Practical Numerical Mathematics CTB2000(Q4)

To be filled in by student:

- *Full completion of this form is obligatory*
- *Every report must be handed in separately*
- ***Do not extend this form to add an extra (third) student***

Number (code) of assignment: 2N4 (one assignment only)
Description of activity: H2 & H3 (e.g. CT, group 2)

Report on behalf of (maximum two students):

Pieter van Halem	name
4597591	student number

Dennis Dane	name
4592239	student number

This must be the first page of your report, to be uploaded using Blackboard.

So, print the cover sheet, fill in and scan together with the (handwritten) theory part of your report.

Data of student taking the role of contact person:

Pieter van Halem	name
pietervanhalem@hotmail.com	email address

Not to be filled in by students:

Receipt:	
date of receipt:	x
corrected by:	x

x = to be filled in by administration
o = to be filled in by corrector

If approved:	
date and initials:	o

If to be adapted:	
date and initials:	o
date of return to student:	x
date of receipt adapted report:	x

Version May 2015

INHOUDSOPGAVE

1.0 PHASE I – THEORY

In this report the behaviour of the following system of differential equations is described:

$$\begin{cases} \frac{du}{dt} = a - (b+1)u + u^2v \\ \frac{dv}{dt} = a - (b+1)u + u^2v \\ u(0) = 0 \\ v(0) = 0 \end{cases} \quad (1)$$

1.1

Write (1) as a system of first-order differential equations.

The answer to this is question is fairly easy. It is just a matter of rewriting equation 1 into a vector form (eq. 2):

$$\begin{bmatrix} u \\ v \end{bmatrix}' = \begin{bmatrix} a - (b+1)u + u^2v \\ bu - u^2v \end{bmatrix} \quad (2)$$

1.2

Determine the equilibrium (critical point, stationary point) of the system (3). Apply local linearization in order to investigate the character of the equilibrium (see the supplement). For this one needs to form the Jacobian matrix taken at the equilibrium point. We denote this matrix with A. Further the values of a and b are given. (a = 2 and b = 4.5).

$$\begin{aligned} &\frac{d\boldsymbol{y}}{dt} = f(\boldsymbol{y}), \\ &\boldsymbol{y} = \begin{bmatrix} u \\ v \end{bmatrix} \\ &f(\boldsymbol{y}) = \begin{bmatrix} 2 - 5.5u + u^2v \\ 4.5u - u^2v \end{bmatrix} \end{aligned} \quad (3)$$

First the equilibrium points have to be determined. This will be done by solving the system (4, filled in with values a and b). The equations will be set equal to 0 to determine these points.

$$\begin{bmatrix} u \\ v \end{bmatrix}' = \begin{bmatrix} 2 - 5.5u + u^2v \\ 4.5u - u^2v \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \xrightarrow{yields} \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} 2.00 \\ 2.25 \end{bmatrix} \quad (4)$$

This system only has one equilibrium point (see eq. 4).

To determine the nature of the equilibrium point, the Jacobian has to be determined (4 – I).

$$J = \begin{bmatrix} \frac{df_1}{du} & \frac{df_1}{dv} \\ \frac{df_2}{du} & \frac{df_2}{dv} \end{bmatrix} = \begin{bmatrix} -5.5 + 2uv & u^2 \\ 4.5 - 2uv & -u^2 \end{bmatrix} \xrightarrow{yields} A = \begin{bmatrix} 3.50 & 4.00 \\ -4.50 & -4.00 \end{bmatrix} \quad (4 - I)$$

The last step is to determine the eigenvalues of the Jacobian (Matrix A in eq. 4 – I). The eigenvalues are computed with the characteristic equation: (3.50 - λ)*(-4.00 - λ) + 4.00 * 4.50 = 0.This results in λ=-0.25 ± 1.984*i . (eq. 5). The real part is negative this yields in a stable equilibrium point. There also is an imaginary part. This results in a spiral point.

$$Det\begin{bmatrix}3.50-\lambda & 4.00\\ -4.50 & -4.00-\lambda\end{bmatrix}=0 \quad (5)$$

Result:

λ=-0.25 ± 1.984*i

$$(3.50-\lambda)*(-4.00-\lambda)+4.00*4.50=0$$

$$\lambda^2+0.5\lambda+4=0 \quad (5-I)$$

$$\lambda=-0.25\pm 1.984313483298443*i$$

1.3 & 1.4

*Using Python, make a graph in the complex plan containing Λ_h for h = 0, 0.25, 0.5, 0.75. The scale along the axis must be set according to **plt.xlim**(-2.25, 0.25) and **plt.ylim**(-2,2). This graph shows eight points in the complex plane. Are these eight points on the line?*

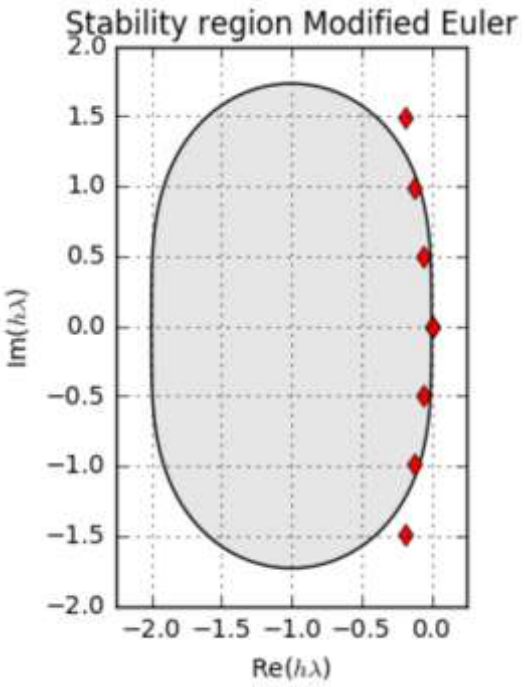


Figure 1 – Stability.

Python code:

```
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
from scipy.stats import norm

L = [[-0.25,-0.25],[1.9843,-1.9843]];
x0 =-2.2
x1 = 0.2
Nx = 300

y0 =-2
y1 = 2
Ny = 300


X,Y = np.meshgrid(np.linspace(x0,x1,Nx+1),np.linspace(y0,y1,Ny+1))
Z = X +1j*Y
Q = 1 + Z + Z**2/2
Q_mag = np.abs(Q)

for i in range(4):
    plt.plot(L[0][0]*0.25*i,L[1][0]*0.25*i,'rd')
    plt.plot(L[0][1]*0.25*i,L[1][1]*0.25*i,'rd')

plt.contour(X,Y,Q_mag,[0,1],colors ='k');
plt.contourf(X,Y,Q_mag,[0,1],colors='0.9');
plt.grid();
plt.axis("scaled");
plt.title("Stability region Modified Euler");
plt.xlabel("Re($h\lambda$)");
plt.ylabel("Im($h\lambda$)");
plt.xlim(-2.25,0.25);
plt.ylim(-2,2);
```

The picture above shows seven points. The assignment states that this graph shows eight points. This is because the point (0, 0) is counted twice.

In the graph is clearly visible that the points are in the stability region in h ≤ 0.5. Thus the numeric method is stable for h ≤ 0.5.

1.5

The time step h has to be chosen such that at least 20 steps per period are taken. For which values of h can we achieve this?

$$\mu + v * i = -0.25 \pm 1.9843 * i \quad (6)$$

Result:
h=0.1583

$$P = \frac{2*\pi}{v} = \frac{2*\pi}{1.9843} = 3.166$$

$$20 * h = P \xrightarrow{yields} h = \frac{3.166}{20} = 0.1583 \quad (6\text{– I})$$

1.6

Of what order in h is the local truncation error of the present numerical method? Present also the order of the global truncation error (discretization error).

This question can be answered by using Lax's Theorem.

> [Lax's equivalence theorem (p75)]
> "If a numerical method is stable and consistent, then the numerical approximation converges for the solution for Δt -> 0. Moreover, the global truncation error and the local truncation error are of the same order."

In this assignment we use the modified Euler method. The local truncation error for this method is well known $O(h^2)$ (p74). This means that the global truncation error also is $O(h^2)$.

1.7

We can estimate the error in the computed approximation at a certain fixed time t (say t = 1) by comparing the results using time step h and those using time step 2h. Let us denote these results by w_h and w_{2h} respectively. The resulting estimation of the error E in w_h, i.e. $E = y(t) – w_h$, takes the form

$E \approx \alpha(w_h - w_{2h})$, for certain α. (7)

Present the value of α for the present numerical method.

The estimate of the global truncation-error is (the p of the modified Euler is 2). See also page 79.

$$w_N^h - w_{\frac{N}{2}}^{2h} = \alpha * h^p * (2^p - 1)$$

$$\frac{1}{(2^p - 1)} * (w_N^h - w_{\frac{N}{2}}^{2h}) = \alpha * h^p$$

$$\alpha = \frac{1}{2^p - 1} = \frac{1}{2^2 - 1} = \frac{1}{3} \quad (7 - \text{II})$$

1.8

The supply mechanism for material A may fail sometimes. This means that a = 0. Determine the equilibrium in this case and use local linearization to study the nature of this equilibrium point. Now suppose that at t = t_1 the supply fails. How do you think the solution of (3) will behave for t >> t_1?

Determining the equilibrium will be done in the same was as in question 1.2. (a = 0 and b = 4.5)

$$\begin{bmatrix} u \\ v \end{bmatrix}' = \begin{bmatrix} -(b+1)u + u^2 v \\ bu - u^2 v \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad (8)$$

Result:

u = 0 ;v ∈ ℝ

$$\begin{matrix} u(uv - b + 1) = 0 \\ u(b - uv) = 0 \end{matrix} \xrightarrow{yields} u = 0 \text{ or}$$

$$\begin{matrix} uv - b + 1 = 0 \\ b - uv = 0 \end{matrix} \rightarrow \begin{matrix} uv = b + 1 \\ uv = b \end{matrix} \xrightarrow{yields} \text{is impossible, so u = 0 and v} \in \mathbb{R} \quad (8\text{– I})$$

This result gives not a finite number of equilibrium points but infinitely many along the v-axis. This means there is an equilibrium line. The Jacobian will be determined for u = 0 and v ∈ ℝ. The phase plane has infinite many equilibrium point on the u-axis.

$$J = \begin{bmatrix} \frac{df_1}{du} & \frac{df_1}{dv} \\ \frac{df_2}{du} & \frac{df_2}{dv} \end{bmatrix} = \begin{bmatrix} -(b+1) + 2uv & u^2 \\ b - 2uv & -u^2 \end{bmatrix} \xrightarrow{yields} A = \begin{bmatrix} -b+1 & 0 \\ b & 0 \end{bmatrix} \quad (9 - I)$$

$$Det \begin{bmatrix} -b-1-\lambda & 0 \\ b & 0-\lambda \end{bmatrix} = 0 \rightarrow triangular\ matrix\ so\ \lambda_1 = -b - 1 = -5.5\ \ and\ \lambda_2 = 0 \quad (9\text{- II})$$

For these eigenvalues holds that RE(λ) ≤ 0. So the equilibrium line is stable. When the supply fails at t_1 the solution for u and v will converge to a value (for t >> t_1 depending on the initial conditions), because the equilibrium is stable.

Equilibrium analysis Chemical reaction

Number (code) of assignment: 2N4

Description of activity: H2 & H3

Report on behalf of:

name : Pieter van Halem student number (4597591)

name : Dennis Dane student number (4592239)

Data of student taking the role of contact person:

name : Pieter van Halem

email address : pietervanhalem@hotmail.com

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
```

Function definitions:

In the following block the function that are used for the numerical analysis are defined. These are functions for calculation of the various time steps, plotting tables and plotting graphs.

In [2]:
```
def f(t,y,a,b,i):
    if (t>i):
        a = 0
    du = a-(b+1)*y[0,0]+(y[0,0]**2)*y[0,1]
    dv = b*y[0,0]-(y[0,0]**2)*y[0,1]
    return np.matrix([du,dv])

def FE(t,y,h,a,b,i):
    f1 = f(t,y,a,b,i)
    pred = y + f1*h
    corr = y + (h/2)*(f((t+h),pred,a,b,i) + f1)
    return corr

def Integrate(y0, t0, tend, N,a,b,i):
    h = (tend-t0)/N

    t_arr = np.zeros(N+1)
    t_arr[0] = t0

    w_arr = np.zeros((2,N+1))
    w_arr[:,0] = y0

    t = t0
    y = y0
    for k in range(1,N+1):
        y = FE(t,y,h,a,b,i)
        w_arr[:,k] = y
        t = t + h
        t_arr[k] = t

    return t_arr, w_arr

def PrintTable(t_arr, w_arr):
    print ("%6s %6s: %17s %17s" % ("index", "t", "u(t)", "v(t)"))
    for k in range(0,N+1):
        print ("{:6d} {:6.2f}: {:17.7e} {:17.7e}".format(k,t_arr[k],
                                                        w_arr[0,k],w_arr[1,k]))

def PlotGraphs(t_arr, w_arr):
    plt.figure("Initial value problem")
    plt.plot(t_arr,w_arr[0,:],'r',t_arr,w_arr[1,:],'--')
    plt.legend(("$u(t)$", "$v(t)$"),loc="best", shadow=True)
    plt.xlabel("$t$")
    plt.ylabel("$u$ and $v$")
    plt.title("Graphs of $u(t)$ and $v(t)$")
    plt.show()

def PlotGraphs2(t_arr, w_arr):
    plt.figure("Initial value problem")
    plt.plot(w_arr[0,:],w_arr[1,:],'g')
    plt.legend(("$u,v$",""),loc="best", shadow=True)
    plt.xlabel("$u(t)$")
    plt.ylabel("$v(t)$")
    plt.title("$Phase$ $plane$ $(u,v)$")
    plt.axis("scaled")
    plt.show()
```

Assignment 2.9

Integrate the system with Modified Euler and time step h = 0.15. Make a table of u and v on the time interval $0 \le t \le 1.5$. The table needs to give u and v in an 8-digit floating-point format.

In [3]:
```
y0 = np.matrix([0.0,0.0])
t0 = 0.0
tend = 1.5
N = 10

t_array, w_array = Integrate(y0, t0, tend, N,2,4.5,11)
print("The integrated system using Modified Euler with time step h = 0.15 is sho
wn in the following table: \n")
PrintTable(t_array, w_array)
```

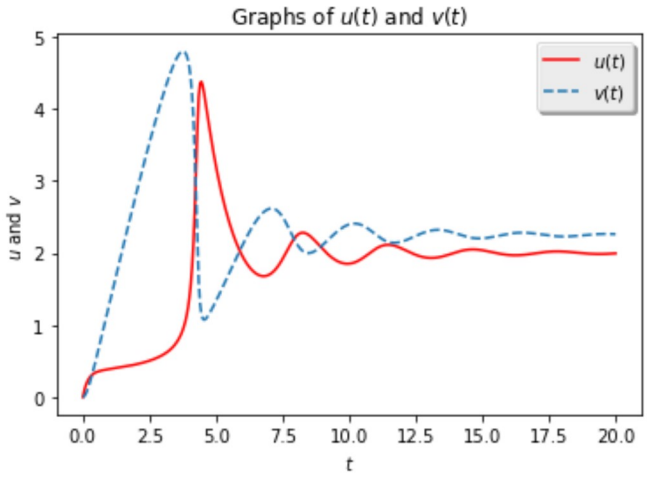
```
The integrated system using Modified Euler with time step h = 0.15 is shown in
the following table:

 index      t:              u(t)              v(t)
     0   0.00:     0.0000000e+00     0.0000000e+00
     1   0.15:     1.7625000e-01     1.0125000e-01
     2   0.30:     2.6892423e-01     2.7050835e-01
     3   0.45:     3.1921278e-01     4.7380077e-01
     4   0.60:     3.4812212e-01     6.9371767e-01
     5   0.75:     3.6633149e-01     9.2138423e-01
     6   0.90:     3.7927846e-01     1.1522633e+00
     7   1.05:     3.8975403e-01     1.3839990e+00
     8   1.20:     3.9921620e-01     1.6153246e+00
     9   1.35:     4.0845332e-01     1.8455103e+00
    10   1.50:     4.1792028e-01     2.0740846e+00
```

Assignment 2.10

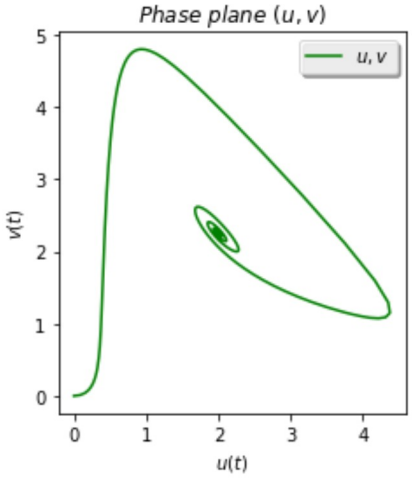
Integrate the system with Modified Euler and time step h = 0.05 for the interval [0,20]. Make plots of u and v as functions of t (put them in one figure). Also make a plot of u and v in the phase plane (u,v-plane). Do your plots correspond to your results of part 2?

In [4]:
```
y0 = np.matrix([0.0,0.0])
t0 = 0.0
tend = 20
N = 400

t_array, w_array = Integrate(y0, t0, tend, N,2,4.5, 25)

print("In this assignment the system has to be integrated using Modified Euler w
ith a time step of h = 0.05 on \na interval of [0,20].")
print("The first graph is u(t) and v(t) against time (t).")
PlotGraphs(t_array, w_array)
print("The second graph shows the u-v plane")
PlotGraphs2(t_array, w_array)
```

```
In this assignment the system has to be integrated using Modified Euler with a
time step of h = 0.05 on
a interval of [0,20].
The first graph is u(t) and v(t) against time (t).
```




```
The second graph shows the u-v plane
```



Although the direction cannot be seen in the phase plane graph. Using the first plot one can see that u(t) and v(t) will go to an equilibrium as time is increasing. Therefor the system is stable and a spiral. Which is consistent with the conclusion from assignment 1.3.

Assignment 2.11

Using the formula derived in question 7, estimate the accuracy of u and v computed with h = 0.05 at t = 8. Hence, integrate once more with time step h = 0.1.

The error can be estimated with Richardson's method. We will use α = 1/3 found in assignment 7. Here the estimated error is: E ≈ α(w(h) - w(2h)).

In [5]:
```
y0 = np.matrix([0.0,0.0])
t0 = 0.0
tend = 20
N = 400

t_array, w_array = Integrate(y0, t0, tend, N, 2, 4.5,25)

y0 = np.matrix([0.0,0.0])
t0 = 0.0
tend = 20
N = 200

t_array2, w_array2 = Integrate(y0, t0, tend, N, 2, 4.5, 25)

print("The value for u and v at t = 8.00 with h = 0.05 is: {:.2f} {:.7e} {:.7e}"
.format(t_array[160], w_array[0,160],w_array[1,160]))
print("The value for u and v at t = 8.00 with h = 0.10 is: {:.2f} {:.7e} {:.7e}"
.format(t_array[80], w_array[0,80],w_array[1,80]))
```

```
The value for u and v at t = 8.00 with h = 0.05 is: 8.00 2.2294777e+00 2.14650
17e+00
The value for u and v at t = 8.00 with h = 0.10 is: 4.00 1.3812297e+00 4.57092
11e+00
```

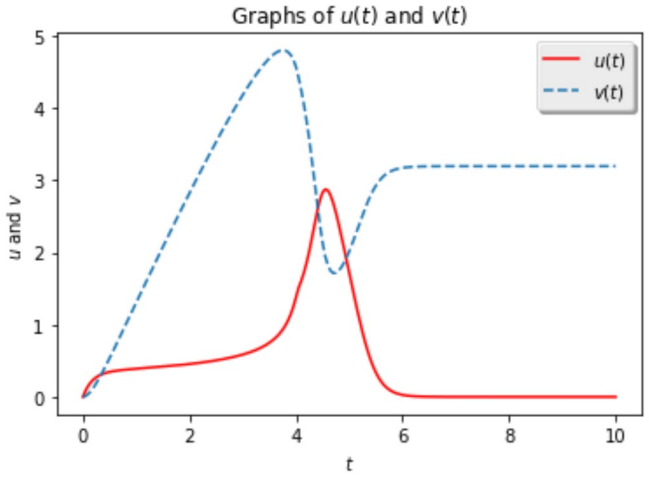
In [6]:
```
E1 = (w_array[0,160]-w_array2[0,80])*(1/3)
E2 = (w_array[1,160]-w_array2[1,80])*(1/3)
print("The estimated accuracy for u is: {:.7e}".format(E1))
print("The estimated accuracy for v is: {:.7e}".format(E2))
```

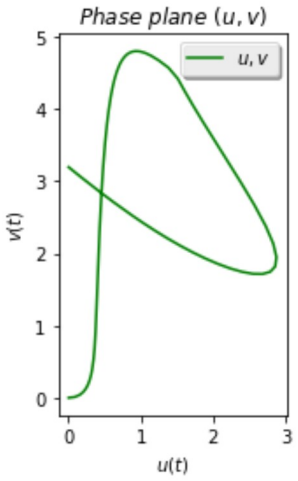
```
The estimated accuracy for u is: 3.5890366e-03
The estimated accuracy for v is: -3.9656575e-03
```

Assignment 2.12

Apply Modified Euler with h = 0.05. For 0 ≤ t ≤ t1 it holds that a = 2. At t = t1 the supply of materials A fails, and therefore a = 0 for t > t1. Take t1 = 4.0. Make plot of u and v as a function of t on the intervals [0, 10] in one figure and a plot of u and v in the uv-plane. Evaluate your results by comparing them to your findings form part 8.

In [7]:
```
y0 = np.matrix([0.0,0.0])
t0 = 0.0
tend = 10.0
N = 200

t_array, w_array = Integrate(y0, t0, tend, N, 2, 4.5, 4)


PlotGraphs(t_array, w_array)
PlotGraphs2(t_array, w_array)
```





The first plot shows that u and v indeed converges to a certain value, as predicted in assignment 8. The phase plane shows that uv goes to a point on the u-axis. This was also predicted in assignment 8. The first plot shows a "corner" in the u and v graph (a discontinuity in the first derivative). This does not contradict the theory, the system of differential eqution its first derivative does not have to be continuous. The line itself is continuous because of the initial conditions.

Assignment 2.13

Take t1 = 3.0, 3.5, 4.0, 4.5, 5.0, 5.5, 6.0. Make a table of the value of v-tilde and t-tilde. Evaluate your results.

In [8]:
```
for i in np.arange(3.0,6.5,0.50):
    t0 = 0
    tend = 10
    N = 200
    t_array2, w_array2 = Integrate(y0, t0, tend, N, 2.0, 4.5,i)
    indices = np.nonzero(w_array2[0,:] >= 0.01)
    index = np.max(indices[0])
    t_tilde = t_array2[index+1]
    v_tilde = w_array2[1,N]
    if i == 3:
        print("%6s %17s: %17s " % ("t1", "t_tilde", "v_tilde"))
    print("{:6.2f} {:17.2f} {:17.7e}".format(i,t_tilde,v_tilde))
```

```
    t1           t_tilde:           v_tilde
  3.00              3.90     4.6548679e+00
  3.50              4.55     5.2387123e+00
  4.00              6.20     3.1942867e+00
  4.50              6.30     3.0999532e+00
  5.00              6.60     3.1138961e+00
  5.50              6.95     3.1923155e+00
  6.00              7.35     3.3476197e+00
```

In the table above are the values for t-tilde and v-tilde shown. There is not an obvious relation between the time(t1) and t-tilde nor an obvious relation between t1 and v-tilde. However we can conclude that if t1 becomes larger t-tilde becomes larger. This cannot be concluded for v-tilde. The value for v-tilde seems to converge to an arbitrarily value between 3.0 and 3.5 (clearly dependent on the initial conditions). The convergence of v_tilde is also consistent with the findings in assignment 1.8.