Numerical Simulation of Dynamic Systems: Hw1 - Solution

Prof. Dr. François E. Cellier
Department of Computer Science
ETH Zurich

March 5, 2013

[H1.2] Discretization of State Equations

Given the following explicit ODE model:

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{A}\cdot\mathbf{x}+\mathbf{b}\cdot u \\ y &= \mathbf{c}'\cdot\mathbf{x}+d\cdot u \end{aligned}$$

where:

$$\mathbf{A} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -2 & -3 & -4 & -5 \end{pmatrix} \qquad \mathbf{b} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

$$\mathbf{c}' = \begin{pmatrix} 1 & 0 & 0 & 0 \end{pmatrix} \qquad d = 10$$

Engineers would usually call such a model a *linear single-input, single-output (SISO) continuous-time state–space model*.

[H1.2] Discretization of State Equations II

We wish to simulate this model using the following integration algorithm:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + h\cdot\dot{\mathbf{x}}_k$$

which is known as the *Forward Euler (FE) integration algorithm*. If $\mathbf{x}_k$ denotes the state vector at time t^*:

$$\mathbf{x}_k = \mathbf{x}(t)\Big|_{t=t^*}$$

then $\mathbf{x}_{k+1}$ represents the state vector one time step later:

$$\mathbf{x}_{k+1} = \mathbf{x}(t)\Big|_{t=t^*+h}$$

[H1.2] Discretization of State Equations III

Obtain an *explicit difference equation (ΔE) model* by substituting the state equations into the integrator equations. You obtain a model of the type:

$$\begin{aligned} \mathbf{x}_{k+1} &= \mathbf{F}\cdot\mathbf{x}_k+\mathbf{g}\cdot u_k \\ y_k &= \mathbf{h}'\cdot\mathbf{x}_k+i\cdot u_k \end{aligned}$$

which engineers would normally call a *linear single-input, single-output (SISO) discrete-time state-space model*.

Let $h = 0.01\ sec$, $t_f = 5\ sec$, $u(t) = 5\cdot sin(2t)$, $x_0 = \mathrm{ones}(4,1)$, where t_f denotes the final time of the simulation.

Simulate the ΔE model using MATLAB by iterating over the difference equations. Plot the output variable as a function of time.

[H1.2] Discretization of State Equations IV

We merge the model and solver equations:

$$\begin{aligned} \dot{\mathbf{x}}_\mathbf{k} &= \mathbf{A} \cdot \mathbf{x}_\mathbf{k} + \mathbf{b} \cdot u_k \\ y_k &= \mathbf{c}' \cdot \mathbf{x}_\mathbf{k} + d \cdot u_k \\ \mathbf{x}_{\mathbf{k}+1} &= \mathbf{x}_\mathbf{k} + h \cdot \dot{\mathbf{x}}_\mathbf{k} \end{aligned}$$

By substitution, we obtain:

$$\begin{aligned} \mathbf{x}_{\mathbf{k}+1} &= \mathbf{x}_\mathbf{k} + h \cdot (\mathbf{A} \cdot \mathbf{x}_\mathbf{k} + \mathbf{b} \cdot u_k) \\ y_k &= \mathbf{c}' \cdot \mathbf{x}_\mathbf{k} + d \cdot u_k \end{aligned}$$

Therefore:

$$\begin{aligned} \mathbf{x}_{\mathbf{k}+1} &= (\mathbf{I}^{(\mathbf{n})} + \mathbf{A} \cdot h) \cdot \mathbf{x}_\mathbf{k} + (\mathbf{b} \cdot h) \cdot u_k \\ y_k &= \mathbf{c}' \cdot \mathbf{x}_\mathbf{k} + d \cdot u_k \end{aligned}$$

is the equivalent discrete-time model.

[H1.2] Discretization of State Equations V

I simulated the original continuous-time model using Matlab's lsim function, and I simulated the equivalent discrete-time model by iterating on the difference equations.

I then plotted the two solution trajectories on top of each other.

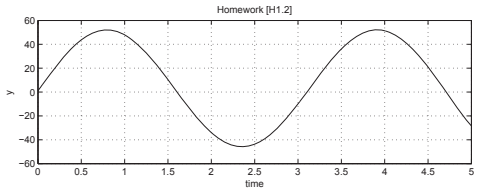

Figure: Simulation results

To the naked eye, the two solution trajectories are indistinguishable from each other.

[H1.4] Van-der-Pol Oscillator and Time Reversal

Given the following non-linear system:

$$\ddot{x} - \mu(1 - x^2)\dot{x} + x = 0$$

This system exhibits an oscillatory behavior. It is commonly referred to as the *Van-der-Pol oscillator*. We wish to simulate this system with $\mu = 2.0$ and $x_0 = \dot{x}_0 = 0.1$.

Draw a block diagram of this system. The output variable is x. The system is autonomous, i.e., it doesn't have an input variable.

Derive a state-space description of this system. To this end, choose the outputs of the two integrators as your two state variables.

[H1.4] Van-der-Pol Oscillator and Time Reversal II

Simulate the system across 2 *sec* of simulated time. Since the system is non-linear, you cannot use MATLAB's lsim function. Use function ode45 instead.

At time $t = 2.0$ *sec*, apply the time reversal algorithm, and simulate the system further across another 2 *sec* of simulated time. This is best accomplished by adjusting the model such that it contains a factor c in front of each state equation. $c = +1$ during the first 2 *sec* of simulated time, and $c = -1$ thereafter. You can interpret c as an input variable to the model. Make sure that $t = 2.0$ *sec* defines an output point.

As you simulate the system backward through time for the same time period that you previously used to simulate the system forward through time, the final values of your two state variables ought to be identical to the initial values except for numerical inaccuracies of the simulation. Verify that this is indeed the case. How large is the accumulated error of the final values? The accumulated simulation error is defined as the norm of the difference between final and initial values.

Plot $x(t)$ and $\dot{x}(t)$ on the same graph.

[H1.4] Van-der-Pol Oscillator and Time Reversal III

Repeat the previous experiment, this time simulating the system forward during 20 *sec* of simulated time, then backward through another 20 *sec* of simulated time.

What do you conclude?

[H1.4] Van-der-Pol Oscillator and Time Reversal IV

The first part of the problem with time reversal at time 2 *sec* could be simulated without any problem. The time reversal worked without a snitch. The accumulated simulation error turns out to be:

$$err = \|x_f - x_0\| = 5.003 \cdot 10^{-6}$$

The remaining error was probably caused by an inaccuracy in the switching time, but the accuracy obtained is certainly quite acceptable for most engineering purposes.

The second part of the problem caused problems. The simulation died shortly after the time reversal. This is the first non-linear model that we simulate, and already we seem to be in difficulties!

What happened?

[H1.4] Van-der-Pol Oscillator and Time Reversal V

The original problem exhibits a *stable limit cycle* in the *phase plane*, i.e., in the plane spanned by $x2(x1)$. Every trajectory ends up on the limit cycle.

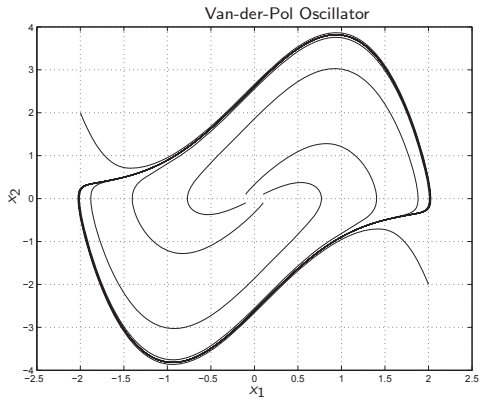

Figure: Trajectory behavior of Van-der-Pol oscillator

[H1.4] Van-der-Pol Oscillator and Time Reversal VI

The time-reversed problem thus exhibits a *finite analytical domain of stability* that coincides with the limit cycle of the original problem. Every trajectory starting within that region of the phase plane ends up at the origin, whereas every trajectory beginning outside that region escapes to infinity.

When we simulate over 2 *sec* with initial conditions of $x_1 = x_2 = 0.1$, we are still far away from the limit cycle at switching time. Hence the time-reversed trajectory returns on its old path back to where it had come from.

On the other hand, when we simulate the original problem during 20 *sec* of simulated time, we end up very close to the limit cycle. After switching, small changes in the initial conditions will have huge effects on the simulation outcome. Even fairly small inaccuracies during one step may carry us across the border into the unstable region, from where the trajectory will quickly escape. This is exactly what happened.

[H1.4] Van-der-Pol Oscillator and Time Reversal VII

Thus, I needed to reduce the relative error to:

$$reltol = 10^{-9}$$

to make sure that the integration steps remain sufficiently small so that the border won't be crossed. Now the simulation no longer died.

[H1.4] Van-der-Pol Oscillator and Time Reversal VIII

Unfortunately, the results are still incorrect, as the trajectory does not follow its original path back to where it came from.

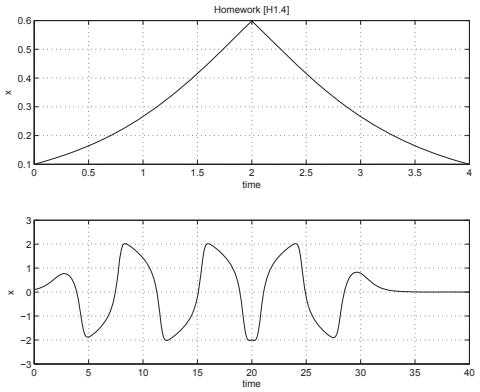

Figure: Simulation results

[H1.4] Van-der-Pol Oscillator and Time Reversal IX

The reason for this disappointing result has to do with the *very large sensitivity of the time-reversed problem to initial conditions* that are close to the limit cycle. Thus, small numerical errors committed during the first steps of the time-reversed simulation have a huge effect on the further development of the trajectory. This problem is unsolvable within the constraints of the given machine resolution.