Detour: Butcher Tableaux

- We have seen several different solvers already.
- There are “zillions” more out there.
- How to categorize and make sense of all of them?
- Many look like this:

$$
\begin{aligned}
k_1 &= hf(t_n, y_n) \\
k_2 &= hf(t_n + c_2 h, y_n + a_{21} k_1) \\
k_3 &= hf(t_n + c_3 h, y_n + a_{31} k_1 + k_{32} k_2) \\
&\vdots \\
y_{n+1} &= y_n + (b_1 k_1 + b_2 k_2 + b_3 k_3 + \cdots)
\end{aligned}
$$

Recall how we defined Runge-Kutta 4

From: Numerically Solving Ordinary Differential Equations, Brorson (linked on Canvas)

Butcher Tableaux

- A Butcher tableau is a table which organizes all the coefficients into a standard format

$$
\begin{aligned}
k_1 &= hf(t_n, y_n) \\
k_2 &= hf(t_n + c_2 h, y_n + a_{21}k_1) \\
k_3 &= hf(t_n + c_3 h, y_n + a_{31}k_1 + k_{32}k_2) \\
&\vdots \\
y_{n+1} &= y_n + (b_1 k_1 + b_2 k_2 + b_3 k_3 + \cdots)
\end{aligned}
$$

$$
\begin{array}{c|cccc}
c_1 & a_{11} & a_{12} & a_{13} & \cdots \\
c_2 & a_{21} & a_{22} & a_{23} & \cdots \\
c_3 & a_{31} & a_{32} & a_{33} & \cdots \\
\vdots & \vdots & \vdots & \vdots & \ddots \\
\hline
 & b_1 & b_2 & b_3 & \cdots
\end{array}
$$

- There is no deep math here – Butcher tableaux are just are a way to try to make sense of all the different ODE solvers out there.

Butcher tableau examples

- From my book

$$\begin{array}{c|cc} 0 & 0 & 0 \\ 1 & 1 & 0 \\ \hline & 1/2 & 1/2 \end{array}$$

$$\begin{aligned} k_1 &= hf(t_n, y_n) \\ k_2 &= hf(t_n + h, y_n + k_1) \\ y_{n+1} &= y_n + k_1/2 + k_2/2 \end{aligned}$$

Table 4.2: Heun's method

$$\begin{array}{c|cccc} 0 & 0 & 0 & 0 & 0 \\ 1/2 & 1/2 & 0 & 0 & 0 \\ 1/2 & 0 & 1/2 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ \hline & 1/6 & 1/3 & 1/3 & 1/6 \end{array}$$

$$\begin{aligned} k_1 &= hf(t_n, y_n) \\ k_2 &= hf(t_n + h/2, y_n + k_1/2) \\ k_3 &= hf(t_n + h/2, y_n + k_2/2) \\ k_4 &= hf(t_n + h, y_n + k_3) \\ y_{n+1} &= y_n + (k_1 + 2k_2 + 2k_3 + k_4)/6 \end{aligned}$$

Table 4.4: Fourth-order Runge-Kutta

Next: Adaptive integrators

- Problem: Stiff systems
  - ODE may have regions requiring tiny time steps for accuracy or stability.
  - Other regions might not need tiny time steps.
  - Fixed time step h means you waste lots of time stepping through regions not needing tiny time steps just because of some short region requiring tiny time steps.

Consider the Logistic Equation

- Simple population model
  - y = population of rabbits in a field.
  - Ymax is "carrying capacity" of the field. (Limited due to finite supply of e.g. lettuce or carrots.)

$$\frac{dy}{dt} = \left(1 - \frac{y}{Y_{max}}\right) y$$

- For small y, growth is exponential.
- For y approaching Ymax, growth saturates.

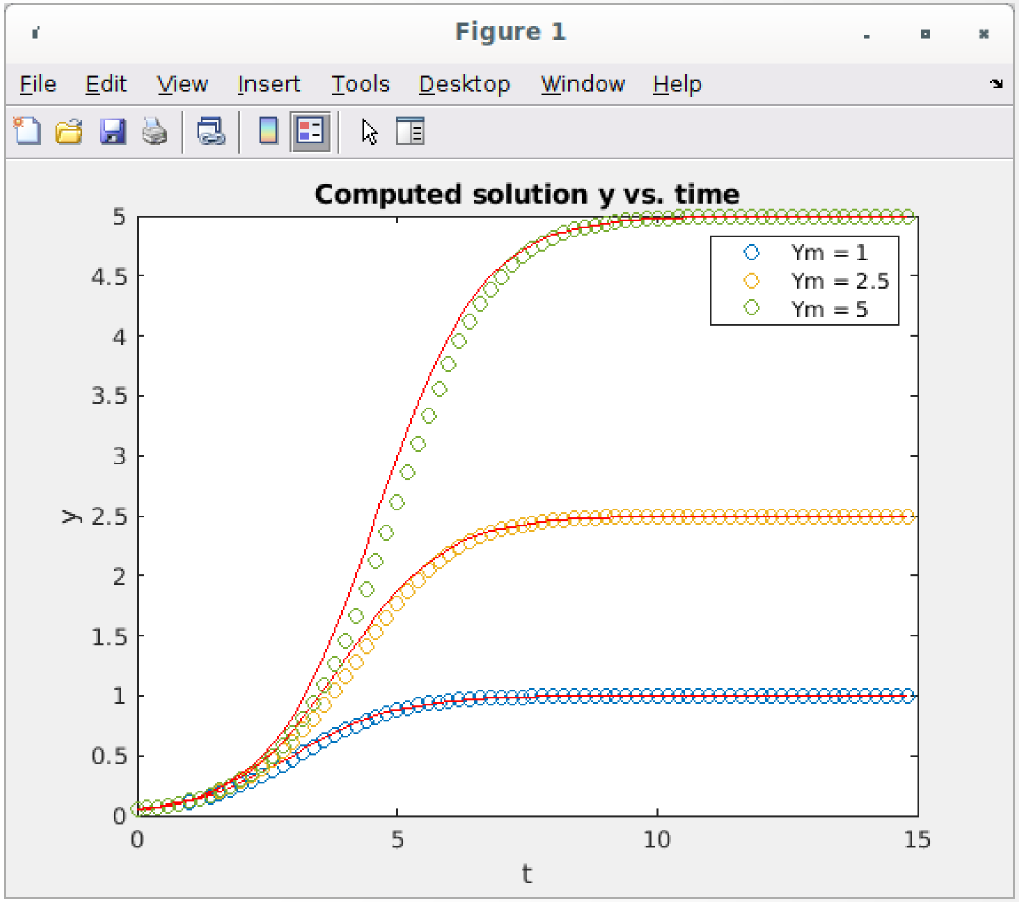
Forward Euler solution

- Original ODE

$$\frac{dy}{dt}=\left(1-\frac{y}{Y_{max}}\right)y$$

- Forward Euler discretization

$$y_{n+1}=y_n+h\left(1-\frac{y_n}{Y_{max}}\right)y_n$$



- Solution has two regimes:
  - Fast initial growth.
  - Slow change after saturation.

Simple example of stiff system

Adaptive time stepping

- Use small h when solution is changing rapidly.
- Use large h when solution is changing slowly.
- But how to distinguish the two cases?
  - Note that you can’t just rely on monitoring your solution since you don’t know if the slow vs. fast change is an artifact of your stepsize h.
- Maybe use smaller step size when computed solution deviates from true solution.
  - But how do I know what is the true solution?
- Idea: Use an Oracle!

Oracle

- Priestess at many temples in ancient Greece.
- You can ask a question, she will tell you the true answer.

Wikipedia. Painting by John William Waterhouse

- In software testing, an oracle is a program which tells you if your answer is right or not.

Idea for adaptive time stepping

- Have two tols: tol1 >> tol2.
- Take forward Euler step. $$y_{n+1} = y_n + h\left(1 - \frac{y_n}{Y_{max}}\right) y_n$$
- Ask oracle what is error of y_{n+1} compared to true.
- If error > tol1: reduce step size $h \rightarrow h/2$ and try again.
- If error < tol2 then increase step size $h \rightarrow 2h$, accept new point y_{n+1} and keep going
- Otherwise, just accept new point y_{n+1} and keep going.
- But what about the oracle?

Oracle – use a different solver

- Tols: $\mathrm{tol1}=0.01 \qquad \mathrm{tol2}=0.0002$
- Take forward Euler step.

$$f_n=f(t_n, y_n) \qquad y^e_{n+1}=y_n+hf_n$$

- Take e.g. Heun's method step

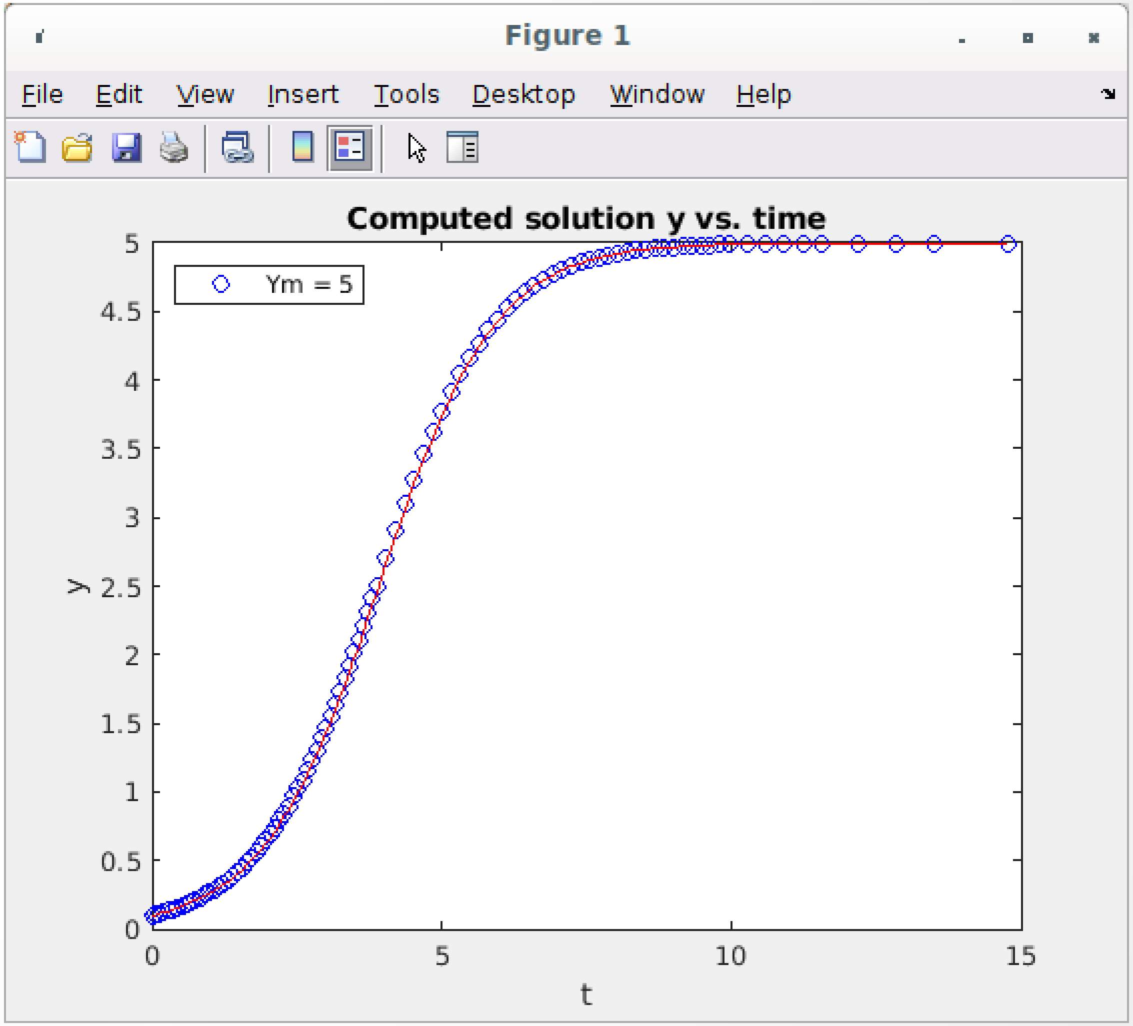
$$f_{n+1}=f(t_n+h, y^e_{n+1}) \qquad y^h_{n+1}=y_n+\frac{h}{2}(f_n+f_{n+1})$$

- Is $|y^e_{n+1}-y^h_{n+1}|>\mathrm{tol1}$?
  - Yes: reduce step size $h\rightarrow h/2$ and try again.
- Is $|y^e_{n+1}-y^h_{n+1}|<\mathrm{tol2}$?
  - Yes: Keep y^e_{n+1}, increase step size $h\rightarrow 2h$ and keep going.
- Else: Keep y^e_{n+1} and keep going.

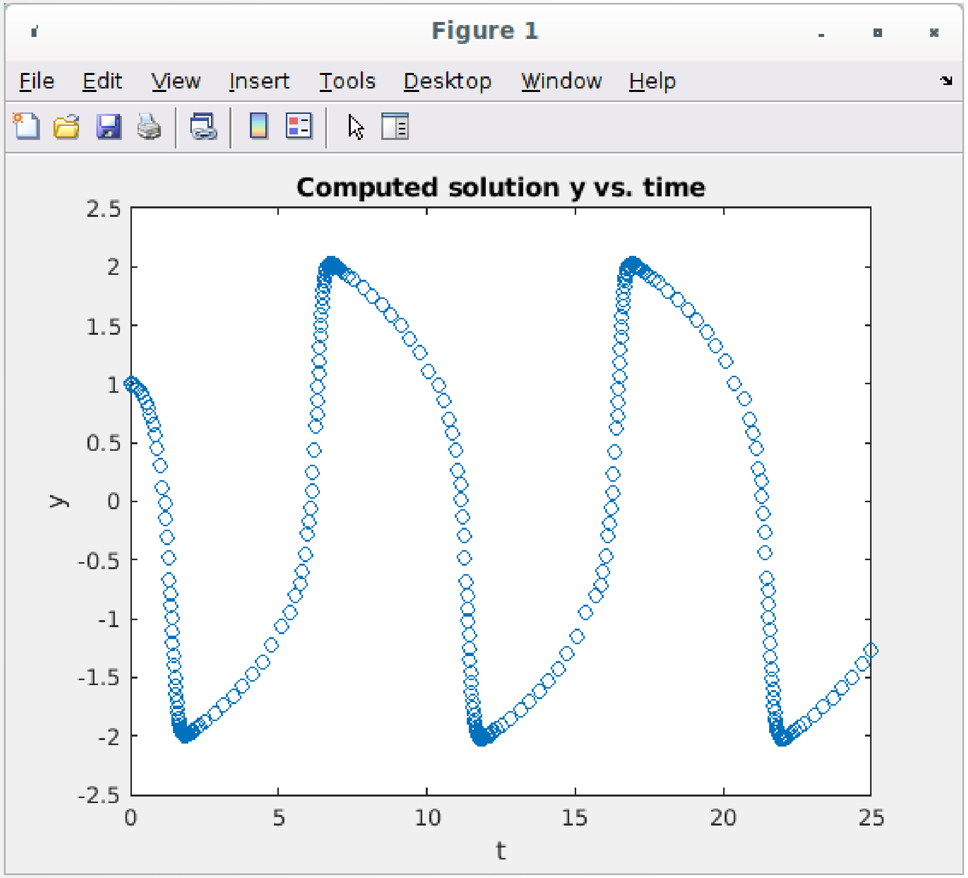
```
while (tn < Tend)

  fprintf('step n = %d, tn = %f, y = %f, h = %e ... ', n, tn, y(1,n), h)

  % First compute trial steps
  s1 = f(tn, y(:,n));                  % Slope at current location.
  yfe = y(:,n) + h*s1;                % Forward Euler step.
  s2 = f(tn, yfe);                     % Slope at new t
  yh = y(:,n) + h*(s1+s2)/2;          % Heun step.

  % Now compare trials steps
  diff = norm(yfe-yh);
  fprintf('diff = %e ... ', diff)
  if (diff>tol1)
    % Too much error - shrink step and try again.
    fprintf('Shrink h\n')
    h = h/2;
    continue
  elseif (diff<tol2)
    % Error nice and small -- grow step for next time.
    h = h*2;
    fprintf('Grow h\n')
  else
    % Error neither large nor small -- keep this step.
    fprintf('Keep h\n')
  end

  % If we get here the computation was good. Store the results.
  n = n+1;
  y(:,n) = yh;
  t(:,n) = tn;
  tn = tn+h;
end
```

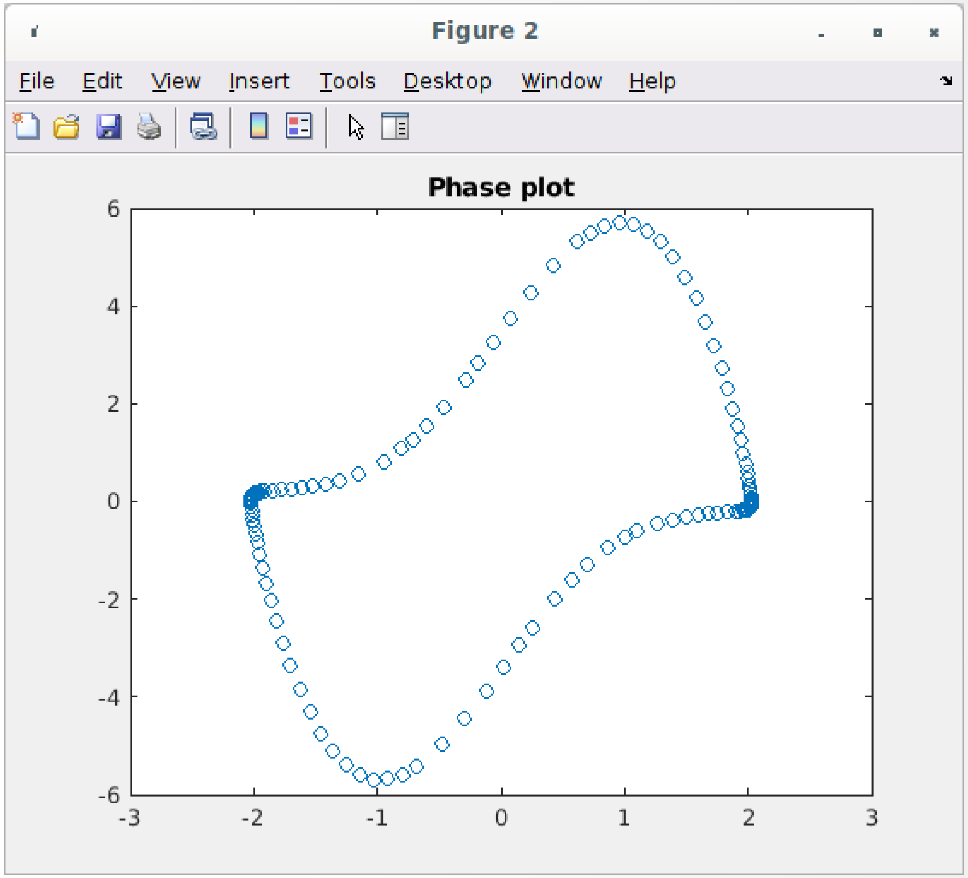
Adaptive time stepping the logistic eq.

- Note timestep h varies in time.
- Stepsize bunched up at turning points.
- When y doesn't vary much in time, the stepsize grows.



Adaptive time stepping the Vanderpol osc.





- Note timestep h varies in time.
- Stepsize bunched up at turning points.

Waste of computation?

- Problem: My naive implementation does twice as many computations as needed.
- However, Matlab's ode45 uses Dormand-Prince DP5(4)7M – one 4th order and one 5th order.

4th and 5th order computations share most coefficients

Compare results from these two computations as oracle.

0	0	0	0	0	0	0	0
1/5	1/5	0	0	0	0	0	0
3/10	3/40	9/40	0	0	0	0	0
4/5	44/45	-56/15	32/9	0	0	0	0
8/9	19372/6561	-25360/2187	64448/6561	-212/729	0	0	0
1	9017/3168	-355/33	46732/5247	49/176	-5103/18656	0	0
1	35/384	0	500/1113	125/192	-2187/6784	11/84	0
	35/384	0	500/1113	125/192	-2187/6784	11/84	0
	5179/57600	0	7571/16695	393/640	-92097/339200	187/2100	1/40

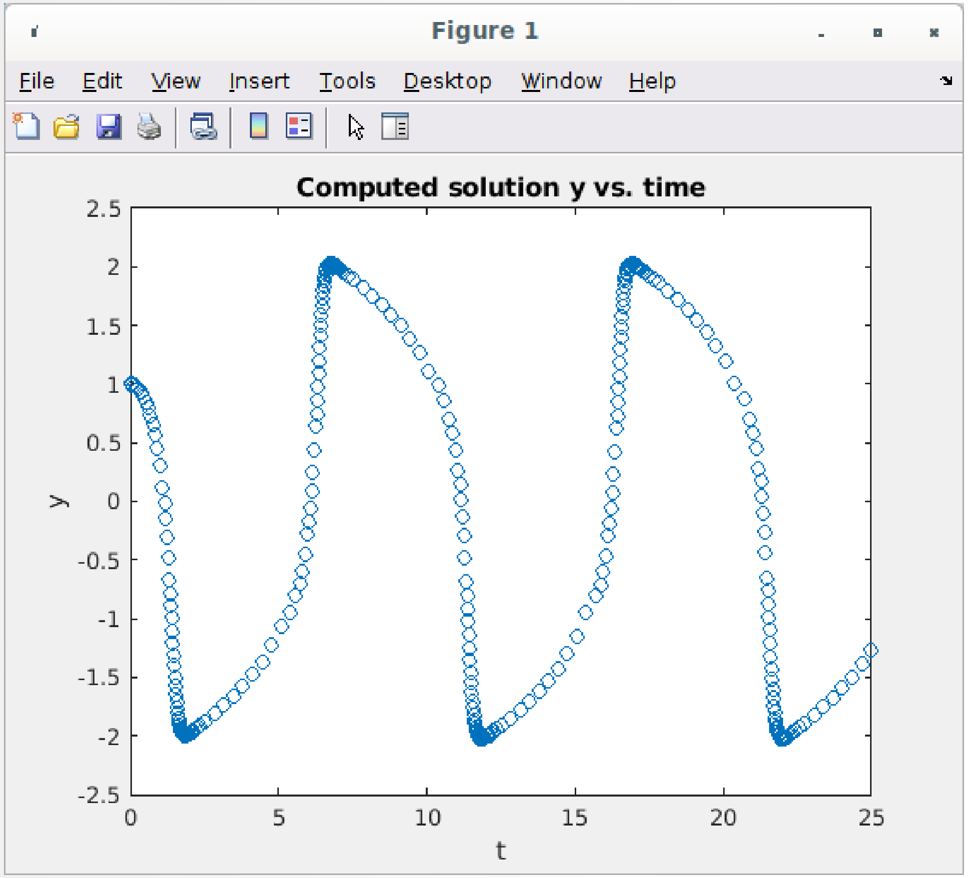
Table 5.1: Butcher tableau for the Dormand-Prince 4/5 solver used in Matlab's ode45(). The two rows below the horizontal line correspond to the two corrector steps which are compared against each other.

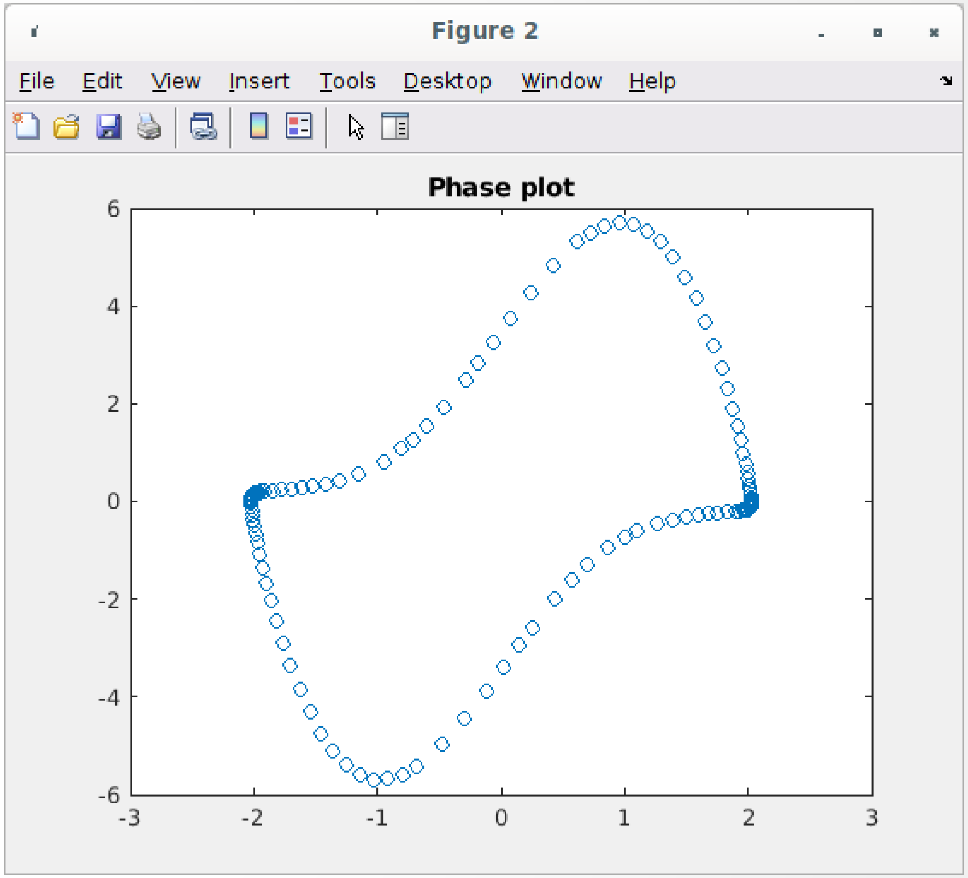
Peek into ode45

- dbtype – print out source code

```
% Initialize method parameters.
pow = 1/5;
A = [1/5, 3/10, 4/5, 8/9, 1, 1]; % Still used by restarting criteria
% B = [
%      1/5         3/40    44/45   19372/6561      9017/3168       35/384
%      0           9/40    -56/15  -25360/2187     -355/33         0
%      0           0       32/9    64448/6561      46732/5247      500/1113
%      0           0       0       -212/729        49/176          125/192
%      0           0       0       0               -5103/18656     -2187/6784
%      0           0       0       0               0               11/84
%      0           0       0       0               0               0
%      ];
% E = [71/57600; 0; -71/16695; 71/1920; -17253/339200; 22/525; -1/40];
```

Vanderpol osc. is a stiff system

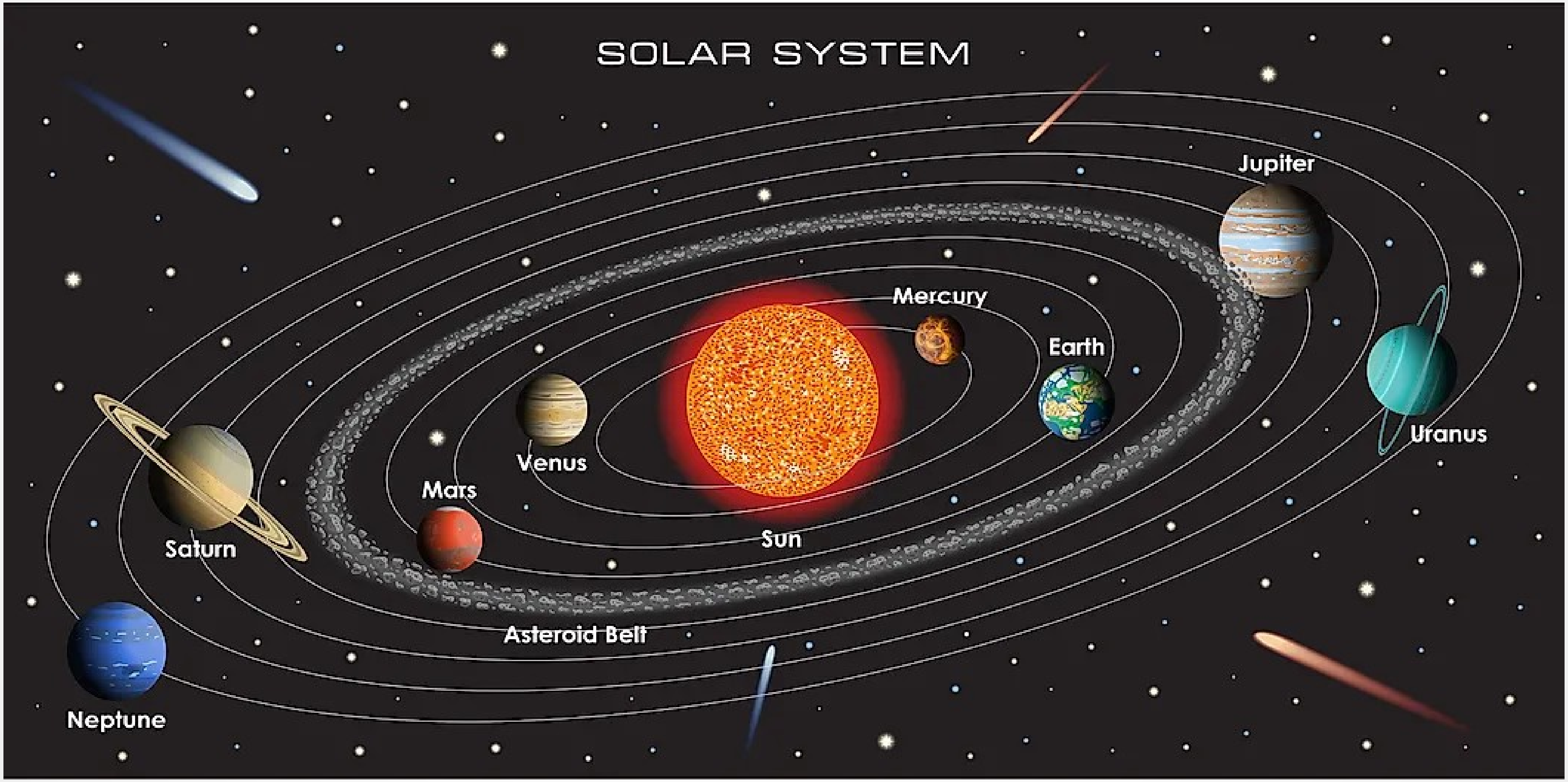




- This ODE requires tiny steps in portions of its orbit.
- Larger steps are OK in other portions of the orbit.
- This is the definition of a stiff sytsem.
- Use adaptive solvers for stiff systems.

Look at help strings for ode45, ode23, ode15s, etc.

Numerical Solutions of ODEs and Conservative Systems

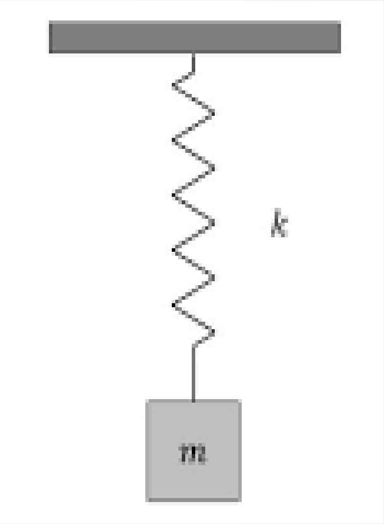


Energy

- Many important systems described by ODEs have an “energy functional”.
- Examples:
  - Free mass:

$$m\frac{d^2x}{dt^2}=0 \qquad \Rightarrow \qquad E=\frac{1}{2}m\left(\frac{dx}{dt}\right)^2$$

  - Harmonic oscillator



$$\frac{d^2x}{dt^2}=-\omega^2x \qquad \Rightarrow \qquad E=\frac{1}{2}\omega^2x^2+\frac{1}{2}\left(\frac{dx}{dt}\right)^2$$

Math aside: functional

- Functional: Mapping from a function (space) f(x) to a scalar (scalars).
- Examples:
  - Definite integral
  - Norm of function
  - Energy

$$L[f(t)] = \int_0^\infty dt\, f(t) e^{-t}$$

$$N[f(t)] = \int_{-\infty}^{\infty} dt\, (f(t))^2$$

$$E = H(p(t), x(t))$$

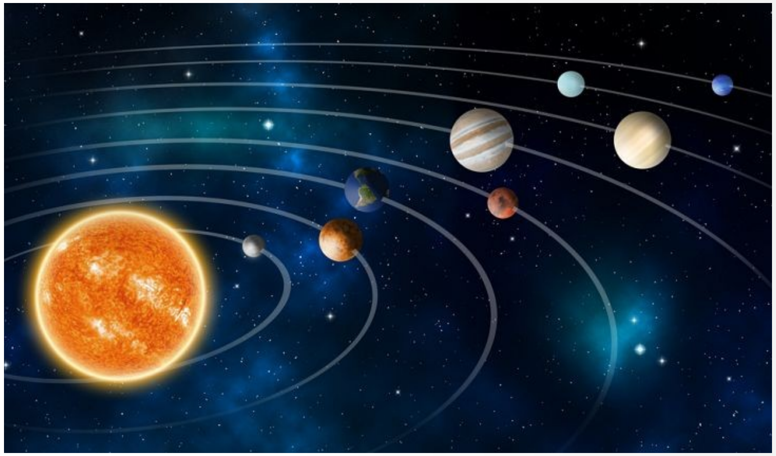
- Different from function, which maps a number (or set of numbers) to a number.
- Examples:
  - f(x)
  - Indefinite integral

$$f(x, y, \ldots): R^N \rightarrow R$$

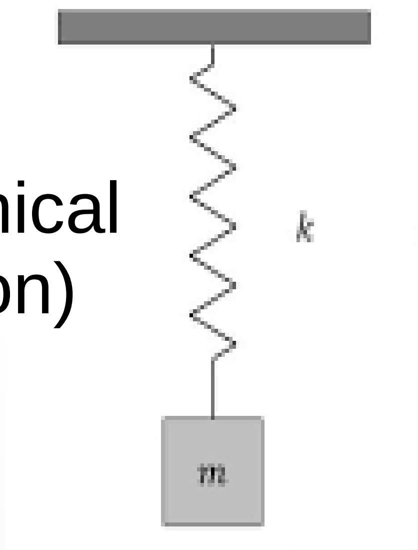
$$g(x) = \int_0^x dt\, f(t) e^{-t}$$

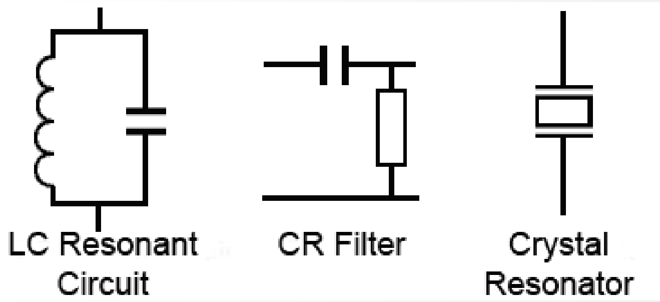
Conservative systems

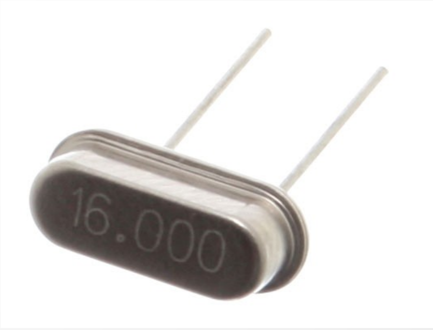
- Conservative system --> one in which its energy is constant in time (conserved).
  - Many important applications



- Examples:
  - Planetary motion.
  - Lots of simple mechanical systems (approximation)
  - Electronic oscillators (approximation)







Example: Harmonic oscillator

- Equation of motion: $$\frac{d^2x}{dt^2} = -\omega^2 x$$

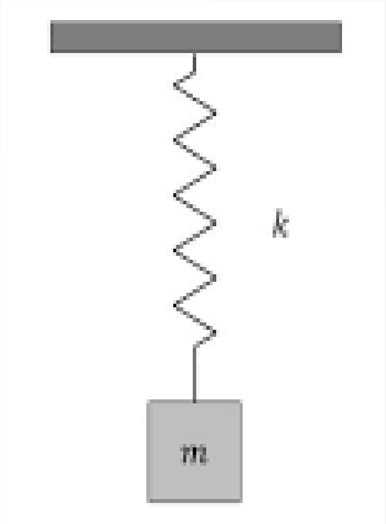
- Energy $$E = \frac{1}{2}\omega^2 x^2 + \frac{1}{2}\left(\frac{dx}{dt}\right)^2$$

- Energy change with time $$\frac{dE}{dt} = \omega^2 x \frac{dx}{dt} + \left(\frac{dx}{dt}\right)\left(\frac{d^2x}{dt^2}\right)$$

$$= \omega^2 x \frac{dx}{dt} - \omega^2 x \left(\frac{dx}{dt}\right)$$

$$= 0$$

Energy doesn't change = constant = conserved.



Mass on a spring

Forward Euler for Harmonic Oscillator

- Continuous system

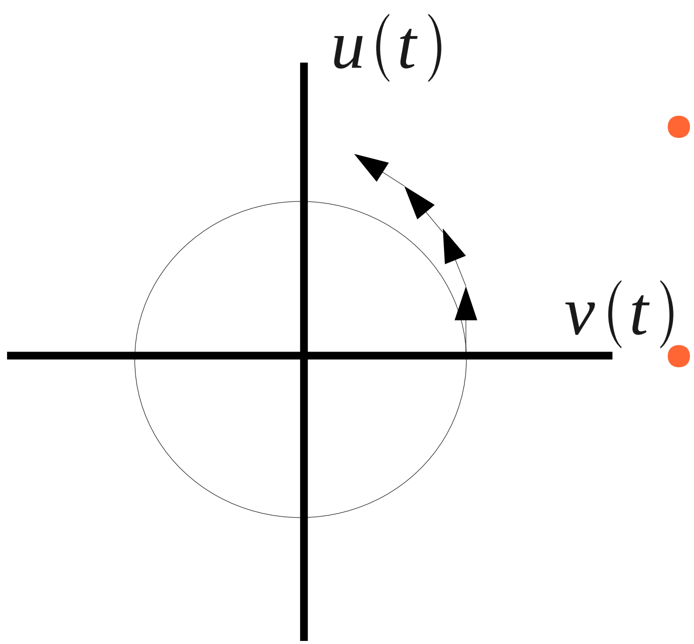
$$\frac{du}{dt}=v$$

$$\frac{dv}{dt}=-\omega^2 u$$

- Discretized fwd Euler

$$u_{n+1}=u_n+\Delta t\, v_n$$

$$v_{n+1}=v_n-\omega^2 \Delta t\, u_n$$



- Forward Euler solution spirals outward from true solution.
- Energy is not conserved by fwd Euler solution.

Compute energy at each fwd Euler step

Top level fcn

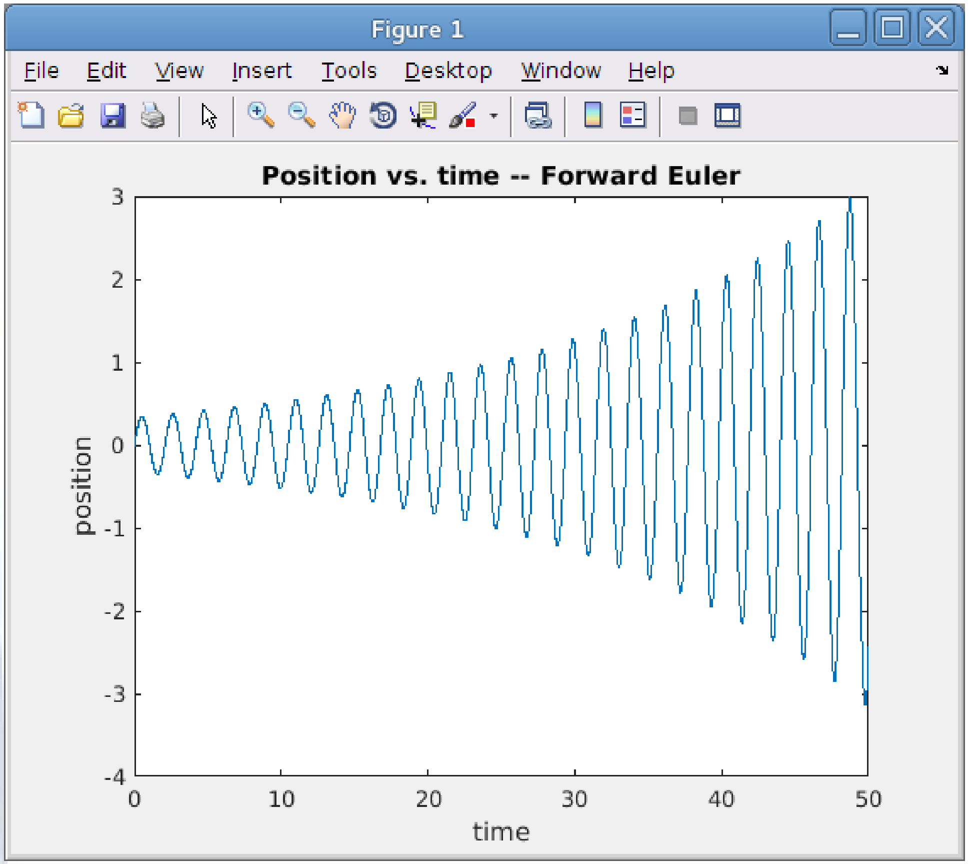
```
% Allocate energy vector
e = zeros(1, N);

% Initial cond
y(:,1) = [0; 1];  % row 1 = pos, row 2 = veloc
e(1) = (omega*omega*y(1,1)*y(1,1) + y(2,1)*y(2,1))/2;

% Take steps.  Compute energy at each step
for i=1:N-1
  y(:,i+1) = forward_euler_step(y(:,i), t(i), deltat);
  e(i+1) = (omega*omega*y(1,i+1)*y(1,i+1) + y(2,i+1)*y(2,i+1))/2;
end
```

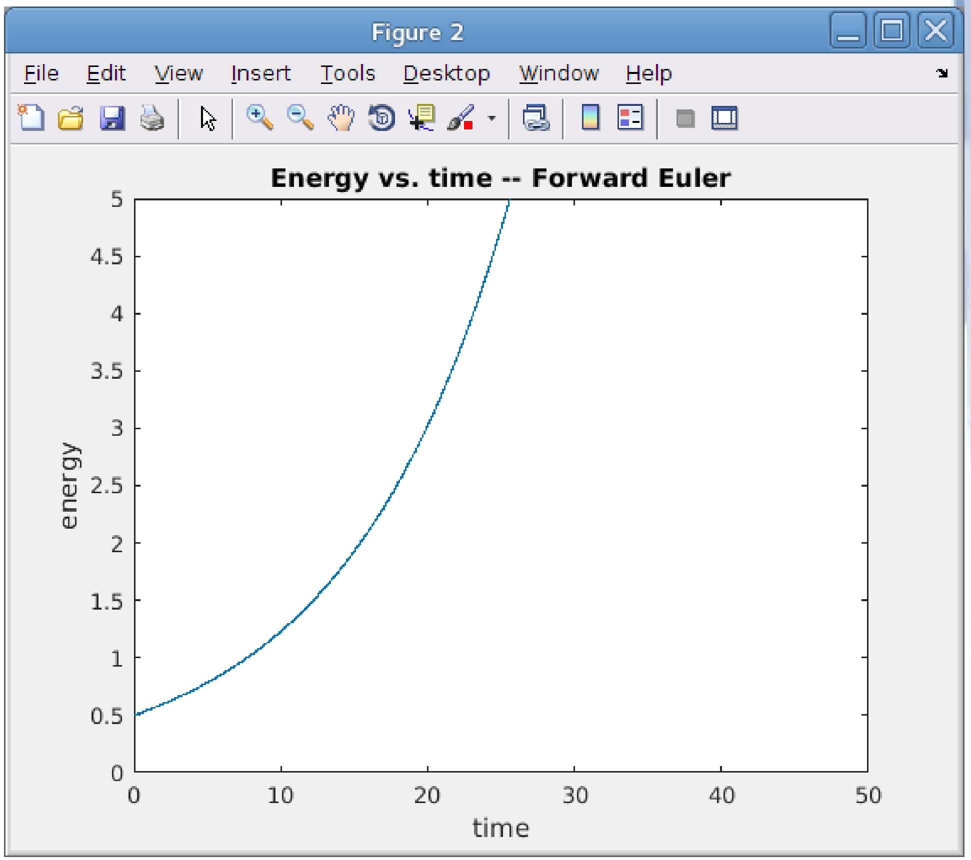
$$\left(\frac{1}{2}\right)\left[\omega^2 y^2 + \left(\frac{dy}{dt}\right)^2\right]$$

Subfcn

```
function ynp1 = forward_euler_step(yn, t, h)
  % This function takes a Forward Euler step starting
  % at position yn.  Stepsize is h.
  ynp1 = yn + h*f(t, yn);  % Return should be row vec.
end
```

Forward Euler for Harmonic Osc





- Stinks

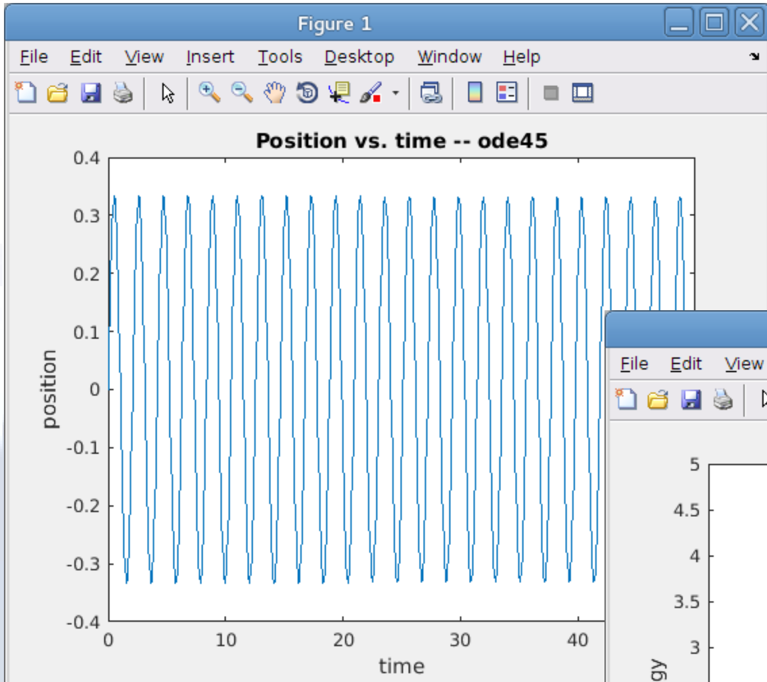
```
% Set up time axis of problem
Tend = 50;
deltat = .01;
```

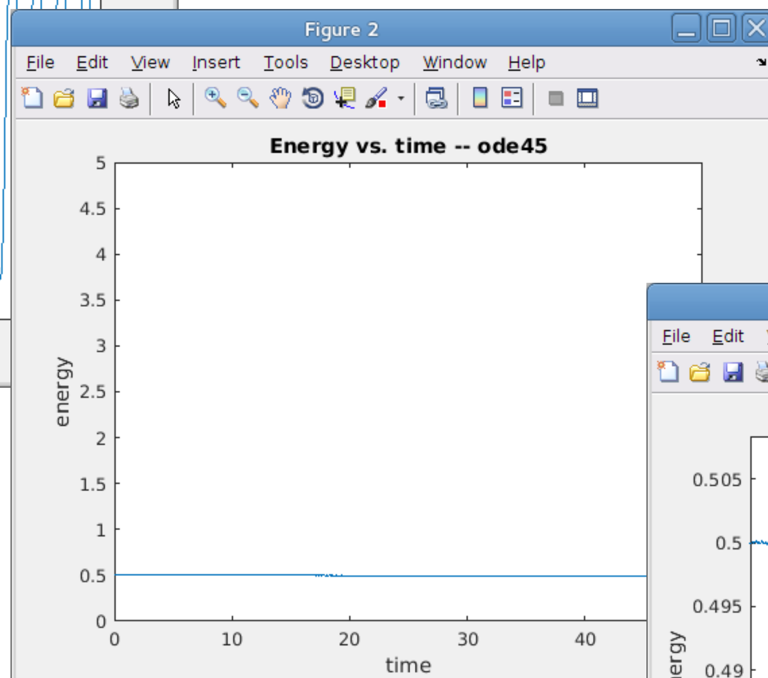
Energy non-conservation bad for many important systems

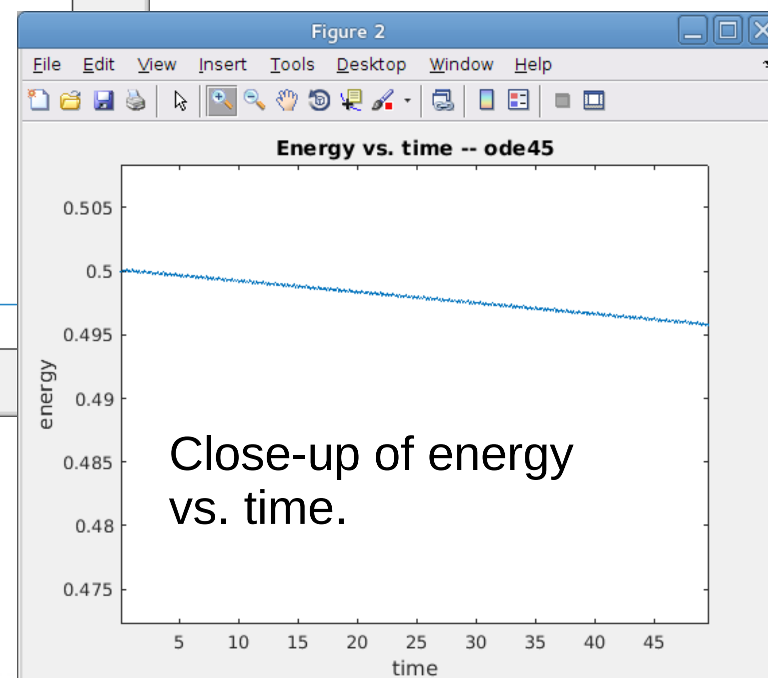
- What about ode45?

```
% Set up time axis of problem
Tend = 50;
deltat = .01;

% Initial cond
y0 = [0, 1];  % col 1 = pos, col 2 = veloc

% Solve system using ode45
[t, y] = ode45(@f, [0, Tend], y0);

% Allocate energy vector
N = length(t);
e = zeros(N, 1);

% Compute energy at each step
for i=1:N
  e(i) = (omega*omega*y(i,1)*y(i,1) + y(i,2)*y(i,2))/2;
end
```

ode45 for Harmonic Osc







- Ode45 is much better, but still non-conservative.

Symplectic integrators

- ODE solvers especially designed to handle conservative systems.
  - Semi-implicit (symplectic) Euler (1st order)
  - Verlet's method (2nd order)
  - 3rd and 4th orders

GTE ~ O(h)

- Area of recent development (1980s – 1990s)
- Semi-implicit Euler (2 dimensional):

$$\frac{dv}{dt} = g(u,t)$$

$$\frac{du}{dt} = f(v,t)$$

Note special form!

$$v_{n+1} = v_n + h\,g(u_n, t_n)$$

$$u_{n+1} = u_n + h\,f(v_{n+1}, t_n)$$

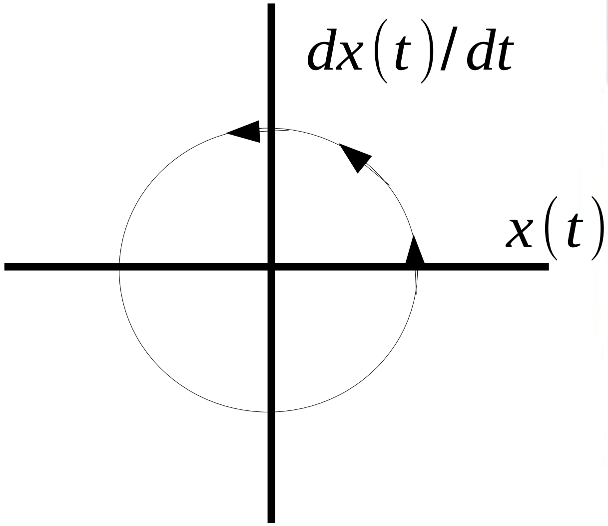
Note v_{n+1}

Symplectic?

- Symplectic = having to do with the phase space of a Hamiltonian system.
- Hamiltonian system = a dynamic system in which energy is conserved.

Harmonic oscillator

$$E = \frac{1}{2}\omega^2 x^2 + \frac{1}{2}\left(\frac{dx}{dt}\right)^2$$



- Phase space = plot the state of the system as a point in x, v (position & velocity) coordinates.
- Symplectic integrator = energy conserving.

Symplectic integrators

- ODE solvers especially designed to handle conservative systems.
  - Semi-implicit (symplectic) Euler (1st order)
  - Verlet's method (2nd order)
  - 3rd and 4th orders
- Area of recent development (1980s – 1990s)
- Semi-implicit Euler (2 dimensional):

$$\frac{dv}{dt} = g(u,t)$$

$$\frac{du}{dt} = f(v,t)$$

Note special form!

$$v_{n+1} = v_n + h\,g(u_n, t_n)$$

$$u_{n+1} = u_n + h\,f(v_{n+1}, t_n)$$

Note v_{n+1}

Harmonic Oscillator using Symplectic Euler

$$\frac{du}{dt} = v$$

$$\frac{dv}{dt} = -\omega^2 u$$

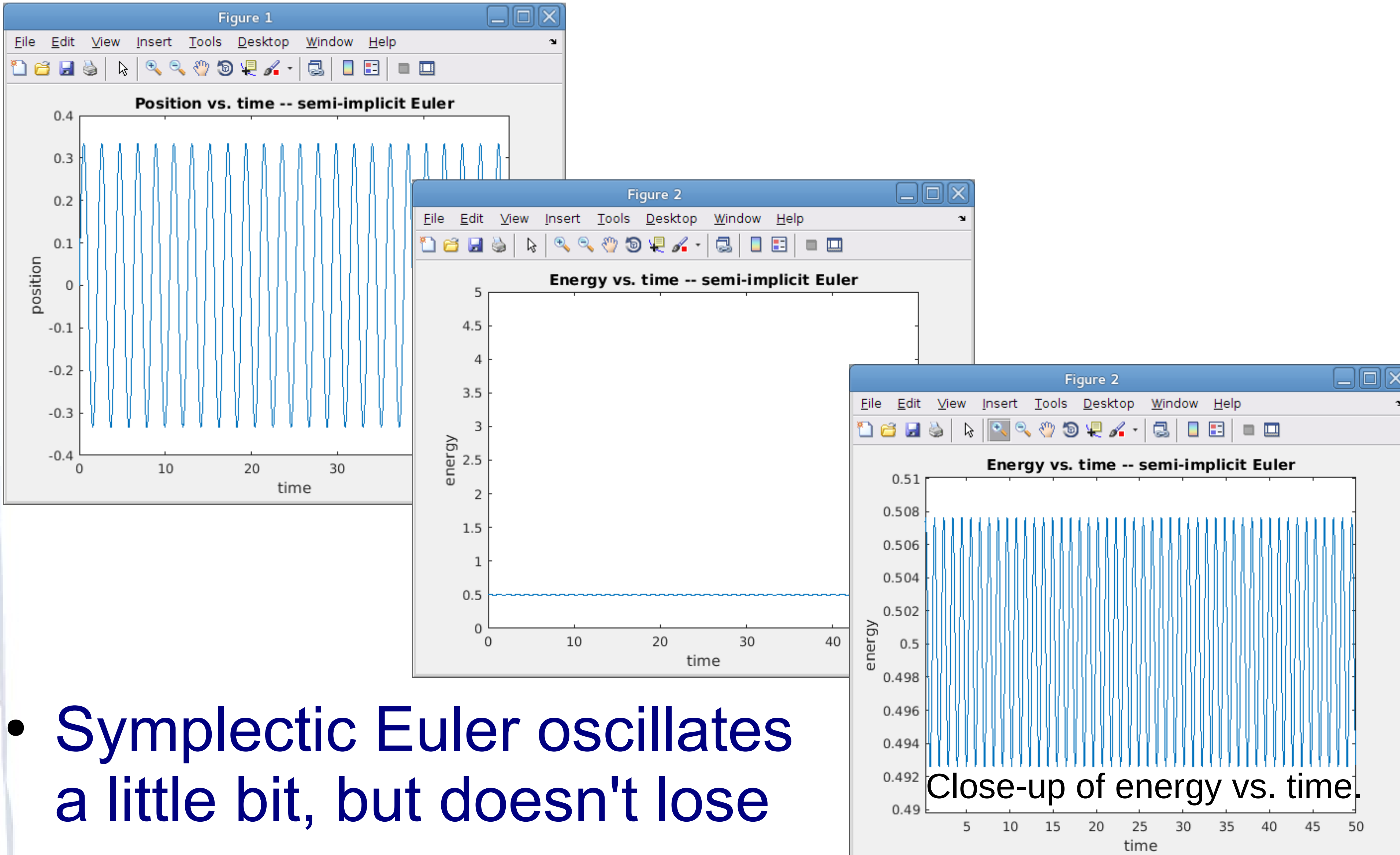
$$v_{n+1} = v_n - \omega^2 h u_n$$

$$u_{n+1} = u_n + h v_{n+1}$$

Note v_{n+1}

```
function ynp1 = semi_implicit_euler_step(yn, t, h)
  % This function takes a semi-implicit Euler step starting
  % at position yn.  Stepsize is h.
  global omega

  % row 1 = pos, row 2 = veloc.
  un = yn(1);
  vn = yn(2);

  vnp1 = vn - h*omega*omega*un;
  unp1 = un + h*vnp1;

  ynp1 = [unp1, vnp1];  % Return should be row vec.

end
```

Symplectic Euler for Harmonic Osc



Close-up of energy vs. time.

- Symplectic Euler oscillates a little bit, but doesn't lose energy.

Why does this work?

- Symplectic Euler

$$v_{n+1} = v_n - \omega^2 h u_n \qquad (1)$$
$$u_{n+1} = u_n + h v_{n+1} \qquad (2)$$

- Insert (1) into (2)

$$v_{n+1} = v_n - \omega^2 h u_n$$
$$u_{n+1} = u_n + h(v_n - h\omega^2 u_n)$$

- Rearranging

$$v_{n+1} = v_n - h\omega^2 u_n$$
$$u_{n+1} = h v_n + (1 - h^2\omega^2) u_n$$

- Matrix form

$$\begin{pmatrix} v_{n+1} \\ u_{n+1} \end{pmatrix} = \begin{pmatrix} 1 & -h\omega^2 \\ h & (1-h^2\omega^2) \end{pmatrix} \begin{pmatrix} v_n \\ u_n \end{pmatrix}$$

- Assume solution

$$\begin{pmatrix} v_n \\ u_n \end{pmatrix} = e_n \begin{pmatrix} i \\ 1 \end{pmatrix} e^{-i\omega t_n}$$

- Insert into matrix

$$e_{n+1} \begin{pmatrix} i \\ 1 \end{pmatrix} e^{-i\omega t_{n+1}} = \begin{pmatrix} 1 & -h\omega^2 \\ h & (1-h^2\omega^2) \end{pmatrix} e_n \begin{pmatrix} i \\ 1 \end{pmatrix} e^{-i\omega t_n}$$

- Rearrange

Recall $h = t_{n+1} - t_n$

$$\frac{e_{n+1}}{e_n} e^{-i\omega h} \begin{pmatrix} i \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & -h\omega^2 \\ h & (1-h^2\omega^2) \end{pmatrix} \begin{pmatrix} i \\ 1 \end{pmatrix}$$

- Define growth factor

$$g_n = \frac{e_{n+1}}{e_n} e^{-i\omega h}$$

- Eigenvalue problem – solve for *g*

$$g_n \begin{pmatrix} i \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & -h\omega^2 \\ h & (1-h^2\omega^2) \end{pmatrix} \begin{pmatrix} i \\ 1 \end{pmatrix}$$

- Condition for solution

$$det \begin{pmatrix} 1-g_n & -h\omega^2 \\ h & (1-h^2\omega^2)-g_n \end{pmatrix} = 0$$

- Characteristic equation $g_n^2 + (h^2\omega^2 - 2)g_n + 1 = 0$

- Solution for g $g_n = \frac{1}{2}\left(2 + (\sqrt{h^2\omega^2 - 4})h\omega - h^2\omega^2\right)$

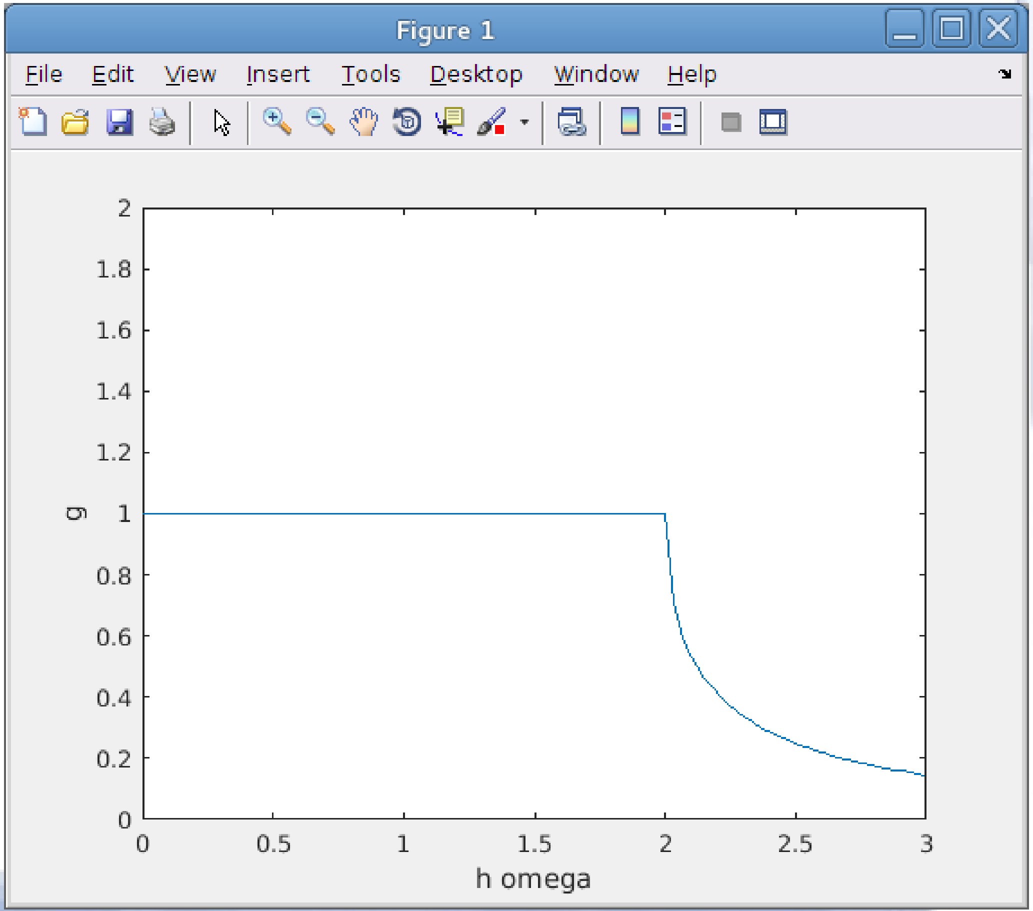
- Not very illuminating, but we can plot it...

Plot growth factor

$$g_n = \frac{1}{2}\left(2 + \left(\sqrt{h^2\omega^2 - 4}\right)h\omega - h^2\omega^2\right)$$

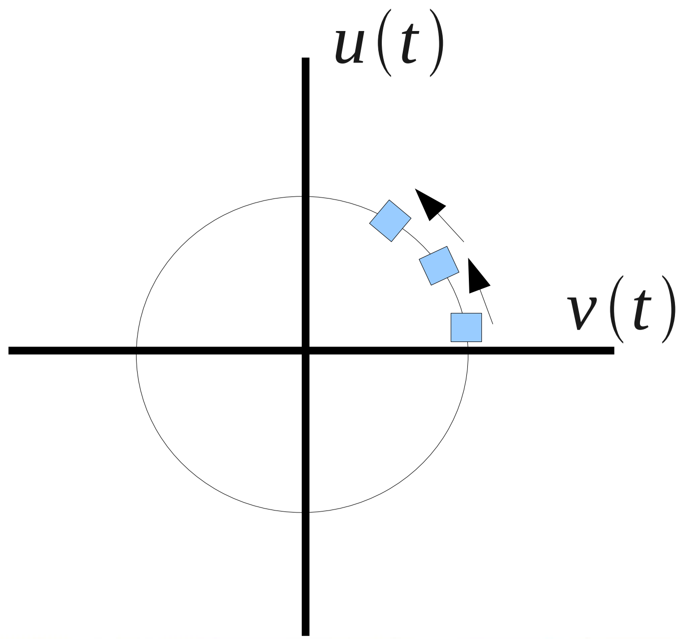
```
>> g = @(x) (2 + x.*sqrt(x.*x-4)-x.*x)/2

>> close all; plot(x, abs(g(x))); hold on; axis([0 3 0 2]);
xlabel('h omega'); ylabel('g')
```

- Growth factor is complex for small values of $h\omega$.
  - Phase can vary.
- Magnitude is 1 for small values of hw.
  - No exponential growth



What happens in phase space?

- Phase space: plot u = x vs. v = dx/dt
- Liouville's theorem: For Hamiltonian (conservative) systems, volumes in phase space are conserved as the system evolves
- Size of volumes should not change as time increases and system evolves.



Test forward Euler

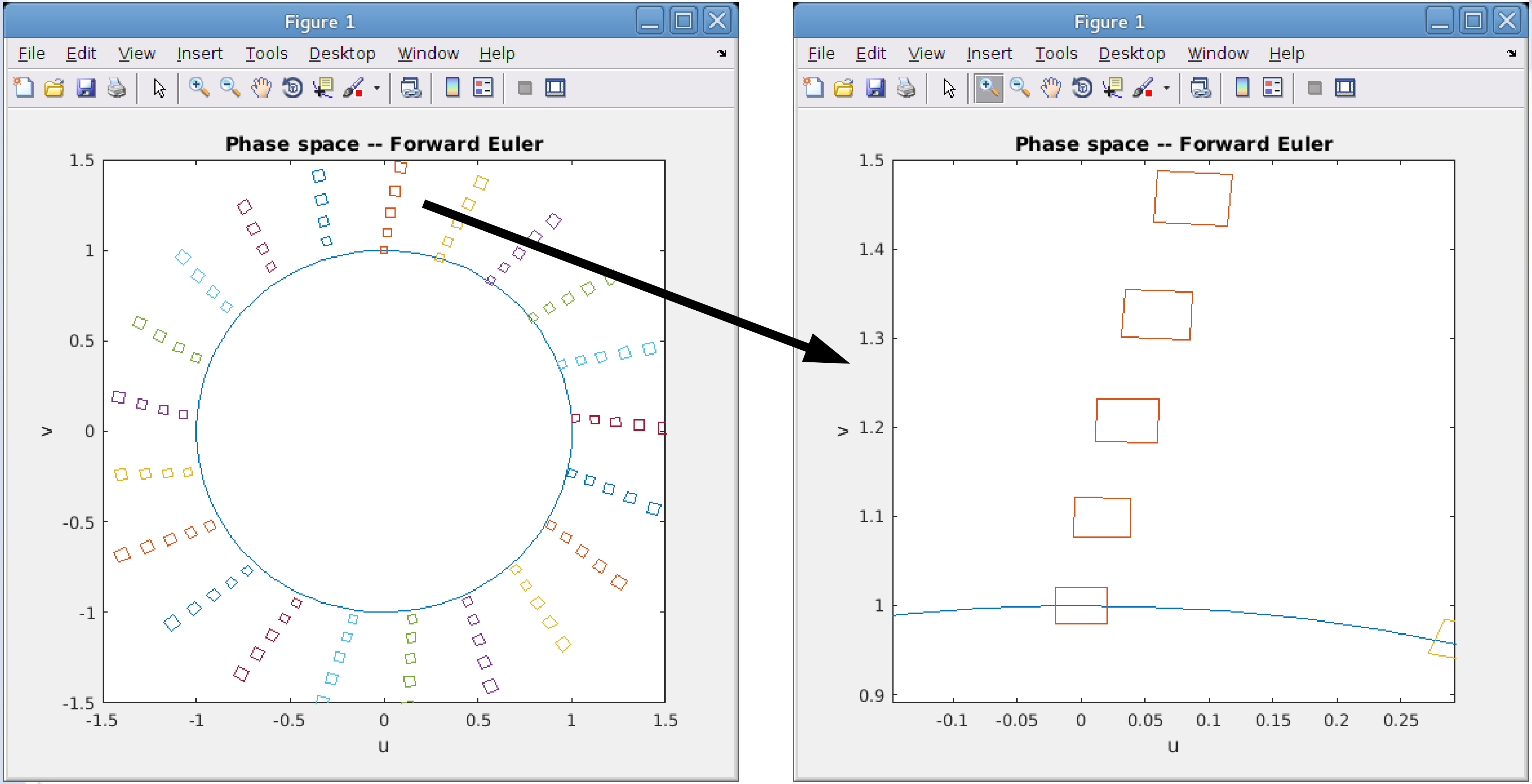
```
% Initial conds
y1 = [-.02; .98];
y2 = [-.02; 1.02];
y3 = [.02; 1.02];
y4 = [.02; .98];

% Plot initial point
u = [y1(1), y2(1), y3(1), y4(1), y1(1)];
v = [y1(2), y2(2), y3(2), y4(2), y1(2)];
plot(u, v)
hold on
xlabel('u')
ylabel('v')
title('Phase space -- Forward Euler')
pause()

% Take 10 steps, then plot.
t = 0;
while (t<Tend)
   for i = 1:10
    y1 = forward_euler_step(y1, t, deltat);
    y2 = forward_euler_step(y2, t, deltat);
    y3 = forward_euler_step(y3, t, deltat);
    y4 = forward_euler_step(y4, t, deltat);
    t = t+deltat;
  end

  u = [y1(1), y2(1), y3(1), y4(1), y1(1)];
  v = [y1(2), y2(2), y3(2), y4(2), y1(2)];
  plot(u, v)
end
```

Draw a box at starting point

Evolve all four corners of box

Draw box after evolution

Evolving forward Euler until t = 30



- Phase space volume not constant.

Test symplectic Euler

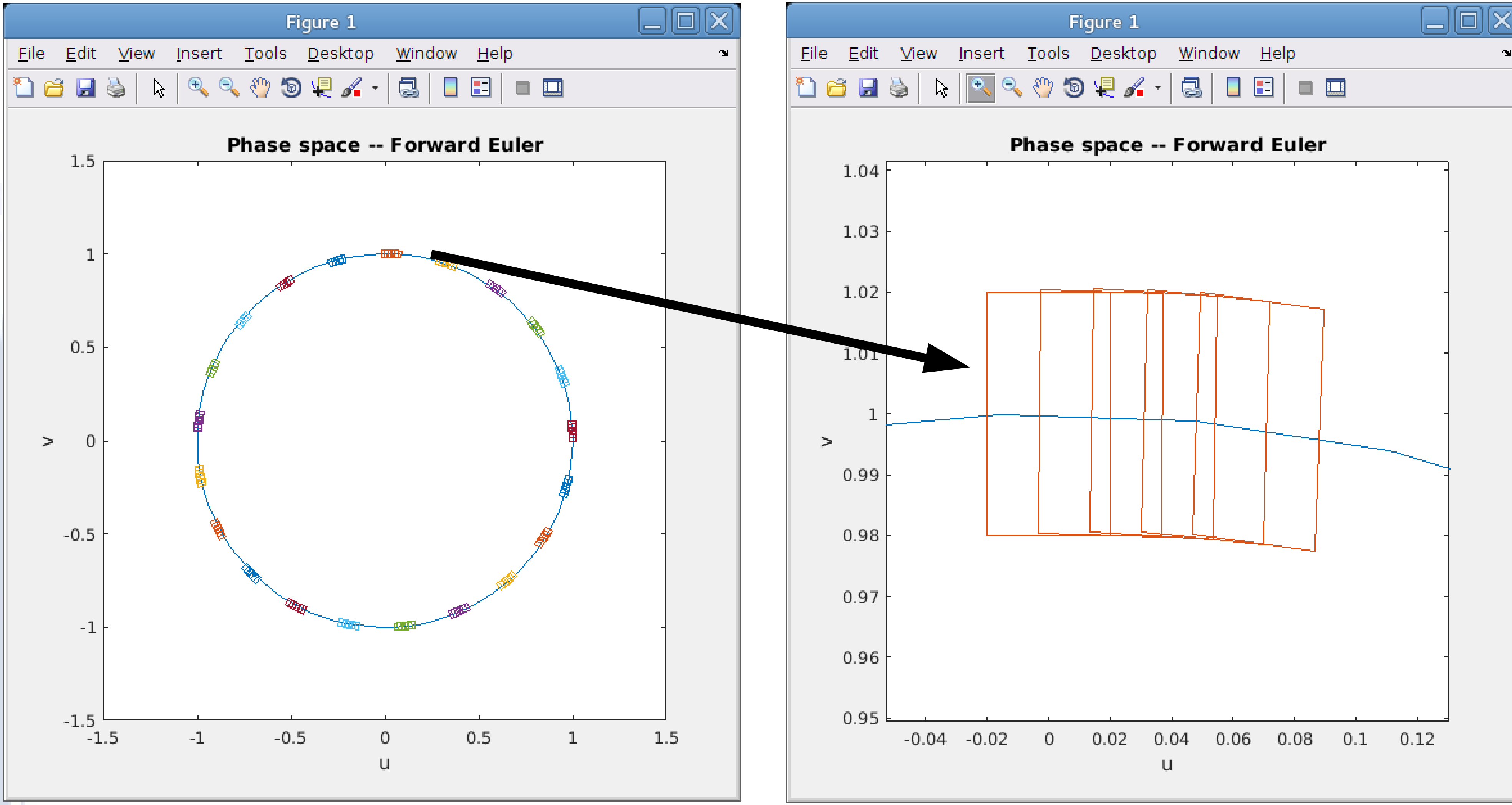
```
% Initial conds
y1 = [-.02; .98];
y2 = [-.02; 1.02];
y3 = [.02; 1.02];
y4 = [.02; .98];

% Plot initial point
u = [y1(1), y2(1), y3(1), y4(1), y1(1)];
v = [y1(2), y2(2), y3(2), y4(2), y1(2)];
plot(u, v)
hold on
xlabel('u')
ylabel('v')
title('Phase space -- Forward Euler')
pause()

% Take 10 steps, then plot.
t = 0;
while (t<Tend)
  for i = 1:10
    y1 = semi_implicit_euler_step(y1, t, deltat);
    y2 = semi_implicit_euler_step(y2, t, deltat);
    y3 = semi_implicit_euler_step(y3, t, deltat);
    y4 = semi_implicit_euler_step(y4, t, deltat);
    t = t+deltat;
  end

  u = [y1(1), y2(1), y3(1), y4(1), y1(1)];
  v = [y1(2), y2(2), y3(2), y4(2), y1(2)];
  plot(u, v)
end
```

Draw a box at starting point

Evolve all four corners of box

Draw box after evolution

Symplectic Euler



- Phase space volume conserved.

Remarks

- Symplectic integrators are important for conservative systems.
  - Hanging pendulum mini-projects.
- They preserve area in phase space.
  - This is an important property since it matches a fundamental behavior of any Hamiltonian (conservative) system.
- I have demonstrated symplectic Euler, but real simulations use higher-order methods.
  - Look for a library ...

Topics covered

- Bucher Tableaux
- Adaptive stepping
- ODE45
- Concept of stiff system
- Energy functional.
- Concept of conservative systems.
- Symplectic integrators
    - Symplectic Euler