
\textbf{E}\textit{valuate} & f(t_{n+1}, \hat{x}_{n+1}) \\
\textbf{C}\textit{orrect} & x_{n+1} = x_n + hf(t_{n+1}, \hat{x}_{n+1}) \\
\textbf{E}\textit{valuate} & f(t_{n+1}, x_{n+1})
\end{array}
\tag{353}
$$

where $n = 0, 1, 2, \ldots$

The error constant of the predictor and corrector can be computed from the local truncation error as follows:

$$x(t_{n+1}) - \hat{x}_{n+1} = h^2 \hat{C}_2 x''(t_n), \quad \hat{C}_2 := \frac{1}{2},$$

and

$$x(t_{n+1}) - x_{n+1} = h^2 C_2 x''(t_n), \quad C_2 := -\frac{1}{2}.$$

The local truncation error of the predictor-corrector method in this form is $\mathcal{O}(h^2)$ *as we have shown it before. If we use the Milne estimator, then in accordance with* (352) *the new solution of the corrector becomes:*

$$\widetilde{x}_{n+1} := x_{n+1} - \frac{1}{2}(x_{n+1} - \hat{x}_{n+1}).$$

Therefore, the local truncation error of the new solution becomes

$$x(t_{n+1}) - \widetilde{x}_{n+1} = \mathcal{O}(h^3).$$

As you can see, the Milne estimator is easy to implement. and more importantly, its computational complexity (the number of arithmetic operations) is negligible compared with the predictor-corrector method. ▲

Lecture 13 Extrapolation

Let us consider an initial value problem

$$x' = f(t, x), \quad x(0) = x_0, \quad t \in [t_0, t_N]. \tag{354}$$

Let $z(t, h)$ denote a numerical approximation of the exact solution $x(t)$ of (354) at time t using step size h. Suppose the global error of a numerical method has an expansion in powers of h of the form

$$z(t, h) = x(t) + \sum_{i=1}^{q} c_i(t)h^i + O(h^{q+1}), \tag{355}$$

with $c_j(t) = 0$, $j = 1, 2, \dots, p-1$ for a method of order $p < q$.

It is known that every explicit one-step method has such an expansion when $x(t)$ is smooth. Implicit Runge-Kutta methods may also have expansions of this form. The idea of extrapolation is to combine solutions $z(t, h_j)$ computed with different step sizes h_j with $j = 0, 1, \dots$ so that successive terms of the error expansion (355) are eliminated, thus, resulting in a higher-order approximation.

Let us consider

$$z(t, h) = x(t) + c_1 h + c_2 h^2 + \dots \tag{356}$$

and compute two solutions at time t using time steps h_0 and $h_0/2$, namely

$$z(t, h_0) = x(t) + c_1 h_0 + c_2 h_0^2 + \dots \tag{357a}$$

$$z\left(t, \frac{h_0}{2}\right) = x(t) + c_1\frac{h_0}{2} + c_2\frac{h_0^2}{4} + \dots \tag{357b}$$

We have to find $x(t)$ and c_1 from system (357). Subtracting the second equation from the first one, we eliminate $x(t)$ and obtain

$$c_1\frac{h_0}{2} = z(t, h_0) - z\left(t, \frac{h_0}{2}\right) - c_2\frac{3}{4}h_0^2 + \dots \tag{358}$$

Substituting (358) into, say, the second expansion yields

$$2z\left(t, \frac{h_0}{2}\right) - z(t, h_0) = x(t) - \frac{c_2}{2}h_0^2 + \dots. \tag{359}$$

Thus, $2z\,(t, h_0/2) - z(t, h_0)$ provides a higher-order (namely, $O(h^2)$) approximation of $x(t)$ than either $z(t, h_0)$ or $z\,(t, h_0/2)$.

The same results can be obtained by approximating $z(t, h)$ by a linear polynomial $R_1(t, h)$ that interpolates $z(t, h)$ between $h = h_0$ and $h = h_0/2$:

$$\text{Richardson's extrapolationn:} \qquad R_1(t, h) = \alpha_0(t) + \alpha_1(t)h. \tag{360}$$

The extrapolation condition requires

$$z(t, h_0) = \alpha_0(t) + \alpha_1(t)h_0, \tag{361}$$

and

$$z\left(t, \frac{h_0}{2}\right) = \alpha_0(t) + \alpha_1(t)\frac{h_0}{2}. \tag{362}$$

Therefore,

$$
\begin{aligned}
\alpha_1(t) &= \frac{2}{h_0}(z(t,h_0) - z\left(t, \frac{h_0}{2}\right)),\\
\alpha_0(t) &= 2z(t, \frac{h_0}{2}) - z\left(t, h_0\right),\\
R_1(t,0) = \alpha_0 &= 2z\left(t, \frac{h_0}{2}\right) - z(t,h_0).
\end{aligned}
\tag{363}
$$

Richardson's extrapolation is called extrapolation, because we use old values $z(t, h_0/2)$, $z(t, h_0)$ to obtain an extrapolated value at $h \to 0$.

Example 13.1 (Euler method with Richardson's extrapolation) *Let us compute the local truncation error for the Euler method using Richardson's extrapolation. We write the Euler method in the continuous form, since we have to differentiate the right hand side and do some substitutions. The Euler method for $x' = f(t,x)$ is given by*

$$x(t+h) = x(t) + hf(t, x(t)) = x(t) + hx'(t) =: z(t,h).$$

In order to compute $R_1(t,0)$ in (363), we have to compute $z(t, h/2)$ at the same moment as $z(t,h)$, i.e. at $t+h$. Since $z(t,h/2)$ is the solution computed with step $h/2$, therefore we have to compute it twice to get it at time $t+h$. At time $t+h/2$ we have

$$x(t+\frac{h}{2}) = x(t) + \frac{h}{2}x'(t).$$

At time $t+h$ we get

$$x(t+h) = x(t+\frac{h}{2}) + \frac{h}{2}x'(t+\frac{h}{2}). \tag{364}$$

Upon substitution of $x(t+h/2)$ and $x'(t+h/2)$ into (364) we get

$$
\begin{aligned}
x(t+h) =& x(t) + \frac{h}{2}x'(t) + \frac{h}{2}(x'(t) + \frac{h}{2}x''(t))\\
=& x(t) + hx'(t) + \frac{h^2}{4}x''(t) =: z(t, \frac{h}{2}).
\end{aligned}
\tag{365}
$$

We can now compute the new solution based on the Richardson extrapolation as follows:

$$
\begin{aligned}
\widetilde{x}(t+h) &= 2z(t, \frac{h}{2}) - z(t,h)\\
&= x(t) + hx'(t) + \frac{h^2}{2}x''(t).
\end{aligned}
\tag{366}
$$

Thus, the local truncation error is given by

$$x(t_{n+1}) - \widetilde{x}(t+h) = O(h^3),$$

with $x(t_{n+1})$ being the exact solution. As we can see, the Richardson extrapolation gives a more accurate solution compared with the Euler method; the local truncation error of the Euler method is of order 2. ▲

In fact, $z(t, h_0)$ can be not only the solution of an ODE, but also an approximated value of, say, a derivative. For example, let us use Richardson's extrapolation to improve the accuracy of a finite difference approximation of $\sin(x)$ at point $x = 1$.

Let us use the forward difference approximation of the derivative:

$$\big(\sin(x)\big)'_x \approx \frac{\sin(x+h)-\sin(x)}{h} = z(x,h). \tag{367}$$

For $h=0.5$ and $h/2=0.25$ we have:

$$\begin{aligned} z(x,h) =& \frac{\sin(1.5)-\sin(1)}{0.5} = 0.31205, \\ z(x,h/2) =& \frac{\sin(1.25)-\sin(1)}{0.25} = 0.43005. \end{aligned} \tag{368}$$

If we compare these values with the exact value

$$\big(\sin(x)\big)'_x|_{x=1} = \cos(x)|_{x=1} = 0.540302, \tag{369}$$

we can see that neither of them is close enough to the exact derivative. However, if we use the Richardson extrapolation

$$R_1(t,0) = 2z\left(t,\frac{h}{2}\right) - z(t,h) = 0.548061, \tag{370}$$

the accuracy increases significantly.

For a more accurate solution, one can compute the solution at three different time steps, say h, $h/2$, $h/4$ and find $x(t)$ from the third-order system

$$\begin{aligned} z(t,h) &= x(t) + c_1 h + c_2 h^2 + O\left(h^3\right), \\ z\left(t,\frac{h}{2}\right) &= x(t) + c_1\frac{h}{2} + c_2\left(\frac{h}{2}\right)^2 + O\left(h^3\right), \\ z\left(t,\frac{h}{4}\right) &= x(t) + c_1\frac{h}{4} + c_2\left(\frac{h}{4}\right)^2 + O\left(h^3\right). \end{aligned} \tag{371}$$

Solving (371) for $x(t)$, $c1$, and c_2 we have

$$x(t) = \frac{z(t,h)}{3} - 2z\left(t,\frac{h}{2}\right) + \frac{8}{3}z\left(t,\frac{h}{4}\right) + O(h^3). \tag{372}$$

This gives the accuracy of order $O(h^3)$. Thus, if one computes $z(t,h)$, $z\left(t,\frac{h}{2}\right)$, $z\left(t,\frac{h}{4}\right)$ with, say the Euler method, and then compute the numerical solution as

$$x(t) = \frac{z(t,h)}{3} - 2z\left(t,\frac{h}{2}\right) + \frac{8}{3}z\left(t,\frac{h}{4}\right), \tag{373}$$

it will give the accuracy of order $O(h^4)$.

Example 13.2 (Euler method of improved accuracy) *Let us compute the local truncation error of the Euler method using approximation* (373). *As above, we write the Euler method in the continuous form:*

$$x(t+h) = x(t) + hf(t,x(t)) = x(t) + hx'(t) =: z(t,h).$$

Function $z(t, h/2)$ at time $t+h$ is given by

$$x(t+h) = x(t) + hx'(t) + \frac{h^2}{4}x''(t) =: z(t, \frac{h}{2}). \tag{374}$$

As with $z(t, h/2)$, function $z(t, h/4)$ has to be computed at time $t+h$ to be used in (373)*. Namely, at time $t + h/4$ we have*

$$x(t+\frac{h}{4}) = x(t) + \frac{h}{4}x'(t).$$

At time $t + 2h/4$ we have

$$\begin{aligned} x(t+2\frac{h}{4}) &= x(t+\frac{h}{4}) + \frac{h}{4}x'(t+\frac{h}{4}) \\ &\quad \textit{Substitution of } x(t+\frac{h}{4}) \textit{ and } x'(t+\frac{h}{4}) \textit{ gives} \\ &= x(t) + \frac{h}{4}x'(t) + \frac{h}{4}(x'(t) + \frac{h}{4}x''(t)) \\ &= x(t) + \frac{h}{2}x'(t) + \frac{h^2}{16}x''(t). \end{aligned} \tag{375}$$

At time $t + 3h/4$ we get

$$\begin{aligned} x(t+3\frac{h}{4}) &= x(t+2\frac{h}{4}) + \frac{h}{4}x'(t+2\frac{h}{4}) \\ &= x(t) + \frac{3}{4}hx'(t) + \frac{3}{16}h^2x''(t) + \frac{h^3}{64}x'''(t). \end{aligned} \tag{376}$$

At time $t + 4h/4$ we obtain

$$\begin{aligned} x(t+4\frac{h}{4}) &= x(t+3\frac{h}{4}) + \frac{h}{4}x'(t+3\frac{h}{4}) \\ &= x(t) + hx'(t) + \frac{6}{16}h^2x''(t) + \frac{4}{64}h^3x'''(t) + \frac{h^4}{256}x''''(t) =: z(t, h/4). \end{aligned} \tag{377}$$

We can now compute the new solution as follows:

$$\begin{aligned} \widetilde{x}(t+h) &= \frac{z(t,h)}{3} - 2z\left(t, \frac{h}{2}\right) + \frac{8}{3}z\left(t, \frac{h}{4}\right) \\ &= x(t) + hx'(t) + \frac{h^2}{2}x''(t) + \frac{h^3}{6}x'''(t) + \frac{h^4}{96}x''''(t). \end{aligned} \tag{378}$$

Thus, the local truncation error is given by

$$x(t_{n+1}) - \widetilde{x}(t+h) = O(h^4),$$

where $x(t_{n+1})$ is the exact solution. Now, the new solution is two orders of magnitude more accurate compared with the solution computed with the Euler method. ▲

Lecture 14 Runge-Kutta methods

14.1 The first Runge-Kutta method

Let us consider the following initial value problem with the right hand side depending only on t

$$x' = f(t), \quad x(t_0) = x_0, \quad t \in [t_0, t_N]. \tag{379}$$

The solution to (379) is given by

$$x(t_N) = x(t_0) + \int_{t_0}^{t_N} f(t)dt. \tag{380}$$

The integral can be computed with the standard mid-point rule:

$$\int_{t_0}^{t_N} f(t)dt = (t_N - t_0) f\left(\frac{t_0 + t_N}{2}\right), \tag{381}$$

or with the repeated mid-point rule:

$$\int_{t_0}^{t_N} f(t)dt = \sum_{n=0}^{N-1} \int_{t_n}^{t_{n+1}} f(t)dt \approx \sum_{n=0}^{N-1} hf\left(\frac{t_n + t_{n+1}}{2}\right), \ h := t_{n+1} - t_n. \tag{382}$$

The application of the repeated mid-point rule to (380) gives

$$x(t_0 + h) = x_0 + hf\left(t_0 + \frac{h}{2}\right), \tag{383a}$$

$$x(t_1 + h) = x_1 + hf\left(t_1 + \frac{h}{2}\right), \tag{383b}$$

$$\dots \qquad \dots$$

$$x(t_{N-1} + h) = x_{N-1} + hf\left(t_{N-1} + \frac{h}{2}\right). \tag{383c}$$

In 1895, Runge asked whether it would also be possible to extend the mid-point rule to the problem

$$x' = f(t, x(t)), \quad x(t_0) = x_0, \quad t \in [t_0, t_N]. \tag{384}$$

Note that now the right hand side depends on both t and $x(t)$. Let us take the first step of the mid-point rule and apply it to (384):

$$x(t_0 + h) = x_0 + hf\left(t_0 + \frac{h}{2}, x\left(t_0 + \frac{h}{2}\right)\right). \tag{385}$$

The only question is how to compute $x\left(t_0 + \frac{h}{2}\right)$. Runge proposed to use Euler's method for that, namely

$$x\left(t_0 + \frac{h}{2}\right) = x(t_0) + \frac{h}{2} f(t_0, x(t_0)). \tag{386a}$$

Thus, one can write (385) as

$$x(t_0 + h) = x_0 + hk_2, \tag{387}$$

where $k_1 = f(t_0, x_0)$, $k_2 = f\left(t_0 + \frac{h}{2}, x_0 + \frac{h}{2}k_1\right)$. Rewriting it in the discrete form gives a **2-stage Runge-Kutta (RK) method (also known as the Modified Euler method)**:

$$\begin{aligned} k_1 &= f(t_n, x_n), \\ k_2 &= f\left(t_n + \frac{h}{2}, x_n + \frac{h}{2}k_1\right), \\ x_{n+1} &= x_n + hk_2. \end{aligned} \tag{388}$$

14.2 The local truncation error of the 2-stage RK method

In order to calculate the local truncation error of the 2-stage RK method, we have to find the difference between the exact solution and its approximation given by the RK method

$$x_{n+1} = x_n + hf\left(t_n + \frac{h}{2}, x_n + \frac{h}{2}k_1\right). \tag{389}$$

For this, we have to Taylor expand $f\left(t_n + \frac{h}{2}, x_n + \frac{h}{2}k_1\right)$. But, before doing this, let me remind you the Taylor expansion of a function $g(t + \alpha h, x + \beta h)$:

$$g(t + \alpha h, x + \beta h) = g(t, x) + h\left(\alpha\frac{\partial g}{\partial t} + \beta\frac{\partial g}{\partial x}\right) + \frac{h^2}{2}\left(\alpha^2\frac{\partial^2 g}{\partial t^2} + 2\frac{\partial^2 g}{\partial x \partial t}\alpha\beta + \beta^2\frac{\partial^2 g}{\partial x^2}\right) + O(h^3). \tag{390}$$

In our case, $\alpha = \beta = \frac{1}{2}$. Hence, the Taylor expansion of $f\left(t_n + \frac{h}{2}, x_n + \frac{h}{2}k_1\right)$ is given by

$$\begin{aligned} f\left(t_n + \frac{h}{2}, x_n + \frac{h}{2}k_1\right) &= f(t_n, x_n) + \left(\frac{h}{2}f_t + \frac{h}{2}k_1 f_x\right) + \frac{h^2}{8}(f_{tt} + 2f_t k_1 + f_{xx}k_1^2) + O(h^3) \\ &= f(t_n, x_n) + \frac{h}{2}(f_t + f f_x) + \frac{h^2}{8}(f_{tt} + 2f_{tx}f + f_{xx}f^2) + O(h^3). \end{aligned} \tag{391}$$

Using (391) in (389) gives

$$x_{n+1} = x_n + hf + \frac{h^2}{2}(f_t + f f_x) + \frac{h^3}{8}(f_{tt} + 2f_{tx}f + f_{xx}f^2) + O(h^4). \tag{392}$$

Now we can find the local truncation error as the difference between the exact solution

$$x(t_{n+1}) = x(t_n) + hx'(t_n) + \frac{h^2}{2}x''(t_n) + \frac{h^3}{6}x'''(t_n) + O(h^4) \tag{393}$$

and the approximated solution given by the RK method (392):

$$x(t_{n+1}) - x_n = \frac{h^3}{24}\Big(f_{tt} + 2f_{tx}f + f_{xx}f^2 + 4f_x(f_t + f_x f)\Big) + O(h^4). \tag{394}$$

Thus local truncation erorr is $O(h^3)$, and the method is said to be of order $p = 2$.

Definition 14.1 (The order of the RK method) *If the local truncation error of the RK method is of order $O(h^{p+1})$, with $p > 0$, then the method is said to be of order p.*

Example 14.1 (The modified Euler method for a scalar equation) *The modified Euler method* *for the initial value problem*

$$x' = (1-2t)x,\ x(t_0) = 1,\ t \in [t_0, t_N] \tag{395}$$

is given by

$$k_1 = f(t_n, x_n) = (1-2t_n)x_n, \tag{396a}$$

$$k_2 = f\left(t_n + \frac{h}{2}, x_n + \frac{h}{2}k_1\right) = \left(1 - 2\left(t_n + \frac{h}{2}\right)\right)\left(x_n + \frac{h}{2}k_1\right), \tag{396b}$$

$$x_{n+1} = x_n + hk_2. \tag{396c}$$

▲

14.3 The general form of RK methods

In 1901, Kutta formulated the general scheme of what is now called Runge-Kutta methods:

$$x_{n+1} = x_n + h\sum_{i=1}^{s} b_i k_i, \tag{397a}$$

with

$$k_i = f\left(t_n + hc_i, x_n + h\sum_{j=1}^{s} a_{ij}k_j\right),\ i = 1, 2, \ldots, s \tag{397b}$$

Another form of (397), known as a Butcher array (or Butcher tableau), is given by

$$\begin{array}{c|cccc} c_1 & a_{11} & a_{12} & \cdots & a_{1s} \\ c_2 & a_{21} & a_{22} & \cdots & a_{2s} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_s & a_{s1} & a_{s2} & \cdots & a_{ss} \\ \hline & b_1 & b_2 & \cdots & b_s \end{array}$$

with

$$c_i = \sum_{j=1}^{s} a_{ij},\ i = 1, 2, \ldots, s. \tag{398}$$

Thus, given a value of s, the method depends on $s^2 + s$ parameters $\{a_{ij}, b_j\}$. The method is called explicit if $c_1 = 0$, $a_{ij} = 0$, $j \geq i$, or implicit otherwise.

For instance, the Butcher array of the 2-stage RK method (the modified Euler method) is

$$\begin{array}{c|cc} 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 \\ \hline & 0 & 1 \end{array}$$

Again, this is an explicit method, and therefore it does not require solving a nonlinear system of equations to find the solution (if the right hand side is nonlinear).

14.4 RK methods for systems of ODEs

RK methods for systems of ODEs are exactly the same as for scalar equations. The only difference is that k's are not scalars but vectors. Let us consider an example of how to apply a 2-stage RK method to a system of ODEs.

Example 14.2 *Consider the 2-stage RK method (also known as the improved Euler method) given by the Butcher tableau*

$$\begin{array}{c|cc} 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array}$$

with

$$\boldsymbol{k}_1 = f(t_n, \boldsymbol{x}_n), \tag{399a}$$

$$\boldsymbol{k}_2 = f\left(t_n + \frac{h}{2}, \boldsymbol{x}_n + \frac{h}{2}\boldsymbol{k}_1\right), \tag{399b}$$

$$\boldsymbol{x}_{n+1} = \boldsymbol{x}_n + \frac{h}{2}(\boldsymbol{k}_1 + \boldsymbol{k}_2), \tag{399c}$$

where $\boldsymbol{k}_1$, $\boldsymbol{k}_2$, $\boldsymbol{x}_n$, and $\boldsymbol{x}_{n+1}$ are vectors. Let us apply the improved Euler method to the following system of equations

$$\begin{cases} u' = tu, \\ v' = u^2 v. \end{cases} \tag{400}$$

Then the RK method becomes

$$\boldsymbol{k}_1 = \begin{pmatrix} t_n u_n \\ u_n^2 v_n \end{pmatrix} =: \begin{pmatrix} k_{11} \\ k_{21} \end{pmatrix}, \quad \boldsymbol{k}_2 = \begin{pmatrix} \left(t_n + \frac{h}{2}\right)\left(u_n + \frac{h}{2}k_{11}\right) \\ \left(u_n + \frac{h}{2}k_{11}\right)^2 \left(v_n + \frac{h}{2}k_{21}\right) \end{pmatrix}, \tag{401}$$

and

$$\boldsymbol{x}_{n+1} = \boldsymbol{x}_n + \frac{h}{2}(\boldsymbol{k}_1 + \boldsymbol{k}_2). \tag{402}$$

▲

Lecture 15 Order conditions for explicit RK methods

In this part we consider consider order conditions for RK methods. These are conditions imposed on the coefficients a_{ij}, b_i, and c_i, $i, j = 1, 2, \dots, s$ to guarantee a given order of accuracy.

15.1 1-stage RK methods

The general form of 1-stage RK methods is given by

$$k_1 = f(t_n, x_n), \tag{403a}$$
$$x_{n+1} = x_n + hb_1k_1. \tag{403b}$$

Note that for $b_1 = 1$ it is the Euler method.

The local truncation error of method (403) is given by

$$x(t_{n+1}) - x_{n+1} = h(1 - b_1)x'(t_n) + \frac{h^2}{2}x''(t_n) + O(h^3). \tag{404}$$

Thus, for $b_1 = 1$ we have the one order method, and the condition $b_1 = 1$ is called *the first order condition*.

15.2 2-stage RK methods

Let us consider the general form of 2-stage RK methods:

$$\begin{aligned} k_1 &= f(t_n, x_n), \\ k_2 &= f(t_n + ah, x_n + ahk_1), \\ x_{n+1} &= x_n + h(b_1k_1 + b_2k_2), \end{aligned} \tag{405}$$

with coefficients

$$a = c_2 = a_{21}. \tag{406}$$

As before, we Taylor expand function f and substitute it into the equation for x_{n+1}:

$$x_{n+1} = x_n + hb_1f_n + hb_2f(t_n + ah, x_n + ahk_1) \tag{407a}$$
$$= x_n + hb_1f_n + h\Big(f_n + ah(f_t + f_xf)|_{t=t_n} + O(h^2)\Big) \tag{407b}$$
$$= x_n + h(b_1 + b_2)f_n + ab_2h^2(f_t + f_xf)|_{t=t_n} + O(h^3). \tag{407c}$$

The local truncation error is given by

$$x(t_{n+1}) - x_{n+1} = h(1 - b_1 - b_2)f_n + h^2\left(\frac{1}{2} - ab_2\right)\Big(f_t + f_xf\Big)|_{t=t_n} + O(h^3). \tag{408}$$

Now, we can choose the parameters to get the order conditions:

For $b_1 + b_2 = 1$ and $\forall a$ the local truncation error is $O(h^2)$, and the method is of order $p = 1$,

For $b_1 + b_2 = 1$ and $ab_2 = \dfrac{1}{2}$ the local truncation error is $O(h^3)$, and the method is of order $p = 2$.

We cannot manipulate with the parameters to get a higher-order method.

Can you show that there is no 2-stage explicit RK method of order $p = 3$?

Let us define $b_2 = \theta$, then from

$$\begin{aligned} b_1 + b_2 = 1 &\Rightarrow b_1 = 1 - \theta, \\ ab_2 = \frac{1}{2} &\Rightarrow a = \frac{1}{2\theta}, \end{aligned}$$

and then the Butcher array is given by

$$\begin{array}{c|cc} 0 & 0 & 0 \\ a & a & 0 \\ \hline & 1-\theta & \theta \end{array}$$

This defines a one-parameter family of RK methods. Some of them, we already know

$$\begin{aligned} \theta = \frac{1}{2} &\Rightarrow \text{ The improved Euler method}, \\ \theta = 1 &\Rightarrow \text{ The modified Euler method}. \end{aligned}$$

For $\theta \neq 0$, all methods in the family are of order 2.

15.3 3-stage RK methods

The general form of 3-stage RK method is given by the following Butcher table

$$\begin{array}{c|ccc} 0 & 0 & 0 & 0 \\ c_2 & a_{21} & 0 & 0 \\ c_3 & a_{31} & a_{32} & 0 \\ \hline & b_1 & b_2 & b_3 \end{array}$$

with

$$c_2 = a_{21}, \; c_3 = a_{31} + a_{32}. \tag{409}$$

In the conventional form, the method is

$$k_1 = f(t_n, x_n), \tag{410a}$$
$$k_2 = f(t_n + c_2 h, x_n + a_{21} h k_1), \tag{410b}$$
$$k_3 = f(t_n + c_3 h, x_n + a_{31} h k_1 + a_{32} h k_2), \tag{410c}$$
$$x_{n+1} = x_n + h(b_1 k_1 + b_2 k_2 + b_3 k_3). \tag{410d}$$

Thus, 3-stage RK methods have 6 free parameters to be determined.

To compute the local truncation error of the method we plug k_1, k_2, and k_3 into (410d):

$$x_{n+1} = x_n + hb_1 f_n + hb_2 f(t_n + c_2 h, x_n + a_{21} h k_1) + hb_3 f(t_n + c_3 h, x_n + a_{31} h k_1 + a_{32} h k_2) \tag{411}$$

and Taylor expand k_2

$$\begin{aligned} k_2 &= f(t_n + c_2 h, x_n + a_{21} h k_1) \\ &= f(t_n, x_n) + c_2 h f_t + a_{21} h k_1 f_x + \frac{1}{2}\Big((c_2 h)^2 f_{tt} + 2c_2 h^2 a_{21} k_1 f_{tx} + (a_{21} h k_1)^2 f_{xx}\Big) + O(h^3) \\ &= f(t_n, x_n) + h(c_2 f_t + a_{21} f f_x) + \frac{h^2}{2}(c_2^2 f_{tt} + 2c_2 a_{21} f f_{tx} + a_{21}^2 f^2 f_{xx}) + O(h^3). \end{aligned} \tag{412}$$

and k_3

$$
\begin{aligned}
k_3 &= f(t_n + c_3 h, x_n + a_{31} h k_1 + a_{32} h k_2) \\
&= f(t_n, x_n) + c_3 h f_t + (a_{31} k_1 + a_{32} k_2) h f_x \\
&+ \frac{1}{2}\Big((c_3 h)^2 f_{tt} + 2 c_3 h^2 (a_{31} k_1 + a_{32} k_2) f_{tx} + h^2 (a_{31} k_1 + a_{32} k_2)^2 f_{xx}\Big) + O(h^3) \\
&= f(t_n, x_n) + h\Big(c_3 f_t + (a_{31} f + a_{32} k_2) f_x\Big) \\
&+ \frac{h^2}{2}\Big(c_3^2 f_{tt} + 2 c_3 (a_{31} f + a_{32} k_2) f_{tx} + (a_{31} f + a_{32} k_2)^2 f_{xx}\Big) + O(h^3).
\end{aligned}
\tag{413}
$$

1350 Substitution of $k_2 \approx f(t_n, x_n)$ into k_3 gives

$$
k_3 = f_n + h\Big(c_3 f_t + (a_{31} + a_{32}) f f_x\Big) + \frac{h^2}{2}\Big(c_3^2 f_{tt} + 2 c_3 (a_{31} + a_{32}) f f_{tx} + (a_{31} + a_{32})^2 f^2 f_{xx}\Big) + O(h^3).
\tag{414}
$$

Substitution of k_2 and k_3 into (410d) results in

$$
\begin{aligned}
x_{n+1} &= x_n + h b_1 f_n && \text{(415a)} \\
&+ h b_2 \left(f_n + h(c_2 f_t + a_{21} f f_x) + \frac{h^2}{2}(c_2^2 f_n + 2 c_2 a_{21} f t_{tx} + a_{21}^2 f^2 f_{xx} \right) \\
&+ h b_3 \left(f_n + h\Big(c_3 f_t + (a_{31} + a_{32}) f f_x\Big) \right. && \text{(415b)} \\
&\left. + \frac{h^2}{2}\Big(c_3^2 f_{tt} + 2 c_3 (a_{31} + a_{32}) f f_{tx} + (a_{31} + a_{32})^2 f^2 f_{xx}\Big)\right) + O(h^4) && \text{(415c)} \\
&= x_n + h(b_1 + b_2 + b_3) f + h^2\Big((c_2 b_2 + c_3 b_3) f_t + \big(b_2 a_{21} + b_3 (a_{31} + a_{32})\big) f f_x\Big) + \\
&\frac{h^3}{2}\Big((c_2^2 b_2 + c_3^2 b_3) f_{tt} + 2\big(b_2 c_2 a_{21} + b_3 c_3 (a_{31} + a_{32})\big) f f_{tx} && \text{(415d)} \\
&+ \big(b_2 a_{21}^2 + b_3 (a_{31} + a_{32})^2\big) f^2 f_{xx}\Big) + O(h^4). && \text{(415e)}
\end{aligned}
$$

1351 The Taylor expansion of the exact solution is given by

$$
x(t_{n+1}) = x(t_n) + h f(t_n, x_n) + \frac{h^2}{2}(f_t + f f_x) + \frac{h^3}{6}\Big(f_{tt} + 2 f_{tx} f + f_{xx} f^2 + f_x (f_t + f_x f)\Big) + O(h^4).
\tag{416}
$$

1352 Hence, the local truncation error is

$$
\begin{aligned}
x(t_{n+1}) - x_{n+1} &= h\Big(1 - (b_1 + b_2 + b_3)\Big) f \\
&+ \frac{h^2}{2}\Big(\big(1 - 2(c_2 b_2 + c_3 b_3)\big) f_t + \big(1 - 2(b_2 a_{21} + b_3 (a_{31} + a_{32}))\big) f f_x\Big) \\
&+ \frac{h^3}{6}\Big(\big(1 - 3(c_2^2 b_2 + c_3^2 b_3)\big) f_{tt} + \big(1 - 6(b_2 c_2 a_{21} + b_3 c_3 (a_{31} + a_{32}))\big) f f_{tx} \\
&+ \big(1 - 3(b_2 a_{21}^2 + b_3 (a_{31} + a_{32})^2)\big) f^2 f_{xx}\Big) + O(h^4).
\end{aligned}
\tag{417}
$$

1353 Using $c_3 = a_{31} + a_{32}$ in the equation above gives

$$\begin{aligned} x(t_{n+1}) - x_{n+1} = {} & h\big(1 - (b_1 + b_2 + b_3)\big) f \\ & + \frac{h^2}{2}\Big(\big(1 - 2(c_2 b_2 + c_3 b_3)\big) f_t + \big(1 - 2(b_2 a_{21} + b_3 c_3)\big) f f_x\Big) \\ & + \frac{h^3}{6}\Big(\big(1 - 3(c_2^2 b_2 + c_3^2 b_3)\big) f_{tt} + \big(2 - 6(b_2 c_2 a_{21} + b_3 c_3^2)\big) f f_{tx} \\ & + \big(1 - 3(b_2 a_{21}^2 + b_3 c_3^2)\big) f^2 f_{xx} + f_x(f_t + f_x f)\Big) + O(h^4). \end{aligned} \tag{418}$$

1354 Using $c_2 = a_{21}$ in the equation above gives

$$\begin{aligned} x(t_{n+1}) - x_{n+1} = {} & hf\big(1 - (b_1 + b_2 + b_3)\big) \\ & + h^2\Big(f_t + f f_x\Big)\big(\frac{1}{2} - (c_2 b_2 + c_3 b_3)\big) \\ & + \frac{h^3}{6}\Big(\big(1 - 3(c_2^2 b_2 + c_3^2 b_3)\big) f_{tt} + \big(2 - 6(b_2 c_2^2 + b_3 c_3^2)\big) f f_{tx} \\ & + \big(1 - 3(b_2 c^2 + b_3 c_3^2)\big) f^2 f_{xx} + f_x(f_t + f_x f)\Big) + O(h^4). \end{aligned} \tag{419}$$

1355 Thus we have the following order conditions:

1356

The first order condition:

$$b_1 + b_2 + b_3 = 1,\ x(t_{n+1}) - x_{n+1} = O(h^2).$$

The second order condition:

$$c_2 b_2 + c_3 b_3 = \frac{1}{2},\ x(t_{n+1}) - x_{n+1} = O(h^3).$$

We cannot get the third order condition because the term $f_x(f_t + f_x f)$ is left. However, if we substitute $k_2 = f + h(c_2 f_t + a_{21} f f_x)$ into k_3, we will be able to recover **the third order conditions**:

$$b_2 c_2^2 + b_3 c_3^2 = \frac{1}{3},\ c_2 a_{32} b_3 = \frac{1}{6},\ x(t_{n+1}) - x_{n+1} = O(h^4).$$

1357 **Example 15.1** *Let us consider two 3-stage RK methods given by the following Butcher arrays:*

Heun's method

$$\begin{array}{c|ccc} 0 & 0 & 0 & 0 \\ \frac{1}{3} & \frac{1}{3} & 0 & 0 \\ \frac{2}{3} & 0 & \frac{2}{3} & 0 \\ \hline & \frac{1}{4} & 0 & \frac{3}{4} \end{array}$$

Kutta's rule

$$\begin{array}{c|ccc} 0 & 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ 1 & -1 & 2 & 0 \\ \hline & \frac{1}{6} & \frac{2}{3} & \frac{1}{6} \end{array}$$

If we take the order conditions for the 3-stage method and plug the values for Heun's method and Kutta's rule we will find that these methods are of order 3. ▲

It is also worth mentioning that explicit RK methods with s stages have order s, if $s \leq 4$. However, it is not true in general for $s > 4$.

Theorem 15.1 (Butcher barrier theorem) *For $p > 4$, there is no explicit Runge-Kutta method of order p with $s = p$ stages.*

Lecture 16 Absolute stability of explicit RK methods

The concept of absolute stability developed for LMMs is equally relevant to RK methods. In other words, absolute stability for RK methods requires $x_n \to 0$ as $n \to \infty$ for the linear ODE $x' = \lambda x$, $\mathrm{Re}(\lambda) < 0$.

Example 16.1 (Stability of the θ-method) *Let us study the absolute stability of the RK method with the Butcher tableaux*

$$\begin{array}{c|cc} 0 & 0 & \\ a & a & 0 \\ \hline & 1-\theta & \theta \end{array}$$

with $a = 1/(2\theta)$. Application of the method to $x' = \lambda x$ results in

$$\begin{aligned}
k_1 &= f(t_n, x_n) = \lambda x_n, && \text{(420a)}\\
k_2 &= f(t_n + ah, x_n + ahk_1) = \lambda(x_n + ah\lambda x_n) && \text{(420b)}\\
x_{n+1} &= x_n + h(1-\theta)k_1 + h\theta k_2 && \text{(420c)}\\
&= x_n + h(1-\theta)\lambda x_n + h\theta\lambda(x_n + ah\lambda x_n), \quad \textit{with } \widehat{h} = \lambda h, && \text{(420d)}\\
&= x_n + \widehat{h}(1-\theta)x_n + \widehat{h}\theta(x_n + a\widehat{h}x_n) && \text{(420e)}\\
&= x_n + \widehat{h}(1-\theta)x_n + \widehat{h}\theta(1 + a\widehat{h})x_n && \text{(420f)}\\
&= \big(1 + \widehat{h} - \widehat{h}\theta + \widehat{h}\theta + \widehat{h}\theta a\widehat{h}\big)x_n && \text{(420g)}\\
&= \big(1 + \widehat{h}(1 + \theta a\widehat{h})\big)x_n, \quad \textit{and } a = \frac{1}{2\theta}, x_{n+1} \quad = \left(1 + \widehat{h}\left(1 + \frac{\widehat{h}}{2}\right)\right)x_n. && \text{(420h)}
\end{aligned}$$

16.1 Stability function

The stability polynomial for (420h) *is*

$$p(r) = r - \left(1 + \widehat{h}\left(1 + \frac{\widehat{h}}{2}\right)\right), \tag{421}$$

and

$$x_{n+1} = R(\widehat{h})x_n, \tag{422}$$

where

$$R(\widehat{h}) = 1 + \widehat{h}\left(1 + \frac{\widehat{h}}{2}\right). \tag{423}$$

$R(\widehat{h})$ *is called the stability function of the RK method.*

16.2 Interval of absolute stability

As you can see, based on the definition of absolute stability for LMMs, the stability function $R(\widehat{h})$ is the root of the stability polynomial $p(r)$. Therefore, in order for RK methods to be absolutely stable, we require $|r| < 1$, or $\left|R(\widehat{h})\right| < 1$. The interval of absolute stability is the set of real $\widehat{h}$ for *which* $\left|R(\widehat{h})\right| < 1$. *This leads to*

$$\left|1 + \widehat{h}\left(1 + \frac{\widehat{h}}{2}\right)\right| < 1, \tag{424a}$$

$$-2 < \widehat{h}\left(1 + \frac{\widehat{h}}{2}\right) < 0. \tag{424b}$$

Thus, the interval of absolute stability is $\widehat{h} \in (-2, 0)$.

16.3 The region of absolute stability

In order to plot the region of absolute stability, we substitute $r = e^{is}$ *into the stability polynomial* $p(r)$:

$$e^{is} - 1 + \widehat{h}\left(1 + \frac{\widehat{h}}{2}\right) = 0, \tag{425}$$

and solve it for $\widehat{h}$, *and then plot all the roots* $\widehat{h}_1$, $\widehat{h}_2$, *depending on the parameter* $s \in [0, 2\pi)$.

▲

Example 16.2 (The interval of absolute stability of the Heun method) *Find the interval of absolute stability of Heun's method given by the Butcher tableau*

$$\begin{array}{c|ccc}
0 & 0 & & \\
\frac{1}{3} & \frac{1}{3} & 0 & \\
\frac{2}{3} & 0 & \frac{2}{3} & 0 \\
\hline
 & \frac{1}{4} & 0 & \frac{3}{4}
\end{array}$$

with

$$k_1 = f(t_n, x_n), \tag{426a}$$

$$k_2 = f\left(t_n + \frac{h}{3}, x_n + \frac{h}{3}k_1\right), \tag{426b}$$

$$k_3 = f\left(t_n + \frac{2}{3}h, x_n + \frac{2}{3}hk_2\right), \tag{426c}$$

$$x_{n+1} = x_n + h\left(\frac{k_1}{4} + \frac{3}{4}k_3\right). \tag{426d}$$

Application of the method to equations $x' = \lambda x$ gives:

$$
\begin{aligned}
k_1 &= \lambda x_n, && \text{(427a)}\\
k_2 &= \lambda \left(x_n + \frac{h}{3}\lambda x_n \right), && \text{(427b)}\\
k_3 &= \lambda \left(x_n + \frac{2}{3}h \left(\lambda x_n + \frac{h}{3}\lambda^2 x_n \right) \right), && \text{(427c)}\\
x_{n+1} &= x_n + h\left(\frac{\lambda x_n}{4} + \frac{3}{4}\lambda \left(x_n + \frac{2}{3}h \left(\lambda x_n + \frac{h}{3}\lambda^2 x_n \right) \right) \right), \quad \textit{and } \widehat{h} := \lambda h, && \text{(427d)}\\
&= x_n + \widehat{h}\frac{x_n}{4} + \frac{3}{4}\widehat{h}\left(x_n + \frac{2}{3} \left(\widehat{h} x_n + \frac{\widehat{h}^2}{3} x_n \right) \right) && \text{(427e)}\\
&= x_n + \widehat{h} x_n + \frac{\widehat{h}^2}{2} x_n + \frac{\widehat{h}^3}{6} x_n && \text{(427f)}\\
&= R(\widehat{h}) x_n, \quad \textit{where } R(\widehat{h}) = 1 + \widehat{h} + \frac{\widehat{h}^2}{2} + \frac{\widehat{h}^3}{6}. && \text{(427g)}
\end{aligned}
$$

The interval of absolute stability is

$$
\begin{aligned}
\left| 1 + \widehat{h} + \frac{\widehat{h}^2}{2} + \frac{\widehat{h}^3}{6} \right| &< 1 && \text{(428a)}\\
-2 < \widehat{h} + \frac{\widehat{h}^2}{2} + \frac{\widehat{h}^3}{6} &< 0 && \text{(428b)}\\
-2 < \widehat{h} \left(1 + \frac{\widehat{h}}{2} + \frac{\widehat{h}^2}{6} \right) &< 0 && \text{(428c)}\\
-12 < \widehat{h} \left(6 + 3\widehat{h} + \widehat{h}^2 \right) &< 0 && \text{(428d)}\\
\widehat{h}_1 = 0,\ \widehat{h}_{2,3} &= \frac{-3 \pm 4i}{2}. && \text{(428e)}
\end{aligned}
$$

1381 ▲

Example 16.3 (Stability function of Kutta's rule) *Consider the Kutta rule, 3-stage explicit RK method given by the Butcher tableau*

$$
\begin{array}{c|ccc}
0 & 0 & & \\
\frac{1}{2} & \frac{1}{2} & 0 & \\
1 & -1 & 2 & 0 \\
\hline
 & \frac{1}{6} & \frac{2}{3} & \frac{1}{6}
\end{array}
$$

with

$$
\begin{aligned}
k_1 &= f(t_n, x_n), && (429a)\\
k_2 &= f\left(t_n + \frac{h}{2}, x_n + \frac{h}{2}k_1\right), && (429b)\\
k_3 &= f\left(t_n + h, x_n - hk_1 + 2hk_2\right), && (429c)\\
x_{n+1} &= x_n + h\left(\frac{k_1}{6} + \frac{2}{3}k_2 + \frac{k_3}{6}\right). && (429d)
\end{aligned}
$$

Application of the method to equations $x' = \lambda x$ gives:

$$
\begin{aligned}
k_1 &= \lambda x_n, && (430a)\\
k_2 &= \lambda\left(x_n + \frac{h}{2}\lambda x_n\right), && (430b)\\
k_3 &= \lambda\left(x_n - h\lambda x_n + 2h\lambda\left(x_n + \frac{h}{2}\lambda x_n\right)\right), \quad \textit{and } \widehat{h} := \lambda h, && (430c)\\
x_{n+1} &= x_n + h\left(\frac{\lambda x_n}{6} + \frac{2}{3}\lambda\left(x_n + \frac{\widehat{h}}{2}x_n\right) + \frac{\lambda}{6}\left(x_n - \widehat{h}x_n + 2\widehat{h}\left(x_n + \frac{\widehat{h}}{2}x_n\right)\right)\right) && (430d)\\
&= x_n + \frac{\widehat{h}}{6}x_n + \frac{2}{3}\widehat{h}\left(x_n + \frac{\widehat{h}}{2}x_n\right) + \frac{\widehat{h}}{6}\left(x_n + \widehat{h}x_n + \widehat{h}^2 x_n\right) && (430e)\\
&= x_n + \widehat{h}x_n + \frac{\widehat{h}^2}{2}x_n + \frac{\widehat{h}^3}{6}x_n && (430f)\\
&= R(\widehat{h})x_n,\ R(\widehat{h}) = 1 + \widehat{h} + \frac{\widehat{h}^2}{2} + \frac{\widehat{h}^3}{6}. && (430g)
\end{aligned}
$$

As can be seen, the interval and the region of absolute stability are the same as for Heun's method.
▲

Example 16.4 (The interval and region of absolute stability for a 4-stage RK method) *Let us consider a 4-stage explicit RK method given by the Butcher tableau*

$$
\begin{array}{c|cccc}
0 & 0 & & & \\
\frac{1}{2} & \frac{1}{2} & 0 & & \\
\frac{1}{2} & 0 & \frac{1}{2} & 0 & \\
1 & 0 & 0 & 1 & \\
\hline
 & \frac{1}{6} & \frac{1}{3} & \frac{1}{3} & \frac{1}{6}
\end{array}
$$

with

$$
\begin{aligned}
k_1 &= f(t_n, x_n), && (431a)\\
k_2 &= f\left(t_n + \frac{h}{2}, x_n + \frac{h}{2}k_1\right), && (431b)\\
k_3 &= f\left(t_n + \frac{h}{2}, x_n + \frac{h}{2}k_2\right), && (431c)\\
k_4 &= f(t_n + h, x_n + hk_3), && (431d)\\
x_{n+1} &= x_n + h\left(\frac{k_1}{6} + \frac{k_2}{3} + \frac{k_3}{3} + \frac{k_4}{6}\right). && (431e)
\end{aligned}
$$

Application of the method to equations $x' = \lambda x$ gives:

$$
\begin{aligned}
k_1 &= \lambda x_n, && (432a)\\
k_2 &= \lambda\left(x_n + \frac{h}{2}\lambda x_n\right), && (432b)\\
k_3 &= \lambda\left(x_n + \frac{h}{2}\lambda\left(x_n + \frac{h}{2}\lambda x_n\right)\right), && (432c)\\
k_4 &= \lambda\left(x_n + h\lambda\left(x_n + \frac{h}{2}\lambda\left(x_n + \frac{h}{2}\lambda x_n\right)\right)\right), \quad \textit{and } \lambda h := \widehat{h}, && (432d)\\
x_{n+1} &= x_n + h\left(\lambda x_n + \frac{\lambda}{3}\left(x_n + \frac{\widehat{h}}{2}x_n\right) + \frac{\lambda}{3}\left(x_n + \frac{\widehat{h}}{2}\left(x_n + \frac{\widehat{h}}{2}x_n\right)\right)\right. && (432e)\\
&\left.+ \frac{\lambda}{6}\left(x_n + \widehat{h}\left(x_n + \frac{\widehat{h}}{2}\left(x_n + \frac{\widehat{h}}{2}x_n\right)\right)\right)\right) && (432f)\\
&= x_n + \widehat{h}x_n + \frac{\widehat{h}}{3}\left(x_n + \frac{\widehat{h}}{2}x_n\right) + \frac{\widehat{h}}{3}\left(x_n + \frac{\widehat{h}}{2}\left(x_n + \frac{\widehat{h}}{2}x_n\right)\right) && (432g)\\
&+ \frac{\widehat{h}}{6}\left(x_n + \widehat{h}\left(x_n + \frac{\widehat{h}}{2}\left(x_n + \frac{\widehat{h}}{2}x_n\right)\right)\right), && (432h)\\
&= x_n + \widehat{h}x_n + \frac{\widehat{h}^2}{2}x_n + \frac{\widehat{h}^3}{6}x_n + \frac{\widehat{h}^4}{24}x_n, && (432i)\\
&= R(\widehat{h})x_n, \; R(\widehat{h}) = 1 + \widehat{h} + \frac{\widehat{h}^2}{2} + \frac{\widehat{h}^3}{6} + \frac{\widehat{h}^4}{24}. && (432j)
\end{aligned}
$$

Then, the interval of absolute stability is a set of real $\widehat{h}$ such that

$$
\left|1 + \widehat{h} + \frac{\widehat{h}^2}{2} + \frac{\widehat{h}^3}{6} + \frac{\widehat{h}^4}{24}\right| < 1. \tag{433}
$$

▲

16.4 Stability function for 3-stage third-order RK methods

Let's take a retrospective look at, say, the 3-stage RK methods we have studied. As you have noticed, all of them have the same stability function. Does this mean that all 3-stage RK methods inherit this feature? Let us check it. For this, we consider the whole family of 3-stage RK methods, which are represented by the Butcher table

$$\begin{array}{c|ccc} 0 & 0 & & \\ c_2 & a_{21} & 0 & \\ c_3 & a_{31} & a_{32} & 0 \\ \hline & b_1 & b_2 & b_3 \end{array}$$

with

$$k_1 = f(t_n, x_n), \tag{434a}$$
$$k_2 = f\left(t_n + c_2 h, x_n + h a_{21} k_1\right), \tag{434b}$$
$$k_3 = f\left(t_n + c_3 h, x_n + h a_{31} k_1 + h a_{32} k_2\right), \tag{434c}$$
$$x_{n+1} = x_n + h\left(b_1 k_1 + b_2 k_2 + b_3 k_3\right). \tag{434d}$$

The order conditions for 3-stage methods to be of third-order are

$$b_2 c_2^2 + b_3 c_3^2 = \frac{1}{3},\ b_1 + b_2 + b_3 = 1, \text{ where } c_2 = a_{21}, \text{ and } a_{31} + a_{32} = c_3, \tag{435a}$$
$$c_2 a_{32} b_3 = \frac{1}{6},\ b_2 c_2 + b_3 c_3 = \frac{1}{2}. \tag{435b}$$

In order to find the interval of absolute stability for all 3-stage third-order RK methods we apply its apply them to the equation $x' = \lambda x$:

$$k_1 = \lambda x_n \Rightarrow h k_1 = \widehat{h} x_n, \text{ with } \widehat{h} = \lambda h, \tag{436a}$$
$$k_2 = (x_n + \lambda h a_{21} x_n)\lambda \Rightarrow h k_2 = \widehat{h}\left(x_n + \widehat{h} a_{21} x_n\right), \tag{436b}$$
$$k_3 = \Big(x_n + h\lambda a_{31} x_n + h a_{32}\lambda\big(x_n + \lambda h a_{21} x_n\big)\Big)\lambda \Rightarrow \tag{436c}$$
$$h k_3 = \Big(x_n + \widehat{h} a_{31} x_n + \widehat{h} a_{32}\big(x_n + \widehat{h} a_{21} x_n\big)\Big)\widehat{h}, \tag{436d}$$
$$x_{n+1} = x_n \widehat{h}\Big(b_1 x_n + b_2\big(x_n + \widehat{h} a_{21} x_n\big) + b_3\big(x_n + \widehat{h} a_{31} x_n + \widehat{h} a_{32}(x_n + \widehat{h} a_{21} x_n)\big)\Big) \tag{436e}$$
$$= x_n + \widehat{h}\Big(x_n\big(b_1 + b_2 + b_3\big) + \widehat{h} x_n\big(b_2 a_{21} + b_3(a_{31} + a_{32})\big) + b_3 \widehat{h}^2 a_{32} a_{21} x_n\Big) \tag{436f}$$
$$\text{using } b_1 + b_2 + b_3 = 1,\ b_2 a_{21} + b_3(a_{31} + a_{32}) = \frac{1}{2},\ a_{21} a_{32} b_3 = \frac{1}{6} \text{ gives} \tag{436g}$$
$$= x_n + \widehat{h}\left(x_n + \frac{\widehat{h}}{2} x_n + \frac{\widehat{h}^2}{6} x_n\right) \tag{436h}$$
$$= R(\widehat{h}) x_n,\ R(\widehat{h}) = 1 + \widehat{h} + \frac{\widehat{h}^2}{2} + \frac{\widehat{h}^3}{6}. \tag{436i}$$

We have obtained the same stability polynomial for all 3-stage third-order RK methods. Thus, their interval of absolute stability is also the same.

16.5 Stability function for s-stage RK methods

Our goal is to find the stability function for an s-stage method. For this, we apply an s-stage RK method to the equation $x' = \lambda x$. It leads to $x_{n+1} = R(\widehat{h}) x_n$. Let us Taylor expand the exact solution $x(t_{n+1})$ about $t = t_n$

$$x(t_{n+1}) = x(t_n) + hx'(t_n) + \frac{h^2}{2}x''(t_n) + \frac{h^3}{6}x'''(t_n) + \cdots + \frac{h^s}{s!}x^{(s)}(t_n) + O(h^{s+1}). \tag{437}$$

Now, we can substitute $x' = \lambda x$, $x'' = \lambda x' = \lambda^2 x$, $x''' = \lambda^3 x$, $\ldots$ into the Taylor expansion that results in

$$x(t_{n+1}) = \left(1 + \widehat{h} + \frac{\widehat{h}^2}{2} + \cdots + \frac{\widehat{h}^s}{s!}\right) x(t_n) + O(\widehat{h}^{s+1}), \tag{438}$$

where $\widehat{h} := \lambda h$. But, this is nothing more than the Taylor expansion of $e^{\widehat{h}}$. Thus, we have $x(t_{n+1}) - x_{n+1} = O(\widehat{h}^{s+1})$, and

$$R(\widehat{h}) = e^{\widehat{h}} + O(\widehat{h}^{s+1}). \tag{439}$$

It is now clear that $|R(\widehat{h})| \to \infty$ as $\widehat{h} \to -\infty$, so that no explicit RK method can be A-stable. However, since $|R(\widehat{h})| \leq 1$, and $R(\widehat{h}) = 1$ at $\widehat{h} = 0$, RK methods are always zero stable. Note that the interval and region of absolute stability increase with s.

16.6 Absolute stability of RK methods for systems

We have already discussed the absolute stability of LMMs for systems of ODEs. The similar argument is valid for RK methods applied to

$$u' = Au, \quad u \in \mathbb{R}^m,\ A \in \mathbb{R}^{m\times m}. \tag{440}$$

Let us consider the application of the 2-stage method to system (440)

$$\begin{array}{c|cc} 0 & 0 & \\ a & a & 0 \\ \hline & 1-\theta & \theta \end{array}, \tag{441}$$

with $a = 1/(2\theta)$ and

$$\begin{aligned} k_1 &= f(t_n, x_n), && (442a)\\ k_2 &= f\left(t_n + ah, x_n + ahk_1\right), && (442b)\\ x_{n+1} &= x_n + h\big((1-\theta)k_1 + \theta k_2\big), && (442c) \end{aligned}$$

Application of the method to (440) gives

$$\begin{aligned} k_1 &= Au_n, && (443a)\\ k_2 &= A(u_n + ahAu_n), && (443b)\\ u_{n+1} &= u_n + h\big((1-\theta)Au_n + \theta A(u_n + ahAu_n)\big) && (443c)\\ &= u_n + h\big((1-\theta)Au_n + \theta Au_n\big) + \frac{(Ah)^2}{2}u_n && (443d)\\ &= u_n + Ahu_n + \frac{(Ah)^2}{2}u_n && (443e)\\ &= R(Ah)u_n,\ R(Ah) = 1 + Ah + \frac{(Ah)^2}{2}. && (443f) \end{aligned}$$

We have the same stability polynomial as for the scalar equation. Using the substitution $u = Vx$ in (440), we find

$$Vx' = AVx. \tag{444}$$

Multiplication of this equation by V^{-1} gives

$$x' = \Lambda x, \tag{445}$$

which leads to

$$x_{n+1} = R(\Lambda h)x_n, \tag{446}$$

where Λ is a diagonal matrix of eigenvalues of A. This is the system in which all equations are uncoupled. Thus, to ensure absolute stability of the RK method for the system, $\widehat{h}_i := h\lambda_i$ has to be within the region of absolute stability for every eigenvalue λ_i, $i = 1, 2, \ldots, m$.

Lecture 17 Implicit RK methods

In this part we will derive order conditions for 2-stage implicit RK methods and discuss absolute stability of these methods.

17.1 Order conditions for 2-stage implicit RK methods

Let us skip the order conditions for the 1-stage implicit RK methods, and start from the 2-stage methods, which have the following Butcher tableaux

$$
\begin{array}{c|cc}
c_1 & a_{11} & a_{12} \\
c_2 & a_{21} & a_{22} \\
\hline
 & b_1 & b_2
\end{array}
$$

with

$$c_1 = a_{11} + a_{12}, \tag{447a}$$
$$c_2 = a_{21} + a_{22}. \tag{447b}$$

In the conventional form, the method can be written as

$$k_1 = f\left(t_n + c_1 h, x_n + (a_{11}k_1 + a_{12}k_2)h\right), \tag{448a}$$
$$k_2 = f\left(t_n + c_2 h, x_n + (a_{21}k_1 + a_{22}k_2)h\right), \tag{448b}$$
$$x_{n+1} = x_n + h(b_1 k_1 + b_2 k_2). \tag{448c}$$

As for explicit RK methods, the derivation of order conditions for implicit RK methods is based on Taylor expansions of k_1 and k_2. Let us Taylor expand k_1 and k_2 about (t_n, x_n) up to order $O(h^3)$:

$$k_1 = f+h\big(c_1 f_t+(a_{11}k_1+a_{12}k_2)f_x\big)+\frac{h^2}{2}\big(c_1^2 f_{tt}+2c_1(a_{11}k_1+a_{12}k_2)f_{xt}+(a_{11}k_1+a_{12}k_2)^2 f_{xx}\big)+O(h^3). \tag{449}$$

$$k_2 = f+h\big(c_2 f_t+(a_{21}k_1+a_{22}k_2)f_x\big)+\frac{h^2}{2}\big(c_2^2 f_{tt}+2c_2(a_{21}k_1+a_{22}k_2)f_{xt}+(a_{21}k_1+a_{22}k_2)^2 f_{xx}\big)+O(h^3). \tag{450}$$

Substitution of the Taylor expansions of k_1 and k_2 up to order $O(h^2)$ into the second term of k_1 and k_2, and the Taylor expansions of k_1 and k_2 up to order $O(h)$ into the third term of k_1 and k_2 gives:

$$k_1 = f+h\Big(c_1 f_t+\big(a_{11}(f+h(c_1 f_t+(a_{11}k_1+a_{12}k_2)f_x))+a_{12}(f+h(c_2 f_t+(a_{21}k_1+a_{22}k_2)f_x))\big)f_x\Big)+ \frac{h^2}{2}\Big(c_1^2 f_{tt}+2c_1^2 f f_{xt}+c_1^2 f^2 f_{xx}\Big)+O(h^3). \tag{451}$$

And substitute the Taylor expansion of k_1 and k_2 up to order $O(h)$ into the second term of k_1 again:

$$k_1 = f+h\Big(c_1 f_t+\big(a_{11}(f+h(c_1 f_t+c_1 f_x f))+a_{12}(f+h(c_2 f_t+c_2 f f_x))\big)f_x\Big)+ \frac{h^2}{2}\Big(c_1^2 f_{tt}+2c_1^2 f f_{xt}+c_1^2 f^2 f_{xx}\Big)+O(h^3). \tag{452}$$

Then, we have

$$k_1 = f + h\Big(c_1 f_t + \big(a_{11}(f + c_1 h f') + a_{12}(f + c_2 h f')\big) f_x\Big) + \frac{h^2}{2} c_1^2 (f_{tt} + 2 f f_{xt} + f^2 f_{xx}) + O(h^3), \tag{453}$$

with $f' = f_t + f_x f$.
The same for k_2, namely

$$k_2 = f + h\Big(c_2 f_t + \big(a_{21}(f + c_2 h f') + a_{22}(f + c_2 h f')\big) f_x\Big) + \frac{h^2}{2} c_2^2 (f_{tt} + 2 f f_{xt} + f^2 f_{xx}) + O(h^3). \tag{454}$$

Substitution of the Taylor expansion of k_1 and k_2 into (448c) gives

$$\begin{aligned} x_{n+1} = x_n + h b_1 \Big(f + h\big(c_1 f_t + (a_{11}(f + c_1 h f') + a_{12}(f + c_2 h f')) f_x\big) + \frac{h^2}{2} c_1^2 (f_{tt} + 2 f f_{xt} + f^2 f_{xx})\Big) \\ h b_2 \Big(f + h\big(c_2 f_t + (a_{21}(f + c_2 h f') + a_{22}(f + c_2 h f')) f_x\big) + \frac{h^2}{2} c_2^2 (f_{tt} + 2 f f_{xt} + f^2 f_{xx})\Big) + O(h^4). \end{aligned} \tag{455}$$

The Taylor expansion of the exact solution is given by

$$x(t_{n+1}) = x(t_n) + h f + \frac{h^2}{2}(f_t + f_x f) + \frac{h^3}{6}\big(f_{tt} + 2 f_{xt} f + f_{xx} f^2 + f_x(f_t + f f_x)\big) + O(h^4). \tag{456}$$

In order to get the order conditions we compute the local truncation error and find the restrictions on the coefficients which give the local truncation error of different orders. In particular, we find:
The first order condition:

$$b_1 + b_2 = 1, \ x(t_{n+1}) - x_{n+1} = O(h^2), \tag{457}$$

The second order condition:

$$b_1 c_1 + b_2 c_2 = \frac{1}{2}, \ x(t_{n+1}) - x_{n+1} = O(h^3), \tag{458}$$

The third order conditions:

$$b_1 c_1^2 + b_2 c_2^2 = \frac{1}{3}, \ b_1(a_{11} c_1 + a_{12} c_2) + b_2(a_{21} c_1 + a_{22} c_2) = \frac{1}{6}, \ x(t_{n+1}) - x_{n+1} = O(h^4). \tag{459}$$

17.2 Absolute stability of 1-stage implicit RK methods

The concept of absolute stability for implicit RK methods does not change, and remains the same as for all methods studied in the course. However, since our focus is on implicit RK methods, one has to explicitly express the stability polynomial. Let us consider the general form of 1-stage implicit RK methods

$$\begin{array}{c|c} c_1 & a_{11} \\ \hline & b_1 \end{array}$$

with $c_1 = a_{11} = a$, and

$$k_1 = f(t_n + ha, x_n + h a k_1), \tag{460a}$$

$$x_{n+1} = x_n + h b_1 k_1. \tag{460b}$$

Application of method (460) to the equation

$$x' = \lambda x,\ Re(\lambda) < 0 \tag{461}$$

gives

$$\begin{aligned} k_1 &= \lambda(x_n + hak_1), \\ k_1 &= \frac{\lambda x_n}{1 - \widehat{h}a},\ \widehat{h} := \lambda h, \\ x_{n+1} &= x_n + \frac{\widehat{h}x_n}{1 - \widehat{h}a} b_1. \end{aligned} \tag{462}$$

Using the 1st and 2nd order conditions for 1-stage implicit RK methods ($b_1 = 1, a = \frac{1}{2}$) leads to

$$\begin{aligned} x_{n+1} &= x_n + \frac{\widehat{h}x_n}{1 - \dfrac{\widehat{h}}{2}}, \\ p(r) &= r - \left(1 + \frac{\widehat{h}}{1 - \frac{\widehat{h}}{2}}\right) \\ &= r - R(\widehat{h}),\ R(\widehat{h}) := \left(1 + \frac{\widehat{h}}{1 - \frac{\widehat{h}}{2}}\right). \end{aligned} \tag{463}$$

Thus, the interval of absolute stability is

$$|r| = |R(\widehat{h})| < 1,\ \widehat{h} \in (-\infty, 0). \tag{464}$$

17.3 Absolute stability of 2-stage implicit RK methods

In order to study the absolute stability of the 2-stage implicit RK method in its general form

$$k_1 = f\left(t_n + c_1 h, x_n + (a_{11}k_1 + a_{12}k_2)h\right), \tag{465a}$$
$$k_2 = f\left(t_n + c_2 h, x_n + (a_{21}k_1 + a_{22}k_2)h\right), \tag{465b}$$
$$x_{n+1} = x_n + h(b_1 k_1 + b_2 k_2). \tag{465c}$$

we apply it to the equation $x' = \lambda x,\ Re(\lambda) < 0$ that gives

$$\begin{aligned} k_1 &= \lambda(x_n + (a_{11}k_1 + a_{12}k_2)h), \\ k_2 &= \lambda(x_n + (a_{21}k_1 + a_{22}k_2)h). \end{aligned} \tag{466}$$

To proceed, we have to solve this system for k_1, k_2 and then plug the solution into (465c) to get the stability polynomial.

Can you do it?

Lecture 18 Adaptive step size control

All methods we have discussed so far use a fixed time step to integrate over the interval $[t_0, t_N]$. Thus, the number of steps is $(t_N - t_0)/h$, and the accuracy of the results is of order $O(h^p)$ for a method of order p. In this lecture, we explore the possibility of using different time steps h_n at different times t_n in order to improve the methods efficiency, i.e. to obtain the same accuracy with fewer steps or better accuracy with the same number of steps. Based on this simple idea, one can consider using an adaptive time step size: small time steps for rapidly varied solutions, and large time steps for slowly varied solutions.

18.1 How to choose the step size?

There are many step size control strategies. In this course, we will consider adaptive step size control for one-step methods which is based on the control of the local truncation error given by the reminder

$$R_{n+1}(h_n) = H(t_n)h_n^{p+1} + O(h_n^{p+2}), \tag{467}$$

where $H(t_n)$ is a $p+1$ time derivative of the solution multiplied by an error constant.

Suppose that a numerical solution x_n has already been computed with a time step h_n. To compute the solution x_{n+1} we have to choose a tolerance $\varepsilon > 0$ and compute the reminder $R_{n+1}(h_n)$. Then, there are three options:

1. if $|R_{n+1}(h_n)| = \varepsilon$ then h_n is accepted, and we can compute x_{n+1} with h_n.
2. if $|R_{n+1}(h_n)| < \varepsilon$ then h_n is rejected (too small), and we compute x_{n+1} with a larger time step h_{new}.
3. if $|R_{n+1}(h_n)| > \varepsilon$ then h_n is rejected (too large), and we compute x_{n+1} with a smaller time step h_{new}.

Note that we take the absolute value of the reminder, since ε is positive. The question is how to compute h_{new}? From (467) we know that

$$R_{n+1}(h_n) \approx H(t_n)h_n^{p+1}$$

and then

$$H(t_n) = \frac{R_{n+1}(h_n)}{h_n^{p+1}}. \tag{468}$$

On the other hand, we want

$$R_{n+1}(h_{\text{new}}) = \varepsilon \approx |H(t_n)|h_{\text{new}}^{p+1}$$

which in turn gives

$$h_{\text{new}}^{p+1} = \frac{\varepsilon}{|H(t_n)|}. \tag{469}$$

Upon substitution of (468) into (470) we have

$$h_{\text{new}}^{p+1} = h_n^{p+1}\left|\frac{\varepsilon}{R_{n+1}(h_n)}\right|. \tag{470}$$

Raising the last equation to the power $\dfrac{1}{p+1}$

$$h_{\text{new}} = h_n\left|\frac{\varepsilon}{R_{n+1}(h_n)}\right|^{\frac{1}{p+1}}. \tag{471a}$$

Thus, if we know how to compute the reminder $R_{n+1}(h_n)$, we can compute the new time step h_{new}.

18.2 Taylor series methods

The $TS(p)$ method is given by

$$x_{n+1} = x_n + hx'_n + \frac{h^2}{2}x''_n + \cdots + \frac{h^p}{p!}x_n^{(p)}, \tag{472}$$

where $x'_n = f_n$. The local truncation error is

$$R_{n+1} = \frac{h_n^{p+1}}{(p+1)!}x^{(p+1)}(t_n) + O(h_n^{p+2}), \tag{473}$$

in which the leading term can be approximated by replacing $x^{(p+1)}(t_n)$ with $x_n^{(p+1)}$, thus giving

$$R_{n+1} = \frac{h_n^{p+1}}{(p+1)!}x_n^{(p+1)}, \tag{474}$$

for which

$$H(t_n) = \frac{x_n^{(p+1)}}{(p+1)!},$$

and therefore

$$R_{n+1} = |H(t_n)|h_n^{p+1} \Rightarrow H(t_n) = \frac{R_{n+1}}{h_n^{p+1}}. \tag{475}$$

Now, we require

$$\varepsilon = |H(t_n)|h_{\text{new}}^{p+1}. \tag{476}$$

Then, we have

$$h_{\text{new}}^{p+1} = \left|\frac{\varepsilon}{R_{n+1}}\right| h_n^{p+1} \Rightarrow h_{\text{new}} = h_n \left|\frac{\varepsilon}{R_{n+1}}\right|^{\frac{1}{p+1}}. \tag{477a}$$

If the current step h_n is accepted, we progress with h_n. Otherwise, the time step h_n is rejected, and we compute the solution with h_{new}.

It remains to choose the suitable value for h_0. It can be done by setting $|R_0| = \varepsilon$ that gives

$$\varepsilon = h_0^{p+1}\left|\frac{x_0^{(p+1)}}{(p+1)!}\right| \Rightarrow h_0 = \left|\frac{\varepsilon(p+1)!}{x_0^{(p+1)}}\right|^{\frac{1}{p+1}}. \tag{478}$$

Example 18.1 (Automatic step size control for the Euler method; scalar ODE) *Let us consider the ODE*

$$x' = (1-2t)x,\ x(t_0) = 1,\ t \in [t_0, t_N],$$

and apply the Euler method:

$$x_{n+1} = x_n + h_n(1-2t_n)x_n. \tag{479}$$

To compute h_{new}, we use the previously derived formula

$$h_{\text{new}} = h_n \left|\frac{\varepsilon}{R_{n+1}}\right|^{\frac{1}{p+1}},\ \textit{with } p=1 \textit{ and } R_{n+1} = \frac{h_n^2}{2}x''_n. \tag{480}$$

Then, we have

$$h_{\text{new}} = h_n \left| \frac{2\varepsilon}{h_n^2 x_n''} \right|^{\frac{1}{2}} = \left| \frac{2\epsilon}{x_n''} \right|^{\frac{1}{2}}, \tag{481}$$

where

$$\begin{aligned} x_n'' &= -2x_n + (1-2t)x_n' && (482a)\\ &= -2x_n + (1-2t)^2 x_n && (482b)\\ &= \big((1-2t)^2 - 2\big)x_n. && (482c) \end{aligned}$$

The initial step is given by

$$h_0 = \left| \frac{2\varepsilon}{x_0''} \right|^{\frac{1}{2}} = \sqrt{2\varepsilon},\ x_0'' = -1.$$

▲

Example 18.2 (Automatic step size control for the TS(2) method; scalar ODE) *Using formula* (471a) *for* $TS(2)$ *applied to*

$$x' = (1-2t)x,\ x(t_0) = 1,\ t \in [t_0, t_N],$$

we have

$$h_{new} = \left| \frac{6\varepsilon}{x_n'''} \right|^{\frac{1}{3}}. \tag{483}$$

$$\begin{aligned} x_n''' &= \Big(\big((1-2t)^2 - 2\big)x_n\Big)' && (484a)\\ &= -4(1-2t)x_n + \Big((1-2t)^2 - 2\Big)x_n' && (484b)\\ &= -4(1-2t)x_n + \Big((1-2t)^2 - 2\Big)\Big(1-2t\Big)x_n && (484c)\\ &= \Big(1-2t\Big)\Big((1-2t)^2 - 6\Big)x_n, \quad x'''(0) = -5, && (484d) \end{aligned}$$

and therefore (for $p = 2$*) we have*

$$h_0 = \left| \frac{\varepsilon(p+1)!}{x^{(p+1)}} \right|^{\frac{1}{p+1}} \overset{p=2}{=} \left| \frac{6\varepsilon}{x_0'''} \right|^{\frac{1}{3}} = \left(\frac{6}{5}\varepsilon \right)^{\frac{1}{3}}. \tag{485}$$

▲

Example 18.3 (Automatic step size control for the Euler method; system of ODEs) *Let us consider the following system of equations*

$$x' = Ax + g,\ A = \begin{pmatrix} -8 & 8 \\ 0 & -\frac{1}{8} \end{pmatrix},\ g = \begin{pmatrix} 0 \\ \frac{5}{8} \end{pmatrix},\ x = \begin{pmatrix} 100 \\ 20 \end{pmatrix}. \tag{486a}$$

(486b)

The Euler method for this system is

$$x_{n+1} = x_n + h(Ax + g). \tag{487}$$

As for the scalar ODE, the step size criterion is

$$h_{\text{new}} = \left(\frac{2\varepsilon}{\|x_n''\|}\right)^{\frac{1}{2}}, \quad \textit{where } \|x\| = (x^T x)^{\frac{1}{2}}, \tag{488a}$$

$$h_0 = \left(\frac{2\varepsilon}{\|x_0''\|}\right)^{\frac{1}{2}}, \tag{488b}$$

with $x'' = Ax' = A^2x + Ag$. ▲

18.3 One-step LMMs

The general procedure of step size control for LMMs is essentially the same as that of TS methods. The main difference is that repeated differentiation of the ODE cannot be used to estimate the local trunction error, as this would negate the benefits of using LMMs. New technique is required to estimate higher order derivatives.

As an example of a 1-step LMM, we take the Euler method. The local truncation error of the method is given by

$$R_{n+1} = \frac{h_n^2}{2} x''(t_n) + O(h_n^3). \tag{489}$$

To approximate $x''(t_n)$, we Taylor expand $x(t_{n+1})$ and differentiate it

$$x'(t_{n+1}) = x'(t_n) + h_n x''(t_n) + \frac{h_n^2}{2} x'''(t_n) + O(h_n^3), \tag{490a}$$

$$x''(t_n) = \frac{x'(t_{n+1}) - x'(t_n)}{h_n} + O(h_n), \tag{490b}$$

$$x''(t_n) \approx \frac{x'_{n+1} - x'_n}{h_n}. \tag{490c}$$

Thus,

$$R_{n+1} = \frac{h_n}{2}(x'_{n+1} - x'_n), \tag{491}$$

and this can be used to update the step size:

$$h_{\text{new}} = h_n \left|\frac{\varepsilon}{R_{n+1}}\right|^{\frac{1}{2}}. \tag{492}$$

Example 18.4 (The Trapezoidal rule with adaptive step size control) *The goal of this example is to show how to develop the Trapezoidal rule with automatic step size control. As we know, the Trapezoidal rule is*

$$x_{n+1} - x_n = \frac{h_n}{2}(f_{n+1} + f_n). \tag{493}$$

Let us compute the local truncation error:

$$x_{n+1} = x_n + \frac{h_n}{2}(x'_{n+1} + x'_n), \tag{494a}$$

$$x'_{n+1} = x'_n + h_n x''_n + \frac{h_n^2}{2} x'''_n, \tag{494b}$$

$$x_{n+1} = x_n + \frac{h_n}{2}(x'_n + h_n x''_n + \frac{h_n^2}{2} x'''_n + x'_n). \tag{494c}$$

Thus, we have

$$R_{n+1} = -\frac{h_n^3}{12} x_n''' + O(h^4). \tag{495}$$

To compute $x'''(t_n)$, we can use the central difference approximation:

$$x'''(t_n) \approx \frac{x'_{n+1} - 2x'_n + x'_{n-1}}{h_n^2}. \tag{496}$$

Thus, the step size can be updated as

$$h_{\text{new}} = h_n \left| \frac{\varepsilon}{R_{n+1}} \right|^{\frac{1}{3}}, \ R_{n+1} = -\frac{h_n}{12} \left(x'_{n+1} - 2x'_n + x'_{n-1} \right). \tag{497}$$

The process is initiated with using two small time steps $h_0 = h_1 = \varepsilon$. ▲

18.4 Runge-Kutta methods

To use the step size control machinery with RK methods, typically, two related RK methods are used. Namely, one RK method is of order p, whereas the other is of order $p+1$. Let us assume that these two methods give numerical solutions $x_{n+1}^{<p>}$ and $x_{n+1}^{<p+1>}$, respectively. Then, the difference $x_{n+1}^{<p+1>} - x_{n+1}^{<p>}$ provides an estimate of the error of the lower order method. In particular, the error of the lower-order method is given by

$$e_n^{<p>} = x(t_n) - x_n^{<p>} = O(h^p), \tag{498a}$$

while the error of the higher-order method is

$$e_n^{<p+1>} = x(t_n) - x_n^{<p+1>} = O(h^{p+1}), \tag{499a}$$

where $x(t_n)$ is the exact solution. We can express $x(t_n)$ from (499a)

$$x(t_n) = x_n^{<p+1>} + O(h^{p+1})$$

and plug it into (498a):

$$e_n^{<p>} = x_n^{<p+1>} - x_n^{<p>} + O(h^{p+1}).$$

Thus, the difference between the numerical solution of the higher-order method and the lower-order method

$$x_n^{<p+1>} - x_n^{<p>} \tag{500}$$

can be used as an estimation of the leading term in the error of the lower order method. Generally speaking, such an approach would be inefficient, since it would involve computing two sets of k-values in both RK methods. This duplication can be avoided if the k-values of the lower order method are a subset of those of the higher order method. If this is the case then one can get two values for the price of one.

Example 18.5 (Adaptive step size control for Euler's method and the improved Euler method) *As an example of adaptive step size control, we consider two methods. The first one is the Euler method (1st order method in terms of global error)*

$$\begin{aligned} k_1 &= f(t_n, x_n), \\ x_{n+1}^{<1>} &= x_n + hk_1. \end{aligned} \tag{501}$$

The second method is the improved Euler method (2nd order method in terms of global error):

$$\begin{aligned} k_1 &= f(t_n, x_n), \\ k_2 &= f(t_n + h_n, x_n + h_n k_1), \\ x_{n+1}^{<2>} &= x_n + \frac{h}{2}(k_1 + k_2). \end{aligned} \tag{502}$$

Note that the k_1 *in the improved Euler method is computed in the Euler method, therefore we do not need to recompute it again.*

The local truncation error of the Euler method can be estimated as

$$R_{n+1} = x_{n+1}^{<2>} - x_{n+1}^{<1>}, \tag{503}$$

so that the step size control formula becomes

$$h_{new} = h_n \left| \frac{\varepsilon}{R_{n+1}} \right|^{\frac{1}{2}}, \tag{504}$$

where

$$R_{n+1} = x_n + \frac{h_n}{2}(k_1 + k_2) - (x_n + h_n k_1) = \frac{h_n}{2}(k_2 - k_1). \tag{505}$$

▲

Example 18.6 (Adaptive step size control for RK(2,3) method) *Let us consider a third-order method with the Butcher tableau*

$$\begin{array}{c|ccc} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ \frac{1}{2} & \frac{1}{4} & \frac{1}{4} & 0 \\ \hline & \frac{1}{6} & \frac{1}{6} & \frac{2}{3} \end{array}$$

with

$$\widetilde{k}_1 = f(t_n, x_n), \tag{506a}$$

$$\widetilde{k}_2 = f(t_n + h, x_n + h k_1), \tag{506b}$$

$$\widetilde{k}_3 = f\left(t_n + \frac{h}{2}, x_n + \frac{h}{4}k_1 + \frac{h}{4}k_2\right), \tag{506c}$$

$$x_{n+1}^{<3>} = x_n + h\left(\frac{\widetilde{k}_1}{6} + \frac{\widetilde{k}_2}{6} + \frac{2}{3}\widetilde{k}_3\right). \tag{506d}$$

If we consider the improved Euler method

$$\begin{array}{c|cc} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 1 & 0 \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array}$$

with

$$k_1 = f(t_n, x_n), \tag{507a}$$

$$k_2 = f(t_n + h, x_n + h), \tag{507b}$$

$$x_{n+1}^{<2>} = x_n + \frac{h}{2}(k_1 + k_2). \tag{507c}$$

As we can see, the first two stages of the original 3-stage RK method $\widetilde{k}_1$ *and* $\widetilde{k}_2$ *can be used to compute the two stages of the improved Euler method, since*

$$k_1 = \widetilde{k}_1,\ k_2 = \widetilde{k}_2. \tag{508}$$

This gives the following step size control formula:

$$h_{new} = h_n \left| \frac{\varepsilon}{R_{n+1}} \right|^{\frac{1}{3}},\ R_{n+1} = x_{n+1}^{<3>} - x_{n+1}^{<2>}. \tag{509}$$

▲

Lecture 19 Boundary value problems

In this lecture, we will consider Boundary Value Problems (BVPs). Let us start from a first-order initial value problem (IVP) of the form

$$x' = f(t,x), \quad x(t_0) = x_0, \quad t \in [t_0, t_N]. \tag{510}$$

Since this is an IVP we need an initial condition to define the solution $x(t)$ uniquely.

In case of a second-order equation

$$x''(t) = f(t,x,x'), \quad t \in [t_0, t_N], \tag{511}$$

we need two conditions

$$x(t_0) = x_0, \tag{512a}$$

$$x'(t_0) = x'_0. \tag{512b}$$

Note that it is still an IVP, since these two conditions are given at time t_0. If, however, we prescribe the solution at two different points t_0 and t_N then we have a boundary value problem:

$$x'' = f(t,x,x'), \quad x(t_0) = x_0, \quad x(t_N) = x_N, \quad t \in [t_0, t_N].$$

19.1 The shooting method

19.1.1 Linear case

Let us consider a second order ODE of the form

$$x'' + c(t)x' + d(t)x = g(t), \quad t \in [t_0, t_N] \tag{513a}$$

subject to the boundary conditions

$$x(t_0) = \alpha, \tag{513b}$$

$$x(t_N) = \beta. \tag{513c}$$

Our goal is to solve BVP (513a)-(513c). Up to now, we have used numerical methods to solve initial value problems. Can we use the same methods to solve boundary value problems? Let us try it. For this, we replace the boundary conditions (513) with the initial conditions:

$$\begin{aligned} x(t_0) &= \alpha, \\ x'(t_0) &= \gamma, \end{aligned} \tag{514}$$

where γ is unknown. We now have to solve the IVP

$$x'' + c(t)x' + d(t)x = g(t), \quad t \in [t_0, t_N], \tag{515a}$$

$$x(t_0) = \alpha, \tag{515b}$$

$$x'(t_0) = \gamma. \tag{515c}$$

Let the solution of this IVP be $\omega(t)$. The solution $\omega(t_N)$ can be thought of as an estimation of the solution $x(t_N)$. In general, $\omega(t_N) \neq x(t_N)$. Therefore, we now solve IVP (515) with another set of initial conditions $x(t_0) = \alpha,\ x'(t_0) = \delta$ to compute another approximated solution, which we denote by $r(t)$. Again, in general $r(t_N) \neq x(t_N)$.

Now, we take a linear combination of both solutions

$$x(t) = \theta\omega(t) + (1-\theta)r(t), \tag{516}$$

which is the solution of BVP (513a)-(513c) with θ chosen so that $x(t_N)$ fulfills (513c). Assuming that $\omega(t_N) \neq r(t_N)$, we can isolate θ from (542):

$$\theta = \frac{x(t_N) - r(t_N)}{\omega(t_N) - r(t_N)} = \frac{\beta - r(t_N)}{\omega(t_N) - r(t_N)}. \tag{517}$$

19.1.2 Nonlinear case

Let us consider the general nonlinear equation of the form

$$x'' = f(t, x, x'), \quad t \in [t_0, t_N], \tag{518a}$$
$$x(t_0) = \alpha, \tag{518b}$$
$$x(t_N) = \beta. \tag{518c}$$

Following the procedure considered in the linear case, we replace the last boundary value (518c) with the initial condition

$$x'(t_0) = s, \tag{518d}$$

where s is unknown.

Starting with a first guess for s, we solve (518a) subject to the initial conditions

$$x(t_0) = \alpha, \tag{519a}$$
$$x'(t_0) = s, \tag{519b}$$

and denote its solution by $x(t, s)$, which is an estimation of the boundary condition $x(t_N) = \beta$. Our goal is to find a value of s, say s^*, such that $x(t, s^*) = \beta$. In contrast with the linear case, finding the value of s^* requires the solution of a nonlinear algebraic equation. To solve this equation, we transform the second order ODE (518a) and the initial conditions (519) into a system of second order. For this, we use the change of variables

$$\begin{aligned} u(t, s) &= x(t, s), \\ v(t, s) &= \frac{\partial x(t, s)}{\partial t}. \end{aligned} \tag{520}$$

Hence, we can rewrite (518a) and (519) as follows:

$$\frac{\partial u(t, s)}{\partial t} = v(t, s), \tag{521a}$$
$$\frac{\partial v(t, s)}{\partial t} = f\Big(t, u(t, s), v(t, s)\Big), \tag{521b}$$
$$u(t_0, s) = \alpha, \tag{521c}$$
$$v(t_0, s) = s. \tag{521d}$$

The solution $u(t, s)$ of the initial value problem (521) coincides with the solution of BVP (518a)-(518c) provided that we can find a value of s such that

$$\varphi(s) = u(t_N, s) - \beta = 0. \tag{522}$$

The nonlinear equation (522) can be solved with any standard method for nonlinear equations. We will consider two methods: the method of bisection and the Newton method.

The method of bisection

Let us suppose that we have already solved the initial value problem (521) for two different values s_1 and s_2. Thus, we have

$$\varphi(s_1) = u(t_N, s_1) - \beta, \tag{523a}$$
$$\varphi(s_2) = u(t_N, s_2) - \beta. \tag{523b}$$

Let us assume that

$$\varphi(s_1) < 0 \text{ and } \varphi(s_2) > 0. \tag{524}$$

If this is not the case, then one should take a larger interval $[s_1, s_2]$. We also assume, for the sake of definiteness, that $s_1 < s_2$.

Given that the solution of the initial value problem (521) depends continuously on the initial data, there exists at least one value of s in the interval $[s_1, s_2]$ such that $\varphi(s) = 0$. Thus, the interval $[s_1, s_2]$ contains the root of equation (522). This root can be found by the method of bisection.

Take the midpoint, say s_3, of the interval $[s_1, s_2]$ and solve IVP (521) with $v(t_0, s) = s_3$. Its solution at time t_N is $u(t_N, s_3)$. Now compute $\varphi(s_3) = u(t_N, s_3) - \beta$.

If $\varphi(s_3) = 0$ then the s_3 is the root, and we stop the computation.

If $\varphi(s_3) > 0$ then the interval $[s_1, s_3]$ contains the root of $\varphi(s)$, and we have to take the midpoint of this interval, s_4 and solve the IVP again with $v(t_0, s) = s_4$.

If $\varphi(s_3) < 0$ then the interval $[s_3, s_2]$ contains the root of $\varphi(s)$, and we have to take the midpoint of this interval, s_4 and solve the IVP again with $v(t_0, s) = s_4$.

Repeating this process, one can construct a sequence of numbers $\{s_n\}_{n=1}^{\infty}$ converging to the root s. In practice, the bisection is terminated after a finite number of steps when the length of the interval containing s has become sufficiently small.

The Newton method

To solve the equation

$$\varphi(s) = 0 \tag{525}$$

one can use the Newton method

$$s_{n+1} = s_n - \frac{\varphi(s_n)}{\varphi'(s_n)}, \tag{526}$$

with the starting value s_0. A suitable s_0 can be found by performing a few steps of the method of bisection. The Newton method has quadratic convergence, and hence it is much faster than the method of bisection. The first question is how to find $\varphi'(s_n)$. In order to do it, we introduce new variables

$$\xi(t, s) = \frac{\partial u(t, s)}{\partial s},\ \eta(t, s) = \frac{\partial v(t, s)}{\partial s}, \tag{527}$$

and differentiate IVP (521) with respect to s

$$\frac{\partial^2 u}{\partial t \partial s} = \frac{\partial v(t,s)}{\partial s} \Rightarrow \frac{\partial \xi}{\partial t} = \eta, \tag{528a}$$

$$\frac{\partial^2 v}{\partial t \partial s} = \frac{\partial f}{\partial u}\frac{\partial u}{\partial s} + \frac{\partial f}{\partial v}\frac{\partial v}{\partial s} \Rightarrow \frac{\partial \eta}{\partial t} = \frac{\partial f}{\partial u}\xi + \frac{\partial f}{\partial v}\eta, \tag{528b}$$

$$\frac{\partial u(t_0,s)}{\partial s} = 0 \Rightarrow \xi(t_0,s) = 0, \tag{528c}$$

$$\frac{\partial v(t_0,s)}{\partial s} = 1 \Rightarrow \eta(t_0,s) = 1. \tag{528d}$$

Then, we solve IVP (521) with $v(t_0,s) = s_n$ to get $u(t_N, s_n)$ from which we can calculate $\varphi(s_n) = u(t_N, s_n) - \beta$, and in addition, we obtain $\varphi'(s_n) = \xi(t_N, s_n)$. Having computed $\varphi(s_n)$ and $\varphi'(s_n)$, we can compute the next iteration

$$s_{n+1} = s_n - \frac{\varphi(s_n)}{\varphi'(s_n)}. \tag{529}$$

This process is repeated until

$$\frac{|s_{n+1} - s_n|}{|s_{n+1}|} < \varepsilon, \tag{530}$$

where ε is a given tolerance. Note that we have to solve a coupled system of two IVPs (521) and (528) in each iteration of the Newton method.

Lecture 20 Finite Difference Method

The idea behind the Finite Difference Method (FDM) is to approximate derivatives in the differential equation by linear combinations of function values at grid points.

20.1 1D case

Let us consider an ODE of the form

$$a(t)\frac{d^2x}{dt^2} + b(t)\frac{dx}{dt} + c(t)x = d(t),\ t \in [t_0, t_N], \tag{531}$$

where $a(t)$, $b(t)$, $c(t)$ are some real-valued functions. Consider the Dirichlet BVP (the solution is given on the boundary)

$$x(t_0) = \alpha,\ x(t_N) = \beta. \tag{532}$$

In order to use the FDM, we need to introduce a grid

$$t_i = ih, \quad \frac{t_0 - t_N}{N}, \quad i = 0, 1, \ldots, N. \tag{533}$$

Now, we can approximate the 1st and 2nd order derivatives in (531). Let us use the central difference approximation (scheme) for both the first and second order derivatives:

$$a_i\frac{x_{i+1} - 2x_i + x_{i-1}}{h^2} + b_i\frac{x_{i+1} - x_{i-1}}{2h} + c_i x_i = d_i, \tag{534}$$

with $i = 1, 2, \ldots, N-1$, and we set $x_0 = \alpha$ and $x_N = \beta$.

$$i = 1 \qquad a_1\frac{x_2 - 2x_1 + \overbrace{x_0}^{\alpha}}{h^2} + b_1\frac{x_2 - \overbrace{x_0}^{\alpha}}{2h} + c_1x_1 = d_1, \tag{535a}$$

$$i = 2 \qquad a_2\frac{x_3 - 2x_2 + x_1}{h^2} + b_2\frac{x_3 - x_1}{2h} + c_2x_2 = d_2, \tag{535b}$$

$$i = 3 \qquad a_3\frac{x_4 - 2x_3 + x_2}{h^2} + b_3\frac{x_4 - x_2}{2h} + c_3x_3 = d_3, \tag{535c}$$

$$\ldots \tag{535d}$$

$$i = N-1 \qquad a_{N-1}\frac{\overbrace{x_N}^{\beta} - 2x_{N-1} + x_{N-2}}{h^2} + b_{N-1}\frac{\overbrace{x_N}^{\beta} - x_{N-2}}{2h} + c_{N-1}x_{N-1} = d_{N-1}. \tag{535e}$$

We can also write the linear system of equations (535) in the matrix form $\mathbf{Ax} = \mathbf{b}$, i.e.

$$\begin{pmatrix} -\frac{2a_1}{h^2} + c_1 & \frac{a_1}{h^2} + \frac{b_1}{2h} & 0 & 0 & 0 & 0 \\ \frac{a_2}{h^2} - \frac{b_2}{2h} & -\frac{2a_2}{h^2} + c_2 & \frac{a_2}{h^2} + \frac{b_2}{2h} & 0 & 0 & 0 \\ 0 & \frac{a_3}{h^2} - \frac{b_3}{2h} & -\frac{2a_3}{h^2} + c_3 & \frac{a_3}{h^2} + \frac{b_3}{2h} & 0 & 0 \\ \ldots & & & & & \\ 0 & 0 & 0 & 0 & \frac{a_{N-1}}{h^2} - \frac{b_{N-1}}{2h} & -\frac{2a_{N-1}}{h^2} + c_{N-1} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \ldots \\ x_{N-1} \end{pmatrix}$$
$$= \begin{pmatrix} d_1 - (\frac{a_1\alpha}{h^2} + \frac{b_1\alpha}{2h}) \\ d_2 \\ d_3 \\ \ldots \\ d_{N-1} - (\frac{a_{N-1}\beta}{h^2} + \frac{b_{N-1}\beta}{2h}) \end{pmatrix} \tag{536}$$

Note that the matrix is of size $(N-1)\times(N-1)$, since the solution is given on the left and right boundaries, and we do not have to compute it.

Let us consider the Neumann BVP and study how the system of linear equations changes. The Neumann boundary condition is a boundary condition that specifies the derivative of the solution:

$$\left.\frac{dx}{dt}\right|_{t=t_0} = \alpha, \quad \left.\frac{dx}{dt}\right|_{t=t_N} = \beta. \tag{537}$$

If we approximate the derivative on the boundary with the central difference scheme we have

$$\frac{dx}{dt} \approx \frac{x_{i+1}-x_{i-1}}{2h}.$$

Therefore, at $t=t_0$ we have

$$\frac{x_{i+1}-x_{i-1}}{2h} = \alpha \Rightarrow x_{i-1} = x_{i+1} - 2\alpha h. \tag{538}$$

and at $t = t_N$ we get

$$\frac{x_{i+1}-x_{i-1}}{2h} = \beta \Rightarrow x_{i+1} = x_{i-1} + 2\beta h. \tag{539}$$

Thus, for $i=0$ we have

$$a_0\frac{x_1-2x_0-x_{-1}}{h^2} + b_0\frac{x_1-x_{-1}}{2h} + c_0x_0 = d_0. \tag{540}$$

Using (538) in (540) gives

$$a_0\frac{x_1-2x_0-(x_1-2\alpha h)}{h^2} + b_0\frac{x_1-(x_1-2\alpha h)}{2h} + c_0x_0 = d_0. \tag{541}$$

For $i=N$ we have

$$a_N\frac{x_{N+1}-2x_N+x_{N-1}}{h^2} + b_N\frac{x_{N+1}-x_{N-1}}{2h} + c_Nx_N = d_N. \tag{542}$$

Using (539) in (542) gives

$$a_N\frac{(x_{N-1}+2\beta h)-2x_N+x_{N-1}}{h^2} + b_N\frac{(x_{N-1}+2\beta h)-x_{N-1}}{2h} + c_Nx_N = d_N. \tag{543}$$

Thus, the system of equations (536) becomes of size $N\times N$, and the first and the last rows of the matrix and the right hand side are defined by equations (541) and (543), respectively.

20.2 The method of undetermined coefficients

The method of undetermined coefficients within the context of finite difference approximations is used to find an approximation to a derivative with a given order of accuracy. Suppose we have to approximate $\frac{dx(t_n)}{dt}$ with a third order of accuracy using information at points $x(t_n)$, $x(t_{n-1})$, and $x(t_{n-2})$. The first step of the method of undetermined coefficients is to fix the form of the approximation. Let it be

$$\frac{dx(t_n)}{dt} = ax(t_n) + bx(t_{n-1}) + cx(t_{n-2}). \tag{544}$$

The next step is to Taylor expand $x(t_{n-1})$ and $x(t_{n-2})$ about t_n

$$x(t_{n-1)} = x(t_n) - hx'(t_n) + \frac{h^2}{2}x''(t_n) + O(h^3), \tag{545a}$$
$$x(t_{n-2}) = x(t_n) - 2hx'(t_n) + 2h^2x''(t_n) + O(h^3), \tag{545b}$$

and plug these expansions into (544)

$$\frac{dx(t_n)}{dt} = ax(t_n) + b\left(x(t_n) - hx'(t_n) + \frac{h^2}{2}x''(t_n)\right) + c\Big(x(t_n) - 2hx'(t_n) + 2h^2x''(t_n)\Big) + O(h^3) \tag{546a}$$
$$= (a+b+c)x(t_n) - (b+2c)hx'(t_n) + \frac{h^2}{2}(b+4c)x''(t_n) + O(h^3). \tag{546b}$$

In order to have a third order approximation to $x'(t_n)$, we have to solve the system

$$a + b + c = 0, \tag{547a}$$
$$b + 2c = -\frac{1}{h}, \tag{547b}$$
$$b + 4c = 0. \tag{547c}$$

The solution is

$$a = \frac{3}{2h},\ b = -\frac{2}{h},\ c = \frac{1}{2h}. \tag{548}$$

Thus, the finite difference approximation of $\frac{dx(t_n)}{dt}$ is

$$\frac{dx(t_n)}{dt} \approx \frac{3}{2h}x(t_n) - \frac{2}{h}x(t_{n-1}) + \frac{1}{2h}x(t_{n-2}).$$

Can you check that this approximation is of order 3?

Lecture 21 Partial differential equations

In this lecture we focus on how to use the methods studied in the course for partial differential equations. Before going into details, it is instructive to introduce some definitions.

Definition 21.1 (Consistency) *A finite difference scheme $P_h u_h = f$ is called consistent with a partial differential equation $Pu(t,x) = f(t.x)$ if $\|P_h u_h - Pu\| \to 0$ as $h \to 0$, where P is a differential operator and $h = (\Delta t, \Delta x)$.*

Definition 21.2 (Stability) *A finite difference scheme is called stable for a partial differential equation if the numerical solution remains bounded as the space and time steps tend to zero.*

Theorem 21.1 (The Lax-Richtmyer equivalence theorem) *A consistent finite difference scheme for a well-posed linear BVP is convergent if and only if it is stable.*

This theorem is the analog of the Dahlquist equivalence theorem for ordinary differential equations.

Let us consider a one-dimensional partial differential equation

$$\frac{\partial u}{\partial t} = -c\frac{\partial u}{\partial x}, \quad t \in [0,T], \quad x \in [0,L], \quad c = const \tag{549}$$

subject to initial and boundary conditions. In order to solve this equation numerically, we have to approximate it in both space and time. For space approximation we will use the finite difference method, while for time approximation we can use any method studied in the course. For this, we introduce the following space grid

$$x_i = i\Delta x, \quad i = 0,1,\ldots,M, \quad \Delta x = \frac{L}{M}, \tag{550}$$

and the time grid

$$t_i = n\Delta t, \quad n = 0,1,\ldots,N, \quad \Delta t = \frac{T}{N}, \tag{551}$$

where Δx and Δt are the space and time steps, respectively.

Let us use the central difference to approximate the space derivative in (549), namely

$$\frac{du_i}{dt} = -c\frac{u_{i+1} - u_{i-1}}{2\Delta x}, \quad i = 0,1,\ldots,M. \tag{552}$$

Note that this is a semi-discrete system of equations, since it is only approximated in space; the sub-index refers to the space approximation. To solve this equation, we can use one of the methods studied in the course. Let us use the Euler method

$$u_i^{n+1} = u_i^n - \Delta tc\frac{u_{i+1}^n - u_{i-1}^n}{2\Delta x}, \quad i = 0,1,\ldots,M, \quad n = 0,1,\ldots,N. \tag{553}$$

Here, the super index refers to the time approximation.

As with numerical methods for ODEs, we have to ensure that the proposed method is convergent, otherwise it is of no use to us. In order to check whether the method is convergent we will consider different approaches.

21.1 Von Neumann stability analysis (Fourier stability analysis)

Let us assume that a numerical scheme admits a solution of the form

$$u_j^n = \zeta^n(k)e^{ikj\Delta x}, \tag{554}$$

where $i = \sqrt{-1}$, and k is the wave number defined as

$$k = \frac{\pi j}{L},\ j = 0, \ldots, M,\ M = \frac{L}{\Delta x}. \tag{555}$$

Define the amplification factor

$$\zeta(k) = \frac{\zeta^{n+1}(k)}{\zeta^n(k)}. \tag{556}$$

Then, the von Neumann stability condition is given by

$$|\zeta(k)| \le 1, \quad \forall k. \tag{557}$$

In other words, if this condition is met then the numerical method is stable.

In order to check (557) we substitute u_j^n into (553) and get

$$\zeta^{n+1} e^{ikj\Delta x} = \zeta^n e^{ikj\Delta x} - c\frac{\zeta^n e^{ik(j+1)\Delta x} - \zeta^n e^{ik(j-1)\Delta x}}{2\Delta x}\Delta t, \tag{558a}$$

$$|\zeta(k)| = \left|\frac{\zeta^{n+1}}{\zeta^n}\right| = \left|1 - \lambda(e^{ik\Delta x} - e^{-ik\Delta x})\right|,\ \lambda := \frac{c\Delta t}{2\Delta x} \tag{558b}$$

$$= \left|1 - \lambda(\cos(k\Delta x) + i\sin(k\Delta x) - \cos(k\Delta x) + i\sin(k\Delta x)\right|, \tag{558c}$$

$$= \left|1 - \frac{c\Delta t}{\Delta x} i \sin(k\Delta x)\right| \tag{558d}$$

$$= \sqrt{1 + \left(-\frac{c\Delta t}{\Delta x}\sin(k\Delta x)\right)^2} > 1. \tag{558e}$$

Thus the scheme is unstable, and cannot therefore be used to solve the equation. However, this scheme can be made stable by the Lax substitution.

21.1.1 The Lax substitution

The idea behind the Lax substitution is to replace u_j^n with

$$u_j^n = \frac{u_{j+1}^n + u_{j-1}^n}{2}. \tag{559}$$

Substitution of (559) into (553) gives

$$u_j^{n+1} = \frac{u_{j+1}^n + u_{j-1}^n}{2} - \frac{c\Delta t}{2\Delta x}\left(u_{j+1}^n - u_{j-1}^n\right). \tag{560}$$

Thus, the amplification factor becomes

$$|\zeta| = \left|\cos(k\Delta x) - i\frac{c\Delta t}{\Delta x}\sin(k\Delta x)\right| \le 1, \tag{561}$$

if

$$\Delta t \le \frac{\Delta x}{c}. \tag{562}$$

This condition is known as the Courant–Friedrichs–Lewy (CFL) condition. Physically, the CFL condition means that the distance the fluid particle (or the wave as in our case) can cover during one time step must not exceed the space step size Δx (i.e. the grid must "see" the movement of the wave).

To see the effect of the Lax substitution onto the PDE, we rewrite (560) in the form:

$$\frac{u_j^{n+1}-u_j^n}{\Delta t} = -c\frac{u_{j+1}^n-u_{j-1}^n}{2\Delta x} + \frac{(\Delta x)^2}{2\Delta t}\frac{u_{j+1}^n-2u_j^n+u_{j-1}^n}{(\Delta x)^2} \tag{563a}$$

$$\text{since } \frac{\partial^2 u}{\partial x^2} \approx \frac{u_{j+1}^n-2u_j^n+u_{j-1}^n}{(\Delta x)^2} \text{ we have} \tag{563b}$$

$$\frac{\partial u}{\partial t} = -c\frac{\partial u}{\partial x} + \frac{(\Delta x)^2}{2\Delta t}\frac{\partial^2 u}{\partial x^2} \tag{563c}$$

The last term in the equation is called dissipation. It inhibits growing modes and keeps the numerical scheme stable. However, such an artificial dissipation (it is artificial, since it is not a part of the original PDE) can significantly contaminate the solution, i.e. it can dissipate too much. To get rid off the numerical dissipation, one can use the Leap-frog scheme.

21.1.2 Leap-frog scheme

As we know from the course (see equation (214) on page 38), the Leap-frog scheme for the ODE $x' = f(t,x)$ is given by

$$x_{n+1} - x_{n-1} = 2hf_n, \quad h = \Delta t. \tag{564}$$

Using (564) in equation (552) gives

$$\frac{u_j^{n+1}-u_j^{n-1}}{2\Delta t} = -c\frac{u_{j+1}^n-u_{j-1}^n}{2\Delta x}, \tag{565a}$$

$$u_j^{n+1}-u_j^{n-1} = -\frac{c\Delta t}{\Delta x}\left(u_{j+1}^n-u_{j-1}^n\right). \tag{565b}$$

Substitution of $u_j^n = \zeta^n e^{ikj\Delta x}$ into (565b) results in

$$\zeta^{n+1}e^{ikj\Delta x} - \zeta^{n-1}e^{ikj\Delta x} = -\frac{c\Delta t}{\Delta x}\zeta^n\left(e^{ik(j+1)\Delta x} - e^{ik(j-1)\Delta x}\right) \tag{566a}$$

$$\frac{\zeta^{n+1}}{\zeta^{n-1}} - 1 = -\frac{c\Delta t}{\Delta x}\zeta 2i\sin(k\Delta x) \tag{566b}$$

$$\zeta^2 = 1 - \frac{c\Delta t}{\Delta x}\zeta 2i\sin(k\Delta x) \tag{566c}$$

$$\text{we have to solve the quadratic equation to find } \zeta \tag{566d}$$

$$\zeta_{1,2} = -i\frac{c\Delta t}{\Delta x}\sin(k\Delta x) \pm \sqrt{1-\left(\frac{c\Delta t}{\Delta x}\sin(k\Delta x)\right)^2}. \tag{566e}$$

Thus we have

$$|\zeta_{1,2}| = 1, \ \text{ if } \Delta t \le \frac{\Delta x}{c}. \tag{567}$$

This means that there is no diffusion for the the Leapfrog scheme. Note that if $|\zeta| < 1$ then there is a dissipation, and if $|\zeta| > 1$ than the modes amplitude grow. However, the Leap-from scheme still requires the CFL condition to ensure its stability. Is there a method that can offer an unconditionally stable scheme?

21.1.3 The Backward Euler method

Let us apply the Backward Euler method to equation (549) and analyse the stability of the numerical scheme. The backward Euler method for the equation $x' = f(t,x)$ is given by $x_{n+1} = x_n + hf_{n+1}$, where $h = \Delta t$. Using the Backward Euler method in (552) leads to

$$\frac{u_j^{n+1} - u_j^n}{\Delta t} = -c\frac{u_{j+1}^{n+1} - u_{j-1}^{n+1}}{2\Delta x}. \tag{568}$$

Note that the right hand side of equation (568) depends on the solution at time t_{n+1}. This is because the Backward Euler method is an implicit method.

Substitution of $u_j^n = \zeta^n e^{ikj\Delta x}$ into (568) gives

$$\zeta^{n+1} e^{ikj\Delta x} = \zeta^n e^{ikj\Delta x} - \lambda\zeta^{n+1}\left(e^{i(j+1)k\Delta x} - e^{i(j-1)k\Delta x}\right), \quad \lambda := \frac{c\Delta t}{2\Delta x}. \tag{569}$$

Dividing (569) by $\zeta^n e^{ikj\Delta x}$ yields

$$\zeta = 1 - \lambda\zeta\left(e^{ik\Delta x} - e^{-ik\Delta x}\right) \Rightarrow \Big(1 + \lambda\big(e^{ik\Delta x} - e^{-ik\Delta x}\big)\Big)\zeta = 1 \tag{570a}$$

$$\zeta = \frac{1}{1 + \lambda 2i\sin(k\Delta x)} \tag{570b}$$

$$|\zeta| = \left|\frac{1}{1 + \dfrac{c\Delta t}{2\Delta x}2i\sin(k\Delta x)}\right| \le 1 \tag{570c}$$

$$|\zeta| = \frac{1}{\sqrt{1^2 + \left(\frac{c\Delta t}{\Delta x}\right)^2 \sin^2(k\Delta x)}} \le 1 \text{ for all } \Delta t \text{ and } \Delta x, \tag{570d}$$

since $\sqrt{1^2 + \left(\frac{c\Delta t}{\Delta x}\right)^2 \sin^2(k\Delta x)} \ge 1$ for all Δt and Δx. A numerical scheme the stability of which does not depend on the time and space steps is called absolutely stable. Thus, the Backward Euler method and the central difference approximation of the first order space derivative is an absolutely stable scheme for equation (549).

Although von Neumann stability analysis allows to analysis the stability of numerical methods, it is restricted to linear equations with constant coefficients, and does not account for boundary conditions. The matrix stability analysis is free of some of these shortcomings, but as von Neumann analysis it works for linear equations or systems of equations.

21.2 The matrix stability analysis

The matrix stability analysis can be used for linear equations of the form

$$\frac{d\mathbf{u}}{dt} = \mathbf{A}\mathbf{u} + \mathbf{b}, \tag{571}$$

where $\mathbf{A} \in \mathbb{R}^{M\times M}$ is a constant matrix, $\mathbf{b} \in \mathbb{R}^{M\times 1}$ is a constant vector, and $\mathbf{u} = \mathbf{u}(t)$ is a vector function. Equation (571) is a semi-discrete system of equation akin to (552). Since $\mathbf{b}$ is a constant vector, the behaviour of the solution is essentially determined by the eigenvalues of matrix $\mathbf{A}$, and therefore we can study a homogeneous equation of the form

$$\frac{d\mathbf{u}}{dt} = \mathbf{A}\mathbf{u}, \tag{572}$$

and use the stability analysis for systems of ODEs (see lecture 10, page 57).

Lecture 22 The energy method (The method of energy inequalities) by example

The energy method (also known as the method of energy inequalities) is free of shortcoming of von Neumann analysis and the matrix stability analysis, and can be applied to nonlinear equations both continuous and discrete. The principle behind the energy method is to show that the solution of a given problem is bounded by something that we can control.

Let us consider an ODE of the form

$$\frac{d^2y}{dx^2} + f(x) = 0, \quad x \in [0,1], \quad y(0) = y(1) = 0. \tag{573}$$

Here, $f(x)$ is a function of x. The application of the Finite Difference method (approximate the derivative with the central difference scheme) to (573) results in

$$\frac{y_{i+1} - 2y_i + y_{i-1}}{h^2} + f_i = 0, \quad y_0 = y_N = 0, \quad i = 0, 1, \ldots, N, \quad h = \frac{1}{N}. \tag{574}$$

Using the notation

$$y_x = \frac{y_{i+1} - y_i}{h}, \; y_{\overline{x}} = \frac{y_i - y_{i-1}}{h}, \tag{575}$$

multiplying (574) by hy, and summing up over the grid nodes i we have

$$\sum_{i=1}^{N-1} (y_{\overline{x}x})_i y_i h + \sum_{i=1}^{N-1} f_i y_i h = 0, \tag{576}$$

which can be recast in terms of the scalar product as follows:

$$(y_{\overline{x}x}, y) + (f, y) = 0, \tag{577}$$

or

$$-(y_{\overline{x}}, y_{\overline{x}}] + (f, y) = 0, \tag{578}$$

where $(u, v] = h\sum_{i=1}^{N} u_i v_i$.

Can you show that (577) **can be written as** (578)**?**

In terms of l_2-norm, $\|u]|^2 = \sum_{i=1}^{N} u_i^2$ and $\|u\|^2 = \sum_{i=1}^{N-1} u_i^2$, equation (578) becomes

$$\|y_{\overline{x}}]|^2 = (f, y), \tag{579}$$

Using the Cauchy-Schwarz inequality ($|(f, y)| \leq \|f\|\|y\|$) on the right hand side we have

$$\|y_{\overline{x}}]|^2 \leq \|f\|\|y\|. \tag{580}$$

To proceed we need

Lemma 22.1 *For any continuous function $y(x)$ defined on an equidistant grid*

$$\omega_h = \{x_i = ih, i = 0, 1, \ldots, N, \;\; x_0 = 0, x_N = l\}, \tag{581}$$

and $y(x_0) = 0$, $y(x_N) = 0$, the following estimate holds

$$\frac{h^2}{4}\|y_{\overline{x}}]|^2 \leq \|y\|^2 \leq \frac{l^2}{8}\|y_{\overline{x}}]|^2. \tag{582}$$

Lemma 22.2 *For any continuous function $y(x)$ defined on the grid*

$$\omega_h = \{x_i = ih, 0 \leq i \leq N, \ \ x_0 = 0, x_N = l\}, \tag{583}$$

and $y(x_0) = y(x_N) = 0$, the following estimation holds

$$\|y\|_c \leq \frac{\|y_{\bar{x}}]|}{2}, \quad \|y\|_c = \max_{x \in \omega_h} |y(x)|. \tag{584}$$

Based on Lemma 22.1 ($l = 1$ in our case), we have

$$\|y_{\bar{x}}]| \leq \frac{\|f\|}{\sqrt{8}}, \tag{585}$$

while using Lemma 22.2 in inequality (585) gives

$$\|y\|_c \leq \frac{\|f\|}{2\sqrt{8}}. \tag{586}$$

Thus we have shown that the solution y does not grow if $\|f\|$ is bounded.