Spring-Mass-Damper System: Computational Solutions using MATLAB® [A Topic in Engineering Math]

Presented by:
Ryan M. Cabrera

Department of Mechanical Engineering
College of Engineering
University of the Philippines

About the Course/Topic

The topic (Spring-Mass-Damper System: Computational Solutions using MATLAB®), is part of a course called ***Engineering Mathematics 1***, offered at the Department of Mechanical Engineering, University of the Philippines.

- **Number of Students:** Prior to the pandemic (≈ 30); currently (≈ 20).
- **Course Description:** First course on differential equation (first order and second order ODE only); applications; introduction to numerical calculus; numerical solutions to first-order ODEs; then basic linear algebra in preparation for systems of ODEs. Also includes intro to MATLAB®.
- **Course Delivery (in the context of online learning):**
  - **Lecture Class:** Asynchronous (pre-recorded Lecture Videos) **[3hrs/week]**
  - **Laboratory Class:** Synchronous sessions via Zoom; practice problems and class discussion; supplementary materials **[3hrs/week]**
- **Course Assessments:** Problem sets (by group); Exams (individual); *personal learning journal* (individual), which is a narrative of their experience in the context of online learning.
- **Significance of this Course in the Curriculum:** Required in subsequent computational courses; useful in courses such as *machine dynamics*, *control systems*, and the like.

The Problem

The Physical System:

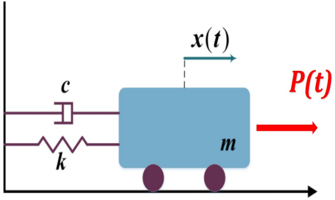

Figure 1: Spring-mass-damper system. [Taken from Parsa, B., Rajasekaran, K., Meier, F., and Banarjee, A.G. "A Hierarchical Bayesian Linear Regression Model with Local Features for Stochastic Dynamics Approximation."]

The Mathematical Problem:

$$m\frac{\mathrm{d}^2x}{\mathrm{d}t^2} + c\frac{\mathrm{d}x}{\mathrm{d}t} + kx = P(t) \tag{1}$$

where the applied force $P(t)$ can be $P(t) = P_o \sin \alpha t$.

Teaching Approach to the Computational Solutions

Teaching Approach to the Computational Solutions: It should be noted that the entire course is divided into *modules*.

1. The physical system and mathematical problem are presented at the beginning of the semester to motivate the students of the importance of differential equations and computational solutions.
2. They begin their study with first-order ODEs and their applications.
3. After first-order ODEs, they proceed with second-order ODEs, particularly linear ODEs. They will learn the following exact solution methods (for constant coefficients): *method of undetermined coefficients*, *method of variation of parameters*, and the *Laplace transform method.*
4. After learning the exact solutions, they will learn how to solve numerically using the following:
   - The Euler method and Runge-Kutta methods
   - MATLAB® built-in functions, particularly `ode45()`.
5. Only then can they solve the given mathematical problem. (They can even solve, numerically, a problem with non-constant coefficients, *e.g.*, the damping coefficient as $c = c(t)$.)

The Computational Solutions

The Computational Solutions:

Exact Solutions of Position and Velocity:

Note:

$x_e(t)$ = exact solution of position function, which solves the ODE: $\ddot{x} + 4\dot{x} + 8x = \sin t$

$v_e(t) = \dfrac{d^2x_e}{dt^2}$ exact solution of velocity

```
x_exact = @(t) -(4/65)*cos(t) + (7/65)*sin(t) + ...
        (69/65)*exp(-2*t).*cos(2*t)+(131/130)*exp(-2*t).*sin(2*t);
v_exact = @(t) (7/65)*cos(t) + (4/65)*sin(t) + ...
        (1/65)*exp(-2*t).*(-7*cos(2*t) - 269*sin(2*t));
```

Figure 2: The exact solutions for position $x(t)$ and velocity $v(t)$.

The Computational Solutions

The Computational Solutions (Continued):

Euler Method (Numerical):

```
%Time Domain
t_o = 0; t_e = 10; % Time interval, [t_o,t_e], in [s]
N = 100; % Number of subintervals
t_span = linspace(t_o,t_e,N+1); %[1 by (N+1)] vector of t-nodes

%Define RHS function
f_1 = @(t,x,w) w;
f_2 = @(t,x,w) (P_o/m)*sin(alpha*t) - (c/m)*w - (k/m)*x;

%Discretize
h = (t_e - t_o)/N;
X = zeros(1,N+1); % Setting up a [1 by (N+1)] vector of x-values
V = zeros(1,N+1); % Setting up a [1 by (N+1)] vector of v-values
X(1) = x_o; % Applying initial position, x_o
V(1) = v_o; % Applying initial velocity, v_o

for j = 1:N
    X(j+1) = X(j) + h*f_1(t_span(j),X(j),V(j));
    V(j+1) = V(j) + h*f_2(t_span(j),X(j),V(j));
end
x_EEM = X';
v_EEM = V';
```

Figure 3: The Euler method solutions for position $x(t)$ and velocity $v(t)$.

The Computational Solutions

The Computational Solutions (Continued):

MATLAB Built-in Function ODE45()

```
z_o = [x_o,v_o]; % Vector containing ICs
% Creating the function handle for the system of ODEs
ODEsystemFunc = @(t,z) [z(2);...
                        (P_o/m)*sin(alpha*t) - (c/m)*z(2) - (k/m)*z(1)];
%Implementing MATLAB's built-in function, ode45()
[time,Z] = ode45(ODEsystemFunc,t_span,z_o);
x_ode45 = Z(:,1); % Extracting the x-values [(N+1) by 1]
v_ode45 = Z(:,2); % Extracting the v-values [(N+1) by 1]
```

Figure 4: The MATLAB® built-in function ode45() solutions for position $x(t)$ and velocity $v(t)$.

The Required Output: A Report: The given problem presented herein is just part of a Problem Set (PSet), for which students are required to submit a Report. For each item in the PSet, whenever applicable, the following must be included in the **PSet Report**:

- Problem statement and related figures.
- Computational solution algorithm.
- MATLAB code.
- Results and discussion.

The Computational Solutions

Results: Plot of $x(t)$

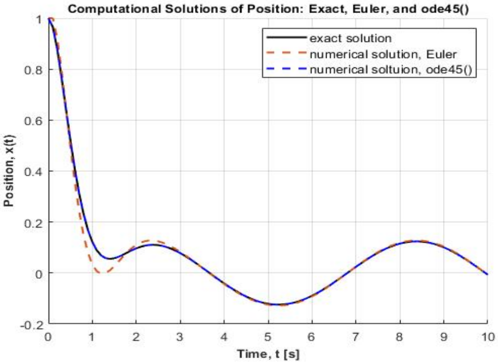

Figure 5: The MATLAB® plots for position $x(t)$: Exact, Euler, and ode45().

The Computational Solutions

Results: Plot of $v(t)$

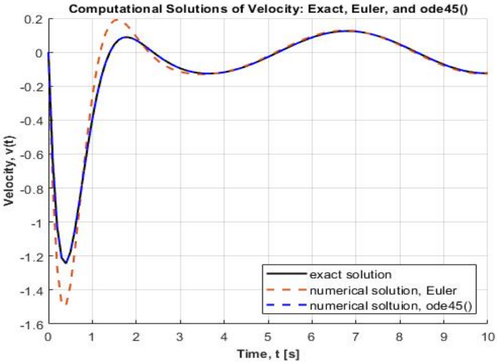

Figure 6: The MATLAB® plots for velocity $v(t)$: Exact, Euler, and ode45().

Summary of Outcomes

Summary of Outcomes:

- At the beginning of each topic in an engineering mathematics course (particularly for undergraduates), motivate the students by introducing physical problems and their associated mathematical models (*e.g.*, in the form of ODEs), which to be solved at a later time once the solution methods have been studied.
- Require students to solve a mathematical/computational problem using two or more solutions for computational verification purposes; thus, they would know if their solutions are correct even before submitting their work.
- Make them submit their work in Report Format containing problem statement, computational algorithm, MATLAB code, as well as results in the form of plots with accompanying discussion.