Implicit Euler with Newton-Raphson for Mass-Spring-Damper System

Auralius Manurung

1 Oct 2015

1 Mass-Spring-Damper System

Define a second order system:

$$M\ddot{x}(t) + B\dot{x}(t) + Kx(t) = f(t) \tag{1}$$

where: M, B, K are the coefficients for mass, damping, and stiffness respectively.

2 Newton-Raphson

Newton-Raphson is used to approximate x that fulfills $g(x) = 0$ by iterating:

$$x_{k+1} = x_k - \frac{g(x_k)}{g'(x(k))} \tag{2}$$

until x_{k+1} converges to a certain value.

3 Implicit Euler

Implicit Euler method says that solution to: $\dot{x} = f(t, x)$ can be found using:

$$x_{k+1} = x_k + hf(t_{k+1}, x_{k+1}) \tag{3}$$

where h is the time step.

The equation above can not be solved directly since there is x_{k+1} in both left and right side of the equation. Therefore, it needs to be changed to:

$$g(x_{k+1}) = x_{k+1} - x_k - hf(t_{k+1}, x_{k+1}) = 0 \tag{4}$$

Since the equation above is now in form of: $g(x_{k+1}) = 0$, thus, we can use Newton-Raphson to solve $g(x_{k+1})$ and acquire x_{k+1} as the result.

4 Implementation

First, we need to write the system equation into $\dot{x} = f(t, x)$ and apply the implicit Euler on it. However, since mass-spring-damper system is a second order system, we change the system equation into: $\ddot{x} = f(t, x, \dot{x})$

$$\begin{aligned} M\ddot{x} + B\dot{x} + Kx &= F \\ \ddot{x} &= \frac{F - B\dot{x} - Kx}{M} \end{aligned} \tag{5}$$

Next, using the implicit Euler, we solve $\ddot{x}$ to get $\dot{x}$:

$$\begin{aligned} 0 &= \dot{x}_{k+1} - \dot{x}_k - h\ddot{x}_{k+1} \\ 0 &= \dot{x}_{k+1} - \dot{x}_k - h\left(\frac{F - B\dot{x}_{k+1} - Kx_{k+1}}{M}\right) \\ 0 &= g(\dot{x}_{k+1}) = \frac{M}{h}\dot{x}_{k+1} - \frac{M}{h}\dot{x}_k - F + B\dot{x}_{k+1} + Kx_{k+1} \end{aligned} \tag{6}$$

x_k and $\dot{x}_k$ are known, while x_{k+1} is not yet known. Therefore, we need another implicit Euler to expand: $x_{k+1} = x_k + h\dot{x}_{k+1}$. As the result, we can then write:

$$0 = g(\dot{x}_{k+1}) = \frac{M}{h}\dot{x}_{k+1} - \frac{M}{h}\dot{x}_k - F + B\dot{x}_{k+1} + K(x_k + h\dot{x}_{k+1}) \tag{7}$$

Thus, the Jacobian of $g(\dot{x}_{k+1})$ can be written as:

$$g'(\dot{x}_{k+1}) = \frac{M}{h} + B + Kh \tag{8}$$

As final step, Newton-Raphson can then be applied to find $\dot{x}_{k+1}$ for given x_k and $\dot{x}_k$. Once $\dot{x}_{k+1}$ is found, x_{k+1} can be calculated in an implicit Euler way with $x_{k+1} = x_k + h\dot{x}_{k+1}$ (see Eqn. 3).

5 Discussions

It is actually not necessary to use Newton-Rapshon, as finding $\dot{x}_{k+1}$ can also be done directly by grouping $\dot{x}_{k+1}$ in Eqn. 7:

$$\begin{aligned} \dot{x}_{k+1}\left(\frac{M}{h} + B + Kh\right) &= \frac{M}{h}\dot{x}_k + F - Kx \\ \dot{x}_{k+1} &= \frac{\frac{M}{h}\dot{x}_k + F - Kx}{\frac{M}{h} + B + Kh} \\ \dot{x}_{k+1} &= \frac{M\dot{x}_k + Fh - Khx}{M + Bh + Kh^2} \end{aligned} \tag{9}$$

However, as pointed by [1], the direct solution above is not preferable when there are a large number of interconnected particles involved. At such condition, large sparse matrix inversion must be done if doing direct method. In [1], the authors use conjugate-gradient method. Additionally, simulating the system in 3 dimensions will greatly contribute to the larger matrix dimension.

Implicit Euler is unconditionally stable as long as the system itself is a stable system. See [2], page 21-23 for more details.

References

[1] D. Baraff and A. Witkin, Large steps in cloth simulation, in Proceedings of the 25th annual conference on Computer graphics and interactive techniques - SIGGRAPH 98, 1998, pp. 4354

[2] Szymon Rusinkiewicz, ODE and PDE stability analysis., [ONLINE] `http://www.cs.princeton.edu/courses/archive/fall12/cos323/notes/cos323_f12_lecture13_ode.pdf`

6 Appendices

```
% ==========================================================================
% Solve M x_ddot + B x_dot + K x = F
% Implicit Euler with Newton Raphson
% ==========================================================================

function implicit_euler_newton_raphson()
    clear all;
    close all;
    clc;

    figure;
    hold;

    K = 100; % spring stiffness. N/m
    M = 5; % mass, kg
    B = 2*sqrt(K*M); %c ritical damping
    %B = 0;

    % initial condition:
    x0 = 0.1;
    v0 = 0;
    F0 = 0;

    %%%%%%%%%%%%%%%%%%%%%%%%%%
    h = 0.01; % sampling rate, try from 1ms to 1s
    %%%%%%%%%%%%%%%%%%%%%%%%%%

    t_start = 0;
    t_end = 5;
    t=t_start:h:t_end;

    % Use ode45, 1kHz as ground truth:
    [tode45,xode45]=ode45(@msd, [t_start:0.001:t_end], [x0 v0], [], ...
                        M, B, K);
    plot(tode45,xode45(:,1), '--r')

    for k = 1 : length(t)
```

```
        % Newton-Raphosn relies on good intial value
        % As an initial guess, a 1-step forward Euler is used
        %v1 = v1_hat(M, B, K, 0, x0, v0, h);

        % Initializing v1 = 0 also works. It just needs more iterations
        v1 = 0;

        % Newton-Rapshon
        % x(k+1) = x(k) - g(x(k))/g'(x(k))
        % Since the system is linear, g'(x(k)) = constant
        g_d = M/h + B + K * h;
        while(1) % Newton-Raphson iteration
            v1_ = v1 - (g(M, B, K, F0, x0, v0, v1, h) / g_d);
            if abs(v1 - v1_) <0.0001
                break;
            end
            v1 = v1_;
        end

        x1 = x0 + v1 * h; % xk1 = xk + h * xdotk1
        x0 = x1;
        v0 = v1;

        x(k, :) = [x1 v1];
    end
    plot(t, x(:,1), 'b')

    legend('ode45', 'Implicit␣Euler');
    xlabel('Time␣(s)')
    s = strcat('h␣=␣', num2str(h), '␣seconds');
    title(s);
end

function output = v1_hat(M, B, K, F, x0, v0, h)
    v0_dot = (F - B * v0 - K * x0) / M;
    output = v0 + h * v0_dot;
end

function output = g(M, B, K, F, x0, v0, v1, h)
    output = M*(v1-v0)/h-F+B*v1+K*(x0+h*v1);
```

```
end

function xdot=msd(t,x, M, B, K)
    xdot_1 = x(2);
    xdot_2 = -(B/M)*x(2) - (K/M)*x(1);

    xdot = [xdot_1 ; xdot_2 ];
end
```