315010xxxx
武鑫
Wu Hsin

Exercise 2-1 Block oriented ODE integration

(a)

Building up the transfer function for the system:

$$G(s) = \frac{0.99s}{0.01s^3 + 1.02s^2 + 2.02s + 1}$$

From the transfer function can we get the eigenvalues: $\lambda_1 = -100$, $\lambda_2 = -1$, $\lambda_3 = -1$.

Obviously λ_1 is the critical value to determine the h_{crit} since it is the maximal among all three values.

According to Table 3.2 in the book, the real stable field of RK4 is [-2.78, 0]. Thus, the maximum step size for numerical integration with Runge-Kutta 4^{th} order is:

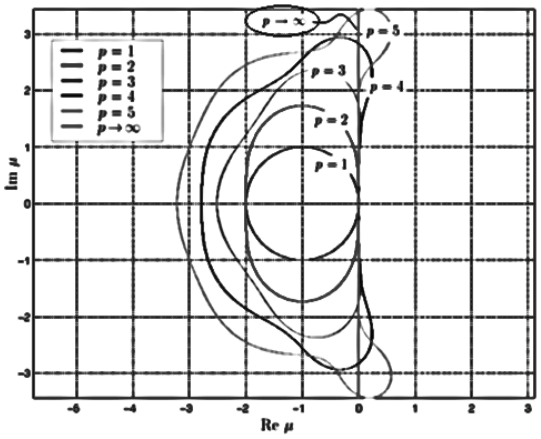
$$h_{crit} = \frac{2.78}{-\lambda_1} = 0.0278$$

which is also the maximum value that satisfies the following condition:

$$\left|1 + \mu + \frac{1}{2}\mu^2 + \frac{1}{3!}\mu^3 + \frac{1}{4!}\mu^4\right| < 1, \quad \mu = \lambda h$$

(b)

To make the step size $h = 10s$ valid, we should extend the RK method. In other words, under RK4 paradigm, since the maximum step size is 0.0278, $h = 10s$ will definitely cause instability in the simulation of the system.



As this figure shows, the higher the order of RK method is, the wider the real stable interval field is. So, when the order p is raised towards ∞, a bigger h will must be acceptable. The fact is that in this case, Runge-Kutta method becomes LIN method. Any $h > 0$ allows high accuracy with the help of the following formula:

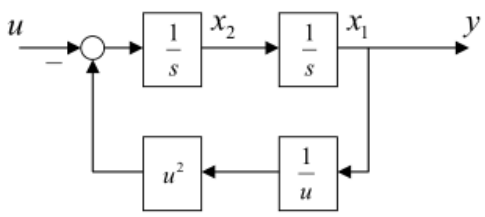
$$\Phi(h) := e^{A \cdot h} = I + h \cdot A + \frac{h^2}{2!} \cdot A^2 + \frac{h^3}{3!} \cdot A^3 + \cdots$$

Appendix—MATLAB Script

```
clear;clc;
disp('Exercise 2-1  Block oriented ODE integration');disp('(a)')
% Building up the transfer function for the system
b1 = tf([1], [0.01,1]);b2 = tf([1], [1, 1]);b3 = tf([1], [1, 0]);
tfModel = (b1-b2)*feedback(b3, 1)
% Calculating the lambda value of the system
ssModel = ss(tfModel);A = ssModel.a;B = ssModel.b;
% the max() function actually returns the smallest one because:
% 1) eig(A) returns an array of complex numbers with 0 imaginary part
% 2) the real part of these numbers is all minus
mainLambda = max(eig(A))
% The real stable field of RK4 is [-2.78, 0], thus:
h_crit = 2.78 / -mainLambda
```

Exercise 2-2 Block oriented ODE integration

(a)



Translating the block model into differential equation:

$$\begin{cases} \dot{x_1} = x_2 \\ \dot{x_2} = \dfrac{1}{x^2} - u \end{cases}$$

$$\mathrm{y} = \mathrm{x}_1$$

The model above is exactly the state space model we need.

(b)

Local linearization near time $\mathrm{t} = 0$ by applying Jacobian matrix:

$$\mathrm{J}_1(0) = \begin{bmatrix} 0 & 1 \\ -2 & 0 \end{bmatrix}, \ \mathrm{x}_1(0) = 1, \ \mathrm{x}_2(0) = 0$$

$$\mathrm{J}_1(0) = \begin{bmatrix} 0 & 1 \\ -250 & 0 \end{bmatrix}, \ \mathrm{x}_1(0) = 0.2, \ \mathrm{x}_2(0) = 0$$

We should ensure $|\varphi_{\mathrm{R}}(\mu)| < 1$ to make a stable simulation. In RK4, $\varphi_{\mathrm{R}}(\mu) = 1 + \mu + \frac{1}{2}\mu^2 + \frac{1}{3!}\mu^3 + \frac{1}{4!}\mu^4$.

The h value is known as 0.5s.

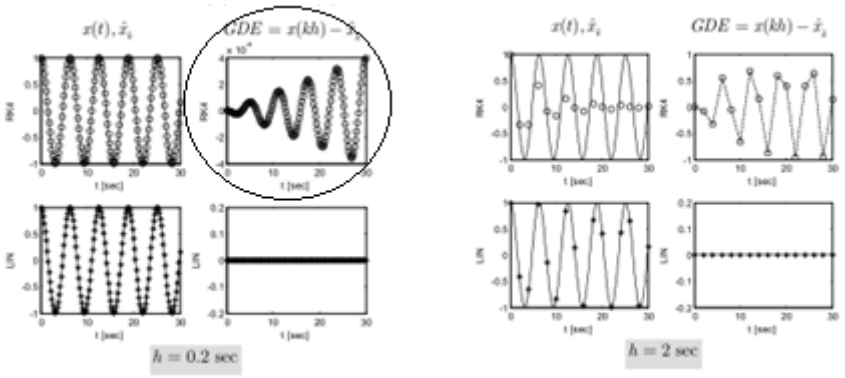
<1>

The eigenvalues of $\mathrm{J}_1(0)$ are: $\lambda_1 = 1.4142\mathrm{i}, \lambda_2 = -1.4142\mathrm{i}$.

Using either one to calculate $|\varphi_{\mathrm{R}}(\mu)|$:

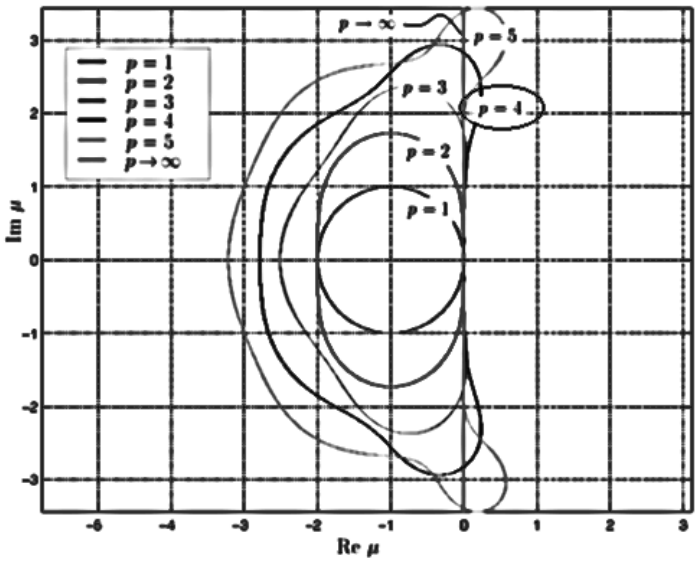
$$|\varphi_{\mathrm{R}}(\mu)| = \left|1 + \mu + \frac{1}{2}\mu^2 + \frac{1}{3!}\mu^3 + \frac{1}{4!}\mu^4\right| = |0.7604 \ + \ 0.6482\mathrm{i}| = 0.9992 < 1$$

However, as a low damped system, it is not enough to make the simulation stable by making $|\varphi_{\mathrm{R}}(\mu)| < 1$. The GDE is relatively big since $|\varphi_{\mathrm{R}}(\mu)|$ is too close to 1, which means it will cause nonnegligible error after a period of time. The situation of the GDE of this system will just be like what the circuited part in the following figure from the slide shows.



<2>

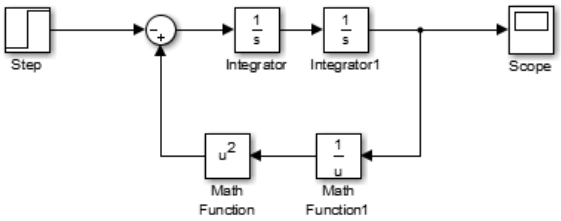
The eigenvalues of $\mathrm{J}_2(0)$ are: $\lambda_1 = 15.8114\mathrm{i}, \ \lambda_2 = -15.8114\mathrm{i}$, with corresponding $\mu_1 = 7.9057\mathrm{i}, \mu_2 = -7.9057\mathrm{i}$.



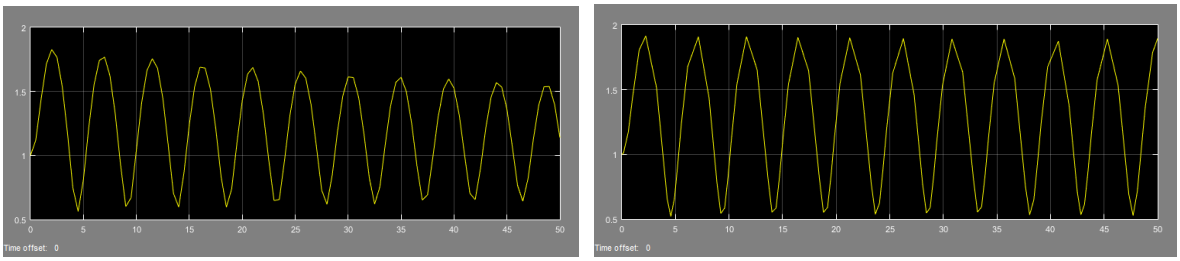
It is obvious according to the figure that both μ are out of the stable field of the RK4 method. So, the simulation result must be wrong.

Appendix—SimuLink Simulation

Simulating the system by SimuLink:

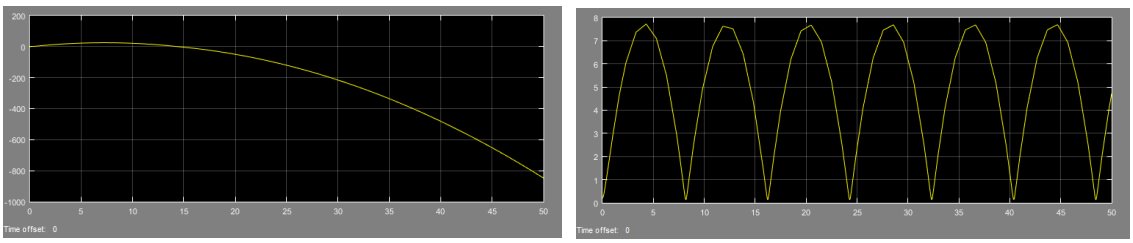


<1>



The left part is with fixed step size $h = 0.5s$, and the right part is with variable size (we assume this result represents the reality). It can be revealed that GDE of the left part grows as the time pass, which proves the idea of the previous discussion.

<2>



It is obvious that because of an invalid h value, the simulation in the exercise (left part) goes definitely wrong comparing to the correct result (right part).

Exercise 2-3 DC-motor simulation

Philosophy: At the beginning instant the variables of the system changes the fastest, and as time goes by the change becomes slow eventually.

(a)

We assume that the maximum step size appears at the beginning instant.

Local linearization:

$$\mathrm{J}(0)=\begin{bmatrix}-45.75-0.525x_2 & -0.525x_1\\ 2x_1 & 0\end{bmatrix}=\begin{bmatrix}-45.75 & 0\\ 0 & 0\end{bmatrix},\ \mathrm{x}_1(0)=0,\ \mathrm{x}_2(0)=0$$

The eigenvalues of $\mathrm{J}(0)$ are: $\lambda_1=-45.75,\lambda_2=0$.

$$\mathrm{h}_{\mathrm{crit}}=\frac{2.78}{-\lambda_1}=0.0608$$

(b)

We focus on the time period when $\mathrm{t}\to\infty$, the slope being close to 0:

$$\begin{cases}0=-45.75x_1-0.525x_1x_2+1027.5u_1\\ 0=x_1^2-1500u_2\end{cases}$$

The results are: $\mathrm{x}_1(\infty)=17.32,\ \mathrm{x2}(\infty)=25.86$.

Comparing the results above and the given figures, we can conclude that the simulation result of x_1 is more likely to be correct since the value finally stays around 17.32. On the contrary, the simulation result of x_2 is highly suspicious because it keeps its value at a state more than 30 instead of being 25.86 in the late time.

Appendix—MATLAB Script

```
clear;clc;
disp('Exercise 2-3 DC-motor simulation');disp('(a)')
j0=[-45.75, 0; 0, 0];
mainLambda = min(eig(j0))
h_crit = 2.78 / -mainLambda
```

Exercise 2-4 Simulation of a pendulum

Specially, when value φ is close enough to 0, we can replace $\sin\varphi$ by a linear φ.
In this case, the state space module of the system is:

$$\begin{bmatrix}\dot{\varphi}\\ \dot{\omega}\end{bmatrix}=\begin{bmatrix}0 & 1\\ -\frac{g}{l} & -k\end{bmatrix}\begin{bmatrix}\varphi\\ \omega\end{bmatrix}=\begin{bmatrix}0 & 1\\ -9.81 & -0.1\end{bmatrix}\begin{bmatrix}\varphi\\ \omega\end{bmatrix}$$

The eigenvalues: $\lambda_1=-0.0500+3.1317\mathrm{i}$, $\lambda_1=-0.0500-3.1317\mathrm{i}$.
Calculating the max h value by:

$$|\varphi_R(\mu)|=\left|1+\mu+\frac{1}{2}\mu^2+\frac{1}{3!}\mu^3+\frac{1}{4!}\mu^4\right|<1,\ \ \mu=h\lambda$$

The result is: $h_{crit}=0.91$.

Generally, given any $\varphi(0)$, do local linearization at the initial time:

$$J(0)=\begin{bmatrix}0 & 1\\ -\frac{g}{l}cos\varphi(0) & -k\end{bmatrix}=\begin{bmatrix}0 & 1\\ -9.81cos\varphi(0) & -0.1\end{bmatrix}$$

First, we should get the eigenvalues from $J(0)$. Second, we can determine the maximum possible step size by calculating $|\varphi_R(\mu)|<1$.
With the help of MATLAB, we do can figure out the h_{crit} (after endless computing!).

Appendix—MATLAB Script

```
% findMaxH.m
function h_crit = findMaxH()
    h_crit = -1;
    for angle = 0:0.1:90
        J_0 = [0, 1; -9.81*cos(angle/180*pi), -0.1];
        mainLambda = min(eig(J_0));
        h_crit = min(calH(mainLambda), h_crit);
    end
end

function h = calH(lambda)
    accuracy = 1e-4;
    h_upField = 10;
    for h = h_upField-4:-accuracy:accuracy
        fai = calFai(h*lambda);
        if abs(fai) < 1 && abs(fai) >= 0
            break
        end
    end
end

function fai = calFai(miu)
    fai = 1+miu+0.5*miu^2+1/6*miu^3+1/24*miu^4;
end
```