Power System Dynamics

Lab Session 1

Responsible instructor: Dr. -Ing. J. L. Rueda Torres *
Responsible for lab instructions: Ir. A. A. van der Meer†
Lab Instructor: Ir. M. Ndreko‡
Lab Instructor: I. Tyuryukanov, M.Sc.§

April 22, 2015

Introduction

In this lab course, part of the course *Power System Dynamics* (ET 4113), we are going to implement a dynamic model of the synchronous machine (SM), its associated controls, and a simplified model of a wind generation system (WGS). The lab classes will be organised as follows:

Wednesday 22 April 2015: Introduction to MATLAB;

Wednesday 13 May 2015: Initial value calculation, 5^{th} order SM model;

Wednesday 20 May 2015: 5^{th} order SM model (continuation);

Wednesday 3 June 2015: Voltage and frequency control;

Wednesday 10 June 2015: Eigenvalue analysis of 5^{th} order SM model;

Wednesday 17 June 2015: Wind generating system;

Although this lab is not compulsory, attendance is strongly recommended. This is because the evaluation for the course is partly based on a written report describing the outcomes and supplementary assignments of each lab session.

The lab instructions start at 08:45 and end at 12:45

All models will be written in MATLAB, version R2013B. It is assumed that you are at least familiar with MATLAB; the first lab session aims at a quick introduction on

*EEMCS Faculty, LB 03.490, tel.: 86239, e-mail: J.L.RuedaTorres@tudelft.nl.
†EEMCS Faculty, LB 03.210, tel.: 88007, e-mail: a.a.vandermeer@tudelft.nl.
‡EEMCS Faculty, LB 03.170, tel.: 84051, e-mail: M.Ndreko@tudelft.nl.
§EEMCS Faculty, LB 03.210, tel.: 88007, e-mail: I.Tyuryukanov@tudelft.nl.

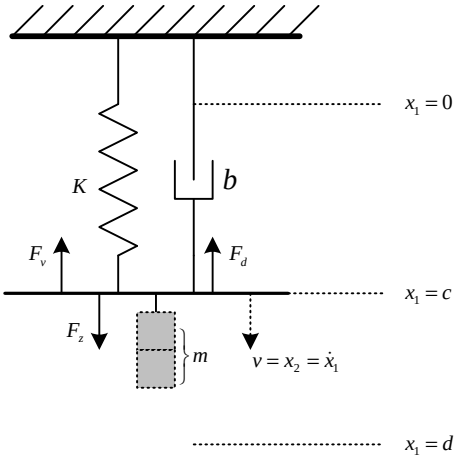

Figure 1: Mass-spring-damper system

the use of Matlab for modelling and simulation of dynamic systems as conceived in the theoretical lectures of the course. Self-learning after this session and outside the lectures is strongly recommended, since thorough teaching of Matlab is not in the scope of this course.

In addition, several introductory materials to this mathematical programming framework are placed on the Blackboard system under "Course Documents". This also applies to the MATLAB scripts containing the correct implementation, which we will make available after 4 working days from the corresponding lab session.

1 Illustrative Modelling Approach: the mass-spring-damper system

To get you more familiar with the modelling strategy that is used during the lab sessions as well as during your final exam, we start with a simple mass-spring-damper system. First, the model itself is derived and the exact solution is given. Then, the implementation in MATLAB is addressed. If you are already familiar with differential equations and how to solve these analytically, you may skip the next section.

Although not very much related to power system dynamics or to power engineering in general, the simulation of a mass-spring-damper system resembles much of the modelling approach used in this course.

1.1 Model and Exact Solution

We consider the mass-spring-damper system shown in Fig. 1. A mass m [kg] is attached to the reference by a spring with spring constant K [N/m]. It consists of two parts of equal mass. The vertical displacement of the mass, which is denoted by $x_1(t)$, ($x_1(t) > 0$ indicates direction downwards, origin is at the point no suspension force acts on the mass). The damper constant is b [N s/m].

One can easily imagine that the mass will swing from one position to another when it is released from an initial position that differs from the equilibrium point. We

are interested in investigating the behaviour of this system accurately. Examining this includes two different aspects:

Initial condition: What is the position of the system right before the mass is released? The position of the mass is a measure for how far the spring is compressed or extended. When the initial position of the mass is $x_1 = 0$, it will drop down immediately after release. Then the system swings towards a new position until it comes to rest at $x_1 = c$. On the other hand, if the initial position equals $x_1 = d$, the spring will compress immediately after release and the mass will go upwards first. Eventually, it will swing towards $x_1 = c$ and come to rest as well. Finally, if the initial position is $x_1 = c$, it won't swing at all. Calculation of the forces, displacement and velocity of the mass-spring-damper system is called the initial value calculation.

Dynamic behaviour: The way in which the mass swings towards its final value (steady-state value) depends on the stiffness of the spring, the amount of damping and the actual mass. The behaviour of the system is determined by the combination of the response of the individual elements within the system. This behaviour can be analysed by Newton's laws of motion. In analogous terms, the dynamics inside an electrical network revolve around Maxwell's laws and the dynamics of a generator can be regarded as a combination of both Newtonian mechanics and electromagnetic phenomena. In all cases, this behaviour is reflected by how quantities that resemble *flow* of energy (current, velocity) change in time due to external *efforts* (force, voltage). Mathematically, the relationships between these quantities are represented by *differential equations*.

There exist several methods to describe the system in Fig. 1 mathematically. In this case, since the entire system is mechanical, the force balance can be used to derive the differential equations. At any time instant it holds that

$$m\ddot{x}_1(t) = \Sigma F = F_z + F_v + F_d = mg - Kx_1(t) - b\dot{x}_1(t) \tag{1}$$

with $\ddot{x}_1(t)$ the total acceleration of the system, F_z the gravitational force, F_v the restoring force of the spring, F_d the damping force (caused by friction), and g the gravitational acceleration constant [m/s^2]. Note that $\dot{x}_1(t)$ means: "take the derivative of $x_1(t)$ with respect to time".

Equation (1) indicates how the position of the mass (x_1), its speed ($\dot{x_1}$) and its acceleration ($\ddot{x_1}$) are related. x_1 And $\dot{x}_1$ are the *state-variables*. It can be seen that the highest derivative is of the second order, which means that this is a second-order differential equation. Since the coefficients (m, g, K, b) are independent of time, the system is called time-invariant. Moreover, the system is linear (i.e. derivatives of the states linearly depend on each other), which allows us to obtain an exact solution quite easily. The model of the synchronous generator is also time-invariant, but it is non-linear.

We will now obtain an exact solution of the previously described mass-spring-damper system. This solution will serve as a verification for the numerical integration method, which will be described later on in Section 1.2. Equation (1) is now

rewritten as follows:

$$m\ddot{x}_1(t) + b\dot{x}_1(t) + Kx_1(t) = F_z = mg \tag{2}$$

Since gravitation acts as an external force, the differential equation is *non-homogeneous*. This means that its solution consists of a *particular solution* and a *homogeneous solution*. The homogeneous solution can be obtained by solving

$$m\ddot{x}_{1,h}(t) + b\dot{x}_{1,h}(t) + Kx_{1,h}(t) = 0 \tag{3}$$

for $x_{1,h}(t)$. A good choice in this case is $x_{1,h}(t) = e^{\lambda t}$ because its derivatives ($\dot{x}_{1,h}(t) = \lambda e^{\lambda t}$ and $\ddot{x}_{1,h}(t) = \lambda^2 e^{\lambda t}$) resemble the original function quite well. Substitution into (3) yields

$$\left(m\lambda^2 + b\lambda + k\right) e^{\lambda t} = 0 \tag{4}$$

The corresponding *characteristic equation* is:

$$m\lambda^2 + b\lambda + k = 0 \tag{5}$$

from which λ can be solved by the quadratic formula

$$\lambda_{1,2} = \frac{-b \pm \sqrt{b^2 - 4mK}}{2m} \tag{6}$$

Values λ_1 and λ_2 are the *eigenvalues* of the mass-spring-damper system. Note that the eigenvalues can be either real or complex conjugated, depending on the sign of the discriminant in (6). As will become clear during future lab-courses, eigenvalues provide a lot of information about the dynamic response of the system. However, because of its non-linearity, a synchronous generator model must be linearised first in order to obtain these eigenvalues.

In general, λ_1 and λ_2 are distinct roots. Then, the homogeneous solution of the differential equation becomes

$$x_{1,h}(t) = C_1 \mathrm{e}^{\lambda_1 t} + C_2 \mathrm{e}^{\lambda_2 t} \tag{7}$$

where C_1 and C_2 are constants that depend on the initial conditions. When λ_1 and λ_2 are equal, i.e. λ has *multiplicity* 2, an homogeneous solution of the form

$$x_{1,h}(t) = (C_1 + C_2 t)\, \mathrm{e}^{\lambda t} \tag{8}$$

must be sought for. The text below considers only the case $\lambda_1 \neq \lambda_2$. The other solution is left as an exercise, but will be included in the m-files to be provided on the Blackboard system after this lab session.

The particular solution is obtained as follows: First, a solution $x_{1,p}(t)$ must be chosen such that it looks somewhat like (has the same order as) the *driving function*, which is the gravitational force in this case. Then, this solution is substituted into the original differential equation, in this case (2), and rearranged to obtain the correct coefficients.

Since $F_z = mg$ is a polynomial of order 0, $x_{1,p}(t) = A$ where A is a constant, is chosen as a particular solution. Substitution into (2) yields

$$
\begin{aligned}
m\ddot{x}_{1,p}(t) + b\dot{x}_{1,p}(t) + Kx_{1,p}(t) &= mg \\
KA &= mg \\
x_{1,p}(t) = A &= \frac{mg}{K}
\end{aligned}
$$

which is added to the homogeneous solution to obtain the general solution $x_1(t)$, which fully describes how the mass-spring-damper system will swing to its final point.

$$x_1(t) = x_{1,h}(t) + x_{1,p}(t) \tag{9}$$

$$x_1(t) = C_1 \mathrm{e}^{\lambda_1 t} + C_2 \mathrm{e}^{\lambda_2 t} + \frac{mg}{K} \tag{10}$$

As can be seen, C_1 and C_2 must still be determined. These can in fact be calculated because we know the state of the system at $t = 0\,\mathrm{s}$, namely $x_1 = 0$ and $\dot{x}_1 = 0$.

$$
\begin{aligned}
x_1 = 0 \quad &\Longrightarrow \quad 0 = \frac{mg}{K} + C_1 + C_2 \quad &&\Longrightarrow \quad \frac{-mg}{K} = C_1\left(1 - \frac{\lambda_1}{\lambda_2}\right) \\
\dot{x}_1 = 0 \quad &\Longrightarrow \quad 0 = C_1\lambda_1 + C_2\lambda_2 \quad &&\Longrightarrow \quad C_2 = \frac{-C_1\lambda_1}{\lambda_2}
\end{aligned}
$$

$$C_1 = \frac{-mg}{K\left(1 - \frac{\lambda_1}{\lambda_2}\right)} \tag{11}$$

$$C_2 = \frac{mg\lambda_1}{K\lambda_2\left(1 - \frac{\lambda_1}{\lambda_2}\right)} \tag{12}$$

Now, all coefficients of $x_1(t)$ are determined and an exact solution of a mass-spring-damper system can be calculated for any given m, K and b. In the next section we will provide a different method to calculate the behaviour of the system based on numerical integration. In light of the simulation of power system dynamic behaviour this will also be necessary, since an analytical solution is not readily available for complex non-linear models.

1.2 Numerical Solution

The mass-spring-damper system can be solved exactly as described in the previous paragraph. Many systems, such as the power system, are described by non-linear models and are therefore difficult, if not impossible, to solve analytically. In such cases, a solution can only be approximated numerically. The accurateness of these solutions highly depends on the applied method and the integration step size. To get you familiar with the structure of the MATLAB files that will be used in the lab-courses, the mass-spring-damper system is solved with an easily understandable

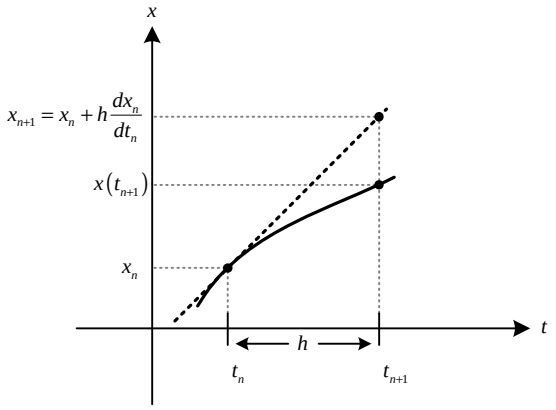

Figure 2: The forward Euler method

numerical integration method: the forward Euler method. This method works as follows: The time-interval of interest, T, is discretized into time steps of size h. Then, each time step (we start at $t_1 = 0\,\mathrm{s}$) the derivatives of the state variables must be calculated. To achieve this, (2) is rewritten as

$$\ddot{x}_1(t) = \frac{mg - b\dot{x}_1(t) - Kx_1(t)}{m} \tag{13}$$

and, after replacing the dot notation by the normal differential operator,

$$\frac{\mathrm{d}^2 x_1(t)}{\mathrm{d}t^2} = \frac{-b}{m}\frac{\mathrm{d}x_1(t)}{\mathrm{d}t} - \frac{K}{m}x_1(t) + g \tag{14}$$

The speed and position of the mass determine the state of the system and are therefore chosen as the state variables. Before we continue to the integration scheme, (14) is written into *state-space form* by separating the *dependent* variables, namely the state variables, and the *independent* variables, in this case the gravitational force. The state-space representation also involves rewriting an n-dimensional differential equation into a system of n first-order equations [1]. For the mass-spring-damper system this can be expressed as

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{pmatrix} x_1(t) \\ x_2(t) \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ \frac{-K}{m} & \frac{-b}{m} \end{pmatrix}\begin{pmatrix} x_1(t) \\ x_2(t) \end{pmatrix} + \begin{pmatrix} 0 \\ g \end{pmatrix} \tag{15}$$

with $x_2(t) = \frac{\mathrm{d}x_1(t)}{\mathrm{d}t}$.

At each time step t_n, the left-hand side of (15) is calculated using values of the *state vector* $\mathbf{x}_n = [x_{1,n} \quad x_{2,n}]^T$ obtained from the previous time step t_{n-1}[2]. This is shown in Fig. 2 for just one element of the state vector. This derivative can be used to construct a tangent line (the dashed line). Using this tangent line, the state vector for the *next* time step is approximated by

$$\mathbf{x}_{n+1} = \mathbf{x}_n + h\frac{\mathrm{d}\mathbf{x}_n}{\mathrm{d}t_n} \tag{16}$$

[1] G. F. Franklin. *Feedback Control of Dynamic Systems*. Prentice Hall, 4 edition, 2002.

[2] For t_1 the initial conditions are the 'previous values'.

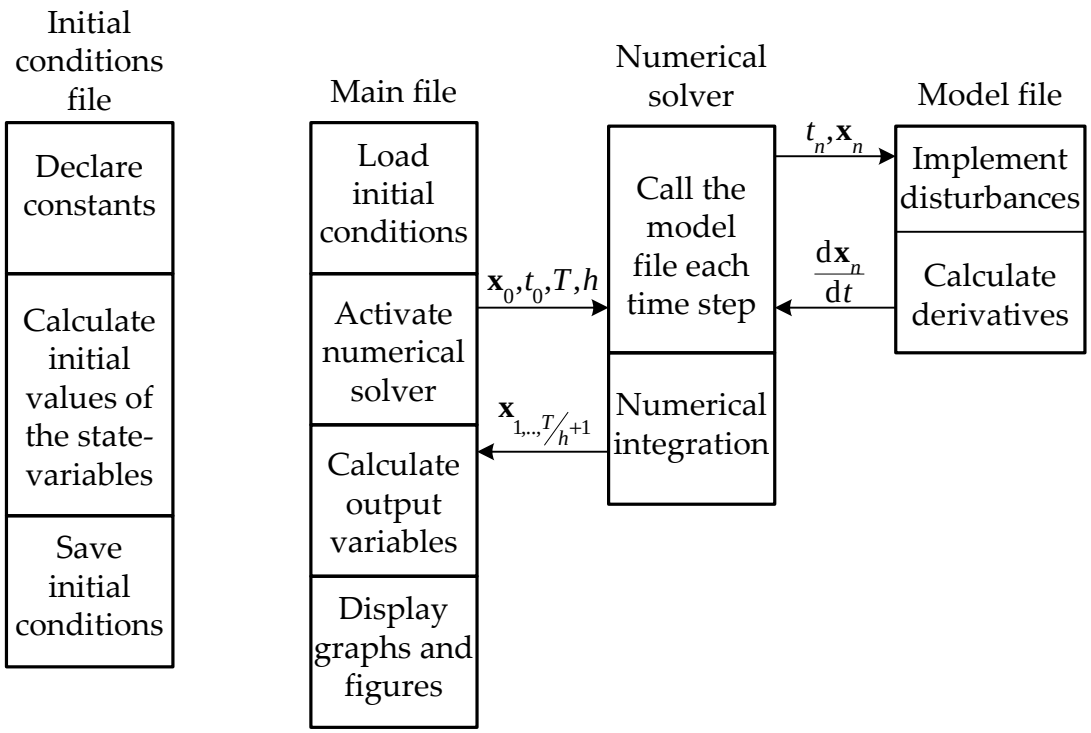

Figure 3: The structure of the MATLAB files that will be used.

Subsequently, the tangent line is calculated again using t_{n+1} and x_{n+1} and the next value of the state-variable, x_{n+2} is calculated. This process is continued until $n = \frac{T}{h} + 1$. As can be seen in Fig. 2, the approximated value x_{n+1} can differ quite a lot from the exact value $x(t_{n+1})$. This error depends on the used method and the time-step size. In this case, the integration method uses a fixed time-step h. Some other methods calculate the error each time step and reduce (or increase) the step size if necessary. These methods are called *variable step solvers*. These type of solvers will be used during the simulations of future lab-courses.

In the next section we will implement the numerical integration scheme into MATLAB. The exact solution derived in the previous section will be used for verification purposes.

1.3 Implementation in MATLAB

The previous sections outlined two different approaches to solve the mass-spring-damper system. Thanks to its linearity, an exact solution can be obtained in a relatively simple manner. By rewriting the differential equations of the system into state-space form, a numerical integration scheme was developed. Now, both methods will be implemented in MATLAB according to the structure shown in Fig. 3. Each bar is an .m-file, a file that contains MATLAB commands that are executed sequentially. Assume that the mass-spring-damper system has the following parameters:

$$m = 1\,\mathrm{kg} \qquad K = 10\,\mathrm{N\,m^{-1}}$$
$$g = 9.81\,\mathrm{m\,s^{-2}} \qquad b = 1\,\mathrm{N\,s\,m^{-1}}$$

Assignment 1: Write a MATLAB M-file that declares these parameters. Also, calculate the initial conditions. For that, assume that at $t = 0\,\mathrm{s}$ the mass it fixed, i.e. $x_1 = 0$ and $x_2 = 0$. Save the parameters by using the statement `save('mass-spring-init')`. Call the M-file `init.m`. It is also advisable to clear all variables in your workspace at the beginning of your M-file, you can do this by issuing the statement `clear all`.

If you managed to execute the initial conditions file without errors, a file called `mass-spring-init.mat` will be generated. This contains numerical values for all the variables that have been declared in `init.m`. We can load these again to save time efforts. We have now finished the left block of Fig. 3 and can proceed to the main file. As you may see, this file invokes several other files, namely the numerical solver and the file that contains the actual differential equations associated to the model. We will now build both the main file and the model file. The numerical solver can be downloaded from Blackboard and it is called `ode1.m`.

Assignment 2: Create the MATLAB M-file `msmain.m` and load the variables that are stored in `mass-spring-init.mat`. It is wise to load these variables as a *struct*: `in=load('<filename>')`. The mass can now for instance be recalled by typing `in.m` inside the command window. Because these variables are needed throughout the entire program, it is advisable to make them *global*. We have not defined the time interval for the simulation yet. Create an array that contains a time interval from 0 to 10 s, discretized with a time-step size of 10 ms.

Before we continue with the numerical solver in `msmain.m` we will create the file that contains the actual differential equations. This file is called at each time step. It needs the actual time and state-variables as input an it provides the derivatives of these variables as an output.

Assignment 3: Create the MATLAB M-file `msmodel.m`. This file will serve as a *function* that is called by `ode1.m` at each time-step. the syntax is as follows:

```
function dx=msmodel(t,x)
...
end
```

Include the right-hand side of (15) into this file and calculate the output array `dx`. Do not forget to declare `in` as a global variable in this file as well.

Now, the rightmost part of Fig. 3 has also been built. We now continue with `msmain.m` to solve the differential equations numerically.

Assignment 4: Put `ode1.m` into the same directory as `msmain.m`. The syntax for calling `ode1.m` inside `msmain.m` is
`X=ode1(@<functionname>,<time array>,<array with initial values>);`
Note that `X` is a matrix which contains the state-variables for each time step. Now, you may proceed to plot the state variables as a function of time.

If you managed to execute `msmain.m` without errors, you will see that the mass-spring-damper system gently swings to a steady-state value. To check whether this solution is correct or not, we will now include the exact solution as well.

Assignment 5: Include the calculation of the eigenvalues from (6) into `msmain.m`. Do the same for C_1 and C_2 and substitute them into (10). Then, calculate $x_1(t)$ and $x_2(t) = \dot{x}_1(t)$. Add the exact solution to the existing graphs and compare the results. Repeat the entire calculation with a larger time-step size for the numerical integration method. Do you notice any differences?

As you may have seen, the numerical solution resembles the exact solution quite well, especially when the time step-size h is very small. Particularly for non-linear systems with very small *time constants*, the integration method that is used in this example becomes unstable. Therefore, different numerical integration methods (such as `ode23t` or `ode45`) will be introduced in future lab sessions.

1.4 Inclusion of a disturbance

Unlike the mass-spring-damper system, which is described in previous sections, a simulation of a power system or a generator usually starts in an equilibrium point. That means that the derivatives of the state variables equal zero at initial time, i.e. $t = 0\,\mathrm{s}$. For the mass-spring-damper system, this equilibrium point lies at $x_1 = c$. Then, at a predefined instant $t = t_{dist}$ a disturbance occurs, i.e. a system parameter or an independent variable promptly changes. In a power system, these disturbances can for instance be the (dis)connection of a load, a short-circuit, a change in a voltage reference, etc. This section familiarises you with the simulation of disturbances, and we will do this again using the mass-spring-damper system of Fig. 1 by detaching half of the mass at $t = t_{dist}$.

Assignment 6: A disturbance can be regarded as a change in system parameters and will affect the structure of the differential equations. Therefore, a disturbance should be included into `msmodel.m`. This can be done in two ways:

- By an `if-else` construction that halves the mass m for $t \geq t_{dist}$
- By a function `disturbance.m` that returns the relevant system parameters for each time step.

Here, we will use the first approach, whereas the second one is also provided on the Blackboard system. Rewrite the `msmodel.m` file in such a way that the mass is halved after 15 s. Use the same time-step size as in the previous assignments, but adjust the total simulation time in `msmain.m` to 25 s. Account for what you notice. As you may see, the system will swing to an equilibrium point corresponding to the full mass first before the disturbance can be executed. For large systems this may cost a lot of simulation time; how can this be improved? Make the necessary adjustments and reflect on the obtained results.