2.004 Dynamics and Control II
Spring 2008

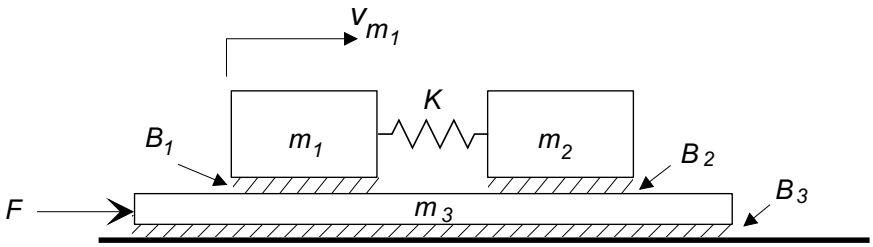
Massachusetts Institute of Technology
Department of Mechanical Engineering

2.004 *Dynamics and Control II*
Spring Term 2008

Lecture 11[1]

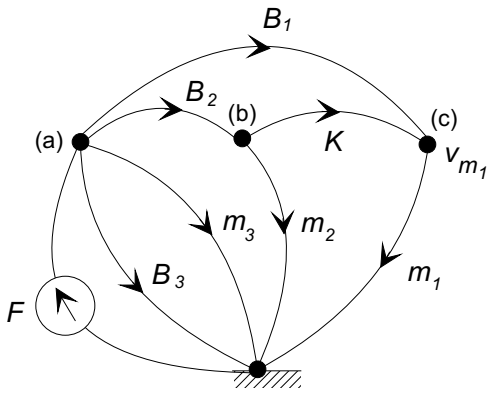
Classroom Example: Symbolic Transfer Function Generation Using MATLAB and Maple

Consider the mechanical system shown below:



The task is to find the transfer function relating the velocity of mass m_1 to the input force $F(s)$ using the symbolic math package Maple.

The system linear graph is shown below:



We start with the node equation method:

(a) Write the continuity equations at the nodes (a), (b), and (c):

$$\begin{aligned} F - F_{B_1} - F_{B_2} - F_{B_3} - F_{m_3} &= 0 \\ F_{B_2} - F_K - F_{m_2} &= 0 \\ F_{B_1} + F_K - F_{m_1} &= 0 \end{aligned}$$

(b) Substitute for the elemental forces using elemental impedance relationships (in this case it is more convenient to use admittances $Y = 1/Z$) and velocities at the nodes.

$$\begin{aligned} F - Y_{B_1}(v_a - v_c) - Y_{B_2}(v_a - v_b) - Y_{B_3}v_a - Y_{m_3}v_a &= 0 \\ Y_{B_2}(v_a - v_b) - Y_K(v_b - v_c) - Y_{m_2}v_b &= 0 \\ Y_{B_1}(v_a - v_c) + Y_K(v_b - v_c) - Y_{m_2}v_c &= 0 \end{aligned}$$

where $Y_{m_1} = m_1 s$, $Y_{m_2} = m_2 s$, $Y_{m_3} = m_3 s$, $Y_{B_1} = B_1$, $Y_{B_2} = B_2$, $Y_{B_3} = B_3$, and $Y_K = K/s$.

(c) These three linear algebraic equations are used to solve for $v_a = v_{m_1}$, which defines the transfer function.

While the solution is easy in principle, the algebraic manipulations become messy and so we look at the use of symbolic math software contained in MATLAB and Maple to assist in the solution:

- Using MATLAB's Symbolic Toolbox,
- Using Maple directly, and
- Using the software package Maple Syrep.

Using MATLAB's Symbolic Toolbox to Find the Transfer Function:

The Symbolic Toolbox is an optional add-on to MATLAB. It is a sub-set of Maple, and allows the symbolic solution of linear algebraic equations, such as required in this example. The following is a "'m" file, `Lecture11Example.m` that illustrates how MATLAB can be used:

```
%   2.004 Spring 2008
%   Classroom Example: Symbolic Transfer Function Generation
%                      Using MATLAB's Symbolic Toolbox.
%
%---------------------------------
% Declare all of the symbolic variables as 'syms'.   The specifier 'real'
% at the end says that we expect all of these variables to be real.
syms  s F va vb vc YB1 YB2 YB3 Ym1 Ym2 Ym3 YK B1 B2 B3 m1 m2 m3 K real
% Solve for the three nodal velocities using the nodal equations
[va, vb, vc] = solve('F-YB1*(va -vc) - YB2*(va -vb) - YB3*va - Ym3*va = 0',...
                     'YB2*(va-vb) - YK*(vb -vc) - Ym2*vb = 0',...
                     'YB1*(va-vc) + YK*(vb-vc) - Ym1*vc = 0');
% The output variable of interest is vc.  Substitute for the impedance of
% all of the elements (Note - we could have made the substitutions directly
% into the equations above and saved these steps):
vc=subs(vc,YB1,B1);
vc=subs(vc,YB2,B2);
vc=subs(vc,YB3,B3);
```

```
vc=subs(vc,Ym1,m1*s);
vc=subs(vc,Ym2,m2*s);
vc=subs(vc,Ym3,m3*s);
vc=subs(vc,YK,K/s);
% THe transfer function is vc/F
H=vc/F;
% The rest is "fluff".  Use "simplify(H)" to make H a rational fraction,
% use "collect(H,s)" to collect all of the coefficients for each power of s,
% use "sort(H)" to order the terms in descending powers of s.
H = sort(collect(simplify(H),s))
```

The following is a MATLAB command line fragment showing the output from the ".m" file, and how numerical values are substituted into the result:

```
>>
>> Lecture11Example
H =
(s^2*B1*m2+s*B1*B2+B1*K+B2*K)/(s^4*m1*m3*m2
   +(B3*m1*m2+m1*B1*m2+m1*B2*m3+m1*B2*m2+B1*m3*m2)*s^3
   +(B3*m1*B2+B3*B1*m2+m1*B1*B2+m1*m3*K+B1*B2*m3+B1*B2*m2+m3*K*m2)*s^2
   +(B3*m1*K+B3*B1*B2+B3*K*m2+m1*B1*K+m1*B2*K+B1*m3*K+B1*K*m2+B2*m3*K+B2*K*m2)*s
   +B3*B1*K+B3*B2*K)
>>
>> m1=3; m2=4; m3=5; B1=1; B2=2; B3=3; K=1000;
>> H=subs(H)
H =
(4*s^2+2*s+3000)/(60*s^4+122*s^3+9000+35054*s^2+57006*s)
>>
```

Using Maple to Find the Transfer Function:

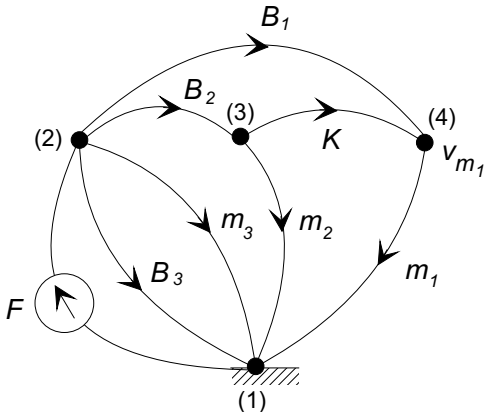
The following page shows a Maple 11 "classic" work-sheet to solve the algebraic equations. This example uses the procedures `GenerateMatrix()` and `LinearSolve()` from the `LinearAlgebra` package.

- Maple uses a ":" to end a command if no output is to be generated, or a";" if the output is to be displayed.
- The `LinearAlgebra` package is loaded using the command `with(LinearAlgebra):`
- Commands are grouped within *execution groups* indicated at the very left margin. A group is executed by placing the cursor within a group and hitting *Enter*.
- `GenerateMatrix()` requires that the equations and the output variables entered as lists, that is `[item1, item2, ...]`.
- `LinearSolve(A,b)` solves linear equations in matrix form $\mathbf{Ax} = \mathbf{b}$ where $\mathbf{A}$, and $\mathbf{b}$ have been generated from the equations using `GenerateMatrix()`.

Using *Maple Syrep* to Find the Transfer Function:

Maple Syrep (for *Sy*stem *Rep*resentation is a collection of approximately 100 functions for symbolic and numerical analysis of dynamic systems. In this handout we examine two of its functions for deriving a system directly from its linear graph or impedance graph.

We start using the graph for the system, but this time numbering each node:



The system is specified as a Maple list-of-lists, each contained in square brackets, for example for the above diagram:

```
graph:=[[1,2,Force,Fs],
        [2,1,mass,m3],
        [2,1,damper,B3],
        [2,3,damper,B2],
        [2,4,damper,B1],
        [3,1,mass,m2],
        [4,1,mass,m1,vm1,Fm1],
        [3,4,spring,K]]:
```

Each entry in the list represents a branch with the first two items the nodes that it connects to (in the direction of the arrow), the third describes the branch, the fourth the parameter, and optionally two items naming the variables on the branch.

The system is generated using the procedures

- `LGraph_to_system(graph,output)` - if the system is represented in linear graph form, or
- `ZGraph_to_system(graph,output)` - if the system is represented in impedance graph form (where the elemental impedances are used).

Note: The newer versions of Maple allow a new mode of input, the *2D-Math Input* mode that allows formatting as you enter the data. With this mode turned on (it is now the default), you can enter subscripts etc. Maple Syrep was written before this mode was available, and uses the underscore (_) in many function names, for example `Transfer_function()`. In 2D-Math mode, this has to be entered as `Transfer\_funtion()` where the `\` acts as an *escape* character. I prefer to set Maple into the classic *1-D Math Input* mode while using Syrep. Examples of both modes are shown below.