LECTURE NOTES FOR

COMPUTATIONAL METHODS FOR ENGINEERING APPLICATIONS (CMEA - II)

Siddhartha mishra

Introduction

The aim of this course is to present numerical (computational) methods for discretizing (approximating, Simulating) differential equations that arise in engineering.

1.1. What is a differential Equation?

Let x denote the (spatial) position, $x \in \mathbb{R}, \mathbb{R}^2, \mathbb{R}^3$ depending on the dimension of your problem $x = (x_i)_{i=1,2,3}$

Let t denote the time variable, $t \in [0,T]$ for some final time T.

Let $u = u(x,t)$ be a function.

u could be a Scalar i.e. $u \in \mathbb{R}$

or

u could be a vector, i.e $u \in \mathbb{R}^M$, for some $M \geq 1$

(we will consider examples of both);

Denote the partial derivatives are:

$$u_t = \frac{\partial u}{\partial t}, \quad u_{tt} = \frac{\partial^2 u}{\partial t^2}, \; \dots \quad u_{tt\dots t} = \frac{\partial^k u}{\partial t^k}, \dots$$

$$u_{x_i} = \frac{\partial u}{\partial x_i}, \quad \cancel{\frac{\partial^2 u}{\partial x}} \quad u_{x_i x_j} = \frac{\partial^2 u}{\partial x_i \, \partial x_j}, \; \dots$$

and so on

then a Differential equation is given by

$$\mathbb{F}\left(x, t, u, u_t, u_{x_i}, u_{tt}, u_{x_i x_j}, \dots \right) = 0 \qquad - (1.1)$$

(1.1) implies that the function u and its partial derivatives are ~~specified~~ related.

The task of solving (1.1) amounts to finding u, given information on its partial derivatives.

The abstract form (1.1) is not very instructive. We need to look at concrete cases of differential equations.

1.2) Ordinary differential equations (ODEs)

ODEs are the simplest type of differential equations. we assume that $u = u(t)$ is the unknown of interest i.e, we consider a (scalar or vector) function of one variable (say time). Then (1.1) is:

$$\mathbb{F}(t, u, u_t, u_{tt}, \cdots\cdots, \ldots) = 0 \quad - (1.2).$$

The task of solving (1.2) amounts to finding u, given information about its derivatives !!!

A simple example of (1.2) is instructive:

$$u_t = 1 \quad - (1.3)$$

Clearly all solutions of (1.3) are of the "general" form.

$$u(t) = t + C \quad \text{(for some constant } C\text{)}$$

We need to specify C in order to find a unique solution of (1.3). We can do so by setting the value of C initially i.e.

$$u(0) = C.$$

This leads us to considering initial value problems (IVPs)

1.3: IVPs for ODEs

Denote
$$u' = u_t$$
$$u'' = u_{tt}$$
$$\vdots$$
$$u^{(k)} = \underbrace{u_{tt\cdots t}}_{k\text{-times}}$$

Then the general form of an IVP for ODEs is given by,

$$\mathbb{F}(t, u, u', u'', u^{(3)}, \dots u^{(k)}) = 0$$
$$u(0) = u_0$$
$$u'(0) = u_1 \qquad \text{——} \qquad (1.4)$$
$$\vdots$$
$$u^{(k)}(0) = u_k$$

Thus, solving (1.4) amounts to finding a function $u(t)$, given a relation between its derivatives and their starting values;

IVPs for ODEs arise in a wide variety of models in engineering. A few prototypical examples are in order.

1.3.1: Single particle (body) dynamics.

Consider a single particle of mass $(m=1)$ at position $x(t)$ and velocity $v(t)$. Then its dynamics (trajectory) is given by;

$$x'(t) = v(t) \qquad \text{(definition of velocity)}$$
$$v'(t) = f \qquad \text{(Newton's second law)}$$

The force $F = F(t)$ needs to be specified. For simplicity, we can consider a given external force $f(t)$, the single-particle dynamics is completely specified by

$$\begin{aligned} x'(t) &= v(t) & x(0) &= x_0 \\ v'(t) &= f(t) & v(0) &= v_0 \end{aligned} \quad — (1.5)$$

Here, x_0, v_0 are the starting position and velocity of the particle.

Denote, $u = [x, v]$

and $F = [v, f]$

(1.5) can be rewritten as

$$u' = F(t, u) \quad — (1.6)$$

note that u is a vector and the force $f = f(t)$.

1.3.2 Many (N)- body dynamics.

The more general situation arises when we consider N bodies with masses $(m_1, ---, m_N)$

positions $(x_1, ---, x_N)$

and velocities $(v_1, ---, v_N)$

Then applying Newton's law of motion to this collection of particles (bodies), we obtain

$$\begin{aligned} x_i' &= v_i \\ m_i v_i' &= f_i \end{aligned} \quad \text{for the i-th particle.}$$

Suppose that the only force acting on the particles is gravity, then we have:

$$f_i = \sum_{j \neq i, j=1}^{N} \frac{G\, m_i m_j (x_i - x_j)}{|x_i - x_j|^3}.$$

Here, we assume that we are in three- space dimensions and use Newton's law of gravity.

denoting;

$$u = [x_1, \dots, x_N, v_1, \dots, v_N] \qquad \text{with} \quad f_i = \sum_{\substack{j=1 \\ i \neq j}}^{N} \frac{G m_j (x_i - x_j)}{|x_i - x_j|^3}$$

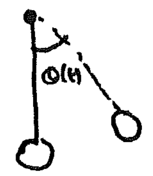
$$F = [v_1, \dots v_N, f_1, \dots, f_N]$$

we again obtain an ODE of form (1.6)

$$u'(t) = \cancel{F(t, u(t))} \; F(u(t))$$

1.3.3: A Simple pendulum.

Consider a pendulum of mass m at the end of a rigid, masgless bar of length L (see figure 1.1)



The motion of the pendulum is described in terms of the angle $\theta(t)$, which in turn obeys the angular version of Newton's law:

$$\theta''(t) = -(g/L)\, \sin(\theta(t))$$

Setting $g = L$, we obtain the pendulum equation.

$$\theta''(t) = -\sin(\theta(t)) \quad \text{—} \quad (1.7)$$

$$\theta(0) = \theta_0, \quad \theta'(0) = v_0$$

Ex: Check that for small angular deviations, $\theta(t) \ll 1$, (1.7) can be approximated by

$$\theta''(t) = -\theta(t).$$

Note that we have to specify starting angle $\theta(0)$ and starting angular velocity $\theta' = v(0)$.

1.3.4 : A population (Concentration) model from Cell Biology.

Proteins are functional units of living cells. They are manufactured by the sequential processes of transcription and translation.

In transcription, a gene is expressed in terms of mRNAs

In translation, mRNAs produce proteins.

Let the Concentration of a gene (X) be denoted by $X(t)$
Concentration of mRNA (Y) be denoted by $Y(t)$
Concentration of protein (Z) be denoted by $Z(t)$

then we have the following reaction Scheme,

$$\begin{tikzcd} X \arrow[r, "\text{transcription}"'] & Y \arrow[r, "\text{translation}"'] & Z \end{tikzcd}$$

In the Simplest model, the gene Concentration is given, the mRNA Concentration changes by:

$$Y' = F_1(X) - \alpha_1 Y$$

Here; F_1 = Production of the mRNA as a function of X

It is given by

$$F_1(X) = \frac{\beta_1 X^n}{X^n + k_1^n}$$

Here, β_1 - mRNA production rate
k_1 - mRNA production threshold.

$f_n(x) = \frac{X^n}{X^n + k_1^n}$ is termed as a n- Hill function :
usually : ($n = 4$ or $n = 8$)

An example is of a hill function is shown in figure (2).

Similar $\alpha_1 Y$ is a decay term,

The kinetics of the protein concentration is given by.

$$Z' = F_2(Y) - \alpha_2 Z$$

Here, $F_2 = \gamma\beta_2 \left(\frac{Y^n}{Y^n + k_2^n}\right)$

with β_2 - Protein (maximum) production rate.

$k_2 \rightarrow$ Protein activation threshold

$\gamma \rightarrow$ Transcription rate.

Let $u = [Y, Z]$

$$F = \left[\frac{\beta_1 X(t)^n}{X(t)^n + k_1^n} - \alpha_1 Y,\ \frac{\gamma\beta_2 Y^n}{Y^n + k_2^n} - \alpha_2 Z\right]$$

the resulting ODE is

$u' = F(u,t)$ (again of the form (1.6)).

1.3.5: Models of Chemical kinetics.

Example 1 →. we have the following set of chemical reactions.

(Decay)

$$A \xrightarrow{k_1} B \xrightarrow{k_2} C$$

Here A, B, C are chemicals with concentrations u_1, u_2, u_3

A decays to B at rate k_1

$\therefore \quad u_1' = -k_1 u_1$

B is produced from A at rate k_1 and decays to C at rate k_2

$\therefore \quad u_2' = k_1 u_1 - k_2 u_2$

C is produced from B at rate k_2

$\therefore \quad u_3' = k_2 u_2$

Hence, $u = [u_1, u_2, u_3]$ has the dynamics;

$$u' = F(u) = Au \qquad \text{— (1.8)}$$

with

$$A = \begin{bmatrix} -k_1 & 0 & 0 \\ k_1 & -k_2 & 0 \\ 0 & k_2 & 0 \end{bmatrix}$$

Example 2 : Compound formation.

Consider the reaction scheme.

$$A + B \underset{k_2}{\overset{k_1}{\rightleftharpoons}} AB$$

Consider A, B, AB have concentrations u_1, u_2, u_3 respectively

their kinetics is given by

$$u_1' = -k_1 u_1 u_2 + k_2 u_3$$
$$u_2' = -k_1 u_1 u_2 + k_2 u_3$$
$$u_3' = k_1 u_1 u_2 - k_2 u_3.$$

Here, $u = [u_1, u_2, u_3]$ obeys the ODE

$$u' = F(u) \qquad \text{— (1.9)}$$

$F(u) = [k_2 u_3 - k_1 u_1 u_2,\ k_2 u_3 - k_1 u_1 u_2,\ k_1 u_1 u_2 - k_2 u_3].$

1.4: Generic form of ODEs (IVP)

The above examples, chosen from different fields, suggested that the generic form,

$$u'(t) = F(t, u(t)) \qquad \text{--- (1.10)}$$
$$u(0) = u_0$$

represented nearly all of them. This is not strictly true for the pendulum example. Recall, the motion of a pendulum was given by.

$$\theta''(t) = -\sin(\theta(t)) \qquad \text{--- (1.11).}$$
$$\theta(0) = \theta_0, \quad \theta'(0) = v_0$$

it turns out that a simple rewriting of (1.11) can cast it in the form. (1.10). To this end,

denote $v(t) = \theta'(t)$ (angular velocity), then we have:

$$\theta' = v \qquad \theta(0) = \theta_0 \qquad \text{--- (1.12)}$$
$$v' = -\sin(\theta) \qquad v(0) = v_0$$

Thus writing the ~~system~~ vector,

$$u = [\theta, v]$$
$$F = [v, -\sin(\theta)]$$

(1.12) reduces to the form.

$$u' = F(u) \qquad \text{--- (1.13)}$$
$$u(0) = u_0$$

In general, any IVP for ODEs can be recast in the first-order form (1.10).

Consider another example;

$$v''''(t) = 3\, v'''(t)\, v'(t) + 2\, v''(t)^3 - \sin(t). \quad — (1.14)$$

Introduce auxiliary variables,

$$v(t) = u_1(t) \qquad v''(t) = u_3(t) \qquad \cancel{v''''(t) = u_5(t)}$$
$$v'(t) = u_2(t) \qquad v'''(t) = u_4(t)$$

The above ODE can be rewritten as:

$$u_1' = u_2$$
$$u_2' = u_3$$
$$u_3' = u_4$$
$$u_4' = 3u_4u_2 - 2u_3^3 - \sin(t)$$

$$\therefore \quad u = [u_1, u_2, u_3, u_4], \quad F(t,u) = [u_2, u_3, u_4, 3u_2u_4 - 2u_3^3 - \sin(t)]$$

Thus u obeys the system

$$u'(t) = F(t, u(t)) \; !!!$$

1.5: Types of ODEs

Autonomous ODEs - The ODE system (1.10) is termed as autonomous if $F(t, u(t)) \equiv F(u)$ i.e. the right hand side does not explicitly depend on time, Thus (1.10) is

~~Examples~~ - $$u'(t) = F(u) \quad — (1.13)$$

Examples - N-body dynamics, Pendulum equation, Chemical kinetics.

On the other hand, the transcription-translation ODEs and the model ODE (1.14) have an explicitly time-depended rhs. Hence, they are non-autonomous.

Yet there is a simple trick by which an non-autonomous ODE can be made autononomusou

The most-general form of a non-autonomous ODE is (1.10) with a vector $u \in \mathbb{R}^m$. $u = [u_1, u_2, \dots, u_m]$

<u>Define</u> $W = [u_1, u_2, \dots u_m, u_{m+1}]$

$$G(W) = [F_1(u_1, \dots, u_m, u_{m+1}), F_2(u_1, \dots, u_m, u_{m+1}), \dots, F_m(u_1, \dots, u_{m+1}), 1]$$

then the ODE systen

$$\omega' = G(\omega) \qquad \text{— (1.15)}$$
$$\omega(0) = [u_1(0), \dots, u_m(0), 0]$$

is the Some as the ~~non-au~~ non-autonomous system (1.10)

To check this, we see that: (from (1.15))

$$u_{m+1}' = 1 \quad \Rightarrow \quad u_{m+1}(t) = t.$$
$$u_{m+1}(0) = 0$$

Thus $\omega = [u_1, \dots, u_m, t]$ and (1.10) is recovered □.

<u>Scalar vs. Systems of ODEs</u>

Some examples of ODEs are Scalar i.e. $u = u_1$ in (1.10), (1.13). Examples are the pendulum equation, ~~t-body dynamics~~

However, many ODEs are systems of ODEs ~~such~~ i.e. u is a vector, F is a vector.

Examples are N-body dynanics, ~~DNA trans~~ Protein manufacture, Chenical kinetics.

Note that even pendulum equation, written as (1.12) is a System.

Linear vs Nonlinear ODEs:

An autonomous ODE system is termed linear, if any linear combination of solutions is also a solution. ~~i.e~~. This is ensured if the rhs of (1.13) is given by:

$$F(u) = A(t)\,u \quad \text{here} \quad u \in \mathbb{R}^m$$
$$\text{and} \quad A \in \mathbb{R}^{m\times m} \text{ (m×m)- Matrix}$$

Examples → . Chemical kinetics model of decay.

but most interesting ODEs are nonlinear such as N-body dynamics, Protein manufacture, Pendulum equation and many more.

1.6: Explicit Solutions of ODEs.

We can find explicit solutions for many ~~of~~ ODEs.

The simplest example is provided by the linear scalar ODE

$$u' = \lambda u \quad — (1.16)$$
$$u(0) = u_0$$

with $u \in \mathbb{R}$, $u_0 \in \mathbb{R}$ and $\lambda \in \mathbb{R}$ (constant).

The solution in this case is

$$\boxed{u(t) = u_0 e^{\lambda t}}$$

1.6.1: Solving a linear system explicitly:

Consider the (constant coefficient) linear ODE system,

$$u' = Au \qquad u \in \mathbb{R}^m, \quad A \in \mathbb{R}^{m\times m}$$
$$u(0) = u_0 \quad — (1.17) \quad — (1.17)$$

Further. assume that the matrix A is diagonalizable i.e, $\exists$ a set of eigenvalues $(\lambda_1, \dots, \lambda_m)$ and eigenvectors $(r_1, \dots, r_m)$ such that $\{r_i\}_{i=1}^m$ form a complete set

$$A = R \Lambda R^{-1}$$

Here; $\Lambda = \begin{pmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & \cdots & \cdots & \lambda_m \end{pmatrix} = \operatorname{diag}(\lambda_1, \dots, \lambda_m)$

and $R = \left[r_1 \,\middle|\, r_2 \,\middle|\, \cdots \,\middle|\, r_m \right]$ Eigenvectors r_i form i-th column of R.

Then multiply R^{-1} to both sides of (1.17),

$$(R^{-1}u)' = R^{-1}(R\Lambda R^{-1})u.$$
$$= \Lambda (R^{-1}u)$$

Let $W = R^{-1}u$, then,

$$w' = \Lambda w \quad \text{—} \quad (1.18)$$
$$w(0) = R^{-1}u_0$$

Note that (1.18) is an uncoplpled set of m-scalar equations

$$w_i' = \lambda_i w_i, \quad w_i(0) = R^{-1}u_0|_i.$$

Thus;

$$w_i(t) = w_i(0)\, e^{\lambda_i t}$$

$$w(t) = \text{~~W(0)~~} \begin{pmatrix} e^{\lambda_1 t} & & 0 \\ & e^{\lambda_2 t} & \\ 0 & & e^{\lambda_m t} \end{pmatrix} \begin{pmatrix} w_1(0) \\ \vdots \\ w_m(0) \end{pmatrix}$$

Denote $e^{\Lambda t} = \text{diag}\left(e^{\lambda_1 t}, \dots, e^{\lambda_n t}\right)$

$$\begin{aligned} \omega(t) &= e^{\Lambda t}\omega(0) \\ &= e^{\Lambda t}R^{-1}u_0 \end{aligned}$$

hence,
$$\begin{aligned} Ru(t) &= e^{\Lambda t}R^{-1}u_0 \\ u(t) &= Re^{\Lambda t}R^{-1}u_0. \\ &= e^{At}u_0 \end{aligned}$$

Here, we need the notion of a matrix exponential,

$$e^{At} = I + At + \frac{1}{2}A^2t^2 + \dots + \frac{1}{k!}A^k t^k + \dots$$

to make sense.

Ex: if A is diagonaliable, then:

$$e^{At} = Re^{\Lambda t}R^{-1}.$$

Some non-diagonalizable matrices can also be considered in (1.17) (Such as matrices written in Jordan-Canonical form).

1.6.2: Duhamel's principle:

Consider the scalar, linear, non-autonomous ODE;

$$u' = \lambda u + g(t) \quad \text{—} \quad (1.19)$$
$$u(0) = u_0$$

Define: $v(t) = ue^{-\lambda t}$ $\quad v(0) = u_0$

then $v' = u'e^{-\lambda t} - \lambda ue^{-\lambda t} \underset{(1.19)}{=} (u' - \lambda u)e^{-\lambda t}$

$= g(t)e^{-\lambda t}$

$$\therefore \quad v(t) = u_0 + \int_0^t g(s)e^{-\lambda s}\,ds$$

Hence
$$u(t)e^{-\lambda t} = u_0 + \int_0^t g(s)e^{-\lambda s}\,ds$$
$$\therefore \quad u(t) = u_0 e^{\lambda t} + \int_0^t g(s)e^{\lambda(t-s)}\,ds.$$

For the linear non-autonomous system:
$$u'(t) = Au + g(t) \qquad u \in \mathbb{R}^m.$$
$$u(0) = u_0 \qquad A \in \mathbb{R}^{m\times m}. \quad \text{— } (1.20)$$

Assume that A is diagonalizable,

Ex: Show that
$$u(t) = e^{At}u_0 + \int_0^t e^{A(t-s)}g(s)\,ds$$
is a solution of (1.20).

- Apart from linear ODEs, explicit solutions can only be found for very few nonlinear ODEs. We need to use numerical methods to approximate them.

1.7 Theory on ODEs:

Consider the ODE system;
$$u' = F(u(t),t) \qquad u \in \mathbb{R}^m,\ F \in \mathbb{R}^m.$$
$$u(0) = u_0 \qquad u_0 \in \mathbb{R}^m \quad \text{— } (1.20)$$

Assume that there exist $T^* > 0$ and $a > 0$, such that

on $D = \{(t,u) : 0 \le t \le T^* \text{ (and) } \|u - u_0\| < a\}$

F is Lipshitz in u for all $t \in [0,T^*]$ i.e.

there exists a $L>0$ such that for all $t \in [0,T]$

$$|F(u,t) - F(u_0,t)| < L|u-u_0| \quad \text{— } (1.21)$$

then by the Cauchy-Lipschitz theorem, there exists a unique solution of the ODE (1.20) in $\mathcal{O}$.

<u>Examples</u>:

1. Consider the linear ODE

$$u' = Au$$

<u>Clearly</u> $|F(u) - F(u_*)| = |Au - Au_*| \le \|A\| \, |u-u_*|$

where $\|A\|$ is a matrix norm and is bounded.

2. Consider the pendulum equation;

$$F(\theta, v) = (v, \sin(\theta))$$

$$\|F(\theta,v) - F(\theta_*, v_*)\|_1 = \|(v - v_*, \sin(\theta) - \sin(\theta_*))\|_1$$

$$= |v - v_*| + |\sin(\theta) - \sin(\theta_*)|$$

$$\le |v - v_*| + |\cos(\bar\theta)| \, |\theta - \theta^*|$$

$$\le |v - v_*| + |\theta - \theta_*| = \|u - u^*\|_1$$

The Lipshitz constant is 1 !!!

3. Consider the scalar non-linear ODE,

$$u' = u^2$$
$$u(0) = u_0$$

formal solution; $u(t) = \dfrac{u_0}{1 - u_0 t}$

Clearly if $u_0 > 0$, ~~it is~~ $F(u)$ is not "globally" Lipshitz continuous.

$$\lim_{t \to \frac{1}{u_0}} u(t) = +\infty$$

The solution blows up. So, the solution only exists for

$$0 \le t \le \frac{1}{u_0}$$

4. Consider the scalar non-linear ODE

$$u' = \sqrt{u}$$
$$u(0) = u_0$$

Clearly $F(u)$ is not Lipshitz Continuous near $u = 0$.

In fact, we can find 2 Solutions in this case.

$$u \equiv 0$$
$$u = \frac{1}{4} t^2$$

Thus uniqueness also requires Lipshitz Continuity.

2. Numerical methods for ODEs

In this chapter, we will describe different methods to numerically approximate solutions of the IVP for ODEs,

$$u'(t) = F(u(t), t)$$
$$u(0) = a_0 \qquad \text{— } \boxed{2.1}$$

Here, the unknown u could be a vector or a scalar, depending on the model.

As claimed in the last chapter, it is not possible to find solution formulas for $\boxed{2.1}$ except in very few cases. So, we need numerical methods to approximate $\boxed{2.1}$.

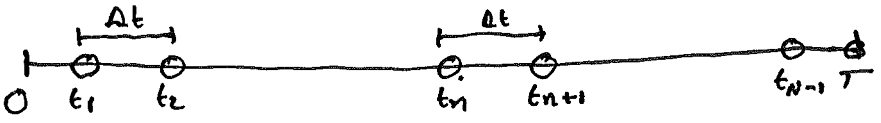
2.1: Time Mesh:

We consider (2.1) in the time interval $[0, T]$. We discretize it by introducing a time step $\Delta t > 0$, such that

$$[0,T] = \bigcup_{n=0}^{N-1} [t_n, t_{n+1}] \qquad \text{and} \quad N = \frac{T}{\Delta t}$$

so with $t_n = n\,\Delta t$, hence $t_0 = 0$, $t_N = T$

Thus, the time period $[0,T]$ is divided into N equally spaced intervals. See figure (2.1)



Note that the points $\{t_n\}$ are termed time levels.

Our aim is to approximate the exact Solution u of (2.1) at these time levels. Hence, we denote:

$$U_n \approx u(t^n) \quad — \quad (2.2)$$

In the other words, numerically solving the ODE (2.1) amounts to finding $\{U_n\}_{n=0}^{N}$.

Clearly, $U_0 \approx u(t_0) = u(0) = u_0$

So we can Set $U_0 = u_0 \quad — \quad (2.3)$

Rem: for simplicity, we have assumed that the time discretization is equally spaced i.e $t_{n+1} - t_n = \Delta t \quad \forall n \in (1, \ldots, n)$. later in the chapter, we consider a Situation where it is not so.

2.2: Forward Euler method.

We need to approximate the ODE (2.1) in order to obtain U_n. Hence, we have to discretize the derivative u_t and the rhs $(f(u,t))$ of (2.1).

The Simplest discretizations of a derivative replace it with a finite difference. ~~The~~ In the forward Euler method, we use a forward difference i.e.

$$u_t(t^n) \simeq \frac{u(t^{n+1}) - u(t^n)}{\Delta t} \approx \frac{U_{n+1} - U_n}{\Delta t}$$

Similarly, ther rhs is evaluated at t^n

$$f(u(t_n), t_n) \simeq f(U_n, t_n)$$

Hence, the Forward Euler method is

$$\frac{U_{n+1} - U_n}{\Delta t} = f(U_n, t^n) \quad — \quad (2.4)$$

$$U_0 = u_0$$

It can be rewritten as:

$$U_{n+1} = U_n + \Delta t\, f(U_n, t^n) \quad — (2.5)$$
$$U_0 = u_0$$

Note that the form (2.5) is readily ~~implic.~~ implemented in a time marching scheme

$$U_0 \longmapsto U_1 \longmapsto U_2 \cdots \longmapsto U_n \longmapsto U_{n+1} \cdots \longmapsto U_N$$

with given U_N, find U_{n+1} by (2.5) using a simple function evaluation.

Therefore, the forward Euler method is termed as an ~~impl.~~ explicit method.

2.3: Backward Euler method.

Another possible discretization of the derivative u_t is to take a backword difference; i.e.

$$u_t(t^{n+1}) \simeq \frac{u(t^{n+1}) - u(t^n)}{\Delta t} \approx \frac{U_{n+1} - U_n}{\Delta t}$$

with rhs being evaluated at t^{n+1} i.e,

$$f(u(t^{n+1}), t^{n+1}) \simeq f(U_{n+1}, t^{n+1})$$

∴, the ODE (2.5) is approximated by;

$$\frac{U_{n+1} - U_n}{\Delta t} = f(U_{n+1}, t^{n+1}) \quad — (2.6)$$
$$U_0 = u_0$$

This method is termed as the backword Euler method.

It can also be rewritten as.

$$U_{n+1} - \Delta t\, f(U_{n+1}, t^{n+1}) = U_n \quad — (2.7)$$

(2.7) can also be implemented as a time marching scheme

+ Given U_n, we need to find U^{n+1} from (2.7)

Note that (2.7) is a non-linear system of equations that needs to be solved numerically using a Newton method. Hence, Backward Euler is an implicit method.

2.4 Trapezoidal method:

Consider the following calculation;

$$U(t_{n+1}) - u(t_n) = \int_{t^n}^{t^{n+1}} u'(s)\, ds. \quad \left(\begin{array}{r}\text{Fundamental} \\ \text{theorem of} \\ \text{calculus}\end{array}\right)$$

$$= \cancel{\int_{t^n}^{t^{n+1}} f(u(s))\, ds \quad (ODE}.$$

$$= \int_{t^n}^{t^{n+1}} f(u(s), s)\, ds \quad (ODE\ (2.1)).$$

now, the integral above may be very difficult to evaluate exactly. However, we can approximate it using a quadrature. One possibility is to use trapezoidal rule, then

$$\int_{t^n}^{t^{n+1}} f(u(s), s)\, ds \approx \frac{\Delta t}{2}\left(f(u(t^n), t^n) + f(u(t^{n+1}), t^{n+1})\right)$$

using the approximation; $U^n \approx u(t_n)$, we obtain the trapezoidal rule:

$$\frac{U_{n+1} - U_n}{\Delta t} = \frac{1}{2}\left(f(U_n, t^n) + f(U_{n+1}, t^{n+1})\right) \quad \text{— (2.8)}$$

Note that (2.8) can be rewritten as

$$U_{n+1} = U_n + \frac{\Delta t}{2} f(U_n, t^n) + \frac{\Delta t}{2} f(U_{n+1}, t^{n+1}) \quad \text{— (2.9)}$$

Check that (2.9) is just the average of the forward Euler method (2.5) and Backward Euler method (2.7)

Furthermore, check that trapezoidal rule (2.9) is an implicit method.

2.5 Mid-point rule:

In the forward and backward Euler methods, we used either the forward or the backward difference. Using the central difference ~~of~~ to approximate u_t results in;

$$u_t(t^n) \approx \frac{u(t^{n+1}) - u(t^{n-1})}{2\Delta t} \approx \frac{U_{n+1} - U_{n-1}}{2\Delta t}$$

$$\cancel{f(u(t^n} \; f(u(t^n), t^n) \approx f(U_n, t_n)$$

The resulting method is.

$$\frac{U_{n+1} - U_{n-1}}{2\Delta t} = f(U_n, t^n) \qquad \text{—} \qquad (2.10)$$

$$U_0 = u_0$$

~~to~~ It can be rewritten as:

$$U_{n+1} = U_{n-1} + 2\Delta t \, f(U_n, t^n) \qquad \text{—} \qquad (2.11)$$

The midpoint rule can be used as a time marching method as other methods. Given the values U_{n-1}, U_n; One can use (2.11) to obtain U_{n+1}.

Therefore, we need ~~U~~ U_0 and U_1 to march forward in time.. by obtaining U_2.

We can use the forward-Euler method to obtain U_1 i.e.

$$U_1 = U_0 + \Delta t \, f(U_0, 0)$$

Check that the midpoint rule is an explicit method.

Numerical examples:

See Slides.

2.6

Truncation Error: How do we distinguish between the different methods proposed so far? How to judge which method is Superior? How to explain the Observed numerical results?

We take the first steps in analysing numerical methods by ~~cdb~~ utilizing the truncation error.

Note that the schemes (2.4, 2.6, 2.8, 2.10) are consistent with the form of the ODE (2.1).

To calculate the truncation error, we insert the true Solution $u(t)$ of (2.1) in the Consistent forms (2.4, 2.6, 2.8, 2.10) etc

Example: The truncation error of the forward Euler method (2.4) is given by

$$T_n := \frac{u(t^{n+1}) - u(t^n)}{\Delta t} - f(u(t_n), t_n) \quad — \quad (2.12).$$

In other words, truncation error is the error made when the exact solution is inserted into the ODE. As an example, we compute the truncation error ~~for~~ (2.12) by using a Taylor expansion.

$$t_n = u'(t_n) + \frac{\Delta t}{2} u''(t_n) + O(\Delta t^2) - f(u(t_n), t_n)$$

$$= \underbrace{(u'(t_n) - f(u(t_n), t_n))}_{=0 \text{ (by (2.1)}} + \frac{\Delta t}{2} u''(t_n) + O(\Delta t^2)$$

$$\therefore \quad T_n = \frac{\Delta t}{2} u''(t_n) + O(\Delta t^2) \approx O(\Delta t).$$

As a second example, consider the truncation error associated with the midpoint rule (2.10),

$$T_n := \frac{U^{n+1} - U^{n-1}}{2\Delta t} - f(u(t_n), t_n)$$

using a Taylor expansion,

$$T_n = \underbrace{\left(u'(t_n) - f(u(t_n), t^n)\right)}_{=0 \text{ (by (2.1))}} + \frac{\Delta t^2}{3} u''(t_n) + O(\Delta t^3)$$

$$\therefore \quad T_n = \frac{u''(t_n)}{3} \Delta t^2 + O(\Delta t^3) \simeq O(\Delta t^2)$$

Ex: Check that truncation error of backward Euler method (2.6) is $O(\Delta t)$

Ex: Check that truncation error of trapezoidal rule (2.8) is $O(\Delta t^2)$

~~Defn~~ ~~Thus,~~

2.7 One-step error:

A related concept to truncation error is one-step error.

We see that the forms (2.5) (2.7) (2.9) (2.11) are in the so-called update form.

~~$U_{n+1} = 2t(U_n, U_{n+1}, t^n, t^n$~~

The one-step error is obtained by substituting the exact solution ~~to~~. u of (2.1) in this update form.

We again consider the update form of the forward Euler method (2.5). The one-step error is defined as:

$$L^n := u(t^{n+1}) - u(t^n) - \Delta t\, f(u(t^n), t^n) \quad - \quad (2.13)$$

we calculate L^n using Taylor expansion as,

$$L^n = u(t_n) + \Delta t\, u'(t_n) + \frac{\Delta t^2}{2} u''(t_n) + \frac{\Delta t^3}{6} u'''(t_n) + \mathcal{O}(\Delta t^4)$$
$$- u(t^n) - \Delta t\, f(u(t^n), t^n)$$
$$= \underbrace{\Delta t\left(u'(t_n) - f(u(t^n), t^n)\right)}_{=0 \quad (2.1)} + \frac{u''(t_n)}{2}\Delta t^2 + \mathcal{O}(\Delta t^3)$$

Hence
$$L^n = \frac{u''(t_n)}{2}\Delta t^2 + \mathcal{O}(\Delta t^3) \quad - \quad (2.14)$$
$$= \mathcal{O}(\Delta t^2)$$

Clearly, in this case.
$$L^n = \Delta t\, T^n$$

Ex: Check that the one-step errors are

Backward Euler	—	$\mathcal{O}(\Delta t^2)$
Midpoint rule	—	$\mathcal{O}(\Delta t^3)$
Trapezoidal rule	—	$\mathcal{O}(\Delta t^3)$

Now, ~~let consider the error at~~ :

Now assume that we know the solution of (2.1) at time t^n exactly. Then the forward Euler method (2.5) is

$$U_{n+1} = u(t_n) + \Delta t\, f(u(t_n), t^n)$$

The exact solution at t^{n+1} is $u(t^{n+1})$. Thus, the error in this single step is:

$$u(t^{n+1}) - U^{n+1} := u(t_{n+1}) - u(t_n) - \Delta t\, f(u(t^n), t^n)$$
$$= L^n \quad (\text{from } (2.13))$$

Hence; L^n is exactly the error at a single step of the method, justifying its nomenclature as One-step error !!!

2.8 Global error.

Given the exact solution $u(t_n)$ at time t^n of ODE (2.1) and U_n is the approximate solution computed using a numerical method, we define the error as:

$$E_n := u(t_n) - U_n$$

Note that we commit a local error L_j at each step of the time marching scheme.

Roughly speaking;

$$E_n \approx \sum_{j=1}^{N} L_j \rightarrow |E_n| \leq \sum_{j=1}^{N} |L_j| \quad ~~\leq O(\Delta t~~$$

Now if we assume that ~~each~~ the one-step error associated with a scheme is: $L_j \simeq O(\Delta t^{q+1})$ for $q \geq 1$, then.

we obtain,
$$E_n \approx O(\Delta t^{q+1})\, N \approx O(\Delta t^{q+1}) \frac{T}{\Delta t}$$
$$\approx O(\Delta t^{q})$$

Thus, if exactly the same amount of error is accumulated at each step (there is no amplification of error), then the global error scales like the truncation error.

This may explain why the midpoint rule and Trapezoidal rule are more accurate than forward (backward) Euler.

However, it is unclear if in a nonlinear ODE like (2.1), there is no amplification of error – this is related to stability of a method and will be discussed later.

2.9 Taylor Expansion methods.

Given the above discussion, one way to improve the order of accuracy of a method is to use Taylor expansions to reduce truncation errors.

We demonstrate this approach in the case of a scalar, autonomous ODE

$$u' = f(u) \qquad \text{— } (2.15)$$
$$u(0) = u_0$$

using a taylor expansion around $u(t_n)$.

$$u(t_{n+1}) = u(t_n) + \Delta t\, u'(t_n) + \frac{\Delta t^2}{2} u''(t_n) + \frac{\Delta t^3}{6} u'''(t_n) + \frac{\Delta t^4}{24} u^{iv}(t_n) + O(\Delta t^5)$$

we know from (2.1) that

$$u'(t_n) = \cancel{f(u_n)}.\ f(u(t_n))$$

differentiating: $u''(t_n) = f'(u(t_n))\, f(u(t_n))$

Similarly $u'''(t_n) = f(u(t_n))^2 f''(u(t_n)) + f'(u(t_n))^2$

using $u(t_n) \approx u_n$,

we can obtain the following method.

$$u_{n+1} = u_n + \Delta t\, f(u_n) + \frac{\Delta t^2}{2} f(u_n)\, f'(u_n) + \frac{\Delta t^3}{6} (f'(u_n))^2 + \frac{\Delta t^3}{6}\left(f(u_n)^2 f''(u_n)\right) \qquad \text{— } (2.16)$$

Check that the ℓ^n, the one-step error, associated with (2.16) is

$$\ell^n \approx O(\Delta t^4)$$

and $$T^n \approx O(\Delta t^3)$$

Thus, (2.16) may be a third-order accurate method. Higher order accurate methods can be similarly obtained. However, using a Taylor expansion and particularly the substitutions associated with it (as above) is very difficult for ~~Autonom~~ non-autonomous ODEs and for systems of ODEs.

Thus, we use different approaches to obtain higher-order accurate ~~time~~ schemes and consider them in the next chapter.