Computational Methods for engeneering applications (CMEA)

Michael van Huffel, Sofia van Huffel

Version: March 3, 2022

This summary has been written based on the Lecture of CMEA (401-0435-00 S) by Prof. Dr. R. Käppeli and Mr. M. Petrella (Autumn 21) and the summary of Mario Millhausler. There is no guarantee for completeness and/or correctness regarding the content of this summary. Use it at your own discretion

0 Contents

1 Introduction

1.1 Definitions:

- **Autonomous**: $F(u(t))$ is not an *expl.* func. of time.
- **Non-Autonomous:** $F(t, u(t))$
- **Linear:** $F(t, u(t)) = A(t) \cdot u(t) + C(t)$, hom.: $C \equiv 0$
- **Non-Linear:** e.g. $u'(t) = \cos(u(t))$
- **Scalar vs. Systems of ODEs**

1.1.1 Converting the ODE:

Consider a non-autonomous IVP: $\begin{cases} u = [u_1, u_2, \cdots, u_m] \\ F = [F_1, F_2, \cdots F_m] \end{cases}$

Let the function u_{m+1} be $u_{m+1}(t) = t$. Then the IVP can be rewritten as the following autonomous IVP:

$\begin{cases} W'(t) = G(W(t)) \\ W(0) = [u_1(0), u_2(0), \cdots, u_m(0), 0] \end{cases}$, where

$\begin{cases} W = [u_1, u_2, \ldots, u_m, u_{m+1}] \\ G(W) = [F_1(u_1, \ldots, u_{m+1}), \ldots, F_m(u_1, \ldots, u_{m+1}), 1] \end{cases}$

Example: Conversion to autonomous IVP

Given $u'(t) = u(t)^2 + t \rightarrow$ we substitute $u_2(t) = t$:

$$\begin{cases} u_1'(t) = u_1(t)^2 + u_2(t) & u_1(0) = u_{1,0} \\ u_2'(t) = 1, & u_2(0) = 0 \end{cases}$$

1.2 Well-Posedness of IVP for ODEs:

Theorem: Cauchy-Lipschitz

Let $U \subseteq \mathbb{R}^m$ be an open set and consider the following non-autonomous IVP:

$$\begin{cases} u'(t) = F(t, u(t)) \\ u(0) = u_0 \end{cases}$$

Where $F : [0, T] \times \mathbb{R}^m \rightarrow \mathbb{R}^m$ and $u : [0, T] \rightarrow \mathbb{R}^m$ are continuous functions in t.
Suppose that the function F is Lipschitz continuous on the set $[0, T] \times U$, i.e., there exists $L > 0$ such that for all $(t, u), (t, u^*) \in [0, T] \times U$ it holds that:

$$\|F(t, u) - F(t, u^*)\| \leq L\|u - u^*\|$$

where $\|\cdot\|$ is any vector norm. Then there exists $T^* \in (0, T)$ so that the IVP has a unique solution in time interval $[0, T^*]$

2 Numerical Methods for ODEs

2.1 Time Discretization:

Find $\{u^n\}_{n=0}^N$ s.t. $u^n \approx u(t^n) \quad \forall n \in \{0, N\}$

$$\Delta t := \frac{T}{N} \qquad \text{time step}$$
$$t^n := n \cdot \Delta t \qquad \text{time level}$$

2.2 Forward Euler Method / Explicit (FE):

The forward Euler Method is one way to obtain the approximate solutions at each time level: $u'(t^n) \approx \frac{u(t^{n+1}) - u(t^n)}{\Delta t} \approx \frac{U_{n+1} - U_n}{\Delta t}$ we get so:

$$U_{n+1} = U_n + \Delta t \cdot F(t^n, U_n), \text{ with } U_0 = u_0$$

⊕ easy, cheap (computation wise)
⊖ limited stability

0	0
	1

2.3 Backward Euler Method/ Implicit (BE):

Using implicit Euler scheme: $u'(t^{n+1}) \approx \frac{u(t^{n+1}) - u(t^n)}{\Delta t} \approx \frac{U_{n+1} - U_n}{\Delta t}$ and so:

$$U_{n+1} = U_n + \Delta t \cdot F(t^{n+1}, U_{n+1}), \text{ with } U_0 = u_0$$

⊕ stable, cheap (if invertible)
⊖ expensive (if !invertible) → Newton's M.

1	1
	1

2.4 Trapezoidal Method/ Implicit:

With the Fundamental theorem of Calculus, we see that:

$$u(t^{n+1}) - u(t^n) = \int_{t^n}^{t^{n+1}} u'(s)ds$$

We approximate this Integral with the Trapezoidal Rule: $\int_a^b f(x)dx \approx \frac{b-a}{2}(f(a) + f(b))$ and we get the implicit scheme:

$$U_{n+1} = U_n + \frac{\Delta t}{2}(F(t^n, U_n) + F(t^{n+1}, U_{n+1}), \text{ with } U_0 = u_0$$

⊕ more accurate, stable
⊖ expensive (if !invertible) → Newton's M.

0	0	0
1	1/2	1/2
	1/2	1/2

2.5 Mid-Point Rule/ Explicit:

We have: $u'(t^n) \approx \frac{u(t^{n+1}) - u(t^{n-1})}{2\Delta t} \approx \frac{U_{n+1} - U_{n-1}}{2\Delta t}$, and so we get ($U_0 = u_0$):

$$U_{n+1} = U_{n-1} + 2\Delta t F(t^n, U_n)$$
$$U_{n+\frac{1}{2}} = U_{n-\frac{1}{2}} + \Delta t F(t^n, U_n)$$

⊕ more accurate
⊖ memory intensive, "jumpstart" (need for two IV)

Example: Comparing the various Numerical Methods

Given an IVP how good is the approximation with:

1. $N = 5$ points: (MP, TR) good, (FE, BE) bad
2. $N = 50$ points: (MP, TR) fast perfect, (FE, BE) good
3. $N = 500$ points: all methods fast perfect

2.6 Newton's Method:

Needed for implementation of *implicit* methods. $G'_{i,j}(\boldsymbol{z}) = \frac{\partial G(\boldsymbol{z})_i}{\partial \boldsymbol{z}_j}$

Data: $G(z)$, "reasonable" z_0 s.t. $(G(z_0) \approx 0)$
Result: Find z s.t. $G(z) = 0$
initialization;
while $(|G(z_k)| > \varepsilon)$ **do**
  $z_{k+1} = z_k - \frac{G(z_k)}{G'(z_k)}$;
end
return z_k

Note: this method is not guaranteed to converge, implement maximal number of iterations or equivalent stopping method.

Example: Newtons Method with BE

Given $u' = F(t, u) := \sin(u) \Rightarrow u^{n+1} = u^n + \Delta t F(t^{n+1}, u^{n+1})$

$$u^{n+1} = u^n + \Delta t \cdot \sin(u^{n+1})$$
$$z = u^n + \Delta t \cdot \sin(z)$$
$$G(z) = z - u^n - \Delta t \cdot \sin(z) = 0$$

2.7 Truncation Error (local error):

The truncation error for a numerical scheme is the error made when the exact solution is inserted in the *consistent form* of numerical methods. It can be calculated by using the Taylor expansion:

$$u(t^{n+1}) = u(t^n) + \Delta t u'(t^n) + \frac{(\Delta t)^2}{2}u''(t^n) + \frac{(\Delta t)^3}{6}u'''(t^n)\dots$$
$$-u(t^{n-1}) = -u(t^n) + \Delta t u'(t^n) - \frac{(\Delta t)^2}{2}u''(t^n) + \frac{(\Delta t)^3}{6}u'''(t^n)\dots$$

Note: Expansion of $f(x)$ about a: $f(x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!}(x-a)^n$

Example: Truncation error of FE

$T_n = \frac{u(t^{n+1})-u(t^n)}{\Delta t} - F(t^n, u(t^n)) =$
$\frac{u(t^n)-u(t^n)+\Delta t u'(t^n)+\frac{(\Delta t)^2}{2}u''(t^n)+\mathcal{O}((\Delta t)^3)}{\Delta t} - F(t^n, u(t^n)) =$
$\underbrace{u'(t^n) - F(t^n, u(t^n))}_{=0 using IVP} + \frac{\Delta t}{2}u''(t^n) + \mathcal{O}((\Delta t)^2) = \mathcal{O}(\Delta t)$

The truncation error of the Forward Euler Method is first order accurate.

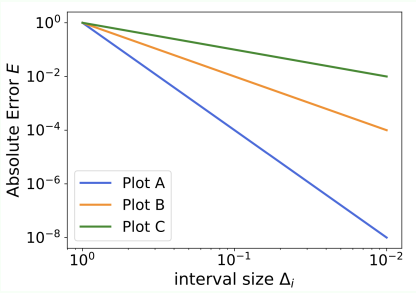
2.8 Empirical Error:

1. Choose $F(u)$ s.t. $F(u) = u'$ with $F(u)$ known.
2. Choose a wide array of values for N.
3. Compute Empirical Error: $\varepsilon_j = |U_{N_j} - u(T)|$

Note: $\varepsilon_j = \mathcal{O}(N_j^{-\alpha}) \Rightarrow |\varepsilon_j| \leq C \cdot N_j^{-\alpha} \Rightarrow \frac{\varepsilon_{j+1}}{\varepsilon_j} = \left(\frac{N_{j+1}}{N_j}\right)^{-\alpha}$:

$$\alpha \approx -\frac{\log\left(\frac{\varepsilon_{j+1}}{\varepsilon_j}\right)}{\log\left(\frac{N_{j+1}}{N_j}\right)} \approx \frac{\log\left|\frac{\varepsilon_{j+2}}{\varepsilon_{j+1}}\right|}{\log\left|\frac{\varepsilon_{j+1}}{\varepsilon_j}\right|}$$

Example: Error of Si (A), Ti (B), Ri (C)



How many function evaluation to get a reduction of 1000 of the error? *Trapezoidal:* $N' = \sqrt{1000} \cdot N$ more evaluation. *Simpson:* $N' = \sqrt[4]{1000} \cdot N$ more evaluation

2.9 One-Step Error (local error):

The One-Step Error for a numerical scheme is the error made when the exact solution is inserted in the *update form* of the numerical scheme: $\mathcal{O}_{OneStep}((\Delta t)^{k+1}) = \mathcal{O}_{Trunc}((\Delta t)^k)$

Example: One Step Error of autonomous RM

1. $L_n = u(t^{n+1}) - U^{n+1}$
2. $U^{n+1} = U^n + \frac{\Delta t}{4}F(U^n) + \frac{3\Delta t}{4}F(U^n + \frac{2\Delta t}{3}F(U^n))$
3. $F(U^n) = F(u(t^n)) = u'(t^n)$, $F(U^n + \frac{2\Delta t}{3}F(U^n)) = u'(t^n) + \frac{2\Delta t}{3}u''(t^n)$
4. Use Taylor to approximate $u(t^{n+1}) = u(t^n + \Delta t)$

2.10 Global Error:

The global error is defined as $E_N := u(t^N) - U_N \approx \sum_{j=0}^{N-1} L_j$, where $|E_N| \leq \sum_{j=0}^{N_1} |L_j|$. Assuming $L_j = \mathcal{O}((\Delta t)^{q+1})$, it holds that $E_N = \mathcal{O}((\Delta t)^{q+1} \cdot N = \frac{\mathcal{O}((\Delta t)^{q+1})T}{\Delta t} \approx \mathcal{O}((\Delta t)^q)$

2.11 Errors of different Methods:

Method	Truncation Error	One-Step Error
Forward Euler	$\mathcal{O}(\Delta t)$	$\mathcal{O}((\Delta t)^2)$
Backward Euler	$\mathcal{O}(\Delta t)$	$\mathcal{O}((\Delta t)^2)$
Trapezoidal	$\mathcal{O}((\Delta t)^2)$	$\mathcal{O}((\Delta t)^3)$
Mid-Point	$\mathcal{O}((\Delta t)^2)$	$\mathcal{O}((\Delta t)^3)$

2.12 Taylor Expansion Methods:

The Taylor expansions imply that for all time levels, it holds:

$$u(t^{n+1}) = u(t^n) + \Delta t u'(t^n) + \frac{(\Delta t)^2}{2}u''(t^n) + \frac{(\Delta t)^3}{6}u'''(t^n) + \cdots$$

With a given IVP $u'(t) = f(u(t^n))$, it holds that

$$u''(t^n) = f'(u(t^n)) \cdot f(u(t^n))$$
$$u'''(t^n) = f''(u(t^n)) \cdot f(u(t^n))^2 + f'(u(t^n))^2 \cdot f(u(t^n))$$

And the approximate solution is given by
$U_{n+1} = U_n + \Delta t f(U_n) + \frac{(\Delta t)^2}{2}f(U_n)f'(U_n) + \frac{(\Delta t)^3}{6}f'(U_n 9^2 f(U_n) + \frac{(\Delta t)^3}{6}f''(U_n)f(U_n)^2$.

3 Higher-Order Methods for ODEs

3.1 The Runge-Kutta-2 (RK-2) Method:

RK-2 is two-stage numerical scheme to the IVP:

$$\text{RK-2}\begin{cases} Y_1 = U_n \\ Y_2 = U_n + \frac{\Delta t}{2}F(t^n, Y_1) \\ U_{n+1} = U_n + \Delta t F(t^n + \frac{\Delta t}{2}, Y_2) \\ U_0 = u_0 \end{cases}$$

Global Error $E_n = \mathcal{O}((\Delta t)^2)$
One-Step Error $L_n = \mathcal{O}((\Delta t)^3)$

Butcher Tableau:

0	0	0
1/2	1/2	0
	0	1

⊕ fast (faster than trapez.), no extra memory

⊖ limited stability (needs small Δt)

3.2 The Classical Runge-Kutta-4 (RK-4) Method:

$$\begin{cases} Y_1 = U_n \\ Y_2 = U_n + \frac{\Delta t}{2}F(t^n, Y_1) \\ Y_3 = U_n + \frac{\Delta t}{2}F(t^n + \frac{\Delta t}{2}, Y_2) \\ Y_4 = U_n + \Delta t F(t^n + \frac{\Delta t}{2}, Y_3) \\ U_{n+1} = U_n + \frac{\Delta t}{6}(F(t^n, Y_1) + 2F(t^n + \frac{\Delta t}{2}, Y_2) + 2F(t^n + \frac{\Delta t}{2}, Y_3) \\ +F(t^n + \Delta t, Y_4)) \\ U_0 = u_0 \end{cases}$$

Global Error $E_n = \mathcal{O}((\Delta t)^4)$
One-Step Error $L_n = \mathcal{O}((\Delta t)^5)$

Butcher Tableau

0	0	0	0	0
1/2	1/2	0	0	0
1/2	0	1/2	0	0
1	0	0	1	0
	1/6	1/3	1/3	1/6

3.3 General Form of the Runge-Kutta Methods:

For increased clarity and simplicity, all the coefficients are usually presented in a so-called *Butcher Tableau*:

$$\begin{array}{c|ccc} c_1 & a_{11} & \cdots & a_{1s} \\ \vdots & \vdots & \ddots & \vdots \\ c_s & a_{s1} & \cdots & a_{ss} \\ \hline & b_1 & \cdots & b_s \end{array}$$

$$\begin{cases} Y_1 = U_n + \Delta t \sum_{j=1}^{s} a_{1j} \cdot F(t^n + c_j \Delta t, Y_j) \\ Y_i = U_n + \Delta t \sum_{j=1}^{s} a_{ij} \cdot F(t^n + c_j \Delta t, Y_j) \\ Y_s = U_n + \Delta t \sum_{j=1}^{s} a_{sj} \cdot F(t^n + c_j \Delta t, Y_j) \\ U_{n+1} = U_n + \Delta t \sum_{j=1}^{s} b_j \cdot F(t^n + c_j \Delta t, Y_j) \\ U_0 = u_0, \quad n = 0, 1, \cdots, \frac{T}{\Delta t} - 1 \end{cases}$$

3.3.1 Consistency for RK Methods ($\Delta t \to 0 \Rightarrow L_n \to 0$):

An s-stage RK method is consistent iff

$$\sum_{j=1}^{s} a_{ij} = c_i \quad \text{and} \quad \sum_{j=1}^{s} b_j = 1$$

3.4 Examples of Runge-Kutta Methods:

3.4.1 Explicit Runge-Kutta Methods:

A RK method is called explicit if all diagonal und upper diagonal terms of the Butcher tableau are 0, i.e. $a_{ij} = 0 \; if \; j \geq i$. (Strictly lower triangular)

3.4.2 Diagonally Implicit RK Methods (DIRK):

A RK method is called diagonally implicit if $a_{ij} = 0 \; if \; j > i$, and $a_{ij} \neq 0$. (Lower triangular)

3.5 Properties:

- Every explicit s-stage Runge-Kutta Method needs s function evaluations to perform one time step. (For implicit depend on function)
- An explicit RK method is never A-Stable
- For both explicit and implicit RK we don't need more info than only u_0
- Every A-Stable RK method is implicit, but not every implicit RK method is A-Stable.

$\oplus$ Adaptive time-stepping is easy to implement

3.6 Order of Accuracy of General RK Methods:

Further conditions are necessary for higher orders:

- At least order 2 if:$\sum_{j=1}^{s} b_j c_j = \frac{1}{2}$
- At least order 3 if: $\sum_{j=1}^{s} b_j c_j^2 = \frac{1}{3} \& \sum_{j=1}^{s} \sum_{i=1}^{s} b_j a_{ij} c_j = \frac{1}{6}$
- Higher Order: Additional conditions

In order to obtain γ-order accuracy where $\gamma < 5$, at least γ stages are necessary. In order to obtain γ-order accuracy where $\gamma \geq 5$, strictly more than γ stages are needed.

4 Multi-Step Methods for Solving ODEs

An alternative approach for obtaining high-order numerical methods is to use multi-step methods. The basic idea behind multi step methods is to compute the approximate solution $U_{n+\gamma}$ at the time level $t^{n+\gamma}$ using the approximate solutions $U_n \cdots, U_{n+\gamma-1}$at the previous γ time levels. The simplest examples of such multi-step methods are the so- called *linear* multi-step methods of the form

$$\sum_{j=0}^{\gamma} \alpha_j U_{n+j} = \Delta t \sum_{j=0}^{\gamma} \beta_j F(t^{n+j}, U_{n+j}).$$

4.1 Adams Methods:

The Adams methods are obtained by setting the coefficients

$$\begin{cases} \alpha_\gamma = 1 \\ \alpha_{\gamma-1} = -1 \\ \alpha_j = 0, \forall j < \gamma - 1 \end{cases} \Rightarrow U_{n+\gamma} = U_{n+\gamma-1} + \Delta t \sum_{j=0}^{\gamma} \beta_j F(U_{n+j})$$

4.1.1 Adams-Bashforth Method:

The Adams-Bashforth Methods for $\gamma = 1, 2, 3$ are given by

$$\begin{cases} (AB1): U_{n+1} = U_n + \Delta t F(U_n) \\ (AB2): U_{n+2} = U_{n+1} + \frac{\Delta t}{2}(-F(U_n) + 3F(U_{n+1})) \\ (AB3): U_{n+3} = U_{n+2} + \frac{\Delta t}{12}(5F(U_n) - 16F(U_{n+1})) + 23F(U_{n+2})) \end{cases}$$

The AB1 method is in fact the Forward Euler method.

4.1.2 Adams-Moulton Methods:

$$\begin{cases} (AM1): U_{n+1} = U_n + \frac{\Delta t}{2}(F(U_n) + F(U_{n+1})) \\ (AM2): U_{n+2} = U_{n+1} + \frac{\Delta t}{12}(-F(U_n) + 8F(U_{n+1}) + 5F(U_{n+2})) \\ (AM3): U_{n+3} = U_{n+2} + \frac{\Delta t}{24}(F(U_n) - 5F(U_{n+1})) + 19F(U_{n+2}) \\ +9F(U_{n+3}) \end{cases}$$

The AM1 method is in fact the Trapezoidal rule.

4.2 Truncation Error:

In order to ensure consistency, we must impose the conditions:

$$\sum_{j=0}^{\gamma} \alpha_j = 0 \quad \text{and} \quad \sum_{j=0}^{\gamma} j\alpha_j = \sum_{j=0}^{\gamma} \beta_j$$

A truncation error of order $(\Delta t)^k$ is obtained by setting $\sum_{j=0}^{\gamma} \frac{j^q}{q!} \alpha_j = \sum_{j=0}^{\gamma} \frac{j^{q-1}}{(q-1)!} \beta_j$.

4.3 Properties:

- A linear γ-step numerical method of this form requires γstarting values $U, \cdots, U_{\gamma-1}$. The usual strategy to specify the remaining starting values is to use a Runge-Kutta method of order $\gamma - 1$ in the case of an explicit Adams-Bashforth method, or to use a Runge-Kutta method of order γ in the case of an implicit Adams-Moulton method.
- One-Step Error $L_n = \mathcal{O}((\Delta t)^\gamma)$
- Global Error $E_N = \mathcal{O}((\Delta t)^\gamma)$
- An explicit multistep method with s steps needs one function evaluation to perform one time step.Implicit depend on function.
- Explicit multistep methods are never A-Stable.
- Implicit multistep methods can be A-Stable.

$\oplus$ Right hand side function F only needs to be evaluated once at each time step.

$\ominus$ Variable time-steps are difficult to implement.

$\ominus$ Require several starting values.

5 Stability of Numerical Methods for ODEs

Theorem: Convergent Numerical Method

A numerical method for approximating solutions to the IVP is said to be *convergent* if the computed solution U_N satisfies the following conditions for every fixed, final time $T > 0$.

1. $\lim_{\Delta \to 0} U_N = u(T)$
2. $\sum_{\Delta t \to 0}^{N \cdot \Delta t = T} U_j(\Delta t) = u_0 \qquad \forall 0 \leq j \leq y - 1$

Note: Convergence in not enough! Take as example the IVP: $u' = \lambda(u - \sin(t)) + \cos(t), u(0) = 0$ (Global error blows up)

5.1 Absolute and conditional Stability:

To determine the stability of a numerical method, we need stronger conditions. Consider the equation

$$\begin{cases} u'(t) = \lambda u(t) \\ u(0) = u_0 \end{cases}, \text{ where } \lambda \in \mathbb{R}^-.$$

The used method is absolutely stable iff $\frac{|U_{n+1}|}{|U_n|} \leq 1, \forall n \in \mathbb{N}$

Note: if Δt must satisfy a condition to garanty stability, so the method is called *conditionally stable*. The majority of method are only conditionally stable.

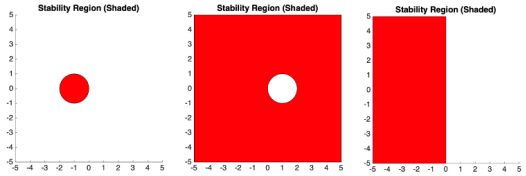
- There is no explicit method that is not conditionally stable
- Implicit method can be conditionally stable (depend on function)

Stabilities:

Method	Condition	Time step
Forward Euler	$\lvert 1+\lambda\Delta t\rvert \leq 1$	$\lambda\Delta t \in [-2, 0]$
Backward Euler	$\frac{1}{\lvert 1-\lambda\Delta t\rvert} \leq 1$	$\lambda\Delta t \in (-\infty, 0] \cup [2, \infty)$
Trapezoidal	$\lvert \frac{1+\frac{\lambda\Delta t}{2}}{1-\frac{\lambda\Delta t}{2}}\rvert \leq 1$	$\lambda\Delta t \in (-\infty, 0]$

5.1.1 Absolute Stability of Systems of ODEs:

Consider the linear system of ODEs $u'(t) = Au(t), where A \in \mathbb{R}^{mxm}$.
To determine the absolute stability of the numerical models, the eigenvalues of the matrix A need to be considered (additionally to the conditions above).



(a) Forward Euler method (b) Backward Euler method (c) Trapezoidal method

5.2 Stiff Problems:

Problems with large negative eigenvalues are called *stiff*. Methods whose stability region contains the entire negative (real) half of the complex plane are called $A-$*stable* numerical methods. A-stable methods (Backward Euler, Trapezoidal, etc.) are particularly well-suited for approximating solutions to stiff problems.

5.3 BFD Methods:

Higher order versions of the Backward Euler method are provided by the *Backward difference formula* (BDF) methods, which are of the general form:
$\alpha_0 U_n + \alpha_1 U_{n+1} + \cdots + \alpha_y U_{n+y} = \Delta t \beta_y F(U_{n+y})$, where $F(u) = u'$

Clearly, the Backward Euler method is a BDF-1 method.
Other examples of BDF methods include

$$\begin{cases} (BDF2): 3U_{n+2} - 4U_{n+1} + U_n = 2\Delta t F(U_{n+2}) \\ (BDF3): 11U_{n+3} - 18U_{n+2} + 9U_{n+1} - 2U_n = 6\Delta t F(U_{n+3}) \end{cases}$$

BDF methods are suited for stiff problems (chemistry).

6 The Poisson Equation

The Poisson equation is an elliptic PDE given by $-\Delta u = f$.
It can be derived as a steady state of the heat equation, potential flows or minimizations of J.

6.1 The Poisson Equation in 1D:

In one space dimension with the domain $\Omega = [0, 1]$, the Poisson Equation takes the form

$$\begin{cases} -u''(x) = f(x) \\ u(0) = \alpha, u(1) = \beta \quad (DBC) \\ u'(0) = d_1, u'(1) = d_2 \quad (NBC) \\ u(0) = \gamma, u'(1) = \eta \quad (MBC) \end{cases}$$

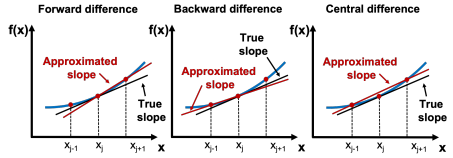
An explicit formula for the solution of this equation is given by the *Green's function*: $u(x) = \int_0^1 G(x, y) f(y) dy$, with:

$$G(x, y) = \begin{cases} y(1-x) 0 \leq y \leq x \\ x(1-y) x \leq y \leq 1 \end{cases}$$

- ⊖ Not possible to evaluate exactly for complicate functions f.
- ⊖ Not available in the case of 2D or 3D cases, except for very simple domains such as a ball.

6.2 Differences for derivatives:

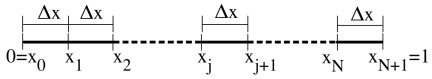
Forward Difference: $u'(x_i) = \frac{U_{i+1} - U_i}{\Delta x}$
Backward Difference: $u'(x_i) = \frac{U_i - U_{i-1}}{\Delta x}$
Central Difference: $u'(x_i) = \frac{U_{i+1} - U_{i-1}}{2\Delta x}$ or $u'(x_{i+\frac{1}{2}}) = \frac{U_{i+1} - U_i}{\Delta x}$



6.3 Finite Difference Method for 1D:

Discretizing the domain: let $N = \frac{1}{\Delta x} - 1$. We discretize the domain $[0; 1]$ into $N + 2$ points by setting:

$$x_0 = 0, x_{N+1} = 1, x_i = i \cdot \Delta x, i = 1 \ldots, N$$



Discretizing the derivatives: we use central difference approximation of the second derivative of u given by:

$$u''(x_i) \approx \frac{U_{i+1} - 2U_i + U_{i-1}}{\Delta x^2}$$

with $U_i \approx u(x_i)$ and $F_i \approx f(x_i)$.
The Finite Difference Scheme: The PE can be solved by

$$-U_{i+1} + 2U_i - U_{i-1} = \Delta x^2 F_i, \forall i = 1, \ldots, N$$

6.3.1 Dirichlet boundary conditions:

- for $i = 1$: $-u_2 + 2u_1 = \Delta x^2 f(x_1) + \alpha$
- for $i = N$: $2u_N - u_{N-1} = \Delta x^2 f(x_N) + \beta$

Then introducing the vectors: $U = [U_1, U_2, \ldots, U_N]^T, F = \Delta x^2 [F_1 + \alpha, F_2, \ldots, F_N + \beta]^T$. We can so solve the problem with the matrix equation $A \cdot U = F$, with:

$$A_{DBC} = \begin{bmatrix} 2 & -1 & 0 & \ldots & 0 \\ -1 & 2 & -1 & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & -1 & 2 & -1 \\ 0 & \ldots & 0 & -1 & 2 \end{bmatrix} \in \mathbb{R}^{N \times N}$$

Note: A is tridiagonal and diagonally dominant $\Rightarrow$ invertible.

6.3.2 Neumann boundary conditions:

We use central difference on $u_i' \approx d_k$ and introduce $i = 0, N + 1$:

- for $i = 0$: $-u_{-1} + 2u_0 - u_1 = \Delta x^2 f(x_0)$, with $u'(x_0) = d_1 \approx \frac{u_1 - u_{-1}}{2\Delta x}$
- for $i = N + 1$: $-u_{N+2} + 2u_{N+1} - u_N = \Delta x^2 f(x_{N+1})$ with $u'(x_{N+1}) = d_2 \approx \frac{u_{N+2} - u_N}{2\Delta x}$

Then introducing the vectors: $U = [U_0, U_1, \ldots, U_N, U_{N+1}]^T, F = \Delta x^2 [F_0 - \frac{2}{\Delta x} d_1, F_1, \ldots, F_N, F_{N+1} + \frac{2}{\Delta x} d_2]^T$. We can so solve the problem with the matrix equation $A \cdot U = F$, with:

$$A_{NBC} = \begin{bmatrix} 2 & -2 & 0 & \dots & \dots & \dots & 0 \\ -1 & 2 & -1 & 0 & \dots & \dots & \vdots \\ 0 & -1 & 2 & -1 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \vdots & \ddots & -1 & 2 & -1 & 0 \\ \vdots & \dots & 0 & 0 & -1 & 2 & -1 \\ 0 & \dots & \dots & \dots & \dots & -2 & 2 \end{bmatrix} \in \mathbb{R}^{(N+2)\times(N+2)}$$

6.3.3 Mixed boundary conditions:

- for $i = 1$: $-u_2 + 2u_1 = \Delta x^2 f(x_1) + \gamma$
- for $i = N$: $-u_{N+1} + 2u_N - u_{N-1} = \Delta x^2 f(x_N)$ with $u'(x_{N+1}) = \eta \approx \frac{u_{N+1}-u_N}{\Delta x}$

Note: Backward approx for $u'(x_{N+1})$. The resulting matrix $A \in \mathbb{R}^{(N+1)\times(N+1)}$.

6.4 Finite Difference Schemes for the 2D PE:

The 2D PE on the domain $\Omega = [0,1]^2$ is given by:

$$\begin{cases} -\Delta u = f(x), \forall x \in \Omega = [0,1]^2 \\ u(x) = 0, \forall x \in \partial\Omega \quad (HBC) \\ u(x) = g(x,y), \forall x \in \partial\Omega \quad (IBC) \end{cases}$$

Discretizing the domain: We can discretize the domain Ω into a set of $(N+2)\times(M+2)$ points by setting:

$$N = \frac{1}{\Delta x} - 1 \quad x_0 = 0 \quad x_{N+1} = 1 \quad x_i = i \cdot \Delta x \quad i = 1, \dots, N$$
$$M = \frac{1}{\Delta y} - 1 \quad y_0 = 0 \quad y_{M+1} = 1 \quad y_i = i \cdot \Delta y \quad i = 1, \dots, M$$

Discretizing the Laplacian: with $U_{i,j} \approx u(x_i, y_j)$ and $F_{i,j} \approx f(x_i, y_j)$ the Laplacian is:

$$u_{xx} \approx \frac{U_{i+1,j} - 2U_{i,j} + U_{i-1,j}}{\Delta x^2}, u_{yy} \approx \frac{U_{i,j+1} - 2U_{i,j} + U_{i,j-1}}{\Delta y^2}$$

The finite Difference Scheme $(\forall i = 1, \dots, N, j = 1, \dots, M)$**:**

$$-(u_{xx} + u_{yy}) = F_{i,j}$$

6.4.1 Homogeneus boundary conditions:

The boundary conditions imposed are:

$$u_{0,j} = u_{N+1,j} = 0, \quad \forall j = 0, 1, \dots, M+1$$
$$u_{i,0} = u_{i,M+1} = 0, \quad \forall i = 0, 1, \dots, N+1$$

If we set $\Delta x = \Delta y$ and introduce the vectors:

$$U = [U_{1,1}, \dots, U_{N,1}, U_{1,2}, \dots, U_{N,2}, \dots, U_{1,N}, \dots, U_{N,N}]^T$$
$$F = \Delta x^2 [F_{1,1}, \dots, F_{N,1}, F_{1,2}, \dots, F_{N,2}, \dots, F_{1,N}, \dots, F_{N,N}]^T$$

The system is: $A \cdot U = F$, with $(B \in \mathbb{R}^{N\times N})$:

$$A = \begin{bmatrix} B & -\mathbb{I} & 0 & \dots & 0 \\ -\mathbb{I} & B & -\mathbb{I} & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & -\mathbb{I} & B & \mathbb{I} \\ 0 & \dots & 0 & -\mathbb{I} & B \end{bmatrix}, B = \begin{bmatrix} 4 & -1 & 0 & \dots & 0 \\ -1 & 4 & -1 & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & -1 & 4 & -1 \\ 0 & \dots & 0 & -1 & 4 \end{bmatrix}$$

6.4.2 Inomogeneus boundary conditions:

Using the lexicographic notation: (x_n, y_n) with $n = 0, \dots, (N+1)^2$. We only have to modify F:

$$F_n = \Delta x^2 f(x_n, y_n) + \delta_{j,1} g(x_n, 0) + \delta_{j,N} g(x_n, 1) + \delta_{i,1} g(0, y_n) + \delta_{i,N} g(1, y_n)$$

where $i, j \in \{1, \dots, N\}$ are such that $n = i + (j-1) \cdot N$.

6.5 Property of FD:

- $\oplus$ Very easy to implement and generalize, efficiently solved
- For both 1D/2D: $E_N = \mathcal{O}(N^{-2})$
- $\ominus$ Suffers curse of dimensionality: given runtime $\tau \Rightarrow \mathcal{O}(\tau) = \mathcal{O}(N^{-1}) = \mathcal{O}(E_N^{-2}), \mathcal{O}(\tau) = \mathcal{O}(E_N^{-1})$. Difficult to use for non uniform grid ($\oplus$ triangle, $\ominus$ rectangular mesh)

7 FEM for the 1D PE

7.1 A Variational Principle:

$$J(u) = \frac{1}{2}\int_\Omega |\nabla u|^2 dx - \int_\Omega u \cdot f dx, \quad \forall x \in \Omega, u_{|\partial\Omega} = 0$$

Definitions:

$$H_a^b([c,d]) = \left\{ u : [c,d] \to \mathbb{R} : u(c) = u(d) = a \text{ and } \int_c^d |u^{(b)}(x)|^2 dx < \infty \right\}$$
$$L^e([c,d]) = \left\{ u : [c,d] \to \mathbb{R} : \int_c^d |u(x)|^e dx < \infty \right\}$$

Norms: $(a = c = 0, b = d = 1 \text{ and } e = 2)$

$$\|u\|_{H_0^1([0,1])} = \left(\int_0^1 |u'(x)|^2 dx\right)^{1/2}, \|u\|_{L^2([0,1])} = \left(\int_0^1 |u(x)|^2 dx\right)^{1/2}$$

Theorem: Poincaré $\oplus$ Chauchy Schwarz Inequality

- PI: $\|u\|_{L^2([0,1])} \le \|u\|_{H_0^1([0,1])}$
- CSI: $\forall f, g \in L^2 \Rightarrow \int_\Omega f \cdot g \, dx \le \int_\Omega |f| \, dx \cdot \int_\Omega |g| \, dx$

Note: $H_0^1([0,1])$ is a prototypical example of Solobov space.

Theorem: Variational Formulation

Given $f \in L^2([0,1])$, find $u \in H_0^1([0,1])$, such that u minimizes the Energy function $J(u) \forall u \in H_0^1([0,1])$:

$$\int_0^1 u'(x) \cdot v'(x) dx = \int_0^1 v(x) \cdot f(x) dx.$$

7.2 Finite Element Formulation:

Discretizing the domain: as in Section 6.3 with $h = \Delta x$ and defining the subspace $V^h \subseteq V$ (N dimensional) with

$$V^h = \Big\{ w : [0,1] \to R : w(0) = w(1) = 0, w \text{ continuous}, \\ w|_{[x_i, x_{i+1}]} \; i = 1, \dots N \text{ is linear} \Big\}$$

Define a basis of V^h is given by the hat functions:

$$\phi_i(x_j) = \begin{cases} 1 & i = j \\ 0 & \text{else} \end{cases}$$

$$\phi_i(x) = \begin{cases} \frac{x - x_{i-1}}{h} & x \in [x_{i-1}, x_i) \\ \frac{x_{i+1} - x}{h} & x \in [x_i, x_{i+1}) \\ 0 & \text{else} \end{cases}, \phi_i(x)' = \begin{cases} \frac{1}{h} & x \in [x_{i-1}, x_i) \\ -\frac{1}{h} & x \in [x_i, x_{i+1}) \\ 0 & \text{else} \end{cases}$$

Discrete variational Formulation: setting $v = \sum_{j=1}^N v_j \cdot \phi_j(x) \in V^h$ and $u_h = \sum_{i=1}^N u_i \cdot \phi_i(x)$, and using the notation $\langle g, h \rangle = \int_0^1 g(x) h(x) dx$, the variational principle becomes

$$\sum_{i=1}^N u_i \cdot \langle \phi_i'(x), \phi_j'(x) \rangle = \langle f(x), \phi_j(x) \rangle$$

which may be rewritten as $A \cdot U = F$ with $A_{ij} = \langle \phi_i', \phi_j' \rangle = A_{ji}$, $F_j = \langle f, \phi_j \rangle$. **Note:** $\forall i, j = 1, \dots, N$ it holds that:

- $A_{ij} = 0$, if $|i - j| > 1$
- $A_{j-1,j} = \int_{x_{j-1}}^{x_j} \left(\frac{x - x_{j-1}}{h}\right)' \cdot \left(\frac{x_j - x}{h}\right)' dx = -\frac{1}{h} = A_{j,j-1}$
- $A_{j+1,j} = \int_{x_j}^{x_{j+1}} \left(\frac{x - x_j}{h}\right)' \cdot \left(\frac{x_{j+1} - x}{h}\right)' dx = -\frac{1}{h} = A_{j,j+1}$
- $A_{j,j} = \int_{x_{j-1}}^{x_j} \left(\frac{x - x_{j-1}}{h}\right)' \cdot \left(\frac{x - x_{j-1}}{h}\right)' dx + \int_{x_j}^{x_{j+1}} \left(\frac{x_{j+1} - x}{h}\right)' \cdot \left(\frac{x_{j+1} - x}{h}\right)' dx = \frac{2}{h}$

$$A = \frac{1}{h} \begin{bmatrix} 2 & -1 & 0 & \dots & 0 \\ -1 & 2 & -1 & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & -1 & 2 & -1 \\ 0 & \dots & 0 & -1 & 2 \end{bmatrix}, F_j = \int_0^1 f(x)\phi_j(x)dx = h \cdot f_j$$

7.3 Convergence Anaysis:

Theorem: Cèa (Lemma)

$\forall v \in V^h$: $\mathcal{O}(h^2) \approx \|u_h - u\|_{L^2} \leq \|u_h - u\|_{H_0^1} = \mathcal{O}(h)$
Note: Poincarè inequality says: $\|u_h - u\|_{L^2} \leq Ch$ but in practice $EOC \approx 2$.

1D: $\mathcal{O}(h^2) = \mathcal{O}(N^{-2})$ **2D:** $\mathcal{O}(h^2) = \mathcal{O}(N^{-1})$

Higher dimension $\Rightarrow$ slower convergence w.r.t. N

8 FEM for the 2D PE

Theorem: Green first identity $\oplus$ Gauss (Divergence)

- GI: $\int_\Omega \langle \nabla u, \nabla v \rangle \, dx = \int_{\partial\Omega} \langle v \nabla u, n \rangle \, dO - \int_\Omega v \cdot \Delta u \, dx$
- DT: $\int_\Omega \nabla u \, dx = \int_{\partial\Omega} \langle u, n \rangle \, dxO$

Definitions:
$H_0^1(\Omega) = \{v : \Omega \to \mathbb{R} : \int_\Omega |\nabla v|^2 dx < \infty \text{ and } v = 0 \text{ on } \partial\Omega\}$
$L^2(\Omega) = \{v : \Omega \to \mathbb{R} : \int_\Omega |v|^2 dx < \infty\}$
Norms: $\|v\|_{H_0^1(\Omega)} = \left(\int_\Omega |\nabla v|^2 dx\right)^{1/2}$, $\|v\|_{L^2(\Omega)} = \left(\int_\Omega |v|^2 dx\right)^{1/2}$
Notation:
$\langle u, v \rangle_{H_0^1(\Omega)} = \int_\Omega \langle \nabla u, \nabla v \rangle dx$, $\langle u, v \rangle_{L^2(\Omega)} = \int_\Omega u(x) \cdot v(x) dx$.
We need to solve:

$$\langle u, v \rangle_{H_0^1(\Omega)} = \langle f, v \rangle_{L^2(\Omega)}$$

Example: Porous Membrane variational formulation

$$\begin{cases} -\nabla \cdot (\sigma \nabla u) + ru = f(x) \text{ in } \Omega \in \mathbb{R}^2 \\ u(x) = g(x) \text{ on } \partial\Omega \end{cases}$$

We multiply both sides by a text function $v \in H_0^1([0,1])$:

$$-\int_\Omega \nabla(\sigma \nabla u) v dx + \int_\Omega ruv dx = \int_\Omega f(x) v dx$$

By applying *Greens Formula*, we get:

$$\int_\Omega \sigma \nabla u \nabla v dx - \int_{\partial\Omega} \sigma \nabla u \cdot \underline{n} v dx + \int_\Omega ruv dx = \int_\Omega f(x) v dx$$

With $v = 0$ on $\partial\Omega$ we get the variational formulation:
Find $u \in V = \{w \in H_0^1(\Omega) : w = g \text{ on } \partial\Omega\}$ such that:

$$\int_\Omega \sigma \nabla u \nabla v dx + \int_\Omega ruv dx = \int_\Omega f(x) v dx, \forall v \in H_0^1(\Omega)$$

Note: The strong formulation assumes that for the solution u, the Laplacian operator is well defined (u twice differentiable). Many problems have solutions which are continuous, but piecewise differentiable. Therefore variational formulation is better.

8.1 The Finite Element Formulation:

We discretize the domain using non-overlapping triangles with $h = \max_{K \in T_h} diam(K)$, where $diam(K)$ is the longest edge of the triangle K. We define the finite-dimensional subspace V^h as:

$$V^h = \{v : \Omega \to \mathbb{R} : v \text{ continuous}, v|_K \text{ linear } \forall K, v|_{\partial\Omega} = 0\}$$

A basis of V^h is given by the 2D hat functions:

$$\phi_i(\mathcal{N}_j) = \begin{cases} 1, & i = j \\ 0, & \text{else} \end{cases} \quad \text{with } \{\mathcal{N}_j\}_{j=1}^N = \text{ interior nodes}$$

As in the 1D formulation, the variational principle becomes:

$$\underbrace{\sum_{i=1}^N u_i}_{U_i} \cdot \underbrace{\int_\Omega \langle \nabla \phi_i, \nabla \phi_j \rangle dx}_{A_{ij}} = \underbrace{\int_\Omega f(x) \phi_j(x) dx}_{F_i}$$

which can be written as $A \cdot U = F$ with A, B as in 6.4.1.

Example: Porous Membrane A and F

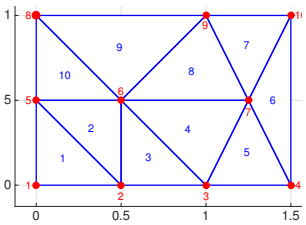
$A_{ij} = \int_\Omega \left(\sigma(x) \nabla \phi_i^N(x) \nabla \phi_j^N(x) + r \phi_i^N(x) \phi_j^N(x)\right) dx$
$F_j = \int_\Omega f(x) \phi_i^N(x) dx$

9 FEM Implementation

9.1 Triangulation:

Let $\{\mathcal{N}_i\}_{i=1}^N$ denote the nodes of the triangulation and $\{\mathcal{K}_i\}_{j=1}^M$ denote the triangles in T_h:

- Z is the $2 \times N$ array of nod values such that:
  - $Z(\cdot, j)$ refers to the node $\mathcal{N}_j$
  - $Z(1, j), Z(2, j)$ represent the x, y coordinates of the node $\mathcal{N}_j$
- T is the $3 \times M$ array of triangles such that:
  - $T(\cdot, j)$ refers to the j^{th} triangle K_j
  - $T(i, j)$, for $i = 1, 2, 3$ represent the three node vertices corresponding to the triangle K_j
- The coordinates of each vertex $T(\alpha, m)$ are given by $Z(i, T(\alpha, m)), i = 1, 2$
- B is the $1 \times N$ matrix which says if a Node is on the boundary or not: $B = [1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 0 \quad 0 \quad 1 \quad 1 \quad 1]$. Node is on boundary if $B(n) == 1$



$$Z = \begin{bmatrix} 0 & 0.5 & 1 & 1.5 & \dots & 1.5 \\ 0 & 0 & 0 & 0 & \dots & 1 \end{bmatrix}$$

$$T = \begin{bmatrix} 1 & 2 & 2 & \dots & 5 \\ 2 & 5 & 3 & \dots & 6 \\ 5 & 6 & 6 & \dots & 8 \end{bmatrix}$$

9.2 Building Stiffness Matrix and Load Vectors:

$$A_{ij} = \int_\Omega \langle \nabla \phi_i, \nabla \phi_j \rangle \, dx = \sum_{m=1}^M \int_{Km} \langle \nabla \phi_i, \nabla \phi_j \rangle \, dx$$

where K_m is the m^{th} triangle in the triangulation T_h. For a triangle K_m with nodes $\mathcal{N}_a$, $\mathcal{N}_b$, $\mathcal{N}_c$ we have so:

$$A^{K_m} = \begin{bmatrix} \int_{K_m} \langle \nabla \phi_a, \nabla \phi_a \rangle \, dx & \int_{K_m} \langle \nabla \phi_a, \nabla \phi_b \rangle \, dx & \int_{K_m} \langle \nabla \phi_a, \nabla \phi_c \rangle \, dx \\ \int_{K_m} \langle \nabla \phi_b, \nabla \phi_a \rangle \, dx & \int_{K_m} \langle \nabla \phi_b, \nabla \phi_b \rangle \, dx & \int_{K_m} \langle \nabla \phi_b, \nabla \phi_c \rangle \, dx \\ \int_{K_m} \langle \nabla \phi_c, \nabla \phi_a \rangle \, dx & \int_{K_m} \langle \nabla \phi_c, \nabla \phi_b \rangle \, dx & \int_{K_m} \langle \nabla \phi_c, \nabla \phi_c \rangle \, dx \end{bmatrix}$$

Reference $\hat{K}$ can be mapped to K using a mapping $\Phi_K : \hat{K} \to K$:

$$x = \Phi_K(\hat{x}) = [\mathcal{N}_b - \mathcal{N}_a \quad \mathcal{N}_c - \mathcal{N}_a] \cdot \hat{x} + \mathcal{N}_a = J_K \cdot \hat{x} + \mathcal{N}_a$$

where J_K is the Jacobian of Φ_K and $\hat{x} = [\hat{x} \quad \hat{y}]^T$ are the coordinates in the reference element. The local stiffness matrix and load vector are computed on the reference element $\hat{K}$ as:

$$A^K_{\alpha\beta} = \int_K \langle \nabla\phi_\alpha, \nabla\phi_\beta \rangle dx = \int_{\hat{K}} \langle J_k^{-T}\hat{\nabla}\hat{\phi}_\alpha, J_K^{-T}\hat{\nabla}\hat{\phi}_\beta \rangle \cdot |\det(J_K)| d\hat{x}$$

$$F^K_\alpha = \int_K f(x)\phi_\alpha(x)dx = \int_{\hat{K}} f(\Phi_K(\hat{x})) \cdot \hat{\phi}_\alpha(\hat{x}) \cdot |\det(J_K)| d\hat{x}$$

where J_K^{-T} is short for $(J_K^{-1})^T$ and $\hat{\phi}_\alpha, \alpha = 1,2,3$ are the local shape functions of $\hat{K}$: $\hat{\phi}_1(\hat{x},\hat{y}) = 1-\hat{x}-\hat{y}, \hat{\phi}_2(\hat{x},\hat{y}) = \hat{x}, \hat{\phi}_3(\hat{x},\hat{y}) = \hat{y}$ with $\hat{\nabla}\hat{\phi}_1 = \begin{bmatrix}-1\\-1\end{bmatrix}, \hat{\nabla}\hat{\phi}_2 = \begin{bmatrix}1\\0\end{bmatrix}, \hat{\nabla}\hat{\phi}_3 = \begin{bmatrix}0\\1\end{bmatrix}$

Example: Porous Membrane Stiffnes Matrix entries

Compute the element stiffness matrix $A^{\hat{K}}$ for the reference triangle $\hat{K}$ considering $\sigma = 1$ and $r = 1$. The PDE suggests that this matrix can be written as the sum of two contributions: $A^{\hat{K}} = A^{\hat{K},1} + A^{\hat{K},2}$, where $A^{\hat{K},1}$ is the stiffness matrix for $c = 0$ and $A^{\hat{K},2}$ is the matrix associated to the $c \cdot u$ term.

1. $A^{\hat{K},1}_{11} = \int_{\hat{K}} \hat{\nabla}\hat{\phi}_1\hat{\nabla}\hat{\phi}_1 dx = 2 \cdot |\hat{K}| = 1$
2. $A^{\hat{K},1}_{12} = \int_{\hat{K}} \hat{\nabla}\hat{\phi}_1\hat{\nabla}\hat{\phi}_2 dx = -|\hat{K}| = \frac{-1}{2} = A^{\hat{K},1}_{21}$
3. ...
4. $A^{\hat{K},2}_{11} = \int_{\hat{K}} \hat{\phi}_1\hat{\phi}_1 dx = \int_0^1 \int_0^{1-\hat{x}} (1-\hat{x}-\hat{y})^2 dy dx = \frac{1}{12}$

9.3 Assembly:

A = zeros(N, N); F = zeros(N);
for $m = 1 : M$ **do**
  fetch $A^m = \{A^m_{\alpha,\beta}\}$, $F^m = \{F^m_\alpha\}$, $\alpha, \beta = 1,2,3$;
  for $\alpha = 1,2,3$ **do**
    $F(T(\alpha, m))$ += F^m_α;
    for $\alpha = 1,2,3$ **do**
      $A(T(\alpha, m), T(\beta, m))$ += $A^m_{\alpha,\beta}$;
    end
  end
end

In case of homogeneus boundary conditions contributions: from boundary nodes should be skipped. In pseudo code:

Data: A, F compute from above;
Compute: InteriorNodes;
$A = A$(InteriorNodes, InteriorNodes);
$F = F$(InteriorNodes);

9.4 Solving the Linear System:

The FEM approximation of the solution u_h can be computed as (solving $A \cdot U = F$): $u_h(x) = \sum_{i=1}^N u_i\phi_i(x)$, with $U = \{u_i\}_{i=1}^N$

9.5 Inhomogeneus Boundary Conditions:

$$\begin{cases} -\Delta u + ru = f \text{ in } \Omega \in \mathbb{R}^2 \\ u = g \text{ on } \partial\Omega \end{cases}$$

We define $u_0 = u - \tilde{g} \to u = u_0 + \tilde{g}$ and $u_{0|\partial\Omega} = u_{|\partial\omega} - \tilde{g} = g - g = 0$ so we can rewrite $f = -\Delta u = -\Delta(u_0 + \tilde{g})$ and we need to solve:

$$\langle u_0, v\rangle_{H^1_0(\Omega)} = \langle f, v\rangle_{L^2(\Omega)} - \langle \tilde{g}, v\rangle_{H^1_0(\Omega)}$$

where $\tilde{g}_h(x) = \sum_{x_i = \text{ boundary nodes}} g(x_i)\phi_i(x)$ and $u_h = u_{0,h} + \tilde{g}_h$.

Data: A, F compute from above;
Compute: InteriorNodes;
U = zeors(N, 1), $U(i) = g(x_i)$ for all boundary nodes x_i;
$F = F - AU$;
$A = A$(InteriorNodes, InteriorNodes);
$F = F$(InteriorNodes);
Solve: $A \cdot U$(InteriorNodes) $= F$;

10 Parabolic Partial Differential Equations

The Heat Equation is a parabolic differential equation given by:

$$\begin{cases} u_t - \Delta u = 0, & \text{on } \Omega \times (0,T) \\ u(x,0) = u_0(x), & \text{on } \Omega \quad IC \\ u(0,t) = u(1,t) = 0, & \text{on } (0,T) \quad BC \end{cases}$$

10.1 Exact solution to the Heat Equation:

Using the separation of variables $u(x,t) = \mathcal{T}(t) \cdot \mathcal{X}(x)$ the exact solution is given by:

$$u(x,t) = \sum_{k=1}^{\infty} u_k^0 e^{-(k\pi)^2 t} \sin(k\pi x)$$

with $u_k^0 = 2\int_0^1 u_0(x)\sin(k\pi x)dx, k \in \mathbb{Z}$.
Note: Error due to finite-truncation of the Fourier sine series, and error due to the use of quadrature rules to approximate integrals.

10.2 Energy Estimate:

We define the energy function $\mathcal{E} : [0,T] \to \mathbb{R}$ as:

$$\mathcal{E} = \frac{1}{2}\int_0^1 |u(x,t)|^2 dx$$

Stability: $\mathcal{E}(t) \le \mathcal{E}(0)$, since $\frac{d\mathcal{E}}{dt} = -\int_0^1 u_x^2 dx \le 0$. Therefore solutions to the HE are stable.
Uniqueness of the solution: Let $u, \bar{u}$ be solutions to the HE, and let $w := u - \bar{u}$. Using the stability of HE it follows, that

$$\int_0^1 w^2(x,t)dx \le \int_0^1 w^2(x,0)dx \Rightarrow \int_0^1 w^2(x,t)dx \le 0 \quad \text{(Using IC)}$$
$$w(x,t) \equiv 0 \iff u(x,t) = \bar{u}(x,t)$$

10.3 Explicit Finite Difference for the HE (FE):

Discretizing the domain into $(N+2) \times (M+2)$ points.

$$N = \frac{1}{\Delta x} - 1 \quad x_0 = 0 \quad x_{N+1} = 1 \quad x_j = j \cdot \Delta x \quad ij = 1, \dots, N$$
$$M = \frac{T}{\Delta t} - 1 \quad t^0 = 0 \quad t^{M+1} = T \quad t^n = n \cdot \Delta t \quad n = 1, \dots, M$$

Discretizing the solution u: $U_j^n \approx u(x_j, t^n)$
Discretizing the derivatives with central difference and Forward Euler:

$$u_{xx}(x_j,t^n) \approx \frac{U^n_{j+1} - 2U^n_j + U^n_{j-1}}{\Delta x^2}, u_t(x_j,t^n) \approx \frac{U^{n+1}_j - U^n_j}{\Delta t}$$

Plugging the approximation in HE and solving for U_j^{n+1} lead to:

$$U_j^{n+1} = (1-2\lambda)U_j^n + \lambda U^n_{j+1} + \lambda U^n_{j-1}, \text{ with } \lambda = \frac{\Delta t}{\Delta x^2}$$

which can be rewritten as $U^{n+1} = (\mathbb{I} - \lambda \cdot A_{DBC}) \cdot U^n$.
In order to get a stable solution ($\mathcal{E}^{n+1} \le \mathcal{E}^n$), the *Courant-Friedrich-Levy Condition* needs to be satisfied: $\lambda \le \frac{1}{2}$.
The Trucation Error T_j^n is defined as:

$$T_j^n = \frac{u_j^{n+1} - u_j^n}{\Delta t} - \frac{u^n_{j-1} - 2u^n_j + u^n_{j+1}}{\Delta x^2} \le C(\Delta t + \Delta x^2)$$

10.4 Maximum Principle:

Theorem: Maximum/Minimum principle

Let u be the solution of the heat equation Then, for all $x \in [0,1]$ and for all $t \in [0,T]$ it holds that:

$$\min\left(0, \min_{\tilde{x}}(u_0(\tilde{x}))\right) \leq u(x,t) \leq \max\left(0, \max_{\tilde{x}}(u_0(\tilde{x}))\right)$$

Max/Min are either at the boundaries or on the initial line $t = 0$.

10.4.1 Discrete Maximum Principle:

$$\min\left(0, \min_{j}(U_j^0)\right) \leq U_j^{n+1} \leq \max\left(0, \max_{j}(U_j^0)\right)$$

Consider the Finite Difference scheme with $\lambda \leq \frac{1}{2}$ (coeff ≥ 0).

$$U_i^{n+1} = (1-2\lambda)\cdot U_i^n + \lambda \cdot U_{i+1}^n + \lambda \cdot U_{i-1}^n$$

Let $\bar{U}_i^n = \max(U_{i-1}^n, U_i^n, U_{i+1}^n)$, and by definition:

$$U_{j-1}^n \leq \bar{U}_j^n, \quad U_j^n \leq \bar{U}_j^n, \quad U_{j+1}^n \leq \bar{U}_j^n$$

Therefore one can write:

$$\begin{aligned} U_i^{n+1} &\leq (1-2\lambda)\cdot \bar{U}_i^n + \lambda \cdot \bar{U}_i^n + \lambda \cdot \bar{U}_i^n \\ &\leq \bar{U}_i^n = \max(U_{i-1}^n, U_i^n, U_{i+1}^n) \end{aligned}$$

10.5 Implicit Finite Difference Scheme (BE):

The CFL condition is extremely constraining as the relation $\Delta t \approx \Delta x^2$ results in very small time steps. To remedy this, we use an implicit finite difference scheme (Backward Euler):

$$\frac{U_j^{n+1} - U_j^n}{\Delta t} = \frac{U_{j-1}^{n+1} - 2U_j^{n+1} + U_{j+1}^{n+1}}{\Delta x^2}$$

which is: $(\mathbb{I} + \lambda \cdot A_{DBC}) \cdot U^{n+1} = A \cdot U^{n+1} = F^n$ with

$$A = \begin{bmatrix} 1+2\lambda & -\lambda & 0 & \dots & 0 \\ -\lambda & 1+2\lambda & -\lambda & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & -\lambda & 1+2\lambda & -\lambda \\ 0 & \dots & 0 & -\lambda & 1+2\lambda \end{bmatrix}$$

The discrete energy of the solution does not increase over time. This, the IFD scheme is termed unconditionally stable. Therefore, the CFL condition will always be satisfied
The Truncation Error is given by $|T_j^n| \leq C \cdot (\Delta t + \Delta x^2)$.

10.6 Crank-Nicolson Scheme (FE/2 + BE/2):

Higher order time accuracy: CN $\rightarrow u_{xx} \approx \frac{u_{xx}(x_j,t^n) + u_{xx}(x_j,t^{n+1})}{2}$

$$\frac{U_j^{n+1} - U_j^n}{\Delta t} = \frac{U_{j-1}^n - 2U_j^n + U_{j+1}^n}{2\Delta x^2} + \frac{U_{j-1}^{n+1} - 2U_j^{n+1} + U_{j+1}^{n+1}}{2\Delta x^2}$$

for all $j = 1, \dots, N$ and for all $n = 0, \dots, M$. With $F_j = \frac{\lambda}{2} \cdot U_{j-1}^n + (1-\lambda) U_j^n + \frac{\lambda}{2} U_{j+1}^n$ we can write the Methos as:

$$(\mathbb{I} + \frac{\lambda}{2} A_{DBC}) U^{n+1} = A U^{n+1} = F^n = B U^n = (\mathbb{I} - \frac{\lambda}{2} A_{DBC}) U^n \text{ with:}$$

$$A = \begin{bmatrix} 1+\lambda & -\frac{\lambda}{2} & 0 & \dots & 0 \\ -\frac{\lambda}{2} & 1+\lambda & -\frac{\lambda}{2} & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & -\frac{\lambda}{2} & 1+\lambda & -\frac{\lambda}{2} \\ 0 & \dots & 0 & -\frac{\lambda}{2} & 1+\lambda \end{bmatrix}$$

Note: Since the discrete Energy produced by CN scheme doesn't increase over time, the CN scheme is *unconditionally stable.*
Attention: Despite the unconditional stability of the CN scheme, it can be shown that the scheme is only stable if $\lambda = \frac{\Delta t}{\Delta x^2} \leq 1$.

Therefore in contrast to implicit scheme, the CN scheme doesn't satisfy the Max principle always! Flip sign in A to obtain B.
The Truncation Error is given by: $|T_j^n| \leq C \cdot (\Delta t^2 + \Delta x^2)$.

11 Linear Transport Equation (hyperbolic)

$$\begin{cases} u_t + a(x,t) \cdot u_x = 0 & \forall (x,t) \in \mathbb{R} \times \mathbb{R}_+ \\ u(x,0) = u_0(x) & \forall x \in \mathbb{R} \end{cases}$$

11.1 Method of Characteristics:

Hence $u = u(x(t),t)$: $\frac{d}{dt} u(x(t),t) = u_t(x(t),t) + u_x(x(t),t) \cdot \dot{x}(t)$
Assume given $x(t)$ along which $u(x,t)$ is constant.

$$\frac{d}{dt} u(x(t),t) = 0 \Rightarrow u_t + \dot{x}(t) \cdot u_x = 0$$

So: $\begin{cases} \dot{x}(t) & = a(x,t) \\ x(0) & = x_0. \end{cases}$, $x(t)$ is called a *characteristic curve.*
From the initial assumption it follows, that:

$$u(x(t),t) = u(x(0),0) = u(x_0,0) = u_0(x_0).$$

11.1.1 Maximum Principle for LTE:

Consider the above LTE with constant a. If the solution u is continuous and the domain bounded ($x \in [a,b]$) the supremum is attained, hence: $\forall t > 0, \quad \max_{x\in[a,b]} u(x,t) = \max_{x\in[a,b]} u_0(x)$
By contradiction: suppose $\exists \hat{x}$ (1), $\exists \hat{t}$ (2) such that the maximum principles is invalid:
1 $\max_{x\in[a,b]} u_0(x) < u(\hat{x},\hat{t}) = u(x(0),0) = u_0(x(0)) \leq \max_{x\in[a,b]} u_0(x)$
2 $\max_{x\in[a,b]} u_0(x) = u_0(x_0) = u(x(0),0) = u(x(\hat{t}),\hat{t}) \leq \max_{x\in[a,b]} u(x,\hat{t}) < max_{x\in[a,b]} u_0(x)$

11.2 Central Difference Schemes for LTE:

$$\frac{U_j^{n+1} - U_j^n}{\Delta t} + a \cdot \frac{U_{j+1}^n - U_{j-1}^n}{2\Delta x} = 0, \quad \forall j = 0,1,\dots,N-1$$

This scheme is *unconditionally unstable*, since the energy grows at every time step for any choice of $\Delta x, \Delta t$. The physical explanation is that information (for $a > 0$) goes from left to right, while the CFD scheme takes information from the left and the right.

11.3 Upwind Scheme for LTE:

$$\frac{U_j^{n+1} - U_j^n}{\Delta t} + a^+ \frac{U_j^n - U_{j-1}^n}{\Delta x} + a^- \frac{U_{j+1}^n - U_j^n}{\Delta x} = 0$$

$$\frac{U_j^{n+1} - U_j^n}{\Delta t} + a \cdot \frac{U_{j+1}^n - U_{j-1}^n}{2\Delta x} = \frac{|a|}{2\Delta x}(U_{j+1}^n - 2U_j^n + U_{j-1}^n)$$

Solutions with upwind scheme are conditionally stable with $\max|a(x,t)| \cdot \frac{\Delta t}{\Delta x} \leq 1$

- if $a == x$ in the scheme $a \rightarrow x_j$.
- $a^+ = \max\{a,0\}$, $a^- = \min\{a,0\}$, $\Delta x = \frac{b-a}{N+1}$, $N = \#$ interior points.

Example: Membrane LTE

$$\begin{cases} u_t + a u_x = cu & \text{(with } a,c \in \mathbb{R}\text{)} \\ u(x,0) = u_0(x) \end{cases}$$

Modify the upwind scheme to include the right hand side.

$$\frac{U_j^{n+1} - U_j^n}{\Delta t} + a \cdot \frac{U_{j+1}^n - U_{j-1}^n}{2\Delta x} = \frac{|a|}{2\Delta x}(U_{j+1}^n - 2U_j^n + U_{j-1}^n) + c \cdot U_j^n$$