Applied Numerical Methods - Computer Lab 1

GOYENS Florentin & WEICKER David

15th September 2015

Part 1. Solution of ODE-systems with constant coefficients

In the electrical circuit that follows, we define the variable $\dot{q} = i$. dessin du circuit to come, je sais pas si on a vraiment besoin... The RLC circuit is described by the equation

$$L\ddot{q} + R\dot{q} + \frac{1}{C}q = E, \qquad q(0) = 0, \qquad \dot{q}(0) = 0.$$

We easily rewrite this as a first order system of linear equations. Setting $y_1 = q$ and $y_2 = \dot{q}$, this gives

$$\begin{aligned} \dot{y_1} &= y_2 \\ \dot{y_2} &= -\tfrac{R}{L}y_2 - \tfrac{1}{LC}y_1 + E \end{aligned}$$

$$\begin{pmatrix} \dot{y_1} \\ \dot{y_2} \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ -\frac{1}{LC} & -\frac{R}{L} \end{pmatrix} + \begin{pmatrix} 0 \\ E \end{pmatrix}$$

Part 2. Stability of ODE-systems and equilibrium points

a. Stability of the solutions of an ODE-system of LCC-type

The purpose of this section is to investigate the stability of an ODE while a parameter changes continuously.

The given third-order equation is the following :

$$\begin{aligned} y''' + 3y'' + 2y' + Ky = {} & 0 \\ y(0) = {} & 1 \\ y'(0) = {} & 1 \\ y''(0) = {} & 1 \end{aligned}$$

It is possible to rewrite this as a system of ODE's, introducing new variables.

$$\mathbf{u'} = \begin{pmatrix} y' \\ y'' \\ y''' \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -K & -2 & -3 \end{pmatrix} \begin{pmatrix} y \\ y' \\ y'' \end{pmatrix} = \mathbf{Au}$$

The initial conditions are rewritten by :

$$\mathbf{u(0)} = \begin{pmatrix} y(0) \\ y'(0) \\ y''(0) \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$$

The solutions, computed analytically for different values of K, are given in figure 1. The Matlab code is given at the end of this section. It is possbile to see that for the first three

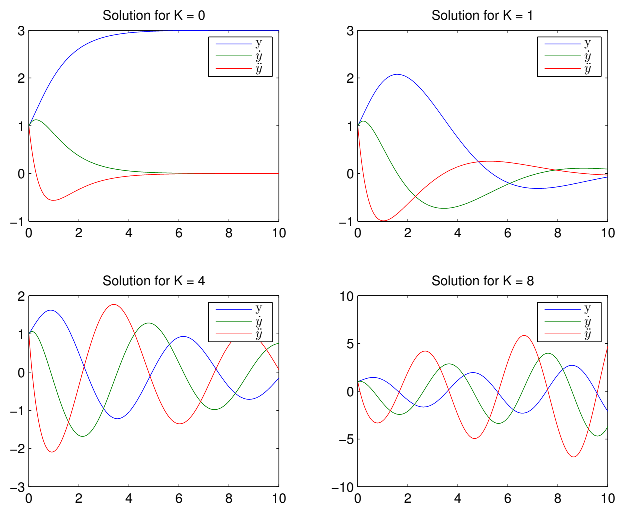

Figure 1: Solutions of the system in part 2a for various values of K

values of K, the maximal amplitude of the blue curve (that is the function y) tends to decrease. This being an LCC system, we know that a perturbation will follow the same ODE so we can conclude that the system is stable for those values of K. On the other hand, for the last value ($K = 8$), this amplitude increases. Hence, the system is not stable for $K = 8$. By continuity, that means that the system becomes unstable for a value of K between 4 and 8. Because we know nothing else, the best estimation that we can give right now is that this value is $K = 6$.

We then drew a root locus to see how the system would evolve if K varies continuously. Again, the code is given at the end of the section. We used the built-in function *rlocus*. In order to do so, we first have to compute the transfert function of the process that is equivalent to our differential equation with an unit feedback with gain K. One can show that this transfert function $H(s)$ is :

$$H(s) = \frac{1}{s^3 + 3s^2 + 2s}$$

The figure 2 shows the rootlocus. The values of k were chosen so that the curve would be smooth. We can see that as K increases, one eigenvalue is getting more and more negative while the two others first are getting closer then leave the real axis. We can also see that at some point, those two eigenvalues cross the imaginary axis thus making the system unstable.

The next step was to compute the first value of K making the system unstable. It is possible to check each time we increase K if the eigenvalue have a positive real part but the result is only an approximation. It is also possible to compute this value analytically and that is what we did. By continuity, we know that for this value, two eigenvalues are located on the imaginary axis so :

$$\lambda = xj$$

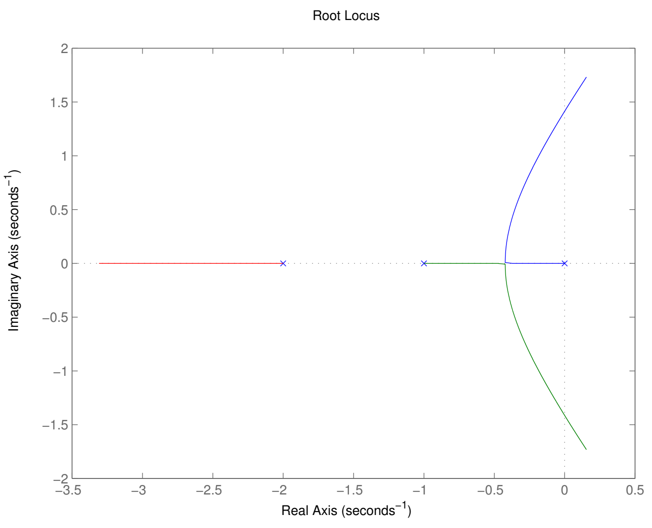

Figure 2: Root locus of $y''' + 3y'' + 2y' + Ky = 0$

With x a real number. We then plug that into the caracteristic equation.

$$\begin{aligned}
\lambda^3 + 3\lambda^2 + 2\lambda + K &= 0 \\
-x^3 j - 3x^2 + 2xj + K &= 0 \\
\begin{pmatrix} K - 3x^2 \\ 2x - x^3 \end{pmatrix} &= \begin{pmatrix} 0 \\ 0 \end{pmatrix}
\end{aligned}$$

We have the trivial solution ($x = K = 0$) but we are not interested in that one. The two others are :

$$\begin{aligned}
x &= \pm\sqrt{2} \\
K &= 6
\end{aligned}$$

So we can conclude that :

$$\textit{The system is unstable} \iff K > 6$$

```
function [] = LAB1_21()
%LAB1_21
close all;
K = [0 1 4 6];
u0 = [1 1 1]';
N = 100;
u = [u0 zeros(3,N)];
tfinal = 10;
h = tfinal/N;
t=0:h:tfinal;

for i=1:4
    A=[0 1 0;0 0 1;-K(i) -2 -3];
    for j=2:N+1
        u(:,j) = expm(A*t(j))*u0;
    end
    subplot(2,2,i);
    string = sprintf('Solution for K = %d',K(i));
    plot(t,u);hl=legend('y','$\dot{y}$','$\ddot{y}$');title(string);
    set(hl, 'Interpreter', 'latex');
end

%root locus
sys = tf(1,[1 3 2 0]);
figure;
k = 0:0.005:10;
rlocus(sys,k);
end
```

b. Stability of the critical points of a nonlinear ODE-system

In this section, we will look at the stability. As this sytem is nonlinear, we have to analyse the stability of the critical points with the help of the jacobian. For this system, the jacobian is analytically given by :

$$D_f(\mathbf{u}) = \begin{pmatrix} 5-u_3 & 4 & -u_1 \\ 1 & 4-u_3 & -u_2 \\ 2u_1 & 2u_2 & 0 \end{pmatrix}$$

This jacobian is a function of $\mathbf{u}$ but we are only interested in the stability of the critical points. So we only have to check the eigenvalues of $D_f(\mathbf{u}^*)$ where $\mathbf{u}^*$ is a solution of the equation $f(\mathbf{u}) = 0$.

So we first have to compute all the critical points. There are four of them. The Matlab code shown at the end of this section is used to do that. We have used the built-in function $fsolve$. This give the following critical points, numeroted u^i :

u^1	u^2	u^3	u^4
7.9446	−7.9446	8.7881	−8.7881
−5.0876	5.0876	3.4308	−3.4308
2.4384	2.4384	6.5616	6.5616

```
function [sol,lambda] = LAB1_22()
%LAB1
close all;

approx = [8 -5 2; -8 5 2; 9 3 7; -9 -3 7];
sol = zeros(3,4);
lambda = zeros(3,4);
for i=1:4
    sol(:,i) = fsolve(@twoB,approx(i,:))';
    jacobian = [5-sol(3,i) 4 -sol(1,i);1 4-sol(3,i) -sol(2,i);2*sol(1,i) 2*sol(2,i) ↩
        0]; %analytical jacobian
    lambda(:,i) = eig(jacobian);
end

end

function dudx = twoB(u)
%Computes derivative of u in part 2b
dudx = u;
dudx(1) = 5*u(1) + 4*u(2) - u(1)*u(3);
dudx(2) = u(1) + 4*u(2) - u(2)*u(3);
dudx(3) = u(1)*u(1) + u(2)*u(2) - 89;
end
```