# 1 Determining the State-Space Form

The target of transforming a given differential equation of **n-th order** into its state form is to transform it into n differential equations of **first** order.

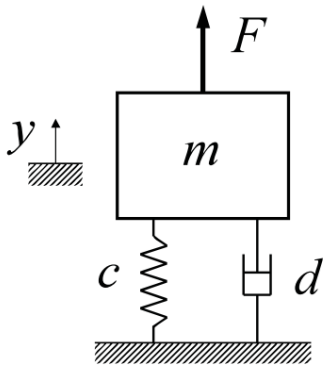


Figure 1: A single mass oscillator system

With the given differential equation of a single degree-of-freedom oscillator:

$$m\ddot{y} + d\dot{y} + cy = F \tag{1}$$

we can use the state vector $x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} y \\ \dot{y} \end{pmatrix}$ to determine the state form.

For this purpose we derive the state vector once, giving us $\dot{x} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \begin{pmatrix} \dot{y} \\ \ddot{y} \end{pmatrix}$.

We can see that

$$\dot{x}_1 = x_2 \tag{2}$$

**Note:**

1. **Explicit Representation**: We sort the differential equation so that the highest derivative stands alone on one side
2. **Implicit Representation**: We sort the differential equation so that everything from the equation is on one side and a zero is on the other side

Furthermore, we can solve equation (1) for $\ddot{y}$, giving us

$$\ddot{y} = \frac{1}{m} * (F - d\dot{y} - cy) \tag{3}$$

to determine the second entry in the derived state vector. Inserting (2) and (3) into the derived state vector and replacing y and with the corresponding entries in the state vector results in the state form needed to solve the differential equation in MATLAB using the ode45-solver:

$$\dot{x} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \begin{pmatrix} \dot{y} \\ \ddot{y} \end{pmatrix} = \dot{x} = \begin{pmatrix} x_2 \\ \frac{1}{m} * (F - dx_2 - cx_1) \end{pmatrix} \tag{4}$$

As you can see, no derivative of a higher order than 1 is left in the equation!

# 2 Defining a state form function in MATLAB

The ode45-solver expects the state form to be delivered in the form a function that returns the derived state vector as a function of the equations we determined before. The solver requires the input parameters to be ordered in a **defined** way: first you need to insert the time, then the state vector, then the parameters needed to specify the equation (random order).

A very common mistake is to mix up the variable order which leads to confusing results (and often long debugging sessions)!

Listing 1: MATLAB Function of the State-Space Form

```
function [ xp ] = StateSpaceForm( t,x,m,c,d,f )
% State space form for single mass oscillator
xp=[x(2); (1/m)*(f-d*x(2)-c*x(1))];

end
```

**x(1)** and **x(2)** call the entries of the state vector, $x_1$ and $x_2$ . The order of parameters in the end (Mass, damping ratio, spring stiffness and force) is not important.

> The function should be saved in another script file called **StateSpaceForm.m** - note that the file should be saved in a **separate** file and that the file name should ideally be identical to the function name!

# 3 Solving the differential equation using the ode45-solver

Before we can solve the equation, we should first set some tolerance parameters. For that, we define a variable, for example:

```
solverOptions = odeset('RelTol', 1e-5, 'AbsTol', 1e-5)
```

By that, we set the relative and absolute tolerance of the numerical solver to $10^{-}5$ each. This allows the solver to choose its time steps autonomously. After this, we can finally solve the differential equation by calling the function

```
[T,Y] = ode45(@StateSpaceForm, tspan, y0, solverOptions, m, d,
   c, f);
```

- **[T,Y]** are the output parameters. The solver writes the time steps itself (or the ones you explicitly forced him to do by fixing a certain step width in the **t** vector) into the vector **T** and the state vectors for each time step into the matrix **Y**. The state vectors are written horizontally each which is very important to note in case you want to concatenate the matrices later on. You can double click the variables in the workspace to have a look yourself
- **@StateSpaceForm** inputs the state form you determined before into the solver
- **tspan** is either a vector containing only a start and an end time or a vector containing all the time steps, forcing the solver to use the time steps you chose
- **y0** declares the initial conditions of the system in state vector form: $\vec{y_0} = \begin{pmatrix} y_0 \\ \dot{y}_0 \end{pmatrix}$
- **solverOptions** passes the tolerances we set earlier on to the solver
- The equation parameters are stated afterwards. **They must be mentioned in the same order as defined in the StateSpaceForm function!**

**Congratulations**! You successfully solved the differential equation and you're on your way to become a MATLAB master! ;-)

In case you have any questions you can post questions below my MATLAB videos on my YouTube channel or contact me via social media - I prefer the first option though! :-P

P.S.: You can find all the code needed in my **repository** as well as a detailed explanation and walk through of the code on my **YouTube Page**