CALCOLO SCIENTIFICO

Project 1

Student
Pranav KASELA
Roll no. 866261

April 11, 2018

Contents

Exercise 1

We will start by studying the equation $\Delta p = f(\ell, D, V, \mu)$ where Δp is the pressure drop measured in Newton/meters^2 with the basic dimension M/(T^2·L)(Mass/Time^2·Length), ℓ is the length of the tube and is measured in meters(m) and it's basic dimension is L(length), D is the diameter of the tube measured in meters(m) with basic dimension is L(length), V, which is the fluid velocity, has unit : meters/second(m/s) with the basic dimension L/T(Length/Time) and μ is the viscosity measured in Newton·second/meters^2 (N·s/m^2) it's and basic dimension is M/(T·L)(Mass/Time·Lenght) since N=kg·m/s^2 so

$$\frac{N \cdot s}{m^2} = \frac{kg \cdot \cancel{m} \cdot \cancel{s}}{m^{\cancel{2}} \cdot s^{\cancel{2}}} = \frac{kg}{m \cdot s}$$

we summarise all this in the following table :

Variable	ℓ	D	V	μ	Δp
I.S. Units	m	m	m/s	kg/(m·s)	kg/(m·s^2)
Basic Dimension	L	L	L/T	M/(T·L)	M/(T^2·L)

Since Δp depends upon 4 variables when we will try to find the law f experimentally we will have to do x experiments for every variable so in total x^4 experiments. To reduce the number of experiments let's to a a-dimensionalization process, for this we will consider the following unitless group:

$$\Pi_1 = D^{a_1} V^{b_1} \mu^{c_1} \Delta p$$
$$\Pi_2 = D^{a_2} V^{b_2} \mu^{c_2} \ell$$

Since the number of variables is five and there are three fundamental units involved so for π-Buckingham theorem we must find two independent unitless group.
Firstly let's compute the coefficients of the unitless group (we will indicate with $[x]$ the basic dimension of the unit) :

$$[\Pi_1] = [D^{a_1} V^{b_1} \mu^{c_1} \Delta p] = [L^{a_1} \frac{L^{b_1}}{T^{b_1}} \frac{M^{c_1}}{T^{c_1} L^{c_1}} \frac{M}{T^2 L}] = [M^{c_1+1} L^{a_1+b_1-c_1-1} T^{-b_1-c_1-2}] = [M^0 L^0 T^0]$$

$$[\Pi_2] = [D^{a_2} V^{b_2} \mu^{c_2} \ell] = [L^{a_2} \frac{L^{b_2}}{T^{b_2}} \frac{M^{c_2}}{T^{c_2} L^{c_2}} L] = [M^{c_2} L^{a_2+b_2-c_2+1} T^{-b_2-c_2}] = [M^0 L^0 T^0]$$

so we have for Π_1

$$\begin{cases} a_1 + b_1 - c_1 - 1 = 0 \\ -b_1 - c_1 - 2 = 0 \\ c_1 + 1 = 0 \end{cases}$$

which has solution :

$$\begin{cases} a_1 = 1 \\ b_1 = -1 \\ c_1 = -1 \end{cases}$$

for Π_2 we have

$$\begin{cases} a_2 + b_2 - c_2 + 1 = 0 \\ -b_2 - c_2 = 0 \\ c_2 = 0 \end{cases}$$

which has solution :

$$\begin{cases} a_2 = -1 \\ b_2 = 0 \\ c_2 = 0 \end{cases}$$

It is clear that Π_1 and Π_2 are two independent unitless group since one contain the variable ℓ but the other doesn't. Thanks to the π Buckingham theorem we have that

$$\Pi_1 = f(\Pi_2)$$

so

$$\frac{D \cdot \Delta p}{V \cdot \mu} = f(\frac{\ell}{D}) \Rightarrow \Delta p = \frac{V \cdot \mu}{D} \cdot f(\frac{\ell}{D})$$

Now we will need to do less experiments based only on the ratio of ℓ and D. Finally we observe that the pressure drop is directly proportional to the fluid velocity and fluid viscosity which was to be expected since more viscous and fast fluid has a faster pressure drop but with the a-dimensional analysis we obtained also that the dependence of Δp from V and μ is linear. We note that Δp decreases as the diameter of the tube increases. Another thing to be noted is that intuitively $f(\ell/D)$ is not constant because if we fix all the variable and change only ℓ the Δp will change too, for example if the we have two tubes with the same diameter and ℓ_1 length of the first tube and ℓ_2 length of the second tube with $\ell_1 \geq \ell_2$ let's assume that we have the same kind of fluid inside and at the same velocity then $\Delta p_1 \geq \Delta p_2$.
The law $f(\ell/D)$ is to be found either experimentally or with computations so it's still unknown. But now we have a better understanding of the law $\Delta p = f(\ell, D, V, \mu)$ especially the kind of proportionality between the variables and for some variables we know even the exact proportionality.

Exercise 2

Given the following non linear problem

$$\begin{cases} y_1'(t) = y_2(t), & y_1(0) = 1/2 \\ y_2'(t) = y_2(t)(y_2(t) - 1)/y_1(t), & y_2(0) = -3 \end{cases}$$

We resolve it by doing

$$dy_1/dy_2 = y_1/y_2 \leftrightarrow dy_1/y_1 = dy_2/(y_2 - 1)$$

Integrating on both sides and due to the initial condition(ICs) we have

$$ln(y_1 \cdot 2) = ln((1 - y_2)/4)$$

thus

$$y_1 = (y_2 - 1)/(-8)$$

We substitute it in the second equation obtaining $y_2'(t) = -8 \cdot y_2(t)$ which has the solution with the ICs

$$y_2(t) = -3e^{-8t}$$

Using this solution and substituting it in the first equation with its ICs we find

$$y_1'(t) = -3e^{-8t} \leftrightarrow y_1(t) = (3/8)e^{-8t} + 1/8$$

so the two exact solutions are

$$y_1(t) = (3/8)e^{-8t} + 1/8$$
$$y_2(t) = -3e^{-8t}$$

Method M1

We study now the numerical methods to resolve the ODEs. Let's start with the M1 method, which is a 2-step implicit method:

$$u_{n+2} + u_{n+1} - 2u_n = \frac{h}{4}[f(t_{n+2}, u_{n+2}) + 8f(t_{n+1}, u_{n+1}) + 3f(t_n, u_n)]$$

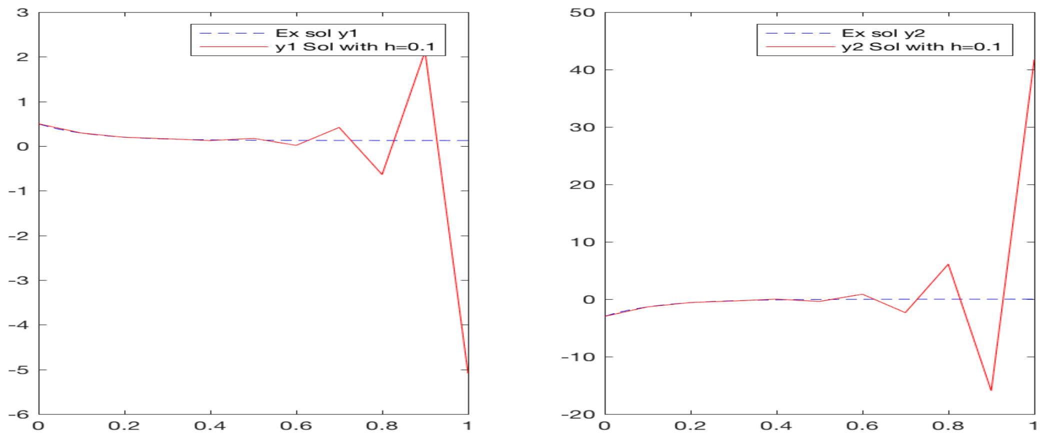


Figure 1: Plot of the exact solution and the solution with M1 method and h=0.1

We will start by studying the consistency of the method, in general given

$$\sum_{i=0}^{k} \alpha_i u_i = \Phi(...)$$

we know that a method is consistent iff using the characteristic polynomial

$$\rho(\zeta) = \sum_{i=0}^{k} \alpha_i \zeta^i$$

these two condition are verified :

$$\rho(1) = 0 \qquad (1)$$

$$\Phi(...)/\rho'(1) = f(t_n, y(t_n)) \quad (2)$$

In our case

$$\rho(\zeta) = \zeta^2 + \zeta - 2 \Rightarrow \rho(1) = 0, \qquad \rho'(\zeta) = 2\zeta + 1 \Rightarrow \rho'(1) = 3$$

then

$$\Phi(f_n, f_n, f_n, t_n) = (1/4)(f_n + 8f_n + 3f_n) = 3f_n \Rightarrow \frac{\Phi(...)}{\rho'(1)} = \frac{\cancel{3}f_n}{\cancel{3}} = f_n$$

which is the condition (1) and (2) so the method M1 is consistent. For the zero stability we use the root condition the roots of $\rho(\zeta)$ are : -2 and 1 since the $|-2| > 1$ we have that the method is not zero stable. The fact that the method is not zero stable is shown in the Figure 1 that we obtained implementing the method on MATLAB.
We can observe that the solution is starting to grow unbounded, that is because the root -2 makes the function grow unbounded.
In Figure 2 we can observe that as h decreases the solution grows even more unbounded.

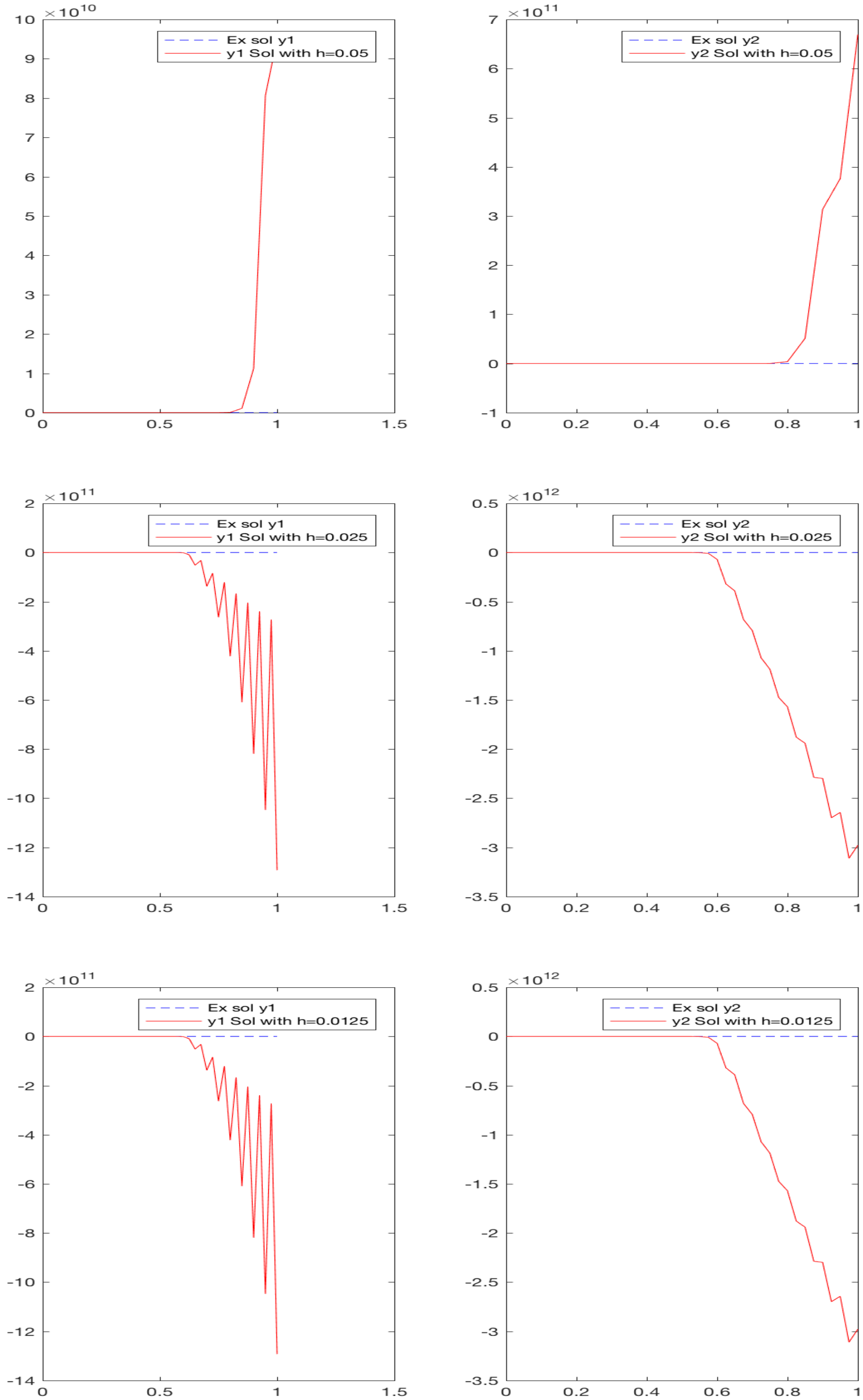


Figure 2: Plot of the exact solution and the solution with M1 method and h=[0.05,0.025,0.0125]

Method M2

The method M2 is a 2-step explicit method :

$$u_{n+2} - u_{n+1} = \frac{h}{3}[3f(t_{n+1}, u_{n+1}) - 2f(t_n, u_n)]$$

Checking the consistency of the method the first one (1) is easy since

$$\rho(\zeta) = \zeta^2 - \zeta \Rightarrow \rho(1) = 0$$

next for (2) we have

$$\Phi = \frac{1}{3}[3f(t_n, y_n) - 2f(t_n, y_n)] = \frac{1}{3}f(t_n, y_n)$$

and

$$\rho'(\zeta) = 2\zeta - 1 \Rightarrow \rho'(1) = 2 - 1 = 1$$

so

$$\frac{\Phi}{\rho'(1)} = \frac{1}{3}f(t_n, y_n) \neq f(t_n, y_n)$$

Note : a method can be convergent even if it is not consistent(the problem is non linear), se let's check to stability using the root condition :

$$\rho(\zeta) = \zeta^2 - \zeta = 0 \Leftrightarrow \zeta = 0, \zeta = 1$$

so the spurious root is 0 and the principal root is 1 and since they lie in the unitary circle the method is zero stable.
Since for the consistency (2) we have that

$$\frac{\Phi}{\rho'(1)} = \frac{1}{3}f(t_n, y_n) \neq f(t_n, y_n)$$

It means that that the method will converge on a different ODE (see Figure 3) which is of the form($y' = f/3$):

$$\begin{cases} y_1'(t) = y_2(t)/3, & y_1(0) = 1/2 \\ y_2'(t) = y_2(t)(y_2(t) - 1)/(3y_1(t)), & y_2(0) = -3 \end{cases}$$

It is easy to check using the same calculations as above that the exact solution of this problem is

$$y_1(t) = (3/8)e^{-8t/3} + 1/8$$

$$y_2(t) = -3e^{-8t/3}$$

Note : the only difference is the /3 in the exponential argument.

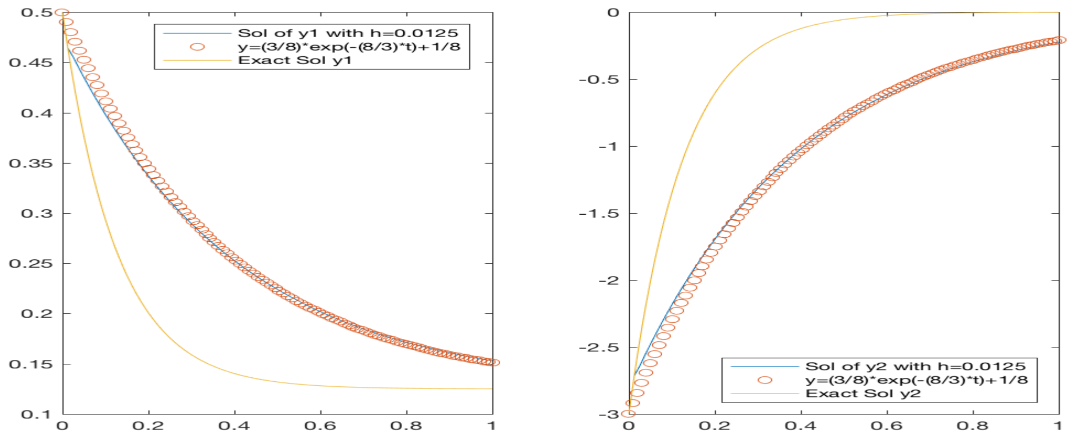


Figure 3: Plot of the solution with M2 method and h=0.0125

In Figure 3 we can see the convergence of the method to the solution we mentioned above. One interesting thing that can be done is to obtain the approximation of our ODEs is to multiply it by 3 and then solve it, so we will solve :

$$\begin{cases} y_1'(t) = 3 \cdot y_2(t), & y_1(0) = 1/2 \\ y_2'(t) = 3 \cdot y_2(t)(y_2(t) - 1)/y_1(t), & y_2(0) = -3 \end{cases}$$

Using the same method but on the new problem we obtain the desired approximation in Figure 4:

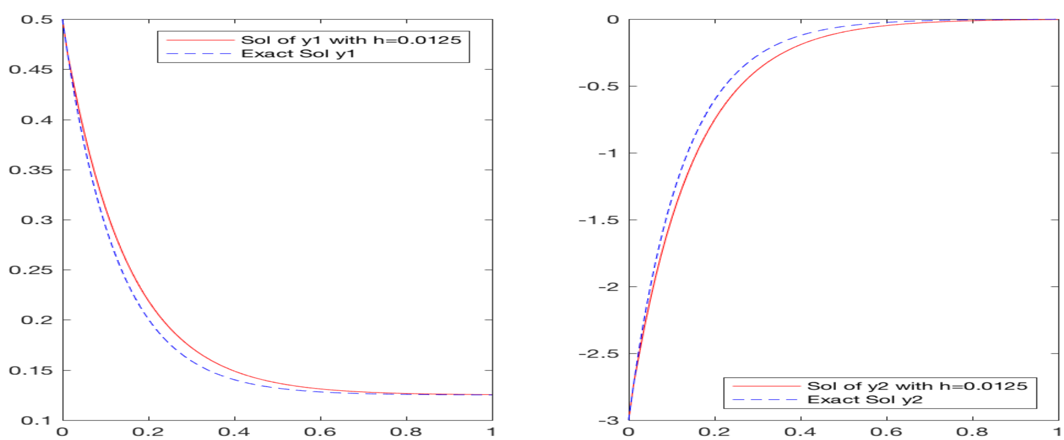


Figure 4: Plot of the exact solution and the solution with M2 method on new ODE and h=0.0125

In Figure 5 we put the approximation for all the other h requested.

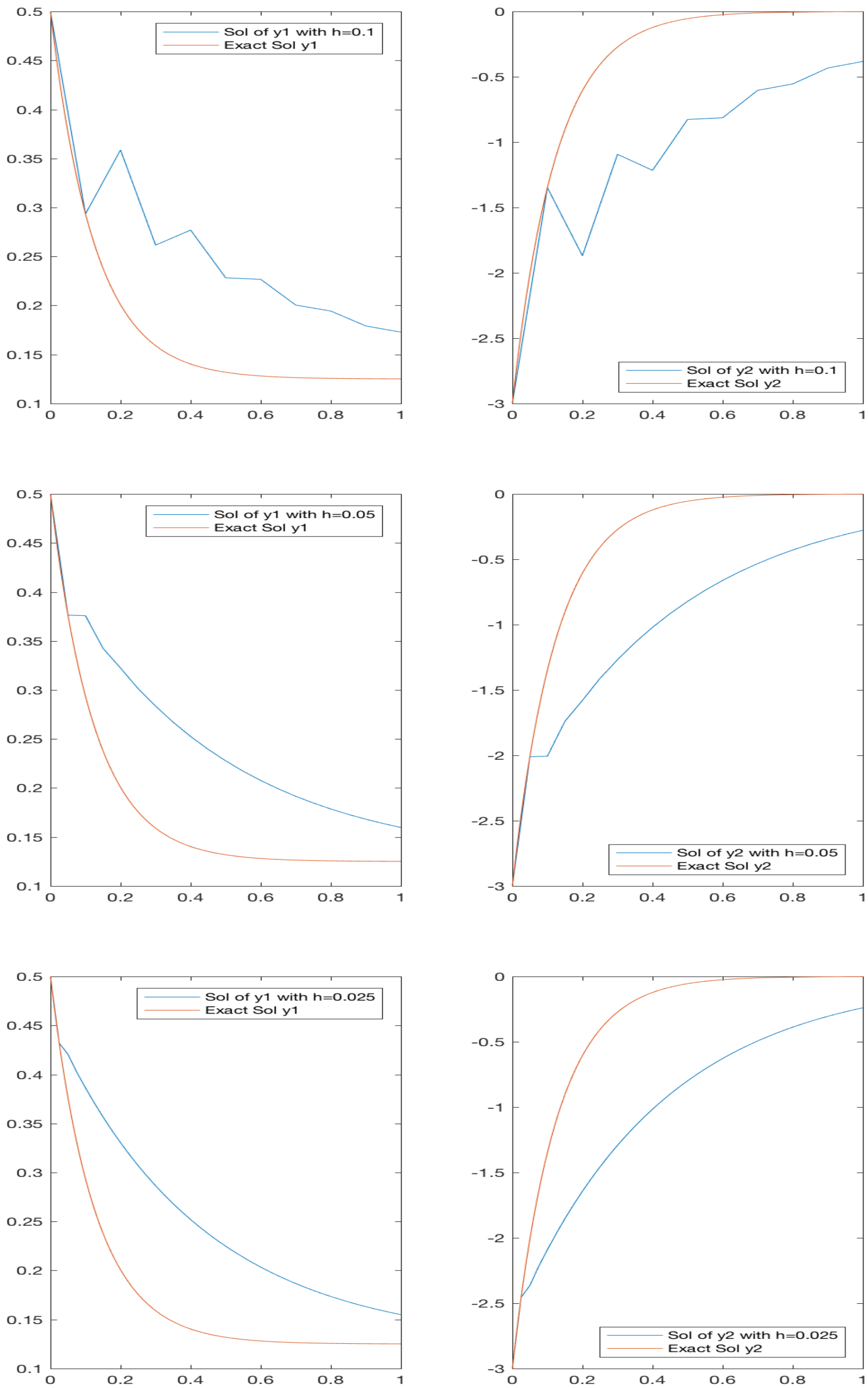


Figure 5: Plot of the exact solution and the solution with M2 method and h=[0.1,0.05,0.025]

Method M3

The method M3 is a 3-step explicit method

$$u_{n+3}+\frac{1}{4}u_{n+2}-\frac{1}{2}u_{n+1}-\frac{3}{4}u_n=\frac{h}{8}[19f(t_{n+1},u_{n+1})+5f(t_n,u_n)]$$

We start by studying it's behaviour theoretically. We consider

$$\rho(\zeta)=\zeta^3+\frac{1}{4}\zeta^2-\frac{1}{2}\zeta-\frac{3}{4}$$

It is clear that $\rho(1)=0$ so the condition (1) for consistency, on the other hand we have

$$\rho'(\zeta)=3\zeta^2+\frac{1}{2}\zeta-\frac{1}{2}$$

so

$$\frac{\Phi}{\rho'(1)}=\frac{\cancel{3}f(t_n,u_n)}{\cancel{3}}=f(t_n,u_n)$$

thus the method is consistent. For the stability we use always the root condition :

$$0=\rho(\zeta)=\zeta^3+\frac{1}{4}\zeta^2-\frac{1}{2}\zeta-\frac{3}{4}=4\zeta^3+\zeta^2-2\zeta-3=(\zeta-1)(4\zeta^2+5\zeta+3)$$

with simple computation we have that the roots are :

$$\zeta_1=1,\ \zeta_2=\frac{-5+i\sqrt{23}}{8},\ \zeta_3=\frac{-5-i\sqrt{23}}{8}$$

where ζ_1 is the principal root and ζ_2 and ζ_3 are spurious root and $|\zeta_2|=|\zeta_3|=\frac{\sqrt{3}}{2}<1$ so the method is zero stable. Thus we conclude that the method is convergent.

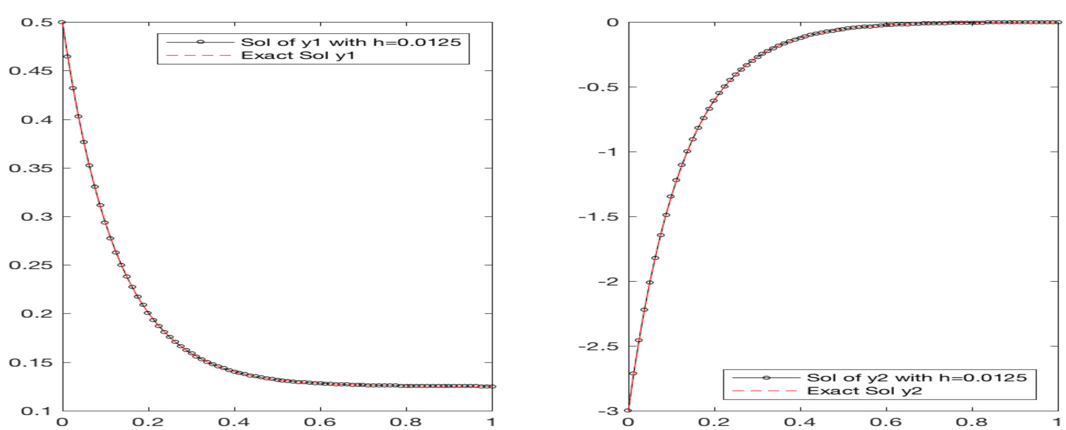


Figure 6: Plot of the exact solution and the solution with M3 method and h=0.0125

Note : the method M3 does not seem to converge for h = 0.1 and 0.05. It may due to the dependence of the method from 3 points before the actual point of interest so if the points are not near enough the error will continuously increase further we go or the absolute stability.

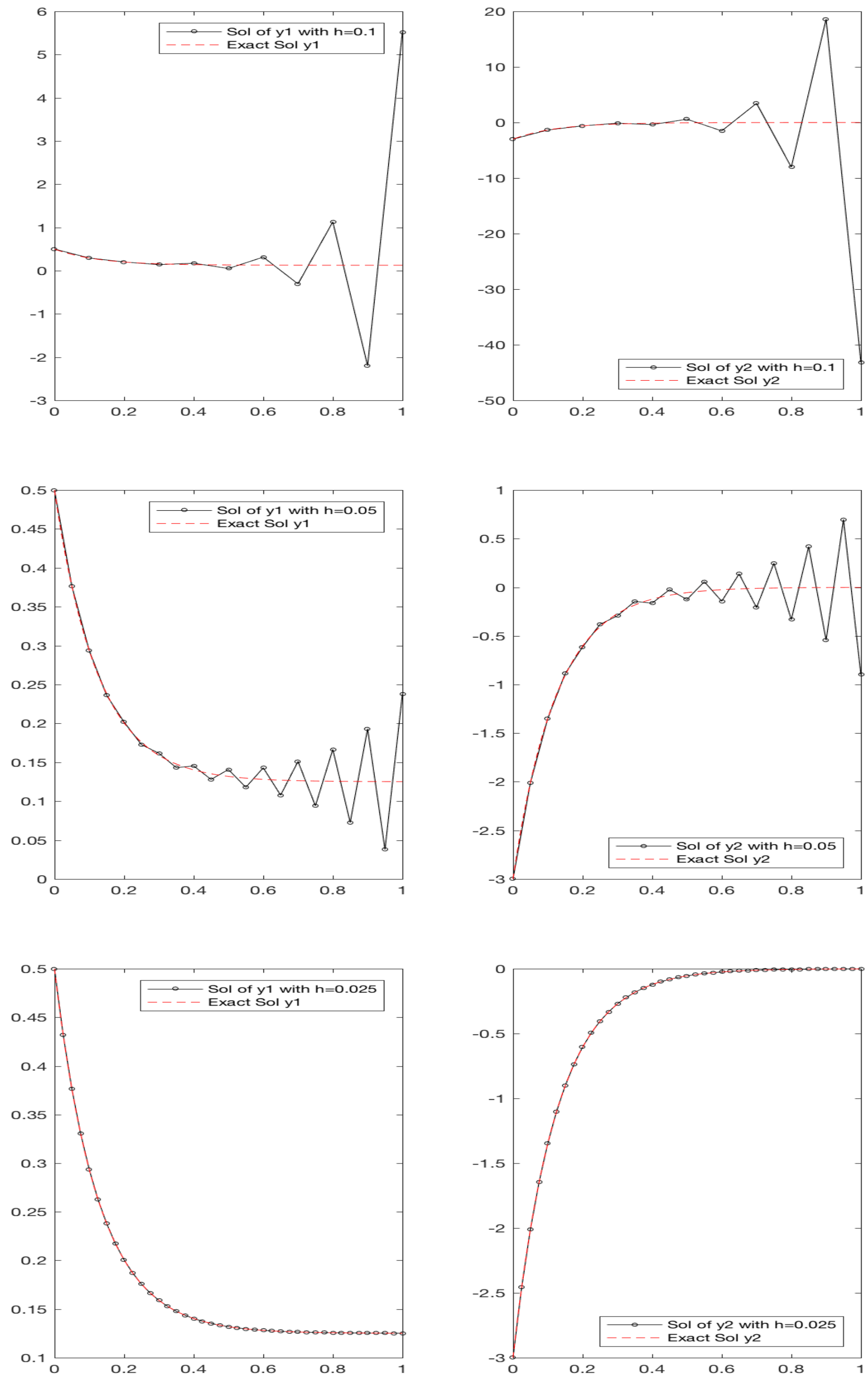


Figure 7: Plot of the exact solution and the solution with M3 method and h=[0.1,0.05,0.025]

Method M4

The M4 method is a RK method with 3 stages :

$$u_{n+1} - u_n = \frac{h}{4}(K1 + K3)$$

with

$$K1 = f(t_n, u_n)$$
$$K2 = f(t_n + \frac{1}{3}h, u_n + \frac{1}{3}hK1)$$
$$K3 = f(t_n + \frac{2}{3}h, u_n + \frac{2}{3}hK2)$$

For this RK method the Butcher's array is

$$\begin{array}{c|ccc} 0 & & & \\ \frac{1}{3} & \frac{1}{3} & & \\ \frac{2}{3} & 0 & \frac{2}{3} & \\ \hline & \frac{1}{4} & 0 & \frac{1}{4} \end{array}$$

Since the diagonal and upper triangular matrix is zero, the method is explicit.

For a RK method with s stages

$$u_{n+1} = u_n + h\sum_{i=0}^{s} b_i \cdot K_i$$

we have that it is consistent iff $\sum_{i=0}^{s} b_i = 1$ in our case

$$\sum_{i=0}^{s} b_i = \frac{1}{4} + \frac{1}{4} = \frac{1}{2}$$

so the method is not consistent, with with similar argument as in method M2 the method M4 converges to a different ODE ($y' = f/2$ instead of $y' = f$)

$$\begin{cases} y_1'(t) = y_2(t)/2, & y_1(0) = 1/2 \\ y_2'(t) = y_2(t)(y_2(t) - 1)/(2y_1(t)), & y_2(0) = -3 \end{cases}$$

which has solution

$$y_1(t) = (3/8)e^{-4t} + 1/8$$
$$y_2(t) = -3e^{-4t}$$

Let's check it through MATLAB (see Figure 8) :

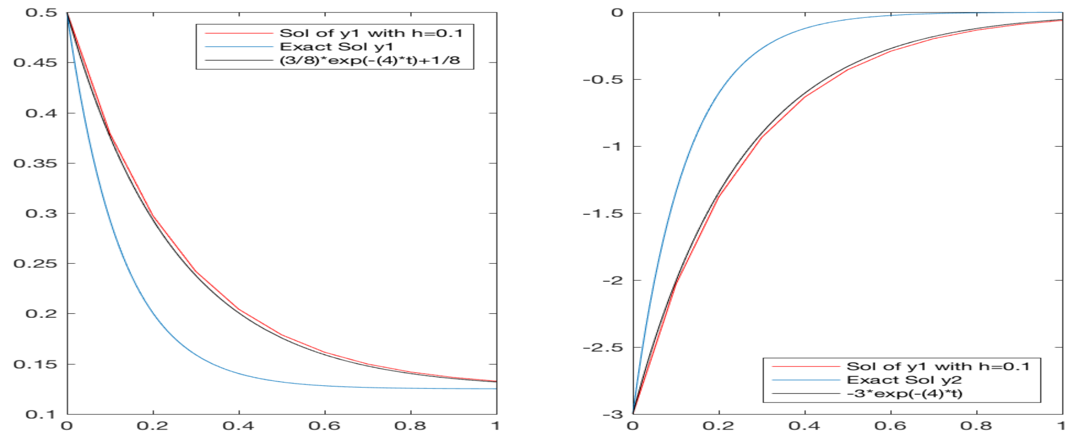


Figure 8: Plot of the exact solution and the solution with M4 method on another ODE and h=0.1

One way to get the solution to our problem is to apply the method on the following new ODE :

$$\begin{cases} y_1'(t) = 2y_2(t), & y_1(0) = 1/2 \\ y_2'(t) = 2y_2(t)(y_2(t) - 1)/y_1(t), & y_2(0) = -3 \end{cases}$$

This way the problem converges to $y' = f/2$ which is our ODE(see Figure 9)

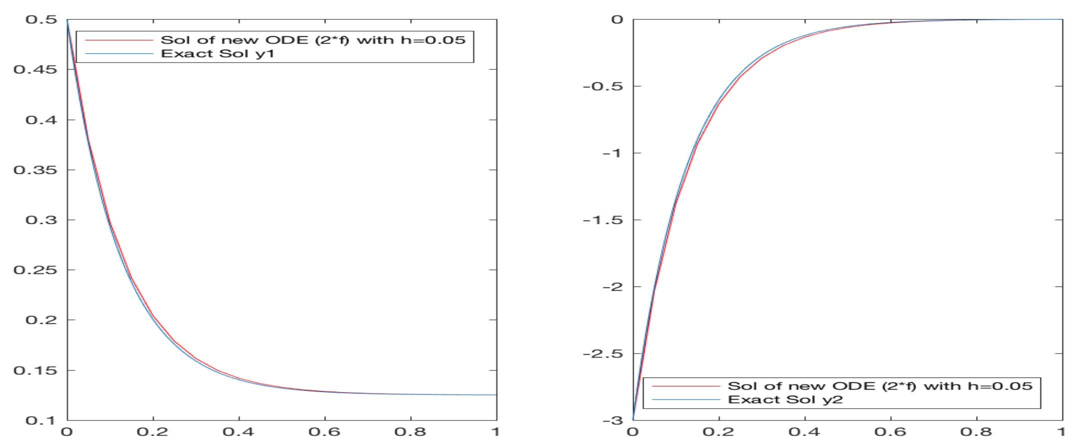


Figure 9: Plot of the exact solution and the solution with M4 method on new ODE and h=0.05

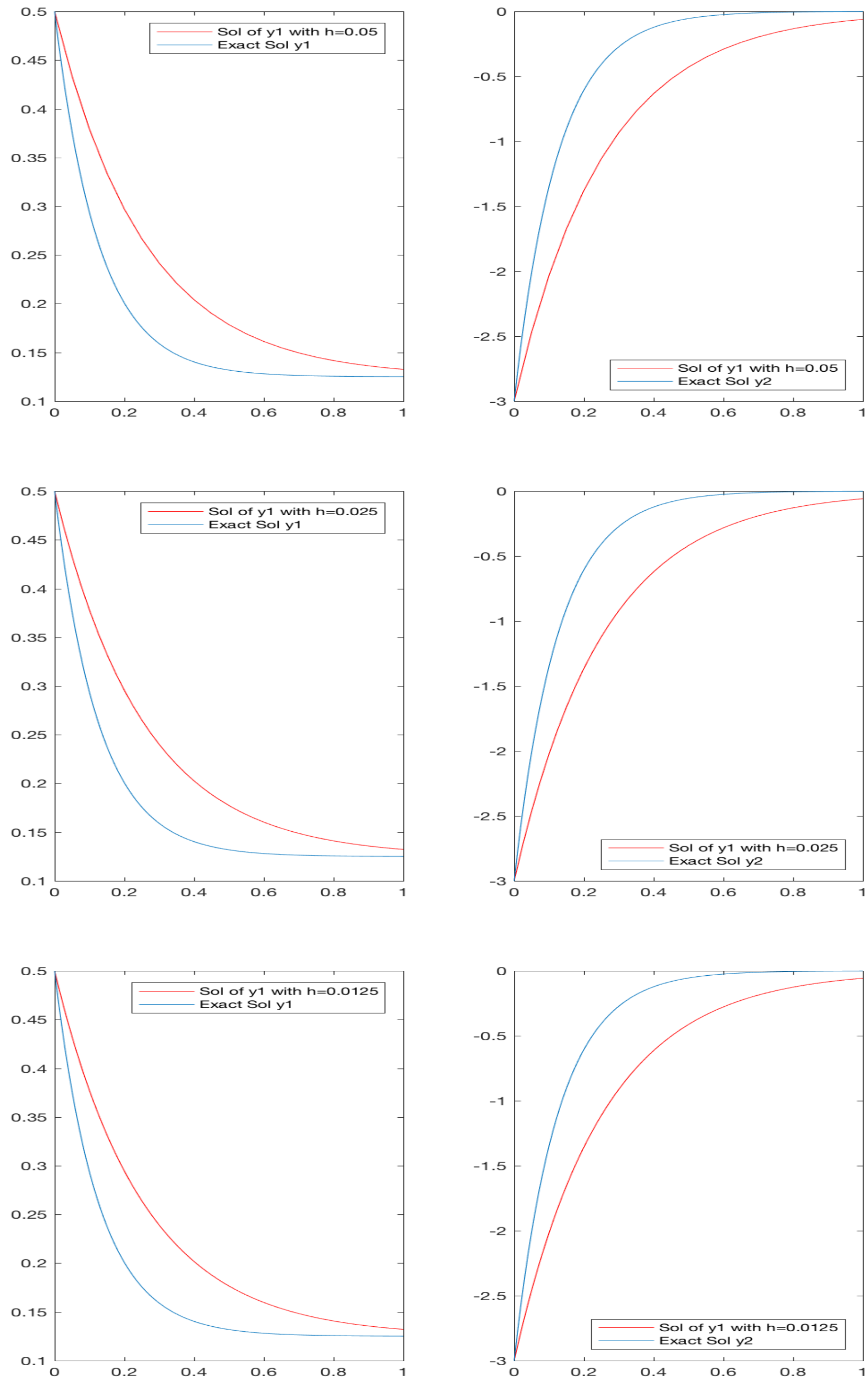


Figure 10: Plot of the exact solution and the solution with M4 method and h=[0.05,0.025,0.0125]

Errors of the Methods

Now we put in Table 1 all the $e_n = ||y(t_n) - u_n||_2$ at $t_n = [0.2, 0.4, 0.6, 0.8, 1.0]$ for all 4 methods with all the required value of h. In the first column there are the values of h and in the first row the values of t:

Table 1: e_n for the M1, M2, M3, M4 methods in the respective order

h/t	0.2	0.4	0.6	0.8	1.0
0.1	0.0265	0.1350	0.9025	6.1568	42.04
0.05	0.0082	0.2085	$1.224 \cdot 10^{3}$	$3.6744 \cdot 10^{9}$	$6.7590 \cdot 10^{11}$
0.025	0.0090	4.0812	$6.9872 \cdot 10^{10}$	$1.6261 \cdot 10^{12}$	$3.2474 \cdot 10^{12}$
0.0125	0.1546	$3.5295 \cdot 10^{11}$	$3.1750 \cdot 10^{12}$	$5.1465 \cdot 10^{12}$	$5.1465 \cdot 10^{12}$

h/t	0.2	0.4	0.6	0.8	1.0
0.1	1.2737	1.1019	0.7950	0.5538	0.3842
0.05	0.9811	0.9052	0.6416	0.4268	0.2788
0.025	1.0459	0.9002	0.6076	0.3858	0.2405
0.0125	1.0984	0.9060	0.5951	0.3684	0.2239

h/t	0.2	0.4	0.6	0.8	1.0
0.1	0	0.2578	1.4975	8.0876	43.5066
0.05	0.0084	0.0411	0.1197	0.3301	0.9051
0.025	$9.3397 \cdot 10^{-4}$	$2.4341 \cdot 10^{-4}$	$1.5586 \cdot 10^{-4}$	$2.1384 \cdot 10^{-5}$	$6.7618 \cdot 10^{-6}$
0.0125	$1.1135 \cdot 10^{-4}$	$4.4996 \cdot 10^{-5}$	$1.4068 \cdot 10^{-5}$	$3.7856 \cdot 10^{-6}$	$9.5801 \cdot 10^{-7}$

h/t	0.2	0.4	0.6	0.8	1.0
0.1	0.7803	0.5165	0.2694	0.1303	0.0612
0.05	0.7774	0.5138	0.2675	0.1292	0.0606
0.025	0.7648	0.5023	0.2596	0.1244	0.0579
0.0125	0.7568	0.4951	0.2547	0.1214	0.0562

We observed that the method M2 and M4 converges to a different ODE so to approximate our problem we changed our initial ODE, for M2 it was y'=3f and for M4 it was y'=2f, (for errors see Table 2) and we can see that the method M3 is better than M4 if we overlook the computational cost since M3 is not stable for h=0.1 and h=0.05, but for a bigger h the M4 method is better.

Table 2: e_n for the M2, M4 methods in the respective order with the changed ODEs

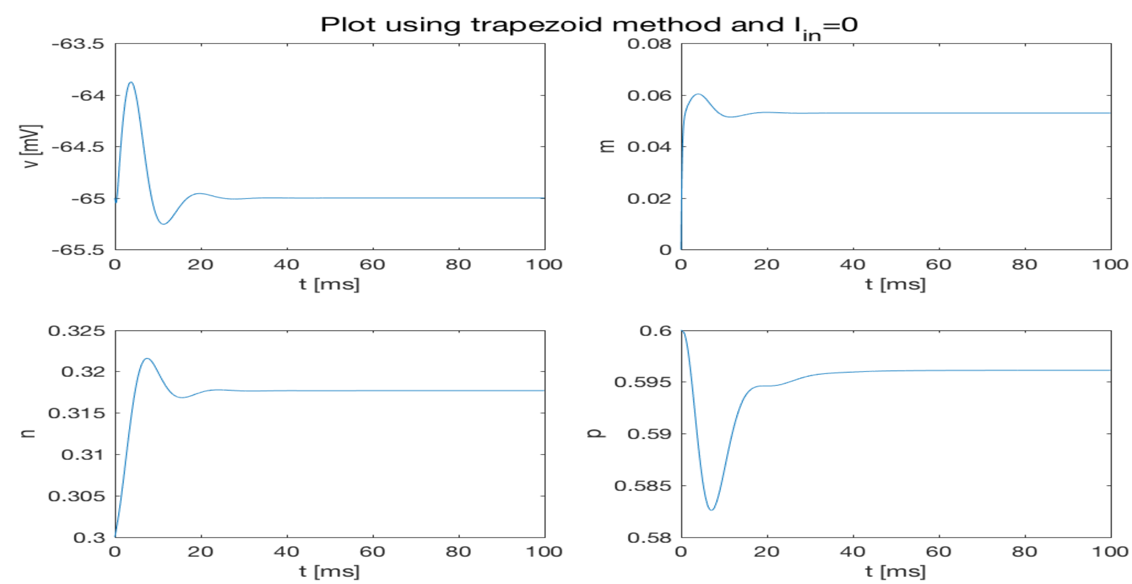
Method(h)/t	0.2	0.4	0.6	0.8	1.0
M2(0.0125)	0.1484	0.0694	0.0240	0.0074	0.0021
M4(0.05)	0.0293	0.0121	0.0038	0.0010	$2.6790 \cdot 10^{-4}$

Exercise 3

We study the Hodgkin-Huxley model with $t \in (0, 100)[ms]$:

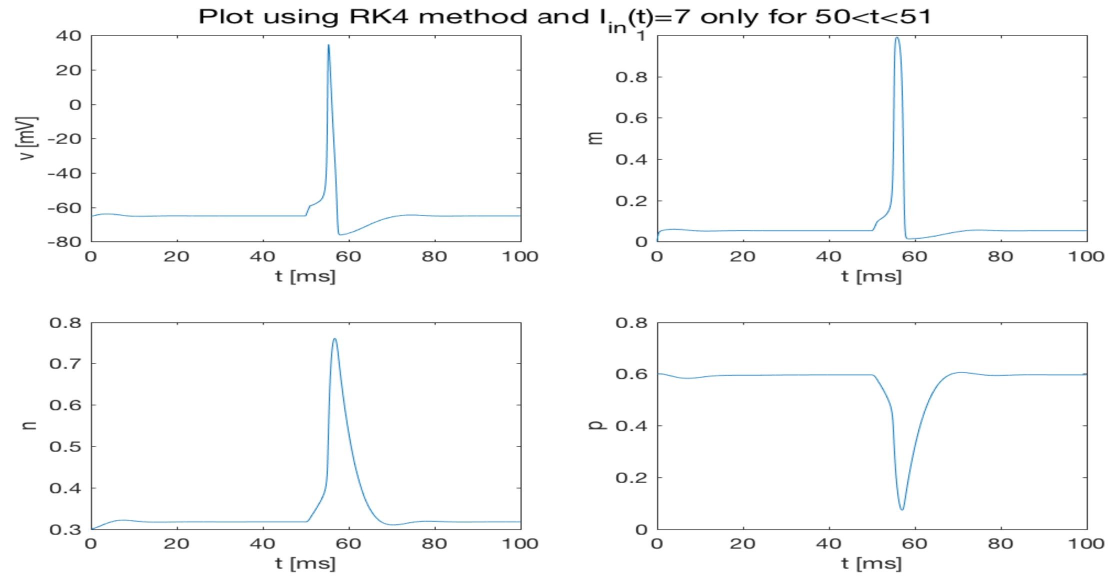
$$\begin{cases} C \cdot v'(t) = I_{in}(t) - g_1 m^3 p \cdot (v - E_1) - g_2 n^4 \cdot (v - E_2) - g_3 \cdot (v - E_3), & v(0) = -65 \\ m'(t) = (1-m)\alpha_m(v - E_0) - m\beta_m(v - E_0), & m(0) = 0 \\ n'(t) = (1-n)\alpha_n(v - E_0) - n\beta_n(v - E_0), & n(0) = 0.3 \\ p'(t) = (1-p)\alpha_p(v - E_0) - p\beta_p(v - E_0), & p(0) = 0.6 \end{cases}$$

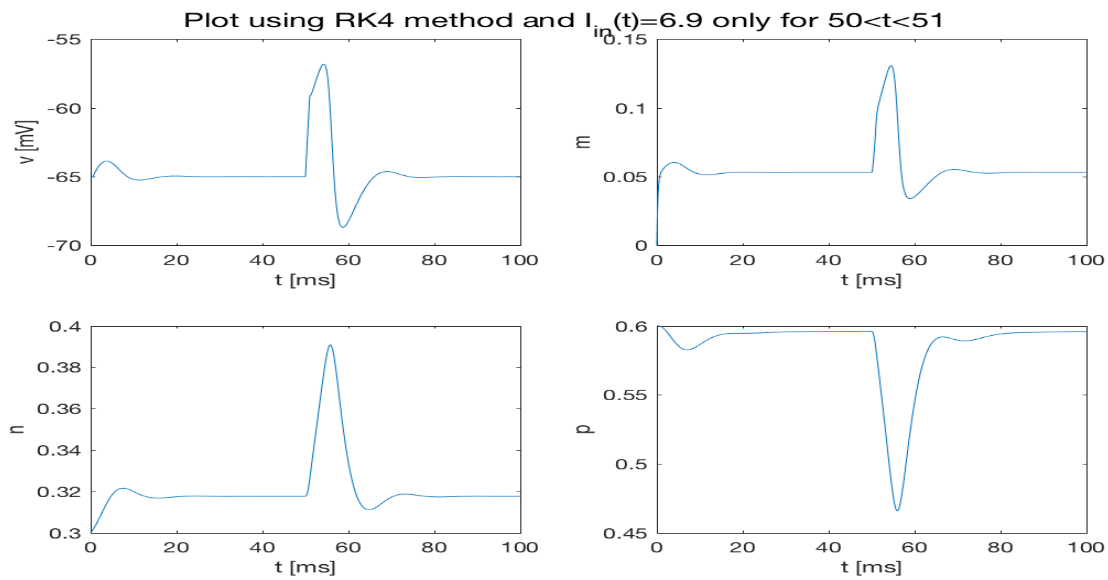
g_i is conductance and we have that $g_i \cdot (v - E_i) = I_i$ where is I_i is the current of a particular type molecule and C is the capacity of the membrane so $C \cdot \frac{dv}{dt} = I_C$ which is also a current, so the first equation is $\sum_{i=1}^{3} I_i - I_{in} + I_C = \sum_{i=1}^{5} I_i = 0$, which is the Kirchhoff's Law.
Initially to simplify our problem we will solve it assuming $I_{in}(t) = 0$ and we solve it using the trapezoid method and we obtain :



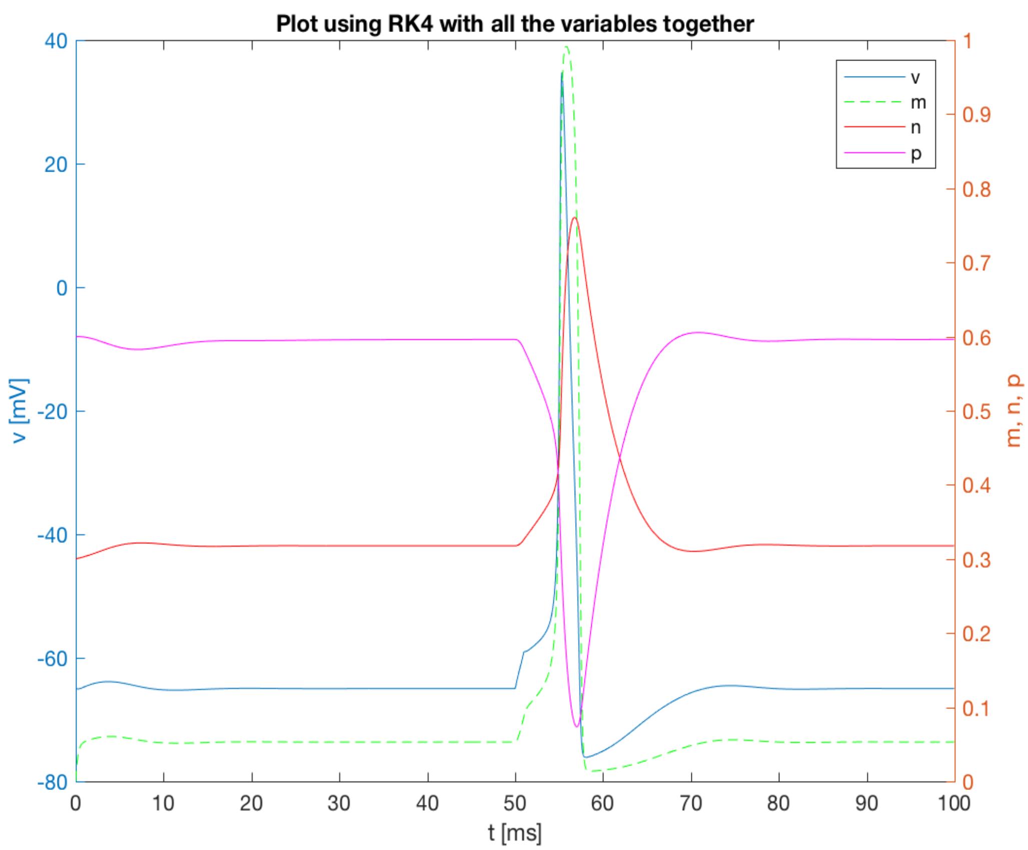
We observe that $v(0) = 65 = E_0$ at the beginning where v is the internal voltage and E_0 is the external voltage. There is a small fluctuation in the beginning due to the initial instability of the gating variables after nearly 20 ms everything stabilises and as expected to reach the balance the interior voltage tends to $E_0 = 65\,mV$.

It is interesting to see the effect of $I_{in}(t)$ when it is not zero, let's check what happens when I_{in} is a impulse of intensity 7 for a duration of 1 ms (at t=50). In this case we will implement the RK4 method :

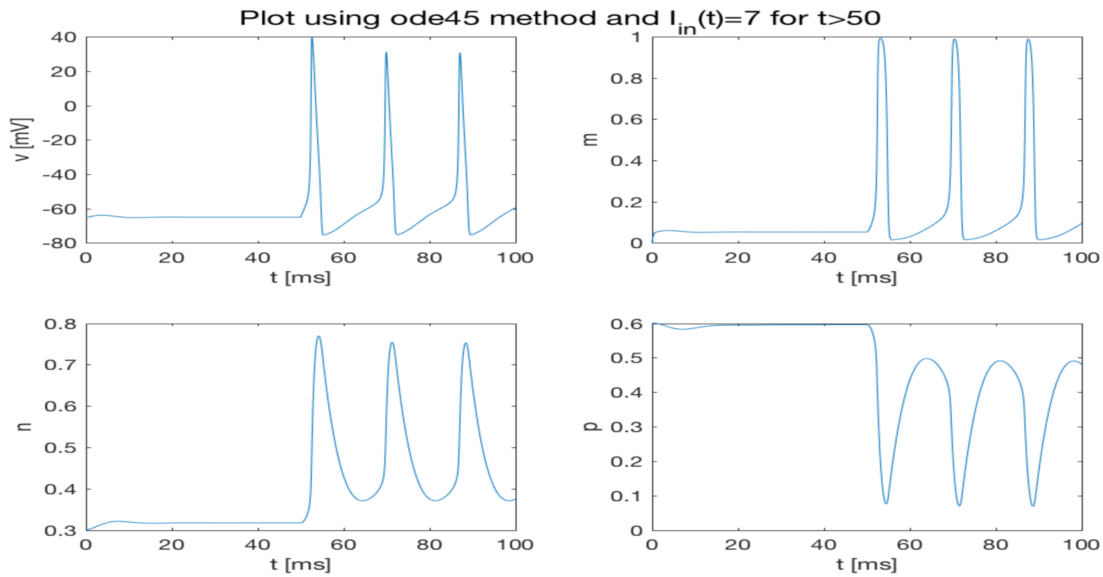




The m, n and p represents the probability of a particular kind of ion gate, respectively Sodium(Na^+) , Potassium(K^+) and other, being open at a certain time t, initially we have K^+ inside the cell and Na^+ outside the cell, as soon as there is $I_{in} > 0$ the cell membrane opens the P^+ and Na^+ gates, it can be seen through the plot of m, n and v since the fluctuation of these three variables starts at the same point. Sodium enters the membrane through the gate and when the intracellular concentration of sodium and potassium molecules are the highest we have the peak in the internal voltage v, after that point nearly 1 ms later the Na^+ gates start to close rapidly and after another ms there is the peak of the percentage of the K^+ gates open so the membrane is throwing outside the potassium molecules and it reduces the internal voltage rapidly and all the gating variables stabilise at a particular percentage in the given conditions, all this process is long roughly 20ms. Another thing to be noted is the difference of the fluctuation when $I_{in} = 7$ and when $I_{in} = 6.9$, the value 7 represents a critical value, below this value the cell membrane is not able to open enough of the K^+ and Na^+ gates for the depolarisation process to begin.



If we take I_{in} = constant = 7 after t=50 :



We observe that all the four functions start behaving like a periodic function after t=50 (when the I_{in} starts playing a role), which has the first peak between 50 and 55 ms the highest of all the peaks, after a certain point the cell releases all of the voltage accumulated for the same reason said in the case of a impulse long 1ms and when it is fully drained and it is back to normal it repeats its cycle.

MATLAB Codes

Exercise 2 Codes

ODE Function

```
%% e' la ODE che studiamo
function f=odefun(t,y)

f=zeros(1,2);
f(1)=y(2);
f(2)=y(2)*(y(2)-1)/(y(1));

return
```

M1 Method

```
%% metodo M1 restituisce u_{n+2}
function yn2=M1(odefun,tn,tn1,tn2,yn,yn1,h)

%usa fsolve per cercare u_{n+2}
options = optimset('Display','off'); %opzione per disabiltiare le
    scritte dovute a fsolve
fun=@(x) x+yn1-2*yn-(h/4)*(odefun(tn2,x)+8*odefun(tn1,yn1)+3*odefun(tn,
    yn)); %la funzione di cui
%calcolare lo zero per noi x e' u_{n+2}
yn2=fsolve(fun,yn,options);%uso questo metodo perch? il metodo di
    Newton non funzionava
%con l'errore della matrice Jacobiana singolare per h=0.025 e 0.0125

return
```

M2 Method

```
%% metodo M2 restituisce u_{n+2}
function yn2=M2(odefun,tn,tn1,yn,yn1,h)

yn2=yn1+(h/3)*(3*odefun(tn1,yn1)-2*odefun(tn,yn));

return
```

M3 Method

```
%% metodo M3 restituisce u_{n+3}
function yn3=M3(odefun,tn,tn2,yn,yn1,yn2,h)

yn3=-(1/4)*yn2+(1/2)*yn1+(3/4)*yn+(h/8)*[19*odefun(tn2,yn2)+5*odefun(tn
    ,yn)];

return
```

M4 Method

```
%% metodo M4 restituisce u_{n+1}
function yn1=M4(odefun,tn,yn,h)
% e' un metodo RK a 3 stadi
K1=odefun(tn,yn);
tK2=tn+(1/3)*h;
yK2=yn+(1/3)*h*K1;
K2=odefun(tK2,yK2);
tK3=tn+(2/3)*h;
yK3=yn+(2/3)*h*K2;
K3=odefun(tK3,yK3);
yn1=yn+(h/4)*(K1+K3);

return
```

2·ODE Function

```
%% e' la ODE che vogliamo studiare moltiplicata per 2
function f=odefun2(t,y)

f=zeros(1,2);
f(1)=2*y(2);
f(2)=2*y(2)*(y(2)-1)/(y(1));

return
```

3·ODE Function

```
%% e' la ODE che vogliamo studiare moltiplicata per 3
function f=odefun3(t,y)

f=zeros(1,2);
f(1)=3*y(2);
f(2)=3*y(2)*(y(2)-1)/(y(1));

return
```

Exercise 2 Main

```
%% Es 2(implementazione di tutti i metodi M1, M2, M3 e M4)
%Scommentare il metodo da eseguire insieme ad altri dati richiesti per tale
%metodo
clear all, close all

ICs=[1/2 -3];

ti=0; tf=1;

hh=[0.1 0.05 0.025 0.0125];

yEx1=@(t) (3/8)*exp(-8*t)+(1/8);
yEx2=@(t) -3*exp(-8*t);

%servono per il metodo M2
%yConv1M2=@(t) (3/8)*exp(-(8/3)*t)+1/8;
%e' dove converge al posto di y1 il metodo M2
%yConv2M2=@(t) -3*exp(-(8/3)*t);
%e' dove converge al posto di y2 il metodo M2

%servono per il metodo M4
% yConv1M4=@(t) (3/8)*exp(-(4)*t)+1/8;
%e' dove converge al posto di y1 il metodo M4
% yConv2M4=@(t) -3*exp(-(4)*t);
%e' dove converge al posto di y2 il metodo M4

err(:,1)=[0; hh']; %la prima colonna di errore(err) sono gli h
x=linspace(0,1);
for l=1:numel(hh)
    clear y t
    j=2; %indice per err
    h=hh(l);
    t(1)=ti;
    t(2)=ti+h;
    y(1,:)=[ICs(1) ICs(2)];
    y(2,:)=[yEx1(t(2)) yEx2(t(2))]; %questo verra riscritto durante M4
    n=ceil((tf-ti)/h);
    %% Metodo M1
%     for i=1:n-1
%         t(i+2)=t(i+1)+h;
%         y(i+2,:)=M1(@odefun,t(i),t(i+1),t(i+2),y(i,:),y(i+1,:),h);
%         if(l==1 && ((i+2)==3||(i+2)==5 || (i+2)==7 || (i+2)==9 || i+2==11))
```

```
%                 err(1,j)=t(i+2);
%                 err(l+1,j)=norm([yEx1(t(i+2)) yEx2(t(i+2))]-[y(i+2,1) y(i+2,2)]);
%                 j=j+1;
%             end
%             if(l==2 && ((i+2)==5||(i+2)==9 || (i+2)==13 || (i+2)==17 || i+2==21))
%                 err(1,j)=t(i+2);
%                 err(l+1,j)=norm([yEx1(t(i+2)) yEx2(t(i+2))]-[y(i+2,1) y(i+2,2)]);
%                 j=j+1;
%             end
%             if(l==3 && ((i+2)==9||(i+2)==17 || (i+2)==25 || (i+2)==33 || i+2==41))
%                 err(1,j)=t(i+2);
%                 err(l+1,j)=norm([yEx1(t(i+2)) yEx2(t(i+2))]-[y(i+2,1) y(i+2,2)]);
%                 j=j+1;
%             end
%             if(l==4 && ((i+2)==17||(i+2)==33 || (i+2)==49 || (i+2)==65 || i+2==81))
%                 err(1,j)=t(i+2);
%                 err(l+1,j)=norm([yEx1(t(i+2)) yEx2(t(i+2))]-[y(i+2,1) y(i+2,2)]);
%                 j=j+1;
%             end
%         end

    %% Metodo M2
%       clear y3;
%       y3(1,:)=ICs;%la sol di ode y'=f*3
%       y3(2,:)=[yConv1M2(t(2)) yConv2M2(t(2))];
%       for i=1:n-1
%             t(i+2)=t(i+1)+h;
%             y(i+2,:)=M2(@odefun,t(i),t(i+1),y(i,:),y(i+1,:),h);
%             %y3(i+2,:)=M2(@odefun3,t(i),t(i+1),y(i,:),y(i+1,:),h); %per ode fun*3
%             if(l==1 && ((i+2)==3||(i+2)==5 || (i+2)==7 || (i+2)==9 || i+2==11))
%                 err(1,j)=t(i+2);
%                 err(l+1,j)=norm([yEx1(t(i+2)) yEx2(t(i+2))]-[y(i+2,1) y(i+2,2)]);
%                 j=j+1;
%             end
%             if(l==2 && ((i+2)==5||(i+2)==9 || (i+2)==13 || (i+2)==17 || i+2==21))
```

```
%                 err(1,j)=t(i+2);
%                 err(l+1,j)=norm([yEx1(t(i+2)) yEx2(t(i+2))]-[y(i+2,1) y(i+2,2)]);
%                 j=j+1;
%             end
%             if(l==3 && ((i+2)==9||(i+2)==17 || (i+2)==25 || (i+2)==33 || i+2==41))
%                 err(1,j)=t(i+2);
%                 err(l+1,j)=norm([yEx1(t(i+2)) yEx2(t(i+2))]-[y(i+2,1) y(i+2,2)]);
%                 j=j+1;
%             end
%             if(l==4 && ((i+2)==17||(i+2)==33 || (i+2)==49 || (i+2)==65 || i+2==81))
%                 err(1,j)=t(i+2);
%                 err(l+1,j)=norm([yEx1(t(i+2)) yEx2(t(i+2))]-[y(i+2,1) y(i+2,2)]);
%                 j=j+1;
%             end
%         end

    %% Metodo M3
    %t(3)=t(2)+h;
    %y(3,:)=[yEx1(t(3)) yEx2(t(3))];
%         for i=1:n-2
%             t(i+3)=t(i+2)+h;
%             y(i+3,:)=M3(@odefun,t(i),t(i+2),y(i,:),y(i+1,:),y(i+2,:),h);
%             if(l==1 && ((i+3)==5 || (i+3)==7 || (i+3)==9 || (i+3)==11))% caso t=0.2 errore = 0
%                 %perche e dato da yEx
%                 j=j+1;
%                 err(1,j)=t(i+3);
%                 err(l+1,j)=norm([yEx1(t(i+3)) yEx2(t(i+3))]-[y(i+3,1) y(i+3,2)]);
%             end
%             if(l==2 && ((i+3)==5||(i+3)==9 || (i+3)==13 || (i+3)==17 || i+3==21))
%                 err(1,j)=t(i+3);
%                 err(l+1,j)=norm([yEx1(t(i+3)) yEx2(t(i+3))]-[y(i+3,1) y(i+3,2)]);
%                 j=j+1;
%             end
%             if(l==3 && ((i+3)==9||(i+3)==17 || (i+3)==25 || (i+3)==33 || i+3==41))
%                 err(1,j)=t(i+3);
%                 err(l+1,j)=norm([yEx1(t(i+3)) yEx2(t(i+3))]-[y(i+3,1) y(i+3,
```

```
+3,2)]);
%                j=j+1;
%           end
%           if(l==4 && ((i+3)==17||(i+3)==33 || (i+3)==49 || (i+3)==65 || i+3==81))
%                err(1,j)=t(i+3);
%                err(l+1,j)=norm([yEx1(t(i+3)) yEx2(t(i+3))]-[y(i+3,1) y(i+3,2)]);
%                j=j+1;
%           end
%      end

 %% Metodo M4
%      clear y2;
%      y2(1,:)=ICs;
%      for i=1:n
%           t(i+1)=t(i)+h;
%           y(i+1,:)=M4(@odefun,t(i),y(i,:),h);
%           %y2(i+1,:)=M4(@odefun2,t(i),y(i,:),h); %la soluzione di y'=2*f;
%           if(l==1 && ((i+1)==3||(i+1)==5 || (i+1)==7 || (i+1)==9 || i+1==11))
%                err(1,j)=t(i+1);
%                err(l+1,j)=norm([yEx1(t(i+1)) yEx2(t(i+1))]-[y(i+1,1) y(i+1,2)]);
%                j=j+1;
%           end
%           if(l==2 && ((i+1)==5||(i+1)==9 || (i+1)==13 || (i+1)==17 || i+1==21))
%                err(1,j)=t(i+1);
%                err(l+1,j)=norm([yEx1(t(i+1)) yEx2(t(i+1))]-[y(i+1,1) y(i+1,2)]);
%                j=j+1;
%           end
%           if(l==3 && ((i+1)==9||(i+1)==17 || (i+1)==25 || (i+1)==33 || i+1==41))
%                err(1,j)=t(i+1);
%                err(l+1,j)=norm([yEx1(t(i+1)) yEx2(t(i+1))]-[y(i+1,1) y(i+1,2)]);
%                j=j+1;
%           end
%           if(l==4 && ((i+1)==17||(i+1)==33 || (i+1)==49 || (i+1)==65 || i+1==81))
%                err(1,j)=t(i+1);
%                err(l+1,j)=norm([yEx1(t(i+1)) yEx2(t(i+1))]-[y(i+1,1) y(i+1,2)]);
```

```
%                   j=j+1;
%             end
%       end
    %% plotting
%       subplot(1,2,1),plot(t,y(:,1)),hold on
%       plot(x,yEx1(x)),hold off
%       subplot(1,2,2), plot(t,y(:,2)), hold on
%       plot(x,yEx2(x)), hold off
%       pause
end
%% stampa errore
%la prima colonna sono i vari h mentre la prima riga sono i tempi a cui
    e'
%calcolato l'errore
%err
```

Exercise 3 Codes

Hodgkin-Huxley Model ODE

```
function f=HH(t,y)
C=1;
g1=120;
g2=36;
g3=0.3;

E0=-65;
E1=50;
E2=-77;
E3= -54.4;

alpha_m=@(v) (2.5-0.1*v)./(exp(2.5-0.1*v)-1);
alpha_n=@(v) (0.1-0.01*v)./(exp(1-0.1*v)-1);
alpha_p=@(v) 0.07*exp(-v/20);

beta_m=@(v)4*exp(-v/18);
beta_n=@(v)1/8*exp(-v/80);%changed from 18 to 80 as told in class
beta_p=@(v)1./(exp(3-0.1*v)+1);
f=zeros(1,4); %f=zeros(4,1); for ode45
v=y(1);
m=y(2);
n=y(3);
p=y(4);
%uncomment the Iin desired.
%Iin=0;

% if(t>50) && (t<=51)
%     Iin=7;
% else
%       Iin=0;
% end

% if(t>50)
%     Iin=7;
% else
%       Iin=0;
% end
f(1)=(1/C)*(Iin-g1*m^3*p*(v-E1)-g2*n^4*(v-E2)-g3*(v-E3));
f(2)=(1-m)*alpha_m(v-E0)-m*beta_m(v-E0);
f(3)=(1-n)*alpha_n(v-E0)-n*beta_n(v-E0);
f(4)=(1-p)*alpha_p(v-E0)-p*beta_p(v-E0);
return
```

RK4 Method

```
function y=rk4step(odefun,t,un,h)

K1=odefun(t,un);%passo di EE
K2=odefun(t+h/2,un+h*K1/2);
K3=odefun(t+h/2,un+h*K2/2);
K4=odefun(t+h,un+h*K3);

y=un+h*(K1+2*K2+2*K3+K4)/6;
return
```

Trapezoid Method

```
function y=trapstep(odefun,t,z,h)

%passo predictor(passo EE)
z1=z+h*odefun(t,z);

%passo corrector
z2=odefun(t+h,z1);
%media
y=z+(odefun(t,z)+z2)*h/2;
return
```

Exercise 3 Main

```
%% Es 3
% parameters and functions for the HH model of neuron firing
%uncomment the desired method
clear all, close all
ti=0; tf=100;
ICs=[-65 0 0.3 0.6];
h=0.05;
y(1,:)=ICs;

n=ceil((tf-ti)/h);
t=zeros(1,n);
t(1)=ti;
for i=1:n-1
    t(i+1)=t(i)+h;
    y(i+1,:)=rk4step(@HH,t(i),y(i,:),h);
    %y(i+1,:)=trapstep(@HH,t(i),y(i,:),h);
end

%[t,y]=ode45(@HH,[ti tf],ICs);
%do not ise ode45 while using I_in = 7 for 50<t<=51 since
```

```
%it uses bigger steps and does not see the I_in


%to plot all toghether
% figure(1),
% title('Plot using RK4 with all the variables together')
% yyaxis left
% plot(t,y(:,1))
% ylabel('v [mV]')
% yyaxis right
% hold on
% plot(t,y(:,2),'g--')
% plot(t,y(:,3),'r')
% plot(t,y(:,4),'m')
% ylabel('m, n, p')
% xlabel('t [ms]')
% legend('v','m','n','p')

% figure(2),suptitle('Plot using ode45 method and I_{in}(t)=7 for t >50'),
% subplot(2,2,1),plot(t,y(:,1))
% ylabel('v [mV]')
% xlabel('t [ms]')
% subplot(2,2,2),plot(t,y(:,2))
% ylabel('m')
% xlabel('t [ms]')
% subplot(2,2,3),plot(t,y(:,3))
% ylabel('n')
% xlabel('t [ms]')
% subplot(2,2,4),plot(t,y(:,4))
% ylabel('p')
% xlabel('t [ms]')
print -dpng grafico_sovra
```