NUMERICAL METHODS FOR PARTIAL DIFFERENTIAL EQUATIONS

AY 2013/2014

Lecturers: Marco Discacciati, Esther Sala Lardies

Qualification system:

- 3 sets of exercises
- 2 assignments (homework)
- final exam

Numerical Methods for Partial Differential Equations

Basics

Marked questions (*) have to be handed in for marking.

1*. In 1225, Leonardo of Pisa (also known as Fibonacci) was requested to solve a collection of mathematical problems in order to justify his fame and prestige in the court of Federico II. One of the proposed problems can be formulated as the solution of a third degree polynomial equation

$$f(x) := x^3 + 2x^2 + 10x - 20 = 0 \tag{1}$$

Note that the solution of cubic equations was a extremely difficult problem in the $13th$ century. Here iterative methods are considered for the solution of equation (1).

Compute the unique real root of (1) with 4 iterations of Newton's method with the initial approximation $x^0 = \sqrt[3]{20}$ (which is obtained neglecting the monomials with x and x^2 in front of the monomial with x^3). Plot the convergence graphic. Does Newton's method behave as expected?

---

2. Solve equation

$$f(x) = x - \cos x$$

- $a)$ with interval-halving starting with $x = 0.5$ and 1.0.
- $b)$ with Newton's method. Use $x_0 = 1.0$ as the starting value.
- $c)$ with the secant method. Use $x = 0.5$ and 1.0 as starting values.

---

3. Solve the following system of nonlinear equations

$$(x-1)^2 + (y-2)^2 = 3$$

$$\frac{x^2}{4} + \frac{y^2}{3} = 1$$

using Newton-Raphson's method

---

4. Use Newton divided differences to construct the interpolating polynomial given the set of data:

$$x_0 = 3.5,\ x_1 = 3.6,\ x_2 = 3.7,\ x_3 = 3.8$$

$$f(x_0) = 0.285714,\ f(x_1) = 0.277778,\ f(x_2) = 0.270270,\ f(x_3) = 0.263158$$

5*. We are interested in the definition of third-order numerical quadratures in interval $(0, 1)$

a) Determine the minimum number of integration points, and specify the integration points and weights.

b) Is it possible to obtain a third-order quadrature with the following four integration points: $x_0 = 1/4$, $x_1 = 1/2$, $x_2 = 3/4$ and $x_3 = 1$? If it is possible, compute the corresponding weights; otherwise, justify why not.

---

6. *a*) If $n + 1$ points Gaussian quadrature is used for numerical integration state the order of the polynomial that is integrated exactly.

b) If $n = 2$ is selected for Gaussian quadrature, which (if any) of the following integrals will be integrated exactly?

i) $\int_0^1 \sin x \, dx$ *ii*) $\int_0^1 x^3 \, dx$ *iii*) $\int_0^1 x^4 \, dx$ *iv*) $\int_0^1 x^{5.5} \, dx$

7*. Compute $\int_0^1 12x \, dx$, $\int_0^1 (5x^3 + 2x) \, dx$ by hand calculation using

i) Trapezoidal rule over 2 uniform intervals

ii) Simpson's rule over 2 uniform intervals

Compute the error of both approximations. Are the methods behaving as expected?

---

8. Transform the integral

$$\int_a^b f(x) \, dx$$

such that integration limits are $[-1, 1]$.

---

9. Decide and justify if the following statements are true or false.

a) For $m \leq n$, a polynomial least-squares fitting with degree m and $n + 1$ different data points can always be computed with the normal equations. For $n = m$ the least-squares fitting leads to the pure interpolation polynomial. For $n < m$

$$< f, g >= \sum_{i=0}^{n} f(x_i) g(x_i)$$

is not an scalar product in the space of polynomials $\mathcal{P}^m$, an the matrix of the normal equations becomes singular.

b) A function $f(x)$ is known at $n + 1$ points. For the computation of the integral of $f(x)$ it is preferable to consider a Gauss quadrature instead of a Newton-Cotes quadrature, because the data points are not necessarily equally spaced.

c) The integral

$$I = \int_0^1 \int_0^1 (x^3 + 1) y^3 \, dx \, dy$$

can be exactly computed using Gauss quadratures with 6 integration points.

---

10. Perform the numerical integration of

$$\int_0^1 \int_0^1 (9x^3 + 8x^2)(y^3 + y) \, dx \, dy$$

using Simpson's rule in each direction. Is the approximation behaving as expected?

EXERCISES 1

STUDENT 1

(APROVAT)

PDE

ASSIGNMENT 1

1.

Equation

$$f(x) = x^3 + 2x^2 + 10x - 20 = 0$$

$$f'(x) = 3x^2 + 4x + 10$$

Initial approximation

$$k = 0$$

$$\Downarrow$$

$$x^k = x^0 = \sqrt[3]{20}$$

x^0

Function $f\left(\sqrt[3]{20}\right) = \left(\sqrt[3]{20}\right)^3 + 2\left(\sqrt[3]{20}\right)^2 + 10\sqrt[3]{20} - 20 = 41.88$

Derivative $f'\left(\sqrt[3]{20}\right) = 3\left(\sqrt[3]{20}\right)^2 + 4\sqrt[3]{20} + 10 = 42.96$

x^1

Value $x^1 = x^0 - \dfrac{f\left(x^0\right)}{f'\left(x^0\right)} = \sqrt[3]{20} - \dfrac{41.88}{42.96} = 1.74$ ✓

Function $f(1.74) = (1.74)^3 + 2(1.74)^2 + 10 \cdot 1.74 - 20 = 8.72$

Derivative $f'(1.74) = 3(1.74)^2 + 4 \cdot 1.74 + 10 = 26.04$

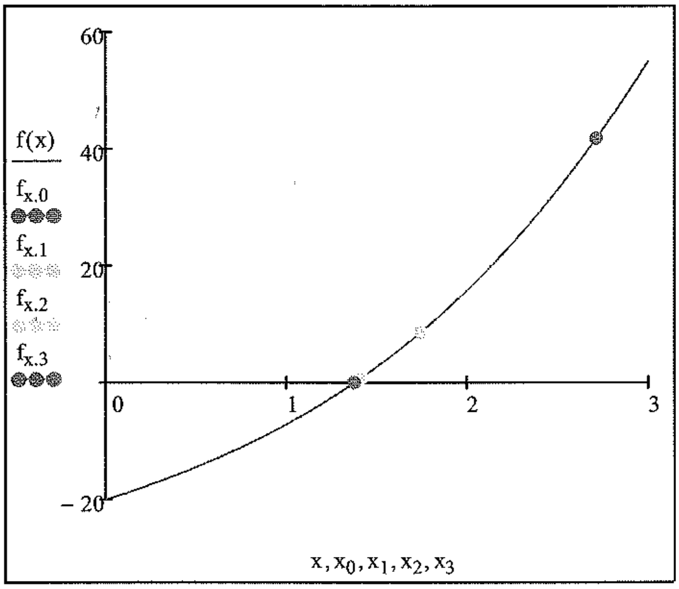
x^2

Value $x^2 = 1.74 - \dfrac{8.72}{26.04} = 1.41$ ✓

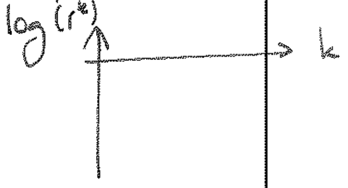
Function $f(1.41) = (1.41)^3 + 2(1.41)^2 + 10 \cdot 1.41 - 20 = 0.88$

Derivative $f'(1.41) = 3(1.41)^2 + 4 \cdot 1.41 + 10 = 21.6$

x^3	Value	$x^3 = 1.41 - \frac{0.88}{21.6} = 1.37$ ✓
	Function	$f(1.37) = (1.37)^3 + 2\,(1.37)^2 + 10 \cdot 1.37 - 20 = 0.03$
	Derivative	$f'(1.37) = 3\,(1.37)^2 + 4 \cdot 1.37 + 10 = 21.11$
x^4	Value	$x^4 = 1.37 - \frac{0.03}{21.11} = 1.37$ ✓



This is not the convergence plot



The root has been found with few iterations by using Newton's Method. The plot shows the analytical solution and the points from Newton's Method.

5.

Assumption	Third-order nummerical quadratures	n = 1
	Interval	[0,1]

a)

Minimum number of integration points $\boxed{n + 1 = 1 + 1 = 2}$ ✓

Point and weights

$$\int_a^b p(z)\,dz = \sum_{i=0}^{n} \left(w_i \cdot p(z_i)\right)$$

for

$p = 1$

$p = z$

$p = z^2$

$p = z^3$

(4 equations and 4 unknowns)

You don't need to solve the system
You can conside the quadrature
in [-1, 1] (which is known)
and use the following change of
variables:

$$x = \frac{1}{2} z + \frac{1}{2}$$

$p = 1$

$$\int_0^1 1\,dz = w_0 \cdot 1 + w_1 \cdot 1$$

$$1 = w_0 + w_1$$

$p = z$

$$\int_0^1 z\,dz = w_0 \cdot z_0 + w_1 \cdot z_1$$

$$\frac{1}{2} = w_0 \cdot z_0 + w_1 \cdot z_1$$

$p = z^2$

$$\int_0^1 z^2\,dz = w_0 \cdot (z_0)^2 + w_1 \cdot (z_1)^2$$

$$\frac{1}{3} = w_0 \cdot (z_0)^2 + w_1 \cdot (z_1)^2$$

$p = z^3$

$$\int_0^1 z^3\,dz = w_0 \cdot (z_0)^3 + w_1 \cdot (z_1)^3$$

$$\frac{1}{4} = w_0 \cdot (z_0)^3 + w_1 \cdot (z_1)^3$$

Solving gives the following values:

$$\boxed{w_0 = \frac{1}{2}}$$

$$\boxed{w_1 = \frac{1}{2}}$$

$$\boxed{z_0 = \frac{1}{2} - \frac{\sqrt{3}}{6} = 0.21}$$ ✓

$$\boxed{z_1 = \frac{\sqrt{3}}{6} + \frac{1}{2} = 0.79}$$

b)

Assumption: Points $x_0 = \frac{1}{4}$, $x_1 = \frac{1}{2}$, $x_2 = \frac{3}{4}$, $x_4 = 1$

Weights

$$\int_a^b p(z)\,dz = \sum_{i=0}^{n} \left(w_i \cdot p(z_i)\right)$$

for

$p = 1$

$p = z$

$p = z^2$

$p = z^3$

(4 equations and 4 unknowns) ✓

$p = 1$

$$\int_0^1 1\,dz = w_0 \cdot 1 + w_1 \cdot 1 + w_2 \cdot 1 + w_3 \cdot 1$$

$$1 = w_0 + w_1 + w_2 + w_3$$

$p = z$

$$\int_0^1 z\,dz = w_0 \cdot \frac{1}{4} + w_1 \cdot \frac{1}{2} + w_2 \cdot \frac{3}{4} + w_3 \cdot 1$$

$$\frac{1}{2} = w_0 \cdot \frac{1}{4} + w_1 \cdot \frac{1}{2} + w_2 \cdot \frac{3}{4} + w_3 \cdot 1$$

$p = z^2$

$$\int_0^1 z^2\,dz = w_0 \cdot \left(\frac{1}{4}\right)^2 + w_1 \cdot \left(\frac{1}{2}\right)^2 + w_2 \cdot \left(\frac{3}{4}\right)^2 + w_3 \cdot 1^2$$

$$\frac{1}{3} = w_0 \cdot \left(\frac{1}{4}\right)^2 + w_1 \cdot \left(\frac{1}{2}\right)^2 + w_2 \cdot \left(\frac{3}{4}\right)^2 + w_3 \cdot 1^2$$

$p = z^3$

$$\int_0^1 z^3\,dz = w_0 \cdot \left(\frac{1}{4}\right)^3 + w_1 \cdot \left(\frac{1}{2}\right)^3 + w_2 \cdot \left(\frac{3}{4}\right)^3 + w_3 \cdot 1^3$$

$$\frac{1}{4} = w_0 \cdot \left(\frac{1}{4}\right)^3 + w_1 \cdot \left(\frac{1}{2}\right)^3 + w_2 \cdot \left(\frac{3}{4}\right)^3 + w_3 \cdot 1^3$$

Solving gives the following values:

$$\boxed{w_0 = \frac{2}{3}}$$

$$\boxed{w_1 = -\frac{1}{3}}$$

$$\boxed{w_2 = \frac{2}{3}}$$

$$\boxed{w_3 = 0}$$

So it it is possible to find the corresponding weights. ✓

7.

Assumption

1) $\int_0^1 12 \cdot x \, dx = 6$

2) $\int_0^1 \left(5 \cdot x^3 + 2x\right) dx = 2.25$

$n = 2 \quad \Rightarrow \quad h = \frac{1-0}{2} = \frac{1}{2}$

Points

$x_0 = 0$

$x_1 = \frac{1}{2}$

$x_2 = 1$

It is assumed that it is a closed quadrature.

1) Trapezoidal rule

Number 1)

$$\int_0^1 12 \cdot x \, dx = 6$$

$$I = \frac{h}{2} \cdot \left(f(x_0) + 2 \cdot f(x_1) + f(x_2)\right) = \frac{\frac{1}{2}}{2} \cdot (0 + 2 \cdot 6 + 12) = 6$$ ✓

There is no error because it is a first order equation. ✓

Number 2)

$$\int_0^1 \left(5\cdot x^3 + 2x\right) dx = 2.25$$

$$I = \frac{h}{2}\cdot\left(f(x_0) + 2\cdot f(x_1) + f(x_2)\right) = \frac{\frac{1}{2}}{2}\cdot\left(0 + 2\cdot\frac{13}{8} + 7\right) = 2.56$$

Error

$$E = 2.56 - 2.25 = 0.31$$

1) Simpson rule

The functions are now separated into two subintervals. A new value h is calculated.

$$n = 2 \qquad \Rightarrow \qquad h = \frac{1-0}{4} = \frac{1}{4}$$

Number 1)

$$\int_0^1 12\cdot x\,dx$$

Subinterval 1

$$I = \frac{h}{3}\cdot\left(f(x_0) + 4\cdot f(x_1) + f(x_2)\right) = \frac{\frac{1}{4}}{3}\cdot(0 + 4\cdot 3 + 6) = 1.5$$

Subinterval 2

$$I = \frac{h}{3}\cdot\left(f(x_2) + 4\cdot f(x_3) + f(x_4)\right) = \frac{\frac{1}{4}}{3}\cdot(6 + 4\cdot 9 + 12) = 4.5$$

$$I = 6$$

There is still no error because it is a first order equation.

Number 2)

$$\int_0^1 \left(5\cdot x^3 + 2x\right) dx$$

Subinterval 1

$$I = \frac{h}{3}\cdot\left(f(x_0) + 4\cdot f(x_1) + f(x_2)\right) = \frac{\frac{1}{4}}{3}\cdot\left(0 + 4\cdot\frac{37}{64} + \frac{13}{8}\right) = \frac{21}{64}$$

Subinterval 2

$$I = \frac{h}{3} \cdot \left(f(x_2) + 4 \cdot f(x_3) + f(x_4)\right) = \frac{\frac{1}{4}}{3} \cdot \left(\frac{13}{8} + 4 \cdot \frac{231}{64} + 7\right) = \frac{123}{64}$$

$I = 2.25$

There is no error because it is a third order equation.

The errors shows that the methods are behaving as expected.

EXERCISES 1
STUDENT 2
(aprovat)

Assignment 1

$f(x) = x^3 + 2x^2 + 10x - 20 = 0$

Which is a third degree polynomial equation

Initial approximation : $x^0 = \sqrt[3]{20}$ (x^k)

$$\boxed{\Delta x^{k+1} = -\frac{f(x^k)}{f'(x^k)}} \qquad \boxed{x^{k+1} = x^k + \Delta x^{k+1}}$$

$f'(x) = 3x^2 + 4x + 10 = 0$

① $f(x^0) = (\sqrt[3]{20})^3 + 2\cdot(\sqrt[3]{20})^2 + 10\cdot\sqrt[3]{20} - 20 = 41,88$

$f'(x^0) = 3(\sqrt[3]{20})^2 + 4\sqrt[3]{20} + 10 = 42,96$

$\Delta x^{0+1} = -\frac{41,88}{42,96} = -0,975$

Then we get $x^1 = x^0 + (-0,975) = \underline{1,74}$ ✓

② $\left.\begin{array}{l} f(x^1) = 8,713 \\ f'(x^1) = 26,037 \end{array}\right\} \Rightarrow \Delta x^{1+1} = -\frac{8,713}{26,037} = -0,335$

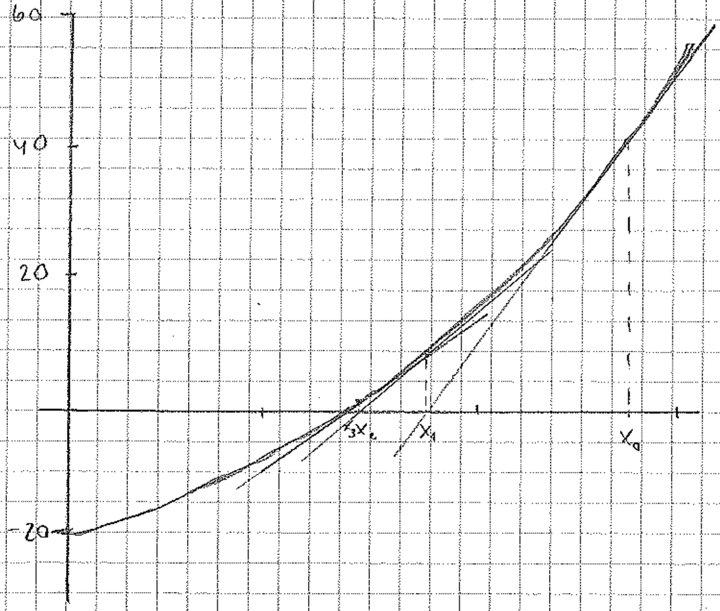
$x^2 = x^1 + (-0,335) = \underline{1,405}$ ✓

③ $\left.\begin{array}{l} f(x^2) = 0,771 \\ f'(x^2) = 21,542 \end{array}\right\} \Rightarrow \Delta x^{2+1} = -\frac{0,771}{21,542} = -0,036 \Rightarrow x^3 = \underline{1,369}$

✓

④ $\left.\begin{array}{l} f(x^3) = 7,9\cdot 10^{-3} \\ f'(x^3) = 21,101 \end{array}\right\} \Rightarrow \Delta x^{3+1} = -\frac{7,9\cdot 10^{-3}}{21,101} = -3,75\cdot 10^{-4} \Rightarrow x^4 = \underline{1,3688}$

✓



The errors are calculated: (E = 1,3688)

$E_1 = \sqrt[3]{20} - 1,3688 = 1,3456$

$E_2 = 1,7394 - 1,3688 = 0,3708$

$E_3 = 1,4050 - 1,3688 = 0,0362$

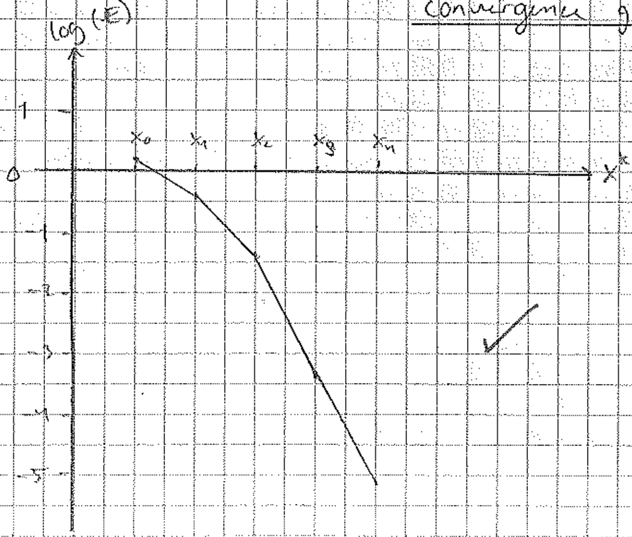
$E_4 = 1,3692 - 1,3688 = 0,0004$

$E_5 = 1,3688 - 1,3688 = 0,000008$

Usually, relative errors are computed:

$$r_k = \left|\frac{x_k - \alpha}{\alpha}\right|$$

Convergence graph



✓

Is the method behaving as expected?

Assignment 5

Third-order numerical quadratures in interval $(0,1) = (a,b)$

Question A:

we need 2 points to make a third-order ~~polynominal~~ quadrature.

To specifi the 2 point and 2 weights i take advantage of two-point Gauss quadrature rule: $\boxed{\int_a^b f(x)\,dx \simeq c_1 f(x_1) + c_2 f(x_2)}$

we want to create a third-order polynominal $f(x)$:

$$f(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3$$

$$\Downarrow$$

$$\int_a^b f(x)\,dx = \int_0^1 \left(a_0 + a_1 x + a_2 x^2 + a_3 x^3\right) dx$$

$$① \quad = a_0 (b-a) + a_1 \left(\frac{b^2 - a^2}{2}\right) + a_2 \left(\frac{b^3 - a^3}{3}\right) + a_3 \left(\frac{b^4 - a^4}{4}\right)$$

Which is the exact value of a third order polynomial.

I now use the two-point Gauss formula

$$\int_0^1 f(x)\,dx \simeq c_1 f(x_1) + c_2 f(x_2)$$

$$② \quad \simeq c_1\left(a_0 + a_1 x_1 + a_2 x_1^2 + a_3 x_1^3\right) + c_2\left(a_0 + a_1 x_2 + a_2 x_2^2 + a_3 x_2^3\right)$$

By seperating the a-terms i get the following:

$$\simeq a_0 (c_1 + c_2) + a_1 (c_1 x_1 + c_2 x_2) + a_2 (c_1 x_1^2 + c_2 x_2^2) + a_3 (c_1 x_1^3 + c_2 x_2^3)$$

I want to get the same results from ② and ①, therefore I put up the following 4 equations (nonlinear):

$$\left.\begin{aligned} c_1 + c_2 &= a - b \\ c_1 x_1 + c_2 x_2 &= \frac{b^2 - a^2}{2} \\ c_1 x_1^2 + c_2 x_2^2 &= \frac{b^3 - a^3}{3} \\ c_1 x_1^3 + c_2 x_2^3 &= \frac{b^4 - a^4}{4} \end{aligned}\right\}$$

By solving this i get the following expression for the points and the weights.

Points

$$x_1 = \frac{b-a}{2}\left(-\frac{1}{\sqrt{3}}\right) + \frac{b+a}{2} = 0{,}211$$

$$x_2 = \frac{b-a}{2}\left(\frac{1}{\sqrt{3}}\right) + \frac{b+a}{2} = 0{,}789$$

weights

$$c_1 = \frac{b-a}{2} = \frac{1}{2}$$

✓

$$c_2 = \frac{b-a}{2} = \frac{1}{2}$$

3/

Question B

We now have four points, $x_0 = 1/4$, $x_1 = 1/2$, $x_2 = 3/4$, $x_3 = 1$

Again I use the expression from question A for a third-order polynomial. $a_0(b-a) + a_1\left(\frac{b^2-a^2}{2}\right) + a_2\left(\frac{b^3-a^3}{3}\right) + a_3\left(\frac{b^4-a^4}{4}\right)$

I now take advantage of gauss n-point rule:

$$\int_a^b f(x)\,dx \cong c_0 f(x_0) + c_1 f(x_1) + c_2 f(x_2) + c_3 f(x_3)$$

Which gives the following expression by seperating the a-terms (same as I did in question A!)

$$\begin{aligned} \cong\ & a_0(c_0 + c_1 + c_2 + c_3) + a_1(c_0x_0 + c_1x_1 + c_2x_2 + c_3x_3) \\ & + a_2(c_0x_0^2 + c_1x_1^2 + c_2x_2^2 + c_3x_3^2) \\ & + a_3(c_0x_0^3 + c_1x_1^3 + c_2x_2^3 + c_3x_3^3) \end{aligned}$$

I now have 4 equations and 4 unknowns (c_0, c_1, c_2, c_3)

$$b - a = c_0 + c_1 + c_2 + c_3$$

$$\frac{b^2-a^2}{2} = c_0x_0 + c_1x_1 + c_2x_2 + c_3x_3$$

$$\frac{b^3-a^3}{3} = c_0x_0^2 + c_1x_1^2 + c_2x_2^2 + c_3x_3^2$$

$$\frac{b^4-a^4}{4} = c_0x_0^3 + c_1x_1^3 + c_2x_2^3 + c_3x_3^3$$

By putting in the known values $(a, b, x_0, x_1, x_2$ and $x_3)$ I get the following four weights:

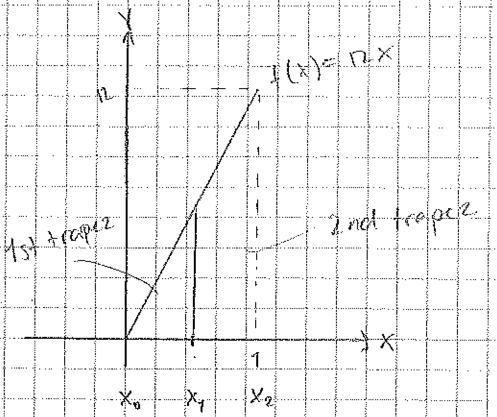
$\underline{\underline{c_0 = 2/3}}$ $\underline{\underline{c_1 = -1/3}}$ $\underline{\underline{c_2 = 2/3}}$ $\underline{\underline{c_3 = 0}}$

✓

4/6

Given integrals : $\int_0^1 12x\,dx$ and $\int_0^1 (5x^3 + 2x)\,dx$

Trapezodal rule for $\int_0^1 12x\,dx$



I have divided the interval
from 0 to 1 into 2 intervals.
This means that each interval has
the length : $\Delta x = \frac{1}{2}$

The trapeziod rule in my case :

$$\int_0^1 12x\,dx \simeq \frac{\Delta x}{2}\left(f(x_0) + 2f(x_1) + f(x_2)\right)$$

$$= \frac{1/2}{2}\left(12\cdot 0 + 2\cdot 12\cdot\tfrac{1}{2} + 12\cdot 1\right)$$

$$= \underline{\underline{6}} \quad ✓$$

$f''(x) = 0 \quad \forall x$

The error is 0, because f(x) dosn't have a 2nd drivative on $[0,1] = [a,b]$!

Trapezodal rule for $\int_0^1 (5x^3 + 2x)\,dx$

use the same formula as shown above

$$\int_0^1 (5x^3 + 2x)\,dx \approx \frac{\Delta x}{2}\left(f(x_0) + 2f(x_1) + f(x_2)\right)$$

$$= \frac{1/2}{2}\left(5\cdot 0^3 + 2\cdot 0 + 2\cdot 5\cdot(1/2)^3 + 2\cdot 2\cdot 1/2 + 5\cdot 1^3 + 2\cdot 1\right)$$

$$= 1/4\left(10/8 + 2 + 7\right) = \cancel{\frac{10\,1/4}{4}} = \frac{82/8}{4} = 2'5625$$

The error is calculated :

$f'(x) = 15x^2 + 2$ and $\underline{f''(x) = 30x}$

The error is given by : $\boxed{E \le \frac{(b-a)^3}{12n^2}\left[\max|f''(x)|\right]}$

$[\max|f''(x)|] = \max|f''(1)| = 30$

Result : $\underline{\underline{I \simeq \frac{10\,1/4}{4} \pm \frac{30}{48}}}$

$$E = \frac{(1-0)^3}{12\cdot 2^2}\cdot 30 = \frac{30}{48}$$

5/6

Simpson rule on $\int_0^1 12x\,dx$

$$\int_0^1 12x\,dx \approx \frac{h}{3}\left(f(x_0)+4f(x_1)+f(x_2)\right)$$

$$= \frac{1/2}{3}\left(12\cdot 0+4\cdot 12\cdot \tfrac{1}{2}+12\cdot 1\right)$$

$$= \frac{1/2}{3}\left(24+12\right)$$

$$= \frac{18}{3} = \underline{\underline{6}} \quad ✓$$

Simpson rule on $\int_0^1 (5x^3+2x)\,dx$

$$\int_0^1 \left(5x^2+2x\right)dx \approx \frac{1/2}{3}\left(f(x_0)+4f(x_1)+f(x_2)\right)$$

$$= \frac{1/2}{3}\left(\cancel{5\cdot 0^3}+\cancel{2\cdot 0}+4\cdot 5\cdot\left(\tfrac{1}{2}\right)^3+4\cdot 2\cdot \tfrac{1}{2}+5\cdot 1^3+2\cdot 1\right)$$

$$= \frac{1/2}{3}\left(\tfrac{20}{8}+4+7\right)$$

$$= \underline{\underline{2{,}25}} \quad ✓$$

The error when using simpsons rule on 3rd-order polyonomial or less, the error will be = 0.
Therefore the above calculations are the exact solutions!

✓

6/6

STUDENT 1

(NOTABLE)

EXERCISES 1

$f(x) := x^3 + 2x^2 + 10x - 20 = 0$

Compute the unique real route using Newton's method with 4 iterations, initial aproximation $x_0 = \sqrt[3]{20}$. Plot the convergence grafic.

Is the method behaving as expected?

Solution:

$f(x_k) = x^3 + 2x^2 + 10x - 20$

$f'(x_k) = 3x^2 + 4x + 10$

$$\Delta x^{k+1} = -\frac{x^3 + 2x^2 + 10x - 20}{3x^2 + 4x + 10}$$

1^{st} *iteration:*

$$\Delta x^1 = -\frac{f(x_0)}{f'(x_0)} = -\frac{(\sqrt[3]{20})^3 + 2(\sqrt[3]{20})^2 + 10\sqrt[3]{20}}{3(\sqrt[3]{20})^2 + 4(\sqrt[3]{20}) + 10} = -0,9748$$

$x^1 = x^0 + \Delta x^1 = 2,7144 - 0,9748 = 1,7396$ ✓

2^{nd} *iteration:*

$f(x^1) = 1,7396^3 + 2 \cdot 1,7396^2 + 10 \cdot 1,7396 - 20 = 0,7708$

$f'(x_1) = 3 \cdot 1,7396^2 + 4 \cdot 1,7396 + 10 = 21,5416$

$$\Delta x^2 = -\frac{f(x^1)}{f'(x^1)} = -0,3346$$

$x^2 = x^1 + \Delta x^2 = 1,4049$ ✓

3^{rd} *iteration:*

$f(x^2) = 1,4049^3 + 2 \cdot 1,4049^2 + 10 \cdot 1,4049 - 20 = 0,0079$

$f'(x_2) = 3 \cdot 1,4049^2 + 4 \cdot 1,4049 + 10 = 21,1007$

$$\Delta x^3 = -\frac{f(x^2)}{f'(x^2)} = -0,0357$$

$x^3 = x^2 + \Delta x^3 = 1,3691$ ✓

4^{th} *iteration:*

$f(x^3) = 1,3691^3 + 2 \cdot 1,3691^2 + 10 \cdot 1,3691 - 20 = 0,0062$

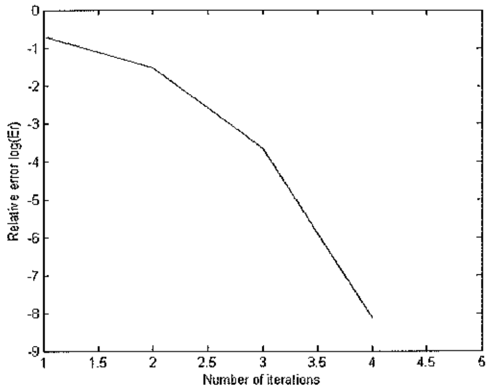
$f'(x_3) = 3 \cdot 1,3691^2 + 4 \cdot 1,3691 + 10 = 21,0997$

$$\Delta x^4 = -\frac{f(x^3)}{f'(x^3)} = -0,0003$$ ✓

$x^4 = x^3 + \Delta x^4 = 1,3688$

The *exact solution* of the problem (obtained with Wolfram Alpha) is $f(x) = 0, forx_e =$ 1,3688.

	x^0	x^1	x^2	x^3	x^4
values	2,7144	1,7396	1,4049	1,3691	1,3688
E_r	0,4957	0,2132	0,0257	$3 \cdot 10^{-4}$	0



✓

---

2 Exercise 5

We are interested in the definition of third-order numerical quadratures in interval (0; 1)

a) Determine the minimum number of integration points, and specify the integration points and weights.

b) Is it possible to obtain a third-order quadrature with the following four integration points: $x_0 = {}^1/_4$, $x_1 = {}^1/_2$, $x_2 = {}^3/_4$, $x_3 = 1$? If it is possible, compute the corresponding weights; otherwise, justify why not.

Solution:

a)

The minimum number of integration points can be obtained using the Gauss-Legendre method, as it results a quadrature of order q^{2n+}.

$n = 1 \longrightarrow 2n+1 = 3 \longrightarrow 3^{rd}$ order

We have to apply a variable change to move the integration limits from (0; 1) to (-1; 1):

$x = \frac{b-a}{2}z + \frac{a+b}{2} = \boxed{\frac{1}{2}z + \frac{1}{2}}$; $F(x) = F\left(\frac{1}{2}z + \frac{1}{2}\right) = f(z)$; $\mathrm{d}x = \frac{b-a}{2}\mathrm{d}z = \boxed{\frac{1}{2}\mathrm{d}z}$;

$I = \int_0^1 F(x)\mathrm{d}x = \frac{1}{2}\int_{-1}^{1} F\left(\frac{1}{2}z + \frac{1}{2}\right)\mathrm{d}z$

→ Don't need to solve the system

- $p = 1; \longrightarrow w_0 \cdot 1 + w_1 \cdot 1 = \frac{1}{2}\int_{-1}^{1}\left(\frac{1}{2} \cdot 1 + \frac{1}{2}\right)\mathrm{d}z = \frac{1}{2}z\Big|_{-1}^{1} = 1;$
- $p = z; \longrightarrow w_0 z_0 + w_1 z_1 = \frac{1}{2}\int_{-1}^{1}\left(\frac{1}{2} \cdot z + \frac{1}{2}\right)\mathrm{d}z = \frac{1}{4}\frac{z^2}{2}\Big|_{-1}^{1} + \frac{1}{4}z\Big|_{-1}^{1} = \frac{1}{2};$
- $p = z^2; \longrightarrow w_0 z_0^2 + w_1 z_1^2 = \frac{1}{2}\int_{-1}^{1}\left(\frac{1}{2} \cdot z^2 + \frac{1}{2}\right)\mathrm{d}z = \frac{1}{4}\frac{z^3}{3}\Big|_{-1}^{1} + \frac{1}{4}\frac{z^2}{2}\Big|_{-1}^{1} = \frac{1}{3};$
- $p = z^3; \longrightarrow w_0 z_0^3 + w_1 z_1^3 = \frac{1}{2}\int_{-1}^{1}\left(\frac{1}{2} \cdot z^3 + \frac{1}{2}\right)\mathrm{d}z = \frac{1}{4}\frac{z^4}{4}\Big|_{-1}^{1} + \frac{1}{4}\frac{z^3}{3}\Big|_{-1}^{1} = \frac{1}{4};$

The equation system is the following:
(1): $w_0 + w_1 = 1$
(2): $w_0 z_0 + w_1 z_1 = \frac{1}{2}$
(3): $w_0 z_0^2 + w_1 z_1^2 = \frac{1}{3}$
(4): $w_0 z_0^3 + w_1 z_1^3 = \frac{1}{4}$
In order to solve the system we operate the following way:
(1)*: (1)·x-(2): $b(x-y) = x - \frac{1}{2}$
(1)**: (1)·x^2-(3): $b(x^2-y^2) = x^2 - \frac{1}{3}$

$$\frac{(1)*}{(1)**} = \frac{b(x-y)}{b(x^2-y^2)} = \frac{2(3x^2-1)}{3(2x-1)} \longrightarrow \boxed{y = \frac{3x-2}{6x-3}}$$

(3)*: (3)·y-(4): $a(x^2y - x^3) = \frac{y}{3} - \frac{1}{4}$ (2)*: (2)·y^2-(4): $a(y^2x - x^3) = \frac{y^2}{2} - \frac{1}{4}$

$$\frac{(3)*}{(4)*} = \boxed{\frac{x^2y - x^3}{y^2x - x^3} = \frac{4y-3}{6y^2-3}}$$

Now in a much more simplified form we can solve the equation using regular algebra, obtaining:

w_0	w_1	z_0	z_1
0,211	0,789	0,5	0,5

✓

We can easily interpret the results:
0.211&0.789 $\in (0;1)$
b)

Is it possible to obtain a third-order quadrature with the following four integration points: $x_0 =^1 /_4$, $x_1 =^1 /_2$, $x_2 =^3 /_4$, $x_3 = 1$? If it is possible, compute the corresponding weights; otherwise, justify why not.

It is not possible to integrate the function $F(x)$ over the interval (0; 1) using these predefined points. The interval is (0; 1) $x_0 =^1 /_4$, so it indicates an *open quadrature* while $x_3 = 1$ (in the endpoint of the interval) indicating a *closed quadrature*!

✗

For a closed quadrature:
$x_0 = 0, x_3 = 1$;
For an open quadrature:
$x_0 = a + h = 0 + \frac{1}{4} = \frac{1}{4}$ which is correct;
$x_3 = b - h = 1 - \frac{1}{4} = \frac{3}{4}$ which is different from 1;
Note: $F(x)$ is only integrable over (0;1) if we neglect $x_3 = 1$ or add $x_0^* = 0$

You can build a 3rd order quadrature from 4 points, even if they are not evenly spaced

3 Exercise 7

Compute $\int_0^1 12x\mathrm{d}x$, $\int_0^1 5x^3 + 2x\mathrm{d}x$ by hand calculation using:

Solution:

We compute the exact solutions and the derivatives for the 2 integrals of $f^1(x) = 12x$ $f^2(x) = 5x^3 + 2x$:

$\int_0^1 12x\mathrm{d}x = 12\frac{x^2}{2}\Big|_0^1 = 6$

$\int_0^1 5x^3 + 2x\mathrm{d}x = 5\frac{x^4}{4}\Big|_0^1 + 2\frac{x^2}{2}\Big|_0^1 = 2,25$

$f^{1^{2)}}(x) = 0$; $f^{1^{4)}}(x) = 0$

$f^{2^{2)}}(x) = 30x$; $f^{2^{4)}}(x) = 0$

1. Trapezoidal rule over 2 uniform intervals

   We split the interval $(0; 1) \longrightarrow \left(0; \frac{1}{2}\right) \cup \left(\frac{1}{2}; 1\right)$ $h = \frac{1}{2}$

   (a) $f^1(0) = 0$; $f^1(\frac{1}{2}) = 6$; $f^1(1) = 12$;
   $\mathrm{I} = \int_0^1 f^1(x)\mathrm{d}x = \int_0^{\frac{1}{2}} f^1(x)\mathrm{d}x + \int_{\frac{1}{2}}^1 f^1(x)\mathrm{d}x = \frac{h}{2}\left[f^1(0) + 2\cdot f^1(\frac{1}{2}) + f^1(1)\right] = \boxed{6}$ ✓; ✓
   There is no error since $f^{1^{2)}}(x) = 0$;

   (b) $f^2(0) = 0$; $f^2(\frac{1}{2}) = \frac{13}{8}$; $f^2(1) = 7$;
   $Er = -\frac{2h^3}{12}\left(f^{2^{2)}}(\frac{1}{2}) + f^{2^{2)}}(1)\right) = 0.46875$
   $\mathrm{I} = \int_0^1 f^2(x)\mathrm{d}x = \int_0^{\frac{1}{2}} f^2\cancel{(x)\mathrm{d}}x + \int_{\frac{1}{2}}^1 f^2(x)\mathrm{d}x$?
   $\simeq \frac{h}{2}\left[f^2(0) + 2\cdot f^2(\frac{1}{2}) + f^2(1)\right] - Er$
   $= 2.5625 - 0.46875 = \boxed{2.09375}$ ×
   The quadtrature obtained is 3^{rd} order, to numerically integrate $f^2(x)$ with good precision.

   $I = I_T + E$

   $I_T =$ [illegible] $2'5625$

2. Simpson's rule over 2 uniform intervals - we add auxiliary points at $\frac{1}{4}$, $\frac{3}{4} \longrightarrow h = \frac{1}{4}$

   We split the interval $(0; 1) \longrightarrow \left(0; \frac{1}{2}\right) \cup \left(\frac{1}{2}; 1\right)$ $h = \frac{1}{2}$

   (a) $f^1(0) = 0$; $f^1(\frac{1}{4}) = 3$; $f^1(\frac{1}{2}) = 6$; $f^1(\frac{3}{4}) = 9$; $f^1(1) = 12$;
   $\mathrm{I} = \frac{h}{3}\left[f^1(0) + 4f^1(\frac{1}{4}) + 2f^1(\frac{1}{2}) + 4f^1(\frac{3}{4}) + f^1(1)\right] = \frac{1}{12}72 = \boxed{6}$ ✓ ✓

   (b) $f^2(0) = 0$; $f^2(\frac{1}{4}) = 0,\cancel{578}1$; $f^2(\frac{1}{2}) = 1,6250$; $f^2(\frac{3}{4}) = 3,6094$; $f^2(1) = 7$;
   $\mathrm{I} \simeq \frac{h}{3}\left[f^2(0) + 4f^2(\frac{1}{4}) \cancel{+} 2f^2(\frac{1}{2}) + 4f^2(\frac{3}{4}) + f^2(1)\right] = \boxed{2{,}2457}$ ✓ ×
   As we increased the number of points and the order of the quadtrature the results got better.

   $I_S = 2'25$

EXERCISES 1

STUDENT 2

(NOTABLE)

Exercise 1 (1/1)

$f(X) = X^3 + 2X^2 + 10X - 20;$

$f'(X) = 3X^2 + 4X + 10;$

$X^0 = \sqrt[3]{20} = 2{'}7144$

$$\Delta X^{k+1} = -\frac{X^3 + 2X^2 + 10X - 20}{3X^2 + 4X + 10};$$

$X^{K+1} = X^K + \Delta X^{K+1};$

K=0; n=1

$X^1 = X^0 + \Delta X^1;$

$$\Delta X^1 = -\frac{(\sqrt[3]{20})^3 + 2(\sqrt[3]{20})^2 + 10\sqrt[3]{20} - 20}{3(\sqrt[3]{20})^2 + 4(\sqrt[3]{20}) + 10} = -\frac{41{'}8803}{42{'}9618} = -0{'}9748;$$

$X^1 = 2{'}7144 - 0{'}9748 = 1{'}7396;$ ✓

$f(X^1) = 8{'}7128;$

$f'(X^1) = 26{'}0370;$

K=1; n=2

$$X^2 = X^1 + \Delta X^2 = 1{'}7396 - \frac{8{'}7128}{26{'}0370} = 1{'}4049;$$ ✓

$f(X^2) = 0{'}7694;$

$f'(X^2) = 21{'}5408;$

K=2; n=3

$$X^3 = X^2 + \Delta X^3 = 1{'}4049 - \frac{0{'}7694}{21{'}5408} = 1{'}3692;$$ ✓

$f(X^3) = 0{'}0082;$

$f'(X^3) = 21{'}1009;$

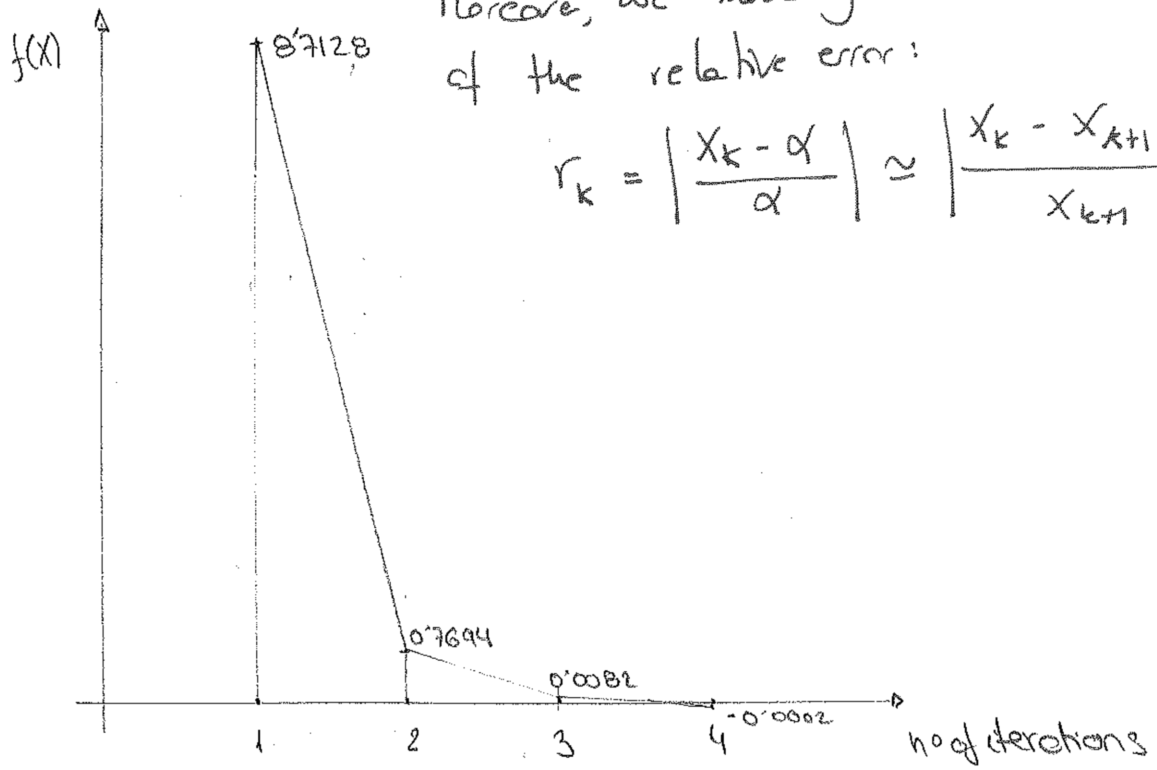
K = 3 ; n=4

$x^4 = x^3 - \Delta x^4 = 1'3692 - \frac{0'0082}{21'1009} = 1'3688;$

$f(x^4) = -0'0002;$
$f'(x^4) = 21'0960;$

In a convergence plot, the logarithm of the error (not just the error) is displayed.
Moreove, we usually show the evolution of the relative error:

$$r_k = \left| \frac{x_k - \alpha}{\alpha} \right| \simeq \left| \frac{x_k - x_{k+1}}{x_{k+1}} \right|$$



Is the method behaving as expected?

Exercise 5 (1/1)

a) Gauss-Legendre quadrature

$2n+1 = 3 \to n=1 \to$ 2 points (x_0, x_1)

$$\int_a^b p(z)dz = \sum_{i=0}^{n} w_i p(z_i);$$

$$p_0(z) = 1 \longrightarrow w_0 \cdot 1 + w_1 \cdot 1 = \int_0^1 1dz;$$
$$p_1(z) = z \longrightarrow w_0 \cdot z_0 + w_1 \cdot z_1 = \int_0^1 z dz;$$
$$p_2(z) = z^2 \longrightarrow w_0 z_0^2 + w_1 \cdot z_1^2 = \int_0^1 z^2 dz;$$
$$p_3(z) = z^3 \longrightarrow w_0 \cdot z_0^3 + w_1 \cdot z_1^3 = \int_0^1 z^3 dz;$$

$w_0 + w_1 = 1;$
$w_0 z_0 + w_1 z_1 = 1/2;$
$w_0 z_0^2 + w_1 \cdot z_1^2 = 1/3;$
$w_0 \cdot z_0^3 + w_1 \cdot z_1^3 = 1/4;$

Solving this nonlinear system of equations the solution is:

z_0 = 0'2113 $\to$ w_0 = 0'5000
z_1 = 0'7886 $\to$ w_1 = 0'5000 ✓

These values for w_0, w_1, z_0, z_1 could also be computed those already obtained for the interval $[-1,1]$ as follows: ✓

$[-1,1] \to w_0', w_1', z_0', z_1'$

$$w_0 = \frac{b-a}{2} w_0' = \frac{1-0}{2} \cdot w_0' = \frac{1}{2} w_0' = \frac{1}{2} \cdot 1 = 1/2;$$

$$w_1 = \frac{b-a}{2} w_1' = \frac{1-0}{2} w_1' = \frac{1}{2} w_1' = \frac{1}{2} \cdot 1 = 1/2;$$

$$z_0 = \frac{b-a}{2} z_0' + \frac{a+b}{2} = \frac{1}{2} z_0' + \frac{1}{2} = -\frac{\sqrt{3}}{6} + \frac{1}{2} = 0'2113;$$

$$z_1 = \frac{b-a}{2} z_1' + \frac{a+b}{2} = \frac{1}{2} z_1' + \frac{1}{2} = \frac{\sqrt{3}}{6} + \frac{1}{2} = 0'7886;$$

b) Since we are interested in the definition of a third-order numerical quadrature in the interval $(0,1)$, if we want to obtain it using four equally spaced points, these should be:

$X_0 = 0.$
$X_1 = 0 + 1/3 = 1/3.$
$X_2 = 0 + 2 \cdot 1/3 = 2/3.$
$X_3 = 0 + 3 \cdot 1/3 = 3/3 = 1.$

However, the proposed points are $X_0 = 1/4$, $X_1 = 1/2$, $X_2 = 3/4$ and $X_3 = 1$ which do not consider the origin of the interval.

Still, these 4 points can be used to obtain a third order quadrature.

Solving the system of equations, we obtain

$w_0 = 2/3$ $\quad w_1 = -1/3$ $\quad w_2 = 2/3$ $\quad w_3 = 0$

Exercise 7 (1/1)

$$\int_0^1 12x\,dx.$$

- Trapezoidal rule over 2 uniform intervals ($m=2$).

$x_0 = 0$; $f(x_0) = 0$;
$x_1 = 1/2$; $f(x_1) = 6$;
$x_2 = 1$; $f(x_2) = 12$;
$m = 2$; $h_1 = h_2 = 1/2$;

$$I = \sum_{i=1}^{m} \frac{h_i}{2}\left(f(x_{i-1}) + f(x_i)\right) - \frac{h_i^3}{12} f''(\mu).$$

$f''(x) = 0 \rightarrow f''(\mu) = 0$;

$$I = \frac{1}{4}(0+6) + \frac{1}{4}(12+6) = \frac{6}{4} + \frac{18}{4} = \frac{24}{4} = \underline{\underline{6}}$$ ✓

Error $= 0$.

- Simpson's rule over 2 uniform intervals ($m=2$).

$x_0 = 0$;
$x_1 = 1/4$;
$x_2 = 1/2$;
$x_3 = 3/4$;
$x_4 = 1$;
$h_i = 1/4, i = 1{:}4$

$$I = \frac{h}{3} \sum_{i=1}^{2} \left(f(x_{2i-2}) + 4f(x_{2i-1}) + f(x_{2i}) \right) - \frac{mh^5}{90} f^{IV}(\mu).$$

$f^{IV}(x) = 0 \rightarrow f^{IV}(\mu) = 0.$

$$I = \frac{h}{3}\left[f(x_0) + 4f(x_1) + f(x_2) + f(x_2) + 4f(x_3) + f(x_4) \right];$$

$$I = \frac{h}{3}\left(0 + 4\cdot 12\cdot 1/4 + 2\cdot 12\cdot 1/2 + 4\cdot 12\cdot 3/4 + 12\cdot 1 \right) = \underline{\underline{6}}$$ ✓

Error $= 0$.

$$\int_0^1 (5x^3+2x)\,dx$$

▨ Trapezoidal rule over two uniform intervals.

$x_0 = 0 \rightarrow f(x_0) = 0;$
$x_1 = 1/2 \rightarrow f(x_1) = 13/8;$
$x_2 = 1 \rightarrow f(x_2) = 7;$

$I = \frac{h}{2}\left[f(x_0) + 2f(x_1) + f(x_2)\right] + E_m^T; \quad h = 1/2;$

$I = \frac{1}{4}(2 \cdot 13/8 + 7) + E_m^T = 2'5625 + E_m^T;$

$E_m^T = 2'5625 - 2'25 = 0'3125;$ ✓

▨ Simpson's rule over two uniform intervals. (q=3).

$m = 2; \quad h = \frac{b-a}{2m} = 1/4$

$x_0 = 0 \longrightarrow f(x_0) = 0;$
$x_1 = 1/4 \longrightarrow f(x_1) = 37/64;$
$x_2 = 1/2 \longrightarrow f(x_2) = 13/8;$
$x_3 = 3/4 \longrightarrow f(x_3) = 231/64;$
$x_4 = 1 \longrightarrow f(x_4) = 7;$

$E_m^S = -\frac{mh^5}{90} f^{IV}(\mu); \quad f^{IV}(x) = 0 \rightarrow f^{IV}(\mu) = 0.$ ✓

$I = \frac{1}{3}\cdot\frac{1}{4}\left(f(x_0) + 4f(x_1) + f(x_2) + f(x_2) + 4f(x_3) + f(x_4)\right);$

$I = \frac{1}{12}\left(0 + 4\cdot\frac{37}{64} + \frac{13}{8}\cdot 2 + 4\cdot\frac{231}{64} + 7\right) = \frac{432}{192} = \underline{\underline{2'25}}$ ✓

Exercise Basics

EXERCISES 1
STUDENT 1 (excel.lent)

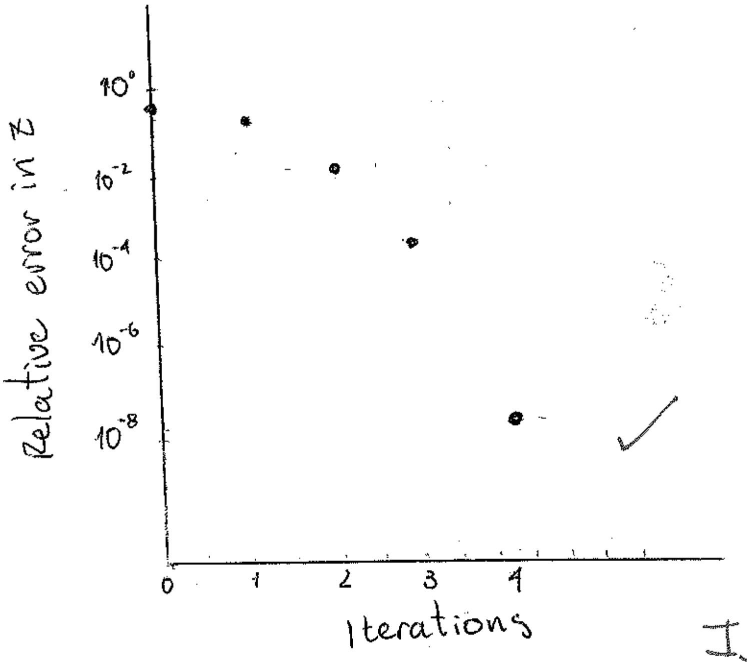
1.

$$f(x) := x^3 + 2x^2 + 10x - 20 = 0$$

$$f'(x) = 3x^2 + 4x + 10$$

$$x^{k+1} = x^k - \frac{f(x^k)}{f'(x^k)}$$

k	x^k	$f(x^k)$	$f'(x^k)$	x^{k+1}
0	$\sqrt[3]{20}$	41,8803	42.9619	1,73959
1	1,73959	8.71261	26.0369	1.40497
2	1,40497	0,77085	21.6417	1.36918
3.	1.36918	0.007912	21.1007	1.36881
4.	1.36881 ✓	$8.587 \cdot 10^{-7}$	21,0961	1.36881 ✓



Relative error

$$\frac{|z^k - z^{k+1}|}{|z^{k+1}|}$$

Values ?

Is the method behaving as expected ?

2. Interval $(0,1)$

a) Gauss-Legendee quadrature

$$2n+1=3$$
$$n=1$$

2 integration points

Integration points and weights are set by:

$$\int_0^1 f_j(x)\,dx = \sum_{i=0}^{1} w_i f_j(x_i) \qquad \text{with} \quad f_j(x) = \begin{cases} 1 \\ x \\ x^2 \\ x^3 \end{cases}$$

=>

$$1 = w_0 + w_1$$
$$\frac{1}{2} = w_0 x_0 + w_1 x_1$$
$$\frac{1}{3} = w_0 x_0^2 + w_1 x_1^2$$
$$\frac{1}{4} = w_0 x_0^3 + w_i x_1^3$$

$$x_i = \frac{1}{2} \pm \frac{\sqrt{3}}{6}$$
$$w_i = \frac{1}{2}$$ ✓

or using Gauss-Legendre points and weights for $(-1,1)$ and setting them for $(0,1)$ by:

$$w_i = \frac{b-a}{2}\,\bar{w}_i \,, \quad x_i = \frac{b-a}{2}\,\bar{x}_i + \frac{a+b}{2}$$

with $(a,b) = (0,1)$, and $\bar{w}_i = 1$ $\quad \bar{x}_i = \pm\sqrt{3}/3$ ✓

b) $x_0 = \frac{1}{4}$ $\quad x_2 = \frac{3}{4}$

$x_1 = \frac{1}{2}$ $\quad x_3 = 1$

Using the same equation:

$$\int_0^1 f_j(x)\,dx = \sum_{i=0}^{3} w_i f_j(x_i) \quad \text{for} \quad j = 0,3$$

We find the weights

$$w_0 = \frac{2}{3} \quad w_1 = -\frac{1}{3} \quad w_2 = \frac{2}{3} \quad w_4 = 0$$ ✓

which is the same set of equations that Newton-Cotes open rule for $n=2$ of order 3, using $h = \frac{1}{4}$

3.

$\int_0^1 12x\,dx$ $\qquad f(x) = 12x$

i) $(a,b) = (0,1)$ $\qquad x_0 = 0$

$m = 2$ $\qquad x_1 = \frac{1}{2}$

$h = \frac{b-a}{m} = \frac{1}{2}$ $\qquad x_2 = 1$

$$\Rightarrow \int_0^1 12x\,dx \simeq \frac{h}{2}\left(f(x_0) + 2f(x_1) + f(x_2)\right) \checkmark$$

$$\simeq 6$$

$$E_m^T = -\frac{(b-a)^3}{12m^2} f^{(2)}(\mu) = 0, \qquad f^{(2)} = 0$$

ii) Simpson's rule

$h = (b-a)/2m = \frac{1}{4}$ $\longrightarrow$ $x_0 = 0 \quad x_1 = \frac{1}{4} \quad x_2 = \frac{1}{2} \quad x_3 = \frac{3}{4} \quad x_4 = 1$

$$\int_0^1 12x\,dx \simeq \frac{h}{3} \sum_{i=1}^{2} \left(f(x_{2i-2}) + 4f(x_{2i-1}) + f(x_{2i})\right)$$

$$\simeq \frac{h}{3}\left(f(x_0) + 4f(x_1) + f(x_2) + f(x_2) + 4f(x_3) + f(x_4)\right)$$

$$\simeq 6 \checkmark$$

$$E_{2m}^S = -\frac{mh^5}{90} f^{(4)}(\mu) = 0 \qquad f^{(4)} = 0$$

Both methods approximate the integral to the exact value, because the function order is less or equal to the method.

$$\int_0^1 (5x^3+2x)\,dx$$

i) Trapezoidal rule

$(a,b) = (0,1)$ $\qquad x_0 = 0$

$m = 2$ $\qquad x_1 = \frac{1}{2}$

$h = \frac{1}{2}$ $\qquad x_2 = 1$

$$\int_0^1 (5x^3+2x)\,dx \approx \frac{h}{2}\left(f(x_0) + 2f(x_1) + f(x_2)\right)$$

$$\simeq 2{,}3125 \qquad 2'5625$$

$$\int_0^1 (5x^3+2x)\,dx = \left(5\frac{x^4}{4} + x^2\right)_0^1 = 2{,}25$$ ✓

$$\Rightarrow E_2^T = -0.0625 = 2.3125 - 2.25$$

ii) Simpson's rule

$h = \frac{1}{4}$ $\qquad x_0 = 0 \qquad x_2 = \frac{1}{2} \qquad x_4 = 1$

$\qquad x_1 = \frac{1}{4} \qquad x_3 = \frac{3}{4}$

$$\int_0^1 (5x^3+2x)\,dx \simeq \frac{h}{3}\left(f(x_0) + 4f(x_1) + 2f(x_2) + 4f(x_3) + f(x_4)\right)$$

$$\simeq 2.25$$

$E_2^S = 0 \qquad f^{(4)} = 0$

In the trapezoidal rule is expected to have some error, cause the polynomial function is higher order than 1. The simpson's rule approaches the integral without error, because for Newton-Cotes quadratures of even n (Simpson is n=2), the order of approximation is n+1, which is the same of the function.

EXERCISES 1

STUDENT 2

(EXCEL·LENT)

4/11/13

NUMERICAL METHODS FOR PDE
HOMEWORK 1

Exercice 1

In 1225, Leonardo of Pisa (also known as Fibonacci) was requested to solve a collection of mathematical problems in order to justify his fame and prestige in the court of Federico II. One of the proposed problems can be formulated as the solution of a third degree polynomial equation

$$f(x) := x^3 + 2x^2 + 10x - 20 = 0 \tag{1}$$

Note that the solution of cubic equation was extremely difficult problem in the 13*th* century. Here iterative methods are considered for the solution of equation 1.

Compute the unique real root of 1 with 4 iteration of Newton's method with the initial approximation $x^0 = \sqrt[3]{20}$ (which is obtained neglecting the monomials with x and x^2 in front of the monomial with x^3). Plot the convergence graphic. Does Newton's method behave as expected?

First of all it's needed to compute the derivative of 1:

$$f'(x) = 3x^2 + 4x + 10 = 0, \tag{2}$$

knowing that $x^0 = \sqrt[3]{20}$ and that the $k+1$ approximation using Newton's method is $x^{k+1} = \frac{-f(x^k)}{f'(x^k)} + x^k$, the first 4 iterations can be computed as:

$$x^1 = -\frac{(\sqrt[3]{20})^3 + 2(\sqrt[3]{20})^2 + 10(\sqrt[3]{20}) - 20}{3(\sqrt[3]{20})^2 + 4(\sqrt[3]{20}) + 10} + \sqrt[3]{20} = \frac{-41.88}{42.962} + \sqrt[3]{20} = 1.7396, \tag{3}$$

✓

$$x^2 = -\frac{(1.7396)^3 + 2(1.7396)^2 + 10(1.7396) - 20}{3(1.7396)^2 + 4(1.7396) + 10} + 1.7396 = \frac{-8.7128}{26.0370} + 1.7396 = 1.4050, \tag{4}$$

✓

$$x^3 = -\frac{(1.4050)^3 + 2(1.4050)^2 + 10(1.4050) - 20}{3(1.4050)^2 + 4(1.4050) + 10} + 1.4050 = \frac{-0.77156}{21.5428} + 1.4050 = 1.3692, \tag{5}$$

✓

$$x^4 = -\frac{(1.3692)^3 + 2(1.3692)^2 + 10(1.3692) - 20}{3(1.3692)^2 + 4(1.3692) + 10} + 1.3692 = \frac{-8.27 \cdot 10^{-3}}{21.10093} + 1.3692 = 1.3688. \tag{6}$$

✓

In order to plot convergence graphic, first we need to compute the residuals. For that case, since the slop of 1 is very high, it's recommended to compute them using $r^k = |f(x^k)|$, therefore:

$$r^0 = f(\sqrt[3]{20}) = 41.88, \tag{7}$$

$$r^1 = f(1.7396) = 8.7128, \tag{8}$$

$$r^2 = f(1.4050) = 0.77156, \tag{9}$$

$$r^3 = f(1.3692) = 8.27 \cdot 10^{-3}, \tag{10}$$

$$r^4 = f(1.3688) = 1.17 \cdot 10^{-4}, \tag{11}$$

↑

so the plot is:

This is not the relative error

$$r^k = \left|\frac{x^k - \alpha}{\alpha}\right| \simeq \left|\frac{x^k - x^{k+1}}{x^{k+1}}\right|$$

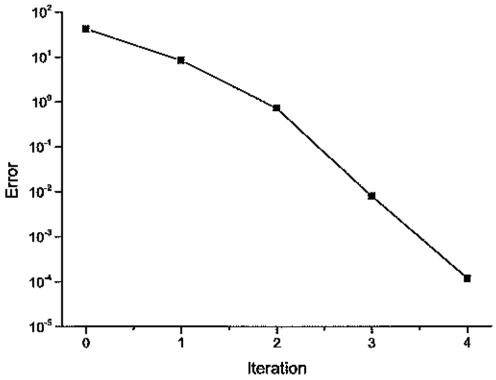


Figure 1: Convergence graphic of the approximation of 1.

The method behaves as expected, analyzing the convergence plot we realize that there are two zones. At the beginning it has slower a convergence, but the last three residuals shows –as we can expect of this method– a quadratic convergence. Your results do not show quadratic convergence.–

$r^{k+1} \leqslant \lambda |r^k|^2$

Exercice 5

We are interested in the definition of third-order numerical quadratures in interval (0,1)

1. Determine the minimum number of integration points, and specify the integration points and weights.
2. Is it possible to obtain a third-order quadratures with the following four integration points: $x_0 = 1/4$, $x_1 = 1/2$, $x_2 = 3/4$, and $x_3 = 1$? If it is possible, compute the corresponding weights; otherwise, justify why not.

1.

In order to get a $q = 3$ quadrature it's possible to use Newton-Cotes method of $n = 2$ (3 points), or Gauss-Legendre method of $n = 1$ (2 points). So the minimum number of points needed to obtain a third-order numerical quadrature is two by using Gauss-Legendre method.

In order to get the points and the weights, exact solutions for $p_0 \cdots p_{2n+2}$ polynomials are imposed:

$$p = 1 \to \int_0^1 1dz = w_0 1 + w_1 1 = 1, \tag{12}$$

$$p = z \to \int_0^1 zdz = w_0 z_0 + w_1 z_1 = \left[\frac{z^2}{2}\right]_0^1 = \frac{1}{2}, \tag{13}$$

$$p = z^2 \to \int_0^1 z^2 dz = w_0 z_0^2 + w_1 z_1^2 = \left[\frac{z^3}{3}\right]_0^1 = \frac{1}{3}, \tag{14}$$

$$p = z^3 \to \int_0^1 z^3 dz = w_0 z_0^3 + w_1 z_1^3 = \left[\frac{z^4}{4}\right]_0^1 = \frac{1}{4}, \tag{15}$$

(16)

so the following nonlinear system of equations is obtained:

$$\begin{cases} w_0 + w_1 = 1 \\ w_0 z_0 + w_1 z_1 = \frac{1}{2} \\ w_0 z_0^2 + w_1 z_1^2 = \frac{1}{3} \\ w_0 z_0^3 + w_1 z_1^3 = \frac{1}{4} \end{cases}. \tag{17}$$

You don't need to solve the system. Consider the quadrature in $[-1,1]$ and perform the change of variable

$x = \frac{1}{2} z + \frac{1}{2}$

The solutions of 17 are $w_0 = w_1 = \frac{1}{2}$, $z_0 = 0.211$ and $z_1 = 0.7887$. The same results can be obtained using the general formula for a $n = 1$ Gauss-Legendre in the interval (a, b):

$$I \cong \frac{a-b}{2} \sum_{i=0}^{n} w_i F\left(\frac{b-a}{2} z_i + \frac{a+b}{2}\right), \tag{18}$$

where for that particular case: $a = 0$, $b = 1$, $w_i = 1$, $z_0 = \frac{\sqrt{3}}{3}$ and $z_0 = \frac{-\sqrt{3}}{3}$.

2.

Since points are given, we should use Newton-Cotes, but given points are not equally spaced in the interval $(0, 1)$ neither for a open Newton-Cotes nor for a closed Newton-Cotes. Therefore is not possible to use them directly.

Notice that points $x_0 = 1/4$, $x_1 = 1/2$ and $x_2 = 3/4$ are equally spaced in the interval $(0, 1)$ for a $n = 2$ open Newton-Cotes which gives us a third-order numerical quadrature. So we can impose $w_3 = 0$ and w_0, w_1 and w_2 will be the ones given by the general formula:

$$I \cong \frac{4h}{3} \left(2f(x_0) - f(x_1) + 2f(x_2)\right) \tag{19}$$

where for that particular case $h = \frac{1}{4}$. At the end of the day, we get that weights are: $w_0 = \frac{2}{3}$, $w_1 = \frac{-1}{3}$ $w_2 = \frac{2}{3}$ and $w_3 = 0$.

Exercice 7

Compute $\int_0^1 12x dx$, $\int_0^1 (5x^3 + 2x) dx$ by hand calculation using

1. Trapezoidal rule over 2 uniform intervals
2. Simpson's rule over 2 uniform intervals

Compute the error of both approximations. Are the methods behaving as expected?

1.

The general formula for Trapezoidal rule over m uniform intervals is:

$$I \cong \frac{h}{2} \left(f(x_0) + 2 \sum_{i=1}^{m-1} f(x_i) + f(x_m)\right) \tag{20}$$

For $I = \int_0^1 12x dx$, $h = (b-a)/m$ and $m = 2$ (2 uniform intervals), it's known that I can be computed as:

$$I \cong \frac{1}{4} (0 + 2 \cdot 6 + 12) = 6 \tag{21}$$

For $I = \int_0^1 (5x^3 + 2x) dx$, $h = (b-a)/m$ and $m = 2$ (2 uniform intervals), it's known that I can be computed as:

$$I \cong \frac{1}{4} \left(0 + 2\left(\frac{5}{8} + \frac{2}{2}\right) + 7\right) = \frac{41}{16} = 2.5625 \tag{22}$$

2.

The general formula for Simpson's rule over m uniform intervals is:

$$I \cong \frac{h}{3}\sum_{i=1}^{m}\left(f(x_{2i-2})+4f(x_{2i-1})+f(x_{2i})\right) \tag{23}$$

For $I=\int_0^1 12x dx$, $h=(b-a)/2m$ and $m=2$ (2 uniform intervals), it's known that I can be computed as:

$$I \cong \frac{1}{12}\left(0+4\cdot 3+6+6+4\cdot 9+12\right)=6 \tag{24}$$

For $I=\int_0^1 (5x^3+2x)dx$, $h=(b-a)/2m$ and $m=2$ (2 uniform intervals), it's known that I can be computed as:

$$I \cong \frac{1}{12}\left(0+4\frac{37}{64}+\frac{13}{8}+\frac{13}{8}+4\frac{231}{64}+7\right)=\frac{144}{64}=\frac{9}{4}=2.25 \tag{25}$$

Now, in order to compute the errors let's compute both integrals analytically:

$$\int_0^1 12xdx = 12\left[\frac{x^2}{2}\right]_0^1=6 \tag{26}$$

and

$$\int_0^1 (5x^3+2x)dx=\left[5\frac{x^4}{4}+x^2\right]_0^1=\frac{9}{4}=2.25. \tag{27}$$

Therefore the error of 24 is 0 and the one for 25 is 0.3125. These results are the ones that we should expect, because Trapezoidal rule is a first-order quadrature, meaning that it computes exactly, at least, up to first-order polynomial integrals. So its normal that the first integral has been computed exactly and the second with some error.

For the Simpson's rule approximation, both errors are zero. This is its normal behavior, because it's a third-order quadrature so it can compute exactly, at least, up to third-order polynomial integrals.

Numerical Methods for Partial Differential Equations

Ordinary Differential Equations

Starred questions (*) have to be handed in for marking.

1. The motion of a non-frictional pendulum is governed by the Ordinary Differential Equation (ODE)

$$\frac{d^2\theta}{dt^2} + \frac{g}{L}\theta = 0$$

where θ is the angular displacement, $L = 1$ m is the pendulum length and the gravity acceleration is $g = 9.8\ \mathrm{m/s^2}$.

The position and velocity at time $t = 1$ s are known:

$$\theta(1) = 0.4 \text{ rad} \qquad ; \qquad \frac{\mathrm{d}\theta}{\mathrm{dt}}(1) = 0 \text{ rad/s}$$

a) Solve the initial boundary value problem in the interval $(0, 1)$ using a second-order Runge-Kutta method to determine the initial position at $t = 0$ s, with 2 and 4 time steps.

b) Using the approximations obtained in *a)*, compute an approximation of the relative error in the solution computed with 2 steps.

c) Propose a time step h to obtain an approximation with a relative error three orders of magnitude smaller.

---

2. Consider the initial value problem

$$\begin{aligned} \frac{dy}{dx} &= y - x^2 + 1 \qquad x \in (0, 1) \\ y(0) &= 1 \end{aligned}$$

a) Solve the initial value problem using the Euler method with step $h = 0.25$.

b) Compute the solution using the Heun method with a step h such that the computational cost is equivalent to the computational cost in *a)*.

Note that the analytical solution of the initial value problem is a second degree polynomial.

c) Compute the pure interpolation polynomial that fits the results in *b)*.

d) Which approximation criterion do you recommend to fit the results obtained in *a)*? Compute the polynomial approximation with the proposed criterion and compare the results with the polynomial obtained in *c)*.

3*. The ordinary differential equation

$$\frac{dy}{dx} = f(x, y)$$

is defined over the domain (0,1), and is to be solved numerically subject to the initial condition $y(0) = 1$, where $y(x)$ is the exact solution. The forward Euler method for integrating the above differential equation is written as

$$Y_{i+1} = Y_i + hf(x_i, Y_i)$$

where Y_i denotes the discrete solution at node i, with position x_i, of a uniform grid of nodes of constant grid interval size h and $x_{i+1} = x_i + h$.

a) Using a Taylor series expansion, deduce the leading truncation error of the scheme. Is the method consistent? Explain your answer.

b) State the backward Euler method for integrating the above differential equation where $f(x, y)$ is a general non-linear function of x and y.

c) Deduce the stability limits of the respective forward Euler method and backward Euler method for the model equation $dy/dx = -\lambda y$ where λ is a positive real constant.

d) Use the backward Euler method to compute the numerical solution of the ordinary differential equation

$$\frac{dy}{dx} = -25y^{3.5}$$

with initial condition $y(0) = 1$, by hand for two steps with grid interval size $h = 1/10$. (Use 2 Newton iterations per step for this calculation.)

e) Use the forward Euler method to compute the numerical solution of the above ordinary differential equation with same initial condition by hand for two steps with grid interval size h=1/10.

f) The analytical solution is

$$y(x) = \left(\frac{125x + 2}{2}\right)^{-2/5}$$

Using Matlab codes, indicate the maximum stable interval size possible for forward Euler method from the following; h=1/10, h=1/15, h=1/30, h=1/45, h=1/90. How does your choice compare with the stability condition?

4*. The second-order ordinary differential equation

$$\frac{d^2y}{dx} + \omega^2 y = 0$$

is defined over the domain $(0, 1)$, and is to be solved numerically subject to the initial conditions $y(0) = 0$, $dy/dx(0) = \omega$, where $y(x)$ is the exact solution.

a) Reduce the above second order ODE to a system of first order ODEs.

b) Set $\omega^2 = 3$. Using the forward Euler method to integrate the system, compute the solution at $t = 1$ by hand with $n = 4$ steps. Use the Forward Euler code to check your results.

c) Using the Matlab code, compute the solution using $n = 8$ steps. Use these solution values to estimate the step size required to obtain a numerical solution with three significative digits. Try your new step size.

EXERCISES 2
STUDENT 1

(APROVAT)

$\boxed{7}$

PDE

ASSIGNMENT 2

3.

a.

Taylor series

$$Y_{1+i} = Y_i + h\cdot\frac{d}{dx}Y(x_i) + \frac{h^2}{2}\frac{d^2}{dx^2}Y(x_i) + ...$$

Forward Euler

$$Y_{i+1} = Y_i + h\cdot f(x_i, Y_i)$$

Truncation error

$$Y_i + h\cdot\frac{d}{dx}Y(x_i) + \frac{h^2}{2}\frac{d^2}{dx^2}Y(x_i) + ... = Y_i + h\cdot f(x_i, Y_i)$$

$$\Downarrow$$

$$h\frac{d}{dx}Y(x_i) + \frac{h^2}{2}\frac{d^2}{dx^2}Y(x_i) + ... = h\cdot f(x_i, Y_i)$$

(1.5)

$$\Downarrow$$

$$h\left(\frac{d}{dx}Y(x_i) - f(x_i, Y_i)\right) = \frac{h^2}{2}\frac{d^2}{dx^2}Y(x_i) + ...$$

$$\Downarrow$$

$$\left(\frac{d}{dx}Y(x_i) - f(x_i, Y_i)\right) = \frac{h}{2}\frac{d^2}{dx^2}Y(x_i) + ...$$

If the the solution is exact $\frac{d}{dx}Y(x_i) - f(x_i, Y_i) = 0$

✓ Then $\frac{h}{2}\frac{d^2}{dx^2}Y(x_i) + ...$ is the error. The method is consistent because by decreasing h the error will also be decreased.

b.

(1)

Backward Euler ✓

$$Y_{1+i} = Y_i + h\cdot f(x_{1+i}, Y_{i+1})$$

The Backward Euler is a non-linear equation, because both sides depend on Y_{i+1}.
and f is non-linear

The letters denotes the same as in the forward Euler.

c.

Assumption $\frac{d}{dx}y = f(x,y) = -\lambda \cdot y$ and $\lambda > 0$

Stability-forward Euler $\mathbf{Y_{1+i}}$

$$Y_{i+1} . = Y_i + h \cdot f\left(x_i, Y_i\right)$$

$$\Downarrow$$

$$Y_{i+1} . = Y_i - h \cdot \lambda \cdot Y_i$$

$$\Downarrow$$

$$Y_{i+1} . = (1 - h \cdot \lambda) \cdot Y_i$$

$\mathbf{Y_i}$

$$Y_i = (1 - h \cdot \lambda) \cdot Y_{i-1}$$

$\mathbf{Y_{1-i}}$

$$Y_{i-1} . = (1 - h \cdot \lambda) \cdot Y_{i-2}$$

$\mathbf{Y_{1+i}}$

$$Y_{i+1} . = (1 - h \cdot \lambda) \cdot (1 - h \cdot \lambda) \cdot Y_{i-1}$$

$$\Downarrow$$

$$Y_{i+1} . = (1 - h \cdot \lambda) \cdot (1 - h \cdot \lambda) \cdot (1 - h \cdot \lambda) \cdot Y_{i-2}$$

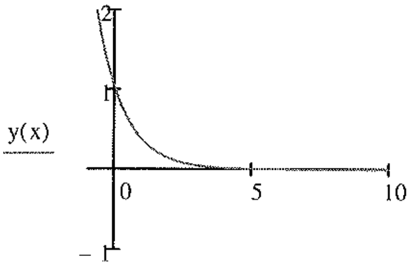
$\mathbf{Y_i}$

$$\boxed{Y_i = (1 - h \cdot \lambda)^i \cdot Y_0}$$

To find the stability conditions for the forward Euler, the following exact solution is used:

$$y(x) = e^{-\lambda \cdot x}$$

where $\frac{d}{dx}y = -\lambda \cdot y = -\lambda \cdot e^{-\lambda \cdot x}$



$$Y_i \to 0 \quad \text{for} \quad \overset{x}{i \to \propto}$$

For the numerical solution to forfill the exact solution, the following requirements fo h must be fulfilled.

$$|1 - h\cdot\lambda| < 1 \quad \Rightarrow \quad 0 < h\cdot\lambda < 2$$

or

$$0 < h < \frac{2}{\lambda}$$ ✓

Stability-backward Euler **Y_{1+i}**

$$Y_{i+1} = Y_i + h\cdot f(x_{1+i}, Y_{i+1})$$

$$\Downarrow$$

$$Y_{i+1} = Y_i - h\cdot\lambda Y_{i+1}$$

$$\Downarrow$$

$$Y_i = (1 + h\cdot\lambda)\cdot Y_{i+1}$$

$$\Downarrow$$

$$Y_{i+1} = \frac{1}{1 + h\cdot\lambda}\cdot Y_i$$

1,5

Y_i

$$\boxed{Y_i = \left[\frac{1}{(1 + h\cdot\lambda)}\right]^i \cdot Y_0}$$

The same exact solution is used in this case.

$$Y_i \to 0 \quad \text{for} \quad i \to \propto$$

$$\frac{1}{1 + h\cdot\lambda} < 1 \quad \Rightarrow \quad \text{no stability conditions}$$ ✓

d.

Assumption $\frac{d}{dx}y = f(x,y) = -25y^{3.5}$

Initial condition $y(0) = 1$

Mesh size $h = \frac{1}{10}$

Solving $Y_{i+1} = Y_i + h \cdot f(x_{i+1}, Y_{i+1})$

Reformulating the equation to

$F(Y) = 0$

$\Downarrow$

$F(Y) = 0 = -Y_i - h \cdot f(x_{i+1}, Y_{i+1}) + Y_{i+1}$

$\Downarrow$

$F(z) = -z_i - h \cdot \left(-25y^{3.5}\right) + z_{i+1}$

Newtons Method

$$z_{kplus1} - z_k = -\frac{F(z_k)}{F'(z_k)}$$

$\mathbf{Y_{0+1}}$

First iteration

$k = 0 \quad \Rightarrow \quad z_0 = Y_{1.z0} = y(0) = 1$

$$F(z_0) = z - 1 + \frac{25}{10} \cdot z^{3.5} = 1 - 1 + \frac{25}{10} \cdot 1^{3.5} = 2.5$$

$$F'(z_0) = 1 + \frac{35}{4} \cdot z^{2.5} = 1 + \frac{35}{4} \cdot 1^{2.5} = 9.75$$

$$z_1 = z_0 - \frac{F(z_0)}{F'(z_0)} = 1 - \frac{2.5}{9.75} = 0.74$$

$z_1 = Y_{1.z1}$

Second iteration

$k = 1$

$$F(z_1) = z - 1 + \frac{25}{10} \cdot z^{3.5} = 0.74 - 1 + \frac{25}{10} \cdot 0.74^{3.5} = 0.61$$

$$F'(z_1) = 1 + \frac{35}{4} \cdot z^{2.5} = 1 + \frac{35}{4} \cdot (0.74)^{2.5} = 5.12$$

$$z_2 = z_1 - \frac{F(z_1)}{F'(z_1)} = 0.74 - \frac{0.61}{5.12} = 0.62$$

$$z_2 = Y_{1.z2} = Y_1$$

Y_{1+1}

First iteration

$k = 0 \qquad \Rightarrow \qquad z_0 = Y_1 = 0.62$ ✓

$$F(0.62) = 0.62 - 0.62 + \frac{25}{10} \cdot 0.62^{3.5} = 0.47$$

$$F'(0.62) = 1 + \frac{35}{4} \cdot 0.62^{2.5} = 3.65$$

$$z_1 = z_0 - \frac{F(z_0)}{F'(z_0)} = 0.62 - \frac{0.47}{3.65} = 0.49$$

$$z_1 = Y_{2.z1}$$

Second iteration

$k = 1$

$$F(0.49) = 0.49 - 0.62 + \frac{25}{10} \cdot 0.49^{3.5} = 0.08$$

$$F'(0.49) = 1 + \frac{35}{4} \cdot 0.49^{2.5} = 2.47$$

②

$$z_2 = 0.49 - \frac{0.08}{2.47} = 0.46$$

✓

$$z_2 = Y_2$$

e.

Assumption $\frac{d}{dx}y = f(x,y) = -25y^{3.5}$

Initial condition $y(0) = 1$

Mesh size $h = \frac{1}{10}$

Forward Euler $\mathbf{Y_{i+1}}$

$Y_{i+1} . = Y_i + h \cdot f(x_i, Y_i)$

$\mathbf{Y_{0+1}}$

Value

$Y_{0+1} . = Y_0 + h \cdot f(x_0, Y_0)$

$\Downarrow$

$Y_1 . = 1 + \frac{1}{10} \cdot \left(-25 \cdot 1^{3.5}\right) = -\frac{3}{2}$

$\mathbf{Y_{1+1}}$

Value

$Y_{1+1} . = Y_1 + h \cdot f(x_1, Y_1)$

$\Downarrow$

$Y_{1+1} . = -1.5 + \frac{1}{10} \cdot \left(-25 \cdot -1.5^{3.5}\right) = 8.83$ N wo

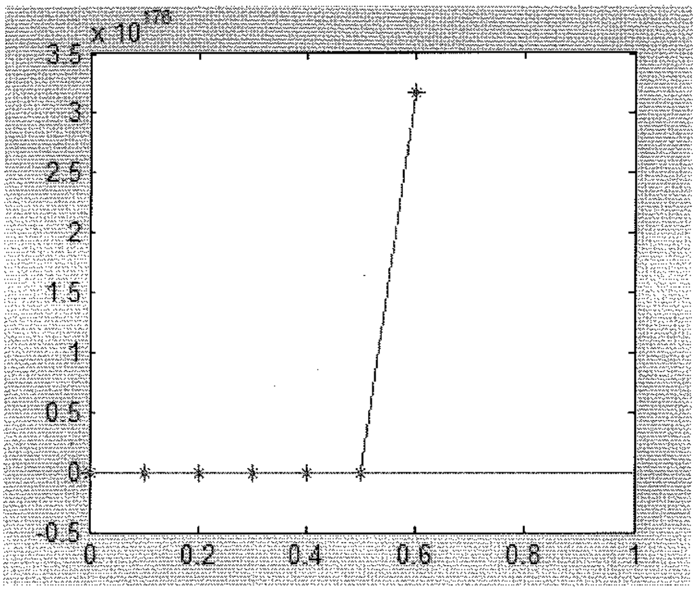
①

f.

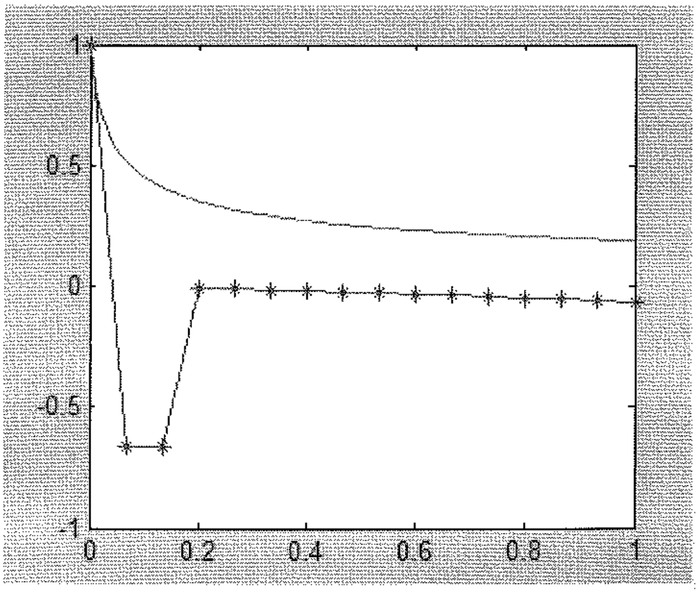
Exact solution $$y(x) = \left(\frac{125 \cdot x + 2}{2}\right)^{-\frac{2}{5}}$$

Matlab The green line denotes the exact solution and the blue line the solution from using Forward Euler in Matlab.

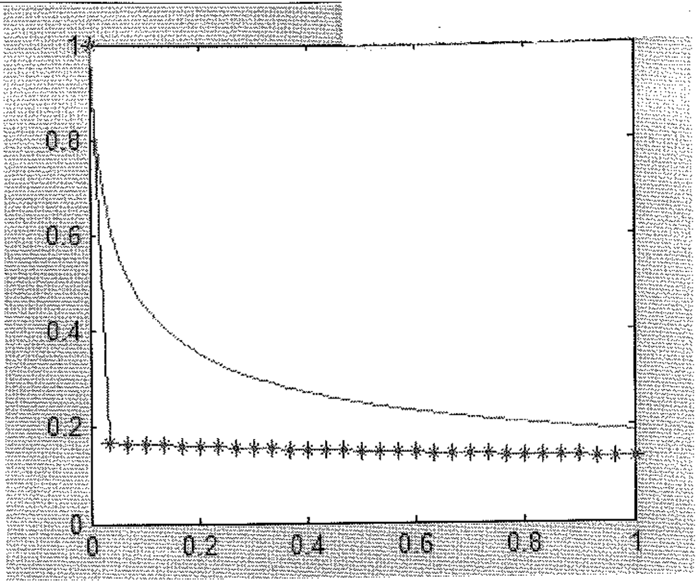
$h = \frac{1}{10}$



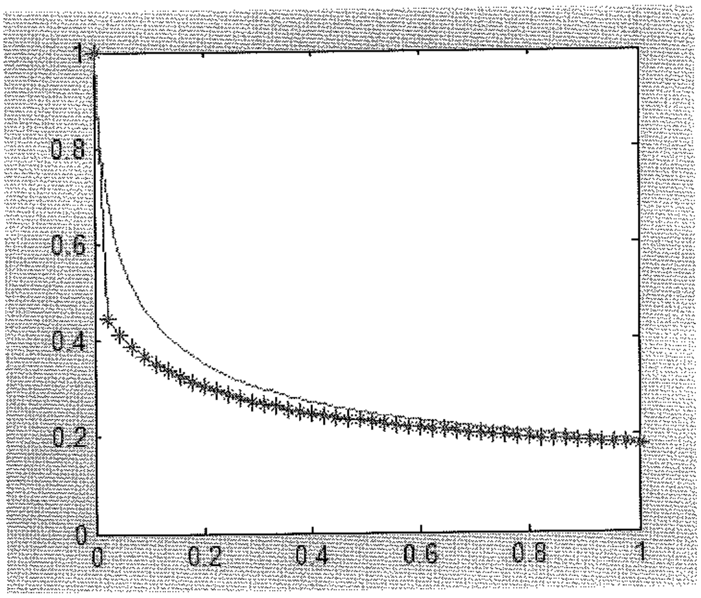
$h = \frac{1}{15}$



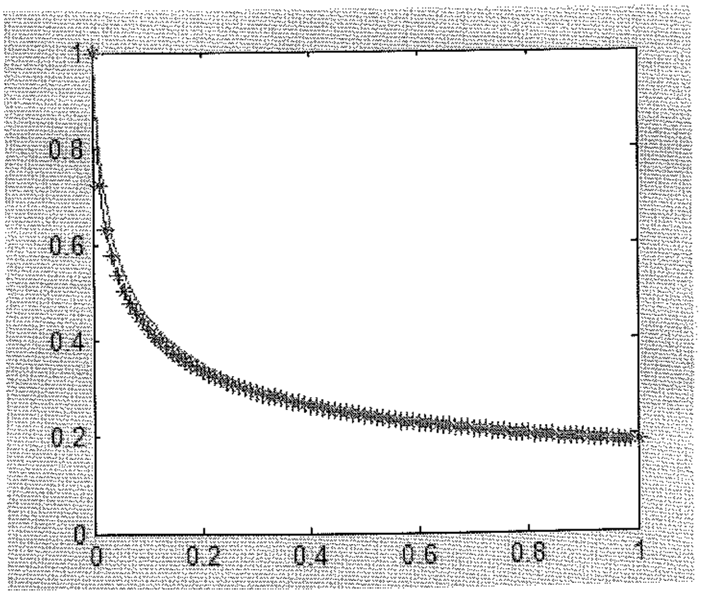
$h = \frac{1}{30}$



$h = \frac{1}{45}$



$h = \frac{1}{90}$



Stability condition

Using the general expression for the stability con

$$-2 < h\cdot\lambda < 0$$

$$-2 < \frac{1}{10}\cdot -25 < 0 \quad\Rightarrow\quad -2 < -2.5 < 0$$ Not stabil

$$-2 < \frac{1}{15}\cdot -25 < 0 \quad\Rightarrow\quad -2 < -1.67 < 0$$ Stabil

linearization

And then the rest should be stabil. Matlab is giving warnings until using $h = \frac{1}{45}$, so maybe it is wrong to use, that stability condition, because it does also look unstabil at the plots, but only until using $h = \frac{1}{30}$.

①

4.

a.

Assumption

$$\frac{d^2}{dx^2}y + \omega^2\cdot y = 0$$

Initial condition $y(0) = 0$

$$\frac{d}{dx}y(0) = \omega$$

y(x) is the exact solution

First order system

$y_{(1;\cdot} = y$ $\qquad y'_{(1;\cdot} = y' = y_{(2)}$

①

$y_{(2;\cdot} = y'$ $\qquad y'_{(2;\cdot} = y'' = -\omega^2\cdot y = -\omega^2\cdot y_{(1)}$

Vectors

$$\vec{Y} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \end{bmatrix} \qquad \frac{d}{dx}\vec{Y} = \begin{bmatrix} y^{(2)} \\ -\omega^2\cdot y^{(1)} \end{bmatrix} \qquad (\vec{Y}) = \begin{pmatrix} 0 \\ \omega \end{pmatrix}$$

✓

The vectors denotes the first order system.

b.

Assumption

$$\omega^2 = 3$$

$$n = 4 \quad\Rightarrow\quad h = \frac{1}{n} = \frac{1}{4}$$

$$i = 0, 1, 2, 3$$

Forward Euler

$\mathbf{Y_{i+1}}$

$$Y_{i+1} := Y_i + h\cdot f(x_i, Y_i)$$

$\mathbf{Y_1}$

$$\overrightarrow{Y_1} = \overrightarrow{Y_0} + h\cdot f\left(x_0, \overrightarrow{Y_0}\right)$$

$$\overrightarrow{Y_1} = \left[\begin{pmatrix} 0 \\ \sqrt{3} \end{pmatrix} + \frac{1}{4}\cdot\begin{pmatrix} \sqrt{3} \\ -3\cdot 0 \end{pmatrix}\right] = \begin{pmatrix} 0.43 \\ 1.73 \end{pmatrix}$$

✓

$\mathbf{Y_2}$

$$\overrightarrow{Y_2} = \overrightarrow{Y_1} + h\cdot f\left(x_1, \overrightarrow{Y_1}\right)$$

$$\overrightarrow{Y_2} = \left[\begin{pmatrix} 0.43 \\ 1.73 \end{pmatrix} + \frac{1}{4}\cdot\begin{pmatrix} 1.73 \\ -3\cdot 0.43 \end{pmatrix}\right] = \begin{pmatrix} 0.86 \\ 1.41 \end{pmatrix}$$

$\mathbf{Y_3}$

②

$$\overrightarrow{Y_3} = \overrightarrow{Y_2} + h\cdot f\left(x_2, \overrightarrow{Y_2}\right)$$

$$\overrightarrow{Y_3} = \left[\begin{pmatrix} 0.86 \\ 1.41 \end{pmatrix} + \frac{1}{4}\cdot\begin{pmatrix} 1.41 \\ -3\cdot 0.86 \end{pmatrix}\right] = \begin{pmatrix} 1.21 \\ 0.76 \end{pmatrix}$$

$\mathbf{Y_4}$

$$\overrightarrow{Y_4} = \overrightarrow{Y_3} + h\cdot f\left(x_3, \overrightarrow{Y_3}\right)$$

$$\overrightarrow{Y_4} = \left[\begin{pmatrix} 1.21 \\ 0.76 \end{pmatrix} + \frac{1}{4}\cdot\begin{pmatrix} 0.76 \\ -3\cdot 1.21 \end{pmatrix}\right] = \begin{pmatrix} 1.4 \\ -0.15 \end{pmatrix}$$

Matlab check

The above results and the results from Matlab are the same.

```
y1 =

          0    0.4330    0.8660    1.2178    1.4073

y2 =

     1.7321    1.7321    1.4073    0.7578   -0.1556
```

✓

⓪

4c) missing

STUDENT 2

EXERCISES 2

(APROVAT)

Joh ild 12/11/13

6.75

Assignment 3

Given the following ODE $\frac{dy}{dx} = f(x,y)$ in $(0;1)$

Initial condition $y(0) = 1$

Question a

The taylor serie is given by: $Y_{i+1} = Y_i + h\frac{d}{dx}Y(x_i) + \frac{h^2}{2}\frac{d^2}{dx}Y(x_i) + O(h^3)$

The forward euler is given by: $Y_{i+1} = Y_i + h\,f(x_i, Y_i)$

The taylor series and euler is set to be equal.

$$\cancel{Y_i} + h\frac{d}{dx}Y(x_i) + \frac{h^2}{2}\frac{d^2}{dx}Y(x_i) + O(h^3) = \cancel{Y_i} + h\,f(x_i, Y_i)$$

$$\Downarrow$$

$$\frac{h^{\cancel{2}}}{2}\frac{d^2}{dx}Y(X_i) + O(h^3) = h\left(f(x_i, Y_i) - \frac{d}{dx}Y(x_i)\right)$$

① The left side of the equation is the truncation error.
And the term in front of the h is equal to zero if the solution is exact.

If $h \to 0$ the truncation error $\to 0$ => the method is consistent! ✓

Question b

✓

① Backward euler: $Y_{i+1} = Y_i + h\,f(x_{i+1}, Y_{i+1})$

Both sides of the equation is depending on Y_{i+1} => the equation is non-linear. (if f is nonlinear)

Question c

Given model equation : $\frac{dy}{dx} = -\lambda y$; λ = positive real constant

Forward euler

$Y_{i+1} = Y_i + h\, f(x_i, Y_i)$

$\Downarrow$

$Y_{i+1} = Y_i + h(-\lambda Y_i)$

$\Downarrow$

$Y_{i+1} = Y_i - h\lambda Y_i$

$\Downarrow$

$Y_{i+1} = G\, Y_i$

$$\left[\frac{dy}{dx} = f(x,y) = -\lambda y\right]$$

$$\left[G = 1 - h\lambda\right]$$

Requirement : $|G| < 1$, because then we are guarenteed that the solution will not grow without bound, and hence it will be stable.

$|1 - \lambda h| < 1$

This gives the following stability limits for forward euler:

$$\underline{\underline{0 < h\lambda < 2}}$$ ✓

$\frac{1}{2}$

Backward euler

$Y_{i+1} = Y_i + h\, f(x_{i+1}, Y_{i+1})$

$Y_{i+1} = Y_i + h(-\lambda Y_{i+1})$

$Y_i = Y_{i+1}(1 + h\lambda)$

$Y_{i+1} = \frac{Y_i}{(1 + h\lambda)}$

$Y_{i+1} = G \cdot Y_i$

$$\left[\frac{dy}{dx} = f(x,y) = -\lambda y\right]$$

$$\left[G = \frac{1}{(1 + h\lambda)}\right]$$

Requirement : $|1 + h\lambda| > 1$

This gives the following stability limits for backward euler: ✓

$$\underline{\underline{\lambda h > 0}}$$

2

Question d

To solve the nonlinear equation that backward euler gives us, we take advantage of newtons method

We have the following:

$\frac{dy}{dx} = -25y^{3{,}5}$, $Y(0) = 1$, $h = 1/10$

Backward euler is given by: $Y_{i+1} = Y_i + h f(x_{i+1}, Y_{i+1})$

I reformulate the nonlinear equation to $F(u) = 0$

Where $u = Y_{i+1}$ and then the nonlinear equation is as followed

$$F(u) = u - Y_i - h f(x_{i+1}, u)$$

I use newtons method for the iterations:

$$u_k = u_{k-1} - \frac{F(u_{k-1})}{F'(u_{k-1})}$$

1st step

I choose now the initial condition: $u_0 = Y(0) = 1$

1st iteration:

$$F(u_0) = u_0 - Y_0 - h \cdot f(x_{i+1}, u_0)$$

$$\Downarrow$$

$$F(u_0) = 1 - 1 - 1/10\left(-25 \cdot 1^{3{,}5}\right) = 2{,}5$$

$$F'(u_0) = 1 + 25/10 \cdot 3{,}5 \cdot u_0^{2{,}5}$$

$$\Downarrow$$

$$F'(u_0) = 1 + 25/10 \cdot 3{,}5 \cdot 1^{2{,}5} = 9{,}75$$

$$u_1 = u_0 - \frac{F(u_0)}{F'(u_0)} \Rightarrow u_1 = 1 - \frac{2{,}5}{9{,}75} = \underline{0{,}74}$$

2nd iteriation

$$F(u_1) = u_1 - Y_0 - h\,(-25\,u_1^{3{,}5}) = 0{,}74 - 1 + {}^1/_{10} \cdot 25 \cdot 0{,}74^{3{,}5} = 0{,}61$$

$$F'(u_1) = 1 + {}^{25}/_{10} \cdot 3{,}5 \cdot u_1^{2{,}5} = 1 + {}^{25}/_{10} \cdot 3{,}5 \cdot 0{,}74^{2{,}5} = 5{,}12$$

$$u_2 = u_1 - \frac{F(u_1)}{F'(u_2)} \Rightarrow u_2 = 0{,}74 - \frac{0{,}61}{5{,}12} = \underline{0{,}62}$$

2nd step

We now have that $Y_1 = u_2 = 0{,}62$

3rd teriation

$$F(u_3) = u_2 - Y_1 - h\,(-25\,u_2^{3{,}5}) = 0{,}62 - 0{,}62 + {}^{25}/_{10} \cdot 0{,}62^{3{,}5} = 0{,}47$$

$$F'(u_3) = 1 + {}^{25}/_{10} \cdot 3{,}5 \cdot u_2^{2{,}5} = 1 + {}^{25}/_{10} \cdot 3{,}5 \cdot 0{,}62^{2{,}5} = 3{,}65$$

$$u_3 = u_2 - \frac{F(u_3)}{F'(u_3)} \Rightarrow u_3 = 0{,}62 - \frac{0{,}47}{3{,}65} = \underline{0{,}49}$$

4th teration

$$F(u_4) = u_3 - Y_1 - h\,(-25\,u_3^{3{,}5}) = 0{,}49 - 0{,}62 - {}^1/_{10}(-25\ 0{,}49^{3{,}5}) = 0{,}08$$

$$F'(u_4) = 1 + {}^{25}/_{10} \cdot 3{,}5 \cdot u_3^{2{,}5} = 1 + {}^{25}/_{10} \cdot 3{,}5 \cdot 0{,}49^{2{,}5} = 2{,}47$$

$$u_4 = u_3 - \frac{F(u_4)}{F'(u_4)} \Rightarrow u_4 = 0{,}49 - \frac{0{,}08}{2{,}47} = \underline{0{,}46}$$

$\underline{\underline{u_2 = Y_1 = 0{,}62}}$ and $\underline{\underline{u_4 = Y_2 = 0{,}46}}$ ✓

②

Question e

Given ODE : $\frac{dy}{dx} = -25y^{3,5}$

With initial condition : $y(0) = 1$ and grid size $h = 1/10$

Forward euler : $Y_{i+1} = Y_i + h\, f(x_i, Y_i)$

We have that $\frac{dy}{dx} = -25y^{3,5} = f(x_i, Y_i)$

1st step

$Y_1 = Y_0 + h\, f(x_0, Y_0)$

$\Downarrow$

$Y_1 = 1 + 1/10\,(-25 \cdot 1^{3,5}) = \underline{\underline{-3/2}}$ ✓

①

2nd step

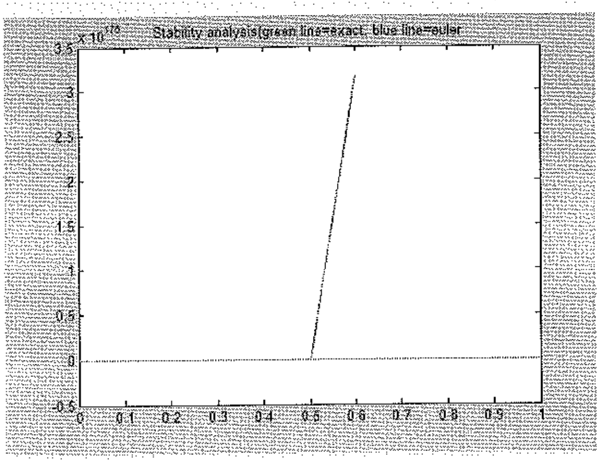
$Y_2 = Y_1 + h\, f(x_1, Y_1)$

$\Downarrow$

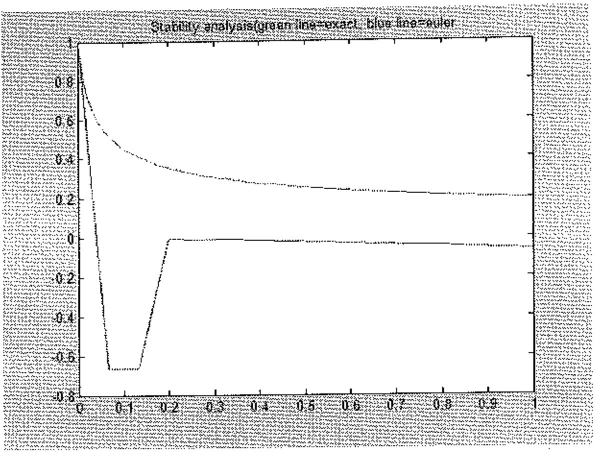
$Y_2 = -3/2 + 1/10\,(-25 \cdot (-3/2)^{3,5}) = \underline{\underline{8,83}}$ WP

Question f

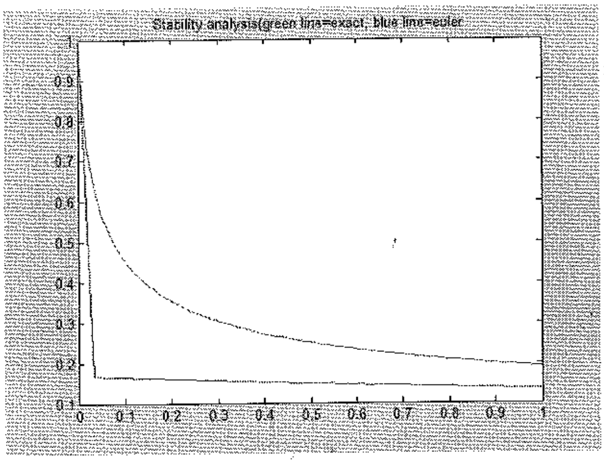
Step size= 1/10



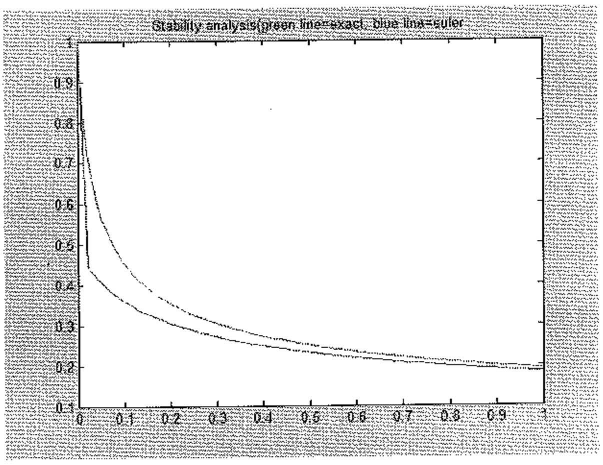
Step size=1/15



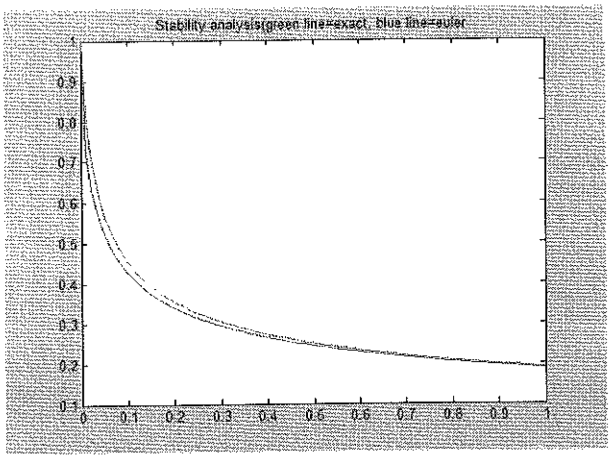
Step size = 1/30



Step size = 1/45



Step size = 1/90



For the generalized formula (from the notes) we have the following

$$0 < h\lambda < -2 \quad \Rightarrow \quad 0 < h < -2/\lambda$$

In our case we have:

$$0 < h < -2/-25$$

$$\Downarrow$$

$$0 < h < 1/12{,}5$$

①

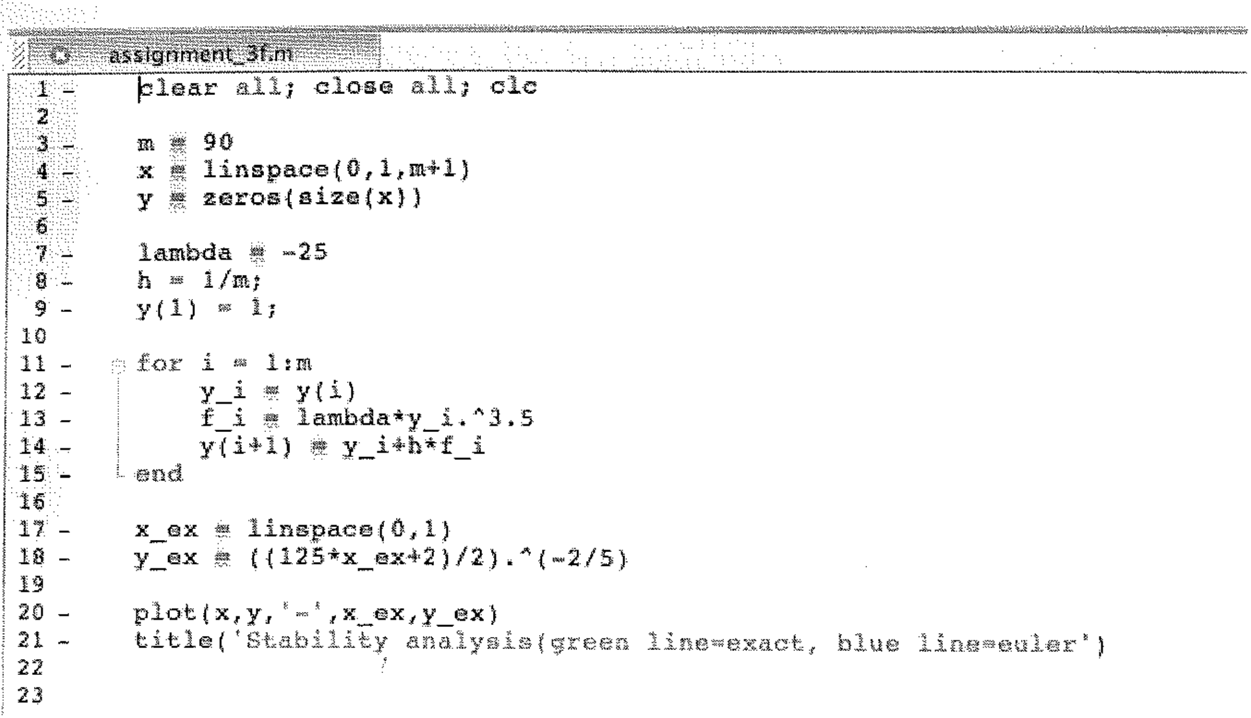
$For\ h = 1/10:\quad 0 < 1/10 < 1/12{,}5$ *Unstable*

$For\ h = 1/15:\quad 0 < 1/15 < 1/12{,}5$ *Stable*

linearization

With stepsize 1/15 the euler varies from the exact solution, though it is stable.
The smaller stepsize, the more exact solution.

Matlab script



```
clear all; close all; clc

m = 90
x = linspace(0,1,m+1)
y = zeros(size(x))

lambda = -25
h = 1/m;
y(1) = 1;

for i = 1:m
    y_i = y(i)
    f_i = lambda*y_i.^3.5
    y(i+1) = y_i+h*f_i
end

x_ex = linspace(0,1)
y_ex = ((125*x_ex+2)/2).^(-2/5)

plot(x,y,'-',x_ex,y_ex)
title('Stability analysis(green line=exact, blue line=euler')


```

Assignment 4

2nd order ODE : $\frac{d^2y}{dx^2} + w^2 y = 0$ $[0;1]$

Initial conditions: $y(0) = 0$ and $\frac{dy}{dx}(0) = w$

Question a

First I introduse 2 new functions:

$Y_1 = Y$

$Y_2 = Y'$

I now convert my 2nd order ODE into two 1st order ODE's

$Y_1' = Y' = Y_2$

$Y_2' = Y'' = -w^2 Y = -w^2 Y_1$ ①

$\vec{Y}' = A \vec{Y}$

$$\Downarrow \quad \vec{Y}' = \begin{bmatrix} 0 & 1 \\ -w^2 & 0 \end{bmatrix} \cdot \begin{bmatrix} Y_1 \\ Y_2 \end{bmatrix} \quad \text{with B.C} \begin{cases} Y_1(0) = 0 \\ Y_2(0) = w \end{cases}$$ ✓

Question b

We now set $w^2 = 3$ and uses the forward euler to integrate the system.

The solution at $t = 1$ over 4 steps $\Rightarrow$ stepsize $= 1/4$

Step 1

$Y(0+h) = Y(0) + h\,F(0, Y(0))$

$$\begin{bmatrix} Y_1(1/4) \\ Y_2(1/4) \end{bmatrix} = \begin{bmatrix} Y_1(0) \\ Y_2(0) \end{bmatrix} + 1/4 \begin{bmatrix} Y_2(0) \\ -w^2 Y_1(0) \end{bmatrix} = \begin{bmatrix} 0 \\ \sqrt{3} \end{bmatrix} + 1/4 \begin{bmatrix} \sqrt{3} \\ -3 \cdot 0 \end{bmatrix} = \begin{bmatrix} \sqrt{3}/4 \\ \sqrt{3} \end{bmatrix}$$

Step 2

$$\begin{bmatrix} Y_1(1/2) \\ Y_2(1/2) \end{bmatrix} = \begin{bmatrix} Y_1(1/4) \\ Y_2(1/4) \end{bmatrix} + 1/4 \begin{bmatrix} Y_2(1/4) \\ -\omega^2 Y_1(1/4) \end{bmatrix} = \begin{bmatrix} \sqrt{3}/4 \\ \sqrt{3} \end{bmatrix} + 1/4 \begin{bmatrix} \sqrt{3} \\ -3 \cdot \sqrt{3}/4 \end{bmatrix} = \underline{\begin{bmatrix} 0{,}866 \\ 1{,}407 \end{bmatrix}}$$

Step 3

$$\begin{bmatrix} Y_1(3/4) \\ Y_2(3/4) \end{bmatrix} = \begin{bmatrix} Y_1(1/2) \\ Y_2(1/2) \end{bmatrix} + 1/4 \begin{bmatrix} Y_2(1/2) \\ -\omega^2 Y_1(1/2) \end{bmatrix} = \begin{bmatrix} 0{,}866 \\ 1{,}407 \end{bmatrix} + 1/4 \begin{bmatrix} 1{,}407 \\ -3 \cdot 0{,}866 \end{bmatrix} = \underline{\begin{bmatrix} 1{,}218 \\ 0{,}758 \end{bmatrix}}$$

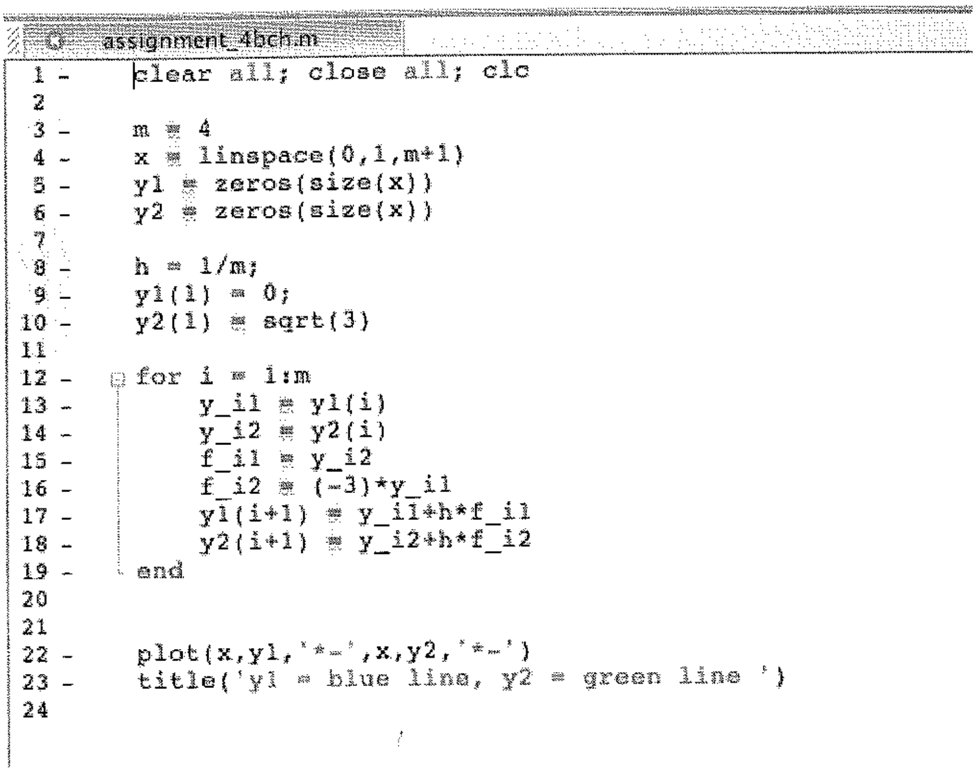
Step 4

$$\begin{bmatrix} Y_1(1) \\ Y_2(1) \end{bmatrix} = \begin{bmatrix} Y_1(3/4) \\ Y_2(3/4) \end{bmatrix} + 1/4 \begin{bmatrix} Y_2(3/4) \\ -\omega^2 Y_1(3/4) \end{bmatrix} = \begin{bmatrix} 1{,}218 \\ 0{,}758 \end{bmatrix} + 1/4 \begin{bmatrix} 0{,}758 \\ -3 \cdot 1{,}218 \end{bmatrix} = \underline{\underline{\begin{bmatrix} 1{,}407 \\ -0{,}155 \end{bmatrix}}}$$

✓

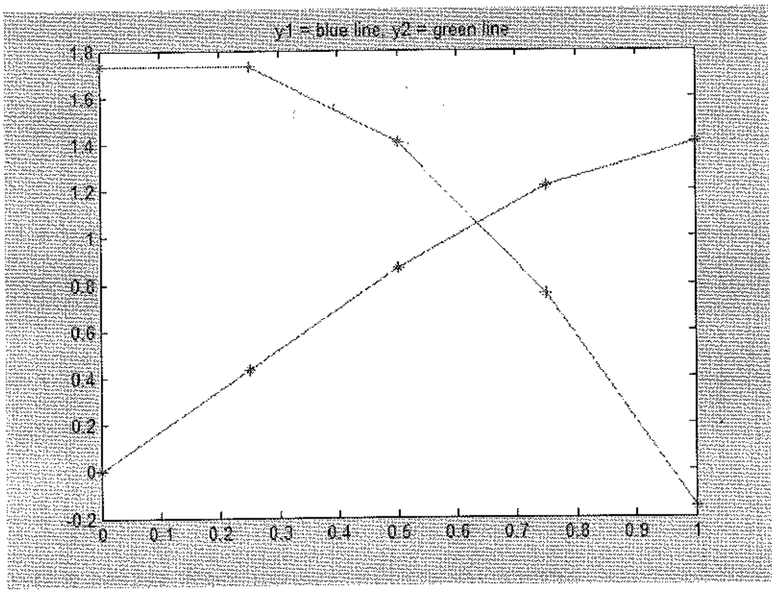
②

Matlab script



```
clear all; close all; clc

m = 4
x = linspace(0,1,m+1)
y1 = zeros(size(x))
y2 = zeros(size(x))

h = 1/m;
y1(1) = 0;
y2(1) = sqrt(3)

for i = 1:m
    y_i1 = y1(i)
    y_i2 = y2(i)
    f_i1 = y_i2
    f_i2 = (-3)*y_i1
    y1(i+1) = y_i1+h*f_i1
    y2(i+1) = y_i2+h*f_i2
end


plot(x,y1,'*-',x,y2,'*-')
title('y1 = blue line, y2 = green line ')

```

Matlab plot



Values computed by matlab

```
y1 =

         0    0.4330    0.8660    1.2178    1.4073


y2 =

    1.7321    1.7321    1.4073    0.7578   -0.1556
```

✓

The soultions from the forward euler matlab code is exactly the same as calculated by hand!

(0)

4c) missing

STUDENT 1

(NOTABLE)

EXERCISES 2

$\boxed{5}$

The ordinary differential equation

$$\tfrac{dy}{dx} = f(x, y)$$

is defined over the domain (0,1), and is to be solved numerically subject to the initial condition $y(0) = 1$, where y(x) is the exact solution. The forward Euler method for integrating the above differential equation is written as:

$$Y_{i+1} = Y_i + hf(x_i, Y_i)$$

where Y_i denotes the discrete solution at node i, with position x_i, of a uniform grid of nodes of constant grid interval size h and $x_{i+1} = xi + h$.

a) Using a Taylor series expansion, deduce the leading truncation error of the scheme. Is the method consistent? Explain your answer.

b) State the backward Euler method for integrating the above differential equation where $f(x; y)$ is a general non-linear function of x and y.

c) Deduce the stability limits of the respective forward Euler method and backward Euler method for the model equation $dy/dx = -\lambda y$ where λ is a positive real constant.

d) Use the backward Euler method to compute the numerical solution of the ordinary differential equation

$$\tfrac{dy}{dx} = -25y^{3,5}$$

with initial condition $y(0) = 1$, by hand for two steps with grid interval size $h = 1/10$. (Use 2 Newton iterations per step for this calculation.)

e) Use the forward Euler method to compute the numerical solution of the above ordinary differential equation with same initial condition by hand for two steps with grid interval size $h = 1/10$.

f) The analytical solution is

$$y(x) = \left(\frac{125x + 2}{2}\right)^{-2/5}$$

Using Matlab codes, indicate the maximum stable interval size possible for forward Euler method from the following; h=1/10, h=1/15, h=1/30, h=1/45, h=1/90. How does your choice compare with the stability condition?

Solution

a) $y_{i+1} = y_i + h\frac{\mathrm{d}y}{\mathrm{d}x}(x_i) + \mathcal{O}(h^2) \longrightarrow Y_{i+1} = Y_i + hf(x_i, Y_i)$

The truncation error is:

$\mathcal{O}(h^2) = \mathcal{O}(\mathcal{R}_i(h)) = \mathcal{O}(mh\mathcal{T}_i(h)) = \mathcal{O}\left(\frac{1}{h}h\mathcal{T}_i(h)\right) = \mathcal{O}(\mathcal{T}_i(h))$

1.5

where $\mathcal{R}_i(h) = y_{i+1} - y_i - hf(x_i, y_i) = h\mathcal{T}_i(h)$

Also: $\mathcal{R}_i(h) = \mathcal{O}(h^{q+1}) = \mathcal{O}(h^2) \longrightarrow q = 1$?

Truncation error $\tau_i(h) = O(h)$

$\lim_{m\to\infty} \mathcal{O}\left(\frac{1}{m}\right) = 0$ ✓consistent

b) $\frac{\mathrm{d}y}{\mathrm{d}x} = f(x, y)$; $\begin{Bmatrix} y_{(1)} \\ y_{(2)} \end{Bmatrix} = \begin{Bmatrix} y \\ \frac{\mathrm{d}y}{\mathrm{d}x} = \frac{\mathrm{d}y_{(1)}}{\mathrm{d}x} \end{Bmatrix}$; $\frac{\mathrm{d}y}{\mathrm{d}x}x_{i+1} = f(x_{i+1}, y_{i+1})$

$y_i = y_{i+1} - h\frac{\mathrm{d}y}{\mathrm{d}x}(x_{i+1}) + \mathcal{O}(h^2) \longrightarrow h\frac{\mathrm{d}y}{\mathrm{d}x}(x_{i+1}) = y_{i+1} - y_i$

$f(x_{i+1}, y_{i+1}) = \frac{y_{i+1} - y_i}{h} + \frac{\mathcal{O}(h^2)}{h} = \frac{y_{i+1} - y_i}{h} + \mathcal{T}(h) \longrightarrow y_{i+1} = hf(x_{i+1}, y_{i+1}) + y_i - h\mathcal{T}(h)$

1

Neglecting the truncation error $\mathcal{T}_i(h)$ we obtain:

✓ $\boxed{Y_{i+1} = hf(x_{i+1}, Y_{i+1}) + Y_i}$ ✓

c) Forward Euler Method:

$\frac{\mathrm{d}y}{\mathrm{d}x} = -\lambda y;\ \lambda \in \mathbb{R}^+$

$y(0) = 1$

$Y_{i+1} = Y_i + hf(x_{i+1}, y_{i+1}) = Y_i - h\lambda Y_i = Y_i(1 - h\lambda)$

$Y_i = Y_{i-1}(1 - h\lambda);\ Y_{i-1} = Y_{i-2}(1 - h\lambda); \longrightarrow Y_i = Y_{i-2}(1 - h\lambda)(1 - h\lambda) = Y_{i-2}(1 - h\lambda)^2$

We can write the following:

$Y_i = Y_{i-i}(1 - h\lambda)^i = Y_0(1 - h\lambda)^i$

where $1 - h\lambda$ is the amplification factor. For the solution to be stable the following must hold:

$|1 - h\lambda| < 1$

1.5

✓ $-2 < -h\lambda < 0 \longrightarrow \boxed{0 < h < \frac{2}{\lambda}}$ ✓

Backward Euler Method:

$Y_{i+1} = -h\lambda Y_{i+1} + Y_i \longrightarrow Y_i = Y_{i+1}(1 + h\lambda) \longrightarrow Y_{i+1} = \frac{Y_i}{1 + h\lambda};\ Y_i = \frac{Y_i - 1}{1 + h\lambda};\ Y_{i-1} = \frac{Y_{i-2}}{1 + h\lambda};$

We can write: $Y_i = \frac{Y_{i-2}}{(1 + h\lambda)^2} \longrightarrow Y_i = \frac{Y_{i-i}}{(1 + h\lambda)^i} \longrightarrow Y_i = \frac{Y_0}{(1 + h\lambda)^i}$

where $1/(1 + h\lambda)$ is the amplification factor. For the solution to be stable:

$\left|\frac{1}{1 + h\lambda}\right| < 1$ which is true for any $h\lambda \geq 0$ so

✓ $\boxed{h \geq 0;\ \lambda \geq 0}$ ✓

d) $Y_{i+1} = Y_i + hf(x_{i+1}, Y_{i+1});\ h = 1/10;\ y(0) = 1;$

Step 1. $Y_1 = Y_0 - \frac{1}{10}25Y_1^{3.5}$

$f(Y_1) = Y_1 + 2.5Y_1^{3.5} - 1$; $f'(Y_1) = 8.75Y_1^{2.5} + 1$; ✓

Newton iterations:

1. $f(0) = -1$; $f'(0) = 1$; $\Delta Y_1^1 = 1$; $Y_1^2 = Y'^1_1 + \Delta Y_1^1 = 0 + 1 = 1$
2. $f(1) = 0.5$; $f'(1) = 9.75$; $\Delta Y_1^2 = -0.051$; $Y_1^3 = Y_1^2 + \Delta Y_1^2 = 1 - 0.051 = 0.949$

2.5

Step 2. $Y_2 = Y_1 - \frac{1}{10}25Y_2^{3/5}$;

$f(Y_2) = Y_2 + 2.5Y_2^{3.5} - 0.949$; $f'(Y_2) = 8.75Y_2^{2.5} + 1$;

Newton iterations:

1. $f(0.949) = 3.03$; $f'(0.949) = 8.676$; $\Delta Y_2^1 = -0.349$;
   $Y_2^2 = 0.949 - 0.349 = 0.6$
2. $f(0.6) = 0.069$; $f'(0.6) = 3.440$; $\Delta Y_2^2 = -0.02$;
   $Y_2^3 = 0.6 - 0.02 = 0.58$

errors but method understood correctly

(1.5)

e) $Y_{i+1} = Y_i + hf(x_i, Y_i)$; $f(x_i; Y_i) = -25Y_i^{3.5}$; $h = 1/10$;

Step 1. $Y_1 = Y_0 + \frac{1}{10}\left(-25 \cdot 0^{3.5}\right) = 1$?

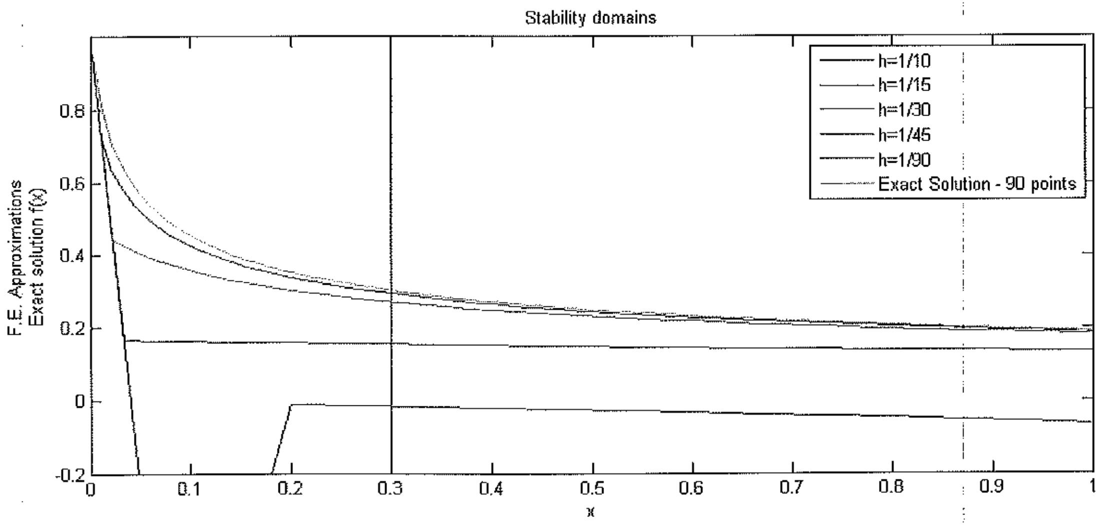
Step 2. $Y_2 = Y_1 + \frac{1}{10}\left(-25 \cdot 1^{3.5}\right) = 1 - 2.5 = -1.5$

no

The initial condition is $y(0) = 1$!

(0.5)

f) We compute the Forward Euler Method for a range of steps $h = [1/10, 1/15, 1/30, 1/45, 1/90]$. All the results are plotted on a same graph, together with the exact solution.



We can observe this method is not stable for step of $h = 1/10$. Also steps $h = 1/15$ and $h = 1/30$ do not provide good approximations.

why? analysis?

(1)

Exercise 4*

The second-order ordinary differential equation

$$\frac{\mathrm{d}^2y}{\mathrm{d}x^2} + \omega^2 y = 0$$

is defined over the domain (0,1), and is to be solved numerically subject to the initial conditions $y(0) = 0, \mathrm{d}y/\mathrm{d}x(0) = \omega$, where $y(x)$ is the exact solution.

a) Reduce the above second order ODE to a system of first order ODEs.

b) Set $\omega^2 = 3$. Using the forward Euler method to integrate the system, compute the solution at $t = 1$ by hand with $n = 4$ steps. Use the Forward Euler code to check your results.

c) Using the Matlab code, compute the solution using $n = 8$ steps. Use these solution values to estimate the step size required to obtain a numerical solution with three significative digits. Try your new step size.

Solution

a) Given the equation $y'' + \omega^{2'} = 0$ we identify the following:

$$\frac{\mathrm{d}^2y}{\mathrm{d}x^2} = -\omega^2 y; \longrightarrow \mathbf{f} = \begin{Bmatrix} y_{(2)} \\ f(x, y_{(1)}, y_{(2)}) \end{Bmatrix}?; \mathbf{y} = \begin{Bmatrix} y_{(1)} \\ y_{(2)} \end{Bmatrix}; \bar{\alpha} = \begin{Bmatrix} y_{(1)}(0) = \alpha_0 \\ y_{(2)}(0) = \alpha_1 \end{Bmatrix};$$

The system of first-order ODE is the following:

0.5

$$\boxed{\begin{array}{l} \dfrac{\mathrm{d}\mathbf{y}}{\mathrm{d}x}(x) = \mathbf{f}(x, \mathbf{y});\ x \in (0,1) \\ \mathbf{y}(a) = \bar{\alpha} \end{array}}$$

Imposing the *Initial Value Conditions* we obtain:

? $\mathbf{f}(0) = \begin{Bmatrix} \omega \\ -\omega^2 y \end{Bmatrix}; \mathbf{y}(0) = \begin{Bmatrix} 0 \\ \omega \end{Bmatrix};$ ✓

b) Apply the Forward Euler method with the following conditions: $\omega^2 = 3$, $n = 4$ steps, $t = 1$;

vector of nodes: $\mathbf{x} = \left[0, \frac{1}{3}, \frac{2}{3}, 1\right]^T$ $n=4 \Rightarrow h = \frac{1}{4}$

①

$$\text{Step 1 } \bar{\mathbf{Y}}_1 = \bar{\mathbf{Y}}_0 + \frac{1}{3}\mathbf{f}\left(0, \bar{\mathbf{Y}}_0\right) \longrightarrow \begin{bmatrix} Y_1^1 \\ Y_2^1 \end{bmatrix} = \begin{bmatrix} 0 \\ \sqrt{3} \end{bmatrix} + \frac{1}{\cancel{3}\,4}\begin{bmatrix} \sqrt{3} \\ 0 \end{bmatrix} \longrightarrow \bar{\mathbf{Y}}_1 = \begin{bmatrix} \frac{\sqrt{3}}{3} \\ \sqrt{3} \end{bmatrix}$$

$$\text{Step 2 } \bar{\mathbf{Y}}_2 = \bar{\mathbf{Y}}_1 + \frac{1}{3}\mathbf{f}\left(\frac{1}{3}, \bar{\mathbf{Y}}_1\right) \longrightarrow \begin{bmatrix} Y_1^2 \\ Y_2^2 \end{bmatrix} = \begin{bmatrix} \frac{\sqrt{3}}{3} \\ \sqrt{3} \end{bmatrix} + \frac{1}{3}\begin{bmatrix} \sqrt{3} \\ \frac{-3\sqrt{3}}{3} \end{bmatrix} \longrightarrow \bar{\mathbf{Y}}_2 = \begin{bmatrix} \frac{2\sqrt{3}}{3} \\ \frac{2\sqrt{3}}{3} \end{bmatrix}$$

$$\text{Step 3 } \bar{\mathbf{Y}}_3 = \bar{\mathbf{Y}}_2 + \frac{1}{3}\mathbf{f}\left(\frac{2}{3}, \bar{\mathbf{Y}}_2\right) \longrightarrow \begin{bmatrix} Y_1^3 \\ Y_2^3 \end{bmatrix} = \begin{bmatrix} \frac{2\sqrt{3}}{3} \\ \frac{2\sqrt{3}}{3} \end{bmatrix} + \frac{1}{3}\begin{bmatrix} \sqrt{3} \\ \frac{-6\sqrt{3}}{3} \end{bmatrix} \longrightarrow \bar{\mathbf{Y}}_3 = \begin{bmatrix} \sqrt{3} \\ 0 \end{bmatrix}$$

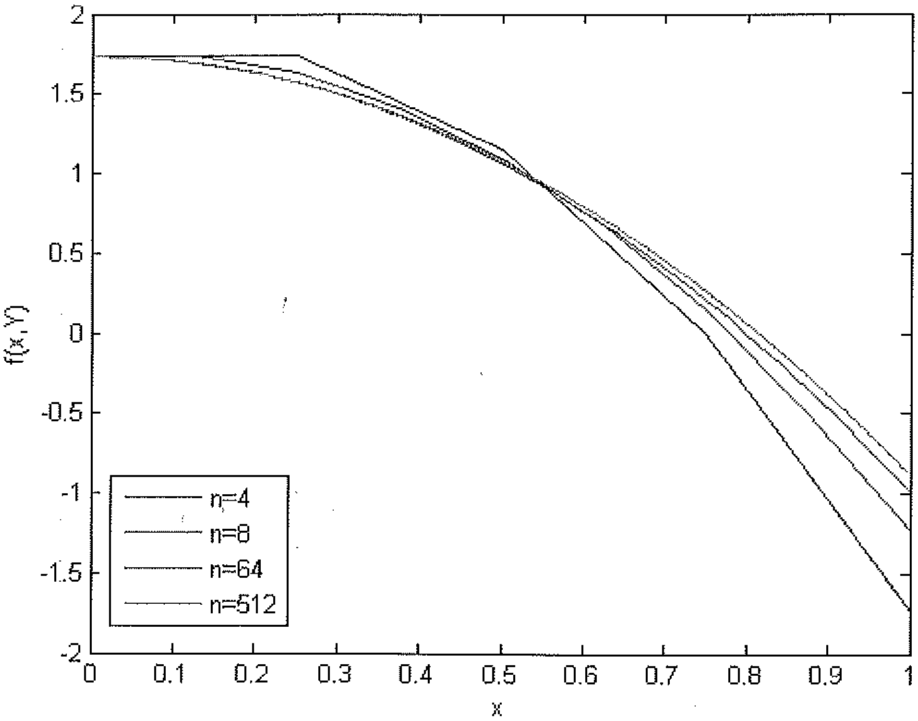
$$\text{Step 4 } \bar{\mathbf{Y}}_4 = \bar{\mathbf{Y}}_3 + \frac{1}{3}\mathbf{f}\left(1, \bar{\mathbf{Y}}_3\right) \longrightarrow \begin{bmatrix} Y_1^4 \\ Y_2^4 \end{bmatrix} = \begin{bmatrix} \sqrt{3} \\ 0 \end{bmatrix} + \frac{1}{3}\begin{bmatrix} \sqrt{3} \\ -3\sqrt{3} \end{bmatrix} \longrightarrow \bar{\mathbf{Y}}_4 = \begin{bmatrix} \frac{4\sqrt{3}}{3} \\ -\sqrt{3} \end{bmatrix}$$

computing error

Calucalted values				
Y_1	0.577	1.154	1.731	2.309
Y_2	1.732	1.155	0	-1.732
Matlab values				
Y_1	0.577	1.154	1.731	2.309
Y_2	1.732	1.155	0	-1.732

errors in implementation!

c) We can observe the convergence of the approximations by the increase of the number of steps n.



this doesn't answer the question!

0

EXERCISES 2

STUDENT 2

(NOTABLE)

Exercise 3

$$\frac{dy}{dx} = f(x,y) \quad x \in [0,1]$$

$$y(0) = 1$$

a) $$f(x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!}(x-a)^n$$

$$y_{i+1} = y_i + h\frac{dy}{dx}(x_i) + \frac{h^2}{2!}\frac{d^2y}{dx}(x_i) + \frac{h^3}{3!}\frac{d^3y}{dx}(x_i) + \dots + \frac{h^n}{n!}\frac{d^ny}{dx^n}(x_i)$$

(1.5)

$$\frac{dy}{dx}(x_i) = \frac{1}{h}\left[y_{i+1} - y_i - \frac{h^2}{2!}\frac{d^2y}{dx}(x_i) - \frac{h^3}{3!}\frac{d^3y}{dx}(x_i) - \dots - \frac{h^n}{n!}\frac{d^ny}{dx^n}(x_i)\right]$$

$$\frac{dy}{dx}(x_i) = \frac{1}{h}(y_{i+1} - y_i) - \sum_{n=2}^{\infty}\frac{h^{n-1}\cdot y^{n)}(x_i)}{n!}$$

$$\frac{y_{i+1} - y_i}{h} - \sum_{n=2}^{\infty}\frac{h^{n-1} y^{n)}(x_i)}{n!} = f(x,y)$$

$$y_{i+1} = y_i + h \cdot f(x_i, y_i) + \underbrace{\boxed{\sum_{n=2}^{\infty}\frac{h^n\, y^{n)}(x_i)}{n!}}}_{\tau_i(h)}$$

✓

$$\lim_{h \to 0} \tau_i(h) = 0$$ → Forward Euler method is consistent

b) $Y_i = Y_{i+1} - h \cdot \frac{dy}{dx}(x_{i+1}) + o(h^2)$.

$$\frac{dy}{dx}(x_{i+1}) = \frac{y_{i+1}-y_i}{h} + o(h).$$

① $$\frac{y_{i+1}-y_i}{h} + o(h) = f(x_{i+1}, y_{i+1})$$

$$y_{i+1} = y_i + h \cdot f(x_{i+1}, y_{i+1}) - o(h^2)$$

$$Y_{i+1} = Y_i + h \cdot f(x_{i+1}, Y_{i+1}).$$ ✓

If $f(x,y)$ is a non-linear function of X_{i+1} and Y_{i+1} the equation $Y_{i+1} = Y_i + hf(X_{i+1}, Y_{i+1})$ is nonlinear aswell.

c) $dy/dx = -\lambda y$. $\lambda > 0$.

Forward Euler method: $Y_{i+1} = Y_i - h\lambda Y_i = (1-h\lambda)Y_i$.

$Y_1 = (1-h\lambda)Y_0$
$Y_2 = (1-h\lambda)Y_1 = (1-h\lambda)^2 Y_0$

$Y_n = (1-h\lambda)^n Y_0$

Since $Y_n \to 0$ when $n \to \infty \to |1-h\lambda| < 1$.

$$\left.\begin{array}{l} 1-h\lambda < 1 \to h\lambda > 0 \\ -1+h\lambda < 1 \to h\lambda > 2 \end{array}\right| \to 0 < h\lambda < 2 \to \boxed{0 < h < \frac{2}{\lambda}}$$

stability condition

✓

Exercise 3

c) Backward Euler method: $Y_{i+1} = Y_i - h\lambda Y_{i+1}$.

$$Y_{i+1}(1+h\lambda) = Y_i \rightarrow Y_{i+1} = \frac{Y_i}{1+h\lambda};$$

$$Y_1 = \frac{Y_0}{1+h\lambda};$$

$$Y_2 = \frac{Y_1}{1+h\lambda} = \frac{Y_0}{(1+h\lambda)^2};$$

$$Y_n = \frac{Y_0}{(1+h\lambda)^n};$$

Since $Y_n \rightarrow 0$ when $n \rightarrow \infty$ $\frac{1}{(1+h\lambda)^n} \rightarrow 0$ when $n \rightarrow \infty$ $\forall h$.

There is no stability condition. ✓ (1.5)

d) $\left.\begin{array}{l} \frac{dy}{dx} = -25 \cdot y^{3/5} \\ y(0) = 1 \end{array}\right\} \rightarrow$ Backward Euler method

$$Y_{i+1} = Y_i + h \cdot f(X_{i+1}, Y_{i+1})$$

$h = 1/10$.

$$Y_{i+1} = Y_i + (1/10)(-25) \cdot Y_{i+1}^{3/5}$$

Step nº01 (i=0)

$Y_1 = Y_0 - 2'5\,Y_1^{3'5} = 1 - 2'5 \cdot Y_1^{3'5};$

$F(Y_1) = Y_1 - 1 + 2'5\,Y_1^{3'5};$

We solve $F(Y_1)=0$ through the Newton method.

$$Y_1^0 = Y_0 = 1;\quad F(Y_1^0) = 1 + 2'5 - 1 = 2'50 \neq 0.$$
$$F'(Y_1) = 1 + 2'5 \cdot 3'5 \cdot Y_1^{2'5};$$
$$F'(Y_1^0) = 1 + 2'5 \cdot 3'5 \cdot 1^{2'5} = 9'75.$$

$$Y_1^1 = Y_1^0 - \frac{F(Y_1^0)}{F'(Y_1^0)} = 1 - 2'5/9'75 = 0'744$$

$$F(Y_1^1) = 0'744 + 2'5 \cdot 0'744^{3'5} - 1 = 0'632 \neq 0.$$
$$F'(Y_1^1) = 1 + 2'5 \cdot 3'5 \cdot 0'744^{2'5} = 5'177.$$

$$Y_1^2 = Y_1^1 - \frac{0'632}{5'177} = 0'744 - 0'122 = 0'622$$

$$\boxed{Y_1 = 0'622}$$

✓

Step nº02 (i=1)

$Y_2 = Y_1 - 2'5 \cdot Y_2^{3'5};$

$F(Y_2) = Y_2 + 2'5 \cdot Y_2^{3'5} - 0'622,$

$F'(Y_2) = 1 + 2'5 \cdot 3'5 \cdot Y_2^{2'5};$

$$\left.\begin{array}{l} Y_2^0 = Y_1 = 0'622 \\ F(Y_2^0) = 0'474; \\ F'(Y_2^0) = 3'670; \end{array}\right| \rightarrow Y_2^1 = Y_2^0 - \frac{F(Y_2^0)}{F'(Y_2^0)} = 0'622 - 0'129 = 0'493.$$

Exercise 3

d) $F(Y_2^1) = 0'493 + 2'5 \cdot 0'493^{3'5} - 0'622 = 0'081 \neq 0.$

$F'(Y_2^1) = 2'493.$

$$Y_2^2 = Y_2^1 - \frac{F(Y_2^1)}{F'(Y_2^1)} = 0'493 - \frac{0'081}{2'493} = 0'461$$

$\boxed{Y_2 = 0'461}$

✓ ②

e) $dy/dx = -25\, y^{3'5}$, $y(0) = 1$ → Forward Euler method $h = 1/10$.

$$Y_{i+1} = Y_i + h \cdot f(X_i, Y_i).$$

$Y_1 = Y_0 + h \cdot f(X_0, Y_0) = 1 - 2'5 \cdot 1^{3'5} = -1'50;$

$Y_2 = Y_1 + h\, f(X_1, Y_1) = -1'50 - 2'5(-1'5)^{3'5}$: this expression gives a complex result since the Forward Euler method with $h = 1/10$ is not stable to solve this problem.

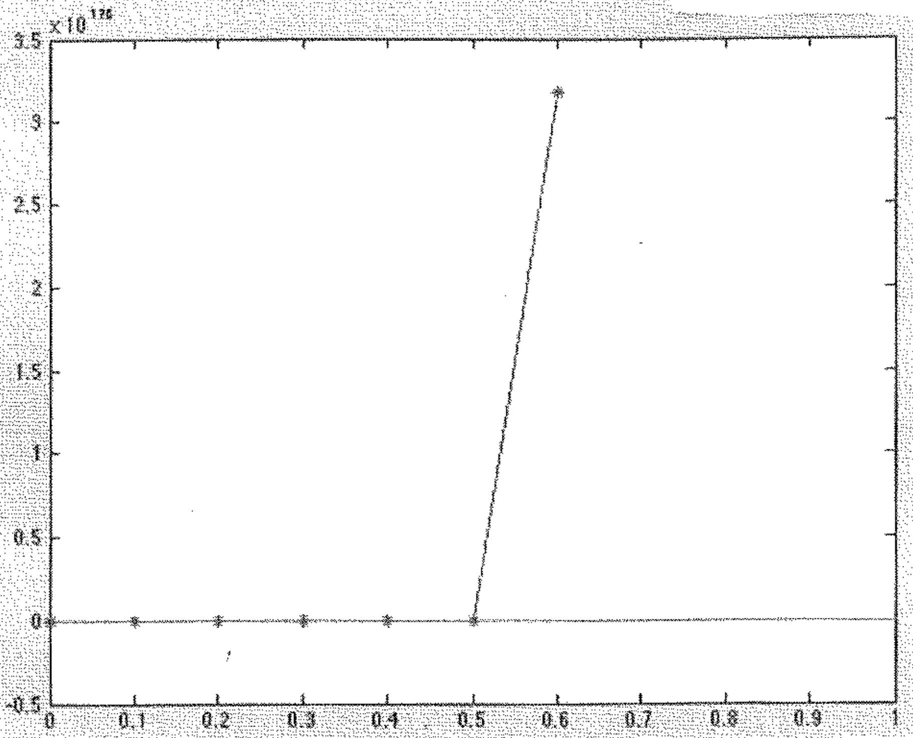
②

$h = 1/10$ does not satisfy the stability condition of $0 < h < 2/\lambda$, that for $\lambda = 25 \rightarrow 0 < h < 1/12'5$.

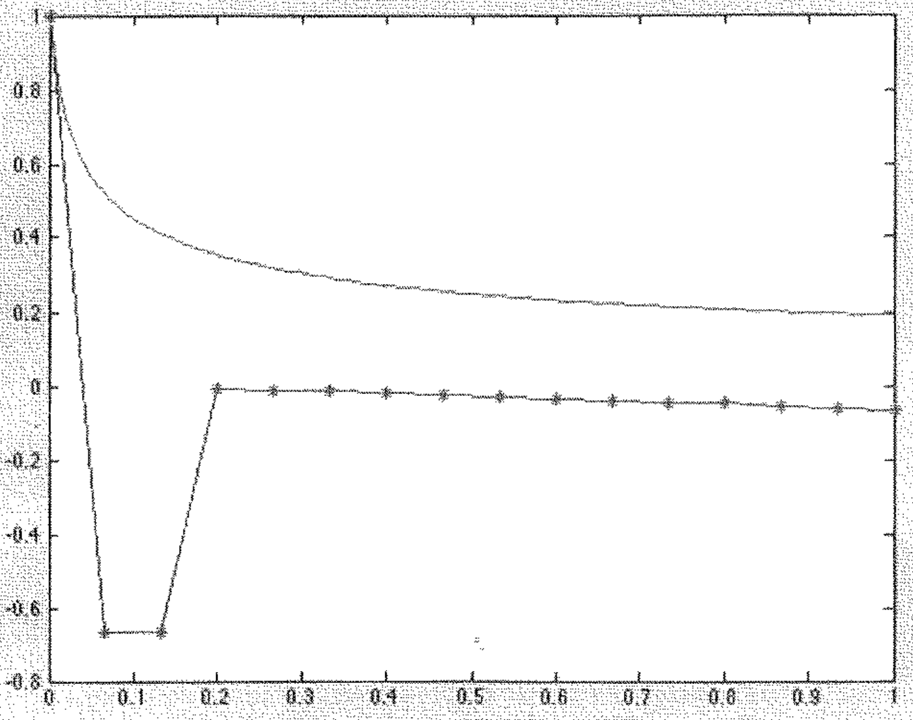
✓

Exercise 3. f)

h=1/10

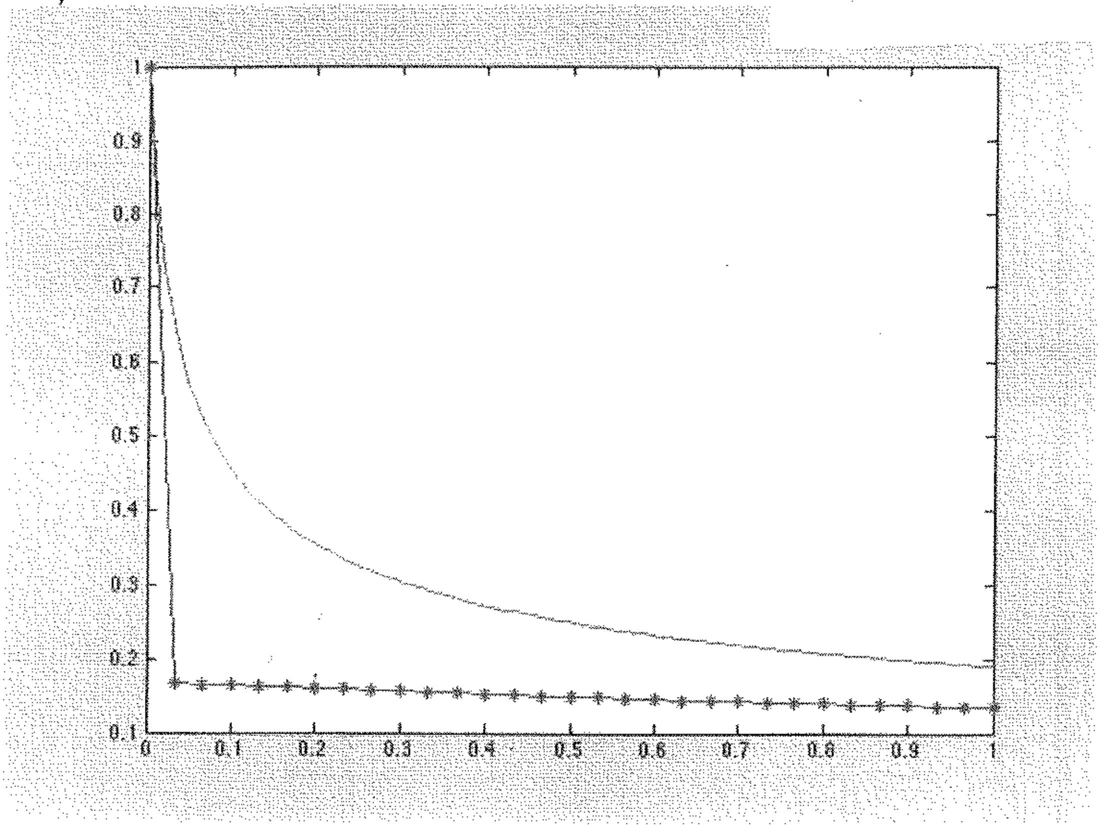


h=1/15

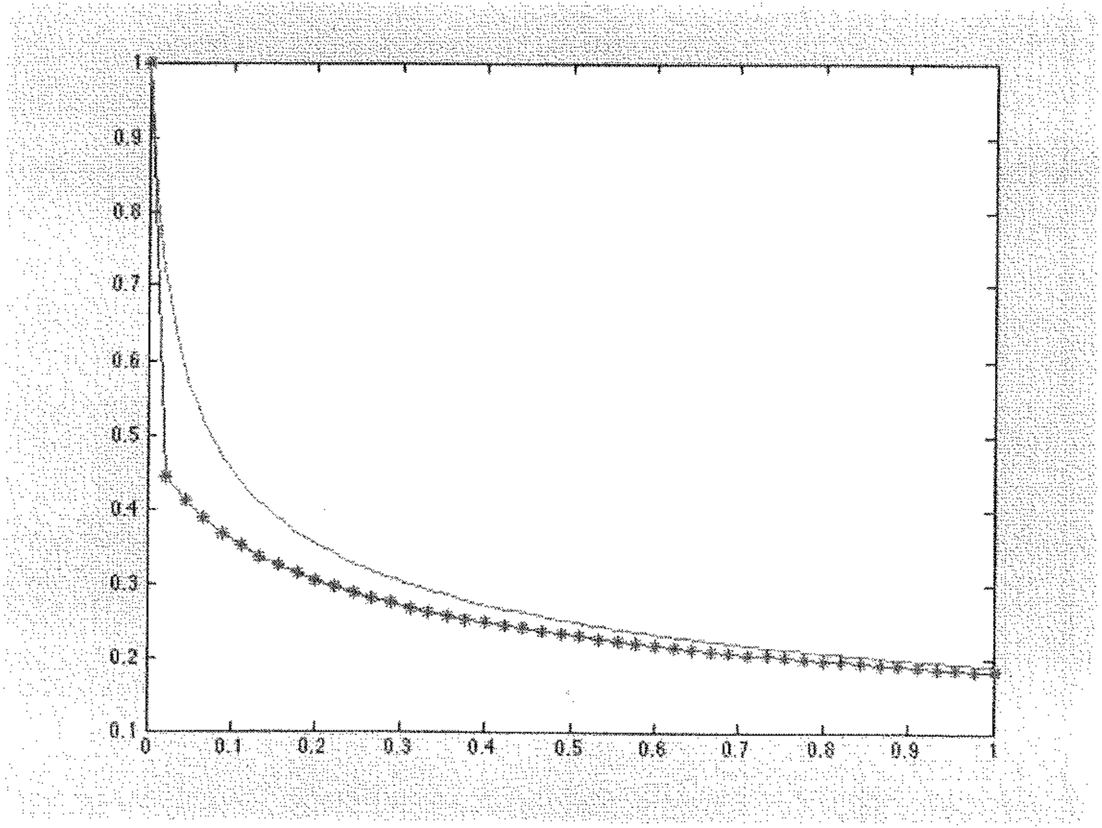


Exercise 3 f)

h=1/30

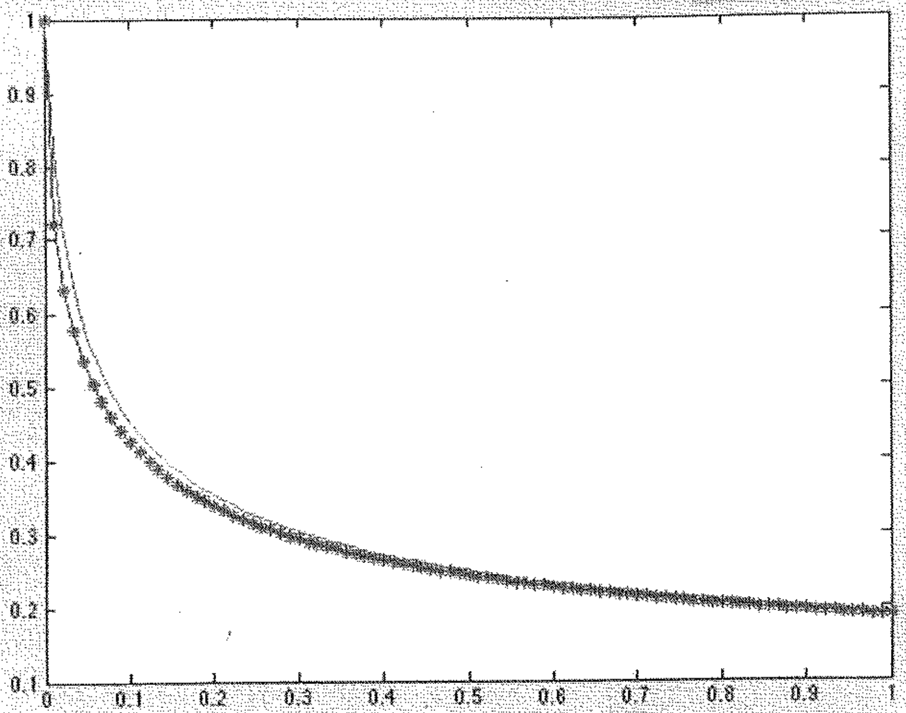


h=1/45



Exercise 3 f)

h=1/90



According to the abovementioned graphs, the maximum stable interval size is 1/30, value that is more restrictive than 1/12'50. This is due to the fact that the function we are studying is not equal to the model function $y' = -\lambda y$.

linearization!

①

Exercise 4

$$\begin{cases} \frac{d^2y}{dx^2} + \omega^2 y = 0 \quad \text{in } x \in [0,1] \\ y(0) = 0; \\ \left.\frac{dy}{dx}\right|_{x=0} = \omega; \end{cases}$$

$m = 4 \rightarrow h = (b-a)/m = 1/4;$

$\omega^2 = 3 \rightarrow \omega = \sqrt{3};$

a) $u_1 = y;$

$u_2 = y' = u_1'; \longrightarrow u_1' = u_2$

$u_3 = y'' = u_2'; \longrightarrow u_2' = y'' = -\omega^2 y = -\omega^2 u_1$

$y(0) = 0 \longrightarrow u_1(0) = 0;$

$y'(0) = \omega \longrightarrow u_2(0) = \omega;$

The equivalent system of two first order ODES is: ①

$$\begin{cases} u_1' = u_2; \\ u_2' = -\omega^2 u_1; \\ u_1(0) = 0; \\ u_2(0) = \omega; \end{cases}$$ ✓

b) We solve the previous system with the aid of the Forward Euler method with four steps to comput $u_1(1) = u_{1,4}$

FD Euler method: $u_{i+1} = u_i + h \cdot f(x_i, u_i)$

For our particular case: $u_{1,i+1} = u_{1,i} + h \cdot u_{2,i}$

$u_{2,i+1} = u_{2,i} + h(-\omega^2) u_{1,i}$

■ Step 1 ($i=0$)

$u_{1,1} = u_{1,0} + h\cdot u_{2,0} = 0 + (1/4)\cdot\sqrt{3} \approx 0'433$;

$u_{2,1} = u_{2,0} + h(-w^2)u_{1,0} = \sqrt{3} + (1/4)(-3)\cdot 0 = \sqrt{3} \approx 1'732$;

■ Step 2 ($i=1$)

$u_{1,2} = u_{1,1} + h\, u_{2,1} = 0'433 + (1/4)\, 1'732 \approx 0'866$;

$u_{2,2} = u_{2,1} + h(-w^2)u_{1,1} = 1'732 + (1/4)(-3)\cdot 0'433 \approx 1'407$;

■ Step 3 ($i=2$)

$u_{1,3} = u_{1,2} + h\cdot u_{2,2} = 0'866 + (1/4)\cdot 1'407 \approx 1'217$;

$u_{2,3} = u_{2,2} + h(-w^2)u_{1,2} = 1'407 + (1/4)(-3)(0'866) \approx 0'757$;

■ Step 4 ($i=3$)

②

$u_{1,4} = u_{1,3} + h\cdot u_{2,3} = 1'217 + (1/4)\cdot 0'757 \approx 1'406$;

$u_{2,4} = u_{2,3} + h(-w^2)u_{1,3} = 0'757 + (1/4)(-3)1'217 \approx -0'155$;

$\boxed{Y(1) = 1'406}$ ✓

c) If we compute the same problem but with 8 steps.

$Y(1) = 1'190$ ✓

In order to have Y(1) with 3 significative digits, we need 2048 steps. for wich the solution is 0'9877

?

How do you obtain this?

$m = 2048 \Rightarrow h = 1/2048$

(1.5)

```
% Forward Euler Method for a 2nd ODE

clear all; close all; clc

m=8;
h = 1/m;

x = linspace(0,1,m+1);

for i=1:m+1
    u1(i) = 0;
end

u1(1) = 0;

for i=1:m+1
    u2(i) = 0;
end

u2(1) = 3^(1/2);

for i = 1:m
    u1(i+1) = u1(i) + h*u2(i);
    u2(i+1) = u2(i) + h*(-3)*u1(i);
end

u1(m+1)

plot(x,u1,'*-')
```

```
% Forward Euler Method for a 2nd ODE with tolerance

clear all; close all; clc

error = 1;

m1 = 2;

while error ~= 0
    m1 = 2*m1;
    h1 = 1/m1;

    for i=1:m1+1
        u1m(i) = 0;
    end
    u1m(1) = 0;

    for i=1:m1+1
        u2m(i) = 0;
    end
    u2m(1) = 3^(1/2);

    for i = 1:m1
        u1m(i+1) = u1m(i) + h1*u2m(i);
        u2m(i+1) = u2m(i) + h1*(-3)*u1m(i);
    end

    m2 = 2*m1;
    h2 = 1/m2;

    for i=1:m2+1
        u1m2(i) = 0;
    end
    u1m2(1) = 0;

    for i=1:m2+1
        u2m2(i) = 0;
    end
    u2m2(1) = 3^(1/2);

    for i = 1:m2
        u1m2(i+1) = u1m2(i) + h2*u2m2(i);
        u2m2(i+1) = u2m2(i) + h2*(-3)*u1m2(i);
    end

    error = abs(fix(u1m(m1+1)*1000) - fix(u1m2(m2+1)*1000));
end

u1m(m1+1)
m1

x = linspace(0,1,m1+1);

plot(x,u1m,'.')
```

R

EXERCISES 2
STUDENT 1
(excel.lent)

yor

$\boxed{9}$

1.

$$\frac{dy}{dx} = f(x,y) \quad x \in (0,1)$$
$$y(0) = 1$$

Forward Euler method

$$Y_{i+1} = Y_i + h f(x_i, Y_i)$$

a)

Taylor series expansion

$$y_{i+1} = y_i + h \frac{dy}{dx}(x_i) + O(h^2)$$

$$\Rightarrow \frac{dy}{dx}(x_i) = \frac{y_{i+1} - y_i}{h} - \left[\frac{O(h^2)}{h}\right] = \frac{y_{i+1} - y_i}{h} - \tau$$

(1.5)

$$\Rightarrow \tau(h) = O(h)$$

given that $O(h^2)$ in Taylor series expansion is:

$$O_i(h^2) \approx \frac{h^2}{2!}\frac{d^2y}{dx^2}(x_i) + \frac{h^3}{3!}\frac{d^3y}{dx^3}(x_i) + \dots + \frac{h^n}{n!}\frac{d^ny}{dx^n}(x_i) \dots$$

✓

then:

$$\text{if } h \to 0 \quad \max_{0 \le i \le m} \tau_i(h) \to 0, \text{ for } \tau_i(h) = \frac{O_i(h^2)}{h}$$

so the forward Euler method is consistent.

b)

①

Backward Euler method ✓

$$Y_{i+1} = Y_i + h f(x_{i+1}, Y_{i+1})$$

c) $\frac{dy}{dx} = -\lambda y$ with $\lambda \in \mathbb{R}^+$

Forward Euler method

$$Y_{i+1} = Y_i - h\lambda Y_i$$
$$Y_{i+1} = G Y_i, \quad G = 1 - h\lambda$$

Backward Euler method

$$Y_{i+1} = Y_i - h\lambda Y_{i+1}$$
$$Y_{i+1} = G Y_i, \quad G = \frac{1}{1+\lambda h}$$

given that the analytical solution always goes to zero when x increases; the method will be stable if the numerical solution goes to zero.

$$|G| < 1$$

Forward Euler

$|1-h\lambda|<1$

$h\lambda>0$ and $h\lambda<2$

Backward Euler

$|1+\lambda h|>1$

$h\lambda>0$

1.5

✓

d)

$$\frac{dy}{dx} = -25y^{3.5} \qquad h = 1/10$$

$$y(0) = 1$$

$$Y_{i+1} = Y_i + hf(x_{i+1}, Y_{i+1})$$

$$Y_{i+1} = Y_i + h(-25\,Y_{i+1}^{3.5})$$

using Newton's method

$$F(Y_{i+1}^{(k)}) = Y_{i+1}^{(k)} - Y_i + h\,25\,Y_{i+1}^{(k)3.5}$$

$$F'(Y_{i+1}^{(k)}) = 1 + 87.5\,h\,Y_{i+1}^{(k)2.5}$$

$$Y_{i+1}^{(k+1)} = Y_{i+1}^{(k)} - \frac{F(Y_{i+1}^{(k)})}{F'(Y_{i+1}^{(k)})}$$

with $\bar{Y}_{i+1}^{(0)} = Y_i - h25Y_i^{3.5} > 0 \;\Rightarrow\; \max(\bar{Y}_{i+1}^{(0)}, Y_i) = Y_{i+1}^{(0)}$

i	K	Y	F	dF
1	0	1	2,5	9,75
	1	0,744	0,630	5,17
	2	0,622	0,0957	3,67
	3	0,596		
2	0	0,596	0,408	3,40
	1	0,476	0,0654	2,36
	2	0,448	0,00266	2,18
	3	0,447		

2

✓

e) using forward Euler method

$$Y_{i+1} = Y_i - 25 h Y_i^{3.5} \qquad h = \frac{1}{10}$$

i	Y_i
0	1
1	-1,5
2	-1,5 + 10.3338 i

② ✓

f) The minimum stable interval size is $h = 1/45$, with $\lambda h = 25/45 = 0,5$. Which is smaller than the $\lambda h < 2$ obtained.
The stable interval size is smaller than expected cause the non-linearity in the equation. ok but then you should linearize!

①

2.

$$\frac{d^2y}{dx^2} + w^2 y = 0 \qquad x \in (0,1)$$

$$y(0) = 0$$

$$\frac{dy}{dx}(0) = w$$

a) $\frac{d\mathbf{y}}{dx}(x) = \mathbf{f}(x, \mathbf{y}) \qquad x \in (0, 1) \qquad \mathbf{y} = \begin{pmatrix} y_{(1)} \\ y_{(2)} \end{pmatrix}$

$\mathbf{y}(0) = \boldsymbol{\alpha}$

$$\mathbf{f} = \begin{pmatrix} y_{(2)} \\ -w^2 y_{(1)} \end{pmatrix}, \qquad \boldsymbol{\alpha} = \begin{pmatrix} 0 \\ w \end{pmatrix}$$

① ✓

b) $w^2 = 3$

solution at $x=1$ with $n=4$

$h = \frac{1}{4}$ $\qquad Y_{i+1} = Y_i + h f(x_i, Y_i)$

i	$Y_{(1)}$	$Y_{(2)}$
0	0	1,7321
1	0,433	1,7321
2	0,866	1,4073
3	1,2178	0,7578 ✓
4	1,4073	-0,1556

$Y_{(1)} = y$

$Y_{(2)} = \frac{dy}{dx}$

②

c) for $n=8$

$Y_{(1)}^{8} = 1,1902$

$Y_{(2)}^{8} = -0,2798$

Using $n \gg 8$

$Y_{(1)}^{\infty} \approx 0,9885$

$E_n = Y_{(1)}^{8} - Y_{(1)}^{\infty} \cong 0,2017$

$h = \frac{1}{n} = \frac{1}{8}$

$\Rightarrow$ for $\boxed{tol = \frac{1}{2} \cdot 10^{-3}}$

$$h^* = \left(\frac{tol}{E_n}\right)^{1/(p+1)} h$$

$\Rightarrow h^* \cong 0,0062$

(1.5)

using h^*

$Y_{(1)}^{*} \cong 0,9962$, $\quad E^* \simeq 7,8 \cdot 10^{-3}$ still not with required precision ...

EXERCISES 2

STUDENT 2

(EXCEL·LENT)

8.25

12/11/13

NUMERICAL METHODS FOR PDEs
HOMEWORK 2

Exercise 3

The ordinary differential equation

$$\frac{dy}{dx} = f(x,\, y)$$

is defined over the domain $(0, 1)$, and is to be solved numerically subject to the initial condition $y(0) = 1$, where $y(x)$ is the exact solution. The forward Euler method for integrating the above differential equation is written as

$$Y_{i+1} = Y_i + hf(x_i, Y_i)$$

where Y_i denotes the discrete solution at node i, with position x_i, of a uniform grid of nodes of constant grid interval size h and $x_{i+1} = x_i + h$.

a) Using a Taylor series expansion, deduce the leading truncation error of the scheme. Is the method consistent? Explain your answer.

b) State the backward Euler method for integrating the above differential equation where $f(x, y)$ is a general non-linear function of x and y.

c) Deduce the stability limits of the respective forward Euler method and backward Euler method for the model equation $dy/dx = -\lambda y$ where λ is a positive real constant.

d) Use the backward Euler method to compute the numerical solution of the ordinary differential equation

$$\frac{dy}{dx} = -25y^{3.5}$$

with initial condition $y(0) = 1$, by hand for two steps with grid interval size $h = 1/10$. (Use 2 Newton iterations per step for this calculation.)

e) Use the forward Euler method to compute the numerical solution of the above ordinary differential equation with same initial condition by hand for two steps with grid interval size $h = 1/10$.

f) The analytical solution is

$$y(x) = \left(\frac{125x + 2}{2}\right)^{-2/5}$$

Using Matlab codes, indicate the maximum stable interval size possible for forward Euler method from the following; $h = 1/10$, $h = 1/15$, $h = 1/30$, $h = 1/45$, $h = 1/90$. How does your choice compare with the stability condition?

a)

1

Taylor series expansion is:

$$y_{i+1} = y_i + h\frac{dy}{dx}(x_i) + \mathcal{O}(h^2),$$

therefore the derivative can be expressed as:

$$\frac{dy}{dx}(x_i) = \frac{y_{i+i} - y_i}{h} - \underbrace{hO(h^2)}_{\tau_i(h)}.$$? order?

✓ Finally, it can be said that the method is consistent, because the truncation error is a function of h, so when h tends to 0 τ also tends to 0.

b)

Backward Euler can be deduced in a way similar to the forward Euler:

$$y_{i+1} = y_i - h\frac{dy}{dx}(x_{i+1}) + O(h^2)$$

$$\frac{dy}{dx}(x_{i+1}) = \frac{y_{i+i} - y_i}{h} + \underbrace{hO(h^2)}_{\tau_i(h)}.$$

① Therefore, the approximation can be written as:

$$f(x_{i+1}, Y_{i+1}) = \frac{Y_{i+i} - Y_i}{h}$$

✓

$$Y_{i+1} = Y_i + hf(x_{i+1}, Y_{i+1})$$

c)

First of all notice that the analytical solution for the model equation is $y = Ae^{-\lambda x}$ where A is a real constant. For forward Euler the i^{th} approximation can be expressed as:

$$Y_i = (1 - \lambda h)^i Y_0.$$

As the analytical expression tends to 0 when x tends to infinity the approximation must tend to 0 when i tend to infinity, so it means that:

$$|1 - \lambda h| < 1 \Rightarrow \lambda h < 2,$$

(1.5) and since h and λ are positive reals the solution will be stable when:

$$0 < \lambda h < 2$$

✓

For backward Euler the i^{th} approximation can be expressed as:

$$Y_i = \left(\frac{1}{1 + \lambda h}\right)^i Y_0.$$

Imposing the same criteria as before and taking into account that h and λ are positive reals,

$$\left(\frac{1}{1 + \lambda h}\right)$$

✓ will be always smaller than 1, so backward Euler is unconditional stable –is stable for any $h > 0$–.

d)

Particularizing backward Euler method:

$$Y_{i+1} = Y_i + \frac{1}{10}(-25Y_{i+1}^{3.5})$$

The first step is:

$$Y_1 = Y_0 + \frac{1}{10}(-25Y_1^{3.5}) \Rightarrow Y_1 + \frac{25}{10}Y_1^{3.5} - Y_0 = 0$$

Applying two steps of Newton's method with $Y_1^0 = 1$, the approximation becomes:

$$Y_1^1 = -\frac{-g(Y_1)}{g'(Y_1)} + Y_1^1 = \frac{-2.5}{1+\frac{25\cdot 3.5}{10}} + 1 = 0.7436$$

$$Y_1^2 = -\frac{-0.7436 + 2.5\cdot 0.7436^{3.5} - 1}{1+\frac{25\cdot 3.5\cdot 0.7436^{2.5}}{10}} + 0.7436 = \frac{-0.62994}{5.17197} + 0.7436 = 0.6218$$ ✓

Therefore, $Y_1 = 0.6218$, second step of backward Euler is: ②

$$Y_2 = Y_1 + \frac{1}{10}(-25Y_2^{3.5}) \Rightarrow Y_2 + \frac{25}{10}Y_2^{3.5} - Y_1 = 0$$

Applying two steps of Newton's method with $Y_2^0 = 0.5$, the approximation becomes:

$$Y_2^1 = -\frac{-0.5 + 2.5\cdot 0.5^{3.5} - 0.6218}{1+\frac{25\cdot 3.5\cdot 0.5^{2.5}}{10}} + 0.5 = \frac{-0.09917}{2.5468} + 0.5 = 0.4611$$

$$Y_2^2 = -\frac{-0.4611 + 2.5\cdot 0.4611^{3.5} - 0.6218}{1+\frac{25\cdot 3.5\cdot 0.4611^{2.5}}{10}} + 0.4611 = \frac{5.727\cdot 10^{-3}}{2.2633} + 0.4611 = 0.4586$$ ✓

Therefore, $Y_2 = 0.4586$.

e)

The forward Euler $i+1$ approximation is $Y_{i+1} = Y_i + hf(x_i, Y_i)$, therefore Y_1 and Y_2 are: ②

$$Y_1 = 1 + \frac{1}{10}(-25\cdot 1^{3.5}) = 1 - 2.5 = -1.5$$

$$Y_2 = -1.5 + \frac{1}{10}(-25\cdot(-1.5)^{3.5}) = -1.5 + \frac{1}{10}\left(-25\cdot\left(\sqrt{-1.5}\right)^{3.5}\right)$$

As it can be seen it's impossible to compute Y_2 with real numbers, that's because forward Euler is unstable for this step size. ✓

f)

Using Matlab code the maximum stable interval size possible is $h = 1/30$. Using the condition obtained from the model equation using $h = 1/15$ would be enough, but it's not because of the fact that y is elevated to the power 3.5. The stability condition for the model equation says that $\lambda h < 2$, but in this case y to the power 3.5 not to the power 1 as it is at model equation. ① linearization!

Exercise 4

The second-order ordinary differential equation

$$\frac{d^2y}{dx} + \omega^2 y = 0$$

is defined over the domain $(0, 1)$, and is to be solved numerically subject to the initial conditions $y(0) = 0$, $dy/dx(0) = \omega$, where $y(x)$ is the exact solution.

a) Reduce the above second order ODE to a system of first order ODEs.

b) Set $\omega^2 = 3$. Using the forward Euler method to integrate the system, compute the solution at $t = 1$ by hand with $n = 4$ steps. Use the Forward Euler code to check your results.

c) Using the Matlab code, compute the solution using $n = 8$ steps. Use these solution values to estimate the step size required to obtain a numerical solution with three significant digits. Try your new step size.

a)

Setting $y_1 = y$, and $y_2 = \frac{dy}{dx}$, is possible to write that:

①

$$\begin{cases} \frac{dy_2}{dx} + \omega^2 y_1 = 0 \\ \frac{dy_1}{dx} = y_2 \\ y_1(0) = 0 \\ y_2(0) = \omega \end{cases}$$

✓

b)

The system of equations asked to solve is:

$$\begin{cases} \frac{dy_2}{dx} = -3y_1 \\ \frac{dy_1}{dx} = y_2 \\ y_1(0) = 0 \\ y_2(0) = \sqrt{3} \end{cases}$$

Applying forward Euler:

$$\begin{cases} Y_1^{i+1} = Y_1^i + hY_2^i \\ Y_2^{i+1} = Y_2^i + h(-3Y_1^i) \end{cases}$$

For $i = 0$:

②

$$\begin{cases} Y_1^1 = Y_1^0 + hY_2^0 = 0 + \frac{1}{4}\sqrt{3} = \frac{\sqrt{3}}{4} \\ Y_2^1 = Y_2^0 + h(-3Y_1^0) = \sqrt{3} - \frac{1}{4}(-3 \cdot 0) = \sqrt{3} \end{cases}$$

For $i = 1$:

$$\begin{cases} Y_1^2 = \frac{\sqrt{3}}{4} + \frac{1}{4}\sqrt{3} = \frac{\sqrt{3}}{2} \\ Y_2^2 = \sqrt{3} - \frac{1}{4}(-3\frac{\sqrt{3}}{4}) = \frac{13}{16}\sqrt{3} \end{cases}$$

For $i = 2$:

$$\begin{cases} Y_1^3 = \frac{\sqrt{3}}{2} + \frac{1}{4}\frac{13}{16}\sqrt{3} = \frac{45}{64}\sqrt{3} \\ Y_2^3 = \frac{13}{16}\sqrt{3} - \frac{1}{4}(-3\frac{\sqrt{3}}{2}) = \frac{7}{16}\sqrt{3} \end{cases}$$

For $i = 3$:

$$\begin{cases} Y_1^4 = \frac{45}{64}\sqrt{3} + \frac{1}{4}\frac{7}{16}\sqrt{3} = \frac{13}{16}\sqrt{3} = 1.4073 \\ Y_2^4 = \frac{7}{16}\sqrt{3} - \frac{1}{4}(-3\frac{45}{64}\sqrt{3}) = \frac{-23}{256}\sqrt{3} = -0.1556 \end{cases}$$

✓

Finally get that at $t = 1$ the approximation of y is 1.4073, and the approximation of the derivative is -0.1556.

c)

①

The approximation using $n = 8$ of y is 1.1902, and the approximation of the derivative is -0.2798. Three significant digits means that the first three nonzero digit of the approximation must be equal to the first three nonzero digits of the analytical solution. For this particular case the analytical solution is $y(x) = \sin\sqrt{3}x$ and for $x = 1$ it values 0.9870. The first approximation that gives this 3 significant digits is when $n = 1523$ and the approximated value is 0.987999.

✓

Numerical Methods for Partial Differential Equations

Finite Differences

Starred questions (*) have to be handed in for marking.

1. Consider a bar with length 1 m and constant thermal conductivity k. The temperature at the ends of the bar is

$$T(0) = 0, \quad T(1) = 1. \tag{1}$$

In order to determine the temperature distribution on the bar, the heat equation is stated

$$k\frac{d^2T}{dx^2} = 0, \quad x \in (0,1) \tag{2}$$

with boundary conditions (1).

a) Derive a numerical scheme for the solution of the boundary problem given by equations (2) and (1) using a centered approximation of order Δx^2. Detail the linear system obtained for $\Delta x = 0.2$.

b) Solve the linear system obtained in *a*) for a bar with constant thermal conductivity $k = 1$. Represent the solution. With no extra computations, justify how would the temperature distribution for a bar be with conductivity $k = 10$.

c) Solve the linear system of equations with the Gauss-Seidel method and the Conjugate Gradient method. Compute 4 iterations with initial approximation $x_i = 1$, and comment the convergence of both methods.

2.* Let us consider the differential equation

$$u_t + au_x = 0, \quad x \in (0,1), \quad t \geq 0, \quad a > 0 \tag{3}$$

with initial condition

$$u(x,0) = \sin(2\pi x),$$

and periodic boundary conditions, that is

$$u(0,t) \;=\; u(1,t).$$

a) Propose an implicit finite difference scheme, with first order in time and space, for the discretization of 3. Justify the selection of the approximation for the spatial derivative.

b) How are periodic boundary conditions treated? Write in detail the system of equations to solve in each time step.

c) Suggest a direct method and an iterative method for the solution of the linear systems of equations.

d) Draw schematically the fill-in of the matrix for the direct method proposed in the previous section.

---

3. Consider the elliptic equation

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = f$$

with homogeneous Dirichlet boundary conditions (i.e $u = 0$ at the whole boundary) and source term $f(x, y) = 2(x^2 + y^2)$.

a) Solve the problem in a square domain $[0, 4] \times [0, 4]$ with $\Delta x = \Delta y = 1$, and determine the temperature at the point $(x, y) = (1, 1)$.

b) Solve the problem in the triangle defined by vertices $(0, 0)$, $(0, 4)$ and $(4, 0)$, with $\Delta x = \Delta y = 1$, and determine the temperature at the point $(x, y) = (1, 1)$.

c) Using the initial approximation $\mathbf{U} = \mathbf{0}$, compute 4 iterations for the solution of the system obtained in *b)* with the Jacobi and Gauss-Seidel methods, and comment the results.

4.* For the numerical modelling of a new technique of contamination control, it is interesting to solve the diffusion-reaction PDE

$$u_t = \nu u_{xx} + \sigma u \qquad \text{in } x \in (0, 1),\ t > 0 \tag{4}$$

with boundary conditions

$$u(0, t) = 0 \quad \text{and} \quad u_x(1, t) = 0 \tag{5}$$

and the initial condition

$$u(x, 0) = \begin{cases} 0 & \text{for} \quad x < 1/4 \\ 4x - 1 & \text{for} \quad 1/4 \le x < 1/2 \\ -4x + 3 & \text{for} \quad 1/2 \le x < 3/4 \\ 0 & \text{for} \quad 3/4 \le x \end{cases} \tag{6}$$

In the PDE (4), $\nu > 0$ is the diffusion coefficient and $\sigma < 0$ is the reaction coefficient. Both coefficients can be considered constant.

a) Propose an explicit finite difference scheme for the solution of the PDE (4) with boundary conditions (5) and initial condition (6). Detail the numerical treatment of boundary conditions.

b) Which scheme is obtained for $\sigma = 0$ (diffusion equation)? And for $\nu = 0$ (reaction equation)?

c) Take $\nu = 0.1$, $\sigma = -0.1$, $\Delta x = 0.25$ and $\Delta t = 0.1$, and compute two time steps with the explicit scheme proposed in section *a*. Are the obtained results reasonable? Discuss with the help of the graphic of the profile of u.

d) Propose an implicit finite difference scheme to solve the PDE (4) with boundary conditions (5) and initial condition (6). Detail how are boundary conditions treated, the structure of the matrix and the most suitable method to solve the linear system of equations.

---

5. The following finite differences schemes

$$\begin{aligned} &(i) \quad U_i^{n+1} = U_i^n - \frac{c}{2}\left(U_{i+1}^n - U_{i-1}^n\right) \\ &(ii) \quad U_i^{n+1} = U_i^n - c\left(U_i^n - U_{i-1}^n\right) \end{aligned} \tag{7}$$

with Courant number $c = a\Delta t/\Delta x$, are considered for the solution of the boundary problem

$$\begin{aligned} &u_t + au_x = 0, \quad x \in (0,4), \quad t \geq 0, \quad a > 0 \\ &u(x,0) = u_0(x), \quad u(0,t) = 0. \end{aligned} \tag{8}$$

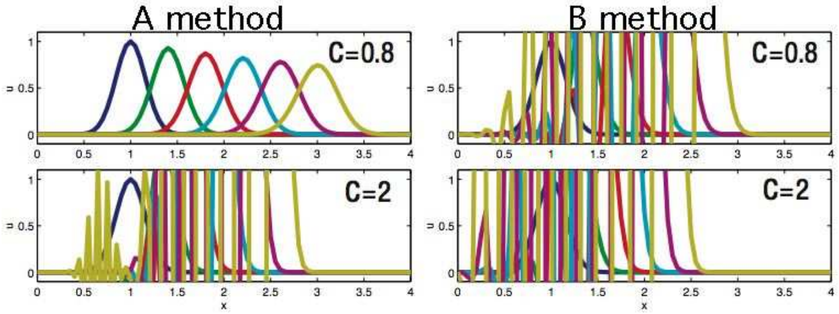


Figure 1: Numerical results with methods A and B

a) Discuss for each method wether it is an explicit or implicit method and indicate the truncation order. Which linear solver would you use in each case?

b) Figure 1 shows the solution with methods A and B for Courant numbers $c = 0.8$ and $c = 2$. Decide reasonably which of the schemes *(i)* and *(ii)* correspond to methods A and B.

c) Comment the stability of both methods. Do the numerical results correspond to the expected behavior?

6.* The partial differential equation

$$\frac{\partial u}{\partial t} = b\frac{\partial^2 u}{\partial x^2}$$

is defined over the domain $0 \leq x \leq 1$, and is to be solved numerically subject to the boundary conditions $u(0,t) = 0.0, \quad u(1,t) = 0.0$ and initial conditions $u(x,0) = u_0(x)$ ($u_0(x)$ defined below), where $u(x,t)$ is the exact solution.

The explicit forward time centred space scheme is used in the form

$$U_i^{n+1} - U_i^n = r(U_{i+1}^n - 2U_i^n + U_{i-1}^n)$$

where $r = b\Delta t/\Delta x^2$, subscript i is a spatial index (x-direction) and superscript n is the time level.

a) Using a Taylor series expansion, deduce the leading truncation error of the scheme and state if the scheme is consistent.

b) A uniform grid with three interior nodes and four equally spaced intervals is used. The initial data $u_0(x)$ is defined by

$$u_1^0 = 0.0, \quad u_2^0 = 3.0, \quad u_3^0 = 6.0, \quad u_4^0 = 3.0, \quad u_5^0 = 0.0,$$

and $b = 1/4$, $\Delta t = 0.25$. Compute the solution at the 3 interior nodes after one time step. Sketch the solution. Is the scheme stable in this case? Explain.

c) State the stability condition.

d) State the implicit backward time centered space scheme for the above problem.

e) Using the above data, write down the resulting system of equations (expressed in terms of general r) that must be solved in order to compute the solution at the 3 interior nodes after one time step.

f) Determine the solution via Gaussian elimination.

g) Will the method always be stable or is there a limitation on time step?

7.* Consider the following system of linear equations:

$$\begin{bmatrix} 10 & 4 & 0 \\ 6 & 12 & 3 \\ 0 & 5 & 9 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 46 \\ 81 \\ 65 \end{bmatrix}$$

a) Using initial data $\mathbf{x}^0 = [2, 3, 4]^T$, apply Jacobi method to the system for 3 iterations. Show your working and the results of each iteration.

b) Using the same initial data, apply the Gauss-Seidel iterative method for the same 3×3 system for 3 iterations. Again, show your working and the results for each iteration.

c) The exact solution of the system is $\mathbf{x}^* = [3, 4, 5]^T$. Determine which method gives the best result and explain why.

8.* Consider the following system of linear equations:

$$\begin{bmatrix} 1 & 5 \\ 5 & 100 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 17 \\ 310 \end{bmatrix}$$

a) As a first attempt, we have tried to solve the system using the steepest descent method, but convergence is too slow. Explain the causes of this slow convergence.

b) Can the conjugate gradient method be used to solve the system? Will it converge? If so, how many iterations are needed?

c) Compute two iterations of the conjugate gradient method applied to this system, using $\mathbf{x}^0 = [5, 5]^T$ as a starting vector. Which will be the results if a different starting vector is considered?

STUDENT 1

(APROVAT)

PDE

EXERCISES 3

4.

Assumptions

$u_t = \nu \cdot u_{xx} + \sigma \cdot u$ in $x \in (0,1),\ t > 0$

$\nu > 0$

$\sigma < 0$

Boundary conditions

$u(0,t) = g = 0$

$u_x(1,t) = h = 0$

Initial conditions

$u(x,0) = 0$ for $x < \frac{1}{4}$

$u(x,0) = 4x - 1$ for $\frac{1}{4} \le x < \frac{1}{2}$

$u(x,0) = -4x + 3$ for $\frac{1}{2} \le x < \frac{3}{4}$

$u(x,0) = 0$ for $\frac{3}{4} \le x$

a.

Scheme

The explicit method is the Forward in Time Centered in Space (FTCS)
The boundary conditions is a combination of Dirichlet and Neuman. The value in note M+1 is therefor unknown but it is a nesserary assumption for being able to solve the equation for all notes. The equation U^{n+1}_{M+1} (se the following) gives the value in the end point and is obtained by making a fictious node $i = m + 1$.

The equation is imposed in $\left(x_i, t^n\right)$

Derivatives

$$u_{t,i}{}^{n} \cdot \approx \frac{U_i^{n+1} - U_i^n}{\Delta t}$$

$$u_{xx,i}{}^{n} \cdot \approx \frac{\left(U_{i-1}\right)^n - 2U_i^n + \left(U_{i+1}\right)^n}{\Delta x^2}$$

Solution for next time step

$U_t = \nu \cdot U_{xx} + \sigma \cdot U$

$\Downarrow$

$$\frac{U_i^{n+1} - U_i^n}{\Delta t} = \nu \frac{\left(U_{i-1}\right)^n - 2U_i^n + \left(U_{i+1}\right)^n}{\Delta x^2} + \sigma \cdot U_i^n$$

$\Downarrow$

$U_i^{n+1} = r \cdot U^n_{i-1} \cdot + (1 - 2r) \cdot U_i^n + r \cdot U^n_{i+1} + \sigma \cdot \Delta t \cdot U_i^n$ $i = 1, \ldots\ldots, M$ ✓

$U_0^{n+1} = g^{n+1} = 0$ $i = 0$ ✓

$U^{n+1}_{M+1} \cdot = 2r \cdot U_M^n + (1 - 2r) \cdot U^n_{M+1} \cdot + 2r \cdot \Delta x \cdot h^n + \sigma \cdot \Delta t \cdot U^n_{M+1}$ $i = M + 1$

$U_i^0 = 0$ for $x < \frac{1}{4}$

$U_i^0 = 4x - 1$ for $\frac{1}{4} \le x < \frac{1}{2}$

$U_i^0 = -4x + 3$ for $\frac{1}{2} \le x < \frac{3}{4}$

$U_i^0 = 0$ for $\frac{3}{4} \le x$

Where $r = \nu \cdot \frac{\Delta t}{\Delta x^2}$

b.

Diffusion equation

Solution for next time step

$U_t = \nu \cdot U_{xx}$

$\Downarrow$

$U_i^{n+1} = r \cdot U^n_{i-1} \cdot + (1 - 2r) \cdot U_i^n + r \cdot U^n_{i+1}$ $i = 1, \ldots\ldots, M$

$$U_0^{n+1} = g^{n+1} = 0 \qquad i = 0$$

$$U^{n+1}_{M+1} = 2r \cdot U_M^{n} + (1 - 2r) \cdot U^{n}_{M+1} + 2r\,\Delta x \cdot h^{n} \qquad i = M + 1$$

$$U_i^{0} = 0 \qquad \text{for} \qquad x < \frac{1}{4}$$

$$U_i^{0} = 4x - 1 \qquad \text{for} \qquad \frac{1}{4} \le x < \frac{1}{2}$$

$$U_i^{0} = -4x + 3 \qquad \text{for} \qquad \frac{1}{2} \le x < \frac{3}{4}$$

$$U_i^{0} = 0 \qquad \text{for} \qquad \frac{3}{4} \le x$$

where $r = \nu \cdot \dfrac{\Delta t}{\Delta x^2}$

Reaction equation

$u_t = \sigma \cdot u$

Solution for next timestep

$$U_t = \sigma \cdot U$$

$$\Downarrow$$

$$\frac{U_i^{n+1} - U_i^{n}}{\Delta t} = \sigma \cdot U_i^{n}$$

$$\Downarrow$$

$$U_i^{n+1} = \sigma \cdot \Delta t \cdot U_i^{n} + U_i^{n} \qquad i = 0, \ldots\ldots, M + 1$$

The equation can be solved only in terms of the initial condition.

$$U_i^{0} = 0 \qquad \text{for} \qquad x < \frac{1}{4}$$

$$U_i^{0} = 4x - 1 \qquad \text{for} \qquad \frac{1}{4} \le x < \frac{1}{2}$$

$$U_i^{0} = -4x + 3 \qquad \text{for} \qquad \frac{1}{2} \le x < \frac{3}{4}$$

$$U_i^{0} = 0 \qquad \text{for} \qquad \frac{3}{4} \le x$$

c.

Assumptions

$\nu = 0.1$

$\sigma = -0.1$

$\Delta x = 0.25$

$\Delta t = 0.1$

$$r = \nu \cdot \frac{\Delta t}{\Delta x^2} = 0.16 \quad < \quad \frac{1}{2} \qquad \Rightarrow \qquad \text{Stabel}$$

Initial conditions notes

$$\begin{pmatrix} U_0^{0} \\ U_1^{0} \\ U_2^{0} \\ U_3^{0} \\ U_4^{0} \end{pmatrix} = \begin{pmatrix} 0 \\ 4x - 1 \\ -4x + 3 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 4 \cdot \frac{1}{4} - 1 \\ -4 \frac{1}{2} + 3 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}$$

Solution in n = 0

$$\begin{pmatrix} U_0^{1} \\ U_1^{1} \\ U_2^{1} \\ U_3^{1} \\ U_4^{1} \end{pmatrix} = \begin{bmatrix} g^{1} \\ r \cdot U_0^{0} + (1 - 2r) \cdot U_1^{0} + r \cdot U_2^{0} + \Delta t \cdot \sigma \cdot U_1^{0} \\ r \cdot U_1^{0} + (1 - 2r) \cdot U_2^{0} + r \cdot U_3^{0} + \Delta t \cdot \sigma \cdot U_2^{0} \\ r \cdot U_2^{0} + (1 - 2r) \cdot U_3^{0} + r \cdot U_4^{0} + \Delta t \cdot \sigma \cdot U_3^{0} \\ 2r \cdot U_3^{0} + (1 - 2r) \cdot U_4^{0} + 2r \cdot \Delta x \cdot h^{0} + \sigma \cdot \Delta t \cdot U_4^{0} \end{bmatrix}$$

$$\Downarrow$$

$$\begin{pmatrix} U_0^{1} \\ U_1^{1} \\ U_2^{1} \\ U_3^{1} \\ U_4^{1} \end{pmatrix} = \begin{bmatrix} 0 \\ r \cdot 0 + (1 - 2r) \cdot 0 + r \cdot 1 + \Delta t \cdot \sigma \cdot 0 \\ r \cdot 0 + (1 - 2r) \cdot 1 + r \cdot 0 + \Delta t \cdot \sigma \cdot 1 \\ r \cdot 1 + (1 - 2r) \cdot 0 + r \cdot 0 + \Delta t \cdot \sigma \cdot 0 \\ 2r \cdot 0 + (1 - 2r) \cdot 0 + 2r \cdot \Delta x \cdot 0 + \sigma \cdot \Delta t \cdot 0 \end{bmatrix} = \begin{pmatrix} 0 \\ 0.16 \\ 0.67 \\ 0.16 \\ 0 \end{pmatrix}$$

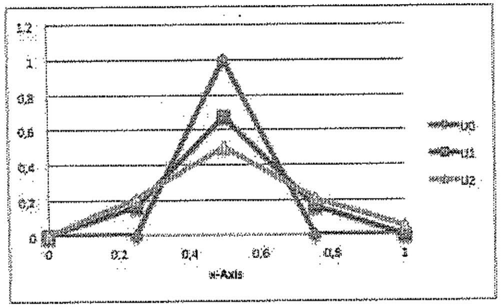
Solution in n = 1

$$\begin{pmatrix} U_0^2 \\ U_1^2 \\ U_2^2 \\ U_3^2 \\ U_4^2 \end{pmatrix} = \begin{bmatrix} g^2 \\ r\cdot U_0^1 + (1-2r)\cdot U_1^1 + r\cdot U_2^1 + \Delta t\cdot\sigma\cdot U_1^1 \\ r\cdot U_1^1 + (1-2r)\cdot U_2^1 + r\cdot U_3^1 + \Delta t\cdot\sigma\cdot U_2^1 \\ r\cdot U_2^1 + (1-2r)\cdot U_3^1 + r\cdot U_4^1 + \Delta t\cdot\sigma\cdot U_3^1 \\ 2r\cdot U_3^1 + (1-2r)\cdot U_4^1 + 2r\cdot\Delta x\cdot h^1 + \sigma\cdot\Delta t\cdot U_4^1 \end{bmatrix}$$

$\Downarrow$

$$\begin{pmatrix} U_0^2 \\ U_1^2 \\ U_2^2 \\ U_3^2 \\ U_4^2 \end{pmatrix} = \begin{bmatrix} 0 \\ r\cdot 0 + (1-2r)\cdot 0.16 + r\cdot 0.67 + \Delta t\cdot\sigma\cdot 0.16 \\ r\cdot 0.16 + (1-2r)\cdot 0.67 + r\cdot 0.16 + \Delta t\cdot\sigma\cdot 0.67 \\ r\cdot 0.67 + (1-2r)\cdot 0.16 + r\cdot 0 + \Delta t\cdot\sigma\cdot 0.16 \\ 2r\cdot 0.16 + (1-2r)\cdot 0 + 2r\cdot\Delta x\cdot 0 + \sigma\cdot\Delta t\cdot 0 \end{bmatrix} = \begin{pmatrix} 0 \\ 0.21 \\ 0.5 \\ 0.21 \\ 0.05 \end{pmatrix}$$

Plot U



Are b.c. fulfilled? The plot shows that the fuction is stable, which was expected. There is no source term in the PDE, which means that the function must go towards a straight line, when t goes to infinity and with a discretization that satisfies the stability condition. This is true for the result.

d.

Scheme The implicit method used is Backward in Time Centered in Space (BTCS).

The equation is imposed in $\left(x_i, t^{n+1}\right)$

Derivatives $\quad u_{t,i}^{n+1} \approx \dfrac{U_i^{n+1} - U_i^n}{\Delta t}$

$$u_{xx,i}^{n+1} \approx \frac{\left(U_{i-1}\right)^{n+1} - 2U_i^{n+1} + \left(U_{i+1}\right)^{n+1}}{\Delta x^2}$$

Solution for next time step

$U_t = \nu\cdot U_{xx} + \sigma\cdot U$

$\Downarrow$

$$\frac{U_i^{n+1} - U_i^n}{\Delta t} = \nu\frac{\left(U_{i-1}\right)^{n+1} - 2U_i^{n+1} + \left(U_{i+1}\right)^{n+1}}{\Delta x^2} + \sigma\cdot U_i^{n+1}$$

$\Downarrow$

$$U_i^n + \sigma\cdot\Delta t\cdot U_i^n = -r\left(U_{i-1}\right)^{n+1} + (1+2r)U_i^{n+1} - r\left(U_{i+1}\right)^{n+1}$$

$\Downarrow$

$U_i^n = -r\left(U_{i-1}\right)^{n+1} + (1 + 2r + \sigma\cdot\Delta t)U_i^{n+1} - r\left(U_{i+1}\right)^{n+1}$ $\quad i = 1,\ldots\ldots,M$

$U_0^{n+1} = g^{n+1} = 0$ $\quad i = 0$

$-2r\cdot U_M^{n+1} + (1 + 2r + \sigma\cdot\Delta t)\cdot U_{M+1}^{n+1} = U_{M+1}^n + 2\cdot r\cdot\Delta x\cdot h^{n+1}$ $\quad i = M + 1$

$U_i^0 = 0$ for $x < \frac{1}{4}$

$U_i^0 = 4x - 1$ for $\frac{1}{4} \le x < \frac{1}{2}$

$U_i^0 = -4x + 3$ for $\frac{1}{2} \le x < \frac{3}{4}$

$U_i^0 = 0$ for $\frac{3}{4} \le x$

Where $\quad r = \nu\cdot\dfrac{\Delta t}{\Delta x^2}$

Linear system

The linear system to be solved can be described by the following equation

$$A \cdot U^{n+1} = IU^{n} + F$$

Where

$$A = \begin{bmatrix} (1 + 2r + \sigma \cdot \Delta t) & -r & . & . & . \\ -r & (1 + 2r + \sigma \cdot \Delta t) & -r & . & . \\ . & -r & \dots & \dots & . \\ . & . & -r & (1 + 2r + \sigma \cdot \Delta t) & -r \\ . & . & . & -2r & (1 + 2r + \sigma \cdot \Delta t) \end{bmatrix}$$

$$F = \begin{pmatrix} r \cdot g^{n+1} \\ 0 \\ \dots \\ \dots \\ 2 \cdot r \cdot \Delta x \cdot h^{n+1} \end{pmatrix} \qquad U^{n} = \begin{bmatrix} U_1^{n} \\ U_2^{n} \\ \dots \\ \dots \\ \left(U_{M+1}\right)^{n} \end{bmatrix}$$

Method

Direct methods

Gauss elimintation will require changing the system on both sides due to row operations, this means that for each time step A and U^n is changed because U^n is time dependent. This means that Gauss elimination is not a suitable method, because of row operations for each time steps.

Cholesky method require that A is symmetric, but A is not symmetric, so it is not an option.

A suitable direct method is the LU factorization, using that $A = L \cdot U$, where A for the system shown above is general for all time steps, and since this method only depends on A, the factorization is only imposed once.

Iterative methods

The Conjugate Gradiant method is normally used for a symmetric A matrix, so it is not the most suitable method.

Gauss-Seidal and Jacobi both converges if A is diagonal dominant. The A matrix is diagonal dominant since $\sigma < 0$, so both methods can be used, but Gauss-Seidal is faster than Jacobi. Gauss-Seidal is therefor the most efficient suitable iterative method.

Conclusion

Since a direct method gives a exact result for the first iteration the LU factorization is the most efficient method for the system.

??

STUDENT 2

(APROVAT)

EXERCISES 3

Homework 3

For the numerical modelling of a new technique of contamination control, it is interesting to solve the following diffusion.reaction PDE

PDE to be solved $u_t = \nu u_{xx} + \sigma u$ in the domain [0,1], and with t > 0

Boundary conditions $u(0,t) = 0$ Dirichlet

$u_x(1,t) = 0$ Neumann

Initial conditions (all dirichlet)

$u(x,0) = 0$ for $x < \frac{1}{4}$

$u(x,0) = 4x - 1$ for $\frac{1}{4} \le x < \frac{1}{2}$

$u(x,0) = -4x + 3$ for $\frac{1}{2} \le x < \frac{3}{4}$

$u(x,0) = 0$ for $\frac{3}{4} \le x$

In the PDE, $\nu > 0$ is the diffusion coefficient and $\sigma < 0$ is the reaction coefficient.
Both coefficients can be considered constant.

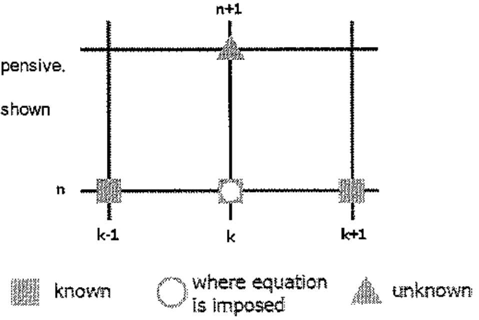
Question a - Propose an explicit finite difference scheme for the solution of the PDE with boundary conditions and initial conditions. Detail the numerical treatment of the boundary conditions.

The explicit scheme to solve the parabolic PDE is the Forward in Time Centered Space (FTCS).
This method is based on central difference in space (x) and forward euler method in time (t).
This leads to a 1st order of convergence in time and a 2nd order convergence in space.

The method is conditionally stable when $r < \frac{1}{2}$

Since the method is explicit the computations are inexpensive.

The method uses 3 points to calculate the solution as shown in the figure.



The discrete problem of the given PDE [illegible] the following

Discrete problem $$\frac{u_k^{n+1} - u_k^n}{\Delta t} = \nu \cdot \frac{(u_{k-1})^n - 2u_k^n + (u_{k+1})^n}{(\Delta x)^2} + \sigma \cdot u_k^n + \tau_k^n$$

Where τ_k^n is the truncation error

By neglecting the truncation error the numerical problem of the PDE can be written as the following

$$u_k^{n+1} = \Delta t \cdot \left[\frac{u_k^n}{\Delta t} + \sigma \cdot u_k^n + \frac{\nu \cdot \left[(u_{k-1})^n + (u_{k+1})^n - 2 \cdot u_k^n \right]}{\Delta x^2} \right]$$

$$\Downarrow$$

$$u_k^{n+1} = u_k^n + \sigma \cdot \Delta t \cdot u_k^n + \frac{\nu \cdot \Delta t}{(\Delta x)^2} \left[(u_{k-1})^n + (u_{k+1})^n - 2 \cdot u_k^n \right]$$

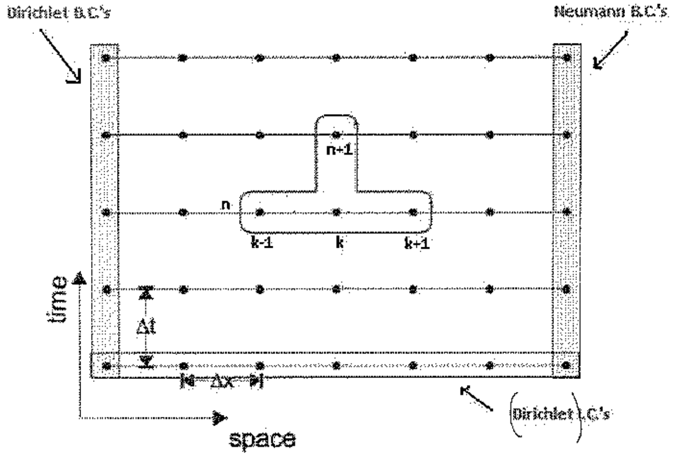
$$\Downarrow$$

Numerical problem $$U_k^{n+1} = r \cdot (U_{k+1})^n + (1 - 2r) \cdot U_k^n + r \cdot (U_{k-1})^n + \sigma \cdot \Delta t \cdot U_k^n$$ ✓

Where s is given by $r = \nu \cdot \frac{\Delta t}{(\Delta x)^2}$

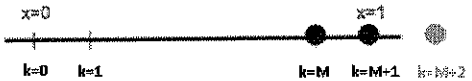
Boundary/Initial conditions

On the following picture the different conditions of the problem is shown.



When calculating the solution both the Dirichlet B.C.'s and I.C.'s are functions on time and can easily be used in the formula. They are both used to specify the starting points.

The neumann B.C.'s on the right side is a bit more difficult to implement, since they are variable with time.
To solve that we introduce a fictitious point at the right-hand side of the right boundary as shon in the figure below.



Using the approximation of the boundary condition

$$\frac{d}{dx}\left(u_{M+1}\right)^n = \frac{\left(u_{M+2}\right)^n - \left(u_M\right)^n}{2\cdot\Delta x} + O\left(\Delta x^2\right)$$

$$\Downarrow$$

$$\left(U_{M+2}\right)^n = \left(U_M\right)^n + 2\cdot\Delta x\cdot h^n$$

So at the final step in space we use the following modified formula

Modified formula $$\left(U_{M+1}\right)^{n+1} = 2\cdot r\cdot U_M{}^n + \left(U_{M+1}\right)^n - 2\cdot r\cdot\left(U_{M+1}\right)^n + 2\cdot r\cdot\Delta x\cdot h^n + \sigma\cdot\Delta t\cdot\left(U_{M+1}\right)^n$$ ✓

Where h^n is the neumann B.C

Question b - Which scheme is obtained for σ = 0 (diffusion equation)? And for v = 0 (reaction equation)?

Assumption

1) Still considering the FTCS scheme

$\sigma = 0$

We then get the following expression $u_t = \nu u_{xx}$

Which is known as the 1D diffusion equation.

The discrete probem is defined by the following:

Discrete problem $$\frac{u_k{}^{n+1} - u_k{}^n}{\Delta t} = \nu\cdot\frac{\left(u_{k-1}\right)^n - 2u_k{}^n + \left(u_{k+1}\right)^n}{(\Delta x)^2} + \tau_k{}^n$$

Neglecting the truncation error we get the numerical problem

$$u_k{}^{n+1} = \Delta t\cdot\left[\frac{u_k{}^n}{\Delta t} + \frac{\nu\cdot\left[\left(u_{k-1}\right)^n + \left(u_{k+1}\right)^n - 2\cdot u_k{}^n\right]}{\Delta x^2}\right]$$

$$\Downarrow$$

$$u_k{}^{n+1} = u_k{}^n + \frac{\nu\cdot\Delta t}{(\Delta x)^2}\left[\left(u_{k-1}\right)^n + \left(u_{k+1}\right)^n - 2\cdot u_k{}^n\right]$$

$$\Downarrow$$

Numerical problem $$U_k{}^{n+1} = r\cdot\left(U_{k+1}\right)^n + (1 - 2r)\cdot U_k{}^n + r\cdot\left(U_{k-1}\right)^n$$

With this we can still use the FTCS methos to solve the diffusion problem.

$v = 0$

The discrete problem is as followed $$\frac{u_k{}^{n+1} - u_k{}^n}{\Delta t} = \sigma\cdot u_k{}^n + \tau_k{}^n$$

Neglecting the truncation error we get the numerical problem

$$U_k{}^{n+1} - U_k{}^n = \Delta t\cdot\sigma\cdot U_k{}^n$$

$$\Downarrow$$

$$U_k{}^{n+1} = \Delta t\cdot\sigma\cdot U_k{}^n + U_k{}^n$$

By looking at the numerical problem we can see that there is no need for boundary conditions to compute the solution at the next time step. why ?

Question c - Take v = 0.1, σ = -0.1, Δx = 0.25 and Δt = 0.1, and compute two time steps with the explicit scheme proposed in section a. Are the obtained results reasonable? Discuss with the help of the graphic of the profile of u.

Diffusion coefficient $\nu := 0.1$

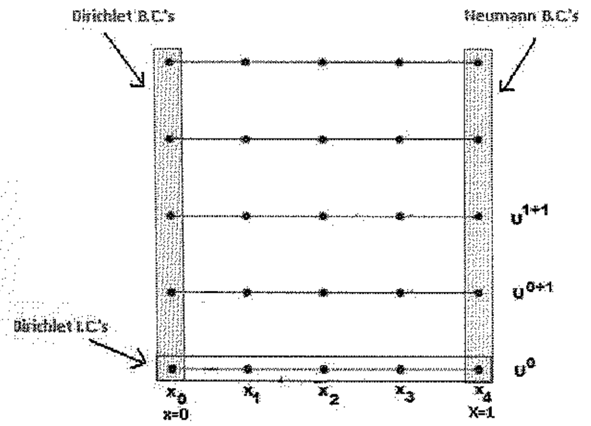
Reaction coefficient $\sigma := -0.1$

Discretization-step in space $\Delta x := 0.25$

Discretization-step in time $\Delta t := 0.1$

First I calculate the value of s $r := \nu\cdot\frac{\Delta t}{(\Delta x)^2}$ $r = 0.16$

This is smaller than 1/2 meaning that the solution is stable!



The above figure is used when computing the solutions at the 2 time steps.

1st time step

Given data $x = \frac{1}{4}$

$(U_0)^0 = 0$ Dirichlet B.C/initial condition

$(U_1)^0 = (4x - 1)$ Initial condition

$(U_2)^0 = \left(-4\frac{1}{2} + 3\right)$ Initial condition

Formula used $U_k^{n+1} = r\cdot(U_{k+1})^n + (1 - 2r)\cdot U_k^n + r\cdot(U_{k-1})^n + \sigma\cdot\Delta t\cdot U_k^n$

Solution at node 7 $(U_1)^{0+1} = r\cdot(U_2)^0 + (1 - 2r)\cdot(U_1)^0 + r\cdot(U_0)^0 + \sigma\cdot\Delta t\cdot(U_1)^0$

$(U_1)^{0+1} = 0.16$

Given data $x = \frac{1}{2}$

$(U_1)^0 = 4\frac{1}{4} - 1$ Initial condition

$(U_2)^0 = -4x + 3$ Initial condition

$(U_3)^0 = 0$ Initial condition

Formula used $U_k^{n+1} = r\cdot(U_{k+1})^n + (1 - 2r)\cdot U_k^n + r\cdot(U_{k-1})^n + \sigma\cdot\Delta t\cdot U_k^n$

Solution at node 8 $(U_2)^{0+1} = r\cdot(U_3)^0 + (1 - 2r)\cdot(U_2)^0 + r\cdot(U_1)^0 + \sigma\cdot\Delta t\cdot(U_2)^0$

$(U_2)^{0+1} = 0.67$

Given data $x = \frac{3}{4}$

$(U_2)^0 = -4\frac{1}{2} + 3$ Initial condition

$(U_3)^0 = 0$ Initial condition

$(U_4)^0 = 0$ Initial condition

Formula used $U_k^{n+1} = r\cdot(U_{k+1})^n + (1 - 2r)\cdot U_k^n + r\cdot(U_{k-1})^n + \sigma\cdot\Delta t\cdot U_k^n$

Solution at node 9 $(U_3)^{0+1} = r\cdot(U_4)^0 + (1 - 2r)\cdot(U_3)^0 + r\cdot(U_2)^0 + \sigma\cdot\Delta t\cdot(U_3)^0$

$(U_3)^{0+1} = 0.16$

Given data $x = 1$

$(U_3)^0 = 0$ Initial condition

$(U_4)^0 = 0$ Initial condition

$h_n = 0$ Neumann B.C.

Formula used $(U_{M+1})^{n+1} = 2\cdot r\cdot U_M^n + (U_{M+1})^n - 2\cdot r\cdot(U_{M+1})^n + 2\cdot r\cdot\Delta x\cdot h^n + \sigma\cdot\Delta t\cdot(U_{M+1})^n$

Solution at node 10 $(U_4)^{0+1} = 2\cdot r\cdot(U_3)^0 + (U_4)^0 - 2\cdot r\cdot(U_4)^0 + 2\cdot r\cdot\Delta x\cdot h_n + \sigma\cdot\Delta t\cdot(U_4)^0$

$(U_4)^{0+1} = 0$

2nd time step

Given data $x = \frac{1}{4}$

$(U_0)^{0+1} = 0$ Dirichlet B.C.

$(U_1)^{0+1} = 0.16$ From time step 1

$(U_2)^{0+1} = 0.67$ From tme step 1

Formula used $U_k^{n+1} = r\cdot\left(U_{k+1}\right)^n + (1-2r)\cdot U_k^n + r\cdot\left(U_{k-1}\right)^n + \sigma\cdot\Delta t\cdot U_k^n$

Solution at node 12 $(U_1)^{1+1} = r\cdot(U_2)^{0+1} + (1-2r)\cdot(U_1)^{0+1} + r\cdot(U_0)^{0+1} + \sigma\cdot\Delta t\cdot(U_1)^{0+1}$

$(U_1)^{1+1} = 0.214$

Given data $x = \frac{1}{2}$

$(U_1)^{0+1} = 0.16$ From time step 1

$(U_2)^{0+1} = 0.67$ From tme step 1

$(U_3)^{0+1} = 0.16$ From tme step 1

Formula used $U_k^{n+1} = r\cdot\left(U_{k+1}\right)^n + (1-2r)\cdot U_k^n + r\cdot\left(U_{k-1}\right)^n + \sigma\cdot\Delta t\cdot U_k^n$

Solution at node 13 $(U_2)^{1+1} = r\cdot(U_3)^{0+1} + (1-2r)\cdot(U_2)^{0+1} + r\cdot(U_1)^{0+1} + \sigma\cdot\Delta t\cdot(U_2)^{0+1}$

$(U_2)^{1+1} = 0.5$

Given data $x = \frac{3}{4}$

$(U_2)^{0+1} = 0.67$ From time step 1

$(U_3)^{0+1} = 0.16$ From tme step 1

$(U_4)^{0+1} = 0$ From tme step 1

Formula used $U_k^{n+1} = r\cdot\left(U_{k+1}\right)^n + (1-2r)\cdot U_k^n + r\cdot\left(U_{k-1}\right)^n + \sigma\cdot\Delta t\cdot U_k^n$

Solution at node 14 $(U_3)^{1+1} = r\cdot(U_4)^{0+1} + (1-2r)\cdot(U_3)^{0+1} + r\cdot(U_2)^{0+1} + \sigma\cdot\Delta t\cdot(U_3)^{0+1}$

$(U_3)^{1+1} = 0.214$

Given data $x = 1$

$(U_3)^{0+1} = 0.16$ From tme step 1

$(U_4)^{0+1} = 0$ From tme step 1

$h_n = 0$ Neumann B.C.

Formula used $\left(U_{M+1}\right)^{n+1} = 2\cdot r\cdot U_M^n + \left(U_{M+1}\right)^n - 2\cdot r\cdot\left(U_{M+1}\right)^n + 2\cdot r\cdot\Delta x\cdot h^n + \sigma\cdot\Delta t\cdot\left(U_{M+1}\right)^n$

Solution at node 15 $(U_4)^{1+1} = 2\cdot r\cdot(U_3)^{0+1} + (U_4)^{0+1} - 2\cdot r\cdot(U_4)^{0+1} + 2\cdot r\cdot\Delta x\cdot h_n + \sigma\cdot\Delta t\cdot(U_4)^{0+1}$

$(U_4)^{1+1} = 0.051$

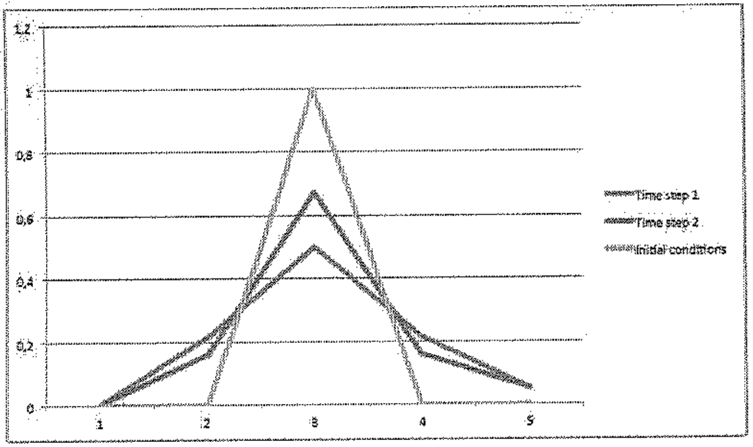
<u>Solution vector</u>

$$U = \begin{bmatrix} (U_0)^0 \\ (U_1)^0 \\ (U_2)^0 \\ (U_3)^0 \\ (U_4)^0 \\ (U_0)^{0+1} \\ (U_1)^{0+1} \\ (U_2)^{0+1} \\ (U_3)^{0+1} \\ (U_4)^{0+1} \\ (U_0)^{1+1} \\ (U_1)^{1+1} \\ (U_2)^{1+1} \\ (U_3)^{1+1} \\ (U_4)^{1+1} \end{bmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0.16 \\ 0.67 \\ 0.16 \\ 0 \\ 0 \\ 0.214 \\ 0.5 \\ 0.214 \\ 0.051 \end{pmatrix}$$

Graph of problem

The graph below shows the initial conditions and the solutions at both time step 1 and 2.
The scheme, with the choosen dicretization in both time and space, are behaving as expected, since it seems to be stable.

The solution is not that accurate, we can improve that by choosing Δx, Δt and v such that the factor r = 1/2.) ??
That is the best we can do when using the scheme FCTS.
To get solutions that are more accurate we can chose to use the BTCS or Crank-Nicolson method. Thay are more expensive in computational cost, but both are unconditionally stable.



Question d - Propose an implicit finite difference scheme to solve the PDE with boundary conditions and initial conditions. Detail how the boundary conditions are treated, the structure of the matrix and the most suitable method to solve the linear system of equations.

The PDE can be solved by using the implicit euler method BTCS. This is more expensive in computational cost than the FTCS. But on the other hand it is unconditionally stable.

To solve the row U^{n+1} there must be drawn up a linear system of equations. The linear system can be written with matrix and vectors.

To write the implicit method we have to make som approximations. These are shown below:

Approximations

$$\frac{d}{dt}(u_i)^{n+1} \approx \frac{(u_k)^{n+1} - (u_k)^n}{\Delta t} + \tau$$

$$\nu \cdot \frac{d^2}{dx^2}(u_k)^{n+1} \approx \nu \cdot \frac{(u_{k-1})^{n+1} - 2\cdot(u_k)^{n+1} + (u_{k+1})^{n+1}}{\Delta x^2} + \tau$$

$$\sigma\cdot(u_k)^{n+1} \approx \sigma\cdot(u_k)^{n+1} + \tau$$

By substituting the equations we can now write the implicit method

Implicit method (BTCS) $$\frac{(U_k)^{n+1} - (U_k)^n}{\Delta t} = \nu \cdot \frac{(U_{k-1})^{n+1} - 2\cdot(U_k)^{n+1} + (U_{k+1})^{n+1}}{\Delta x^2} + \sigma\cdot(U_k)^{n+1}$$

By neglecting the truncation error and make som basic algebra we can write the numerical problem.

$$(U_k)^{n+1} - (U_k)^n = \nu \cdot \frac{\Delta t}{\Delta x^2}\left[(U_{k-1})^{n+1} - 2\cdot(U_k)^{n+1} + (U_{k+1})^{n+1}\right] + \Delta t\cdot\sigma\cdot(U_k)^{n+1}$$

Setting the r factor $$r = \nu \cdot \frac{\Delta t}{\Delta x^2}$$

$$-(U_k)^n = -(U_k)^{(n+1)} + r\cdot\left[(U_{k-1})^{n+1} - 2\cdot(U_k)^{n+1} + (U_{k+1})^{n+1}\right] + \Delta t\cdot\sigma\cdot(U_k)^{n+1}$$

$$\Downarrow$$

$$(U_k)^n = (U_k)^{(n+1)} - r\cdot\left[(U_{k-1})^{n+1} - 2\cdot(U_k)^{n+1} + (U_{k+1})^{n+1}\right] - \Delta t\cdot\sigma\cdot(U_k)^{n+1}$$

$$\Downarrow$$

$$(U_k)^n = -r\cdot(U_{k-1})^{n+1} + (1 + 2\cdot r - \Delta t\cdot\sigma)\cdot(U_k)^{n+1} - r(U_{k+1})^{n+1}$$

$$\Downarrow$$

$$(U_k)^n = -r\cdot(U_{k-1})^{n+1} + (1 + 2\cdot r - \Delta t\cdot\sigma)\cdot(U_k)^{n+1} - r(U_{k+1})^n$$

$$\Downarrow$$

Numerical problem $$U_k{}^n = -r\cdot(U_{k-1})^{n+1} + (1 + 2\cdot r - \Delta t\cdot\sigma)\cdot(U_k)^{n+1} - r(U_{k+1})^{n+1}$$

For an example we use the same step in space as in question c)

Since this is an implicit method, we now establish 4 equations to solve the first time step. They are show below:

1st equation $U_1{}^0 = -r\cdot\left(U_0\right)^{0+1} + (1 + 2\cdot r - \Delta t\cdot\sigma)\cdot\left(U_1\right)^{0+1} - r\left(U_2\right)^{0+1}$

2nd equation $U_2{}^0 = -r\cdot\left(U_1\right)^{0+1} + (1 + 2\cdot r - \Delta t\cdot\sigma)\cdot\left(U_2\right)^{0+1} - r\left(U_3\right)^{0+1}$

3rd equation $U_3{}^0 = -r\cdot\left(U_2\right)^{0+1} + (1 + 2\cdot r - \Delta t\cdot\sigma)\cdot\left(U_3\right)^{0+1} - r\left(U_4\right)^{0+1}$

4th equation $U_4{}^0 = -r\cdot\left(U_3\right)^{0+1} + (1 + 2\cdot r - \Delta t\cdot\sigma)\cdot\left(U_4\right)^{0+1} - r\left[\left(U_3\right)^{0+1} + 2\cdot\Delta x\cdot u_x(1,t)\right]$

$$\Downarrow$$

$$U_4{}^0 = -2r\cdot\left(U_3\right)^{0+1} + (1 + 2\cdot r - \Delta t\cdot\sigma)\cdot\left(U_4\right)^{0+1} - 2\cdot r\cdot\Delta x\cdot u_x(1,t)$$

$$\Downarrow$$

$$U_4{}^0 + 2\cdot r\cdot\Delta x\cdot u_x(1,t) = -2r\cdot\left(U_3\right)^{0+1} + (1 + 2\cdot r - \Delta t\cdot\sigma)\cdot\left(U_4\right)^{0+1}$$

Then I set the equations on matrix and vector form. From that i can compute my unknown solutions

System to be solved $A\cdot U^{n+1} = I\cdot U^n + F$

Where

$$A = \begin{bmatrix} (1 + 2\cdot r - \Delta t\cdot\sigma) & -r & 0 & 0 \\ -r & (1 + 2\cdot r - \Delta t\cdot\sigma) & -r & 0 \\ 0 & -r & (1 + 2\cdot r - \Delta t\cdot\sigma) & -r \\ 0 & 0 & -2r & (1 + 2\cdot r - \Delta t\cdot\sigma) \end{bmatrix}$$

$$I = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$U^{n+1} = \begin{bmatrix} \left(U_1\right)^{0+1} \\ \left(U_2\right)^{0+1} \\ \left(U_3\right)^{0+1} \\ \left(U_4\right)^{0+1} \end{bmatrix}$$

$$F = \begin{pmatrix} r\cdot u(0,t) \\ 0 \\ 0 \\ 2\cdot r\cdot\Delta x\cdot u_x(1,t) \end{pmatrix}$$

Then we evaluate which method is best for solving the linear system.

Direct methods:

- Gauess elimination is not very usefull because it is necessary to make row operations for every time step
- Doolittle and Crout factorization because there is created a lower and upper triangular matrix there can be used for every time step. The difference between this scheme: If L has ones on its diagonal, then you have done a Doolittle
- factorisation. If U has ones on its diagonal, then you've done a Crout factorisation.
- Cholesky is not usefull because the A matrix is not symmetric

Iterative method

- Jacobi can be used when the A matrix is diagonally dominant
- Gauss-Seidel can be used when the A matrix is diagonally dominant or symmetric and positive difinite.
- Gauss-Siedel is faster than Jacobi. And depending on how precise a solution that is required it is olso faster than Doolittle and Crout where it is required to solve 2 linear systems for each time step. And with Gauss-Seidel it is possible to obtain a approximation by solving one linear system.
- Conjugate methods is namely used if A is symmetric

Summery: if we assume the A matrix above and two time steps. We get the following calculation steps

Doolittle or Crout:

1st time step	3 row operations + 2*4 equations
2nd time step	2*4 equations

Gauss-Seidel with 2 iterations

1st time step	2*4 equations
2nd time step	2*4 equations

So we will use Doolittle or Crout because it give us the precise solution by not many more calculations

EXERCISES 3

STUDENT 1

(NOTABLE)

ETHODS for PDEs
Master of Science in Computational Mechanics
Fall Semester 2013
Homework 3: Finite Differences

For the numerical modeling of a new technique of contamination control, it is interesting to solve the diffusion-reaction PDE

$$u_t = \nu u_{xx} + \sigma u \quad \text{in} \quad x \in (0,1), t > 0 \tag{1}$$

with boundary conditions

$$u(0,t) = 0 \quad \text{and} \quad u_x(1,t) = 0 \tag{2}$$

and the initial condition

$$u(x,0) = \begin{cases} 0 & \text{for} \quad x < 1/4 \\ 4x-1 & \text{for} \quad 1/4 \leq x < 1/2 \\ -4x+1 & \text{for} \quad 1/2 \leq x < 3/4 \\ 0 & \text{for} \quad 3/4 \leq x \end{cases} \tag{3}$$

In the PDE (1), $\nu > 0$ is the diffusion coefficient and $\sigma < 0$ is the reaction coefficient. Both coefficients can be considered constant.

a) Propose an explicit finite difference scheme for the solution of the PDE (1) with boundary conditions (2) and initial condition (3). Detail the numerical treatment of boundary conditions.

b) Which scheme is obtained for $\sigma = 0$ (diffusion equation)? And for $\nu = 0$ (reaction equation)?

c) Take $\nu = 0.1$, $\sigma = -0.1$, $\Delta x = 0.25$ and $\Delta t = 0.1$, and compute two time steps with the explicit scheme proposed in section a. Are the obtained results reasonable? Discuss with the help of the graphic of the profile of u.

d) Propose an implicit finite difference scheme to solve the PDE (1) with boundary conditions (2) and initial condition (3). Detail how are boundary conditions treated, the structure of the matrix and the most suitable method to solve the linear system of equations.

a) Solution:

We ~~apply~~ discretize the time-space domain with a uniform grid as follows:

$$\begin{aligned} x_i &= x_0 + i\Delta x \quad i = 0,\ldots,M+1 \\ t^n &= t^0 + n\Delta t \quad n = 0,\ldots \end{aligned} \tag{4}$$

In order to develop the explicit scheme we apply the following numerical approximations using Taylor expansion. Forward in time and centered in space:

$$\left.\frac{\partial u}{\partial t}\right|_i^n = \frac{u_i^{n+1} - u_i^n}{\Delta t} - \frac{1}{2}\Delta t \left.\frac{\partial^2 u}{\partial t^2}\right|_i^n + \mathcal{O}(\Delta t^2) \tag{5}$$

10

$$\left.\frac{\partial^2 u}{\partial x^2}\right|_i^n = \frac{u_{i-1}^n - 2u_i^n + u_{i+1}^n}{\Delta x^2} - \frac{1}{12}\Delta x^2 \left.\frac{\partial^4 u}{\partial x^4}\right|_i^n + \mathcal{O}(\Delta x^4) \tag{6}$$

In the case of *FTCS - explicit scheme* the equation is imposed at node *(i,n)* of the grid. In this case we replace u with the nodal value U_i^n.

$$u|_i^n = U_i^n \tag{7}$$

In this case (1) becomes:

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} = \nu \frac{u_{i-1}^n - 2u_i^n + u_{i+1}^n}{\Delta x^2} + \sigma u_i^n + \mathcal{T}_i(\Delta t, \Delta x^2) \tag{8}$$

$$u_i^{n+1} - u_i^n = \nu \frac{\Delta t}{\Delta x^2}\left(u_{i-1}^n - 2u_i^n + u_{i+1}^n\right) + \sigma \Delta t u_i^n + \mathcal{T}_i(\Delta t, \Delta x^2) \tag{9}$$

We define: $r = \nu \frac{\Delta t}{\Delta x^2}$

$$u_i^{n+1} = r u_{i-1}^n + (1 + \sigma\Delta t - 2r)\, u_i^n + r u_{i+1}^n + \mathcal{T}_i(\Delta t, \Delta x^2) \tag{10}$$

Neglecting the *truncation error* $\mathcal{T}_i$ we obtain the explicit scheme:

$$U_i^{n+1} = r U_{i-1}^n + (1 + \sigma\Delta t - 2r)\, U_i^n + r U_{i+1}^n \tag{11}$$

It is straightforward to see how we apply the *Dirichlet boundary condition* (DBC):

$$u(0,t) = 0; \quad U_0^n = g^n = 0 \quad n > 0; \tag{12}$$

In this case the value of the DBC is already given in the form of g^n we do not need to compute this value. Considering this we set the starting value of i in the numerical scheme to $i = 1, \ldots$.

In the case of the Neumann boundary condition NBC the prescribed value cannot be used in an explicit way, as a result we have to compute the value of this boundary node U_{M+1}^n for all $n > 0$.

$$U_{M+1}^{n+1} = r U_M^n + (1 + \sigma\Delta t - 2r)\, U_{M+1}^n + r U_{M+2}^n \tag{13}$$

The numerical scheme for $i = M+1$ leads to a node outside of our defined domain (1,M+1). To solve this problem we have to introduce a fictious node at U_{M+2}^n.

$$\left.\frac{\partial u}{\partial n}\right|_{M+1}^P = \left.\frac{\partial u}{\partial n}\right|_{M+1}^P = \frac{u_{M+2}^P - u_M^P}{2\Delta x} + \mathcal{O}(\Delta x^2) = h^P \tag{14}$$

$$\begin{aligned} U_{M+2}^P &= U_M^P + \cancel{2\Delta x h^P} = U_M^P \quad ✓ \\ u_x(1,t) &= h^n = 0 \quad n > 0 \end{aligned} \tag{15}$$

Substituting (15) into (13):

$$U_{M+1}^{n+1} = 2rU_M^n + (1+\sigma\Delta t - 2r)\,U_{M+1}^n \tag{16}$$

We obtain the complete definition of the numerical scheme:

$$\boxed{\begin{array}{lll} U_i^{n+1} = rU_{i-1}^n + (1+\sigma\Delta t - 2r)\,U_i^n + rU_{i+1}^n & i=1,\ldots,M & n\geq 0 \\ U_0^{n+1} = g^{n+1} = 0 & n \geq 0 & \\ U_{M+1}^{n+1} = 2rU_M^n + (1+\sigma\Delta t - 2r)\,U_{M+1}^n & n\geq 0 & \\ U_i^0 = f_i & & \\ f_i = \begin{cases} 0 & \text{for } \ i\Delta x < 1/4 \\ 4x-1 & \text{for } \ 1/4 \leq i\Delta x < 1/2 \\ -4x+1 & \text{for } \ 1/2 \leq i\Delta x < 3/4 \\ 0 & \text{for } \ 3/4 \leq i\Delta x \end{cases} & i=0,\ldots,M+1 & \end{array}} \tag{17}$$

b) Solution:

For $\sigma = 0$ (diffusion equation) we obtain a *parabolic equation* of the form:

$$\begin{array}{ll} u_t = \nu u_{xx} & in \quad x\in(0,1),\ t>0 \\ u(0,t) = 0 & \\ u_x(1,t) = 0 & \end{array} \tag{18}$$

In this case the numerical scheme (17) will take the *FTCS* form:

$$\begin{array}{lll} U_i^{n+1} = rU_{i-1}^n + (1-2r)\,U_i^n + rU_{i+1}^n & i=1,\ldots,M & n\geq 0 \\ U_0^{n+1} = g^{n+1} = 0 & n\geq 0 & \\ U_{M+1}^{n+1} = 2rU_M^n + (1-2r)\,U_{M+1}^n & n\geq 0 & \\ U_i^0 = f_i & & \end{array} \tag{19}$$

For $\nu = 0$ (1) will take the form of an *ODE*:

$$\begin{array}{ll} u_t = \sigma u & in \quad x\in(0,1),\ t>0 \\ u(0,t) = 0 & \\ u_x(1,t) = 0 & \end{array} \tag{20}$$

The numerical scheme (17) takes the form of the *Forward Euler method*:

$$\begin{array}{lll} U_i^{n+1} = (1+\sigma\Delta t)\,U_i^n & i=1,\ldots,M & n\geq 0 \\ U_0^{n+1} = g^{n+1} = 0 & n\geq 0 & \\ U_{M+1}^{n+1} = (1+\sigma\Delta t)\,U_{M+1}^n & n\geq 0 & \\ U_i^0 = f_i & & \end{array} \tag{21}$$

c) Solution:

For $\nu = 0.1$, $\sigma = -0.1$, $\Delta x = 0.25$ and $\Delta t = 0.1$ (17) takes the form:

$$r = \nu\frac{\Delta t}{\Delta x^2} = 0.1\frac{0.1}{0.25^2} = 0.16 \tag{22}$$

$$\begin{array}{lll} U_i^{n+1} = 0.16\left(U_{i-1}^n + U_{i+1}^n\right) + 0.67U_i^n & i=1,\ldots,M & n\geq 0 \\ U_0^{n+1} = 0 & n\geq 0 & \\ U_{M+1}^{n+1} = 0.32U_M^n + 0.67U_{M+1}^n & n\geq 0 & \\ \mathbf{U}^0 = \begin{bmatrix}0 & 0 & 1 & 0 & 0\end{bmatrix}^T & & \end{array} \tag{23}$$

- **Step 1:**

$$\begin{aligned} U_1^1 &= 0.16(U_0^0 + U_2^0) + 0.67U_1^0 = 0.16(0+1) + 0.67\cdot 0 = 0.16 \\ U_2^1 &= 0.16(U_1^0 + U_3^0) + 0.67U_2^0 = 0.16(0+0) + 0.67\cdot 1 = 0.67 \\ U_3^1 &= 0.16(U_2^0 + U_4^0) + 0.67U_3^0 = 0.16(1+0) + 0.67\cdot 0 = 0.16 \\ U_4^1 &= 0.32U_3^0 + 0.67U_4^0 = 0.32\cdot 0 + 0.67\cdot 0 = 0 \\ \mathbf{U}^1 &= \begin{bmatrix}0 & 0.16 & 0.67 & 0.16 & 0\end{bmatrix}^T \end{aligned}$$

- **Step 2:**

$$\begin{aligned} U_1^2 &= 0.16(U_0^1 + U_2^1) + 0.67U_1^1 = 0.16(0+0.67) + 0.67\cdot 0.16 = 0.2144 \\ U_2^2 &= 0.16(U_1^1 + U_3^1) + 0.67U_2^1 = 0.16(0.16+0.16) + 0.67\cdot 0.67 = 0.5001 \\ U_3^2 &= 0.16(U_2^1 + U_4^1) + 0.67U_3^1 = 0.16(0.67+0) + 0.67\cdot 0.16 = 0.2144 \\ U_4^2 &= 0.32U_3^1 + 0.67U_4^1 = 0.32\cdot 0.16 + 0.67\cdot 0 = 0.0512 \end{aligned}$$

$$\mathbf{U}^2 = \begin{bmatrix}0 & 0.2114 & 0.5001 & 0.2155 & 0.0512\end{bmatrix}^T$$

Finally we obtain for 3 time steps:

$$\boxed{\mathbf{U} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0.16 & 0.2114 \\ 1 & 0.67 & 0.5001 \\ 0 & 0.16 & 0.2114 \\ 0 & 0 & 0.0512 \end{bmatrix}} \tag{24}$$

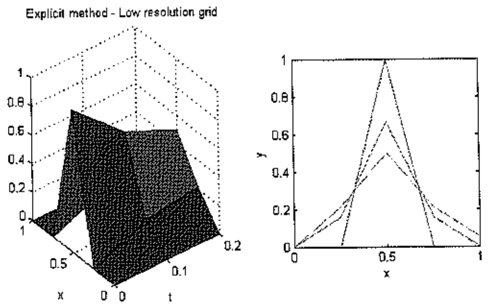


Figure 1: Graphical profile of U

We can draw the following conclusions based on the graphical profile of u:

- The method is stable, given $r = 0.16 \leq 1/2$. There are no oscillations and a convergence trend can be observed. ? What do you mean?
- The boundary conditions are respected. At $x = 0$ the DBC is respected, while the NBC is computed for every step (at $n = 2\Delta t \quad U^n_{M+1} \neq 0$).
- We can see that the initial value of $U^0_3 = 1$ diffuses throughout the interval.
- Due to the presence of the reaction term with the advance of time elements are being consumed: $U^0_3 > U^2_3$.

We can state that the method is behaving as we expected, the results are reasonable. ✓

d) Solution:

In order to obtain the implicit method we have to apply *backward time* we impose the equation in node U_i^{n+1} instead of U_i^n as we did in the previous case.

(5), (6) take the following forms:

$$\left.\frac{\partial u}{\partial t}\right|_i^{n+1} = \frac{u_i^{n+1} - u_i^n}{\Delta t} - \frac{1}{2}\Delta t \left.\frac{\partial^2 u}{\partial t^2}\right|_i^{n+1} + \mathcal{O}(\Delta t^2) \tag{25}$$

$$\left.\frac{\partial^2 u}{\partial x^2}\right|_i^{n+1} = \frac{u_{i-1}^{n+1} - 2u_i^{n+1} + u_{i+1}^{n+1}}{\Delta x^2} - \frac{1}{12}\Delta x^2 \left.\frac{\partial^4 u}{\partial x^4}\right|_i^{n+1} + \mathcal{O}(\Delta x^4) \tag{26}$$

We impose the equation in not U_i^{n+1} so:

$$u\Big|_i^{n+1} = U_i^{n+1} \tag{27}$$

Substituting all (25), (26) and (27) into (1) we obtain:

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} = \nu \frac{u_{i-1}^{n+1} - 2u_i^{n+1} + u_{i+1}^{n+1}}{\Delta x^2} + \sigma u_i^{n+1} + \mathcal{T}_i(\Delta t, \Delta x^2) \tag{28}$$

Neglecting the *truncation error* $\mathcal{T}_i$ and collecting terms:

$$-rU_{i-1}^{n+1} + (1 + 2r - \sigma\Delta t)U_i^{n+1} - rU_{i+1}^{n+1} = U_i^n \tag{29}$$

The DBC can be used directly in the computation of elements U_1^{n+1}. In this case (29) is valid for $i = 1, \ldots, M$.

For the same reasons as stated at point *a)* the NBC cannot be used directly in the computation since it defines u_x over the boundary. We have to evaluate (29) at the boundary node $i = M+1$.

$$-rU_M^{n+1} + (1 + 2r - \sigma\Delta t)U_{M+1}^{n+1} - rU_{M+2}^{n+1} = U_{M+1}^n \tag{30}$$

In (31) the term U_{M+2}^{n+1} is outside our defined domain. It is a fictious node and it's approximation is defined by (15).

$$-2rU_M^{n+1} + (1 + 2r - \sigma\Delta t)U_{M+1}^{n+1} = U_{M+1}^n \tag{31}$$

The *implicit - BTCS* method takes the following form:

$$\boxed{\begin{array}{lll} -rU_{i-1}^{n+1} + (1 + 2r - \sigma\Delta t)U_i^{n+1} - rU_{i+1}^{n+1} = U_i^n & i = 1, \ldots, M & n \geq 0 \\ U_0^{n+1} = g^{n+1} = 0 & n \geq 0 & \\ U_{M+1}^{n+1} = -2rU_M^{n+1} + (1 + 2r - \sigma\Delta t)U_{M+1}^{n+1} & n \geq 0 & \\ U_i^0 = f_i & i = 0, \ldots, M+1 & \end{array}} \tag{32}$$

We can write the *implicit scheme* in the form of a matrix equation:

$$\mathbf{A} \cdot \mathbf{U}^{n+1} = \mathbf{I} \cdot \mathbf{U}^n + \mathbf{F} \tag{33}$$

Where the dimension of $\mathbf{A}$ is [M+1,M+1] since we compute the nodal values of the grid U_i^{n+1} $i = 1, \ldots, M+1$.

$$\mathbf{U}^P = \begin{bmatrix} U_1 & U_2 & \ldots & U_{M+1} \end{bmatrix}^T \tag{34}$$

When we reduce the size of the system from [M+2,M+2] to [M+1,M+1] (considering DBC at (0,t)) we have to conserve the effect of the boundary value $-rg^{n+1}$ on U_1^{n+1}. Thus we introduce $\mathbf{F}$ where $F_1 = rg^{n+1}$ whereas F_{M+1} would contain the boundary value of the approximation of U_{M+2}^{n+1}. In our case $F_{M+1} = -2\Delta x h^{n+1} = 0$.

$$\mathbf{F} = \begin{bmatrix} rg^{n+1} & 0 & \ldots & 0 & -2\Delta x h^{n+1} \end{bmatrix}^T \tag{35}$$

$$\mathbf{A} = \begin{bmatrix} (1 + 2r - \sigma\Delta t) & -r & & & \\ -r & (1 + 2r - \sigma\Delta t) & -r & & \\ & \ddots & \ddots & \ddots & \\ & & -r & (1 + 2r - \sigma\Delta t) & -r \\ & & & -2r & (1 + 2r - \sigma\Delta t) \end{bmatrix} \tag{36}$$

Since $\mathbf{A}$ is *tridiagonal* it is convenient to solve the system using the Thomas algorithm. ✓

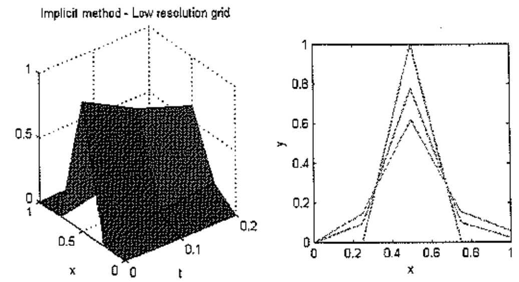


Figure 2: Graphical profile of **U**

We can observe a better treatment of the NBC at *x = 1*.

Appendix:

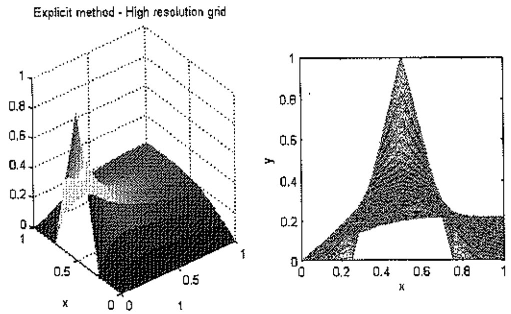


Figure 3: U obtained with a *high resolution* grid, explicit method.

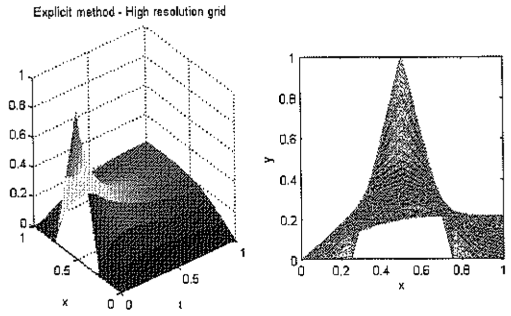


Figure 4: U obtained with a *high resolution* grid, implicit method.

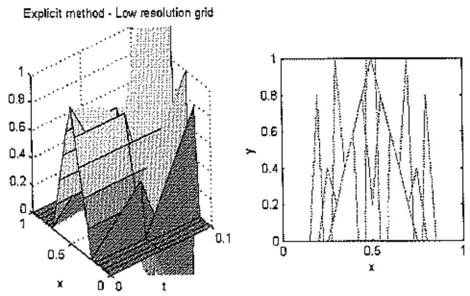


Figure 5: U obtained with a *low resolution* grid for which the explicit method is unstable.

Matlab code:

```
clc; close all; clear all;

% Mode: 1 - Explicit LD; 2 - Explicit HD; 3 - Implicit LD; 4 - Implicit HD
mode = 1;

% Numerical method for solving reaction diffusion equation
% ut = nu * uxx + sigma * u in 0,1 t > 0
% BC: DBC u(0,t) = 0; NBC ux(1,t) = 0
% IC: 0 [0,0.25[ 4x-1 [0.25;0.5[ -4x+3 [0.5;0.75[ 0 [0.75;1]

switch mode
    case {1, 3}
    % Problem given values;
    a = 0; b = 1; Dx = 0.25; Dt = 0.1; m = (b-a)/Dx; nu = 0.1; sig = -0.1;
    tstep = 2;
    case {2, 4}
    % alternate resolution
    a = 0; b = 1; Dx = 0.125; Dt = 0.005; m = (b-a)/Dx; nu = 0.1; sig = -0.1;
    tstep = 200;
    case {5}
    a = 0; b = 1; Dx = 0.05; Dt = 0.05; m = (b-a)/Dx; nu = 0.1; sig = -0.1;
    tstep = 2;
end

t = zeros(1,tstep+1);
t = linspace(0,Dt*tstep,tstep+1);
xx = linspace(a,b,m+1);
% Solution matrix
U = zeros(m+1,tstep+1);

% Apply DBC
g = zeros(1,tstep+1);
% In our case this is enough else: h(1,:) = DBC ct; also U(1,:) = h;

% Apply IC
r = Dt/Dx^2;


figure(1),clf
subplot(1,2,1)
switch mode
    case {1, 2, 5}
    % Explicit method
    ttl = 'Explicit method -';
    U = Explicit(a,b,tstep,Dt,Dx,nu,sig,g);
    case {3, 4}
    % Implicit method
    ttl = 'Implicit method -';
    U = Implicit(a,b,tstep,Dt,Dx,nu,sig,g);
end

h = surf(t,xx,U), axis([0 Dt*tstep a b 0 1]);
if (mode == 2) | (mode == 4)
    set(h, 'linestyle','none');
    ttl = strcat(ttl,' High resolution grid');
else
    ttl = strcat(ttl,' Low resolution grid');
end
```

```
title(ttl);
xlabel('t');
ylabel('x');

subplot(1,2,2)
plot(xx,U)
xlabel('x'), ylabel('y')
axis square, axis([0 1 0 1])
```

STUDENT 2

(NOTABLE)

Page: 1/4

$$\begin{cases} u_t = \nu u_{xx} + \sigma u & \text{in } x \in (0,1),\ t>0,\ \nu>0,\ \sigma<0. \\ u(0,t) = 0; \\ u_x(1,t) = 0; \\ u(x,0) \begin{cases} 0 & x < 1/4 \\ 4x-1 & 1/4 \leq x < 1/2 \\ -4x+3 & 1/2 \leq x < 3/4 \\ 0 & x \geq 3/4 \end{cases} \end{cases}$$

ν: diffusion coefficient

σ: reaction coefficient

a) Explicit scheme: Forward in time, centered in space

$$\left.\frac{\partial u}{\partial t}\right|_i^n = \frac{u_i^{n+1} - u_i^n}{\Delta t} + O(\Delta t)$$

$$\left.\frac{\partial^2 u}{\partial x^2}\right|_i^n = \frac{u_{i-1}^n - 2u_i^n + u_{i+1}^n}{\Delta x^2} + O(\Delta x^2)$$

$$\left.\frac{\partial u}{\partial t}\right|_i^n = \nu \left.\frac{\partial^2 u}{\partial x^2}\right|_i^n + \sigma\, u_i^n;$$

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} + O(\Delta t) = \nu\, \frac{u_{i-1}^n - 2u_i^n + u_{i+1}^n}{\Delta x^2} + O(\Delta x^2) + \sigma u_i^n;$$

Neglecting truncation errors.

$$\frac{U_i^{n+1} - U_i^n}{\Delta t} = \nu\, \frac{U_{i-1}^n - 2U_i^n + U_{i+1}^n}{\Delta x^2} + \sigma U_i^n;$$

$$U_i^{n+1} - U_i^n = \nu \frac{\Delta t}{\Delta x^2} \cdot \left(U_{i-1}^n - 2U_i^n + U_{i+1}^n\right) + \Delta t\, \sigma\, U_i^n$$

$$\alpha = \nu \frac{\Delta t}{\Delta x^2}\ ;\ \beta = \Delta t\, \sigma$$

$$U_i^{n+1} = \alpha U_{i-1}^n + (1-2\alpha+\beta) U_i^n + \alpha U_{i+1}^n \quad i = 1 \dots m.$$ ✓

Since at $x=L$ there is a Neuman boundary condition:

$$\left.\frac{\partial u}{\partial x}\right|_{m+1}^n = \frac{u_{m+2}^n - u_m^n}{2\Delta x} + O(\Delta x^2) = 0;$$

Neglecting truncation errors.

$$\frac{U_{m+2}^n - U_m^n}{2\Delta x} = 0; \quad U_{m+2}^n = U_m^n \quad n \geq 1$$

Now the explicit numerical scheme can be written.

$$\boxed{\begin{array}{l} U_0^{n+1} = 0; \qquad n \geq 0 \\ U_i^{n+1} = \alpha U_{i-1}^n + (1-2\alpha+\beta) U_i^n + \alpha U_{i+1}^n\ ;\ n \geq 0,\ i=1 \dots m \\ U_{m+1}^{n+1} = \alpha U_m^n + (1-2\alpha+\beta) U_{m+1}^n + \alpha U_m^n; \quad n \geq 0 \\ U_i^0 \begin{cases} 0 & 0 \leq i\Delta x < 1/4 \\ 4i\Delta x - 1 & 1/4 \leq i\Delta x < 1/2 \\ -4i\Delta x + 3 & 1/2 \leq i\Delta x < 3/4 \\ 0 & 3/4 \leq i\Delta x \leq 1 \end{cases} \quad i = 0, 1, \dots, m+1 \end{array}}$$

And the following relation has been used: $X_i = X_0 + i\Delta x$,

$X_0 = 0 \rightarrow X_i = i\Delta x$ ✓

b) Diffusion equation: $\sigma = 0 \rightarrow \beta = 0$.

$$U_0^{n+1} = 0; \qquad n \geq 0$$

$$U_i^{n+1} = \alpha U_{i-1}^n + (1-2\alpha) U_i^n + \alpha U_{i+1}^n; \quad n \geq 0;\ i = 1, \dots m$$

$$U_{m+1}^{n+1} = \alpha U_m^n + (1-2\alpha) U_{m+1}^n + 2\alpha U_m^n; \quad n \geq 0$$

And same initial conditions as a).

Reaction equation: $\upsilon = 0 \rightarrow \alpha = 0$.

$$U_0^{n+1} = 0; \qquad n \geq 0$$

$$U_i^{n+1} = (1+\beta) U_i^n; \quad n \geq 0,\ i = 1 \dots m$$

$$U_{m+1}^{n+1} = (1+\beta) U_{m+1}^n; \quad n \geq 0.$$

And same initial conditions as a).

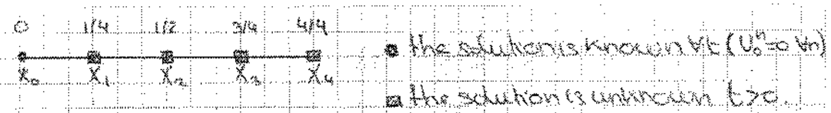
c) $\upsilon = 0'10$.

$\sigma = -0'10$.

$\Delta x = 0'25 \rightarrow m = (b-a)/\Delta x = 4$.

$\Delta t = 0'10 \rightarrow$ 2 time steps $\rightarrow$ final time: $0'20$.

$\alpha = \upsilon \dfrac{\Delta t}{\Delta x^2} = 0'16$; $\beta = -0'01$; $(1-2\alpha+\beta) = 0'67$.


0
1/4
1/2
3/4
4/4
x0
x1
x2
x3
x4
the solution is known ∀t ($U_0^n = 0\ \forall n$)
the solution is unknown t>0.

First step: n = 0.

$U_1^1 = 0'16 \cdot U_0^0 + 0'67 \cdot U_1^0 + 0'16 U_2^0 = 0'67 U_1^0 + 0'16 U_2^0 = 0'1600$.

$U_2^1 = 0'16 \cdot U_1^0 + 0'67 \cdot U_2^0 + 0'16 \cdot U_3^0 = 0'6700$;

$U_3^1 = 0'16 U_2^0 + 0'67 U_3^0 + 0'16 \cdot U_4^0 = 0'1600$;

$U_4^1 = 0'16 \cdot U_3^0 + 0'67 \cdot U_4^0 + 0'16 U_3^0 = 0'0000$;

Second step: n = 1.

$U_1^2 = 0'16 \cdot U_0 + 0'67 \cdot U_1^1 + 0'16 \cdot U_2^1 = 0'2144$;

$U_2^2 = 0'16 U_1^1 + 0'67 \cdot U_2^1 + 0'16 U_3^1 = 0'5001$;

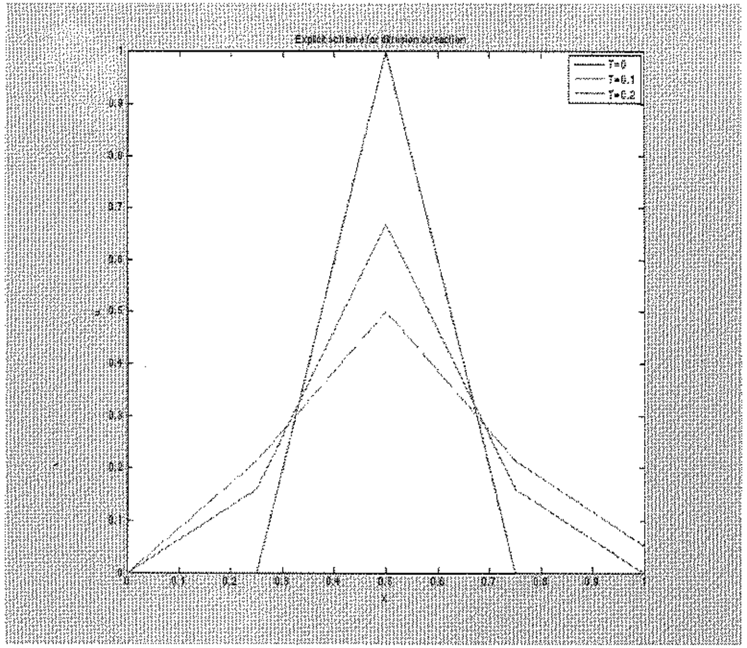
$U_3^2 = 0'16 U_2^1 + 0'67 U_3^1 + 0'16 U_4^1 = 0'2144$;

$U_4^2 = 0'16 \cdot U_3^1 + 0'67 \cdot U_4^1 + 0'16 \cdot U_3^1 = 0'0512$.

The solution is behaving as expected. The diffusion is lowering the peaks and spreading u along the domain whereas the reaction is consuming a small amount of u. This consumption is seen in the coefficient $(1-2\alpha+\beta)$ since $\beta<0$ is reducing the value of U as time increases. Besides since the flux of u at x=b is set to be 0, the amount of u at this point increases with time.

Are b.c. fulfilled?

Diffusion and reaction.


Explicit scheme for diffusion & reaction

---

d) Implicit scheme. Backward in time, centered in space.

$$\left.\frac{\partial u}{\partial t}\right|_i^{n+1} = \frac{u_i^{n+1} - u_i^n}{\Delta t} + \Theta(\Delta t);$$

$$\left.\frac{\partial^2 u}{\partial x^2}\right|_i^{n+1} = \frac{u_{i-1}^{n+1} - 2u_i^{n+1} + u_{i+1}^{n+1}}{\Delta x^2} + \Theta(\Delta x^2);$$

$$\left.\frac{\partial u}{\partial t}\right|_i^{n+1} = \nu \left.\frac{\partial^2 u}{\partial x^2}\right|_i^{n+1} + \sigma u_i^{n+1};$$

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} + \Theta(\Delta t) = \nu \frac{u_{i-1}^{n+1} - 2u_i^{n+1} + u_{i+1}^{n+1}}{\Delta x^2} + \sigma u_i^{n+1} + \Theta(\Delta x^2)$$

Neglecting truncation errors:

$$\frac{U_i^{n+1} - U_i^n}{\Delta t} = \nu \frac{U_{i-1}^{n+1} - 2U_i^{n+1} + U_{i+1}^{n+1}}{\Delta x^2} + \sigma U_i^{n+1}$$

$$U_i^{n+1} - U_i^n = \nu \frac{\Delta t}{\Delta x^2}\left(U_{i-1}^{n+1} - 2U_i^{n+1} + U_{i+1}^{n+1}\right) + \Delta t \sigma U_i^{n+1};$$

$$\alpha = \nu \frac{\Delta t}{\Delta x^2}; \quad \beta = \Delta t \sigma$$

$$U_i^{n+1} - U_i^n = \alpha\left(U_{i-1}^{n+1} - 2U_i^{n+1} + U_{i+1}^{n+1}\right) + \beta U_i^{n+1};$$

$$-\alpha U_{i-1}^{n+1} + (1+2\alpha-\beta)U_i^{n+1} - \alpha U_{i+1}^{n+1} = U_i^n; \quad i = 1, \ldots, m.$$

Treatment of the Neuman boundary condition at $x = b$.

$$\left.\frac{\partial u}{\partial x}\right|_{m+1}^{n+1} = \frac{u_{m+2}^{n+1} - u_m^n}{2\Delta x} + \Theta(\Delta x^2) = 0.$$

Neglecting truncation errors:

$$U_{m+2}^{n+1} - U_m^{n+1} = 0 \rightarrow U_{m+2}^{n+1} = U_m^{n+1} \quad n \geq 1.$$

Derivation of the system of equations to be solved:

$i = 1;$ $-\alpha U_0^{n+1} + (1+2\alpha-\beta)U_1^{n+1} - \alpha U_2^{n+1} = U_1^n$

U_0^{n+1} is known $\forall t$

$(1+2\alpha-\beta)U_1^{n+1} - \alpha U_2^{n+1} = U_1^n + \alpha U_0^{n+1}$

$i = 2;$ $-\alpha U_1^{n+1} + (1+2\alpha-\beta)U_2^{n+1} - \alpha U_3^{n+1} = U_2^n;$

$i = 3;$ $-\alpha U_2^{n+1} + (1+2\alpha-\beta)U_3^{n+1} - \alpha U_4^{n+1} = U_3^n;$

$\vdots$

$i = m;$ $-\alpha U_{m-1}^{n+1} + (1+2\alpha-\beta)U_m^{n+1} - \alpha U_{m+1}^{n+1} = U_m^n;$

$i = m+1;$ $-\alpha U_m^{n+1} + (1+2\alpha-\beta)U_{m+1}^{n+1} - \alpha U_{m+2}^{n+1} = U_{m+1}^n$

$U_{m+2}^{n+1} = U_m^{n+1}$

$-\alpha U_m^{n+1} + (1+2\alpha-\beta)U_{m+1}^{n+1} - \alpha U_m^{n+1} = U_{m+1}^n$

$-2\alpha U_m^{n+1} + (1+2\alpha-\beta)U_{m+1}^{n+1} = U_{m+1}^n;$

Writting the previous equations in matrix form is presented in the next page.

$$\underbrace{\begin{bmatrix} 1+2\alpha-\beta & -\alpha & 0 & 0 & 0 & 0 & \cdots & 0 \\ -\alpha & 1+2\alpha-\beta & -\alpha & 0 & 0 & 0 & \cdots & 0 \\ 0 & -\alpha & 1+2\alpha-\beta & -\alpha & 0 & 0 & \cdots & 0 \\ 0 & 0 & -\alpha & 1+2\alpha-\beta & -\alpha & 0 & \cdots & 0 \\ \vdots & \vdots & & \ddots & \ddots & \ddots & & \vdots \\ 0 & \cdots & & & 0 & -\alpha & 1+2\alpha-\beta & -\alpha \\ 0 & \cdots & & & & 0 & -2\alpha & 1+2\alpha-\beta \end{bmatrix}}_{A} \cdot \underbrace{\begin{bmatrix} U_1^{n+1} \\ U_2^{n+1} \\ U_3^{n+1} \\ \vdots \\ U_{m-1}^{n+1} \\ U_m^{n+1} \\ U_{m+1}^{n+1} \end{bmatrix}}_{U^{n+1}} = \underbrace{\begin{bmatrix} U_1^n \\ U_2^n \\ U_3^n \\ \vdots \\ U_{m-1}^n \\ U_m^n \\ U_{m+1}^n \end{bmatrix}}_{U^n} + \underbrace{\begin{bmatrix} \alpha U_0^{n+1} \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}}_{F}$$

$$A \cdot U^{n+1} = U^n + F$$

And same initial condition as the Explicit schema.

Due to the fact that there is a Neuman boundary condition at $x=b$, the matrix **A** is not symmetric. Moreover, α and β depend on ν, Δt and Δx, and σ and Δt respectively, and these parameter might vary for each particular problem. This makes not possible to guarantee that A is diagonally dominant. Therefore, the most suitable method to solve the system is:

* Case A: A is diagonally dominant.

The system is solved using Gauss-Seidel method.

* Case B: A is not diagonally dominant.

1°) Before computing the solution U at different time steps, and sice A is constant $\forall t$, Doolittle factorization method is applied to the matrix A, such that $A = L\cdot U$.

2°) At each time step, the solution is computed following the next numerical scheme:

$L\cdot\boxed{Y} = U^n + F$; $\boxed{Y}$ is computed with a Forward substitution.

$U\cdot\boxed{U^{n+1}} = Y$; $\boxed{U^{n+1}}$ is computed with a Backward substitution.

✓

Matlab code.

```
function U = parab_ex

% Constants
nu       = input('diffusion constant      = ');
sigma    = input('reaction constant       = ');
a        = input('a                       = ');
b        = input('b                       = ');
m        = input('Number of intervals     = ');
Ax       = (b-a)/m;
x        = [a:Ax:b];
npoints = length(x);
tfin     = input('Final time              = ');
npast    = input('Number of time steps    = ');
At       = tfin/npast;
t        = [0:At:npast*At];
r        = At/(Ax^2);
A        = nu*At/(Ax*Ax);
B        = sigma*At;

% IC
for i=1:npoints
    f(i) = 0;
end

for i=1:npoints
    if x(i)<(1/4)
        f(i) = 0;
    elseif x(i)>=(1/4) && x(i)<(1/2)
        f(i) = 4*x(i)-1;
    elseif x(i)>=(1/2) && x(i)<(3/4)
        f(i) = -4*x(i)+3;
    elseif x(i)>=(3/4) && x(i)<=1
        f(i) = 0;
    end
end

% Dirichlet BC at x = a.
g0 = 0;
[n0 m0] = size(g0);
if((n0==1) && (m0==1)), g = g0*ones(1,npast+1);
elseif((n0==1) && (m0==npast+1)), g = g0;
else error(strcat('Error in the boundary conditions in x=',num2str(a)))
end

% Initialize U matrix
U = zeros(m+1,npast+1);

% IC in U
for i=1:(m+1)
    U(i,1) = f(i);
end

% Dirichlet BC in U
for i=2:(npast+1)
    U(1,i) = g(i);
end

% Computing solution considering Neuman BC at x = b
for j=2:(npast+1)
    for i=2:m
        U(i,j) = A*U(i-1,j-1) + (1-2*A+B)*U(i,j-1) + A*U(i+1,j-1);
    end
    U(m+1,j) = A*U(m,j-1) + (1-2*A+B)*U(m+1,j-1) + A*U(m,j-1);
end

%Postprocess
plot(x,U);
xlabel('x');
ylabel('u');
axis square;
title('Explicit scheme for diffusion & reaction');
legend('T=0','T=0.1','T=0.2');
```

STUDENT 1
(EXCEL·LENT)

EXERCISES 3 yor

PDE- Exercise 3

$$u_t = \nu u_{xx} + \sigma u$$

$$u(0,t) = 0 \quad , \quad u_x(1,t) = 0$$

$$u(x,0) = \begin{cases} 0 & \text{for } x < 1/4 \\ 4x-1 & \text{for } 1/4 \leq x < 1/2 \\ -4x+3 & \text{for } 1/2 \leq x < 3/4 \\ 0 & \text{for } 3/4 \leq x \end{cases}$$

a) FTCS scheme

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} = \nu \frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{\Delta x^2} + \sigma u_i^n + O(\Delta t, \Delta x^2)$$

$$\Rightarrow U_i^{n+1} = \frac{\nu \Delta t}{\Delta x^2}(U_{i+1}^n - 2U_i^n + U_{i-1}^n) + (\sigma \Delta t + 1) U_i^n$$

for $u(0,t)=0$, $i=0$

$$U_0^n = 0 , \quad \forall n$$

for $u_x(1,t)=0$, $i=m$

$$\frac{U_{m+1}^n - U_{m-1}^n}{2\Delta x} = 0$$

$$\Rightarrow U_{m+1}^n = U_{m-1}^n$$

for $x=1$, $i=m$

$$U_m^{n+1} = 2\frac{\nu \Delta t}{\Delta x^2}(U_{m-1}^n - U_m^n) + (\sigma \Delta t + 1) U_m^n$$

✓

b)

- $\sigma = 0$

$$U_i^{n+1} = \frac{\nu \Delta t}{\Delta x^2}(U_{i+1}^n - 2U_i^n + U_{i-1}^n) + U_i^n$$

with

$$U_0^n = 0 \quad , \quad U_m^{n+1} = 2\frac{\nu \Delta t}{\Delta x^2}(U_{m-1}^n - U_m^n) + U_m^n$$

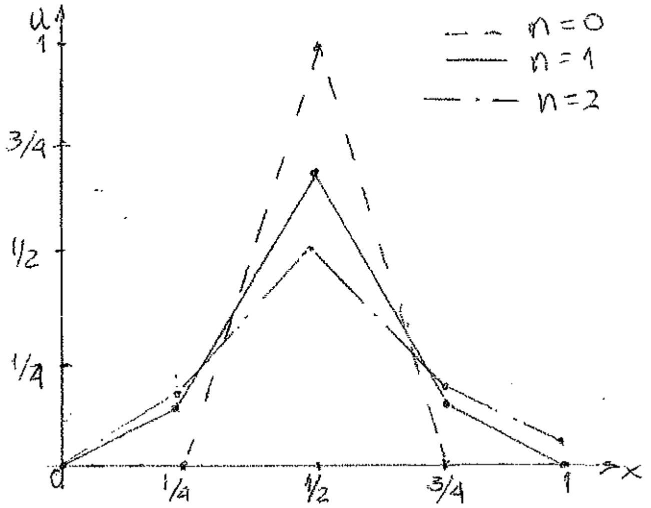
- $\nu = 0$

$$U_i^{n+1} = (\sigma \Delta t + 1)\, U_i^n$$

with

$U_0^n = 0$, $U_m^{n+1} = (\sigma \Delta t + 1)\, U_m^n$ ✓

c) $\nu = 0.1$, $\sigma = -0.1$, $\Delta x = 0.25$, $\Delta t = 0.1$



$x_i \backslash n$	0	1	2
0	0	0	0
1/4	0	0,16	0,2144
1/2	1	0,67	0,5001
3/4	0	0,16	0,2144
1	0	0	0,0512

✓

- The diffusion part of the equation distribute u over all the domain for a increasing time.
- The reaction part of the equation reduces u for every point with $u > 0$ in every time step.
- The boundary condition $u_x = 0$ makes the property (u) to acumulate in that boundary. (The flux of the property in the boundary is zero). ?
- The stability condition for the diffusion part $\left(\frac{\nu \Delta t}{\Delta x^2}\right)$ define a stable ~~and~~ solution without oscillations.

$$\frac{\nu \Delta t}{\Delta x^2} = 0.16 < \frac{1}{4}$$

d) BTCS

$$\frac{u_i^{n+1}-u_i^n}{\Delta t} = \nu \frac{u_{i+1}^{n+1}-2u_i^{n+1}+u_{i-1}^{n+1}}{\Delta x^2} + \sigma u_i^{n+1} + O(\Delta t, \Delta x^2)$$

$$U_i^{n+1} - \frac{\nu\Delta t}{\Delta x^2}\left(U_{i+1}^{n+1} - 2U_i^{n+1} + U_{i-1}^{n+1}\right) - \sigma U_i^{n+1}\Delta t = U_i^n$$

with $\frac{\nu\Delta t}{\Delta x^2} = r$

$$U_{i+1}^{n+1}(-r) + U_i^{n+1}(1+2r-\sigma\Delta t) + U_{i-1}^{n+1}(-r) = U_i^n$$

- for $u(0,t)=0$, $i=0$

$U_0^n = 0$, $\forall n$

- for $u_x(1,t)=0$, $i=m$ $\qquad \frac{U_{m+1}^{n+1}-U_{m-1}^{n+1}}{2\Delta x} = 0$

$U_{m+1}^{n+1} = U_{m-1}^{n+1}$

$$U_m^{n+1}(1+2r-\sigma\Delta t) + U_{m-1}^{n+1}(-2r) = U_m^n$$

$\Rightarrow$ $\underline{\underline{A}}\,\underline{u} = \underline{b}$

$\underline{b} = 0$

$$\underline{\underline{A}} = \begin{bmatrix} 1+2r-\Delta t\sigma & -r & & & \\ -r & 1+2r-\Delta t\sigma & -r & & \\ & \ddots & \ddots & \ddots & \\ & & -r & 1+2r-\Delta t\sigma & -r \\ & & & -2r & 1+2r-\Delta t\sigma \end{bmatrix}$$

✓

For a tri-diagonal matrix the most suitable algorithm is a Thomas. (Tridiagonal matrix) (symmetric. diagonal banded)

Another suitable algorithm is the Cholesky for banded matrices.

For both algorithms the matrix fulfill the minimum requirements.

✓

STUDENT 2

(EXCEL·LENT)

17/12/13

Numerical methods for PDEs
EXERCISE 3

For the numerical modeling of a new technique of contamination control, it is interesting to solve the diffusion-reaction PDE

$$u_t = \nu u_{xx} + \sigma u \qquad \text{in } x \in (0,1),\ t > 0 \tag{1}$$

with boundary conditions

$$u(0,t) = 0 \text{ and } u_x(1,t) = 0 \tag{2}$$

and the initial condition

$$u(x,0) = \begin{cases} 0 & \text{for} \quad x < 1/4 \\ 4x - 1 & \text{for} \quad 1/4 \le x1/2 \\ -4x + 3 & \text{for} \quad 1/2 \le x < 3/4 \\ 0 & \text{for} \quad 3/4 \le x \end{cases} \tag{3}$$

In the PDE (1), ν >0 is the diffusion coefficient and $\sigma < 0$ is the reaction coefficient. Both coefficients can be considered constant.

a) Propose an explicit finite difference scheme for the solution of the PDE (1) with boundary conditions (2) and initial conditions (3). Detail the numerical treatment of boundary conditions.

b) Which scheme is obtained for $\sigma = 0$ (diffusion equation)? And for $\nu = 0$ (reaction equation)?

c) Take $\nu = 0.1$, $\sigma = -0.1$, $\Delta x = 0.25$ and $\Delta t = 0.1$, and compute two time steps with the explicit scheme proposed in section *a*. Are the obtained results reasonable? Discuss with the help of the graphic of the profile of u.

d) Propose an implicit finite difference scheme to solve the PDE (1) with boundary conditions (2) and initial conditions (3). Detail how are boundary conditions treated, the structure of the matrix and the most suitable method to solve the linear system of equations.

a)

Setting an explicit finite difference scheme of the type FTCS, we can approximate the derivatives as

$$\left.\frac{\partial u}{\partial t}\right|_i^n = \frac{u_i^{n+1} - u_i^n}{\Delta t} + \mathcal{O}(\Delta t) \tag{4}$$

$$\left.\frac{\partial^2 u}{\partial x^2}\right|_i^n = \frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{\Delta x^2} + \mathcal{O}(\Delta x^2) \tag{5}$$

Therefore imposing the equation on node i, at time n and neglecting the truncation errors, yields to following numerical scheme:

$$\frac{U_i^{n+1} - U_i^n}{\Delta t} = \nu \frac{U_{i+1}^n - 2U_i^n + U_{i-1}^n}{\Delta x^2} + \sigma U_i^n \quad i = 1, ..., M+1, \tag{6}$$

✓

rearranging

$$U_i^{n+1} = \nu \frac{\Delta t}{\Delta x^2}[U_{i+1}^n - 2U_i^n + U_{i-1}^n] + \Delta t \sigma U_i^n + U_i^n \quad i = 1, ..., M+1 \tag{7}$$

and finally defining $r = \nu \frac{\Delta t}{\Delta x^2}$:

$$U_i^{n+1} = rU_{i+1}^n + (1 + \Delta t\sigma - 2r)U_i^n + rU_{i-1}^n \quad i = 1, ..., M+1. \tag{8}$$

✓

Notice that for $x = 0$ ($i = 0$), we have Dirichlet boundary conditions and there is "no problem", U_0 is known in any time-step. However on the other boundary we have Neumann boundary conditions, this means that the equation has to be imposed on $x = 1$ ($i = M+1$), where we need a fictitious node $i = M+2$ in order to compute the approximations. The value on this node can be obtained from the boundary condition. Using the following approximation of the derivative in the boundary

$$\left.\frac{\partial u}{\partial x}\right|_{M+1}^n \cong \frac{U_{M+2}^n - U_M^n}{2\Delta x} = 0 \tag{9}$$

therefore from the boundary condition we obtain that the value in the fictitious node is:

$$\frac{U_{M+2}^n - U_M^n}{2\Delta x} = 0 \rightarrow U_{M+2}^n = U_M^n. \tag{10}$$

Thus, the scheme is

$$\begin{cases} U_i^{n+1} = rU_{i+1}^n + (1 + \Delta t\sigma - 2r)U_i^n + rU_{i-1}^n & i = 1, ..., M. \\ U_{M+1}^{n+1} = (1 + \Delta t\sigma - 2r)U_{M+1}^n + 2rU_M^n & \\ U_0^n = 0 & \\ U_i^0 = u(i\Delta x, 0) & i = 0, ..., M+1 \end{cases} \tag{11}$$

✓

b)

For $\sigma = 0$:

$$\begin{cases} U_i^{n+1} = rU_{i+1}^n + (1 - 2r)U_i^n + rU_{i-1}^n & i = 1, ..., M. \\ U_{M+1}^{n+1} = (1 - 2r)U_{M+1}^n + 2rU_M^n & \\ U_0^n = 0 & \\ U_i^0 = u(i\Delta x, 0) & i = 0, ..., M+1 \end{cases} \tag{12}$$

which is the FTCS method obtained for a parabolic equation,

✓

For $\nu = 0$ means $r = 0$:

$$\begin{cases} U_i^{n+1} = (1 + \Delta t\sigma)U_i^n & i = 1, ..., M+1. \\ U_0^n = 0 & \\ U_i^0 = u(i\Delta x, 0) & i = 0, ..., M+1 \end{cases} \tag{13}$$

which is Euler method for solving first order ODEs for each node i.

c)

Substituting the given values of ν, Δx and Δt the explicit scheme is

$$\begin{cases} U_i^{n+1} = 0.16U_{i+1}^n + 0.67U_i^n + 0.16U_{i-1}^n & i = 1, 2, 3, \\ U_4^{n+1} = 0.67U_4^n + 2 \cdot 0.16U_3^n & \\ U_0^n = 0 & \\ U_i^0 = u(i\Delta x, 0) & i = 0, ..., 4 \end{cases} \tag{14}$$

Therefore the approximation for $t = 0.1$ ($n = 0$) is:

$$U_1^1 = 0.16U_2^0 + 0.67U_1^0 + 0.16U_0^0 = 0.16 + 0 + 0 = 0.16, \tag{15}$$
$$U_2^1 = 0.16U_3^0 + 0.67U_2^0 + 0.16U_1^0 = 0 + 0.67 + 0 = 0.67, \tag{16}$$
$$U_3^1 = 0.16U_4^0 + 0.67U_3^0 + 0.16U_2^0 = 0 + 0 + 0.16 = 0.16, \tag{17}$$
$$U_4^1 = 0.67U_4^0 + 2 \cdot 0.16U_3^0 = 0 + 0 = 0. \tag{18}$$

And for $t = 0.2$ ($n = 1$) is:

$$U_1^2 = 0.16U_2^1 + 0.67U_1^1 + 0.16U_0^1 = 0.1072 + 0.1072 + 0 = 0.2144, \tag{19}$$
$$U_2^2 = 0.16U_3^1 + 0.67U_2^1 + 0.16U_1^1 = 0.0256 + 0.4489 + 0.0256 = 0.5001, \tag{20}$$
$$U_3^2 = 0.16U_4^1 + 0.67U_3^1 + 0.16U_2^1 = 0 + 0.1072 + 0.1072 = 0.2144, \tag{21}$$
$$U_4^2 = 0.67U_4^1 + 2 \cdot 0.16U_3^1 = 0 + 2 \cdot 0.0256 = 0.0512. \tag{22}$$

Graphically the result is

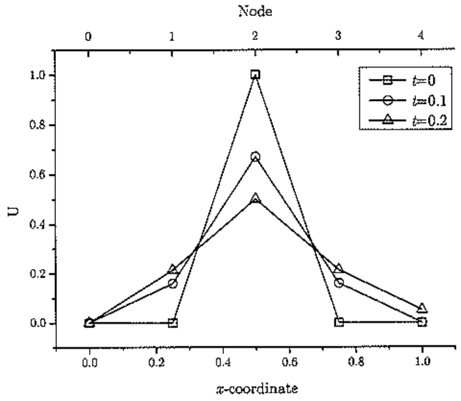


Figure 1: Explicit finite difference approximation.

The numerical results are reasonable, the diffusion effect is appreciated because the pick value is lower as time advances. Meanwhile, the surrounding nodes of 2 (where there is located the pick-value) increase. The reaction term effect can be appreciated because all values are smaller than what they would be without reaction, this effect is clearly appreciated in the central node. At $i = 0$ the boundary condition is always satisfied, U_0^n is always null. At $i = 4$ the boundary condition is also satisfied, the conditions says that the flux is null, therefore an increasing of U_4 in time is expected because of diffusion effect.

d)

Setting an implicit finite difference scheme of the type BTCS, we can approximate the derivatives as

$$\left.\frac{\partial u}{\partial t}\right|_i^{n+1} = \frac{u_i^{n+1} - u_i^n}{\Delta t} + \mathcal{O}(\Delta t) \tag{23}$$
$$\left.\frac{\partial^2 u}{\partial x^2}\right|_i^{n+1} = \frac{u_{i+1}^{n+1} - 2u_i^{n+1} + u_{i-1}^{n+1}}{\Delta x^2} + \mathcal{O}(\Delta x^2) \tag{24}$$

Therefore imposing the equation on node i, at time $n+1$ and neglecting the truncation errors, yields to following numerical scheme:

$$\frac{U_i^{n+1} - U_i^n}{\Delta t} = \nu \frac{U_{i+1}^{n+1} - 2U_i^{n+1} + U_{i-1}^{n+1}}{\Delta x^2} + \sigma U_i^{n+1} \quad i = 1, ..., M+1, \tag{25}$$

rearranging

$$U_i^{n+1} - \nu \frac{\Delta t}{\Delta x^2}[U_{i+1}^{n+1} - 2U_i^{n+1} + U_{i-1}^{n+1}] - \Delta t \sigma U_i^{n+1} = U_i^n \quad i = 1, ..., M+1 \tag{26}$$

and finally recalling the definition $r = \nu \frac{\Delta t}{\Delta x^2}$:

$$-rU_{i+1}^{n+1} + (1 - \Delta t\sigma + 2r)U_i^{n+1} - rU_{i-1}^{n+1} = U_i^n \quad i = 1, ..., M+1. \tag{27}$$

In order to deal with Neumann boundary conditions, the same strategy as in the explicit method can be used:

$$\frac{U_{M+2}^{n+1} - U_M^{n+1}}{2\Delta x} = 0 \rightarrow U_{M+2}^{n+1} = U_M^{n+1}. \tag{28}$$

Thus, the scheme is

$$\begin{cases} -rU_{i+1}^{n+1} + (1 - \Delta t\sigma + 2r)U_i^{n+1} - rU_{i-1}^{n+1} = U_i^n & i = 1, ..., M. \\ (1 - \Delta t\sigma + 2r)U_{M+1}^{n+1} - 2rU_M^{n+1} = U_{M+1}^n & \\ U_0^n = 0 & \\ U_i^0 = u(i\Delta x, 0) & i = 0, ..., M+1 \end{cases} \tag{29}$$

The system of equations that need to be solved is

$$\mathbf{A}\mathbf{U}^{n+1} = \mathbf{U}^n + \mathbf{F} \tag{30}$$

where, taking $\alpha = 1 - \Delta t\sigma + 2r$,

$$\mathbf{A} = \begin{bmatrix} \alpha & -r & & & \\ -r & \alpha & -r & & \\ & \ddots & \ddots & \ddots & \\ & & -r & \alpha & -r \\ & & & -2r & \alpha \end{bmatrix} \quad \mathbf{U}^n = \begin{pmatrix} U_1^n \\ \vdots \\ U_{M+1}^n \end{pmatrix} \quad \mathbf{F} = \begin{pmatrix} ru(0,(n+1)\Delta t) \\ 0 \\ \vdots \\ 0 \\ 2\Delta x r u_x(1,(n+1)\Delta t) \end{pmatrix} \tag{31}$$

Notice that for this particular case, since both boundary conditions are null $\mathbf{F} = \mathbf{0}$. Thus, the system of equations takes the form:

$$\mathbf{A}\mathbf{U}^{n+1} = \mathbf{U}^n \tag{32}$$

In order to solve this system of equations, is better to use a direct method, because the dimensions of the system is quite small. Within the direct methods Cholesky cannot be used because $\mathbf{A}$ matrix is not symmetric positive definite, therefore a suitable method would be LU decomposition, no matter if it's used Doolittle or Crout. Even more, since $\mathbf{A}$ is a band matrix we don't need to worry about fill in issues.

Numerical methods for PDEs — Homework 1

ODEs — **Due: November 26, 2013**

Compute the numerical solution of the following the initial value problem (IVP)

$$y'' = y - x \quad \text{in } (0,1)$$

$$y(0) = 1,\ y'(0) = 2$$

The exact solution of this problem is $y = \exp(x) + x$.

1. Implement a routine for the solution of IVP using Euler, Heun and 4^{th} order Runge-Kutta method. Plot the solution obtained with 8 time steps of RK4 and compare it with the Euler and Heun results for an equivalent computational cost (i.e. same number of function evaluations). Draw some conclusions.
2. Check the convergence of the methods: plot the logarithm of the error at the end point $x = 1$ vs the logarithm of the number of function evaluations. Do the results agree with the theoretical convergence rates? Comment the results.
3. Solve the IVP with the ode45 Matlab function. Which method corresponds to this function? What can you ensure about the accuracy of the obtained solution? How could you improve the accuracy?

STUDENT 1

(APROVAT)

8.5

PDE

HOMEWORK 1

Assumption $y'' = y - x$ in $(0,1)$

Initial conditions $y(0) = 1$

$y'(0) = 2$

Exact solution

$y = \exp(x) + x$

1.

Scripts Euler.m Heun.m RungeKutta.m

First order system $z^{(1)} = y$ $z'^{(1)} = y' = z^{(2)}$

$z^{(2)} = y'$ $z'^{(2)} = y'' = y - x = z^{(1)} - x$

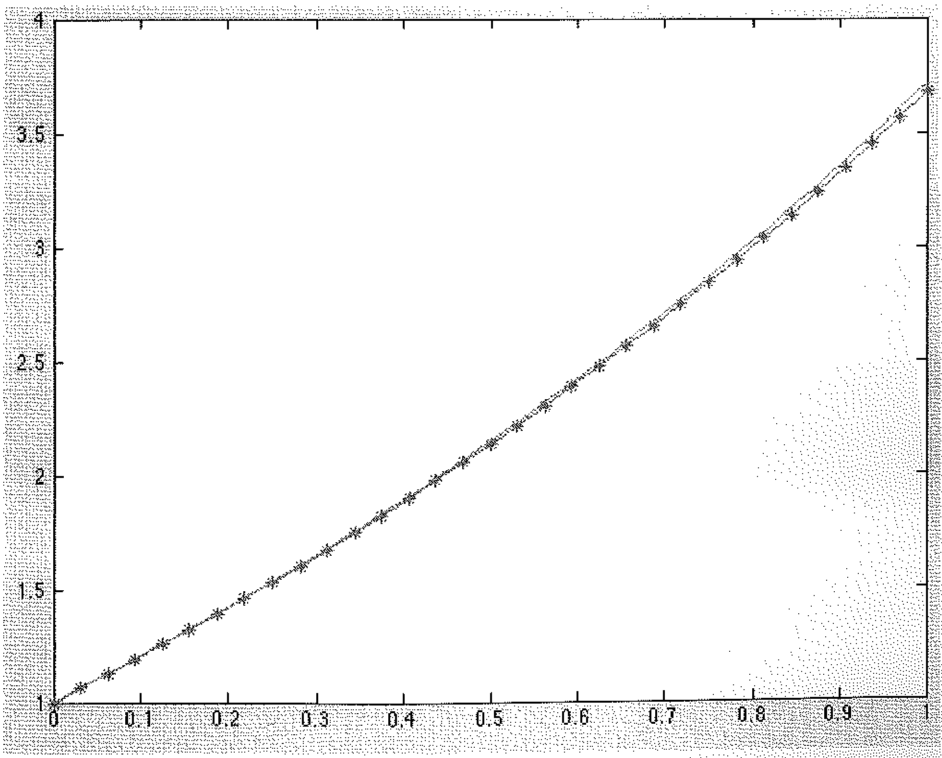
Vectors

$$\vec{z} = \begin{bmatrix} z^{(1)} \\ z^{(2)} \end{bmatrix} \qquad \frac{d}{dx}\vec{z} = \begin{bmatrix} z^{(2)} \\ z^{(1)} - x \end{bmatrix} \qquad \overrightarrow{z(0)} = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$$

The vectors denotes the first order system.

Euler Equation $Y^{i+1} = Y^i + h \cdot f\left(x^i, Y^i\right)$

Function evaluations $m = 32$



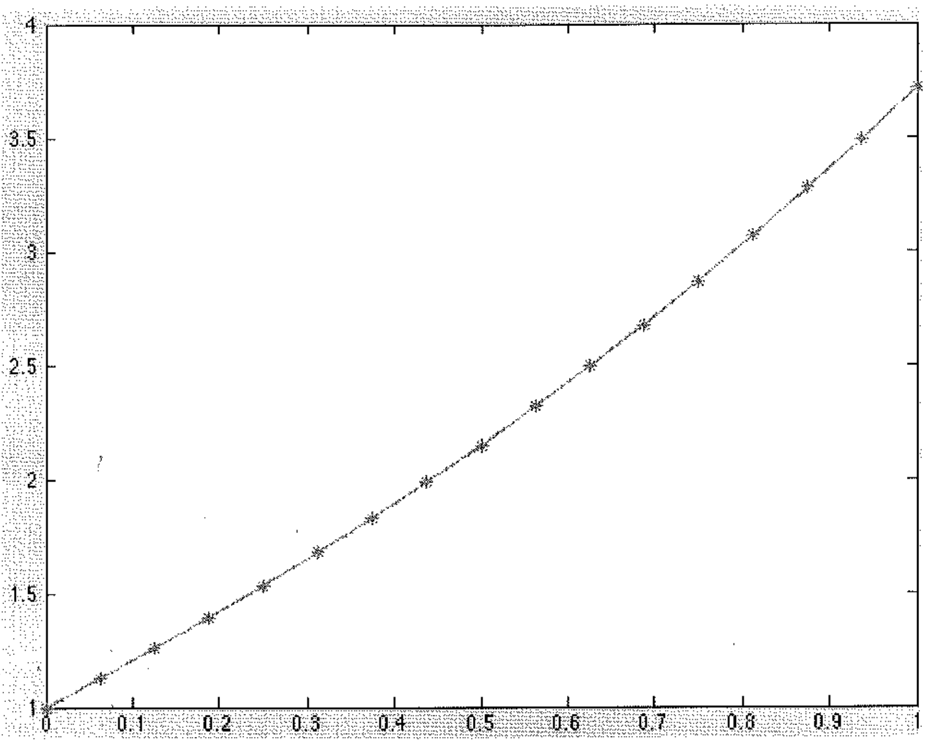
Heun

Equation

$$Y^{i+1*} = Y^i + h \cdot f\left(x^i, Y^i\right)$$

$$Y^{i+1} = Y^i + \frac{h}{2} \cdot \left(f\left(x^i, Y^i\right) + f\left(x^{i+1}, Y^{i+1*}\right)\right)$$

Function evaluations $m = 16$



Runge-Kutta

Equation

$$k_1 = f\left(x^i, Y^i\right)$$

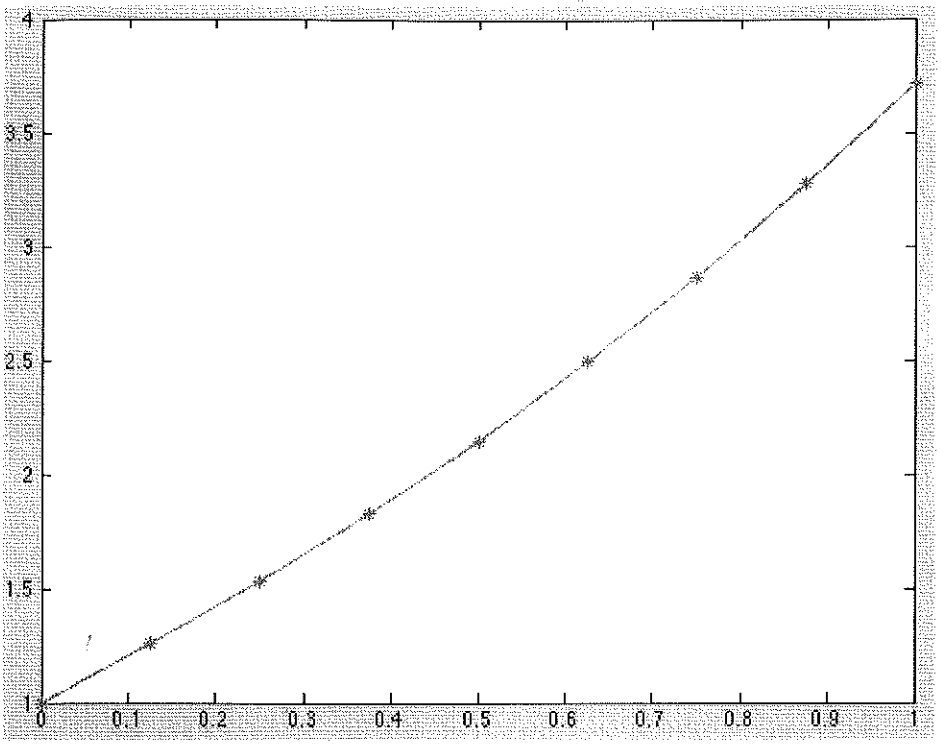
$$k_2 = f\left(x^i + \frac{h}{2}, Y^i + \frac{h}{2} \cdot k_1\right)$$

$$k_3 = f\left(x^i + \frac{h}{2}, Y^i + \frac{h}{2} \cdot k_2\right)$$

$$k_4 = f\left(x^i + h, Y^i + h \cdot k_3\right)$$

$$Y^{i+1} = Y^i + \frac{h}{6} \cdot \left(k_1 + 2 \cdot k_2 + 2 \cdot k_3 + k_4\right)$$

Function evaluations $m = 8$



Conclusions

The plots show the numerical solution (blue) and the exact solution (green). The Runge-Kutta method is here a 4^{th} order, the Heun method is 2^{nd} order and the Euler method is a 1^{st} order, so to make the same function evaluations and here by equivalent computational cost there are used different time steps for each method.

Even though we are using the same function evaluations, it is clearly to se that the Euler method is less exact than the to others by looking at the plots. It is a bit more difficult to se witch one of Heun ore Runge-Kutta that is the most exact method, but by theory the Runge-Kutta should give the most exact solution. So if you need an exact solution instead of a fast, Runge-kutta is the best method to obtain this.

2.

Scripts

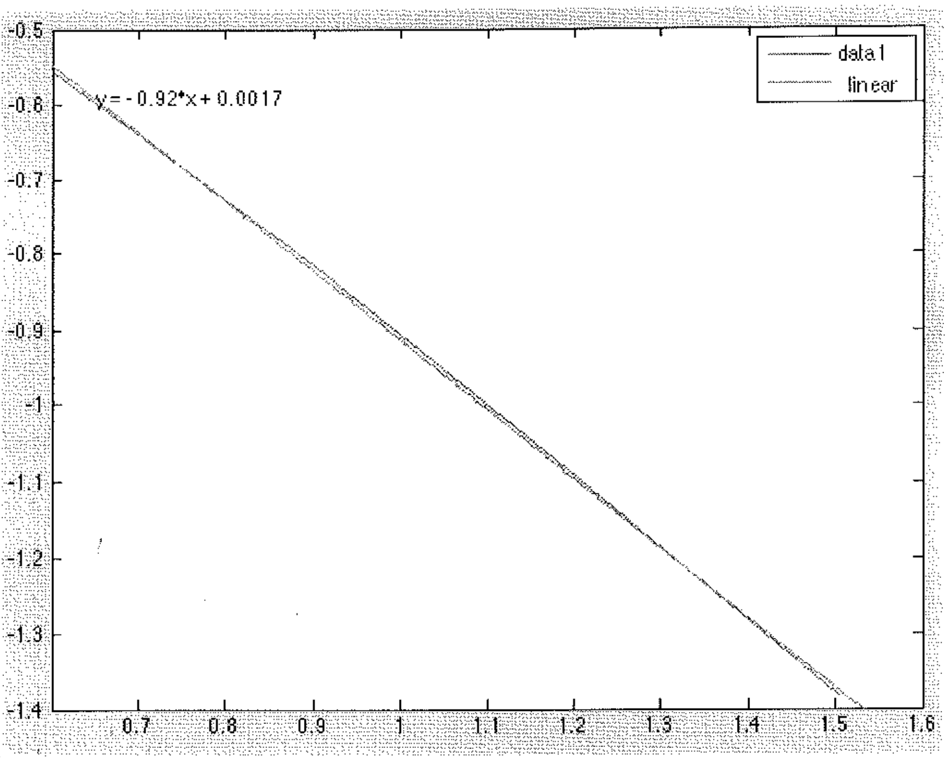
Euler.m Heun.m RungeKutta.m

Convergence

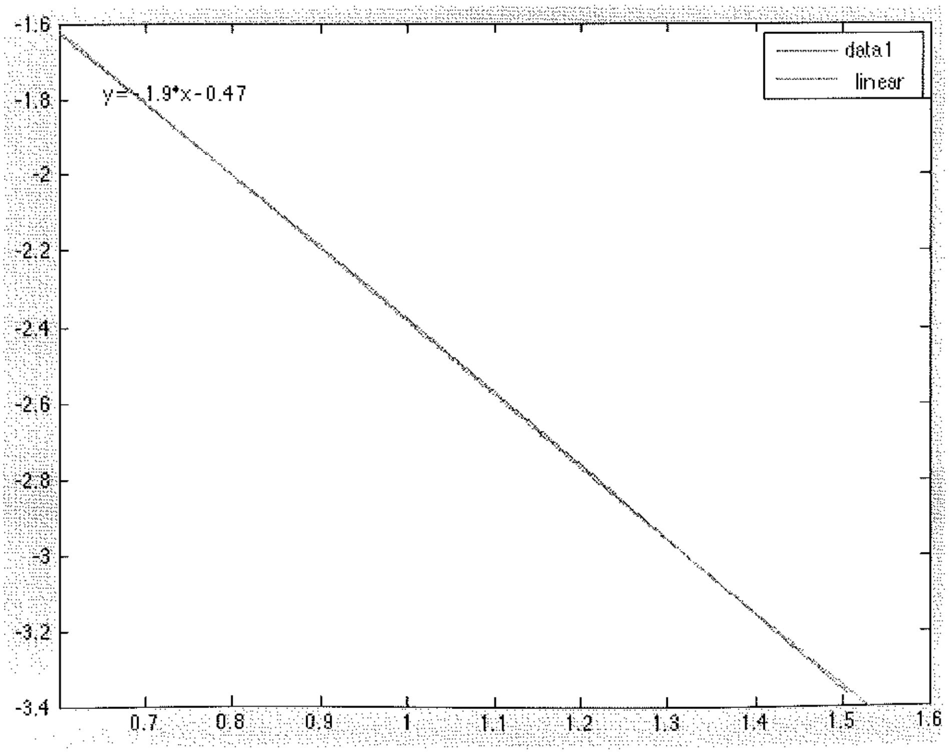
If a method is convergence the error should decrease when the size of the time steps is decreased.

The plots show the error as a function of time steps. The slope shows the order of convergence.

Euler	Expected order		Order Matlab
	1	≈	0.92

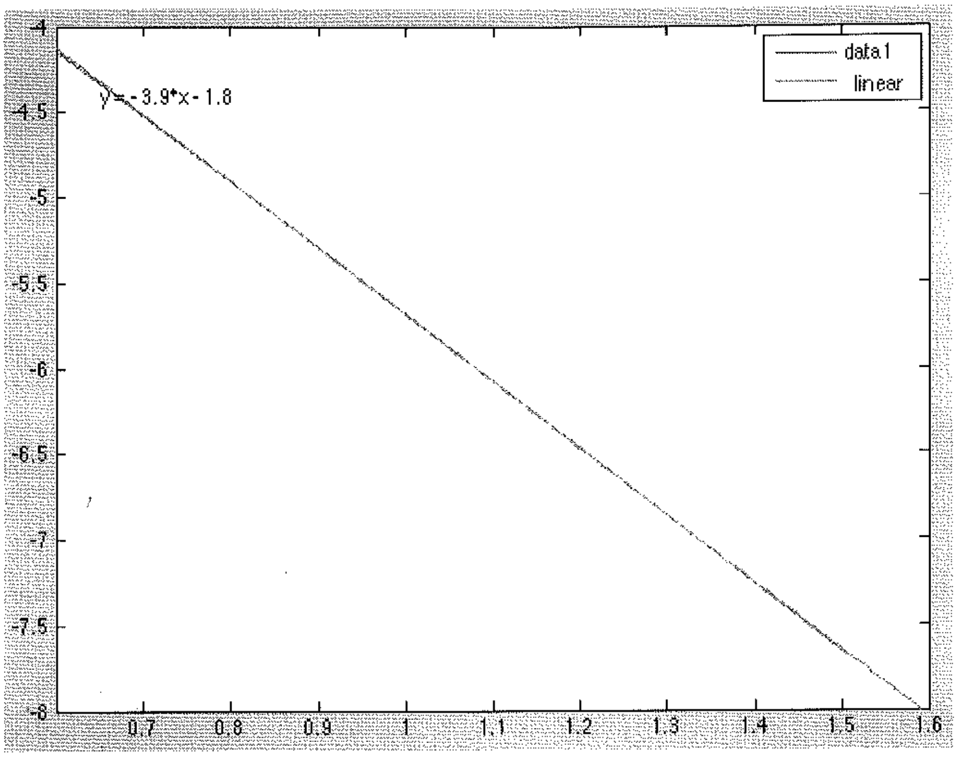


Heun	Expected order		Order Matlab
	2	≈	1.9



Runge-Kutta

Expected order		Order Matlab
4	≈	3.9



Conclusions

The results for the orders of convergence could have been even more exact by using a higher number of time steps, but you can see that the order from matlab is almost equal to the expected order.

By looking at the order of the convergence for each method, you can see that the error of the Runge-Kutta would decrease 4 times as fast as the error of the Euler method and the error of the Heun method would decrease 2 times as fast as the Euler method.

You should consider several time steps!

1,5

3.

Scripts

F.m

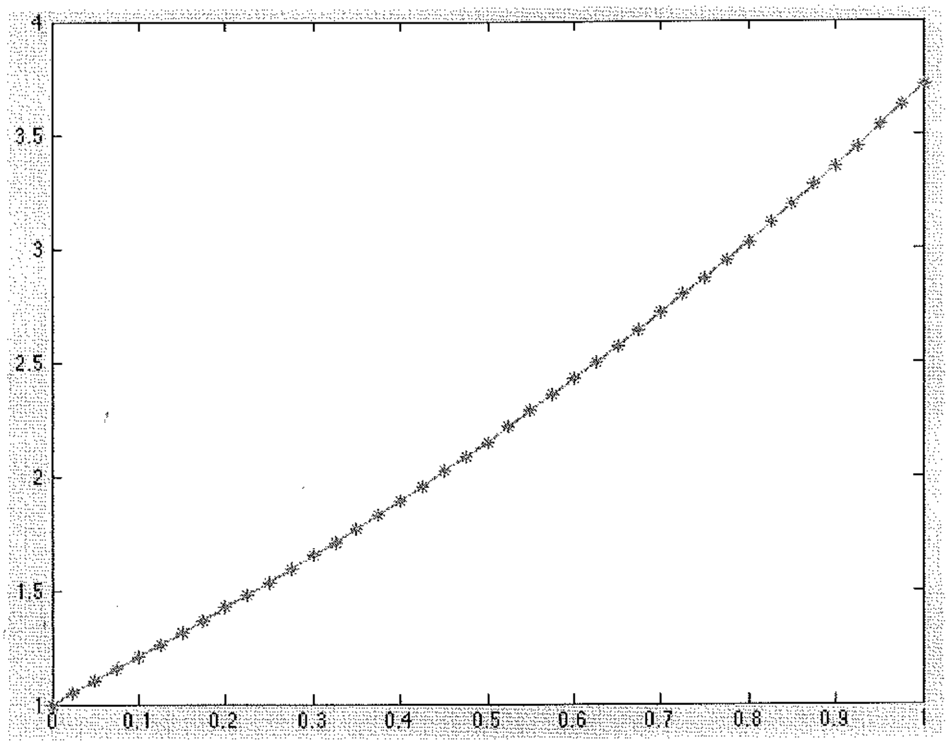
Ode45.m

Comparison_ODE45_Euler.m

ode45

Matlab is using the function RKF45, when using ode45.

3



Accuracy Matlab

Matlab is programmed so it controls the local error. The maximum value of the local error is equal to $1 \cdot 10^{-6}$.

The global error for ode45 (the error at the end point $x = 1$) is calculated in the script, and has the value $e_{global} = 6.388 \cdot 10^{-9}$ witch is really small, and thereby the ode45 gives good accuracy in terms of getting the best approximation. As a comparison the error at $x = 1$ for the Euler method using $m = 6 \cdot 41$, witch gives the same computational cost as ode45, is equal to $e_{global} = 0.0055$. So the accuracy method of ode45 and Euler would be ode45.

Why do you compare with the worse of the previous methods?!

Improve Accuracy

The approximation accuracy could by improved by using more time steps and by using a method of higher order and thereby doing more evaluations of f for each time step, witch would result in a truncation error of a higher order.

STUDENT 2

(APROVAT)

7.5

Homework 1

In the assignment we are giving a 2nd order ODE, but Euler, Heun and Runge-Kutta all assume a first order ODE. Therefore it is necessarry to reduce the 2nd order ODE into a system of first order ODEs.

Initial value problem $y'' = y - x \quad in\ (0,1)$

Boundary conditions $y(0) = 1, \quad y'(0) = 2$

System of first order ODEs, where we set introduze two new functions (z_1 and z_2)

$$y_1 = y \quad and \quad y_2 = y'$$

We now get the two new 1st order ODE's

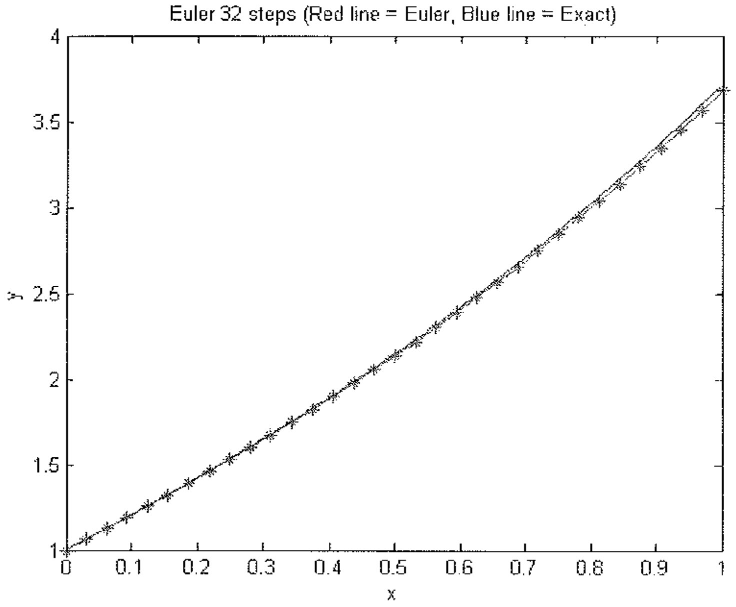
$$y_1' = y' = y_2$$

$$y_1' = y' = y_2 \quad and \quad y_2' = y'' = y - x = y_1 - x$$

We have now reduced the 2nd order ODE into a system of 1st order ODE. This means that we can use the schemes of Euler, Heun and Runge-Kutta.

Question 1

The following plots (figure 1-3) show the numerical solutions of respectively Euler, Heun and Runge-Kutta compared to the exact solution given in the problem.



4

Figure 1 - Euler method with time step 32.

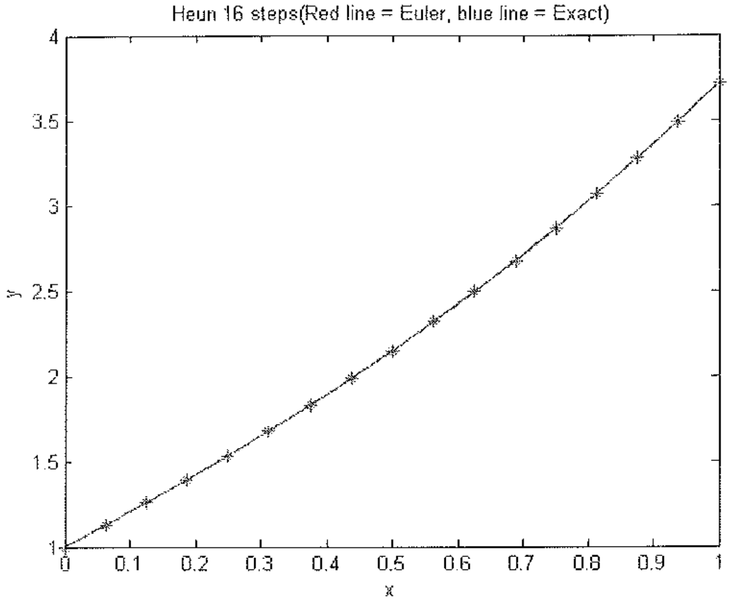


Figure 2 - Heun method with time step 16.

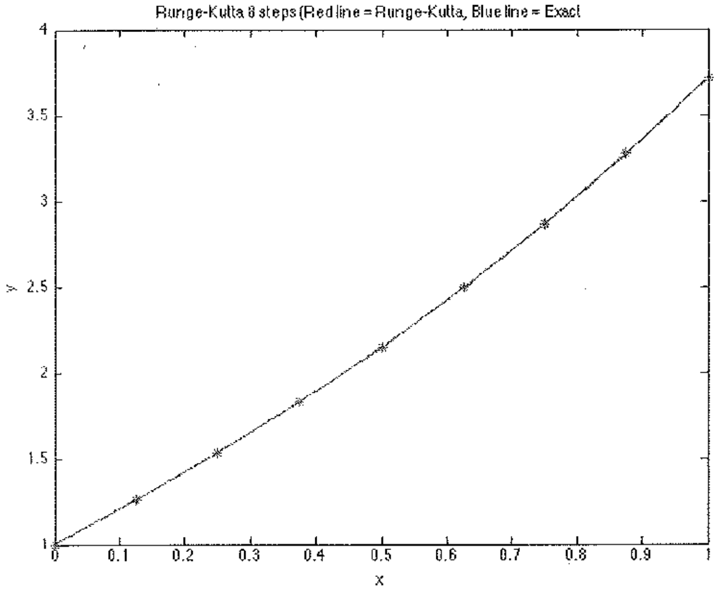


Figure 3 - Runge-Kutta with time step 8.

Because of the different orders of the schemes (Euler = 1st order, Heun = 2nd order and Runge-Kutta = 4th order), we use three different time steps to compare the three schemes.
When the three plots with the equivalent computational cost are compared the differences looks small, but at the plots it can be seen that Euler is the less precise one.
If we zoom in, the plots show that Runge-Kutta is more precise than Heun.
Therefore for the same number of steps Runge-Kutta is more precise than both Heun and Euler. But at the same time it is a more expensive scheme.

Question 2

1,5

The following three plots (figure 4-6) show the convergence for Euler, Heun and Runge-Kutta. Here the blue line defines the convergence of the scheme and the red line defines the linear fit.

You should consider several time steps

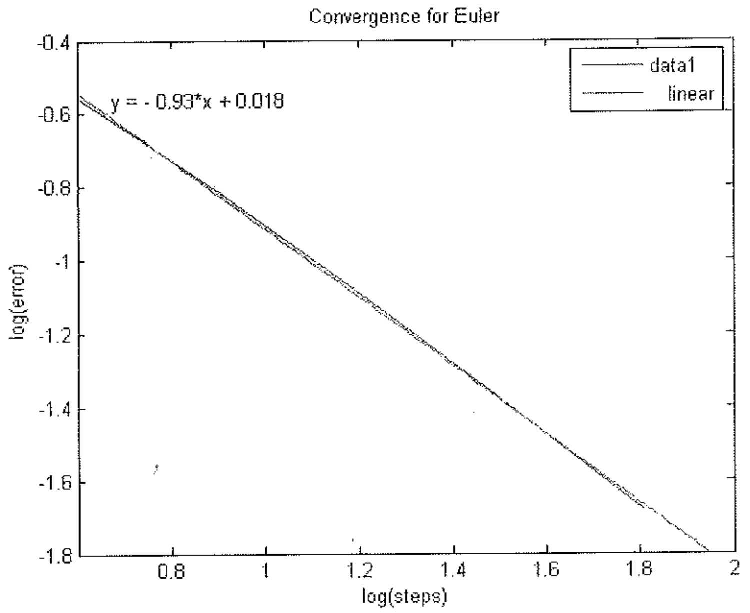


Figure 4 - Convergence plot for Euler.

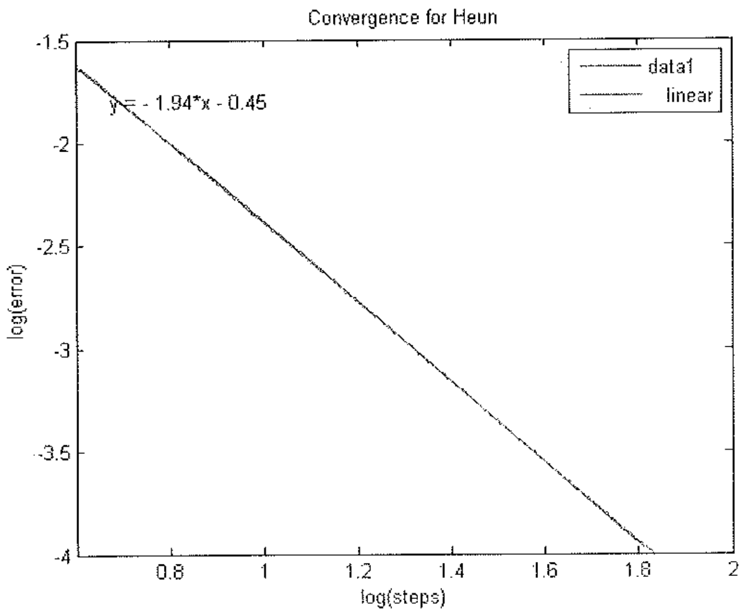


Figure 5 - Convergence plot for Heun.

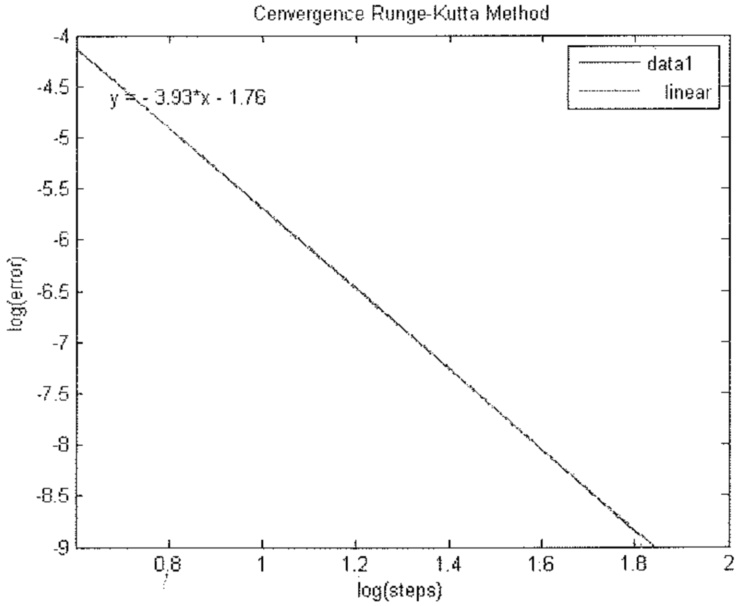


Figure 6 - Convergence plot for Runge-Kutta.

As shown both in the above figures (figure 4-6) and table 1, the three methods are very close to the theoretical convergence rate.
A scheme is said to be convergent if the error becomes smaller, when the time steps becomes smaller.

Method	Euler	Heun	Runge-Kutta
Line	-0,93*x+0,018	-1,94*x-0,45	-3,93*x-1,76
Expected slope	-1	-2	-4

Table 1 - Comparrison of schemes vs. convergence.

By looking at the plots and the linear fittings it can be seen that Heun approaches the exact solution approximately twice as fast as Euler. And furthermore Runge-Kutta is approaching aproximately twice as fast as Heun.

Question 3

2

Figure 7 shows the comparison of the ODE45 and the exact solution.
The ODE45 function in matlab uses 6 function evaluations using Runge-Kutta45.
The error calculated at x = 1 is -6.3380e-09, which is very small. That tells us that the method is very accurate.

The error at 40 time steps with respect to the Runge-Kutta45 scheme is calculated to 8.6662e-09.

So by using the same amount of time steps (40) the ODE45 function is better, but also the computational cost is bigger. ||

What do you mean?

The accuracy can be improved by taking smaller time steps or by using a scheme with a higher order.

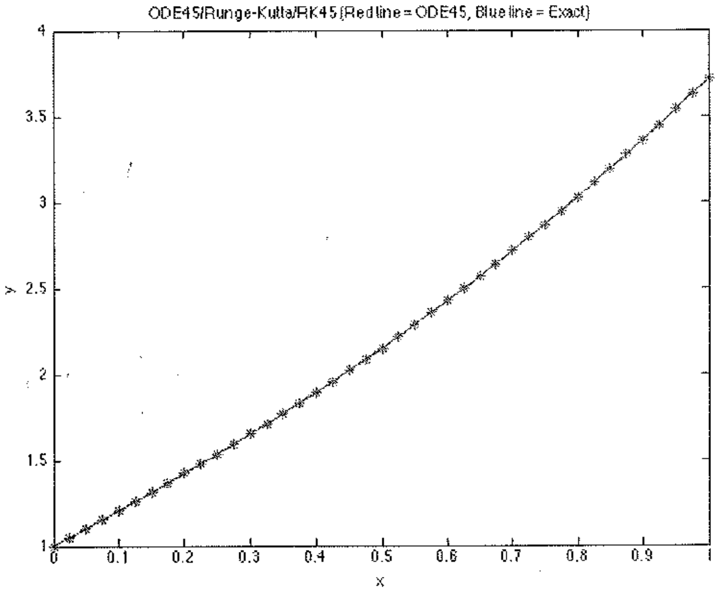


Figure 7 – ODE45 with 40 time steps.

STUDENT 1

(NOTABLE)

$\boxed{8}$

NUMERICAL METHODS for PDEs
Master of Science in Computational Mechanics
Fall Semester 2013
Homework 1: ODEs

Compute the numerical solution of the following the initial value problem (IVP)

$$y'' = y - xin(0,1)$$
$$y(0) = 1, y'(0) = 2$$

The exact solution of this problem is $y = exp(x) + x$

1. Implement a routine for the solution of IVP using Euler, Heun and 4^{th} order RungeKutta method. Plot the solution obtained with 8 time steps of RK4 and compare it with the Euler and Heun results for an equivalent computational cost (i.e. same number of function evaluations). Draw some conclusions.

1. Solution:

First of all, in order to solve the system we have to reduce the 2^{nd} order ODE to a system of first order ODEs, the following way:

$$\left\{\begin{matrix} y_1 = y \\ y_2 = \frac{dy}{dx} = \frac{dy_1}{dx} = y' \\ y_3 = \frac{d^2y}{dx^2} = \frac{dy_2}{dx} = y'' \end{matrix}\right\} \qquad y' = y_2 \qquad y'(1) = y(2)$$

Using this notation we can write the following:

$$\frac{d^2y}{dx^2} = \frac{dy_2}{dx} = f(x, y_1, y_2)$$

$$\bar{\mathbf{y}} = \begin{Bmatrix} y_1 \\ y_2 \end{Bmatrix}, \text{ also } \bar{\mathbf{f}} = \begin{Bmatrix} y_2 \\ f(x, y_1, y_2) \end{Bmatrix} = \begin{Bmatrix} \overset{y_2}{y'} \\ y - x \end{Bmatrix} \leftarrow$$

with initial values: $\uparrow\bar{\mathbf{y}}(0) = \begin{Bmatrix} 1 \\ 2 \end{Bmatrix}$ and $\boxed{\uparrow\bar{\mathbf{f}}(0) = \begin{Bmatrix} 2 \\ 1 \end{Bmatrix}}$ $\underline{\underline{\text{no}}}$

We obtain the 1^{st} order system:

$$\frac{d\bar{\mathbf{y}}}{dx} = \mathbf{f}(x, \bar{\mathbf{y}}) \qquad x \in (0, 1)$$
$$\bar{\mathbf{y}}(0) = \begin{Bmatrix} 1 \\ 2 \end{Bmatrix}$$

This function can be defined in *matlab* the following way:

```
% ODE Vector function
% Parameter list:
%    x - x coordinate at which the numerical approximation is computed;
%    Y - 2 row column vector containing the values y_1 & y_2 of ...
   vector Y_i;
%
% Result: 2 row column vector containing the values of the vector ...
   function
% f computed at x_i

function result = ODE_VECT_F (x,Y)
    result = zeros(2,1);
    result(1) = Y(2);
    result(2) = Y(1) - x;
end
```

(a) The **Forward Euler Method** can be implemented in matlab using the following function:

```
% Forward Euler Method for vector ODE system
% Parameter list:
%    x - x_{i+1} coordinate at which the numerical approximation is
%    computed;
%    h - interval step;
%    Y - 2 row column vector containing the values y_1 & y_2 of ...
   vector Y_i;
%
% Result: 2 row column vector containing Y_{i+1} element ...
   computed with
% Forward Euler Method.

function result = EulerM (x,h,Y)
    result = zeros(2,1);
    result(:,1) = Y + h*ODE_VECT_F(x,Y);
end
```

We can compute the numerical apprixmation for $n = 32$ steps:

```
% Euler Method
n = 32;
h = 1/n;
x = 0:h:1;

Y = zeros(2,length(x));
Y(:,1) = [1;2]; %initial conditions

for i = 2 : n+1
    Y(:,i) = EulerM(x(i),h,Y(:,i-1));
end
```

(b) The **Heun Method** can be implemented in matlab using the following function:

```
% HEUN Method for vector ODE system
% Parameter list:
%    x - x_i coordinate at which the numerical approximation is
%    computed;
%    h - interval step;
%    Y - 2 row column vector containing the values y_1 & y_2 of ...
    vector Y_i;
%
% Result: 2 row column vector containing Y_{i+1} element ...
    computed with HEUN Method.

function result = HeunM(x,h,Y)
    K1 = zeros(1,2);
    K2 = zeros(1,2);
    K1 = ODE_VECT_F(x,Y);
    K2 = ODE_VECT_F(x+h,Y+h*K1);
    result = Y + h/2*(K1+K2);
end
```

We can compute the numerical apprixmation for $n = 16$ with a similar **for** loop.

(c) The 4^{th} **order Runge-Kutta** can be implemented in matlab using the following function:

```
% Runge-Kutta Method
% 4th order Runge-Kutta Method for vector ODE system
% Parameter list:
%    x - x_i coordinate at which the numerical approximation is
%    computed;
%    h - interval step;
%    Y - 2 row column vector containing the values y_1 & y_2 of ...
    vector Y_i;
%
% Result: 2 row column vector containing Y_{i+1} element ...
    computed with RK4 Method.

function result = RK4M(x,h,Y)
    K1 = zeros(1,2);
    K2 = zeros(1,2);
    K3 = zeros(1,2);
    K4 = zeros(1,2);

    K1 = ODE_VECT_F(x,Y);
    K2 = ODE_VECT_F(x+h/2,Y+h/2*K1);
    K3 = ODE_VECT_F(x+h/2,Y+h/2*K2);
    K4 = ODE_VECT_F(x+h,Y+h*K3);
    result = Y + h/6*(K1+2*K2+2*K3+K4);
end
```

We can compute the numerical apprixmation for $n = 8$ with a similar **for** loop.

By plotting the numerical approximations, and the exact solution on the same graph we can see a direct visual comparison of their precision:

4

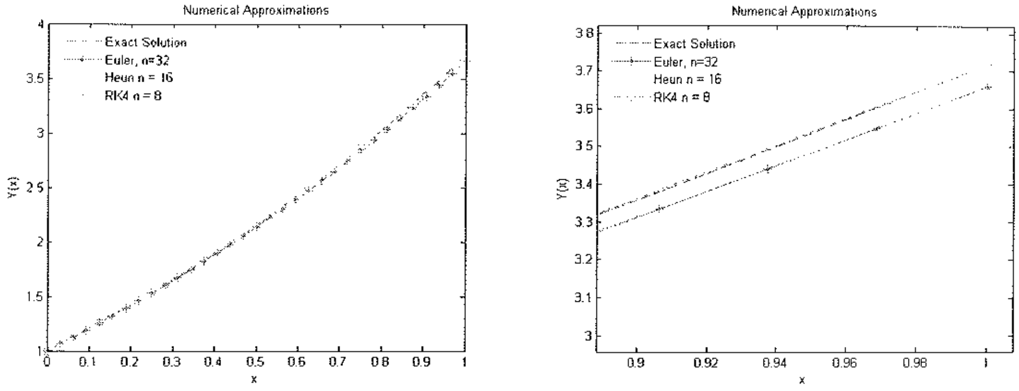


Figure 1: Method comparison graph; zoomed end section.

We can clearly observe that even if the Euler Method has been computed in $n = 32$ steps it is the least performant method in terms of precision. At this detail we can say that the Heun and RK4 methods deliver far better precision with approximately the same computation costs.

2. Check the convergence of the methods: plot the logarithm of the error at the end point $x = 1$ vs the logarithm of the number of function evaluations. Do the results agree with the theoretical convergence rates? Comment the results.

1.5

9

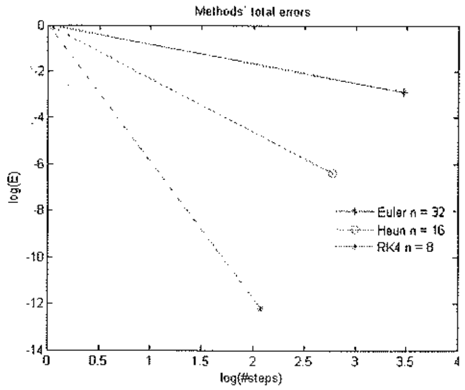


Figure 2: Error comparison graph

Observing the above graph we can clearly state that at approximately the same computational cost the the **RK4 method** is the most accurate.

Approximations at $x = 1$			Exact solution
Euler	Heun	RK4	
3.6608	3.7166	3.71823	3.7183
Total errors			-
Euler	Heun	RK4	-
0.0575	0.0017	$4.984 \cdot 10^{-6}$	-
Slopes of the plots			-
Euler	Heun	RK4	-
0.8239	2.3025	5.8714	-

3. (a) *Solve the IVP with the ode45 Matlab function.*

2.5

The built-in **ode45** function can be implemented using the following code:

```
% Computing with ODE45
% Parameters:
%   - T row vector containing the x or t coordinates at which ...
    the numerical approximations are computed;
%   - Y4 2 row matrix containing the approximated values of y_1 ...
    and y_2;
%   - [0,1] interval of function;
%   - [1,2] initial conditions;

[T,Y4] = ode45('ODE_VECT_F',[0,1],[1,2]);
```

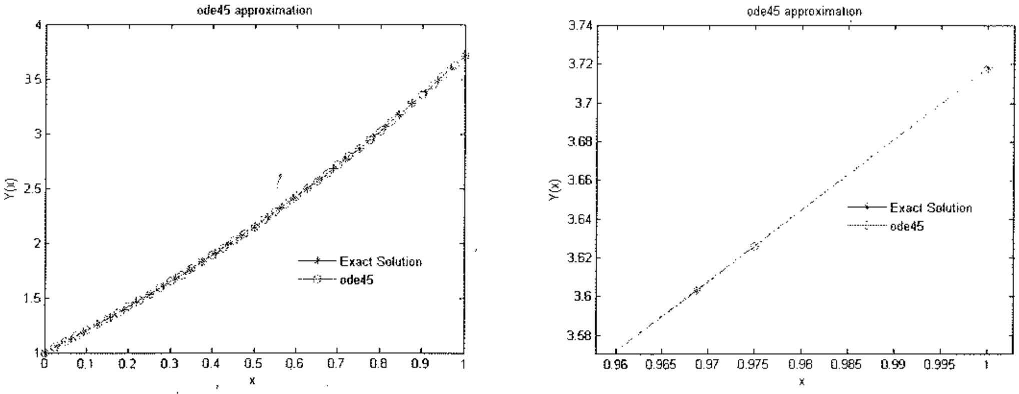


Figure 3: Numerical approximation with ode45 function; zoomed end section

We can clearly observe that the obtained numerical approximation is almost identical with the exact solution. Even at a high scale zoom no difference can be observed.

Although the total error is significantly lower than any previous result when we plot it related to the computational cost in terms of the step numbers (log(steps)) we can conclude that an ideal compromise would still be the fixed step RK4 method.

unclear how you reach this conclusion!

Total errors	
RK4	ode45
$4.984 \cdot 10^{-6}$	$6.338 \cdot 10^{-9}$

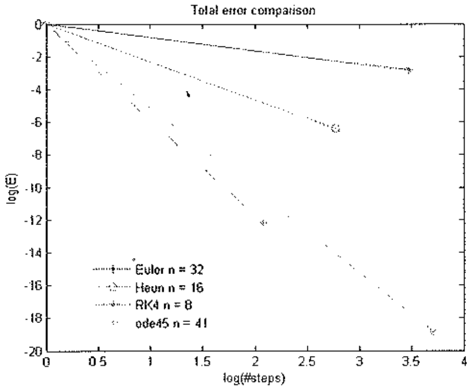


Figure 4: Total error comparison in terms of computational cost.

(b) *Which method corresponds to this function?*

The **ode45** function corresponds to the Runge-Kutta method, it uses a variable step size and also the order can change between 4^{th} or 5^{th}.

(c) *What can you ensure about the accuracy of the obtained solution?*

We can ensure two functional parameters of the accuracy of this function:

- the relative or absolute tolerance of the function which by default is set to: 10^{-6}; we can set this using:

```
% Set the relative tolerance: 1e-9 | 10^-9
option = odeset('RelTol',(1e-9));
% Call the ode45 function with the option parameter
[T,Y4] = ode45('ODE_VECT_F',[0,1],[1,2],option);
```

- the *maximum step size*; we can set this using this code:

```
% Set the maximum step size to: 1/8
option = odeset('MaxStep',(1/8));
% Call the ode45 function with the option parameter
[T,Y4] = ode45('ODE_VECT_F',[0,1],[1,2],option);
```

(d) *How could you improve the accuracy?*

Without significantly increasing the computational cost by adding further points to our approximation we can enhance the accuracy using the **refine** option. This parameter sets the number of iterations executed by the function to get from $Y_i(x_i)$ to $Y_{i+1}(x_{i+1})$.

9,5

Numerical methods for PDEs
ODEs
Homework 01.

STUDENT 2

Compute the numerical sol

(NOTABLE)

The exact solution of this problem is:

$$y = e^x + x$$

1. **Implement a routine for the solution of the IVP using Euler, Heun and 4th order Runge-Kutta (RK4) method. Plot the solution obtained with 8 time steps of RK4 and compare it with the Euler and Heun results for an equivalent computational cost.**

Before applying any of the abovementioned methods, the second order ODE of the problem must be converted into a system of first order ODEs. This requirement is due to they are intended for solving first order ODEs. This conversion is made with the aid of an auxiliary function u_i as follows:

$$u_1 = y$$

$$u_2 = y' = u_1'$$

$$u_3 = y'' = u_2' = y - x = u_1 - x$$

And the resulting system of first order ODEs to be solved is:

$$\begin{cases} u_1' = u_2 \\ u_2' = u_1 - x \\ u_1(0) = 1 \\ u_2(0) = 2 \end{cases}$$

✓ 4

Using 8 time steps in RK4 method implies to evaluate a function 32 times, since at each step the function is evaluated 4 times. Considering that at each step of Euler and Heun method a function is respectively evaluated once and twice, 32 time steps are going to be used for the Euler method and 16 times steps for the Heun method.

In order to be able to implement a routine for each method on Matlab, the main calculations at each one of them are presented.

Forward Euler method (m = 32)

$$u_{1,i+1} = u_{1,i} + h \cdot f(x_i, u_{1,i}) = u_{1,i} + h \cdot u_{2,i}$$

$$u_{2,i+1} = u_{2,i} + h \cdot f(x_i, u_{2,i}) = u_{2,i} + h \cdot (u_{1,i} - x_i)$$

$$i = 0,1 \dots m$$

$$h = 1/m$$

Being the result with this method for 32 times steps: $Y(1) = 3{,}6770$

Heun method (m = 16)

$$u^*_{1,i+1} = u_{1,i} + h \cdot f(x_i, u_i) = u_{1,i} + h \cdot u_{2,i}$$

$$u^*_{2,i+1} = u_{2,i} + h \cdot f(x_i, u_i) = u_{2,i} + h \cdot [u_{1,i} - x_i]$$

$$u_{1,i+1} = u_{1,i} + \frac{h}{2} \cdot \left[f(x_i, u_i) + f\left(x_{i+1}, u^*_{1,i+1}\right)\right] = u_{1,i} + \frac{h}{2} \cdot \left[u_{2,i} + u^*_{2,i+1}\right]$$

$$u_{2,i+1} = u_{2,i} + \frac{h}{2} \cdot \left[f(x_i, u_i) + f\left(x_{i+1}, u^*_{2,i+1}\right)\right] = u_{2,i} + \frac{h}{2} \cdot \left[u_{1,i} - x_i + u^*_{1,i+1} - x_{i+1}\right]$$

Substituting $u^*_{1,i+1}$ and $u^*_{2,i+1}$ in the last two equations, the new equations to be evaluated at each time step are:

$$u_{1,i+1} = u_{1,i} + \frac{h}{2} \cdot \left[u_{2,i} + u_{2,i} + h \cdot \left(u_{1,i} - x_i\right)\right]$$

$$u_{2,i+1} = u_{2,i} + \frac{h}{2} \cdot \left[u_{1,i} - x_i + u_{1,i} + h \cdot u_{2,i} - x_{i+1}\right]$$

$$i = 0,1 \ldots m$$

$$h = 1/m$$

Being the result with this method for 16 time steps: $Y(1) = 3{,}7166$

RK4 (m = 8)

$$K_{11} = u_{2,i}$$

$$K_{12} = u_{1,i} - x_i$$

$$K_{21} = u_{2,i} + \frac{h}{2} \cdot K_{12}$$

$$K_{22} = u_{1,i} + \frac{h}{2} \cdot K_{11} - x_i - \frac{h}{2}$$

$$K_{31} = u_{2,i} + \frac{h}{2} \cdot K_{22}$$

$$K_{32} = u_{1,i} + \frac{h}{2} \cdot K_{21} - x_i - \frac{h}{2}$$

$$K_{41} = u_{2,i} + h \cdot K_{32}$$

$$K_{42} = u_{1,i} + h \cdot K_{31} - x_i - h$$

$$u_{1,i+1} = u_{1,i} + \frac{h}{6} \cdot \left(K_{11} + 2 \cdot K_{21} + 2 \cdot K_{31} + K_{41}\right)$$

$$u_{2,i+1} = u_{2,i} + \frac{h}{6} \cdot \left(K_{12} + 2 \cdot K_{22} + 2 \cdot K_{32} + K_{42}\right)$$

$$i = 0,1 \ldots m$$

$$h = 1/m$$

Being the result with this method for 8 times steps: $Y(1) = 3{,}7183$

Comparing the results with the exact solution at $y(1) = 3{,}7183$, it is possible to conclude that despite the computational effort between methods has been equalled, only RK4 gives a result equal to the exact solution for five significant digits. Forward Euler method with $m = 32$ is still far away from the exact solution, whereas Heun method with $m = 16$ is able to give a solution with three significant digits.

2. Check the convergence of the methods: plot the logarithm of the error at the end point $x = 1$ vs. the logarithm of the number of function evaluations.

3

Theoretically, the analysed methods of solving ODEs in this assignment, are of the following orders:

- Forward Euler method: order 01.
- Heun method: order 02.
- RK4: order 04.

Therefore the truncation error of each method is:

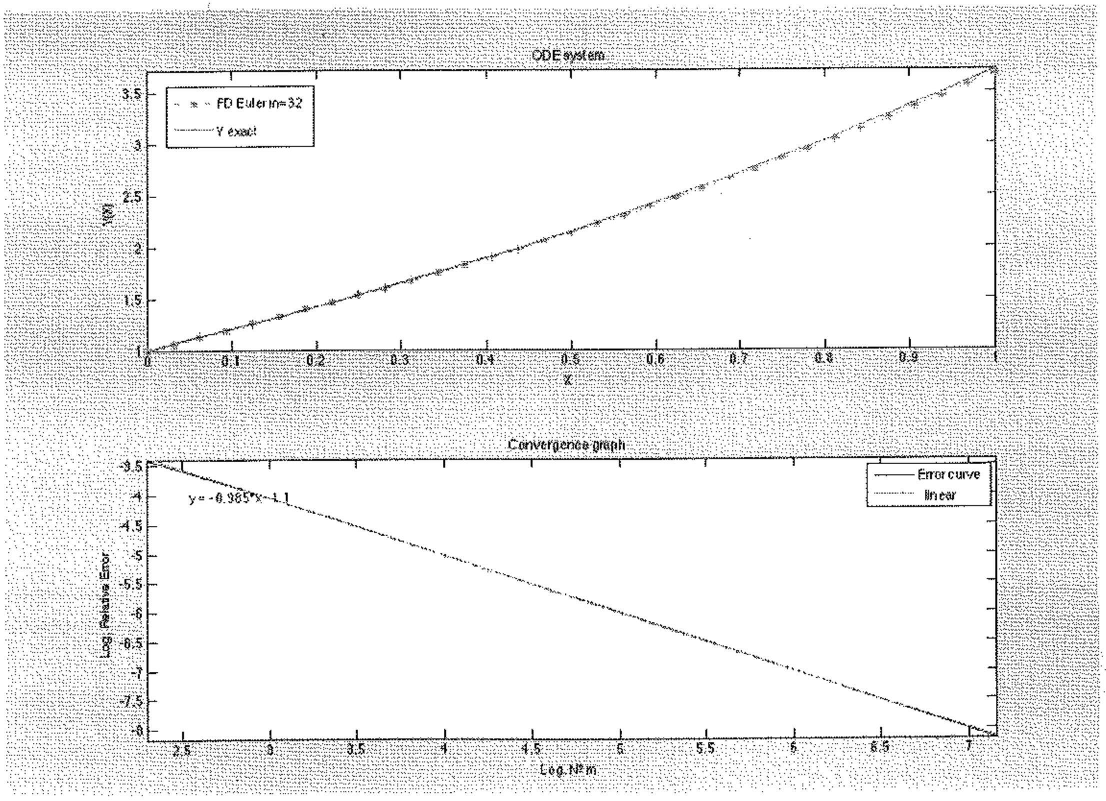
- Forward Euler method: $\vartheta(h)$.
- Heun method: $\vartheta(h^2)$.
- RK4: $\vartheta(h^4)$.

Then, if the logarithm of the relative error is plotted vs. the logarithm of the number of time steps for each method, it is expected to see three different straight lines with the following slopes:

- Forward Euler method: -1.
- Heun method: -2.
- RK4: -4.

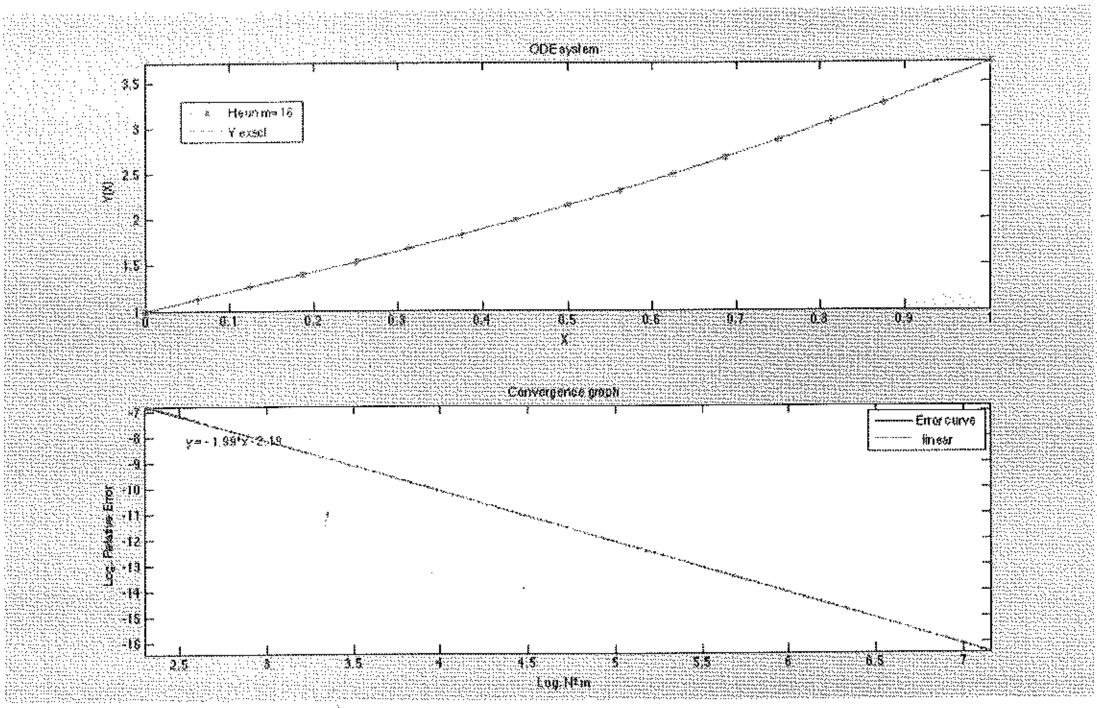
Studying this convergence in Matlab leads to theses graphs:

Forward Euler method



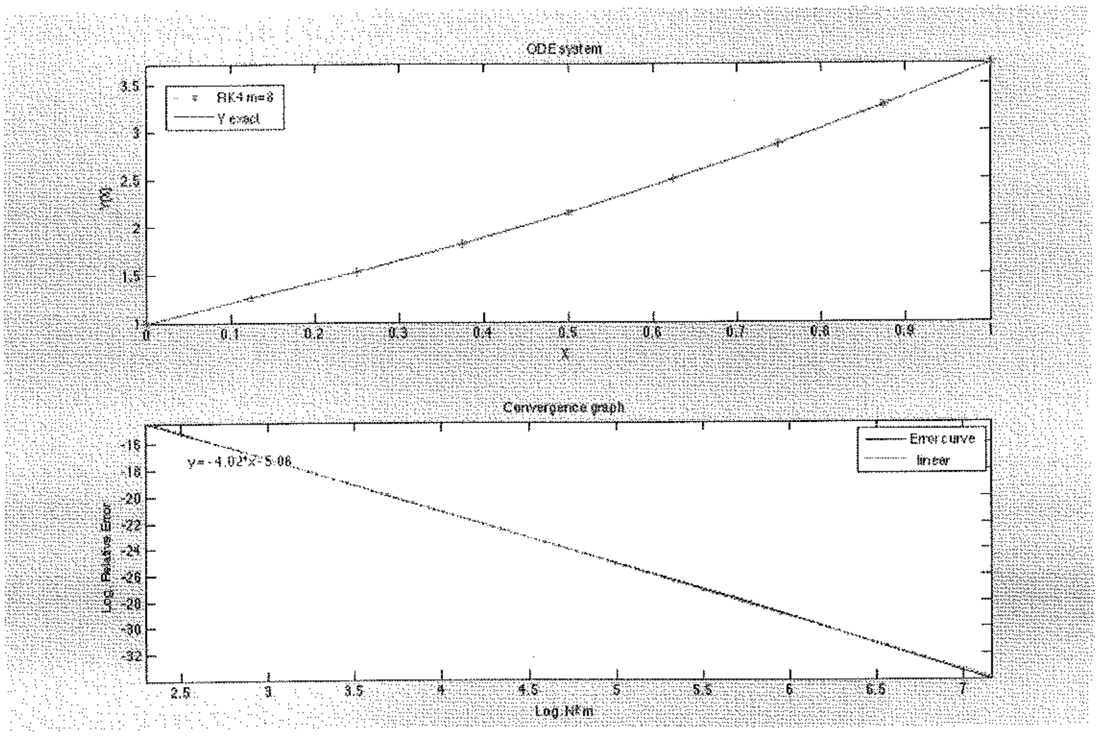
The computed slope is -0,985, which is approximately -1,0.

Heun method



The computed slope is -1,99, which is approximately -2,0.

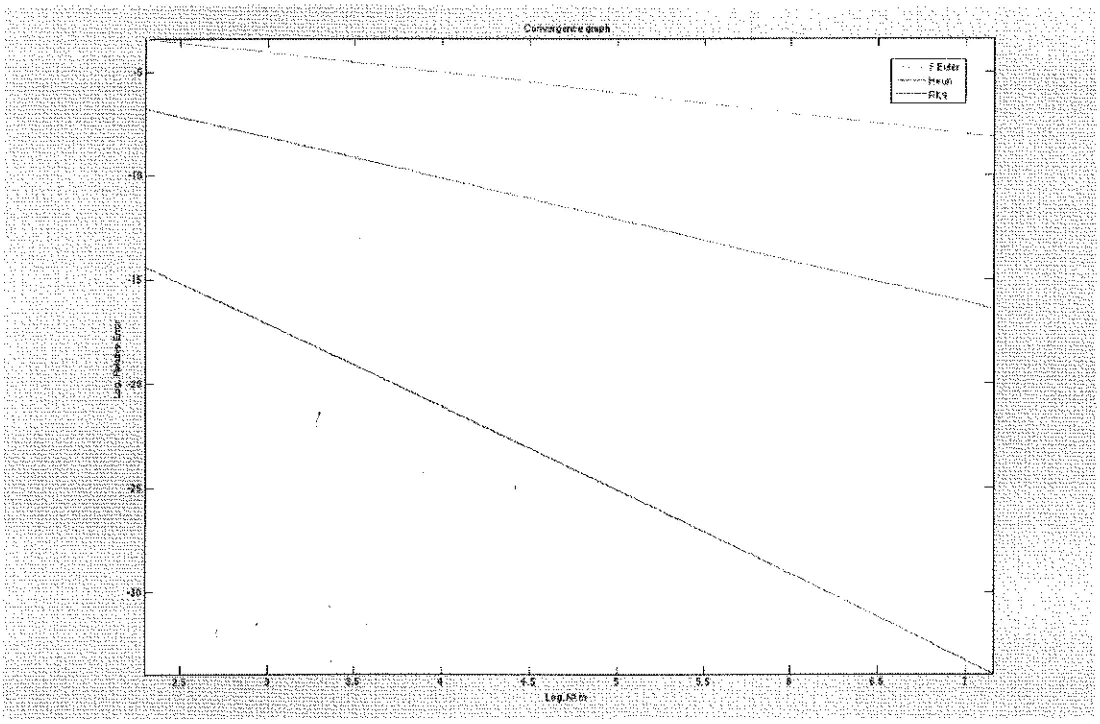
RK4 method



The computed slope is -4,02, which is approximately -4,0.

The theoretical convergence rates are achieved on the base of these last results.

Finally, in order to illustrate the difference in the convergence rates for the three studied methods, the following graph is presented.



2.5

3. Solve the IVP problem with the code *ode45* Matlab function.

This function corresponds with the Runge-Kutta-Fehlberg 45 method. In contrast to the methods presented in this assignment, the code *ode45* is adaptive. It means that the method adapts the number and position of the points, and then *h*, during the iteration in order to ensure that the local error is within the specified bond of accuracy or tolerance.

The local error is computed at each iteration as the difference of the approximation of the solution using a Runge-Kutta method of order 4, and the approximation of the solution using a Runge-Kutta method of order 5. If the difference between the two approximations satisfies the tolerance, the approximation is accepted; otherwise the step size is reduced. Also if both approximations agree to more significant digits than required, the step size can be increased.

By default, the approximation of the solution calculated with *ode45* has been computed with a relative error tolerance of $1 \cdot 10^{-3}$ and an absolute error tolerance of $1 \cdot 10^{-6}$. The more are decreased both tolerances, the better accuracy the approximation will have. These parameters can be changed through the function *odeset*.

Solving this IVP problem with the tolerances set by default, the computed approximation is 3,7183.

Computational cost (in terms of # of evaluations)?

Appendix 01. Routine for the Forward Euler method.

```
close all; clc

m=32;
h = 1/m;
x = linspace(0,1,m+1);

for i=1:m+1
    u1(i) = 0;
    u2(i) = 0;
end

u1(1) = 1;
u2(1) = 2;

for i = 1:m
    u1(i+1) = u1(i) + h*u2(i);
    u2(i+1) = u2(i) + h*(u1(i)-x(i));
end

u1(m+1)

for i=1:m+1
    yex(i) = exp(x(i))+x(i);
end

yex(m+1)

subplot(2,1,1),plot(x,u1,'*-.r','Linewidth',1)
hold on
subplot(2,1,1),plot(x,yex,'-b','Linewidth',1)
xlabel('X')
ylabel('Y(X)')
legend('FD Euler m=32','Y exact')
title('ODE system')
axis tight

exac = yex(m+1);
m = [10 20 40 80 160 320 640 1280];

for i=1:length(m)

    h = 1/m(i);

    xe = linspace(0,1,m(i)+1);

    for j=1:m(i)+1
        u1e(j) = 0;
        u2e(j) = 0;
    end

    u1e(1) = 1;
    u2e(1) = 2;

    for j = 1:m(i)
        u1e(j+1) = u1e(j) + h*u2e(j);
        u2e(j+1) = u2e(j) + h*(u1e(j)-xe(j));
    end

    errorvecfeuler(i) = (abs(exac - u1e(m(i)+1))/exac);

end

subplot(2,1,2),plot(log(m),log(errorvecfeuler),'-k','Linewidth',1)
xlabel('Log. Nº m')
ylabel('Log. Relative Error')
legend('Error curve')
title('Convergence graph')
axis tight

clear a b h u1 u2 u1e u2e x xe yex exac i j
```

Appendix 02. Routine for the Heun method.

```
close all; clc

m=16;
h = 1/m;
x = linspace(0,1,m+1);

for i=1:m+1
    u1(i) = 0;
    u2(i) = 0;
end

u1(1) = 1;
u2(1) = 2;

for i = 1:m
    u1(i+1) = u1(i) + (h/2)*(u2(i) + u2(i) + h*(u1(i) - x(i)));
    u2(i+1) = u2(i) + (h/2)*(u1(i) - x(i) + u1(i) + h*u2(i) - x(i+1));
end

u1(m+1)

for i=1:m+1
    yex(i) = exp(x(i))+x(i);
end

yex(m+1)

subplot(2,1,1),plot(x,u1,'*-.r','Linewidth',1)
hold on
subplot(2,1,1),plot(x,yex,'-b','Linewidth',1)
xlabel('X')
ylabel('Y(X)')
legend('Heun m=16','Y exact')
title('ODE system')
axis tight

exac = yex(m+1);
m = [10 20 40 80 160 320 640 1280];

for i=1:length(m)

    h = 1/m(i);

    xe = linspace(0,1,m(i)+1);

    for j=1:m(i)+1
        u1e(j) = 0;
        u2e(j) = 0;
    end

    u1e(1) = 1;
    u2e(1) = 2;

    for j = 1:m(i)
        u1e(j+1) = u1e(j) + (h/2)*(u2e(j) + u2e(j) + h*(u1e(j) - xe(j)));
        u2e(j+1) = u2e(j) + (h/2)*(u1e(j) - xe(j) + u1e(j) + h*u2e(j) - xe(j+1));
    end

    errorvecheun(i) = (abs(exac - u1e(m(i)+1))/exac);;

end

subplot(2,1,2),plot(log(m),log(errorvecheun),'-k','Linewidth',1)

xlabel('Log. N} m')
ylabel('Log. Relative Error')
legend('Error curve')
title('Convergence graph')
axis tight

clear a b h u1 u2 u1e u2e x xe yex exac i j
```

Appendix 03. Routine for the RK4 method.

```
close all; clc

m = 8;
a = 0;
b = 1;
h = (b-a)/m;
x = linspace(a,b,m+1);

for i=1:(m+1)
    u1(i)=0;
    u2(i)=0;
end

u1(1) = 1;
u2(1) = 2;

for i=1:m
    k11 = u2(i);
    k12 = u1(i) - x(i);
    k21 = u2(i) + (h/2)*k12;
    k22 = u1(i) + (h/2)*k11 - x(i) - h/2;
    k31 = u2(i) + (h/2)*k22;
    k32 = u1(i) + (h/2)*k21 - x(i) - h/2;
    k41 = u2(i) + h*k32;
    k42 = u1(i) + h*k31 - x(i) - h;
    u1(i+1) = u1(i) + (h/6)*(k11 + 2*k21 + 2*k31 + k41);
    u2(i+1) = u2(i) + (h/6)*(k12 + 2*k22 + 2*k32 + k42);
end

u1(m+1)

for i=1:m+1
    yex(i) = exp(x(i))+x(i);
end

yex(m+1)

subplot(2,1,1),plot(x,u1,'*-.r','Linewidth',1)
hold on
subplot(2,1,1),plot(x,yex,'-b','Linewidth',1)
xlabel('X')
ylabel('Y(X)')
legend('RK4 m=8','Y exact')
title('ODE system')
axis tight

exac = yex(m+1);

m = [10 20 40 80 160 320 640 1280];

for i=1:length(m)

    h = 1/m(i);

    xe = linspace(0,1,m(i)+1);

    for j=1:m(i)+1
        u1e(j) = 0;
        u2e(j) = 0;
    end

    u1e(1) = 1;
    u2e(1) = 2;

    for j = 1:m(i)
        k11 = u2e(j);
        k12 = u1e(j) - xe(j);
        k21 = u2e(j) + (h/2)*k12;
        k22 = u1e(j) + (h/2)*k11 - xe(j) - h/2;
        k31 = u2e(j) + (h/2)*k22;
        k32 = u1e(j) + (h/2)*k21 - xe(j) - h/2;
        k41 = u2e(j) + h*k32;
        k42 = u1e(j) + h*k31 - xe(j) - h;

        u1e(j+1) = u1e(j) + (h/6)*(k11 + 2*k21 + 2*k31 + k41);
        u2e(j+1) = u2e(j) + (h/6)*(k12 + 2*k22 + 2*k32 + k42);
    end
    errorvecrk4(i) = (abs(exac - u1e(m(i)+1))/exac);;
end

subplot(2,1,2),plot(log(m),log(errorvecrk4),'-k','Linewidth',1)
xlabel('Log. Nº m')
ylabel('Log. Relative Error')
legend('Error curve')
title('Convergence graph')
axis tight

clear a b h u1 u2 u1e u2e x xe yex exac i j k11 k12 k21 k22 k31 k32 k41 k42
```

Appendix 04. Routine for the global convergence graph.

```
plot(log(m),log(errorvecfeuler),'-r','Linewidth',1);

hold on

plot(log(m),log(errorvecheun),'-b','Linewidth',1);

hold on

plot(log(m),log(errorvecrk4),'-k','Linewidth',1);

xlabel('Log. Nº m')
ylabel('Log. Relative Error')
legend('F Euler','Heun','RK4')
title('Convergence graph')
axis tight
```

Appendix 05. Definition of the problem to be solved using *ode45*.

```
function xp = F(x,u)
xp = zeros(2,1);
xp(1) = u(2);
xp(2) = u(1) - x;
```

```
[x,u] = ode45('F',[0,1],[1,2]);

[x,u(:,1)]
```

9

Homework 1: ODE

STUDENT 1

(EXCEL·LENT)

$$\begin{aligned} y'' &= y - x \quad in\ (0,1) \\ y(0) &= 1,\ y'(0) = 2 \end{aligned} \tag{1}$$

Reducing the 2^{nd} order ODE to 2 1^{st} order ODE.

$$\begin{aligned} \frac{d\boldsymbol{y}}{dx}(x) = \boldsymbol{f}(x,\boldsymbol{y}) \quad x \in (0,1) \\ \boldsymbol{y}(0) = \boldsymbol{\alpha} \\ \boldsymbol{y} = \begin{bmatrix} y_{(1)} \\ y_{(2)} \end{bmatrix},\ \boldsymbol{f} = \begin{bmatrix} y_{(2)} \\ y_{(1)} - x \end{bmatrix},\ \boldsymbol{\alpha} = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \end{aligned} \tag{2}$$

1) Comparision between Euler, Heun and 4^{th} order Runge-Kutta methods

4

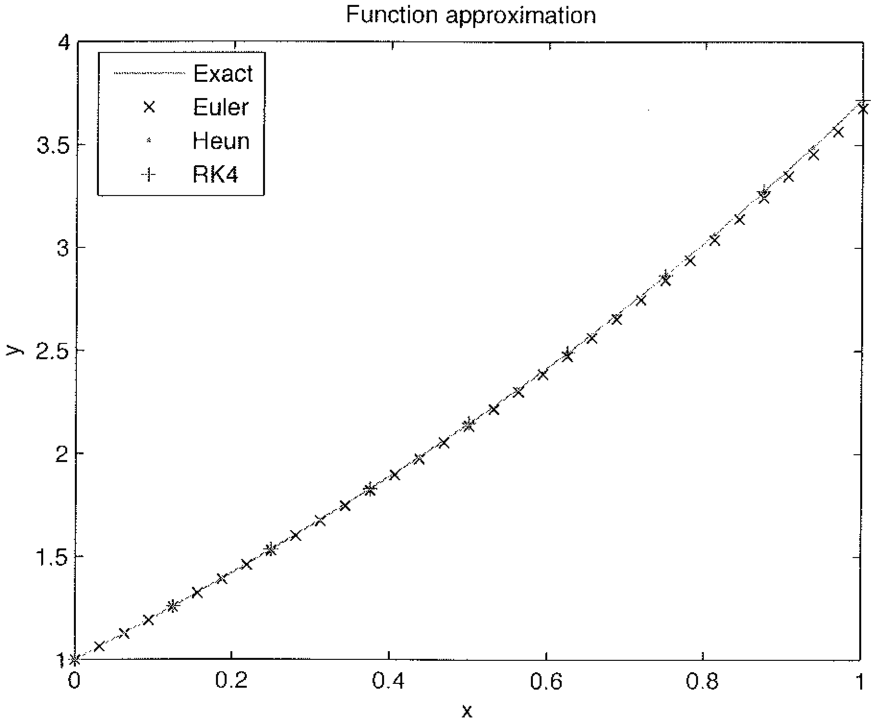
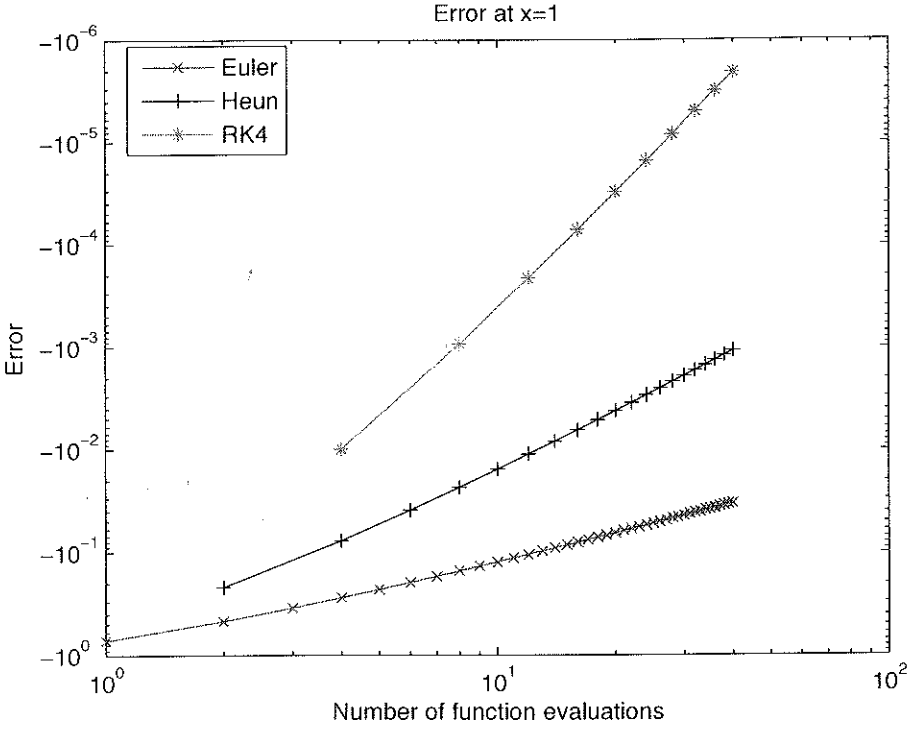


Figure 1: Function approximation

? For the same computational cost the 3 methods behave similar, although the better approximation is given by the order of the method. So the 4^{th} order Runge-Kutta method exhibits better results than the Heun method, and this presents better results than the Euler method. The 3 methods show suitable results given that the equations to be solve are of first order, and the order of approximation of the methods is equal or greater than one.

2) Logarithm of the error

3



✓

Figure 2: Log Error at x=1

The error results agree with the theoretical ones, each method present better approximation with more function evaluations; and the slope of each curve is the same than the method order of convergence (Euler=1, Heun=2, RK4=4).

3) ODE45

2

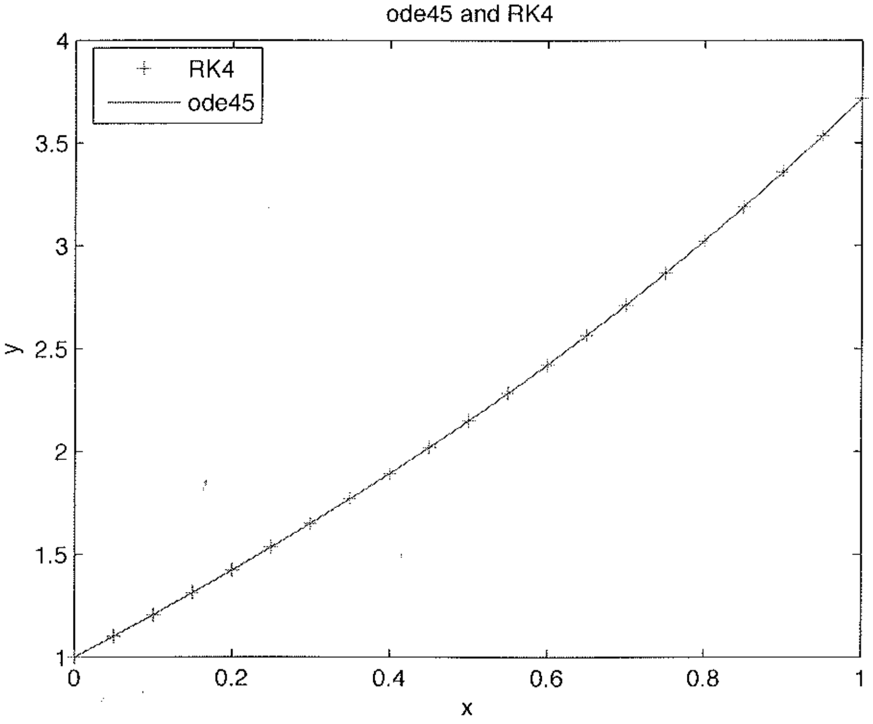


Figure 3: ODE45 - RK4

The ode45 Matlab function uses a RK4 method for solving single or multiple differential equations, as shown in figure 3 the Matlab function and the one written from the RK4 method have the same behavior. The RK4 method have an accuracy of 4^{th} order using 4 function evaluations, which present the better relation between accuracy and number of function evaluations in contrast with higher order methods. As the RK4, the ode45 result could improve increasing the number of function evaluations or decreasing the step (which is the same), so it is necessary to adjust the options using the optimset Matlab function.

HOMEWORK 1

STUDENT 2

(EXCEL·LENT)

9.5

02/12/13

Numerical Methods for PDEs
HOMEWORK 03

Compute the numerical solution of the following the initial value problem (IVP)

$$\begin{gathered} y'' = y - x \quad \text{in } (0,1) \\ y(0) = 1,\ y'(0) = 2 \end{gathered}$$

The exact solution of this problem is $y = exp(x) + x$.

- Implement a routine for the solution of IVP using Euler, Heun and 4^{th} order Runge-Kutta method. Plot the solution obtained with 8 time steps of RK4 and compare it with the Euler and Heun results for an equivalent computational cost (i.e. same number of function evaluations). Draw some conclusions.
- Check the convergence of the methods: plot the logarithm of the error at the end point $x = 1$ vs the logarithm of the number of function evaluations. Do the results agree with the theoretical convergence rates? Comment the results.
- Solve the IVP with the ode45 Matlab function. Which method corresponds to this function? What can you ensure about the accuracy of the obtained solution? How could you improve the accuracy?

1.

First of all notice that the IVP has a second order ODE, therefore is needed to define a system of equations of first order ODEs and then apply Euler, Heun and 4^{th} order Runge-Kutta to the system.

Let us define $y_1 = y$ and $y_2 = y'$, therefore the IVP reads:

$$\begin{gathered} y_1' = y_2 \\ y_2' = y_1 - x \quad \text{in } (0,1) \\ y_1(0) = 1,\ y_2(0) = 2 \end{gathered}$$

and in matrix form $\vec{y}' = \vec{f}(x, \vec{y})$, where:

$$\vec{y}' = \begin{bmatrix} y_1' \\ y_2' \end{bmatrix} \quad \vec{f} = \begin{bmatrix} y_2 \\ y_1 - x \end{bmatrix} \quad \vec{y} = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$$

Once we have the system of equations that we have to solve, let's recall the equations of each method:

- Euler:

$$\vec{Y}_{i+1} = \vec{Y}_i + h\vec{f}(x_i, \vec{Y}_i)$$

- Heun:

$$\vec{Y}_{i+1} = \vec{Y}_i + \frac{h}{2}[\vec{k}_1 + \vec{k}_2] \quad \text{where} \begin{cases} \vec{k}_1 = \vec{f}(x_i, \vec{Y}_i) \\ \vec{k}_2 = \vec{f}(x_i + h, \vec{Y}_i + h\vec{k}_1) \end{cases}$$

- Runge-Kutta 4:

$$\vec{Y}_{i+1} = \vec{Y}_i + \frac{h}{6}[\vec{k}_1 + 2\vec{k}_2 + 2\vec{k}_3 + \vec{k}_4] \quad \text{where} \begin{cases} \vec{k}_1 = \vec{f}(x_i, \vec{Y}_i) \\ \vec{k}_2 = \vec{f}(x_i + \frac{h}{2}, \vec{Y}_i + \frac{h}{2}\vec{k}_1) \\ \vec{k}_3 = \vec{f}(x_i + \frac{h}{2}, \vec{Y}_i + \frac{h}{2}\vec{k}_2) \\ \vec{k}_4 = \vec{f}(x_i + h, \vec{Y}_i + h\vec{k}_3) \end{cases}$$

In order to implement all three methods in matlab, three different function has been written. But all three has similar structures. First, all three need the same four parameters to work, initial approximation (Y0) an interval (int:=[a,b]) where the differential equation is going to be evaluated, the number of steps (s) used to approximate the solution and of course the function f (f(x,Y)). Second, the first part of all three functions is common and computes:

u

1. The step size (h).
2. Sets the matrix where solutions at each step are going to be stored (Y1), which for this particular case has dimensions #equations by s+1.
3. Computes a vector that contains all x_i equally spaced from a to b.
4. Finally the first row of Y1 is defined using initial conditions.

After this part all three methods can be implemented straight forward inside a for loop. Which runs i form 1 to s, and in each step computes the approximation i+1. Finally, at the end of the loop, (Y1) contains all approximations and it's returned as the value of the function. Notice that, in the first row the approximation of the function is written and in the next rows the approximations of the derivatives are written. (See files FEulerSODE.m, FHeunSODE.m, FRK4SODE.m).

Once we have all three methods implemented if we compute the approximation of the previous function with 8 steps, for Runge Kutta 4 (32 evaluations of f), 16 steps for Heun and 32 steps for Euler. We obtain the following result:

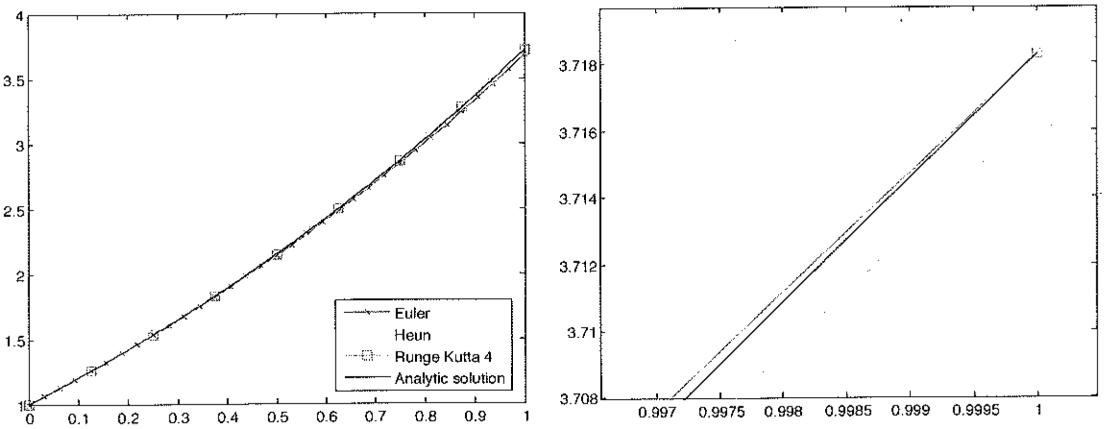
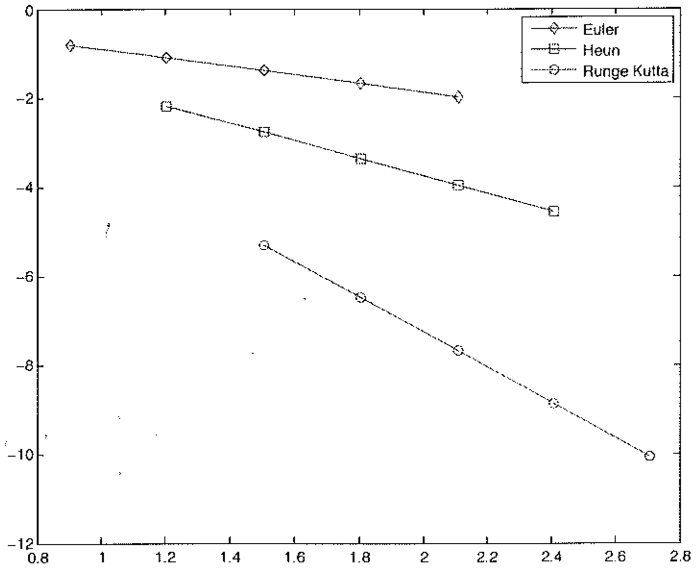


Figure 1: Approximation comparison. a) Analytical solution vs approximation with all three methods and b) Detail comparison of analytical solution vs Runge-Kutta and Heun.

As it can be seen, all three approximations are close to the analytic solution, only Euler at the end is slightly different. But even Heun and Runge-Kutta give very close approximation, if we zoom in the right side of the chart we notice that Runge-Kutta approximates better the solution. Notice for the same computational effort Runge-Kutta gives better results, this is due to the fact that using this method, several evaluations of f are performed in order to compute the slope of approximated solution at each step, while Euler only uses information at point i and Heun at points i and $i+1$.

2.

In order to compute the convergence plot, the error at point $x = 1$ has been computed using all three methods for 8, 16, 32, 64 and 128 steps. The resulting plot is:



3

Figure 2: Convergence plot. Error vs number of function evaluations.

Directly in Figure 2, can not be appreciated but using basic fitting toolbox from matlab, the slopes are -0.96 for Euler, -2 for Heun and -4 for Runge-Kutta. This match ~~with~~ the theoretical convergence rates, because Euler is a first order method, Heun is second order and Runge-Kutta is forth order. In the previous plot is not shown but for a low number of steps (< 8) theoretical convergence rates are not achieved, because these rates are proved for a sufficient quantity of steps. For these particular case, in order to achieve theoretical convergence rates, we have to use at least 8 steps.

3.

9

This method corresponds to the Runge-Kutta 45 method which performs 6 evaluations of f at each step, it is a forth order method, but uses a fifth order approximation for computing an estimation of the relative error at each step. By using this strategy, this method is able to control step size goodness. This means that for at each step if the relative error is bigger than a tolerance the step size can be reduced; and viceversa, if the relative error is lower than a given number the step size can be increased and therefore solve the ode in a more efficient way.

As well as the relative, the absolute error can also be computed, and this means that the accuracy of this method can be ensured by imposing small enough tolerances on the relative and the absolute error. Therefore the accuracy can be improved by reducing the tolerances.

In matlab this tolerances are set with the function "odeset" and then introduced in the function ode45 (See lines 67 and 68 of file main.m). By using this ode45 the method is more efficient and the difference between using this matlab function and the Runge-Kutta 4 with 8 steps is $6.42 \cdot 10^{-9}$.

unclear conclusion: which one would you recommend to use?

2.5

Numerical methods for PDEs — Homework 2

Finite Differences — **Due: December 17, 2013**

The compressed file `FD_Parabolic1D.zip` contains Matlab codes for the finite difference numerical solution of a 1D parabolic equation. *Incomplete* codes for the explicit (FTCS) and implicit (BTCS) methods are provided. Routines to check the convergence in time and in space are also provided.

1. Complete the coding of the FTCS and BTCS methods.
2. Code the Crank-Nicolson method.
3. Test the three finite difference methods with the following numerical parameters:
   - M=10, final time=0.1, number of steps=24
   - M=10, final time=0.1, number of steps=20
   - M=10, final time=0.1, number of steps=18
   - M=10, final time=0.1, number of steps=5

   Do the finite difference methods behave as expected? Discuss your results.
4. Check the convergence of the explicit, implicit and Crank-Nicolson methods (plot them in the same graph) and discuss the results.

STUDENT 1

(APROVAT)

PDE

HOMEWORK 2

Assumptions $u_t = u_{xx}$

Interval $a = 0$

$b = 1$

1.

Coding For the complete FTCS se the file parabolic_ex.m and for the complete BTCS se the file parabolic_im.m.

Equations FTCS

$$U_i^{n+1} = r\cdot U^n_{i-1} + (1 - 2r)\cdot U_i^n + r\cdot U^n_{i+1}$$

BTCS

$$U_i^n = -r\left(U_{i-1}\right)^{n+1} + (1 + 2r)U_i^{n+1} - r\left(U_{i+1}\right)^{n+1}$$

2.

Coding For the complete Crank-Nicoson se the file parabolic_ex.m.

Equation

$$-\frac{1}{2}\cdot r\cdot\left(U_{i-1}\right)^{n+1} + (1 + r)\cdot\left(U_i\right)^{n+1} - \frac{1}{2}\cdot r\cdot\left(U_{i+1}\right)^{n+1} = \frac{1}{2}\cdot r\cdot\left(U_{i-1}\right)^n \ldots$$
$$+ (1 - r)\cdot\left(U_i\right)^n + \frac{1}{2}\cdot r\cdot\left(U_{i+1}\right)^n$$

3.

Cases All the cases is solved with the Implict Method (BTCS).
The discretization is the following:

$M = 10$

final time . = 0.1

number of timesteps . = 20

Case 1

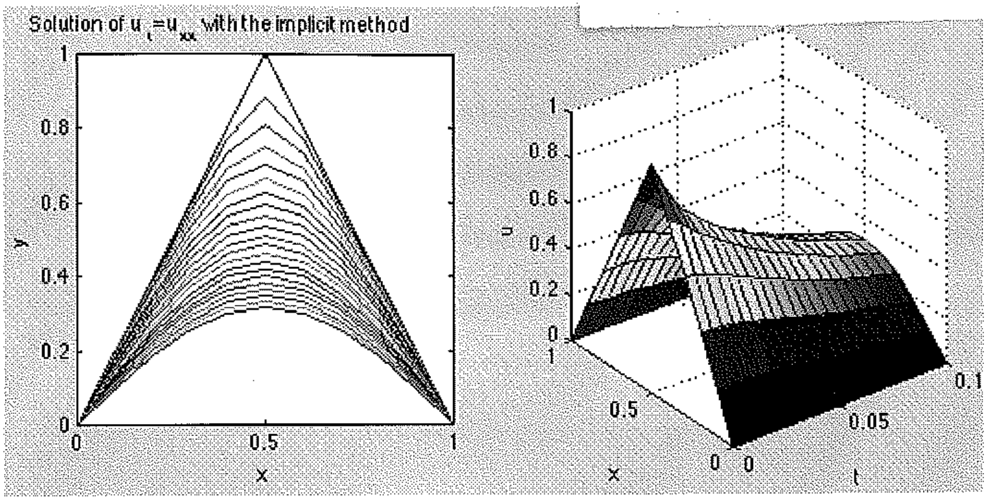
Initial condition

$f = 1 - 2\cdot|x - 0.5|$

Boundary condition

$g = 0$

$h = 0$



Case 2

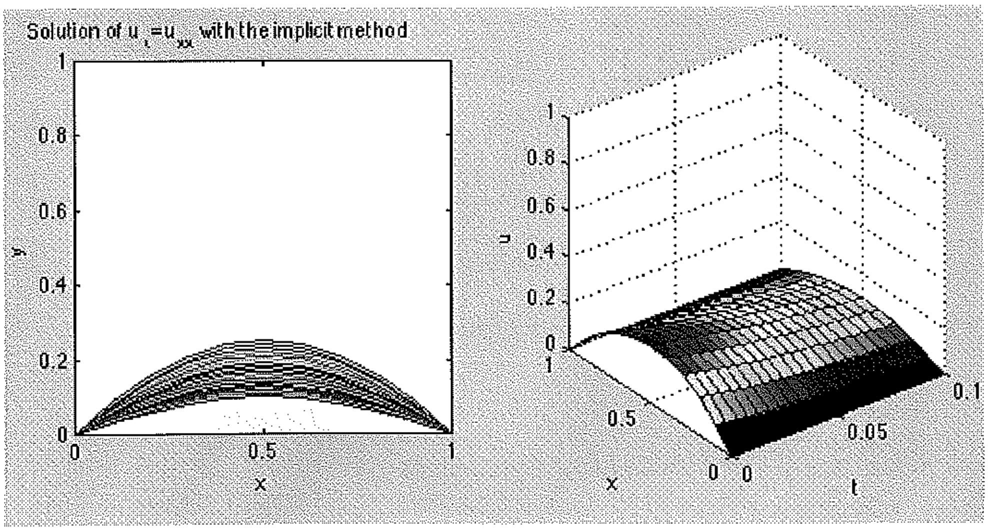
Initial condition

$f = x \cdot (1 - x)$

Boundary condition

$g = 0$

$h = 0$



Case 3

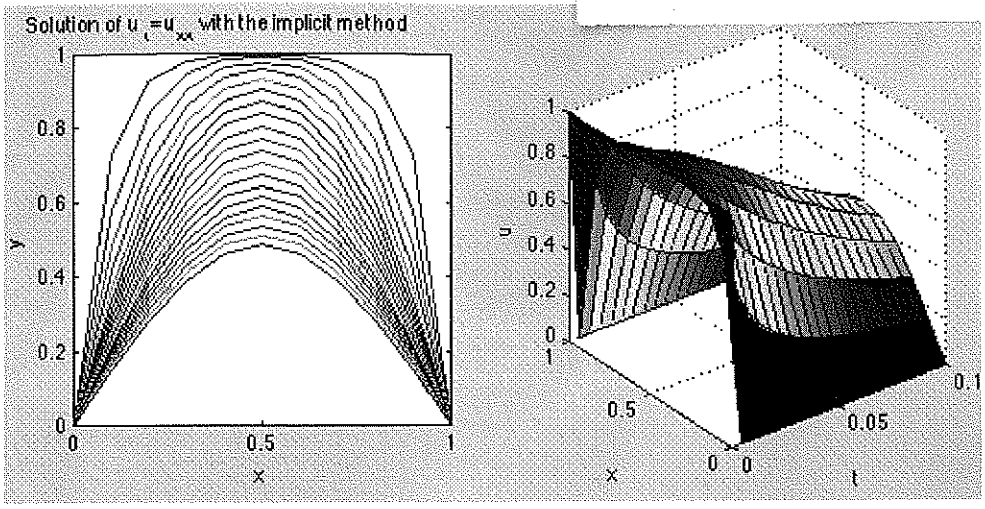
Initial condition

$f = 1$

Boundary condition

$g = 0$

$h = 0$



Case 4

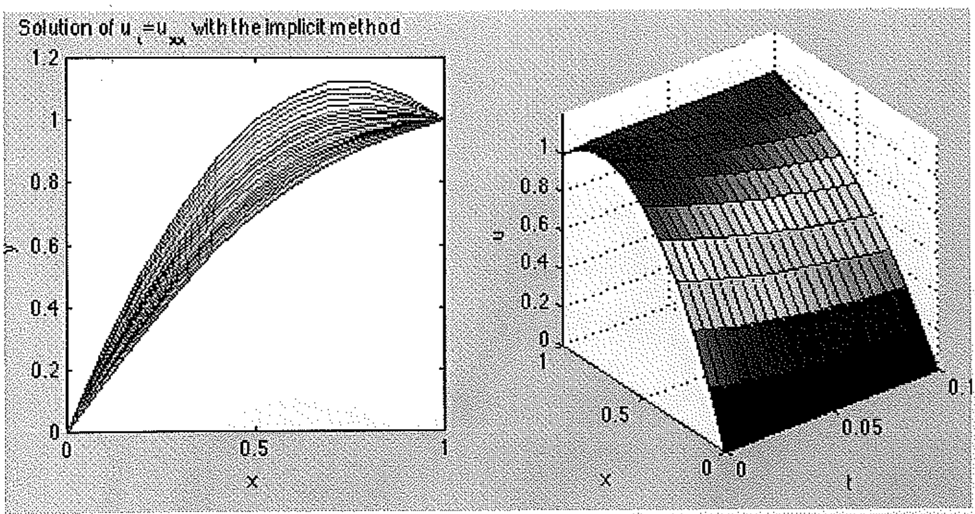
Initial condition

$f = x\cdot(3 - 2x)$

Boundary condition

$g = 0$

$h = 1$



Case 5

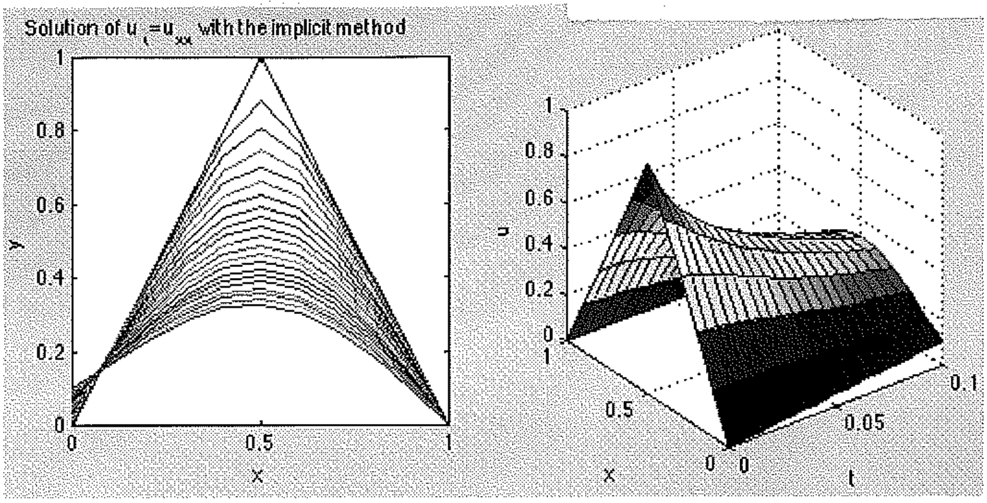
Initial condition

$f = 1 - 2\,|x - 0.5|$

Boundary condition

$g = t$

$h = 0$



All cases behave as expected. Boundary coditions and initial condtions are fulfilled for each case. The equation do not have any external source, and it is therefor expected that the u function is going towards a straight line for time going towards infinity, which is seen to be true for all cases.

Methods

Case 2

Initial condition

$f = x \cdot (1 - x)$

Bondary condition

$g = 0$

$h = 0$

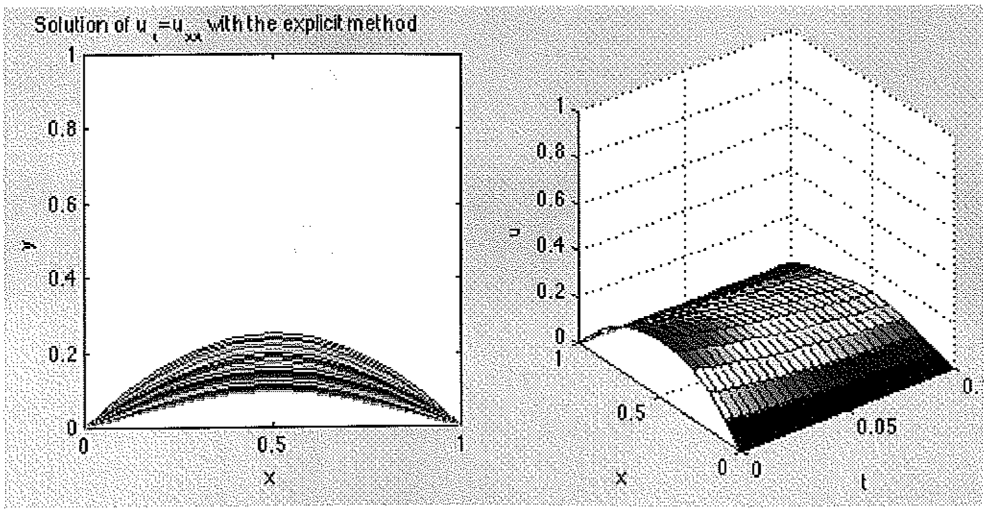
Explicit method (FTCS)

Discretization 1

M = 10

final time . = 0.1

number of timesteps . = 24

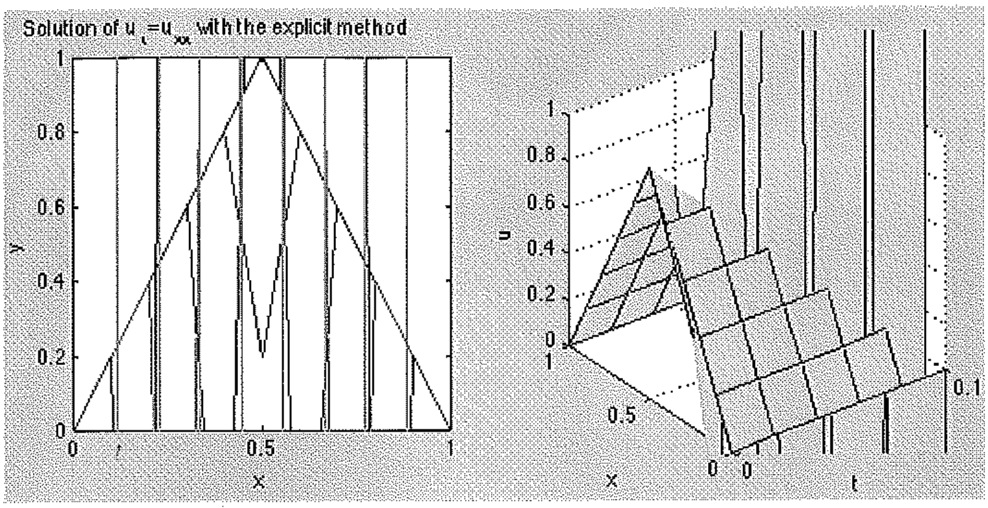


Discretization 2

M = 10

final time . = 0.1

number of time steps . = 5



The explicit method is unstable for certin time steps. The critirum for stability is:

$$r = \frac{\Delta t}{\Delta x^2} \le \frac{1}{2}$$

Discretization 1 $\quad r = \dfrac{\frac{0.1}{24}}{\left(\frac{b-a}{10}\right)^2} = 0.42 \quad < \quad \dfrac{1}{2} \quad \Rightarrow$ stable

Discretization 2 $\quad r = \dfrac{\frac{0.1}{5}}{\left(\frac{b-a}{10}\right)^2} = 2 \quad > \quad \dfrac{1}{2} \quad \Rightarrow$ unstable

That is why the plots in the discretization with only 5 timesteps is giving a wrong result. But the method does behave as expected becuase it is expected to be unstable.

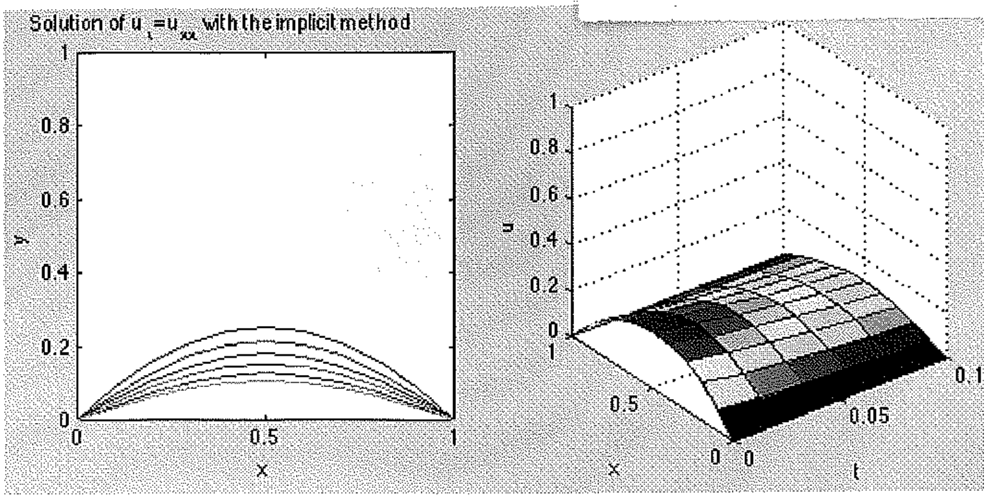
Implicit method (BTCS)

Discretization

M = 10

final time . = 0.1

number of timesteps . = 5



The method is unconditionally stable for all timesteps wich was expeted.

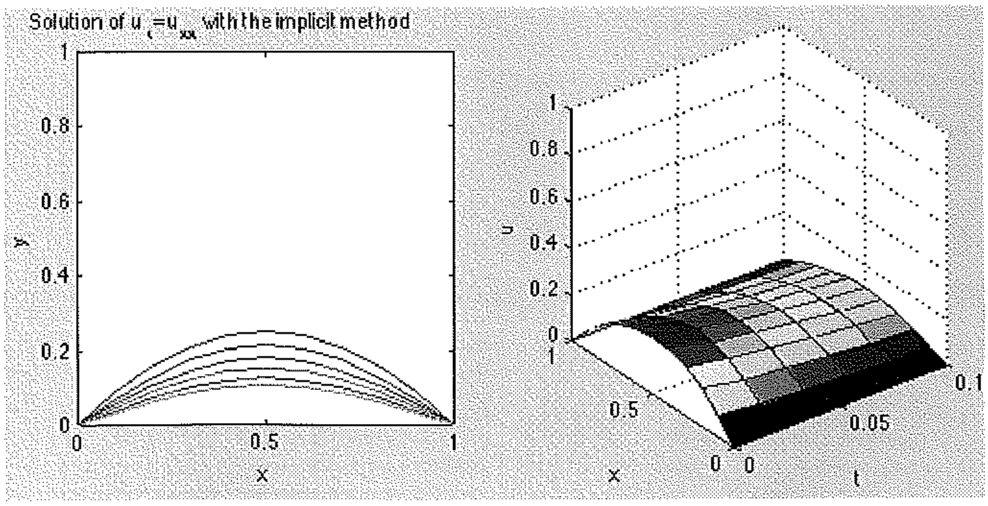
Crank-Nicolsons method

Discretization

M = 10

final time . = 0.1

number of timesteps . = 5



This method is also unconditionally stable for all timesteps.

4.

Convergence

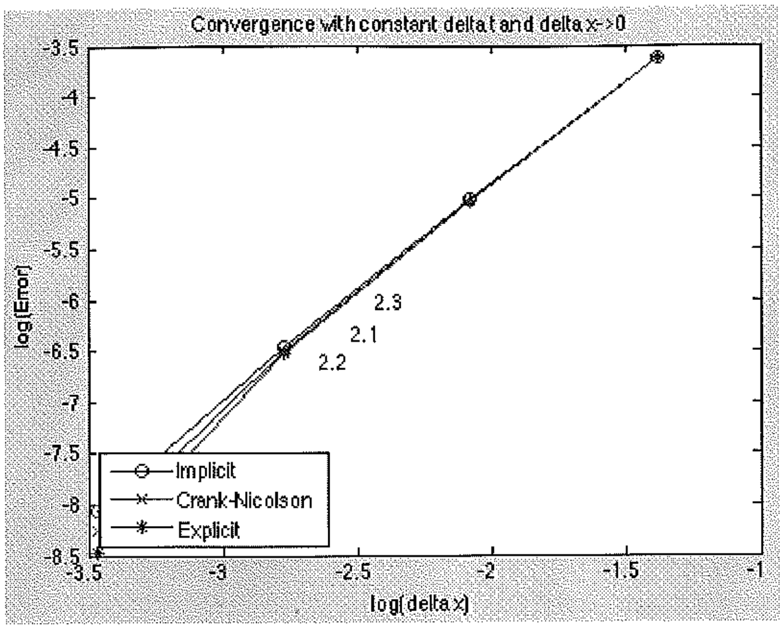
Error

Explicit and implicit $\tau_i^{n+\theta} = O\left(\Delta t, \Delta x^2\right)$ (1)

Crank-Nicolson $\tau_i^{n+\theta} = O\left(\Delta t^2, \Delta x^2\right)$ (2)

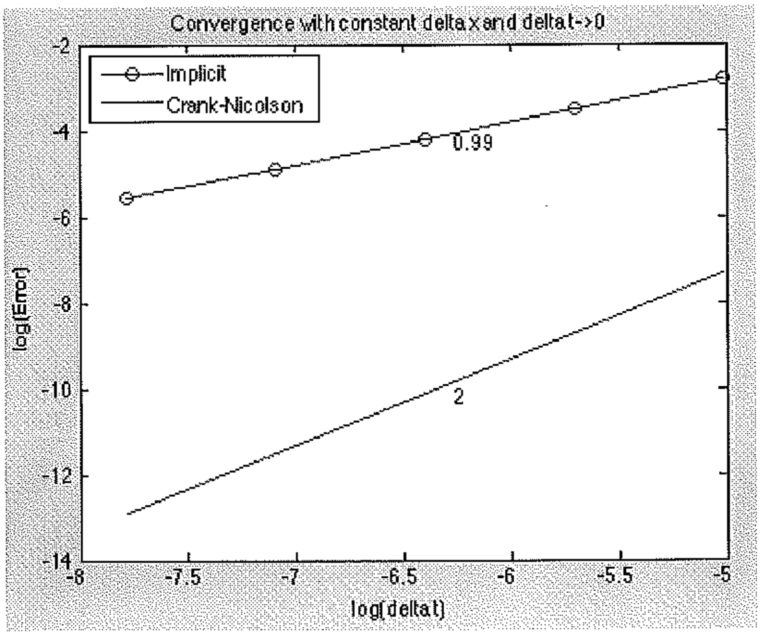
Convergence for the space denpenden part

The expected slope for all the tree diffenrent methods must be equal to 2 do to (1) and (2). The convergence for the space denpenden part is order 2 for all methods.



Convergence for the time denpenden part

The expected slope for the explicit and the implicit methods is 1 and 2 for Crank-Nicolson when it is the convergence conserning time. It does not make sence to plot explicit convergence because of the stability problems, when the time intervals get to big.



It is seen from the plots, that all the methods converges as expected. The Crank-Nicolson method is the most exact, because it has a convergence order equal to 2 in borh space and time, but it is also a method with more calculations steps. The explicit method is the fastes method, but it is unconditionally stable.

STUDENT 2

(APROVAT)

Homework 2

The compressed file FD_Parabolic1D.zip contains Matlab codes for the finite difference numerical solution of a 1D parabolic equation. Incomplete codes for the explicit (FTCS) and implicit (BTCS) methods are provided. Routines to check the convergence in time and in space are also provided.

1. Complete the coding of the FTCS and BTCS methods

The codes made in Matlab are attached this paper when delivered.

FTCS

This scheme is also called Forward in Time Centred in Space, which means that the method is based on central difference in space and the forward Euler method in time. This gives a 1st order convergence in time and a 2nd order convergence in space. The numerical problem can be written as:

$$U_i^{n+1} = rU_{i-1}^n + (1-2r)U_i^n + rU_{i+1}^n \quad where \quad i = 1,....,M, \quad n \geq 0$$

The method is explicit and the solution can easily be computed. But the scheme is conditionally stable, meaning that it is stable and convergent whenever r ≤ ½. With r defined as $r = \frac{\Delta t}{\Delta x^2}$.
The method have no oscillation if r ≤ ¼.

BTCS

The scheme is called Backward in Time Centered in Space. This means that the method is based on central difference in space and the backward Euler method in time. The numerical problem can be written as:

$$U_i^n = U_i^{n+1} - rL(U_i^{n+1}) \quad where \quad i = 1,....,M, \quad n \geq 0$$
$$and \quad L(U_i^{n+1}) = U_{i-1}^{n+1} - 2U_i^{n+1} + U_{i+1}^{n+1}$$

The method is an implicit scheme and is solved by solving a linier system of equations.
BTCS is unconditionally stable and are non-oscillating.

2. Code the Crank-Nicolson method

The codes made in Matlab are attached this paper when delivered.
The numerical problem for solving with Crank-Nicolson is shown below.

$$U_i^{n+1} - \theta * r * L(U_i^{n+1}) = U_i^n + (1-\theta) * r * L(U_i^n) \ \ where\ i = 1, \dots, M, \qquad n > 0$$

$$L(U_i^{n+1}) = U_{i-1}^{n+1} - 2 * U_i^{n+1} + U_{i+1}^{n+1} \ and\ L(U_i^n) = U_{i-1}^n - 2 * U_i^n + U_{i+1}^n$$

The Crank-Nicolson is also an implicit method that is computed by solving a linier system of equations. The scheme is unconditionally stable. But r has to be smaller or equal to 1/2 to give non-oscillation solutions.

3. Test the three finite difference methods with the following numerical parameters:

- M = 10, final time = 0.1, number of steps = 24
- M = 10, final time = 0.1, number of steps = 20
- M = 10, final time = 0.1, number of steps = 18
- M = 10, final time = 0.1, number of steps = 5

Do the finite difference methods behave as expected? Discuss your results.

All of the figures from 1-6 does not show the initial conditions.

The graphs in figure 1 and 2 is made using FTCS for Example 1 with the following BC's and IC's.

Bondary conditions $u(0,t) = 0 \; and \; u(1,t) = 0$

Initial conditions $u(x,o) = 1 - 2 * abs(x - 0{,}5)$

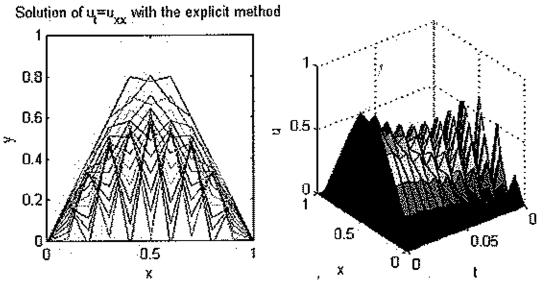


Figure 1 - FTCS for example 1, with 18 steps.

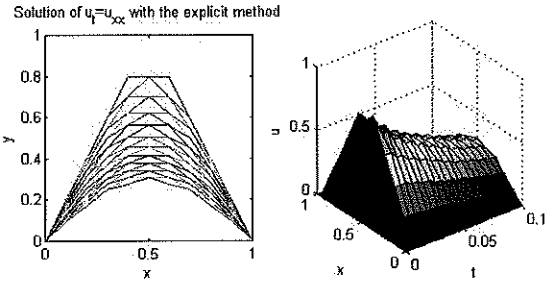


Figure 2 - FCTS for example 1, with 20 steps.

The factor r is calculated for each computation in matlab. They are shown below.

$r = \frac{\Delta t}{\Delta x^2} = \frac{0{,}0056}{0{,}1^2} = 0{,}56 > 0{,}5 \; not \; stable$ $r = \frac{\Delta t}{\Delta x^2} = \frac{0{,}0050}{0{,}1^2} = 0{,}50 \leq 0{,}5 \; stable$

Figure 1 shows that the method is unstable and figure 2 is stable because r = 0,5.
This also indicates that the scheme is acting as expected, since it is conditionally stable as stated in question 1.

The graphs in figure 3 and 4 is made using BTCS for Example 2 with the following BC's and IC's.

Bondary conditions $u(0,t) = 0 \; and \; u(1,t) = 0$

Initial conditions $u(x,o) = x * (1 - x)$

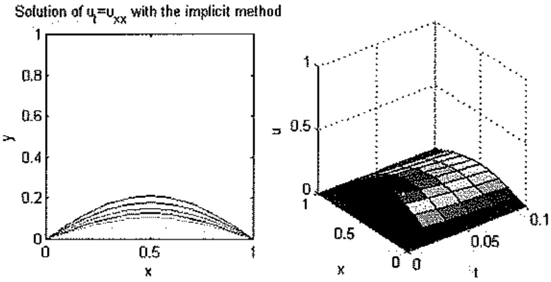


Figure 4 - BTCS for example 2, with 5 steps.

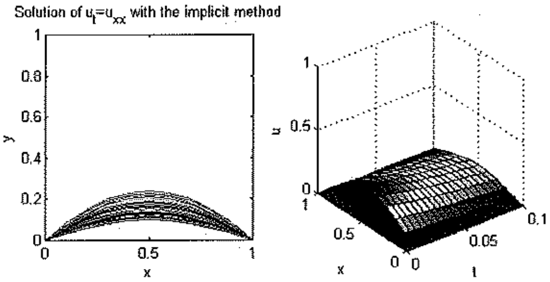


Figure 3 - BTCS for example 2, with 20 steps.

The graphs (figure 3 and 4) behave as expected, since they both are stable and there are no oscillations. By increasing the time steps the solution gets more and more exact.

The graphs in figure 5 and 6 is made using Crank-Nicolson for Example 5 with the following BC's and IC's.

Bondary conditions $u(0,t) = t \text{ and } u(1,t) = 0$

Initial conditions $u(x,o) = 1 - 2 * abs(x - 0{,}5)$

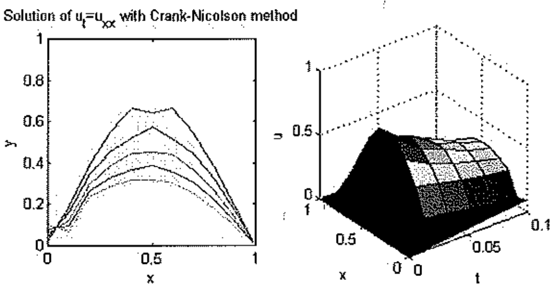


Figure 5 - Crank-Nicolson for example 5, with 5 steps.

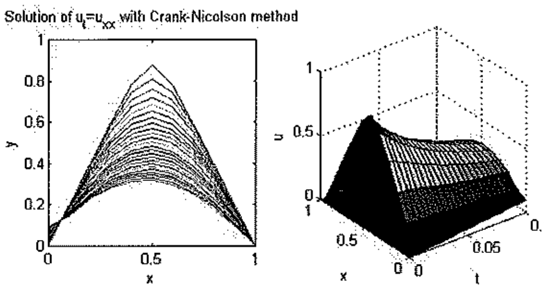


Figure 6 - Crank-Nicolson example 5, with 24 steps.

The factor r is calculated for each computation in matlab. They are shown below.

$$r = \frac{\Delta t}{\Delta x^2} = \frac{0{,}02}{0{,}1^2} = 2 > 0{,}5 \ oscillation$$

$$r = \frac{\Delta t}{\Delta x^2} = \frac{0{,}0042}{0{,}1^2} = 0{,}42 \leq 0{,}5 \ non\ oscillation$$

The graphs (figure 5 and 6) behave as expected. They are both stable, and shows as the time increases so do one of the boundary conditions.
It can also be seen that there is small oscillations on the solution shown in figure 5, since the factor is higher than 0,5. As for the BTCS we can increase the time steps and we will get a more exact solution.

By looking at the 3D plot in figure 1-4 we see that the further we get in time the more the solution gets to zero.

That makes good sense, since we are solving a diffusion equation, with both boundary conditions fixed at the value zero and that there isn't any external force that influences the solution.

The 3D plot in figure 5-6 again shows that the solution approaches 0, the further we get in time. Here only one of the boundaries is fixed at zero while the other is a function of t. The graph here also makes good sense.

4. Check the convergence of the explicit, implicit and Crank-Nicolson methods (plot them in the same graph) and discuss the results.

The convergence is first checked with Δt as a variable. That is shown in figure 7.

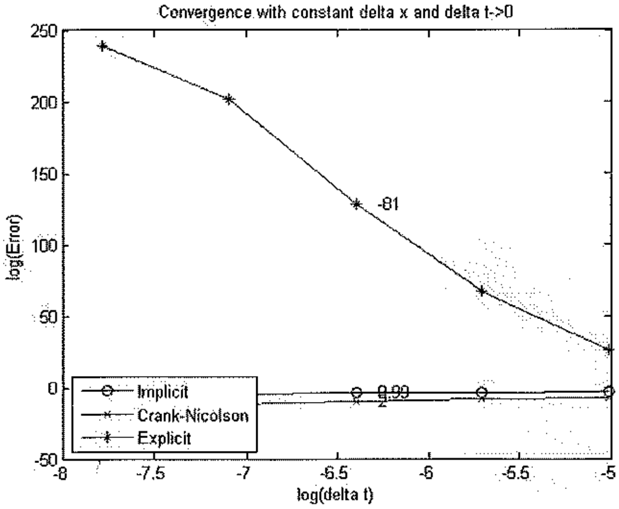


Figure 7 - Convergence with **Δt** as variable.

The figure shows that the implicit method, Crank-Nicolson, have a convergence rate of 2. For the Crank-Nicolson this is as expected because the method have a truncation error of $O(\Delta t^2, \Delta x^2)$.
Furthermore the figure shows that BTCS have a convergence rate of 0,99, this is also as expected because BTCS have a truncation error of $O(\Delta t, \Delta x^2)$. The graph that shows the convergence rate for the explicit method FTCS (red line in figure 7) doesn't make sense because the method is not unconditionally stable.

The convergence is now checked with Δx as a variable. This is shown in figure 8.

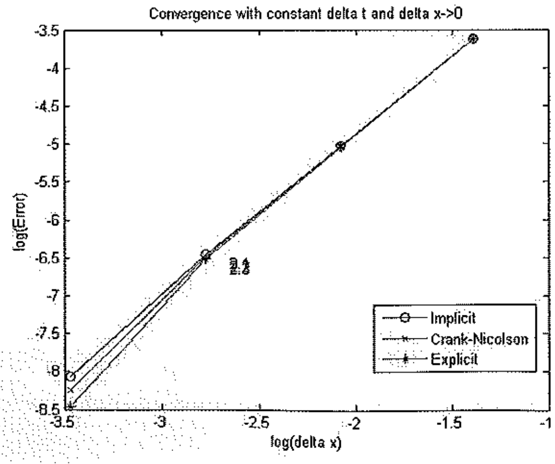


Figure 8 - Convergence with **Δx** as variable.

The three methods are all expected to have the same convergence rate when Δx is the variable, since they all have a truncation error of 2nd order I space. As it is seen on figure 8 they all have a convergence of approximate 2. The reason the explicit method (FTCS) have a stable convergence is that the stability condition $r < ½$ is fulfilled in all 4 points on the graph.

STUDENT 1

(NOTABLE)

NUMERICAL METHODS for PDEs
Master of Science in Computational Mechanics
Fall Semester 2013
Homework 2: Finite Differences

The compressed file `FD_Parabolic1D.zip` contains Matlab codes for the finite difference numerical solution of a 1D parabolic equation. *Incomplete* codes for the explicit (FTCS) and implicit (BTCS) methods are provided. Routines to check the convergence in time and in space are also provided.

1. Complete the coding of the FTCS and BTCS methods.
2. Code the Crank-Nicolson method.
3. Test the three finite difference methods with the following numerical parameters:
   - M=10, final time=0.1, number of steps=24
   - M=10, final time=0.1, number of steps=20
   - M=10, final time=0.1, number of steps=18
   - M=10, final time=0.1, number of steps=5

   Do the finite difference methods behave as expected? Discuss your results.
4. Check the convergence of the explicit, implicit and Crank-Nicolson methods (plot them in the same graph) and discuss the results.

Solution 1:

- FTCS method

```
% Resulting Matrix
U = zeros(m+1, npast+1);

% Initial and boundary conditions in the solution matrix
%(each time step is stored in a column): TO BE CODED
U(1,:) = g; U(m+1, :) = h; U(:,1) = f;

% Loop in time steps: TO BE CODED
for j = 1 : npast % loop over time t 0 : n
for i = 2 : m % loop over x 1 : m
    U(i,j+1) = r*U(i-1,j) + (1-2*r)*U(i,j) + r*U(i+1,j);
end
end
```

The algorithm for this method is straightforward. And can be summarized in three steps:

1. We create the *resulting matrix U* of dimensions [M+1, npast+1]; time steps are stored in columns;

2. We store the boundary conditions g, h in the corresponding rows: 1, M+1; We store the function f at the initial value in the 1st column;
3. Loop over all the time steps:

   Loop over all the spatial nodes:

   $$\text{Compute: } U_i^{j+1} = rU_{i-1}^j + (1-2r)U_i^j + rU_{i+1}^j$$

- BTCS method The solution provided by this method is implicit. We can obtain it by solving the following equation:

  $$\mathbf{A}\mathbf{U}^{n+1} = \mathbf{I}\mathbf{U}^n + \mathbf{F}$$

  We have to fulfill the following properties:

  - dimension of the `A` matrix is `[m-1,m-1]`;
  - $\mathbf{U}^{n+1}, \mathbf{U}^n$ have the dimension `[m-1,1]` omitting the values `U(1)` and `U(m+1)` which contain the boundary conditions;
  - $\mathbf{F} = \begin{bmatrix} rU_0^{n+1} & 0 & \dots & 0 & rU_{M+1}^{n+1} \end{bmatrix}^T$ of dimensions `[m-1,1]`;

```
% Definition of matrix A and decomposition: TO BE CODED

aA = -r; bA = 1+2*r;
% A = zeros(m-1,m-1);
% A(1,1) = bA;
% A(1,2) = aA;
% for i = 2 : m-2
%     A(i,i-1) = aA;
%     A(i,i) = bA;
%     A(i,i+1) = aA;
% end
% A(end,end-1) = aA;
% A(end,end) = bA;

% gamma is the variable container for the Thomas Algorithm
gamma = zeros(m-1,1);
gamma(1) = aA/bA;
for i=2 : m-2
    gamma(i) = aA/(bA-aA*gamma(i-1));
end
% Last element of gamma is 0
```

The *A matrix* is constant in all the iterations so it will be computed only once. Furthermore it is tridiagonal with fixed values of the following structure:

$$\mathbf{A} = \begin{bmatrix} (1+2r) & -r & & \\ -r & (1+2r) & -r & \\ & \ddots & \ddots & \ddots \\ & & -r & (1+2r) \end{bmatrix}$$

Given this simplification we can used the variables: `aA = -r; bA = (1 + 2*r);` as components of the A matrix. Furthermore for the solution of the final matrix equation the *Thomas algorithm* will be implemented, as a result there is no need to store the actual A matrix but it's contents in a matrix `gamma` of dimensions `[m-1,1]`. The matrix will have the following form:

$$\mathbf{A}' = \begin{bmatrix} 1 & \texttt{gamma(1)} & & \\ 0 & 1 & \texttt{gamma(2)} & \\ & \ddots & \ddots & \ddots \\ & & 0 & 1 \end{bmatrix}$$

Since the `A` matrix is not used in matrix form the code generating it is only present in commented form.

$\mathbf{U}^{n+1}$ is obtained from the following system:

$$\mathbf{A}'\mathbf{U}^{n+1} = \rho$$

Where $\rho_i = \frac{U_i^n + F_i + r\rho_{i-1}}{1+2r+r\gamma_{i-1}}$.

```
% Loop in time steps: TO BE CODED
RHS = zeros(m-1,npast);
FB = zeros(m-1,1);

% we compute RHS at t = 0;
FB(1) = r*g(1); FB(end) = r*h(1);
RHS(:,1) = eye(m-1,m-1)*U(2:m,1) + FB;

% Implementation of THOMAS ALGORITHM
rho = zeros(m-1,1);
% Loop over all timesteps (no modifications on original boundary ...
    values 1
% and m+1)
for j = 1 : npast
    % Factorization of the matrices
    % First element for Thomas Algorithm
    rho(1) = RHS(1,j)/bA;

    % Inside elments for Thomas Algorithm
    % Loop over all nodes (except 1 and m-1)
    for k = 2 : length(gamma)-1
        % gamma(k) = aA/(bA-aA*gamma(k-1)); This is computed ...
            initially and
        % kept static during all iterations;
        rho(k) = (RHS(k,j)-aA*rho(k-1))/(bA - aA*gamma(k-1));
    end
    % Last element for Thomas Algorithm
    rho(end) = (RHS(end,j)-aA*rho(end-1))/(bA-aA*gamma(end-1));

    % Thomas algorithm
    % Backward substitution of U n+1 elements
    U(m,j+1) = rho(end);

    for k = 1 : (m-2)
        U(end-1-k,j+1) = rho(end-k) - gamma(end-k)*U(end-k,j+1);
    end

    % Update of RHS
    FB(1) = r*g(j+1); FB(end) = r*h(j+1);
    RHS(:,j+1) = eye(m-1,m-1)*U(2:m,j+1) + FB;
end
```

Solution 2:

The Crank-Nicolson method

The solution of this method is also implicit. We obtain $\mathbf{U}^{n+1}$ by solving the following system:

$$\mathbf{A}\mathbf{U}^{n+1} = \mathbf{B}\mathbf{U}^{n} + \mathbf{F}$$

Where:

$$\mathbf{A} = \begin{bmatrix} 1+r & -\frac{r}{2} & & & \\ -\frac{r}{2} & 1+r & -\frac{r}{2} & & \\ & \ddots & \ddots & \ddots & \\ & & -\frac{r}{2} & 1+r & -\frac{r}{2} \\ & & & -\frac{r}{2} & 1+r \end{bmatrix}; \mathbf{B} = \begin{bmatrix} 1-r & \frac{r}{2} & & & \\ \frac{r}{2} & 1-r & \frac{r}{2} & & \\ & \ddots & \ddots & \ddots & \\ & & \frac{r}{2} & 1-r & \frac{r}{2} \\ & & & \frac{r}{2} & 1-r \end{bmatrix};$$

$$\mathbf{F} = \left[\frac{r}{2}\left(U_0^{n+1} + U_0^n\right) \;\; 0 \;\; \dots \;\; 0 \;\; \frac{r}{2}\left(U_{M+1}^{n+1} + U_{M+1}^n\right)\right]^T$$

The previous properties hold here as well:

- dimensions of $\mathbf{A}, \mathbf{B}$ matrices `[M-1,M-1]`;
- dimensions of $\mathbf{U}^n, \mathbf{U}^{n+1}$ matrices `[M-1,1]` omitting the boundary condition elements: U_0, U_{M+1}
- dimension of $\mathbf{F}$ matrix `[1,M-1]`;

The solution of the system is obtained the same way, using Thomas algorithm. Where the **A** matrix is not used directly but substituted by the `gamma` matrix:

```
% gamma is the variable container for the Thomas Algorithm
gamma = zeros(m-1,1);
gamma(1) = aA/bA;
for i=2 : m-2
    gamma(i) = aA/(bA-aA*gamma(i-1));
end
% Last element of gamma is 0
```

On the other hand the **B** matrix has to be formed to compute the matrix multiplication:

```
B = zeros(m-1,m-1); aB = r/2; bB = 1-r;
% Setup of B matrix (tridiagonal);
B(1,1) = bB; B(1,2) = aB;
for k = 2:(size(B,1)-1)
    B(k,k) = bB;
    B(k,k-1) = aB; B(k,k+1) = aB;
end
B(end,end) = bB; B(end,end-1) = aB;
```

$\mathbf{U}^{n+1}$ is obtained similarly to the implicit method. First we compute the right hand side of the equation stored in the matrix `RHS`. Then we execute the Thomas algorithm to reduce the **A** matrix and the *right hand side*. Once this is obtained we compute the elements of $\mathbf{U}^{n+1}$ in reverse order.

```
% Implementation of THOMAS ALGORITHM
rho = zeros(m-1,1);
% Loop over all timesteps (no modifications on original boundary values 1
% and m+1)
for j = 1 : npast
    % we compute RHS at t = j timestep;
    % Imposing the boundary conditions through F matrix;
    FB(1) = r/2*(g(j)+g(j+1)); FB(end) = r/2*(h(j)+h(j+1));
    % Update of RHS
    % we Compute RHS = B*U(j) + F
    RHS(:,j) = B*U(2:m,j) + FB;

    % First element of tridiag factorization
    rho(1) = RHS(1,j)/bA;

    % Inside elments of tridiag factorization
    % Loop over all nodes (except 1 and m-1)
    for k = 2 : length(gamma)-1
        % gamma(k) = aA/(bA-aA*gamma(k-1)); This is computed initially and
        % kept static during all iterations;
        rho(k) = (RHS(k,j)-aA*rho(k-1))/(bA - aA*gamma(k-1));
    end
    % Last element of tridiag factorization
    rho(end) = (RHS(end,j)-aA*rho(end-1))/(bA-aA*gamma(end-1));

    % Backward substitution of U n+1 elements
    U(m,j+1) = rho(end);

    for k = 1 : (m-2)
        U(end-1-k,j+1) = rho(end-k) - gamma(end-k)*U(end-k,j+1);
    end
end
```

Solution 3:

1. M=10, final time=0.1, number of steps=24 Due to the nature of the FTCS method the effect of the previous elements have a great influence on the following time step. This drawback can be observed in the following comparison

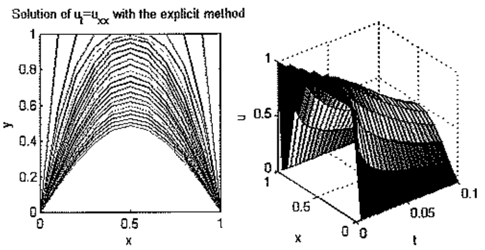


Figure 1: $f = 1$ Explicit method

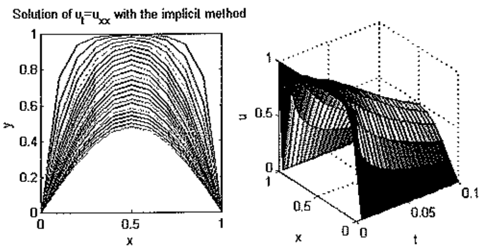


Figure 2: $f = 1$ Implicit method

The initial value of 1 is maintained in a much larger area compared to the implicit method, at which the tendency to lean towards the boundary conditions is greater. The value 1 is maintained until the 5th time step for the Explicit method and only the 2nd for the Implicit method.

2. M=10, final time=0.1, number of steps=20

This effect is further emphasized as we decrease the time resolution.

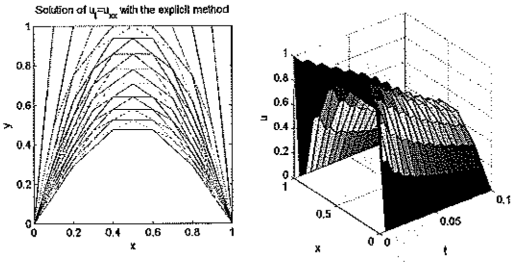


Figure 3: $f = 1$ Explicit method

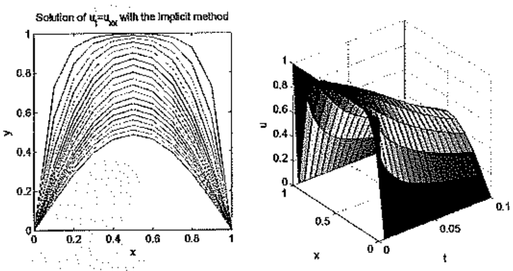


Figure 4: $f = 1$ Implicit method

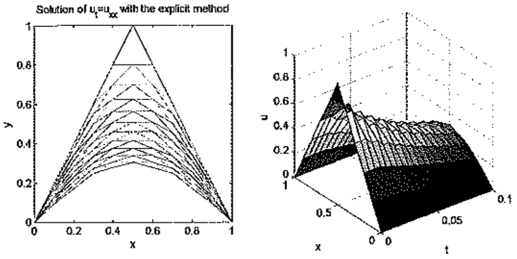


Figure 5: $f = 1 - 2abs(x - 0.5)$
Explicit method

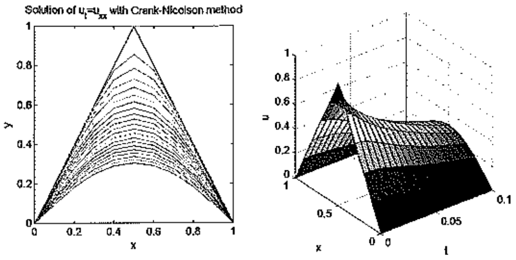


Figure 6: $f = 1 - 2abs(x - 0.5)$
C-N method

At a lower resolution in time the Explicit method provides a worse approximation than the Implicit or C-N methods.

Both the Implicit and C-N methods provide comparable approximations.

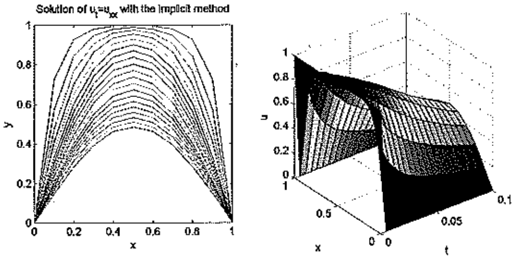


Figure 7: $f = 1$; Implicit method

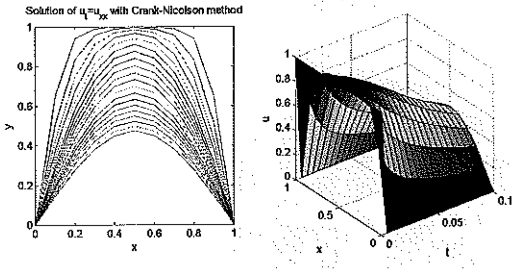


Figure 8: $f = 1$; C-N method

3. M=10, final time=0.1, number of steps=18

   At this resolution we are at the critical value of $r = \frac{\Delta t}{\Delta x^2} = \frac{1}{2}$. The Explicit method should still converge but in the approximation of some functions significant oscillations appeared which make the method unstable for this critical value

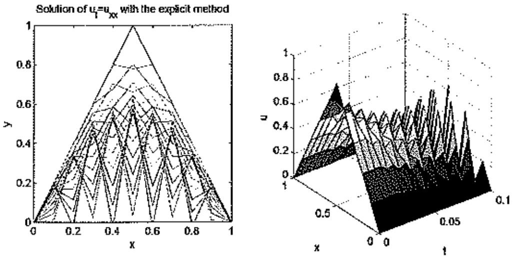


Figure 9: $f = 1 - 2abs(x - 0.5)$; Explicit method

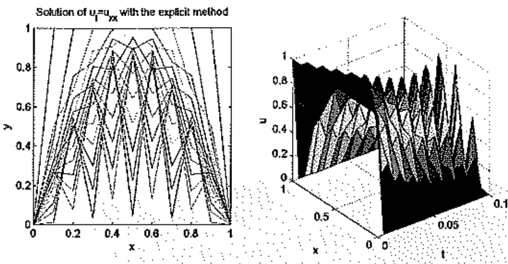


Figure 10: $f = 1$; Explicit method

Although in other cases where the gradient of the obtained **U** field in the t direction is not so big the method is still stable:

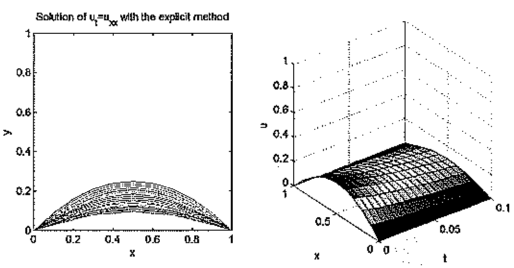


Figure 11: $f = x(1 - x)$;Explicit method

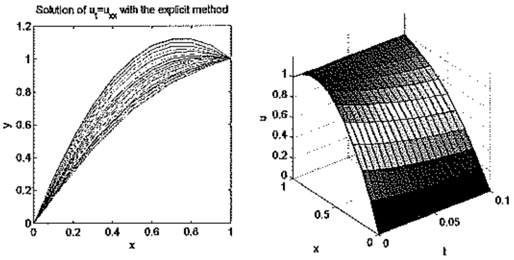


Figure 12: $f = x(3 - 2x)$;Explicit method

4. M=10, final time=0.1, number of steps=5

   With this resolution we go over the critical value value of $r = 1/2$ making the Explicit method completely unstable.

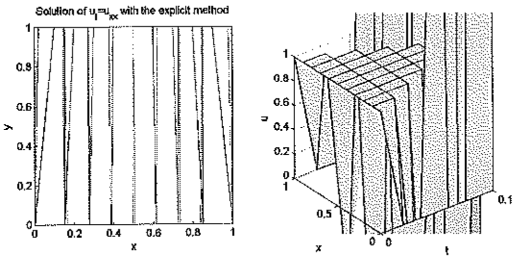


Figure 13: $f = 1$;Explicit method

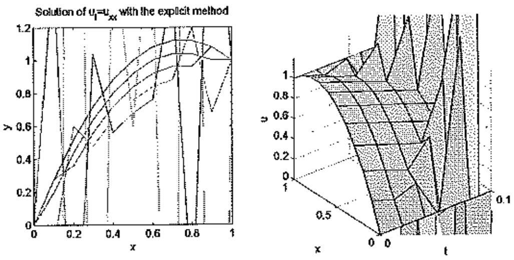


Figure 14: $f = x(3 - 2x)$; Explicit method

Visible difference appears between the Implicit and C-N methods. This is the result of the fact that while the first one has a precision of Δt in time the last one offers Δt^2 precision.

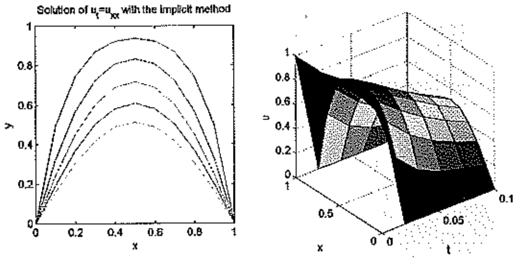


Figure 15: $f = 1$;Implicit method

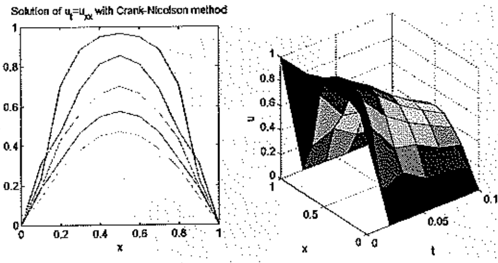


Figure 16: $f = 1$; C-N method

Solution 4:

Since $\mathcal{T} = \mathcal{O}(\Delta t) + \mathcal{O}(\Delta x^2)$ for the Explicit and Implicit methods, and $\mathcal{T} = \mathcal{O}(\Delta t^2) + \mathcal{O}(\Delta x^2)$ we can treat the convergence of the methods from two different viewpoints.

First we fix Δt small enough such that $\mathcal{T} \simeq \mathcal{O}(\Delta x^2)$ and vary Δx to obtain the convergence of the methods with respect to the resolution spatial grid.

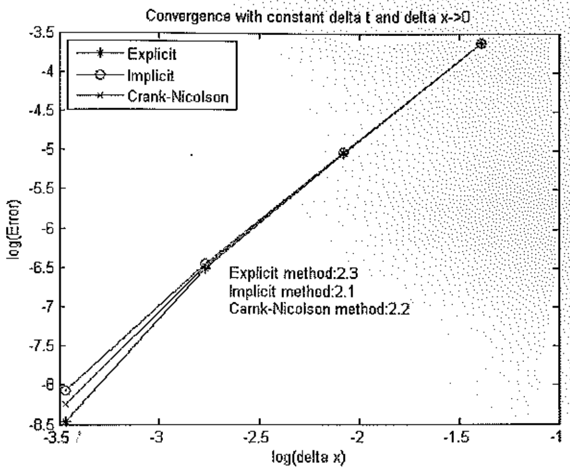


Figure 17: Convergence with respect to Δx^2; $f = sin(x\pi)$

Next we fix Δx small enough such that $\mathcal{T} \simeq \mathcal{O}(\Delta t)$ and vary Δt to obtain the convergence of the methods with respect to the resolution of the time discretization.

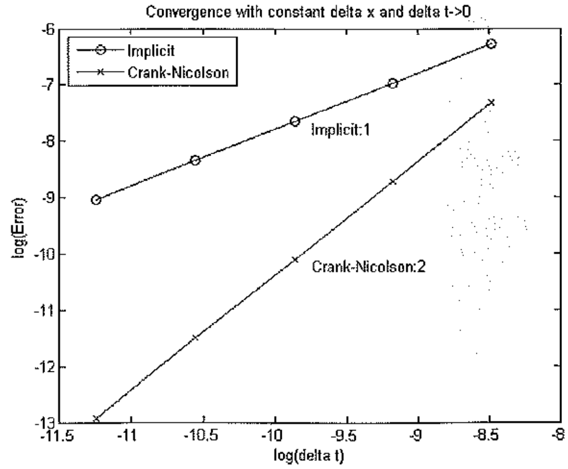


Figure 18: Convergence with respect to Δt; $f = sin(x\pi)$

It is difficult to plot the convergence of the three methods on the same graph, since for many grid resolutions the *Explicit* method is unstable. As a result we plot this convergence on a separate graph. Although in this case Δx is not negligible from total value of $\mathcal{T}$ since it's value is comparable in order to Δt. ($m = 10, t = 0.1, n = [24, 48, 96, 192, 384]$)

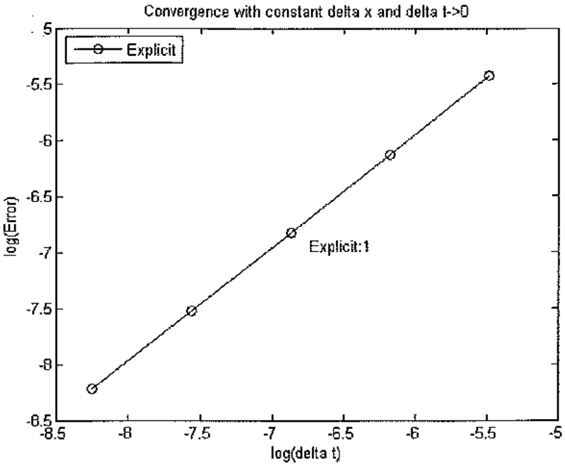


Figure 19: Convergence with respect to Δt - Explicit method; $f = x(1-x)$

Numerical methods for PDEs
Finite Differences.
Homework 02.

STUDENT 2

(NOTABLE)

The compressed file FI **ite difference numerical**
solution of a 1D parab **CS) and implicit (BTCS)**
methods are provided. **ace are also provided.**

1. Complete the code **hods.**

The three codes provided for completion lack the necessary routine to compute the solution at each time step for each point of the domain $x \in [a, b]$. In all of them, the values of the solution have been stored in a matrix with the following characteristics:

- Number of rows = total number of points.
- Number of columns = number of time steps (*nts*) plus one.
- Content of the first column: The values of the solution at the initial time step for each point (Initial condition: $U_i^0 \; i = 1 \dots m+1$ such that m is the number of intervals inside the domain).
- Content of the first row from its second element (1,2): The values of the solution for each time step at the point $x = a$ (Boundary condition at $x = a$: $U_1^n \; n = 1 \dots nts$).
- Content of the last row from its second element (m+1,2): The values of the solution for each time step at the point $x = b$ (Boundary condition at $x = b$: $U_{m+1}^n \; n = 1 \dots nts$).
- Content of the rest of the matrix between its first and last row: at each column the solution vector $\boldsymbol{U}^n = [U_2^n \quad U_3^n \quad \cdots \quad U_{m-1}^n \quad U_m^n]^T \; n = 1 \dots nts$ is stored.

$$U = \begin{bmatrix} U_1^0 & U_1^1 & \dots & U_1^{nts} \\ \vdots & \vdots & \vdots & \vdots \\ U_{m+1}^0 & U_{m+1}^1 & \cdots & U_{m+1}^n \end{bmatrix}$$

Finally, the routines written to complete the code of each method can be consulted in their corresponding Appendices. In order to solve the linear system of equations at each time step in the BTCS and C-N methods, a Cholesky factorization has been used since the matrix of the systems is symmetric and positive definite (because for the solved problem, all its boundaries belong to Dirichlet boundary).

2. Test the three finite difference methods with the following numerical parameters:

Case A. M = 10; final time = 0.1; number of steps = 24.
Case B. M = 10; final time = 0.1; number of steps = 20.
Case C. M = 10; final time = 0.1; number of steps = 18.
Case D. M = 10; final time = 0.1; number of steps = 5.

Among the different problems included in the file *parabolic.m* to test the three methods, the problem chosen is the second one, whose description is as follows:

$$\begin{cases} u_t - u_{xx} = 0 & in\ x \in [0,1] \\ u(x,0) = x \cdot (1-x) & \forall x \in [0,1] \\ u(0,t) = 0 & Dirichlet\ boundary\ condition \\ u(1,t) = 0 & Direchlet\ boundary\ condition \end{cases}$$

Before running the different codes for each case, the discretization in space and time is briefly analysed to foresee problems in the stability of the method and the oscillatory behaviour of the approximated solution.

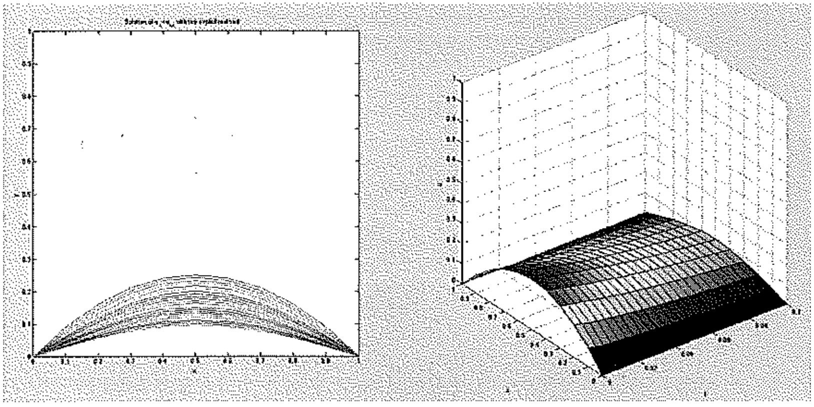
	Δx	Δt	$r = {}^{\Delta t}/_{\Delta x^2}$	**Stability of the method**			**Oscillations in the solution**		
				FTCS	BTCS	C-N	FTCS	BTCS	C-N
Case A	0.1	0.00416	0.416	yes	yes	yes	yes	no	no
Case B	0.1	0.00500	0.500	yes	yes	yes	yes	no	no
Case C	0.1	0.00555	0.555	no	yes	yes	yes	no	yes
Case D	0.1	0.02000	2.000	no	yes	yes	yes	no	yes

As it is known, the FTCS method is unstable for $r > 0.50$ and the solution it computes oscillates unless $r \leq 0.25$. However, BTCS and C-N methods are always stable, although the solution of C-N oscillates for $r > 0.50$.

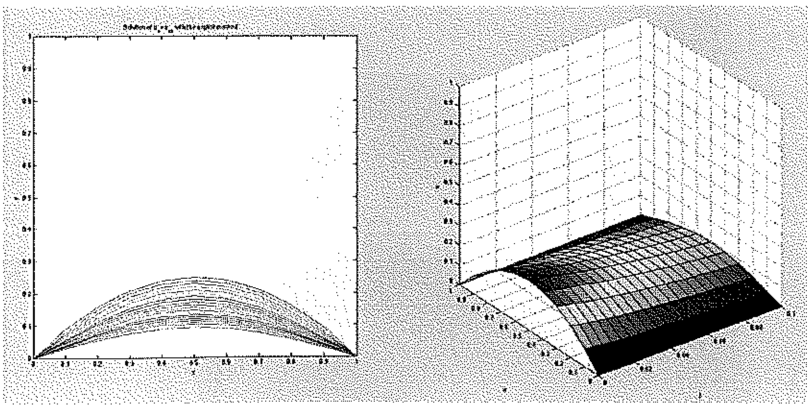
With regard to the oscillating behaviour of the solution for the FTCS and C-N, it is expected to not see it since the final time is only 0.1 seconds.

FTCS

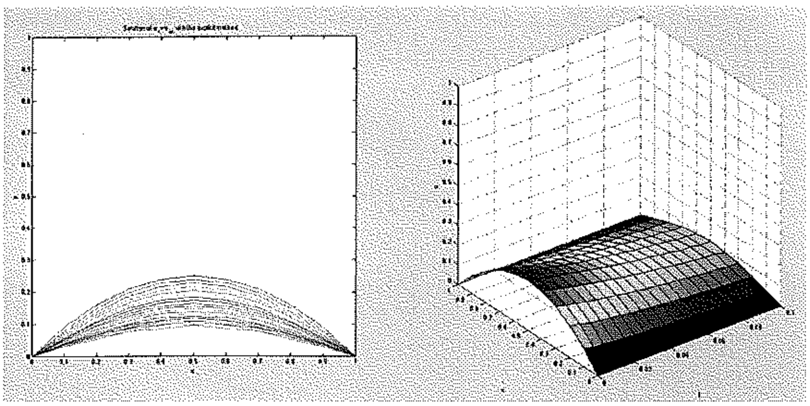
Case A.



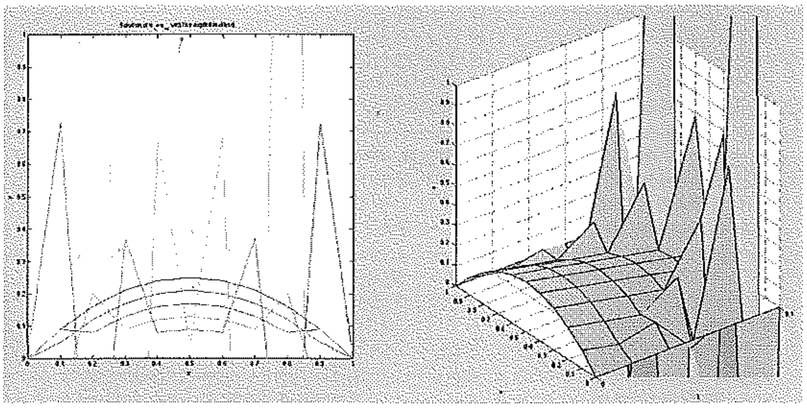
Case B.



Case C.



Case D.

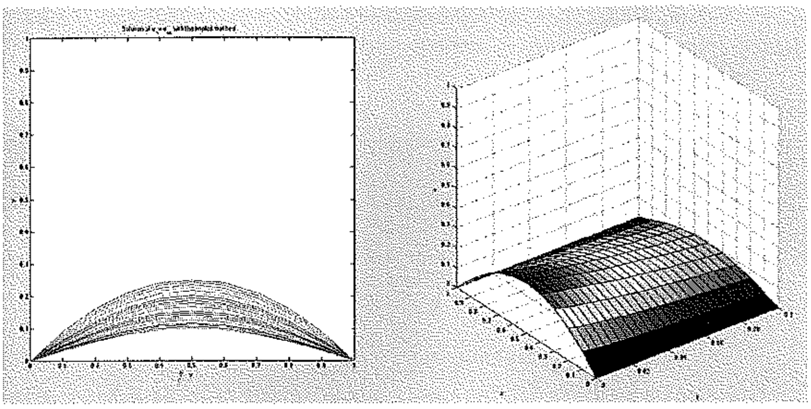


This method shows instability in the *Case D* since the value of r for this case is 2, far away from its critical value (0.5). In the *Case C*, despite of being r greater than 0.5, the instability is not big enough for being visually appreciable.

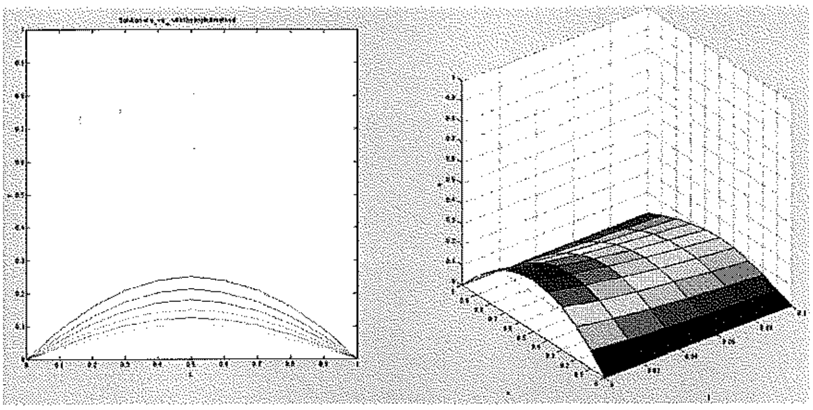
BCTS

Since this method is theoretically stable and its solution does not oscillate for any value of *r*, only the graphs of the extreme cases for the value of *r* (Case A and Case D) are shown below.

Case A.



Case D.

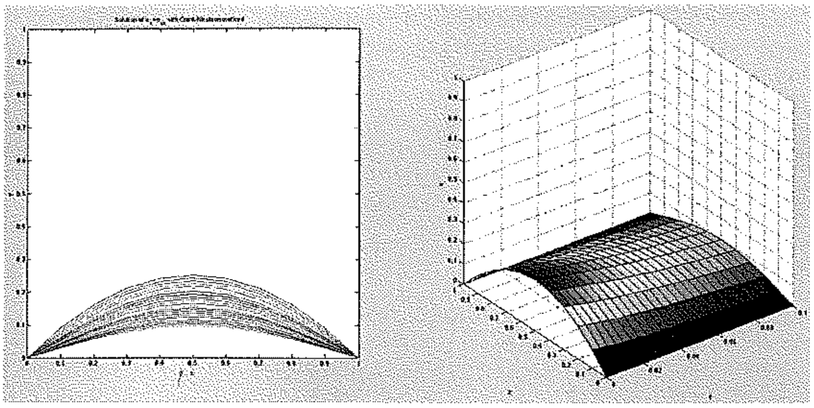


As expected, the method is stable and its solution does not oscillate.

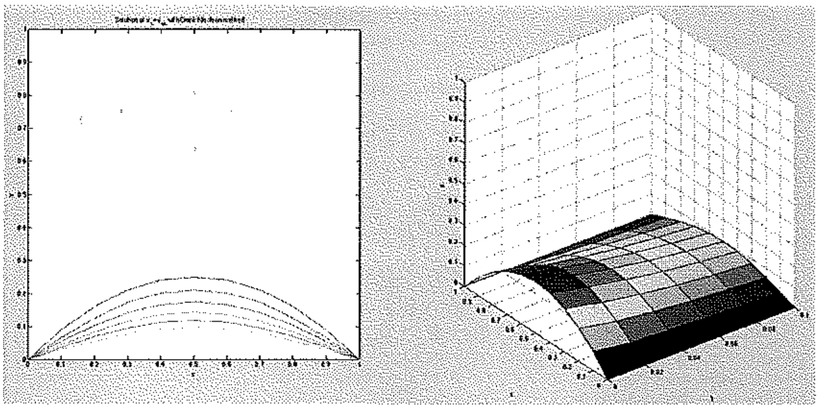
Crank-Nicolson

Since this method is theoretically stable for any value of r and its solution only oscillates when $r > 0.50$, only the graphs of the extreme cases for the values of r (Case A and Case D) are shown below.

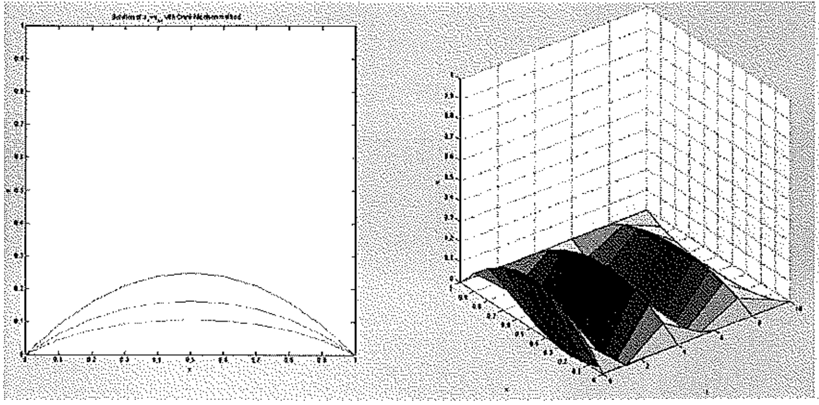
Case A.



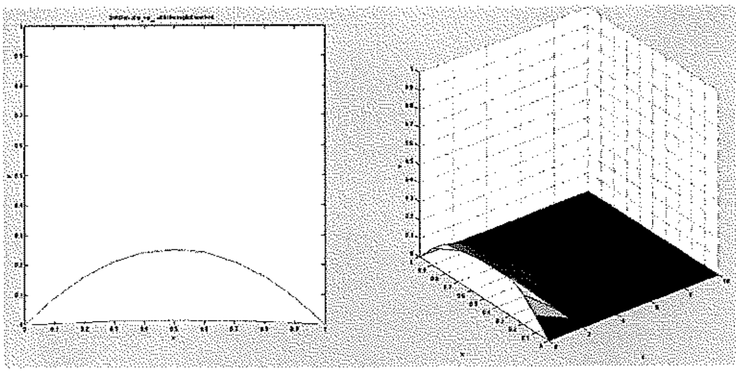
Case D.



As expected, the method is stable, but its solution does not oscillate despite being r bigger than 0.5 because the final time is very small. For example, if in the *Case D* a final time of 10 second is chosen, and the rest of parameters are not changed, the solution of the method oscillates as shown below.



The oscillation is better observed if the *Case D* for a final time of 10 seconds is solved with the BTCS and compared with the solution of C-N.



3. **Check the convergence of the explicit, implicit and Crank-Nicolson methods and discuss the results.**

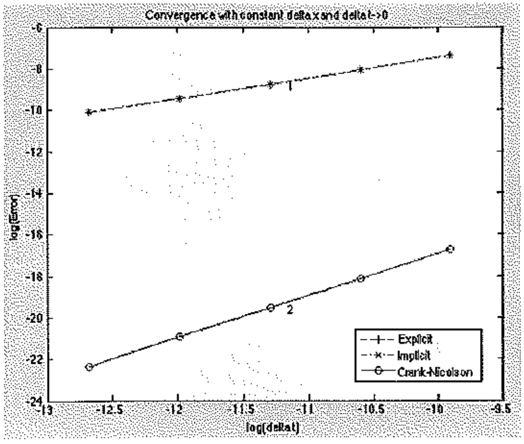
The methods considered have the following theoretical orders of convergence in time and space:

	Order of convergence	
Method	In Time	In Space
FTCS	1	2
BTCS	1	2
Crank-Nicolson	2	2

Since the exact solution is unknown, we assume that a very good approximation of the exact solution would be the one computed with Crank-Nicolson method for a number of points or time steps higher than the maximum value of these quantities used to study the convergence.

Convergence in time

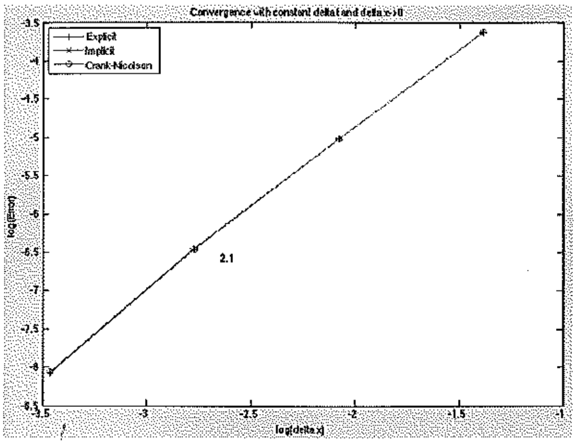
In order to be able to plot the convergence of the methods, the parameter r must be small enough as to guarantee stability for the FTCS. For the interval of study [0, 1], m is chosen to be 100 and nts to be 2000, and therefore r is at most 0.5.



Since the slopes of the lines are 1 for the FTCS (Explicit) and BTCS (Implicit) and 2 for the Crank-Nicolson method, the theoretical convergences in time are achieved.

Convergence in space

To study this convergence, the number of time steps has been chosen big enough as to avoid problems of stability or oscillation while computing the solution for different values of *m*.



Looking at the previous graph, for the three methods a convergence rate in space of 2.1 has been obtained, value that matches well the theoretical value of 2.

Appendix 01. Routine for the FTCS method.

```
% Initialize U matrix
U = zeros(m+1,npast+1);

% IC in U
for i=1:(m+1)
    U(i,1) = f(i);
end

% BC in U
for i=2:(npast+1)
    U(1,i) = g(i);
    U(m+1,i) = h(i);
end

% Computing solution
for j=2:(npast+1)
    for i=2:m
        U(i,j) = r*U(i-1,j-1) + (1-2*r)*U(i,j-1) + r*U(i+1,j-1);
    end
end
```

Appendix 02. Routine for the BTCS method.

```
% Constant theta(O)
O = 1;

% Definition and decomposition of matrix A.
A = zeros(m-1);
A(1,1) = 1 + 2*r*O;
A(1,2) = -r*O;
A(m-1,m-2) = -r*O;
A(m-1,m-1) = 1 + 2*r*O;
if m>3
    for i=2:(m-2)
        A(i,i-1) = -r*O;
        A(i,i) = 1 + 2*r*O;
        A(i,i+1) = -r*O;
    end
end
L = chol(A,'lower');

% Definition of matrix B.
B = zeros(m-1);
B(1,1) = 1 - 2*r*(1 - O);
B(1,2) = r*(1 - O);
B(m-1,m-2) = r*(1 - O);
B(m-1,m-1) = 1 - 2*r*(1 - O);
if m>3
    for i=2:(m-2)
        B(i,i-1) = r*(1-O);
        B(i,i) = 1 - 2*r*(1-O);
        B(i,i+1) = r*(1-O);
    end
end

% Initialize U matrix and F vector
U = zeros(m+1,npast+1);
F = zeros(m-1,1);

% IC in U
for i=1:(m+1)
    U(i,1) = f(i);
end
% BC in U
for i=2:(npast+1)
    U(1,i) = g(i);
    U(m+1,i) = h(i);
end

% Computing solution
for j=2:(npast+1)
    F(1,1) = r*O*U(1,j) + r*(1 - O)*U(1,j-1);
    F(m-1,1) = r*O*U(m+1,j) + r*(1 - O)*U(m+1,j-1);
    for i=2:m
        ujmenos1(i-1,1) = U(i,j-1);
    end
    Y = L\(B*ujmenos1) + L\F;
    ujmas1 = L'\Y;
    for i=2:m
        U(i,j) = ujmas1(i-1);
    end
end
```

Appendix 03. Routine for the Crank-Nicolson method.

```
% Constant theta(O)
O = (1/2);

% Definition and decomposition of matrix A.
A = zeros(m-1);
A(1,1) = 1 + 2*r*O;
A(1,2) = -r*O;
A(m-1,m-2) = -r*O;
A(m-1,m-1) = 1 + 2*r*O;
if m>3
    for i=2:(m-2)
        A(i,i-1) = -r*O;
        A(i,i) = 1 + 2*r*O;
        A(i,i+1) = -r*O;
    end
end
L = chol(A,'lower');

% Definition of matrix B.
B = zeros(m-1);
B(1,1) = 1 - 2*r*(1 - O);
B(1,2) = r*(1 - O);
B(m-1,m-2) = r*(1 - O);
B(m-1,m-1) = 1 - 2*r*(1 - O);
if m>3
    for i=2:(m-2)
        B(i,i-1) = r*(1-O);
        B(i,i) = 1 - 2*r*(1-O);
        B(i,i+1) = r*(1-O);
    end
end

% Initialize U matrix and F vector
U = zeros(m+1,npast+1);
F = zeros(m-1,1);

% IC in U
for i=1:(m+1)
    U(i,1) = f(i);
end
% BC in U
for i=2:(npast+1)
    U(1,i) = g(i);
    U(m+1,i) = h(i);
end

% Computing solution
for j=2:(npast+1)
    F(1,1) = r*O*U(1,j) + r*(1 - O)*U(1,j-1);
    F(m-1,1) = r*O*U(m+1,j) + r*(1 - O)*U(m+1,j-1);
    for i=2:m
        ujmenos1(i-1,1) = U(i,j-1);
    end
    Y = L\(B*ujmenos1) + L\F;
    ujmas1 = L'\Y;
    for i=2:m
        U(i,j) = ujmas1(i-1);
    end
end
```

Appendix 04. Routine for the convergence in time.

```
% Checcks the convergence for the parabolic pde with Dirichlet b.c.
% Order in At

clear all; close all; clc

a = 0;
b = 1;
tfin = 0.1;

% Number of subintervals
m = 100;
Ax = 1/m;
x = [a:Ax:b];
f = sin(x*pi);
g = 0;
h = 0;

% Number of steps for the first computation
npast = 2000;
At = tfin/npast;
t = [0:At:npast*At];

% Reference solution (it is considered as exact)
U   = parabolic_cn(a,b,m,At/(2^6),npast*(2^6),f,g,h);
sol = U(:,npast*(2^6)+1);

npast0 = npast;
At0 = At;

% Computation with different values of At
Ats = [];
errores_ex = [];
errores_im = [];
errores_cn = [];

for i=0:4
    % Explicit
    U_ex = parabolic_ex(a,b,m,At,npast,f,g,h);
    aprox_ex = U_ex(:,npast+1);
    error_ex = norm((sol - aprox_ex));
    errores_ex = [errores_ex,error_ex];
    % Implicit
    U_im = parabolic_im(a,b,m,At,npast,f,g,h);
    aprox_im = U_im(:,npast + 1);
    error_im = norm((sol - aprox_im));
    errores_im = [errores_im,error_im];
    % Crank-Nicolson
    U_cn = parabolic_cn(a,b,m,At,npast,f,g,h);
    aprox_cn = U_cn(:,npast + 1);
    error_cn = norm((sol - aprox_cn));
    errores_cn = [errores_cn,error_cn];
    % Ex. + Im. + Cn.
    Ats = [Ats,At];
    At = At/2;
    npast = npast*2;
end

% Graphical representation
figure(2);clf
plot(log(Ats),log(errores_ex),'--+r','LineWidth',1)
hold on
plot(log(Ats),log(errores_im),'-.xb','LineWidth',1)
hold on
plot(log(Ats),log(errores_cn),'-o')
xlabel('log(delta t)')
ylabel('log(Error)')
title('Convergence with constant delta x and delta t->0')

legend('Explicit','Implicit','Crank-Nicolson')
```

```
% Least squares approximation
p = polyfit(log(Ats),log(errores_ex),1);
x = log(Ats(3)) + 0.125;
y = log(errores_ex(3)) - 0.125;
text(x,y,num2str(p(1),2))

% Least squares approximation
p = polyfit(log(Ats),log(errores_im),1);
x = log(Ats(3)) + 0.125;
y = log(errores_im(3)) - 0.125;
text(x,y,num2str(p(1),2))

% Least squares approximation
p = polyfit(log(Ats),log(errores_cn),1);
x = log(Ats(3)) + 0.125;
y = log(errores_cn(3)) - 0.125;
text(x,y,num2str(p(1),2))

clear all
```

Appendix 05. Routine for the convergence in space.

```
% Checcks the convergence for the parabolic pde with Dirichlet b.c.
% Order in Ax

clear all; close all; clc

a = 0;
b = 1;
tfin = 0.1;

% Number of time steps
npast = 50000;
At = tfin/npast;
t = [0:At:npast*At];

% Number of subintervals for the first computation
m = 4;
Ax = 1/m;

% Reference solution (it is considered as exact)
x = [a:Ax/(2^4):b];
f = sin(x*pi);
g = 0;
h = 0;
U = parabolic_cn(a,b,m*(2^4),At,npast,f,g,h);
sol = U(1:2^4:m*(2^4)+1,npast+1);

% Computation with different values of Ax
Axs = [];
errores_ex = [];
errores_im = [];
errores_cn = [];

for i=0:3
    Ax = 1/m;
    x = [a:Ax:b];
    f = sin(x*pi);
    g = 0;
    h = 0;
    % Explicit
    U_ex = parabolic_ex(a,b,m,At,npast,f,g,h);
    aprox_ex = U_ex(1:2^i:m+1,npast+1);
    error_ex = norm((sol - aprox_ex));
    errores_ex = [errores_ex,error_ex];
    % Implicit
    U_im = parabolic_im(a,b,m,At,npast,f,g,h);
    aprox_im = U_im(1:2^i:m+1,npast+1);
    error_im = norm((sol - aprox_im));
    errores_im = [errores_im,error_im];
    % Crank-Nicolson
    U_cn = parabolic_cn(a,b,m,At,npast,f,g,h);
    aprox_cn = U_cn(1:2^i:m+1,npast+1);
    error_cn = norm((sol - aprox_cn));
    errores_cn = [errores_cn,error_cn];
    Axs = [Axs,Ax];
    m = m*2;
end

% Graphical representation
figure(1),clf
plot(log(Axs),log(errores_ex),'-+',log(Axs),log(errores_im),'-
x',log(Axs),log(errores_cn),'-o')
xlabel('log(delta x)')
ylabel('log(Error)')
title('Convergence with constant delta t and delta x->0')
legend('Explicit','Implicit','Crank-Nicolson',2)

% Least squares approximation. Explicit.
p = polyfit(log(Axs),log(errores_ex),1);
x = log(Axs(3))+ 0.125;
y = log(errores_ex(3)) - 0.125;
text(x,y,num2str(p(1),2))
```

```
% Least squares approximation. Implicit.
p = polyfit(log(Axs),log(errores_im),1);
x = log(Axs(3)) + 0.125;
y = log(errores_im(3)) - 0.125;
text(x,y,num2str(p(1),2))

% Least squares approximation. Crank-Nicolson.
p = polyfit(log(Axs),log(errores_cn),1);
x = log(Axs(3)) + 0.125;
y = log(errores_cn(3)) - 0.125;
text(x,y,num2str(p(1),2))

clear all
```

Homework 2: Finite Differences

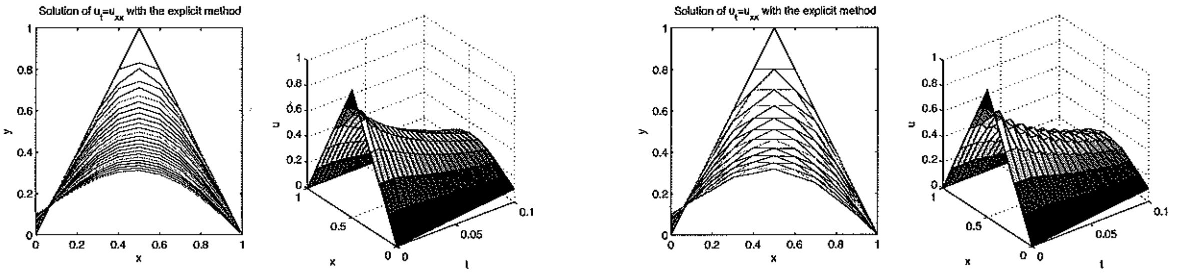
STUDENT 1

(EXCEL·LENT)

1) Results.

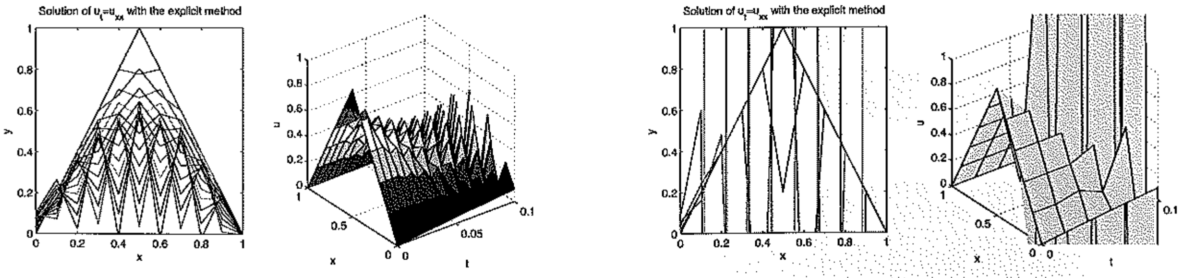
The problem solved is a parabolic 1D problem $\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2}$, with boundary conditions $u(0,t) = t$, $u(1,t) = 0$ and initial condition $u(x,0) = 1 - 2 * (x - 0.5)$.

The problem is solved using backward, forward and Crank-Nicolson schemes for time discretization. For the 3 methods is used $M = 10$ and $t_f = 0.1$.



(a) Steps=24

(b) Steps=20

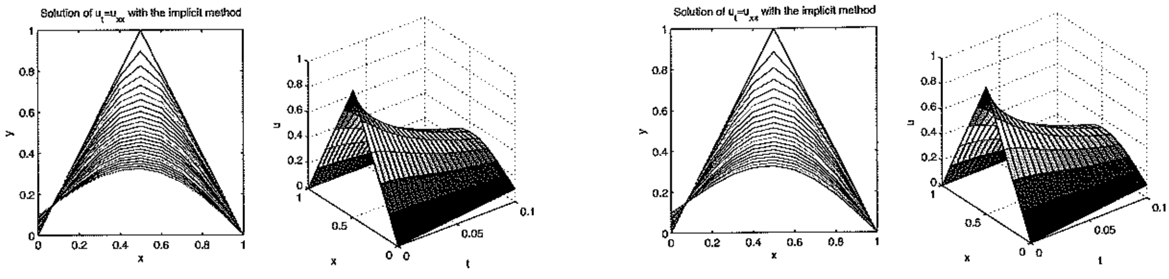


(c) Steps=18

(d) Steps=5

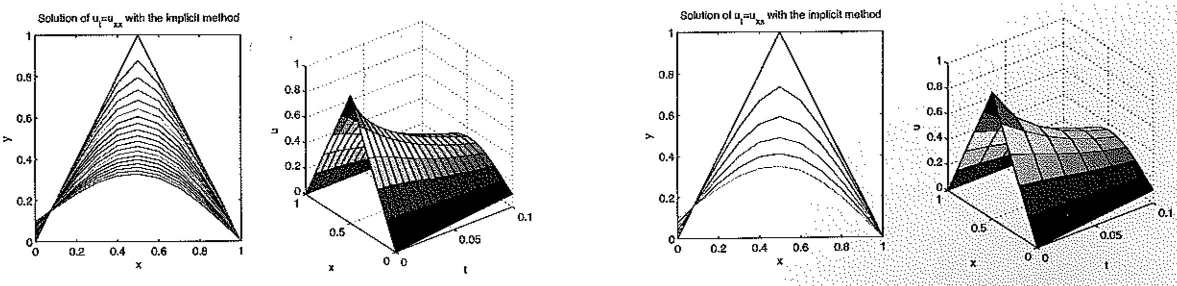
Figure 1: Explicit method

The solution obtained using the explicit method (figure 1) is highly unstable when the condition number $r = \frac{\Delta t}{\Delta x^2}$ is higher than 1/2 (figures 1(c) and 1(d)), and present oscillations when the condition number is higher than 1/4 (figure 1(b)).



(a) Steps=24

(b) Steps=20

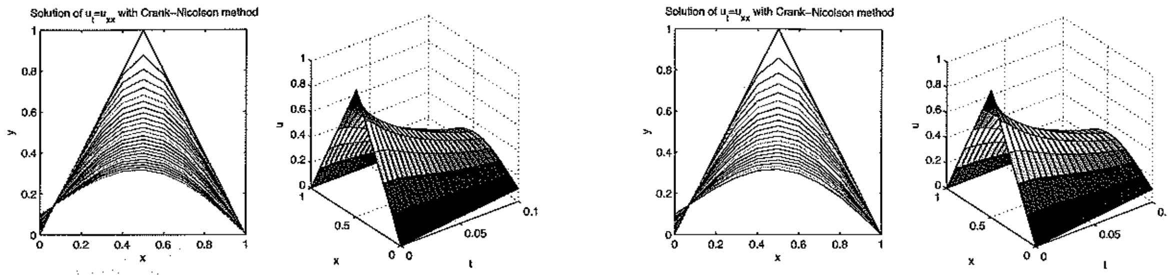


(c) Steps=18

(d) Steps=5

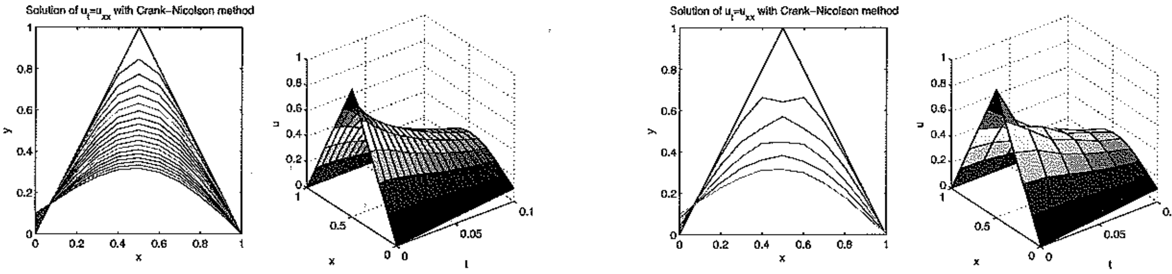
Figure 2: Implicit method

The solution obtained using the implicit method is unconditional stable and without oscillations as expected. The solution obtained with the Crank-Nicolson method is also unconditional stable but present slight oscillations when the condition number is much higher than 1/2 (figure 3(d)).



(a) Steps=24

(b) Steps=20



(c) Steps=18

(d) Steps=5

Figure 3: Crank-Nicolson method

2) Convergence.
The convergence in x for the 3 methods is quadratic as expected, because in the 3 cases a central finite difference method is used for the second derivative.
The convergence in time for the explicit and implicit methods is linear according to the Euler method used; the convergence for the Crank-Nicolson method is second order as expected.

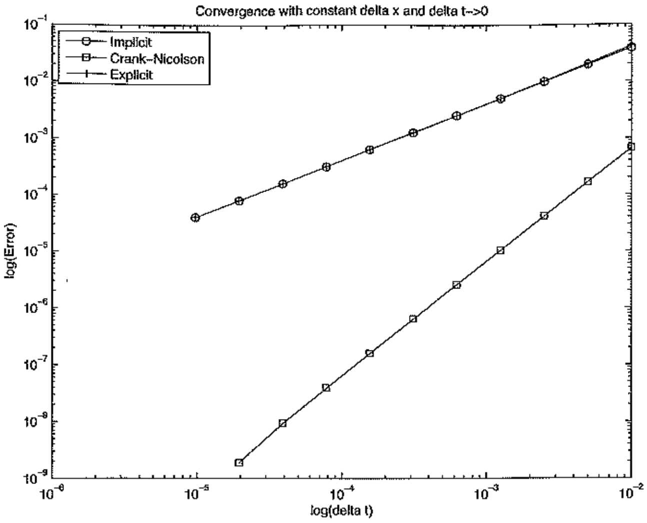


Figure 4: Convergence in time

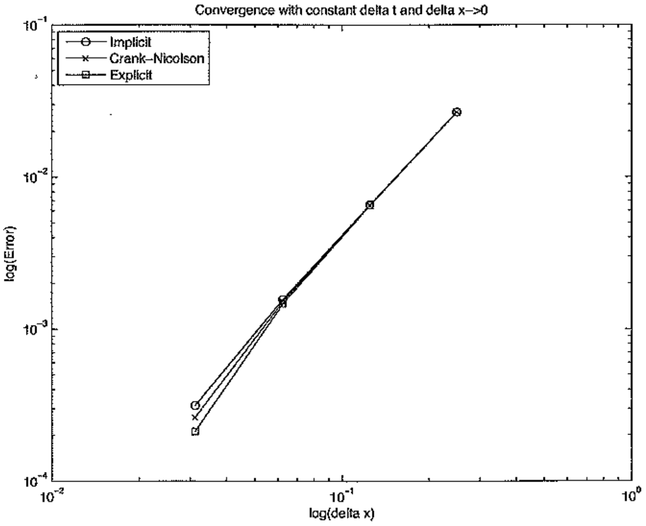


Figure 5: Convergence in space

HOMEWORK 2

STUDENT 2

(EXCEL·LENT)

17/12/13

Numerical methods for PDEs
HOMEWORK 4

The compressed file `FD_Parabolic1D.zip` contains Matlab codes for the finite difference numerical solution of a 1D parabolic equation. Incomplete codes for the explicit (FTCS) and implicit (BTCS) methods are provided. Routines to check the convergence in time and in space are also provided.

1. Complete the coding of the FTCS and BTCS methods.
2. Code the Crank-Nicolson method.
3. Test the three finite difference methods with the following numerical parameters:
   - M=10, final time=0.1, number of steps=24
   - M=10, final time=0.1, number of steps=20
   - M=10, final time=0.1, number of steps=18
   - M=10, final time=0.1, number of steps=5

   Do the finite difference methods behave as expected? Discuss your results.
4. Check the convergence of the explicit, implicit and Crank-Nicolson methods (plot them in the same graph) and discuss the results.

1.

In order to complete the FTCS and BTCS methods, first of all the generation of B and A matrices has been done, respectively. Then the data structure used to store the results is generated. For both methods, the results are stored in a matrix named `U`. The column of this matrix determines the time-step and the row the spatial position. Once we have `U`, boundary and initial conditions are prescribed in it.

Then, after setting all the prescribed information, a loop over all time-steps is performed. For the explicit method, in each time step a matrix times a vector gives us the solution. Meanwhile, for the implicit method a system of equations is solved. In order to compute the solutions, first the system matrix is decomposed using Cholesky method and the two trivial systems of equations are solved. Cholesky has been chosen because we only have Dirichlet boundary conditions and therefore the resulting matrix will be always symmetric positive definite.

2.

For coding the Crank-Nicolson method, the same strategy as in the BTCS method, has been used. The difference between them, lies is the definition of B matrix and A matrix (the last one is the identity for the implicit BTCS). Then, after setting all prescribed information –boundary conditions, initial conditions and the definition of A and B– and also the generation of an empty matrix used to store the solution, a loop over time is performed. In each time step a matrix times a vector is done, but also a system of equations is solved. This system of equations is solved also using Cholesky, as what is done for the implicit case.

3.

All three methods has been tested by using the 5 proposed equations, but at this point only remarkable results are presented. The equation chosen for doing the analysis is the fifth, because it contains a time dependent Dirichlet condition, and if the method works well with this kind of boundary conditions it's also expected to do so with constant or null boundary conditions.

For the explicit method, using 5 or 18 time steps the solution is not converging, this is completely what it's expected, because $r > 1/2$. Then, using 20 or 24 time-steps the solution is converging.

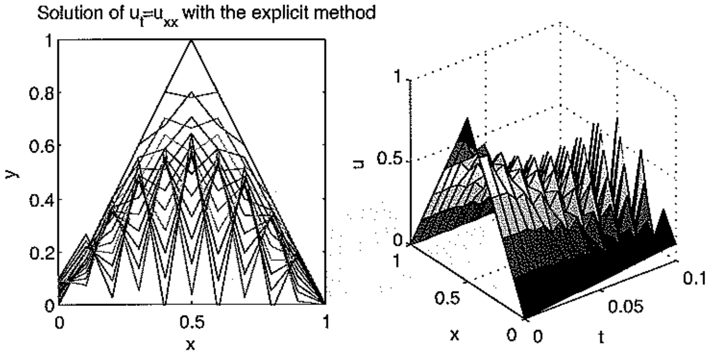


Figure 1: Result with explicit method performing 18 time-steps ($r = 0.5556$).

As it was said before, on Figure 1 we can see that the method is not converging when we are using only 18 time-steps. The implicit method, converges for any number of time-steps, although the solution is more accurate as we increase the number of time-steps used. Here the solution using 5 time-steps is presented.

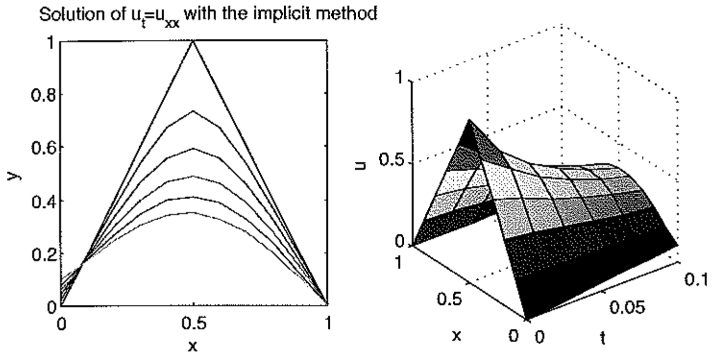


Figure 2: Result with implicit method performing 5 time-steps.

Figure 2 shows that even using a large time-step the solution is converging. Finally the Crank-Nicolson methods also converged for any number of time-steps. However when $r > 1/2$ the solution obtained presents oscillations. This behavior is also expected and characteristic of this method.

STUDENT 2

(EXCEL·LENT)

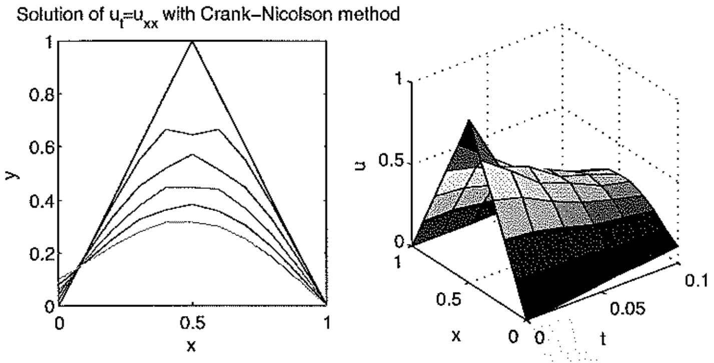


Figure 3: Result with Crank-Nicolson method performing 5 time-steps ($r = 2$).

As it was said before, when we apply Crank-Nicolson method with $r = 2$, we expect to have an oscillating, but converging, solution. This behavior is observed clearly in Figure 3.

4.

The convergence analysis depends on the time discretization, but also on the spatial discretization. Therefore this analysis must be divided in two parts. On one hand, let's focus in the convergence in time. For all three methods, a small Δx has been kept fix in order to see how affects to the error the reduction of Δt, the resulting plot is:

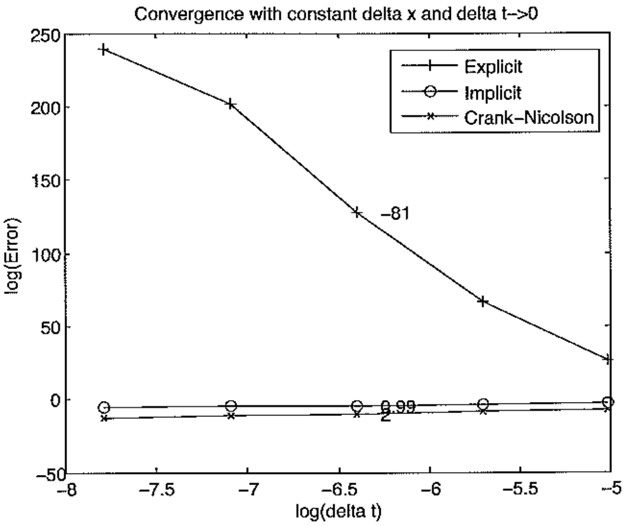


Figure 4: Convergence rate keeping Δx and reducing Δt, for explicit, implicit and Crank-Nicolson methods.

Notice that explicit method is not working well, because a very small spatial discretization has been chosen in order to minimize the error dependence on it. Therefore r values are all over 1/2, thus the method is not converging. In order to reduce all r values, instead of performing the analysis using 15, 30, 60, 120 and 240 time-steps, 500, 1000, 2000, 4000 and 8000 have been used.

Therefore instead of having r's between 16.67 and 1.042, we deal with r's between 0.5 and 0.03125. Then the resulting plot is:

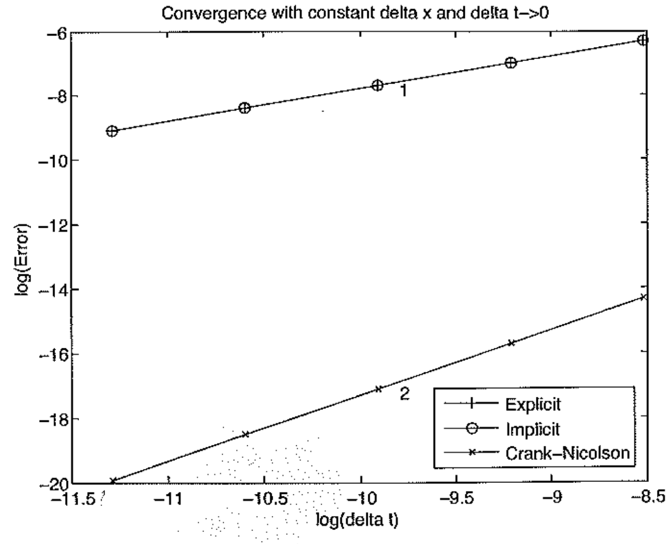


Figure 5: Convergence rate keeping Δx and reducing Δt, for explicit, implicit and Crank-Nicolson methods. (Using small Δt for the analysis).

As it can be appreciated, the slopes of the obtained straight lines are 2 and 1. This is what we were expecting, because for the FTCS and BTCS methods the error depends linearly with the time-step and for the Crank-Nicolson it depends with the square of the time-step. On the other hand, the convergence in space for all three methods is:

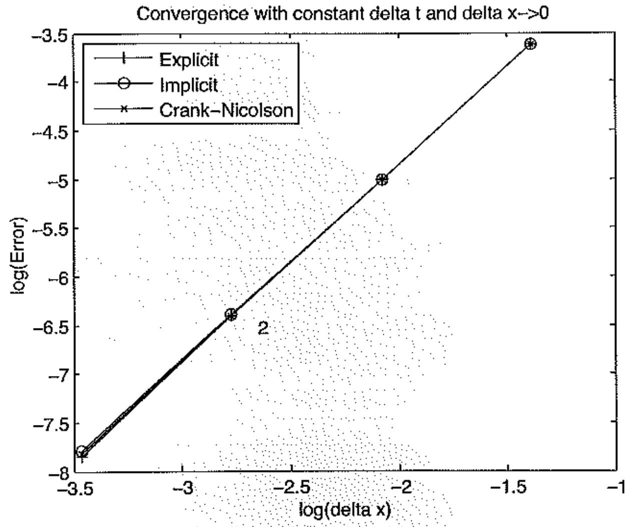


Figure 6: Convergence rate keeping Δt and reducing Δx, for explicit, implicit and Crank-Nicolson methods.

As it was expected the slope of all three straight lines is 2. For all three methods, the spatial discretization is the same and it depends on the square of the time-step.

In order to obtain the results showed above, some changes has been made to the given codes. In the time convergence in time analysis, instead of computing the reference solution with implicit method, Crank-Nicoloson has been used. Although, the same results has been obtained with implicit method and a really small Δt. In the spatial convergence analysis, the given number of time-steps has been increased, because some effect of time discretization was polluting the result for the smaller Δx used.

UNIVERSITAT POLITÈCNICA DE CATALUNYA
UPC BarcelonaTech

SEMESTER 1 EXAMINATIONS
JANUARY 2014

NUMERICAL METHODS
FOR PARTIAL DIFFERENTIAL EQUATIONS

UNIVERSITY CALCULATORS ONLY

Translation dictionaries are not permitted, but an English dictionary may be borrowed from the invigilator on request.

Time allowed: **2 hours** **CLOSED BOOK**

Answer **THREE** questions

Please insert any standard constants here.

TURN OVER

Question 1 [25 marks]

(a) If n point Gaussian quadrature is used for numerical integration state the order of the polynomial that is integrated exactly. *[2 MARKS]*

(b) State which if any, of the following integrals can be integrated exactly using Gaussian quadrature and give the optimal order of the rule in each case.

(i) $\int_0^1 (7x^{10} + 9x^9)dx$, (ii) $\int_0^1 \cosh x \, dx$, (iii) $\int_0^1 x^{14} dx$, (iv) $\int_0^1 x^{15} dx$ *[4 MARKS]*

(c) Use 2-point Gaussian quadrature with weights $w_1 = w_2 = 1$ and Gauss points $\xi_1 = -1/\sqrt{3},\ \xi_2 = +1/\sqrt{3}$ to perform numerical integration of $\int_0^\pi \left(3 - 8(\cos x)^2\right)dx$.

[4 MARKS]

(d) The second order ordinary differential equation

$$\frac{d^2Y}{dt^2} + \frac{dY}{dt} + 4Y^4 = 0$$

is defined over the domain $0 \le t \le 1$, and is to be solved numerically subject to the initial conditions $Y(0) = 0,\ dY(0)/\mathrm{dt} = 1$, where $Y(t)$ is the exact solution.

(d1) Reduce the above second order ordinary differential equation to a system of first order ordinary differential equations. [4 *MARKS*]

(d2) State the backward Euler method for integrating the system of first order ordinary differential equations in (d1) above and hence express the non-linear system of equations that must be solved to take the first time step in terms of time step size Δt. [5 *MARKS*]

(d3) State the advantages versus disadvantages of explicit and implicit methods for solving ordinary and partial differential equations. *[6 MARKS]*

TURN OVER

Question 2 [25 marks]

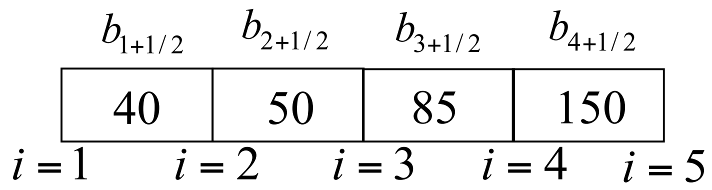
The differential equation

$$\frac{d}{dx}(aU)-\frac{d}{dx}\left(b\frac{dU}{dx}\right)=0$$

is defined over the domain $0 \le x \le 1$ where $U(x)$ is the exact conservation variable, and $a(x)$, $b(x)$ are the respective wave speed and diffusion coefficients.

The equation is to be solved numerically on a uniform grid with three interior nodes and four equally spaced intervals of size h subject to the boundary conditions $U(0) = 3,\ U(1) = 15$.

Given that the wave speed $a(x)$ is constant with $a_{i+1/2} = 60$ over each grid interval $i = 1,2,3,4$ and diffusion coefficient $b(x)$ has piecewise constant variation over the grid intervals (indicated below) where $b_{1+1/2} = 40,\ b_{2+1/2} = 50,\ b_{3+1/2} = 85,\ b_{4+1/2} = 150$,



(a) use the finite-volume approximation $\dfrac{f_{i+1/2}-f_{i-1/2}}{h}=0$, where

$$f_{i+1/2}=a_{i+1/2}\frac{(u_i+u_{i+1})}{2}-b_{i+1/2}\left(\frac{u_{i+1}-u_i}{h}\right),$$

to show that the resulting matrix system takes the form

$$\begin{pmatrix} 1440 & -680 & 0 \\ -920 & 2160 & -1240 \\ 0 & -1480 & 3760 \end{pmatrix}\begin{pmatrix} u_2 \\ u_3 \\ u_4 \end{pmatrix}=\begin{pmatrix} 2280 \\ 0 \\ 34200 \end{pmatrix}$$

[9 MARKS]

(b) Write down the point Jacobi method for solving the linear system $A\mathbf{x}=\mathbf{b}$ for a general matrix A and perform 2 iterations on the above matrix system. *[7 MARKS]*

(c) Write down the point Gauss-Seidel method for solving $A\mathbf{x}=\mathbf{b}$ and perform 2 iterations on the above matrix system. *[7 MARKS]*

(d) Explain any advantage of one method over the other. *[2 MARKS]*

TURN OVER

Question 3 [25 marks]

The one dimensional convection equation is defined over the domain $0 \le x \le 1$ with *constant* wave speed a and is written as

$$U_t + f(U)_x = 0$$

where $f(U) = aU$. The equation is to be solved using an upwind scheme written as

$$u_i^{n+1} = u_i^n - \frac{\Delta t}{h}(f_{i+1/2}^n - f_{i-1/2}^n)$$

where $f_{i+1/2}^n$ denotes the discrete flux at time level n on cell face $i+1/2$, h is the grid interval size, Δt the time step and

$$f_{i+1/2} = \begin{cases} au_i & a \ge 0 \\ au_{i+1} & a < 0 \end{cases} .$$

(a) Is the above method locally conservative? Explain *[4 MARKS]*

(b) Write down the explicit scheme and deduce the stability conditions which ensure that the method is positive for
i) a positive wave speed $a > 0$,
ii) a negative wave speed $a < 0$.
(*Hint:* an explicit method of the form $u_i^{n+1} = \alpha_{-l}u_{i-l}^n + \ldots + \alpha_0 u_i^n + \ldots + \alpha_r u_{i+r}^n$ is called positive and is stable if all coefficients are positive and sum to unity, that is $\alpha_{-l},\ldots,\alpha_r > 0$ and $\alpha_{-l} + \ldots + \alpha_0 + \ldots + \alpha_r = 1$) *[6 MARKS]*

(c) Use the above explicit scheme to compute the solution after one time step for $a = 10$ on a one-dimensional uniform grid with three interior nodes and four equally spaced intervals, with boundary condition $U(0) = 1$, initial conditions $u_2^0 = 0.1$, $u_3^0 = 0.1$, $u_4^0 = 0.1$, $u_5^0 = 0.1$ and $\Delta t = 0.25$. Is the result consistent with your stability analysis? *[4 MARKS]*

Question 3 continued over leaf

TURN OVER

Question 3 continued

The implicit upwind finite volume scheme is written as

$$u_i^{n+1} = u_i^n - \frac{\Delta t}{h}(f_{i+1/2}^{n+1} - f_{i-1/2}^{n+1})$$

where the upwind flux given above is now defined implicitly at time level $n+1$.

(d) Write down the fully implicit scheme and formulate the method to compute the solution after one time step for the same problem as posed in (c) above, again using the same one-dimensional uniform grid with four equally spaced intervals with boundary condition $U(0) = 1$, initial conditions $u_2^0 = 0.1$, $u_3^0 = 0.1$, $u_4^0 = 0.1$, $u_5^0 = 0.1$ and $\Delta t = 0.25$.

[7 MARKS]

(e) Derive the leading truncation error of the implicit method and state if the method is consistent. *[4 MARKS]*

Question 4 [25 marks]

4 : The partial differential equation

$$\frac{\partial U}{\partial t} = b\frac{\partial^2 U}{\partial x^2}$$

is defined over the domain $0 \le x \le 1$ where b is constant, and is to be solved numerically subject to the boundary conditions $U(0,t) = 1.0$, $U(1,t) = 1.0$ and initial condition $U(x,0) = U_0(x)$ (defined below), where $U(x,t)$ is the exact solution.

The explicit forward time centred space scheme is expressed in the form

$$u_i^{n+1} - u_i^n = \mu(u_{i+1}^n - 2u_i^n + u_{i-1}^n)$$

where $\mu = \frac{\Delta t b}{h^2}$, h is the grid interval size, Δt the time step, suffix i is a spatial index (x-direction) and superfix n is the time level.

(a) Compute the solution after one time step using the above method with $b = 3$ and $\Delta t = 0.25$ on a uniform grid with four equally spaced intervals together with the above boundary conditions and initial data $u_2^0 = 4.0$, $u_3^0 = 8.0$, $u_4^0 = 4.0$ defined at the three interior nodes. *[3 MARKS]*

(b) Determine the explicit stability condition for the method (with positive coefficients) and use the condition to determine if the above calculation is stable. *[4 MARKS]*

The implicit backward time centred space scheme is expressed in the form

$$u_i^{n+1} - u_i^n = \mu(u_{i+1}^{n+1} - 2u_i^{n+1} + u_{i-1}^{n+1})$$

(b) Calculate the leading truncation error of the implicit scheme. *[4 MARKS]*

(c) Write down the resulting system of implicit equations expressed in terms of μ , that must be solved in order to compute the solution at the 3 interior nodes of the above grid after one time step. *[7 MARKS]*

(d) Using the above boundary conditions and initial data with $b = 3$ and $\Delta t = 0.25$ compute the solution at the 3 interior nodes after one time step via Gaussian elimination and comment on stability. *[7 MARKS]*

Hints:
Gaussian Elimination involves reducing a matrix to upper triangular form by row operations. The solution is obtained by back substitution.

END OF PAPER

FINAL EXAM

STUDENT 1

(APROVAT)

Question 1

a) Polynomia up to order

1 $\quad 2n+1$ $\quad$! if n+1 points are used

b) $\int_0^1 (7x^{10}+9x^9)\,dx$ $\quad$ can be solved exactly with

2.5 $\quad 10 < 2n+1 \Rightarrow n = \overset{6}{5}$

$\int_0^1 \cosh x \, dx$ $\quad$ can not be solved exactly (not a polynomial) ✓

$\int_0^1 x^{14}\, dx$ $\quad$ can be solved exactly

$14 < 2n+1 \Rightarrow n = 7 \quad 8$

$\int_0^1 x^{15}\, dx$ $\quad$ can be solved exactly

$15 < 2n+1 \Rightarrow n = 7 \quad 8$

c) $\int_0^\pi (13 - 8(\cos x)^2\, dx$

3.5 Transformation to interval $[-1,1]$

$$x = \frac{b-a}{2} z + \frac{b+a}{2} = \frac{\pi - 0}{2} z + \frac{\pi + 0}{2} = \frac{\pi}{2} z + \frac{\pi}{2}$$

✓

$$dx = \frac{b-a}{2} dz = \frac{\pi - 0}{2} = \frac{\pi}{2} dz$$

$$\int_0^\pi (13 - 8(\cos x)^2\, dx = \frac{\pi}{2} \int_{-1}^{1} (13 - 8(\cos \frac{\pi}{2} z + \frac{\pi}{2})^2 dz$$

$$\Downarrow$$

$$\sum w \cdot f(z) = \frac{\pi}{2} \left(1 \cdot (13 - 8(\cos(\frac{\pi}{2} \cdot -\frac{1}{\sqrt{3}} + \frac{\pi}{2}))^2 + (1 \cdot (13 - 8(\cos(\frac{\pi}{2} \cdot \frac{1}{\sqrt{3}} + \frac{\pi}{2}))^2\right)$$

d) $\frac{d^2Y}{dt} + \frac{dY}{dt} + 4Y^4 = 0$, $Y(0) = 0$, $\frac{dY(0)}{dt} = 1$

d1) $Y_{(1)} = Y$ $\qquad Y_{(1)}' = \frac{d^2Y}{dt} + 4Y^4 = Y_{(2)}' + 4Y_{(1)}^4 = Y_{(1)} + 8Y_{(1)}^4$?

2 $\quad Y_{(2)} = Y'$ $\qquad Y_{(2)}' = \frac{dY}{dt} + 4Y_{(1)}^4 = Y_{(2)} + 4Y_{(1)}^4$

$$\vec{Y}^n = \begin{pmatrix} Y_{(1)}^n \\ Y_{(2)}^n \end{pmatrix}$$

$$\underbrace{\vec{Y}'^n}_{\vec{Y}'} = \begin{pmatrix} Y_{(2)} + \cancel{8Y_{(1)}^4} \\ \cancel{-}Y_{(2)} \mp 4Y_{(1)}^4 \end{pmatrix} \qquad \underline{\text{no}}$$

Boundary conditions

$$\vec{Y}(0) = \begin{matrix} Y_{(1)}(0) \\ Y_{(2)}(0) \end{matrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \quad ✓$$

d2) Backward Euler ? $\vec{Y}$

$$\vec{Y}^{n+1} = \vec{Y}^n + \Delta t \, \boxed{\vec{Y}'}^{n+1}$$

2 $$\Downarrow \begin{pmatrix} Y_{(1)}^{n+1} \\ Y_{(2)}^{n+1} \end{pmatrix} = \begin{pmatrix} Y_{(1)}^n \\ Y_{(2)}^n \end{pmatrix} + \Delta t \begin{pmatrix} Y_2^{(n+1)} + \cancel{8Y_{(1)}^{n+1\,4}} \\ \cancel{-}Y_2^{(n+1)} - 4Y_{(1)}^{n+1\,4} \end{pmatrix}$$

$$\Downarrow \begin{pmatrix} Y_{(1)}^{n+1} \\ Y_{(2)}^{n+1} \end{pmatrix} - \Delta t \underbrace{\begin{pmatrix} Y_2^{(n+1)} + 8Y_{(1)}^{n+1\,4} \\ Y_{(2)}^{(n+1)} + 4Y_{(1)}^{n+1\,4} \end{pmatrix}} = \begin{pmatrix} Y_{(1)}^n \\ Y_{(2)}^n \end{pmatrix}$$

$$i=1 \qquad \begin{pmatrix} Y_{(1)}^1 \\ Y_{(2)}^1 \end{pmatrix} - \Delta t \underbrace{\begin{pmatrix} Y_{(2)}^1 + 8Y_{(1)}^{1\,4} \\ Y_{(2)}^1 + 4Y_{(1)}^{1\,4} \end{pmatrix}} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

d3) Ordinary differential equations

Explicit $\quad$ conditionally stable (stability condition $0 < h < \frac{\lambda}{2}$)

$\qquad$ Error of order $O(h)$ depends on the method

Implicit $\quad$ stable

$\qquad$ Error of order $O(h)$ same here

9

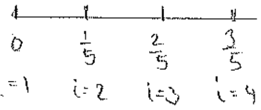
Partial differential equations

Explicit conditionally stable (stability condition $r \leq \frac{1}{2}$)
Error of order $(\Delta x, \Delta y^2$ (it FSCT $(\Delta x, \Delta t^2)$

Implicit Unconditionally stable
Error of order $(\Delta x, \Delta y)$ (it BSCT $(\Delta x, \Delta t^2)$ 3

Question 2

a) $$\frac{f_{i+1/2} - f_{i-1/2}}{h} = 0 \quad , \quad h = \frac{1}{3}$$



$$\Downarrow$$

$$0 = \frac{1}{h}\left(\left(a_{i+1/2}\frac{(U_i + U_{i+1})}{2} - b_{i+1/2}\frac{U_{i+1} - U_i}{h}\right) - \left(a_{i-1/2}\frac{U_{i-1} + U_i}{2} - b_{i-1/2}\frac{U_i - U_{i-1}}{h}\right)\right)$$

$$0 = \frac{a_{i+1/2} \cdot (U_i + U_{i+1})}{2h} - b_{i+1/2} \cdot (U_{i+1} - U_i) - \frac{a_{i-1/2}(U_{i-1} + U_i)}{2h} - b_{i-1/2} \cdot (U_i - U_{i-1})$$

i=1 3

b)

$$\begin{pmatrix} 1440 & -680 & 0 \\ -920 & 2160 & -1240 \\ 0 & -1480 & 3760 \end{pmatrix} \cdot \begin{pmatrix} U_2 \\ U_3 \\ U_4 \end{pmatrix} = \begin{pmatrix} 2280 \\ 0 \\ 34200 \end{pmatrix} \qquad x^0 = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

X^1

$$1440\, x_1^1 - 680\, x_1^0 = 2280 \Rightarrow X_1^1 = \frac{2280}{1440} = 1{,}58$$

$$-920\, X_1^0 + 2160\, X_2^1 - 1240\, X_3^0 = 0 \Rightarrow X_2^1 = 0$$

$$-1480 \times x_2^0 - 3760 x_3^0 = 34200 \Rightarrow X_3^1 = \frac{34200}{3760} = 9{,}10$$

2

$$X^1 = \begin{pmatrix} 1{,}58 \\ 0 \\ 27{,}58 \end{pmatrix}$$

X_2

$$1440 \cdot X_1^2 - 680\, X_2^1 = 2280 \quad X_1^2 = \frac{2280}{1440} = 1{,}58$$

$$-920\, X_1^1 + 2160\, X_2^2 - 1240 x_3^1 = 0 \Rightarrow X_2^2 = \frac{920 \cdot 1{,}58 + 1240 \cdot 9{,}10}{2160} = 5{,}89$$

$$-1480\, X_2^1 - 3760\, X_3^2 = 34200 \Rightarrow X_3^2 = \frac{34200}{3760} = 9{,}10$$

$$X^2 = \begin{pmatrix} 1{,}58 \\ 5{,}89 \\ 9{,}10 \end{pmatrix}$$

2

c)

$$\begin{pmatrix} 1440 & -680 & 0 \\ -920 & 2160 & -1240 \\ 0 & -1480 & 3760 \end{pmatrix} \cdot \begin{pmatrix} U_2 \\ U_3 \\ U_4 \end{pmatrix} = \begin{pmatrix} 2280 \\ 0 \\ 34200 \end{pmatrix}, \quad X^0 = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

X^1

$$1440 \cdot x_1^1 - 680 x_2^0 = 2280 \Rightarrow x_1^1 = \frac{2280}{1440} = 1{,}58$$

$$-920 x_1^1 + 2160 x_2^1 - 1240 x_3^0 = 0 \Rightarrow x_2^1 = \frac{920 \cdot 1{,}58}{2160} = 0{,}67$$

$$-1480 x_2^1 + 3760 x_3^1 = 34200 \Rightarrow x_3^1 = \frac{34200 + 1480 \cdot 0{,}67}{3760} = 9{,}35$$

$$X^1 = \begin{pmatrix} 1{,}58 \\ 0{,}67 \\ 9{,}35 \end{pmatrix}$$

2

X^2

$$1440 \cdot x_1^2 - 680 \cdot x_2^1 = 2280 \Rightarrow x_1^2 = \frac{2280 + (680 \cdot 0{,}67)}{1440} = 1{,}90$$

$$-920 \cdot x_1^2 + 2160 x_2^2 - 1240 x_3^1 = 0 \Rightarrow x_2^2 = \frac{(920 \cdot 1{,}90 + 1240 \cdot 9{,}35)}{2160} = 6{,}17$$

$$-1480 x_2^2 + 3760 x_3^2 = 3420 \Rightarrow x_3^2 = \frac{3420 + 1480 \cdot 6{,}17}{3760} = 3{,}34$$

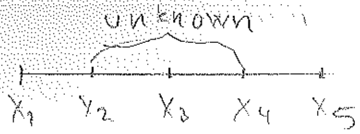
2

$$X^2 = \begin{pmatrix} 1{,}90 \\ 6{,}17 \\ 3{,}34 \end{pmatrix}$$

d) Gauss Seidel is faster than Jacobi, because previous iterations are used ?

1.5

Question 4



$h = \frac{1}{4} \quad \mu = \frac{\Delta t \cdot b}{h^2} = \frac{0{,}25 \cdot 3}{(\frac{1}{4})^2} = 12$

$i=2 \quad U_2^1 = \mu(U_3^0 - 2U_2^0 + U_1^0) \overset{+u_i^0}{=} 12 \cdot (8 - 2 \cdot 4 + 1) = 12$

$i=3 \quad U_3^1 = \mu(U_4^0 - 2U_3^0 + U_2^1) \overset{+u_3^0}{=} 12 \cdot (4 - 2 \cdot 8 + 4) = -96$

$i=4 \quad U_4^1 = \mu(u_5^0 - 2U_4^0 + U_3^0) \overset{+u_4^0}{=} 12 \cdot (1 - 2 \cdot 4 + 8) = 12$

b) stability condition

$\mu < \frac{1}{2} \Rightarrow$ a) is unstable

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

$\Downarrow$

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

$i=2 \quad U_2^1(1+2\mu) - \mu U_3^1 - \mu U_1^1 = U_2^0 \Rightarrow U_2^1(1-2\mu) \cdot \mu U_3^1 = \mu$

$i=3 \quad U_3^1(1+2\mu) - \mu U_4^1 - \mu U_2^1 = U_3^0 = 0$

$i=4 \quad U_4^1(1+2\mu) - \mu U_5^1 - \mu U_3^1 = U_4^0 \Rightarrow U_4^1(1+2\mu) \cdot \mu U_3^1 = \mu$

$$\begin{pmatrix} 1+2\mu & -\mu & 0 \\ -\mu & 1+2\mu & -\mu \\ 0 & -\mu & 1+2\mu \end{pmatrix} \cdot \begin{pmatrix} U_3^1 \\ U_4^1 \\ U_5^1 \end{pmatrix} = \boxed{\begin{pmatrix} \mu \\ 0 \\ \mu \end{pmatrix}}$$

?.

Gauss elimination $\mu = 12$

1st row

$$\left(-12 \quad 1+2\cdot 12 \quad -12 \mid 0\right) - \frac{-12}{1+2\cdot 12} \cdot \left(1+2\cdot 12 \quad -12 \quad 0 \mid 12\right)$$

$$0 \quad 144 \quad -12 \mid 5{,}76$$

$$\begin{pmatrix} 25 & -12 & 0 \\ 0 & 144 & -12 \\ 0 & -12 & 25 \end{pmatrix} \cdot \begin{pmatrix} U_3' \\ U_4' \\ U_5' \end{pmatrix} = \begin{pmatrix} 12 \\ 5{,}76 \\ 0 \end{pmatrix}$$

2nd row

$$\left(0 \; -12 \quad 25 \mid 0\right) - \frac{-12}{144} \cdot \left(0 \quad 144 \quad -12 \mid 5{,}76\right)$$

$$\text{after} \overset{\text{elimination}}{\vee} \text{backward}$$

$-0{,}48$

QUESTION 1

FINAL EXAM

a) $n=0 \Rightarrow$ 1 point , 1 weights $\Rightarrow$ 1st order polynomial

2

$n=1 \Rightarrow$ 2 points , 2 weights $\Rightarrow$ 3rd order polynomial

that is ...

$n=2 \Rightarrow$ 3 points , 3 weights $\Rightarrow$ 5th order polynomial

STUDENT 2

(APROVAT)

b) $\int_0^1 (7x^{10} + 9x^9)\,dx$

$n=5 \Rightarrow$ 6 points , 6 weights $\Rightarrow$ 11th order

$\int_0^1 \cosh x \, dx$

can not be integrated - not a polynomial

4

$\int x^{14} \, dx$

$n=7 \Rightarrow$ 8 points , 8 weights $\Rightarrow$ 15th order

$\int_0^1 x^{15} \, dx$

$n=7 \Rightarrow$ 8 points , 8 weights $\Rightarrow$ 15th order

c) $\int_0^{\pi} (13 - 8\cos(x)^2)\,dx \qquad w_1 = w_2 = 1 \qquad \delta_1 = -1/\sqrt{3} \qquad \delta_2 = 1/\sqrt{3}$

3

Transform from $[-1,1]$ to $[0,\pi]$

$$z_1 = \frac{b-a}{2}\delta_1 + \frac{a+b}{2}$$

$$\Downarrow$$

$$z_1 = \frac{\pi-0}{2} \cdot \left(-1/\sqrt{3}\right) + \frac{\pi+0}{2} = 0{,}664$$

$$z_2 = \frac{b-a}{2}\delta_2 + \frac{a+b}{2}$$

$$z_1 = \frac{\pi-0}{2}\left(1/\sqrt{3}\right) + \frac{\pi+0}{2} = 2{,}478$$

9 pages

• $\boxed{I = \frac{b-a}{2} \sum w_i f(x_i)}$ formula to use

$\Downarrow$

$$I = \frac{\pi}{2}\left(1 \cdot (13 - 8\cos(0{,}664)^2 + 1\left(13 - 8\cos(2{,}478)^2\right)\right)$$

$\Downarrow$

$\underline{\underline{I = 15{,}733}}$ N

d) $\frac{d^2Y}{dt^2} + \frac{dY}{dt} + 4Y^4 = 0$ $\quad 0 \le t \le 1 \quad Y(0) = 0 \quad \frac{dY}{dt}(0) = 1$

d1) New functions defined

$Y_1 = Y$

$Y_2 = Y'$

∴ differentiate now those new functions

$Y_1' = Y' = Y_2$

$Y_2' = Y'' \Rightarrow Y_2' = -Y_2 - 4Y_1^4$

1st order ODEs

$Y_1' = Y_2$ with IC $Y_1(0) = 0$

$Y_2' = -Y_2 - 4Y_1^4$ with IC $Y_2(0) = 1$ ✓

4

d2) Backward euler

$Y_{i+1} = Y_i + \Delta t\, f(x_{i+1}, Y_{i+1})$

1

d3) Explicit schemes

① Forward euler $Y_{i+1} = Y_i + h f(x_i, Y_i)$

- Low computational cost
- conditionally stable $0 < h < 2/-\lambda$

5

② Central approximation $Y_{i+1} = Y_{i-1} + 2h f(x_i, Y_i)$

- Low computational cost
- Has to solve Forward euler first and then the central approximation

③ FTCS

- Low computational cost
- conditionally stable $r \le 1/2$
- Non osscilations $r \le 1/4$

, where $r = \frac{\Delta t}{\Delta x^2}$ (depending on the PDE that has to be solved)

Implicit schemes

① Backward euler $Y_{i+1} = Y_i + h f(x_{i+1}, Y_{i+1})$

- Has to solve a system to find the new values
- Unconditionally stable

② BTCS

- Has to solve a system to find the new values
- Unconditionally stable

③ Crank-Nicolson

- Has to solve system
- unconditionally stable

Question 2

$$\frac{d}{dx} aU - \frac{d}{dx}\left(b\frac{dU}{dx}\right) = 0 \qquad 0 \le x \le 1$$

$U(0) = 3$, $U(1) = 15$

$a_{i+1/2} = 60$ constant

$b_{1+1/2} = 40$ $b_{2+1/2} = 50$ $b_{3+1/2} = 85$ $b_{4+1/2} = 150$

a)

$$f_{i+1/2} = 60 \cdot \frac{(U_i + U_{i+1})}{2} - b_{i+1/2}\left(\frac{U_{i+1} - U_i}{h}\right)$$

$h = 1$

1 equation

$f_{i+1/2} = 60($

$$\begin{pmatrix} 1440 & -680 & 0 \\ -920 & 2160 & -1240 \\ 0 & -1480 & 3760 \end{pmatrix} \begin{pmatrix} U_2 \\ U_3 \\ U_4 \end{pmatrix} = \begin{pmatrix} 2280 \\ 0 \\ 34200 \end{pmatrix} \qquad x^0 = 0$$

0

1st iteration

1st eq

$$1440\, U_2 - 680\, U_3 = 2280 \rightarrow 1440 U_2^1 - 680 \cdot 0 = 2280 \Rightarrow \underline{U_2^1 = 1{,}5833}$$

2nd eq

$$-920\, U_2 + 2160\, U_3 - 1240\, U_4 = 0 \rightarrow -920 \cdot 0 + 2160\, U_3^1 - 1240 \cdot 0 = 0$$

$$\Downarrow$$

$$\underline{U_3^1 = 0}$$

3rd eq

$$-1480\, U_3 + 3760\, U_4 = 34200 \rightarrow 1480 \cdot 0 + 3760\, U_4^1 = 34200 \Rightarrow \underline{U_4^1 = 9{,}096}$$

New solution vectr $\underline{\underline{x^1 = (1{,}5833 \;,\; 0 \;,\; 9{,}096)^T}}$

2

2nd iteration

1st eq

$$1440\, U_2 - 680\, U_3 = 2280 \rightarrow 1440\, U_2^2 - 680 \cdot 0 = 2280 \Rightarrow \underline{U_2^2 = 1{,}5833}$$

2nd eq

$$-920\, U_2 + 2160\, U_3 - 1240\, U_4 = 0 \rightarrow -920 \cdot 1{,}5833 + 2160\, U_3^2 - 1240 \cdot 9{,}096 = 0$$

$$\Downarrow$$

$$\underline{U_3^2 = 5{,}896}$$

3rd eq

$$-1480\, U_3 + 3760\, U_4 = 34200 \rightarrow -1480 \cdot 0 + 3760\, U_4^2 = 34200 \Rightarrow \underline{U_4^2 = 9{,}09574}$$

New solution vector $\underline{\underline{x^2 = (1{,}5833 \;,\; 5{,}8969 \;,\; 9{,}09574)^T}}$

2

c) Gauss - seidel

$$\begin{pmatrix} 1440 & -680 & 0 \\ -920 & 2160 & -1240 \\ 0 & -1480 & 3760 \end{pmatrix} \begin{pmatrix} U_2 \\ U_3 \\ U_4 \end{pmatrix} = \begin{pmatrix} 2280 \\ 0 \\ 34200 \end{pmatrix}$$

Initial vector $x^0 = 0$

0

~~2 iterations~~

1st iteration

1st equation

$1440\,U_2 - 680\,U_3 = 2280 \longrightarrow 1440\,U_2^1 - 680 \cdot 0 = 2280 \Rightarrow U_2^1 = 1{,}5833$

2nd equation

$-920\,U_2 + 2160\,U_3 - 1240\,U_4 = 0 \rightarrow -920(1{,}5833) + 2160\,U_3^1 - 1240 \cdot 0 = 0$

$\Downarrow\ U_3^1 = 0{,}6744$

3rd equation

$-1480\,U_3 + 3760\,U_4 = 34200 \longrightarrow -1480 \cdot (0{,}6744) + 3760\,U_4^1 = 34200$

$\Downarrow\ U_4^1 = 9{,}411$

2

New Solution vector $\underline{(1{,}5833\ ,\ 0{,}6744\ ,\ 9{,}411)^T = x^1}$

2nd iteration

1st equation

$1440\,U_2 - 680\,U_3 = 2280 \longrightarrow 1440\,U_2^2 - 680 \cdot \underline{0{,}000266} = 2280$

?

$\Downarrow\ \underline{U_2^2 = 1{,}583}$

2nd equation

O

$-920\,U_2 + 2160\,U_3 - 1240\,U_4 = 0 \qquad -920 \cdot 1{,}583 + 2160 \cdot U_3^2 - 1240 \cdot \underline{9{,}0959} = 0$

?

$\Downarrow\ \underline{U_3^2 = 5{,}895}$

3rd equation

$-1480\,U_3 + 3760\,U_4 = 34200 \longrightarrow -1480 \cdot 5{,}895 + 3760\,U_4^1 = 34200 \Rightarrow \underline{U_4^1 = 11{,}4161}$

New Solution vector $\underline{x^2 = (1{,}583\ ,\ 5{,}895\ ,\ 11{,}4161)^T}$

d) Gauss Seidel

- Iterative method
- coverges if A is diagonally dominant OR Symmetric AND positive definite

Jacobi

- Iterative
- Converges if A is diagonally dominant

1.5

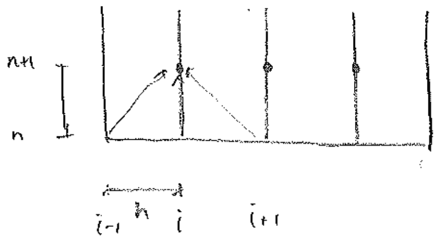
If both converges Gauss-Seidel is faster than Jacobi

Question 4

$$\frac{\partial U}{\partial t} = b\frac{\partial^2 U}{\partial x^2} \qquad 0 \leq x \leq 1$$

$U(0,t) = 1 \qquad U(1,t) = 1 \qquad$ B.C

$U(x,0) = U_0(t) \qquad$ IC



a) $\Delta t = 0{,}25$, $h = 0{,}25$, $b = 3$ $\Rightarrow \mu = \cancel{3}\ 12$ $\qquad \mu = \frac{b\Delta t}{h^2}$

$$U_2^1 - U_2^0 = \mu(U_3^0 - 2U_2^0 + U(0,t))$$
$$\Downarrow$$
$$U_2^1 = 3\cdot 8 - 6\cdot 4 + 1\cdot 3 + 4 = \underline{\underline{7}}$$

$$U_3^1 - U_3^0 = \mu(U_4^0 - 2U_3^0 + U_2^0)$$
$$\Downarrow$$
$$U_3^1 = 3\cdot 4 - 6\cdot 8 + 3\cdot 4 + 8 = \underline{\underline{-16}}$$

$$U_4^1 - U_4^0 = \mu(U_3^0 - 2U_4^0 + U(1,t))$$
$$\Downarrow$$
$$U_4^1 = 3\cdot 8 - 6\cdot 4 + 3\cdot 1 + 4 = \underline{\underline{7}}$$

stability ?

b) Truncation error

$$\left.\frac{dU}{dt}\right|_i^{n+1} = \frac{U_{i+1}^{n+1} - U_i^n}{\Delta t} \qquad ①$$

$$\left.\frac{d^2U}{dx^2}\right|_i^{n+1} = \frac{U_{i+1}^{n+1} - 2U_i^{n+1} + U_{i-1}^{n+1}}{\Delta x^2} \qquad ②$$

3 taylor expansions

$$U_i^n = \cancel{U_i^{n+1}} - \Delta t \frac{dU_i^{n+1}}{dt} + \Delta t^2/2 \frac{d^2U_i^{n+1}}{dt^2}$$

$$U_{i+1}^{n+1} = \cancel{U_i^{n+1}} + \cancel{\Delta x \frac{dU_i^{n+1}}{dx}} + \cancel{\Delta x^2}/2 \frac{d^2U_i^{n+1}}{dx^2} + \cancel{\Delta x^3/6 \frac{d^3U_i^{n+1}}{dx^3}} + \Delta x^4/24 \frac{d^4U_i^{n+1}}{dx^4}$$

$$U_{i+1}^{n+1} = \cancel{U_i^{n+1}} - \cancel{\Delta x \frac{dU_i^{n+1}}{dx}} + \cancel{\Delta x^2}/2 \frac{d^2U_i^{n+1}}{dx^2} - \cancel{\Delta x^3/6 \frac{d^3U_i^{n+1}}{dx^3}} + \Delta x^4/24 \frac{d^4U_i^{n+1}}{dx^4}$$

Inserts these in ① and ② and we know $\frac{dU}{dt} = b\frac{d^2U}{dx^2}$

$$\frac{dU_i^{n+1}}{dt} + 1/2\, \Delta t \frac{d^2U_i^{n+1}}{dt^2} = b\left(\frac{d^2U_i^{n+1}}{dx^2} + 1/12\, \Delta x^2 \frac{d^4U_i^{n+1}}{dx^4}\right)$$

$$\cancel{\frac{dU_i^{n+1}}{dt}} + 1/2\, \Delta t \frac{d^2U_i^{n+1}}{dt^2} = \cancel{b\frac{d^2U_i^{n+1}}{dx^2}} + 1/12\, \Delta x^2 \frac{d^4(U_i^{n+1}}{dx^4} \cdot b$$

$$\underline{\underline{\tau_i = -1/2\, \Delta t \frac{d^2U_i^{n+1}}{dt^2} - 1/2\, \Delta x^2 b \frac{d^4U_i^{n+1}}{dx^4}}} \quad + \text{HOT}$$

c) $U_i^{n+1} - U_i^n = \mu\left(U_{i+1}^{n+1} - 2U_i^{n+1} + U_{i-1}^{n+1}\right)$

$$U_i^n = -\mu U_{i+1}^{n+1} + 2\mu U_i^{n+1} - U_{i-1}^{n+1} + U_i^{n+1}$$

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

$$\begin{bmatrix} (1+2\mu) & -\mu & 0 \\ -\mu & (1+2\mu) & -\mu \\ 0 & -\mu & (1+2\mu) \end{bmatrix} \begin{bmatrix} U_2^1 \\ U_3^1 \\ U_4^1 \end{bmatrix} = \begin{bmatrix} U_2^0 + \mu U(0,t) \\ U_3^0 \\ U_4^0 + \mu U(1,t) \end{bmatrix}$$

d)

$$\begin{bmatrix} 1+6 & -3 & 0 \\ -3 & 1+6 & -3 \\ 0 & -3 & 1+6 \end{bmatrix} \begin{bmatrix} U_2' \\ U_3' \\ U_4' \end{bmatrix} = \begin{bmatrix} 4+1 \\ 8+1 \\ 4+1 \end{bmatrix} \longrightarrow \begin{bmatrix} 1'55 \\ \cancel{3}1'95 \\ 1'55 \end{bmatrix}$$

$$\text{row 2} = \text{row2} - \left(-3/7\right) \cdot \text{row1} \rightarrow \quad a_{21} = -3 - \left(-3/7\right) \cdot 7 = 0$$

$$a_{22} = 7 - \left(-3/7\right) \cdot (-3) = 40/7$$

$$a_{23} = -3 - \left(3/7\right) \cdot 0 = -3$$

$$x_2' = 9 - \left(-3/7\right) \cdot 5 = 78/7$$

$$\begin{bmatrix} 7 & -3 & 0 \\ 0 & 40/7 & -3 \\ 0 & -3 & 7 \end{bmatrix} \begin{bmatrix} U_2' \\ U_3' \\ U_4' \end{bmatrix} = \begin{bmatrix} 5 \\ 78/7 \\ 5 \end{bmatrix}$$

$$\text{row 3} = \text{row3} - \left(-3/(40/7)\right) \cdot \text{row 2} \rightarrow \quad a_{32} = -3 - \left(-3/(40/7)\right) \cdot 40/7 = 0$$

$$a_{33} = 7 - \left(-3/(40/7)\right) \cdot (-3) = 217/40$$

$$x_3' = 5 - \left(-3/40/7\right) \cdot (78/7) = 217/40$$

$$\begin{bmatrix} 7 & -3 & 0 \\ 0 & 40/7 & -3 \\ 0 & 0 & 217/40 \end{bmatrix} \begin{bmatrix} U_2' \\ U_3' \\ U_4' \end{bmatrix} = \begin{bmatrix} 5 \\ 78/7 \\ 217/40 \end{bmatrix}$$

$$U_4' \cdot 217/4 = 217/40 \Rightarrow \underline{\underline{U_4' = 1}}$$

$$40/7\, U_3' - 3 \cdot U_4' = 78/7 \Rightarrow U_3' = \underline{\underline{57/40}} = 1'4250$$

$$7U_2' - 3U_3' = 5 \Rightarrow U_2' = \underline{\underline{53/40}}$$

compored to the results from FTCS we can see that the solution <u>is stable</u>! FTCS was unstable since $\underline{\underline{M = 3 > 1/2}}$

STUDENT 1

FINAL EXAM

(NOTABLE)

Question 1

a) n point Gaussian quadrature: a 2n+2 system is set to define the integration points and weights, thus we obtain a quadrature that can integrate polinomials of up to $\boxed{2n+1}$ order. 1

b) i) polynomial of order 10, $n=\overset{6}{5} \rightarrow q = 2\cdot 5+1 = 11$, can be integrated.

ii) not a polynomial, cannot be integrated exactly 2.5

iii) polynomial of order 14, $n=\overset{8}{7} \rightarrow q = 2\cdot 7+1 = 15$, can be integrated.

iv) polynomial of order 15, $n=\overset{8}{7} \rightarrow q = 2\cdot 7+1 = 15$, can be integrated.

c) $w_1 = w_2 = 1$ - weights $\quad \xi_1 = \frac{-1}{\sqrt{3}}$; $\xi_2 = \frac{1}{\sqrt{3}}$ points.

~~we adopt Gauss-Legendre integration method~~

~~$dx = \frac{\pi - 0}{2} dz = \frac{\pi}{2} dz \quad z_1 =$~~

$$I = \int_0^{\pi} (13 - 8(\cos x)^2)\, dx \simeq w_1 \cdot f(\xi_1) + w_2 \cdot f(\xi_2) =$$

$$= 13 - 8\cdot(\cos(-0{,}57))^2 + 13 - 8\cdot(\cos(0{,}57))^2 =$$ 0

$\xi = -0{,}57$

$\xi = 0{,}57$

$$= 5 + 5 = \underline{\underline{10}}$$

d) $\frac{d^2Y}{dt^2} + \frac{dY}{dt} + 4Y^4 = 0 \qquad Y(0) = 0, \quad \frac{dY}{dt}(0) = 1$

$f(\bar{Y}, t) + Y_2 + 4Y^4 = 0$

we define: $Y_1 = Y$

$Y_2 = \frac{dY_1}{dt} = \frac{dY}{dt}$

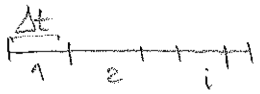
$Y_3 = \frac{dY_2}{dt} = \frac{d^2Y}{dt^2} = f(Y, t)$

$\bar{Y} = \begin{bmatrix} Y_1 \\ Y_2 \end{bmatrix} = \begin{bmatrix} Y \\ \frac{dY}{dt} \end{bmatrix}$ IC: $\bar{Y}(0) = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$

$\bar{f}(\bar{Y}, t) = \begin{bmatrix} Y_2 \\ -Y_2 - 4Y^4 \end{bmatrix}$

d1) the system of first order ODE:

$\bar{Y} = \begin{bmatrix} Y_1 \\ Y_2 \end{bmatrix} = \begin{bmatrix} Y \\ \frac{dY}{dt} \end{bmatrix} \quad \bar{f}(\bar{Y}, t) = \begin{bmatrix} Y_2 \\ -Y_2 - 4Y_1^4 \end{bmatrix}$ with initial condition: $\bar{Y}(0) = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$ ✓ 4



d2) Backward Euler:

$$\bar{y}^{i+1} = \bar{y}^{i} + \Delta t \cdot \bar{f}(t^{i+1}, \bar{y}^{i+1}) \qquad i = 0, 1, \dots$$

$$\bar{y}^{1} = \bar{y}^{0} + \Delta t\, \bar{f}(t^0 + \Delta t, \bar{y}^{1}) \qquad t^0 = 0$$

$$\begin{bmatrix} y_1^1 \\ y_2^1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} + \Delta t \begin{bmatrix} y_2^1 \\ -y_2^1 - 4y_1^4 \end{bmatrix}$$

5

$$\begin{cases} y_1^1 - \Delta t \cdot y_2^1 = 0 \\ y_2^1(1+\Delta t) + 4\Delta t\, y_1^4 = 1 \end{cases}$$

✓

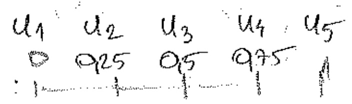
d3)

5

explicit Forward Euler	implicit Backward Euler
+ easy to compute, and not expensive;	− costly to compute, each step involves solving an eq. system
− conditionally stable, concerning the timestep Δt (does not always converge)	+ better convergence, and uconditionally stable

Problem 4

$$u_i^{n+1} - u_i^{n} = \mu(u_{i+1}^{n} - 2u_i^{n} + u_{i-1}^{n}) \qquad \mu = \frac{\Delta t\, b}{h^2}$$



a) $b = 3$, $\Delta t = 0{,}25$ $\quad h = \frac{1}{4} = 0{,}25$ $\qquad [u^0] = [1 \;\; 4 \;\; 8 \;\; 4 \;\; 1]^T$

$$\mu = \frac{0{,}25 \cdot 3}{0{,}25^2} = \frac{3}{0{,}25} = 12$$

$\underline{n=0}$

$$u_i^1 - u_i^0 = 12u_{i+1}^0 - 24u_i^0 + 12u_{i-1}^0$$

$$u_i^1 = 12u_{i+1}^0 + (1-24)u_i^0 + 12u_{i-1}^0 \qquad i = 2, 3, 4$$

$\underline{i=2}$

$$u_2^1 = 12u_3^0 \overset{-}{\times} 23u_2^0 + 12u_1^0 = 12\cdot 8 + 23\cdot 4 + 12\cdot 1 = \underline{200}$$

$\underline{i=3}$

$$u_3^1 = 12u_4^0 \overset{-}{\times} 23u_3^0 + 12u_2^0 = 12\cdot 4 + 23\cdot 8 + 12\cdot 4 = \underline{280}$$

$\underline{i=4}$

$$u_4^1 = 12\cdot u_5^0 \overset{-}{\times} 23u_4^0 + 12u_3^0 = 12\cdot 1 + 23\cdot 4 + 12\cdot 8 = \underline{200}$$

$$[u^1] = [1 \;\; 200 \;\; 280 \;\; 200 \;\; 1]^T$$

b) the stability condition is $r = \frac{b\Delta t}{\Delta x^2} \leq \frac{1}{2}$ $\left(\leq \frac{1}{4} \text{ to avoid oscillations}\right)$

in our case: $r = \mu = \frac{b\Delta t}{\Delta x^2} = \underline{12}$.

As it can be seen by the first iteration u^1 performed, the method is not stable, not converging:

- $u_1^1 << u_2^1$: $1 << 200$, since it is a diffusion equation the information should diffuse smoothly through the domain.
- $u_i^1 >> u_i^0$ $\quad i = 2,3,4$

b2) $u_i^{n+1} - u_i^n = \mu\left(u_{i+1}^{n+1} - 2u_i^{n+1} + u_{i-1}^{n+1}\right)$

we express the terms u_{i+1}^{n+1}, u_{i-1}^{n+1} using Taylor expansion centred in u_i^{n+1}

$$u_{i+1}^{n+1} = u_i^{n+1} + h\frac{du}{dx}\Big|_i^{n+1} + \frac{h^2}{2}\frac{du}{dx}\Big|_i^{n+1} + \frac{h^3}{6}\frac{du^3}{dx^2}\Big|_i^{n+1} + \frac{h^4}{24}\frac{d^4u}{dx^4}\Big|_i^{n+1} + \theta(h^5)$$

$$u_{i-1}^{n+1} = u_i^{n+1} - h\frac{du}{dx}\Big|_i^{n+1} + \frac{h^2}{2}\frac{d^2u}{dx^2}\Big|_i^{n+1} - \frac{h^3}{6}\frac{d^3u}{dx^3}\Big|_i^{n+1} + \frac{h^4}{24}\frac{d^4u}{dx^4}\Big|_i^{n+1} + \theta(h^5)$$

$$u_i^n = u_i^{n+1} - \Delta t\frac{du}{dt}\Big|_i^{n+1} + \frac{\Delta t^2}{2}\frac{d^2u}{dt^2}\Big|_i^{n+1} - \frac{\Delta t^3}{6}\frac{d^3u}{dt^3}\Big|_i^{n+1} + \theta(\Delta t^4)$$

$$\tau = \cancel{u_i^{n+1}} - \cancel{u_i^{n+1}} + \Delta t\frac{du}{dt}\Big|_i^{n+1} - \frac{\Delta t^2}{2}\frac{d^2u}{dt^2}\Big|_i^{n+1} + \frac{\Delta t^3}{6}\frac{d^3u}{dt^3}\Big|_i^{n+1} + \theta(\Delta t^4) \qquad h = \Delta x$$

$$= \mu\Bigg(\cancel{u_i^{n+1}} \cancel{\boxed{+ h\frac{du}{dx}\Big|_i^{n+1}}} + \frac{h^2}{2}\frac{d^2u}{dx^2}\Big|_i^{n+1} \cancel{+ \frac{h^3}{6}\frac{d^3u}{dx^3}\Big|_i^{n+1}} + \frac{h^4}{24}\frac{d^4u}{dx^4}\Big|_i^{n+1} - \cancel{2u_i^{n+1}}$$

$$\cancel{+ u_i^{n+1}} \cancel{\boxed{- h\frac{du}{dx}\Big|_i^{n+1}}} + \frac{h^2}{2}\frac{d^2u}{dx^2}\Big|_i^{n+1} \cancel{- \frac{h^3}{6}\frac{d^3u}{dx^3}\Big|_i^{n+1}} + \frac{h^4}{24}\frac{d^4u}{dx^4}\Big|_i^{n+1} + \theta(h^5)\Bigg)$$

$$\Delta t\frac{du}{dt}\Big|_i^{n+1} - \frac{\Delta t^2}{2}\frac{d^2u}{dt^2}\Big|_i^{n+1} + \frac{\Delta t^3}{6}\frac{du}{dt^3}\Big|_i^{n+1} + \theta(\Delta t^4) = \frac{\Delta t \cdot b}{\Delta x^2}\left[\Delta x^2\frac{d^2u}{dx^2}\Big|_i^{n+1} + \frac{\Delta x^4}{12}\frac{d^4u}{dx^4}\Big|_i^{n+1} + \theta(\Delta x^5)\right]$$

$$\frac{du}{dt}\Big|_i^{n+1} - \frac{\Delta t}{2}\frac{d^2u}{dt^2}\Big|_i^{n+1} + \theta(\Delta t^2) = b\frac{d^2u}{dx^2}\Big|_i^{n+1} + b\frac{\Delta x^2}{12}\frac{d^4u}{dx^4}\Big|_i^{n+1} + b\theta(\Delta x^3)$$

We notice that: $\frac{\partial u}{\partial t} = b\frac{\partial^2 u}{\partial x^2}$

$$\cancel{b\frac{d^2u}{dx^2}\Big|_i^{n+1}} - \frac{\Delta t}{2}\frac{d^2u}{dt^2}\Big|_i^{n+1} + \theta(\Delta t^2) = \cancel{b\frac{d^2u}{dx^2}\Big|_i^{n+1}} + b\frac{\Delta x^2}{12}\frac{d^4u}{dx^4}\Big|_i^{n+1} + b\theta(\Delta x^3)$$

we arrive to:

$$\tau = \cancel{4}\frac{\Delta x^2}{12}\frac{d^4u}{dx^4}\bigg|_i^{n+1} + \frac{\Delta t}{2}\frac{d^2u}{dt^2}\bigg|_i^{n+1} + \cancel{4}\,\mathcal{O}(\Delta x^3) + \mathcal{O}(\Delta t^2)$$

it is straight forward to see that: $\underline{\underline{\tau(\Delta t, \Delta x^2)}}$.

c) $u_i^{n+1} - u_i^n = \mu(u_{i+1}^{n+1} - 2u_i^{n+1} + u_{i-1}^{n+1})$

$n = 0$

$u_i^1 - u_i^0 = \mu(u_{i+1}^1 - 2u_i^1 + u_{i-1}^1)$

$-\mu u_{i+1}^1 + (1+2\mu)u_i^1 - \mu u_{i-1}^1 = u_i^0$

$\underline{i = 2}$

BOUNDARY NODE

$-\mu u_3^1 + (1+2\mu)u_2^1 \boxed{-\mu u_1^1} = u_2^0 \rightarrow -\mu u_3^1 + (1+2\mu)u_2^1 = u_2^0 + \mu u_1^1$

$\underline{i = 3}$

$-\mu u_4^1 + (1+2\mu)u_3^1 - \mu u_2^1 = u_3^0$

$\underline{i = 4}$

$\boxed{-\mu u_5^1} + (1+2\mu)u_4^1 - \mu u_3^1 = u_4^0 \rightarrow (1+2\mu)u_4^1 - \mu u_3^1 = u_4^0 + \mu u_5^1$

Boundary node

$$\begin{array}{c|ccc|} & u_2 & u_3 & u_4 \\ \hline u_2 & 1+2\mu & -\mu & 0 \\ u_3 & -\mu & 1+2\mu & -\mu \\ u_4 & 0 & -\mu & 1+2\mu \end{array}\begin{bmatrix} u_2^1 \\ u_3^1 \\ u_4^1 \end{bmatrix} = \begin{bmatrix} u_2^0 \\ u_3^0 \\ u_4^0 \end{bmatrix} + \begin{bmatrix} \mu u_1^1 \\ 0 \\ \mu u_5^1 \end{bmatrix} \rightarrow \begin{bmatrix} 25 & -12 & 0 \\ -12 & 25 & -12 \\ 0 & -12 & 25 \end{bmatrix}\begin{bmatrix} u_2^1 \\ u_3^1 \\ u_4^1 \end{bmatrix} = \begin{bmatrix} 4 \\ 8 \\ 4 \end{bmatrix} + \begin{bmatrix} 12 \\ 0 \\ 12 \end{bmatrix}$$

d) $\begin{bmatrix} 25 & -12 & 0 \\ -12 & 25 & -12 \\ 0 & -12 & 25 \end{bmatrix}\begin{bmatrix} u_2^1 \\ u_3^1 \\ u_4^1 \end{bmatrix} = \begin{bmatrix} 16 \\ 8 \\ 16 \end{bmatrix}$

$\underline{1\,\text{step}}$: row 2 $- \left(\frac{-12}{25}\right)$ row 1 =

$$= \begin{array}{ccc|c} -12 & 25 & -12 & 16 \\ 12 & -5{,}76 & 0 & 7{,}68 \\ \hline 0 & 19{,}24 & -12 & 15{,}68 \end{array} \quad +0{,}48 \cdot \text{row } 1$$

$\begin{bmatrix} 25 & -12 & 0 \\ 0 & 19{,}24 & -12 \\ 0 & -12 & 25 \end{bmatrix}\begin{bmatrix} u_2^1 \\ u_3^1 \\ u_4^1 \end{bmatrix} = \begin{bmatrix} 16 \\ 15{,}68 \\ 16 \end{bmatrix}$

$\underline{2\,\text{step}}$: row 3 $- \left(\frac{-12}{19{,}24}\right)$ row 2 =

$$\begin{array}{ccc|c} 0 & -12 & 25 & 16 \\ 0 & 0 & 17{,}52 & 25{,}76. \end{array} \quad + 0{,}623 \cdot \text{row } 2$$

$$\begin{bmatrix} 25 & -12 & 0 \\ 0 & 19{,}24 & -12 \\ 0 & 0 & 17{,}52 \end{bmatrix} \begin{bmatrix} u_2^1 \\ u_3^1 \\ u_4^1 \end{bmatrix} = \begin{bmatrix} 16 \\ 15{,}68 \\ 25{,}76 \end{bmatrix}$$

$u_4^1 = 1{,}47$

$19{,}24 u_3^1 = 15{,}68 + 12 \cdot 1{,}47$ $\quad u_3^1 = 1{,}73$

$25 u_2^1 = 16 + 12 \cdot u_3^1$ $\quad u_2^1 = 1{,}47$

$\underline{[u^1] = [1 \ \ 1{,}47 \ \ 1{,}73 \ \ 1{,}47 \ \ 1]^T}$ ✓

Stability:

- it is obvious to see that the information is well diffused over the domain $[0,1]$.
- there are no jumps ~~neither~~ in space along u^1;
- the ~~solution~~ method converges fast $u^1 < u^0$.

Question 2

a). we denote $r^k = b - A\underline{x}$ the residual.

the iterative methods have a generic form ~~of~~ $\underline{C} \cdot \Delta x^{k+1} = r^k$

where C is considered to be the iteration matrix.

for the Jacobi method $C = D_A$, where $A = D_A + L_A + U_A$

D_A - diagonal matrix of A

L_A - lower triangular, w

U_A - upper triangular.

✓

$C \cdot \Delta x^{k+1} = b - A\underline{x} \quad | C = D_A$

3

$\Delta x^{k+1} = b \cdot D_A^{-1} - U_A \cdot x - L_A \cdot x$

in practis we never invert the C matrix.

$x^{k+1} = x^k + \Delta x^{k+1}$

iterations:

1). $1440 u_2^1 - 680 u_3^0 = 2280$ $\quad -920 u_2^0 + 2160 u_3^1 - 1240 \cdot u_4^0 = 0$

$\underline{u_2^1} = \frac{2280}{1440} = 1{,}58$ $\quad \underline{u_3^1} = 0$

2

$-1480 \cdot u_3^0 + 3760 u_4^1 = 34200$

$\underline{u_4^1 = 9{,}09}$ $\quad [u^1] = [3 \ \ 1{,}58 \ \ 0 \ \ 9{,}09 \ \ 15]^T$

$[U^1] = [3 \;\; 1{,}58 \;\; 0 \;\; 9{,}09 \;\; 15]$

2) $1440 \cdot U_2^2 - 680 \cdot U_3^1 = 2280$ $\quad -920 U_2^1 + 2160 U_3^2 - 1240 U_4^1 = 0$ $\quad -1480 U_3^1 + 3760 U_4^2 = 34200$

$\underline{\underline{U_2^2}} = \frac{2280 - 680 \cdot 0}{1440} = 1{,}58$ ✓ $\quad \underline{\underline{U_3^2}} = 12{,}72$ $\quad \underline{\underline{U_4^2}} = 9{,}09$ ✓

$[U^2] = [3 \;\; 1{,}58 \;\; 12{,}72 \;\; 9{,}09 \;\; 15]^T$

1.5

c) Gauss-Seidel. $C = D_A + L_A$

$\Delta x^{k+1} = b(D_A + L_A)^{-1} - U_A x^k$

3

$\underline{\text{iterations:}}$

1) $1440 U_2^1 - 680 U_3^0 = 2280$ $\quad -920 \cdot U_2^1 + 2160 U_3^1 - 1240 U_4^0 = 0$ $\quad -1480 U_3^1 + 3760 U_4^1 = 34200$

$\underline{\underline{U_2^1 = 1{,}58}}$ $\quad \underline{\underline{U_3^1 = 0{,}67}}$ $\quad \underline{\underline{U_4^1 = 9{,}36}}$ $\quad$ 2

$[U^1] = [3 \;\; 1{,}58 \;\; 0{,}67 \;\; 9{,}36 \;\; 15]^T$

2) $1440 U_2^2 - 680 U_3^1 = 2280$ $\quad -920 U_2^2 + 2160 U_3^2 - 1240 U_4^1 = 0$

$-1480 \cdot U_3^2 + 3760 \cdot U_4^2 = 34200$

$\underline{\underline{U_2^2 = 1{,}89}}$ $\quad \underline{\underline{U_3^2 = 6{,}18}}$ $\quad \underline{\underline{U_4^2 = 11{,}52}}$ $\quad$ 2

$[U^2] = [3, 1{,}89, 6{,}18, 11{,}52, 15]^T$

d)
- Gauss-Seidel method converges faster than Jacobi, because during computations / iterations we take advantage of the partial results already obtained.

2

- As we can see U_J^2 has oscillations while U_{GS}^2 is smoother.
- While GS offeres better convergence the computational cost is the same in terms of memory and operations.

FINAL EXAM

STUDENT 2

(NOTABLE)

Question 1

a) q: order of the polynomial that is integrated exactly.
2 n: nº of points of the Gaussian quadrature.

✓ $q = 2\cdot n - 1$

b) Integrals (i), (iii) and (iv) can be integrated exactly using a Gauss quadrature because they are polynomials. On the contrary, integral (ii) cannot be exactly integrated using a Gauss quadrature because it is a trigonometric function.
4 ✓

Integral (i): order of the polynomial: 10.
optimal order of the rule: 10
optimal nº of gaussian points : 6, it gives a rule of order 11.

Integral (iii): order of the polynomial: 14
optimal order of the rule: 14
optimal nº of gaussian points: 8, it gives a rule of order 15.

Integral (iv): order of the polynomial: 15.
optimal order of the quadrature: 15
optimal nº of gaussian points: 8

c) $z_1 = -1/\sqrt{3}$ $w_1 = 1$
4. $z_2 = +1/\sqrt{3}$ $w_2 = 1$

$$I = \int_0^{\pi} (13 - 8(\cos x)^2)\,dx = \int_{-1}^{1} \frac{\pi}{2} \underbrace{\left[13 - 8\left(\cos\left(\frac{\pi}{2}z + \frac{\pi}{2}\right)\right)^2\right]}_{F(z)} dz \approx \left(\sum_{i=1}^{2} w_i F(z_i)\right)\frac{\pi}{2}$$

$$F(z_1) = 13 - 8\left(\cos\left(\frac{\pi}{2}\cdot\frac{-1}{\sqrt{3}} + \frac{\pi}{2}\right)\right)^2 = 8{'}0375;$$

$$F(z_2) = 13 - 8\left(\cos\left(\frac{\pi}{2}\cdot\frac{1}{\sqrt{3}} + \frac{\pi}{2}\right)\right)^2 = 8{'}0375;$$

$$I \approx \frac{\pi}{2}\cdot(8{'}0375\cdot 1 + 8{'}0375\cdot 1) = 25{'}2506;$$

✓

d.1) $y'' + y' + 4y^4 = 0; \quad y'' = -y' - 4y^4$

4 $u_1 = y$

$u_2 = u_1' = y' \Rightarrow u_1' = u_2$

$u_3 = u_2' = y'' \Rightarrow u_2' = y'' = -y' - 4y^4 = -u_2 - 4u_1^4;$

$y(0) = 0 \rightarrow u_1(0) = 0$

$y'(0) = 1 \rightarrow u_2(0) = 1$

$$\boxed{\begin{aligned} u_1' &= u_2 \\ u_2' &= -u_2 - 4u_1^4 \\ u_1(0) &= 0 \\ u_2(0) &= 1 \end{aligned}}$$

✓

d.2) $u_{1,i+1} = u_{1,i} + \Delta t \cdot f_1(x_{i+1}, u_{i+1})$;

5 $u_{1,i+1} = u_{1,i} + \Delta t \cdot u_{2,i+1}$

$u_{2,i+1} = u_{2,i} + \Delta t \cdot f_2(x_{i+1}, u_{i+1})$;

$u_{2,i+1} = u_{2,i} + \Delta t'(-u_{2,i+1} - 4u_{1,i+1}^4)$

Backward Euler method: $\boxed{\begin{aligned} u_{1,i+1} &= u_{1,i} + \Delta t \cdot u_{2,i+1} \\ u_{2,i+1} &= u_{2,i} - \Delta t \cdot u_{2,i+1} - \Delta t \cdot 4u_{1,i+1}^4 \end{aligned}}$

The non-linear system of equations to be solved:

$u_{1,i+1} - \Delta t \cdot u_{2,i+1} - u_{1,i} = 0$

$4\Delta t \, u_{1,i+1}^4 + (1+\Delta t) u_{2,i+1} - u_{2,i} = 0$

Which for the first time step ($i=0$) is:

$u_{1,1} - \Delta t \cdot u_{2,1} - u_{1,0} = 0$ with $u_{1,0} = 0$

$4\Delta t \, u_{1,1}^4 + (1+\Delta t) u_{2,1} - u_{2,0} = 0$ with $u_{2,0} = 1$

and unknowns: $u_{1,1}$ and $u_{2,1}$

✓

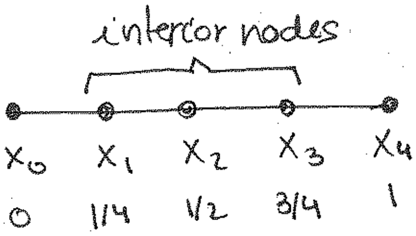
d.3)

4

	Advantages	Disadvantages
Explicit	Low computational cost.	Slow convergence. ?
Implicit	High computational cost.	Fast convergence. ?

Explain better

Question 4



a) $\mu = b.\frac{\Delta t}{h^2}$; $b=3$; $\Delta t = 0'25$;

$h = 0'25$ $\quad \mu = 12$

$$U_i^{n+1} = U_i^n + \mu\left(U_{i+1}^n - 2U_i^n + U_{i-1}^n\right); \quad U_0(t) = 1; \quad U_4(t) = 1;$$

$\boxed{n=0}$ first time step.

$i=1$; $\quad U_1^1 = U_1^0 + 12\left(U_2^0 - 2U_1^0 + U_0^0\right) = 4 + 12(8 - 2\cdot 4 + 1) = 16$;

$i=2$; $\quad U_2^1 = U_2^0 + 12\left(U_3^0 - 2U_2^0 + U_1^0\right) = 8 + 12(4 - 2\cdot 8 + 4) = -88$;

$i=3$; $\quad U_3^1 = U_3^0 + 12\left(U_4^0 - 2U_3^0 + U_2^0\right) = 4 + 12(1 - 2\cdot 4 + 8) = 16$; ✓

b) Stability condition: $\mu = b.\frac{\Delta t}{h^2} \leq 1/2$. Since $\mu = 12$ for $b=3$, $\Delta t = 0'25$ and $h = 0'25$, the above calculations are not ✓ stable.

c) Implicit backward time, centered space:

$$U_i^{n+1} - U_i^n = \mu.\left(U_{i+1}^{n+1} - 2U_i^{n+1} + U_{i-1}^{n+1}\right);$$

$$U_i^{n+1} - U_i^n = \frac{\Delta t}{h^2}.b\left(U_{i+1}^{n+1} - 2U_i^{n+1} + U_{i-1}^{n+1}\right);$$

$$\frac{U_i^{n+1} - U_i^n}{\Delta t} = b.\frac{U_{i+1}^{n+1} - 2U_i^{n+1} + U_{i-1}^{n+1}}{\Delta x^2}; \quad \text{for convenience } h = \Delta x.$$

$$\tau_i(\Delta t, \Delta x) = \frac{u_i^{n+1} - u_i^n}{\Delta t} - b.\frac{u_{i+1}^{n+1} - 2u_i^{n+1} + u_{i-1}^{n+1}}{\Delta x^2};$$

$$u_i^n = u_i^{n+1} - \frac{\partial u}{\partial t}\bigg|_i^{n+1}\Delta t + \frac{\partial^2 u}{\partial t^2}\bigg|_i^{n+1}\frac{\Delta t^2}{2} - \frac{\partial^3 u}{\partial t^3}\bigg|_i^{n+1}\frac{\Delta t^3}{6} + O(\Delta t^4).$$

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} = \frac{\cancel{u_i^{n+1}} - \cancel{u_i^{n+1}} + \frac{\partial u}{\partial t}\Big|_i^{n+1}\cancel{\Delta t} - \frac{\partial^2 u}{\partial t}\Big|_i^{n+1}\frac{\cancel{\Delta t^2}}{2} + \frac{\partial^3 u}{\partial t^3}\Big|_i^{n+1}\frac{\Delta t^{\cancel{3}\,2}}{6} - O(\Delta t^{\cancel{4}})^3}{\cancel{\Delta t}}$$

$$\boxed{\frac{u_i^{n+1} - u_i^n}{\Delta t} = \frac{\partial u}{\partial t}\bigg|_i^{n+1} - \frac{\partial^2 u}{\partial t^2}\bigg|_i^{n+1}\frac{\Delta t}{2} + O(\Delta t^2)}$$

$$u_{i+1}^{n+1} = u_i^{n+1} + \cancel{\frac{\partial u}{\partial x}\bigg|_i^{n+1}\Delta x} + \frac{\partial^2 u}{\partial x^2}\bigg|_i^{n+1}\frac{\Delta x^2}{2} + \cancel{\frac{\partial^3 u}{\partial x^3}\bigg|_i^{n+1}\frac{\Delta x^3}{6}} + \frac{\partial^4 u}{\partial x^4}\bigg|_i^{n+1}\frac{\Delta x^4}{24} + O(\Delta x^5).$$

$$u_{i-1}^{n+1} = u_i^{n+1} - \cancel{\frac{\partial u}{\partial x}\bigg|_i^{n+1}\Delta x} + \frac{\partial^2 u}{\partial x^2}\bigg|_i^{n+1}\frac{\Delta x^2}{2} - \cancel{\frac{\partial^3 u}{\partial x^3}\bigg|_i^{n+1}\frac{\Delta x^3}{6}} + \frac{\partial^4 u}{\partial x^4}\bigg|_i^{n+1}\frac{\Delta x^4}{24} + O(\Delta x^5).$$

$$u_{i+1}^{n+1} - 2u_i^{n+1} + u_{i-1}^{n+1} =$$

$$= \cancel{2u_i^{n+1}} - \cancel{2u_i^{n+1}} + \left.\frac{\partial^2 u}{\partial x^2}\right|_i^{n+1} \Delta x^2 + \left.\frac{\partial^4 u}{\partial x^4}\right|_i^{n+1} \frac{\Delta x^4}{12} + O(\Delta x^6) =$$

$$= \left.\frac{\partial^2 u}{\partial x^2}\right|_i^{n+1} \Delta x^2 + \left.\frac{\partial^4 u}{\partial x^4}\right|_i^{n+1} \frac{\Delta x^4}{12} + O(\Delta x^6).$$

$$\frac{u_{i+1}^{n+1} - 2u_i^{n+1} + u_{i-1}^{n+1}}{\Delta x^2} = \frac{\left.\frac{\partial^2 u}{\partial x^2}\right|_i^{n+1} \cancel{\Delta x^2} + \left.\frac{\partial^4 u}{\partial x^4}\right|_i^{n+1} \frac{\Delta x^{\cancel{4}\,2}}{12} + O(\Delta x^{\cancel{6}\,4})}{\Delta x^2} =$$

$$= \left.\frac{\partial^2 u}{\partial x^2}\right|_i^{n+1} + \left.\frac{\partial^4 u}{\partial x^4}\right|_i^{n+1} \frac{\Delta x^2}{12} + O(\Delta x^4).$$

$$\tau_i = \frac{u_i^{n+1} - u_i^n}{\Delta t} - b \cdot \frac{u_{i+1}^{n+1} - 2u_i^{n+1} + u_{i-1}^{n+1}}{\Delta x^2} =$$

$$= \left.\frac{\partial u}{\partial t}\right|_i^{n+1} + \left.\frac{\partial^2 u}{\partial t^2}\right|_i^{n+1} \frac{\Delta t}{2} + O(\Delta t^2) - b \cdot \left.\frac{\partial^2 u}{\partial x^2}\right|_i^{n+1} - b \cdot \left.\frac{\partial^4 u}{\partial x^4}\right|_i^{n+1} \frac{\Delta x^2}{12} - b\,O(\Delta x^4) =$$

$$= \cancel{\left(\left.\frac{\partial u}{\partial t}\right|_i^{n+1} - b \cdot \left.\frac{\partial^2 u}{\partial x_2}\right|_i^{n+1} \right)} + \left.\frac{\partial^2 u}{\partial t^2}\right|_i^{n+1} \frac{\Delta t}{2} + O(\Delta t^2) - b \cdot \left.\frac{\partial^4 u}{\partial x^4}\right|_i^{n+1} \frac{\Delta x^2}{12} - b\,O(\Delta x^4) =$$

$$\Rightarrow \tau_i = \underbrace{\left.\frac{\partial^2 u}{\partial t^2}\right|_i^{n+1} \frac{\Delta t}{2} + O(\Delta t^2)}_{O(\Delta t)} - \underbrace{b\left(\left.\frac{\partial^4 u}{\partial x^4}\right|_i^{n+1} \frac{\Delta x^2}{12} + O(\Delta x^4) \right)}_{O(\Delta x^2)}$$

The scheme is of order 1 in time and of order 2 in space.

d) After one time step ≡ $n=0$.

$$u_i^1 - u_i^0 = \mu \cdot (u_{i+1}^1 - 2u_i^1 + u_{i-1}^1) -$$

$$u_i^1 - \mu u_{i+1}^1 + 2\mu u_i^1 - \mu u_{i-1}^1 = u_i^0$$

$$\boxed{-\mu \cdot u_{i-1}^1 + (1+2\mu) u_i^1 - \mu u_{i+1}^1 = u_i^0}$$

$i=1 \rightarrow$ $-\mu u_0^1 + (1+2\mu) u_1^1 - \mu \cdot u_2^1 = u_1^0,$

$$(1+2\mu) u_1^1 - \mu \cdot u_2^1 = u_1^0 + \mu \cdot u_0^1$$

$i=2 \rightarrow$ $-\mu u_1^1 + (1+2\mu) u_2^1 - \mu u_3^1 = u_2^0$

$i=3 \rightarrow$ $-\mu u_2^1 + (1+2\mu) u_3^1 - \mu u_4^1 = u_3^0$

$$-\mu \cdot u_2^1 + (1+2\mu) u_3^1 = u_3^0 + \mu \cdot u_4^1$$

Question 4

d)
$$\begin{bmatrix} 1+2\mu & -\mu & 0 \\ -\mu & 1+2\mu & -\mu \\ 0 & -\mu & 1+2\mu \end{bmatrix} \begin{bmatrix} u_1^1 \\ u_2^1 \\ u_3^1 \end{bmatrix} = \begin{bmatrix} u_1^0 + \mu \cdot u_0^1 \\ u_2^0 \\ u_3^0 + \mu \cdot u_4^1 \end{bmatrix}$$

✓

$u_0^1 = 1$; $u_4^1 = 1$

$u_1^0 = 4$; $u_2^0 = 8$; $u_3^0 = 4$

e) $b = 3$, $\Delta t = 0'25$, $h = 0'25 \rightarrow \mu = b \cdot \frac{\Delta t}{h^2} = 12.$

$$\begin{bmatrix} 25 & -12 & 0 \\ -12 & 25 & -12 \\ 0 & -12 & 25 \end{bmatrix} \begin{bmatrix} u_1^1 \\ u_2^1 \\ u_3^1 \end{bmatrix} = \begin{bmatrix} 16 \\ 8 \\ 16 \end{bmatrix} :$$

$$\left[\begin{array}{ccc|c} 25 & -12 & 0 & 16 \\ 0 & 19'24 & -12 & 15'68 \\ 0 & -12 & 25 & 16 \end{array}\right]$$

$X \cdot 25 = -12$; $X = -12/25 = -0'48.$

$$\begin{array}{rrrr} -12 & 25 & -12 & | \; 8 \\ +0'48(25 & -12 & 0 & | \; 16) \\ \hline 0 & 19'24 & -12 & | \; 15'68 \end{array}$$

$$\left[\begin{array}{ccc|c} 25 & -12 & 0 & 16 \\ 0 & 19'24 & -12 & 15'68 \\ 0 & 0 & 17'5156 & 25'78 \end{array}\right]$$

$X \cdot 19'24 = -12$; $X = -0'6237$

$$\begin{array}{rrr} -12 & 25 & | \; 16 \\ 0'6237(\; 19'24 & -12 & | \; 15'68) \\ \hline 0 & 17'5156 & | \; 25'78. \end{array}$$

$$\boxed{\begin{array}{l} u_3^1 = 1'47 \\ u_2^1 = 1'73 \\ u_1^1 = 1'47 \end{array}} \quad \begin{array}{l} = 25'78/17'5156 \\ = (15'68 + 12 \cdot u_3^1)/19'24 \\ = (16 + 12 \cdot u_2^1)/25 \end{array}$$

✓

Question 2

$$f_{i+1/2} = a_{i+1/2}\,\frac{u_i + u_{i+1}}{2} - b_{i+1/2}\cdot\frac{u_{i+1} - u_i}{h};$$

$$f_{i-1/2} = a_{\underset{i-\frac{1}{2}}{i+1/2}}\,\frac{u_i + u_{i-1}}{2} - b_{\underset{i-\frac{1}{2}}{i+1/2}}\,\frac{u_i - u_{i-1}}{h};$$

1.5

$-u_i + u_{i-1}$

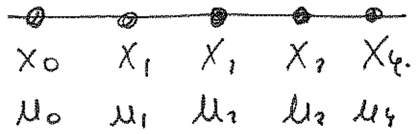
$u_i - u_{i-1}$

$$\frac{f_{i+1/2} - f_{i-1/2}}{h} = 0;$$

$$\frac{a_{i+1/2}}{2h}\left(\cancel{u_i} + u_{i+1} - \cancel{u_i} - u_{i-1}\right) + \underline{\frac{b_{i+1/2}}{h^2}\left(-u_{i+1} + u_i + \underset{\text{no}}{u_i - u_{i-1}}\right)} = 0;$$

$$\frac{a_{i+1/2}}{2h}\left(u_{i+1} - u_{i-1}\right) + \frac{b_{i+1/2}}{h^2}\left(-u_{i+1} + 2u_i - u_{i-1}\right) = 0!$$

$$\boxed{a_{i+1/2}\cdot\frac{u_{i+1} - u_{i-1}}{2h} - b_{i+1/2}\cdot\frac{u_{i+1} - 2u_i + u_{i-1}}{h^2} = 0}$$ scheme.



h = 0'25

$\boxed{i=1}$ $$\underbrace{a_{i+1/2}}_{60}\cdot\frac{u_2 - u_0}{2h} - \underbrace{b_{i+1/2}}_{40}\frac{u_2 - 2u_1 + u_0}{h^2} = 0.$$

$$120u_2 - 120u_0 - 640u_2 + 1280u_1 - 640u_0 = 0$$

$$u_1 - \quad u_2 + 0u_3 =$$

no

FINAL EXAM

STUDENT 1
(EXCEL·LENT)

Question 1.

a) Gaussian quadrature points: n
Order of polynomial integrated exactly: $2n\underline{+}1$ $\quad 2n-1$ $\quad 1$

b) i) Can be integrated exactly $\quad 2n+1=10 \Rightarrow n\geq 4.5 \Rightarrow n=5$ $\quad 6$

ii) Cannot be integrated exactly. cosh is not a polynomial

iii) Can be integrated exactly. $\quad 2n+1=14$, $n\geq 6.5 \Rightarrow n=7$ $\quad 8$ $\quad 2.5$

iv) Can be integrated exactly. $\quad 2n+1=15 \Rightarrow n\geq 7_{8}$

c)
$$\int_0^{\pi}\left(13-8(\cos x)^2\right)dx = \int_{-1}^{1}\frac{\pi}{2}\left(13-8\left(\cos\left[\frac{\pi}{2}x+\frac{\pi}{2}\right]\right)^2\right)dx \qquad 4$$

$$\approx \sum_{i=1}^{2}\frac{\pi}{2}w_i\left(13-8\left(\cos\left[\frac{\pi}{2}\xi_i+\frac{\pi}{2}\right]\right)^2\right)$$

$w_1=w_2=1$
$\xi_1=-\frac{1}{\sqrt{3}} \quad \xi_2=\frac{1}{\sqrt{3}}$

$$\Rightarrow \int_0^{\pi}\left(13-8(\cos x)^2\right)dx \approx \pi\left(8\left(\cos\left[\frac{\pi\sqrt{3}}{6}\right]\right)^2+5\right)\approx 25.2506 \ ✓$$

d)
$$\frac{d^2Y}{dt^2}+\frac{dY}{dt}+4Y^4=0$$

$0\leq t\leq 1$
$Y(0)=0$
$\frac{dY(0)}{dt}=1$

d1) $Y=y_{(1)}$

$\frac{dY}{dt}=y_{(2)}$

$$\frac{dy_{(1)}}{dt}=y_{(2)}$$

$$\frac{dy_{(2)}}{dt}=-y_{(2)}-4y_{(1)}^4$$

$$\Rightarrow \mathbf{y}=\begin{pmatrix}y_{(1)}\\ y_{(2)}\end{pmatrix},\quad \mathbf{f}=\begin{pmatrix}y_{(2)}\\ -y_{(2)}-4y_{(1)}^4\end{pmatrix} \qquad 4$$

for $\frac{d\mathbf{y}}{dt}=\mathbf{f}$, st $\boldsymbol{\alpha}=\mathbf{y}(0)$ ✓

$$\boldsymbol{\alpha}=\begin{pmatrix}0\\ 1\end{pmatrix}$$

d2) using Taylor series (backward)

$$y_{i+1} = y_i + \frac{\Delta t}{1!} f(y,t)_{i+1} + O(\Delta t^2)$$

$$y_i = y_{i+1} - \Delta t \frac{dy_{i+1}}{dt}$$

$$\frac{dy_{i+1}}{dt} = f(y_{i+1}, t)$$

Euler method (backward)

$$y_{i+1} = y_i + \Delta t f(y)_{i+1}$$

$y^{(0)}_{(1)} \to i=0$ (time it)

$$\begin{cases} y^{(1)}_{(1)} - \Delta t\, y^{(1)}_{(2)} = y^{(0)}_{(1)} = 0 \\ y^{(1)}_{(2)} + \Delta t \left(y^{(1)}_{(2)} + 4\left(y^{(1)}_{(1)}\right)^4 \right) = y^{(0)}_{(2)} = 1 \end{cases}$$

5

d3)

- Explicit.

  Advantages: - Direct solving, replacing known values on the equations
  - Good treatment of non-linear problems

6 Disadvantages:
  - Non-stable for big step sizes.
  - Stability depends on the problem and the Δt chosen

- Implicit

  Advantages: - Stable.

  Disadvantages: - Need to solve systems of equations in every step
  - Need linearization for non-linear problems

- Both methods have the same order of accuracy for the same scheme.

Question 3

a) $U_t + f(U)_x = 0 \qquad f(U) = aU \qquad f_{i+1/2} = \begin{cases} au_i & a \geq 0 \\ au_{i+1} & a < 0 \end{cases}$

$U_i^{n+1} = u_i^n - \frac{\Delta t}{h}\left(f_{i+1/2}^n - f_{i-1/2}^n\right)$

It is locally conservative, because independent of the choice of a, the fluxes have the same behavior. (same direction)

b)

i) $a > 0$

$$u_i^{n+1} = u_i^n - \frac{\Delta t}{h}\left(au_i^n - au_{i-1}^n\right)$$

$$u_{i}^{n+1} = u_i^n\left(1 - \frac{a\Delta t}{h}\right) + u_{i-1}^n\left(\frac{a\Delta t}{h}\right)$$

$1 - \frac{a\Delta t}{h}, \frac{a\Delta t}{h} > 0 \qquad \Rightarrow \qquad \frac{a\Delta t}{h} < 1$ ✓

$1 - \frac{a\Delta t}{h} + \frac{a\Delta t}{h} = 1$

ii)

$a < 0$

$$u_i^{n+1} = u_i^n - \frac{\Delta t}{h}\left(au_{i+1}^n - au_i^n\right)$$

$$u_i^{n+1} = u_i^n\left(1 + \frac{a\Delta t}{h}\right) + u_{i+1}^n\left(-\frac{a\Delta t}{h}\right)$$

$1 + \frac{a\Delta t}{h}, -\frac{a\Delta t}{h} > 0 \qquad 1 - \frac{|a|\Delta t}{h} > 0$

$1 + \frac{a\Delta t}{h} - \frac{a\Delta t}{h} = 1 \qquad \Rightarrow \frac{|a|\Delta t}{h} < 1$ ✓

c) $a=10$

$u(0)=1$

$\Delta t=0{,}25$

$h=0{,}25$

$$\frac{a\Delta t}{h}=10>1 \Rightarrow \text{unstable}$$

n \ i	1	2	3	4	5
0	1	0.1	0.1	0.1	0.1
1	1	$\boxed{9.1}$	0.1	0.1	0.1

→ unstability ✓

The result is consistent with the stability analysis.

d)

$$u_i^{n+1}=u_i^n-\frac{\Delta t}{h}\left(f_{i+1/2}^{n+1}-f_{i-1/2}^{n+1}\right)$$

$a=10>0$

$$\Rightarrow u_i^{n+1}=u_i^n-\frac{\Delta t}{h}\left(a u_i^{n+1}-a u_{i-1}^{n+1}\right)$$

$$u_i^{n+1}\left(1+\frac{a\Delta t}{h}\right)+u_{i-1}^{n+1}\left(-\frac{a\Delta t}{h}\right)=u_i^n$$

$$\begin{bmatrix} 1+\frac{a\Delta t}{h} & 0 & 0 & 0 \\ -\frac{a\Delta t}{h} & 1+\frac{a\Delta t}{h} & 0 & 0 \\ 0 & -\frac{a\Delta t}{h} & 1+\frac{a\Delta t}{h} & 0 \\ 0 & 0 & -\frac{a\Delta t}{h} & 1+\frac{a\Delta t}{h} \end{bmatrix} \begin{bmatrix} u_2^{n+1} \\ u_3^{n+1} \\ u_4^{n+1} \\ u_5^{n+1} \end{bmatrix} = \begin{bmatrix} u_2^n+\frac{a\Delta t}{h}u_1^0 \\ u_3^n \\ u_4^n \\ u_5^n \end{bmatrix}$$

n \ i	1	2	3	4	5
0	1	0.1	0.1	0.1	0.1
1	1	0,91818	0,8438	0,77618	0,714712

✓

e) Inside the paper

Question 4.

$$\frac{\partial U}{\partial t} = b\frac{\partial^2 U}{\partial x^2} \qquad \begin{matrix} U(0,t)=1.0 \\ U(1,t)=1.0 \end{matrix} \qquad 0 \le x \le 1$$

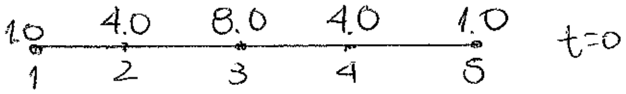
$$u_i^{n+1} - u_i^n = \eta(u_{i+1}^n - 2u_i^n + u_{i-1}^n) \qquad \eta = \frac{\Delta t\, b}{h^2}$$

a)

$b=3$

$\Delta t = 0{,}25$ $\Rightarrow \eta = 12$

$h = 0{,}25$



n \ i	1	2	3	4	5
0	1	4	8	4	1
1	1	16	-88	16	1 ✓

b) $u_i^{n+1} = \eta\, u_{i+1}^n + \eta\, u_{i+1}^n + (1-2\eta)\, u_i^n$

$\eta,\ 1-2\eta > 0 \quad \Rightarrow \quad \eta < \frac{1}{2}$

$\eta + \eta + 1 - 2\eta = 1$ ✓

The calculation on a) is unstable

c) $u_i^{n+1} - u_i^n = \eta(u_{i+1}^{n+1} - 2u_i^{n+1} + u_{i-1}^{n+1})$

Taylor ~~s~~eries for time derivative

$$u_i^{\cancel{n+1}\,n} = u_i^{\cancel{n}\,n+1} + \Delta t\frac{du_i}{dt} + O(\Delta t^2) \Rightarrow \frac{du_i}{dt} = \frac{u_i^{n+1} - u_i^n}{\Delta t} + \tau(\Delta t)$$

Taylor series fo~~s~~r space (backward and forward)

$$u_{i+1}^n = u_i^n + \Delta x\frac{du}{dx} + \frac{\Delta x^2}{2}\frac{d^2u}{dx^2} + O(\Delta x^3) \qquad ①$$

$$u_{i-1}^n = u_i^n - \Delta x\frac{du}{dx} + \frac{\Delta x^2}{2}\frac{d^2u}{dx^2} + O(\Delta x^3) \qquad ②$$

adding ① and ②

$$u_{i+1}^n + u_{i-1}^n = 2u_i^n + \Delta x^2 \frac{d^2u}{dx^2} + O(\Delta x^4)$$

* The sum of the 3rd derivative in ① and ② is also zero, like the 1st one

$$\Rightarrow \frac{d^2u}{dx^2} = \frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{\Delta x^2} + \tau(\Delta x^2)$$

The truncation error is $\tau(\Delta t, \Delta x^2)$

d)

$$u_i^{n+1}(1+2\mu) - \mu u_{i+1}^{n+1} - \mu u_{i-1}^{n+1} = u_i^n$$

$$\begin{bmatrix} 1+2\mu & -\mu & 0 \\ -\mu & 1+2\mu & -\mu \\ 0 & -\mu & 1+2\mu \end{bmatrix} \begin{bmatrix} u_2^{n+1} \\ u_3^{n+1} \\ u_4^{n+1} \end{bmatrix} = \begin{bmatrix} u_2^n + \mu u_1^0 \\ u_3^n \\ u_4^n + \mu u_5^0 \end{bmatrix}$$

e) $b=3$

$\Delta t = 0{,}25$ $\mu = 12$

$h = 0{,}25$

STUDENT 2

FINAL EXAM

(EXCEL·LENT)

a) Using $m+1$ points the quadrature is $q = 2m+1$; using n points ($n = m+1$) the quadrature is $q = 2(n-1)+1$. ✓ 2

b/ Integrals i, iii and iv can be exactly integrated, but not ii

- For $i)$ $q \geq 10 \rightsquigarrow 10 = 2(n-1)+1 \rightsquigarrow n = \frac{10-1}{2} + 1 = 5{,}5 \rightsquigarrow \underline{\underline{n=6}}$
- For $iii)$ $q \geq 14 \rightsquigarrow 14 = 2(n-1)+1 \rightsquigarrow n = \frac{14-1}{2} + 1 = 7{,}5 \rightsquigarrow \underline{\underline{n=8}}$.
- For $iv)$ $q \geq 15 \rightsquigarrow 15 = 2(n-1)+1 \rightsquigarrow n = \frac{15-1}{2} + 1 = \underline{\underline{8}}$ ✓ 4

c/

$$I = \int_{0 \,\rightsquigarrow a}^{\pi \,\rightsquigarrow b} \left(13 - 8(\cos x)^2\right) dx \qquad I \approx \frac{b-a}{2} \sum_{i=1}^{2} w_i f\left(\frac{b-a}{2}\xi_i + \frac{b+a}{2}\right)$$

$$\cancel{I \approx \frac{\pi - 0}{2}\{\xi}$$

4

$$f\left(\frac{b-a}{2}\xi_1 + \frac{b+a}{2}\right) = f\left(\frac{\pi}{2}\cdot\frac{-1}{\sqrt{3}} + \frac{\pi}{2}\right) = f(0{,}6639) = 8{,}0376$$

$$f\left(\frac{b-a}{2}\xi_2 + \frac{b+a}{2}\right) = f\left(\frac{\pi}{2}\cdot\frac{1}{\sqrt{3}} + \frac{\pi}{2}\right) = f(2{,}4777) = 8{,}0375$$

$$I \approx \frac{\pi}{2}\left[1\cdot 8{,}0376 + 1\cdot 8{,}0375\right] = \underline{\underline{25{,}2507}}$$ ✓

d/

d)

$$\vec{Y} := \begin{Bmatrix} y_1 \\ y_2 \end{Bmatrix} = \begin{Bmatrix} Y \\ y_1' \end{Bmatrix} \qquad \frac{d^2 Y}{dt^2} = -4Y^4 - \frac{dY}{dt}$$

$$\vec{y}\,' = \vec{f}(t, \vec{y}) \rightsquigarrow \vec{f}(t, Y) = \begin{Bmatrix} y_2 \\ -4 y_1^4 - y_2 \end{Bmatrix}$$

4

$$\begin{cases} \vec{y}\,' = f(t, \vec{y}) = \begin{Bmatrix} y_2 \\ -4y_1^4 - y_2 \end{Bmatrix} \\ \vec{y}(t) = \begin{Bmatrix} Y \\ y_1' \end{Bmatrix} = \begin{Bmatrix} y_1 \\ y_2 \end{Bmatrix} \\ \vec{y}(0) = \begin{Bmatrix} 0 \\ 1 \end{Bmatrix} \end{cases}$$ ✓

d2/

$$\vec{Y}_{i+1} = \vec{Y}_i + \Delta t \vec{f}(t_{i+1}, Y_{i+1}) \qquad \vec{Y}_i := \begin{Bmatrix} Y_i^{(1)} \\ Y_i^{(2)} \end{Bmatrix}$$

5

$$\vec{Y}_1 = \vec{Y}_0 + \Delta t \cdot \vec{f}(t_1, Y_1) = \begin{Bmatrix} 0 \\ 1 \end{Bmatrix} + \Delta t \cdot \begin{Bmatrix} Y_1^{(2)} \\ -4[Y_1^{(1)}]^4 - Y_1^{(2)} \end{Bmatrix} = \begin{Bmatrix} Y_1^{(1)} \\ Y_2^{(2)} \end{Bmatrix}$$

$$\begin{Bmatrix} 0 \\ 1 \end{Bmatrix} = \begin{Bmatrix} \Delta t \, Y_1^{(2)} \\ -4[Y_1^{(1)}]^4 - Y_1^{(2)} \end{Bmatrix} \rightarrow \left. \begin{array}{c} \Delta t \cdot Y_1^{(2)} = 0 \\ 4[Y_1^{(1)}]^4 + Y_i^{(2)} = -1 \end{array} \right\}$$

✓

d3/

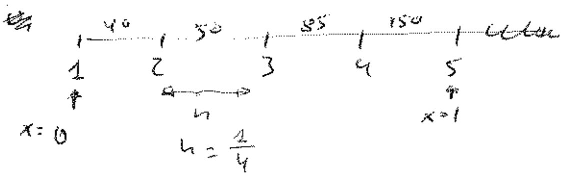
The advantage of the explicit methods is that their computational cost is really low, but the drawback is that they are conditionally stables; step-sizes must be lower than a critical value.

6

The advantage of the implicit methods is that they are unconditionally stables. But they requiere to solve a system of equations in each step. There fore their computational cost is higher than the one for the explicit methods.

Question 2

a)



$$\frac{d}{dx}(aU) - \frac{\partial}{\partial x}\left(b\frac{dU}{dx}\right) = 0$$

$a = 60$ $\qquad u_1 = 3 \qquad u_5 = 15$

6

i = 2

$$\frac{f_{2+1/2} - f_{2-1/2}}{1/4} = 0$$

$$f_{2+1/2} = a_{2+1/2}\frac{u_2+u_3}{2} - b_{2+1/2}\frac{u_3-u_2}{1/4} = 60\frac{u_2+u_3}{2} - 50\frac{u_3-u_2}{1/4}$$

$$f_{2+1/2} = f_{1+1/2} = a_{1+1/2}\frac{u_1+u_2}{2} - b_{1+1/2}\frac{u_2-u_1}{1/4} = 60\frac{u_1+u_2}{2} - 40\frac{u_2-u_1}{1/4}$$

$$\frac{60\frac{u_2+u_3}{2} - 50\frac{u_3-u_2}{1/4} - 60\frac{u_1+u_2}{2} + \frac{40}{1/4}(u_2-u_1)}{1/4} = 0$$

$$\cancel{30u_2} + 30u_3 - \underline{\overset{\frac{50}{1/4}}{12{,}5}}u_3 + \underline{12{,}5}u_2 - 30u_1 - \cancel{30u_2} + \underline{\overset{\frac{40}{1/4}}{10}}u_2 - \underline{10}u_1 = 0$$

$$17{,}5u_3 + 22{,}5u_2 = +40u_1$$

i = 3

$$f_{3+1/2} = a_{3+1/2}\frac{u_3+u_4}{2} - b_{3+1/2}\frac{u_4-u_3}{1/4} = 60\frac{u_3+u_4}{2} - 85\frac{u_4-u_3}{1/4}$$

$$f_{3-1/2} = f_{2+1/2} = 30u_2 + 30u_3 - 12{,}5u_3 + 12{,}5u_2$$

$$\cancel{30u_3} + 30u_4 - 21{,}25u_4 + 21{,}25u_3 - 30u_2 - \cancel{30u_3} + 12{,}5u_3 - 12{,}5u_2$$

$$9{,}75u_4 + 33{,}75u_3 - 42{,}5u_2 = 0$$

i = 4

$$f_{4+1/2} = a_{4+1/2}\frac{u_4+u_5}{2} - b_{4+1/5}\frac{u_5-u_4}{1/4} = 30u_4 + 30u_5 - 37{,}5u_5 + 37{,}5u_4$$

$$f_{4-1/2} = f_{3+1/2} = 30u_3 + 30u_4 - 21{,}25u_4 + 21{,}25u_3$$

$$\cancel{30u_4} + 30u_5 - 37{,}5u_5 + 37{,}5u_4 - 30u_3 - \cancel{30u_4} + 21{,}25u_4 - 21{,}25u_3 = 0$$

$$58{,}75u_4 - 51{,}25u_3 = 7{,}5u_5$$

$$17,5 u_3 + 22,5 u_2 = 40 u_1 = 120$$

$$-42,5 u_2 + 33,75 u_3 + 9,75 u_4 = 0$$

$$-51,25 u_3 + 58,75 u_4 = 7,5 u_5 = 112,5$$

b/

$$\begin{pmatrix} 1440 & -680 & 0 \\ -920 & 2160 & -1240 \\ 0 & -1480 & 3760 \end{pmatrix} \begin{pmatrix} u_2 \\ u_3 \\ u_4 \end{pmatrix} = \begin{pmatrix} 2280 \\ 0 \\ 34200 \end{pmatrix} \qquad \bar{u}^0 = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

0

$$x_1^1 1440 - 680 x_2^0 = 2280 \rightsquigarrow x_1^1 = \frac{2280}{1440} = 1,58\hat{3}$$

$$-x_1^0 920 + 2160 x_2^1 - 1240 x_3^0 = 0 \rightsquigarrow x_2^1 = 0$$

$$-1480 x_2^0 + 3760 x_3^1 = 34200 \rightsquigarrow x_3^1 = \frac{34200}{3760} = 9,0957$$

2

$$\bar{x}^1 = \begin{pmatrix} 1,58\hat{3} \\ 0 \\ 9,0957 \end{pmatrix}$$

$$x_1^2 1440 - 680 x_2^1 = 2280 \rightsquigarrow x_1^2 = \frac{2280 + 680 \cdot 0}{1440} = 1,58\hat{3}$$

$$-x_2^1 920 + 2160 x_2^2 - 1240 x_3^1 = 0 \rightsquigarrow x_2^2 = \frac{1240 \cdot 9,0957 + 920 \cdot 1,58\hat{3}}{2160} = 5,896$$

$$-1480 x_2^1 + 3760 x_3^2 = 34200 \rightsquigarrow x_3^2 = \frac{34200}{3760} = 9,0957$$

$$\bar{x}^2 = \begin{pmatrix} 1,58\hat{3} \\ 5,896 \\ 9,0957 \end{pmatrix}$$

2

c/

$$x_1^1\, 1440 - 680\, x_2^0 = 2280 \rightsquigarrow x_1^1 = 1{,}58\overline{3}$$

$$-920\, x_1^1 + 2160\, x_2^1 - 1240\, x_3^0 = 0 \rightsquigarrow x_2^1 = \frac{920 \cdot 1{,}58\overline{3}}{2160} = 0{,}6744$$

2

$$-1480\, x_2^1 + 3760\, x_3^1 = 34200 \rightsquigarrow x_3^1 = \frac{34200 + 1480 \cdot 0{,}6744}{3760} = 9{,}3612$$

$$x_1^2\, 1440 - 680\, x_2^1 = 2280 \rightsquigarrow x_1^2 = \frac{2280 + 680 \cdot 0{,}6744}{1440} = 1{,}9018$$

$$-920\, x_1^2 + 2160\, x_2^2 - 1240\, x_3^1 = 0 \rightsquigarrow x_2^2 = \frac{1240 \cdot 9{,}3612 + 920 \cdot 1{,}9018}{2160} = 6{,}2369$$

$$-1480\, x_2^2 + 3760\, x_3^1 = 34200$$

$$x_3^2 = \frac{34200 + 1480 \cdot 6{,}2369}{3760} = 11{,}5507$$

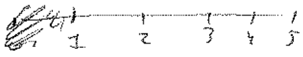
$$\bar{x}^1 = \begin{pmatrix} 1{,}58\overline{3} \\ 0{,}6744 \\ 9{,}3612 \end{pmatrix} \qquad \bar{x}^2 = \begin{pmatrix} 1{,}9018 \\ 6{,}2369 \\ 11{,}5507 \end{pmatrix}$$

2

d/ Apart from converging when $\underline{\underline{A}}$ is diagonaly dominant Gauss-Seidel also does so, when $\underline{\underline{A}}$ is symmetric positive difinite. (Jacobi does not do this)
When both methods converge, Gauss-Seidel converges faster than Jacobi. why?

1.5

QUESTION 4



$v_1 = 1$

$v_5 = 1$

$h = \frac{1}{4}$

a/ $\mu = \frac{0.25 \cdot 3}{1/16} = 12$

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

$U_1^1 = 1$

$U_2^1 = 12 (U_3^0 - 2U_2^0 + U_1^0) + U_2^0 = 12(8 - 2 \cdot 4 + 1) + 4 = 12 + 4 = 16$

$U_3^1 = 12 (U_4^0 - 2U_3^0 + U_2^0) + U_3^0 = 12(4 - 2 \cdot 8 + 4) + 8 = -96 + 8 = -88$

$U_4^1 = 12 (U_5^0 - 2U_4^0 + U_3^0) + U_4^0 = 12(1 - 2 \cdot 4 + 8) + 4) = 12 + 4 = 16$ ✓

$U_5^1 = 1$

b/ The explicit FTCS is stable when $\mu \leq \frac{1}{2}$. Clearly the above μ, makes the method unstable. ✓

b'/

$$U_i^{n+1} - U_i^n - \mu (U_{i+1}^{n+1} - 2U_i^{n+1} + U_{i-1}^{n+1}) = 0$$

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

$$U_i^n = U_i^{n+1} - \Delta t \frac{\partial U_i^{n+1}}{\partial t} + \frac{\Delta t^2}{2} \frac{\partial^2 U_i^{n+1}}{\partial t^2} - O(\Delta t^3)$$

$$U_{i+1}^{n+1} = U_i^{n+1} + \Delta x \frac{\partial U_i^{n+1}}{\partial x} + \frac{\Delta x^2}{2} \frac{\partial^2 U_i^{n+1}}{\partial x^2} + \frac{\Delta x^3}{6} \frac{\partial^3 U_i^{n+1}}{\partial x^3} + \frac{\Delta x^4}{24} \frac{\partial^4 U_i^{n+1}}{\partial x^4} + O(\Delta x^5)$$

$$U_{i-1}^{n+1} = U_i^{n+1} - \Delta x \frac{\partial U_i}{\partial x} + \frac{\Delta x^2}{2} \frac{\partial^2 U_i^{n+1}}{\partial x^2} - \frac{\Delta x^3}{6} \frac{\partial^3 U_i^{n+1}}{\partial x^3} + \frac{\Delta x^4}{24} \frac{\partial^4 U_i^{n+1}}{\partial x^4} - O(\Delta x^5)$$

ok

$$\Delta t \frac{\partial U_i^{n+1}}{\partial t} + \frac{\Delta t^2}{2} \frac{\partial^2 U_i^{n+1}}{\partial t^2} - O(\Delta t^3) - \underset{\downarrow}{\mu} \left(\Delta x^2 \frac{\partial^2 U_i^{n+1}}{\partial x^2} + \frac{\Delta x^4}{12} \frac{\partial^4 U_i^{n+1}}{\partial x^4} + O(\Delta x^6) \right) = \tau$$

$$\mu = \frac{\Delta t}{\Delta x^2} b$$

$$\cancel{\Delta t \frac{\partial U_i^{n+1}}{\partial t}} + \frac{\Delta t^2}{2} \frac{\partial^2 U_i^{n+1}}{\partial t^2} - O(\Delta t^3) - \cancel{b \Delta t \frac{\partial^2 U_i^{n+1}}{\partial x^2}} - \frac{b \Delta t}{12} \Delta x^2 \frac{\partial^4 U_i^{n+1}}{\partial x^4} - O(\Delta x^6)$$

$$\tau = -\frac{\Delta t}{2} \frac{\partial^2 U_i}{\partial t^2} \Bigg|_i^{n+1} - \frac{b}{12} \Delta x^2 \frac{\partial^4 U_i^{n+1}}{\partial x^4} + O(\Delta x^4) - O(\Delta t^2)$$ ✓

QUESTION 4

c/

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

$$\begin{bmatrix} 1+2\mu & -\mu & 0 \\ -\mu & 1+2\mu & -\mu \\ 0 & -\mu & 1+2\mu \end{bmatrix} \begin{bmatrix} U_2^{n+1} \\ U_3^{n+1} \\ U_4^{n+1} \end{bmatrix} = \begin{bmatrix} U_i^n \\ U_3^n \\ U_4^n \end{bmatrix} + \begin{bmatrix} \mu U_1^{n+1} \\ 0 \\ \mu U_5^{n+1} \end{bmatrix}$$

d/ $\mu = 12$

$$\begin{bmatrix} 25 & -12 & 0 \\ -12 & 25 & -12 \\ 0 & -12 & 25 \end{bmatrix} \begin{bmatrix} U_2^{n+1} \\ U_3^{n+1} \\ U_4^{n+1} \end{bmatrix} = \begin{pmatrix} 4+12\cdot 1 \\ 0 \\ 4+12\cdot 1 \end{pmatrix}$$

$$\left(\begin{array}{ccc|c} 25 & -12 & 0 & 16 \\ -12 & 25 & -12 & 0 \\ 0 & -12 & 25 & 16 \end{array}\right) \rightarrow \left(\begin{array}{ccc|c} 25 & -12 & 0 & 16 \\ 0 & 0 & -12 & 33{,}\hat{3} \\ 0 & -12 & 25 & 16 \end{array}\right) \sim \left(\begin{array}{ccc|c} 25 & -12 & 0 & 16 \\ 0 & -12 & 25 & 16 \\ 0 & 0 & -12 & 33{,}\hat{3} \end{array}\right) \begin{matrix} \rightarrow U_2 \\ \rightarrow U_3 \\ \rightarrow U_3 \end{matrix}$$

$$U_3^2 = \cancel{\frac{33{,}3}{-12}}$$