Prof. Raghuveer Parthasarathy
University of Oregon; Fall 2007
Physics 351 – Vibrations and Waves

Programming #P7: Solution

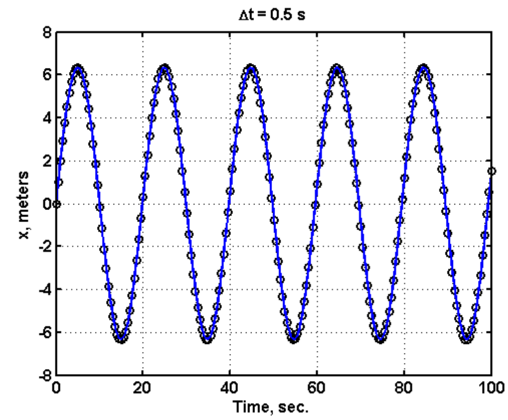
(a) For the exact (analytic) solution:

The general solution $x(t) = A \sin(\omega t - \varphi)$. The initial $x(t=0) = 0$, so $\varphi = 0$. $v(t=0) = A\,\omega \cos(\omega t) = v_0$, so $A = v_0/\omega$. Therefore the particular solution $x(t) = (v_0/\omega)\sin(\omega t)$.

Program listing, with **"new" lines in bold:**

```
% verletSHO_P7.m
% SOLUTION to exercise P7 -- modifying SHO model to have a variable timestep
%    and also to plot the exact SHO solution
%
% Raghuveer Parthasarathy
% Oct. 5, 2007

clear all
close all

x(1) = 0.0;  % initial position, meters
v(1) = 2.0;  % initial velocity, m/s
Deltat = input('Enter Deltat (seconds): ');  % time increment, s
x(2) = x(1) + v(1)*Deltat;   %We'll explicitly write x(1), even though
               % it's zero here, in case we ever want to change
               % our initial conditions.  (Otherwise, we might get
               % confused!)
k = 0.1;  % Newtons / meter
m = 1.0;  % kilograms

% for exact solution
% General solution x = A sin(wt - phi)
% Initial x(t=0) = 0, so phi = 0.  v(t=0) = Aw cos(wt)=v0, so A = v0/w
ta = 0:0.1:100; % time array, seconds
w = sqrt(k/m);  % angular freqency (omega), radians / sec
xa = (v(1) / w)*sin(w*ta);

Tfinal = 100.0;  % ending time, seconds
t = 0:Deltat:Tfinal;  % an array of all the time values -- starts at 0
N = length(t);  % "length" gives the number of elements in an array
for j=3:N;
    x(j) = 2*x(j-1) - x(j-2) + Deltat*Deltat*(-1.0*k/m)*x(j-1);
    v(j-1) = (x(j) - x(j-2))/ (2*Deltat);
end
v(N) = (x(N)-x(N-1))/Deltat;  % Why? Because v(N)
     % is not set by the above For loop
figure; plot(t, x, 'ko:'); grid on;
xlabel('Time, sec. ');
ylabel('x, meters')
hold on
plot(ta, xa, 'b-');
```

Output (Δt = 0.5 s).
We see that the numerical and analytic solutions are very similar.



(b) See above for the program listing with Deltat being a user-input variable.

Output (Δt =3, 6 s).

We see that for larger Δt, the numerical solution is increasingly incorrect. This is because our algorithm, based on a Taylor expansion of x(t), is an increasingly poor approximation to the true "continuous" x(t).

Mr. K. suggests Δt = 0.0001 seconds. He thinks this is a good idea because, as noted above, smaller Δt yields a better approximation to the true x(t). However, it requires lots of calculation time – to model 100 seconds with Δt = 0.0001 seconds requires 10^6 passes through our for-loop, which may be slow.