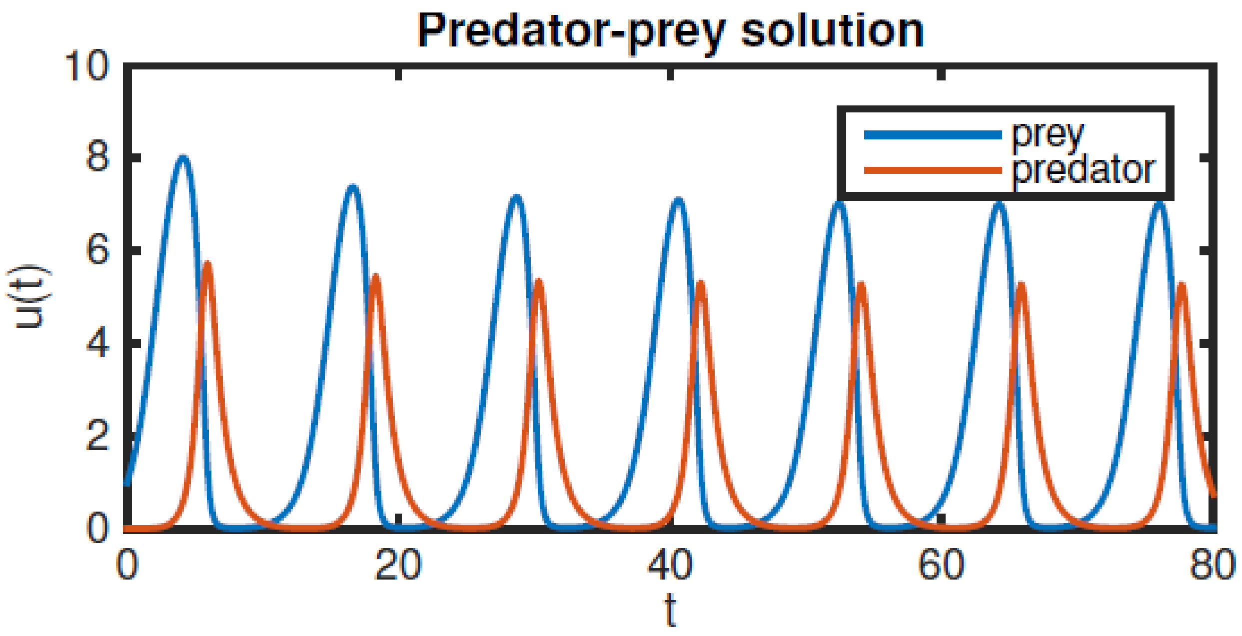
Chapter 6
Initial value problem (IVPs)



Initial value prolbems, or IVPs

- We want to solve differential equations now
- The fundamental problem to solve is for u(t), which must satisfy
$$u'(t) = f\big(t, u(t)\big), \qquad a < t \leq b, \qquad u(a) = u_0$$
- Here the rhs function f is given.
- The constants a, b, u_0 are also given.
- t is the independent variable.
- $u(t)$ is the dependent variable.
- A solution of the problem makes the ODE and the IC identities.

IVPs

- It is possible that u and f could be vector functions, which would make our problem a system of ODEs that usually must be solved simultaneously
- Solving the system (or any ODE) is an integration process, and we need the extra data besides the ODE to find those constants
- What makes the problem an IVP is that all of the data for determining those constants is at $t = a$
- Given that data, we can think of solving the problem as marching across the interval of interest

Initial value prolbems, or IVPs

- The fundamental form

$$u'(t) = f\big(t, u(t)\big), \qquad a < t \leq b, \qquad u(a) = u_0$$

will allow us to solve a wide variety of problems.

- There is much very good software written for this problem
- We will still learn some methods in detail to:
  1. Understand how IVP methods work
  2. Be informed and competent users of software
  3. Maybe even develop your own methods someday

Example IVPs

- We want to solve some problems to see what can happen, then generalize
- Prob 1: solve is for $u(t)$ with
- $u'(t) - ku = 0,\ t > 0,\ u(0) = u_0$
- In standard form, we have
$$f\big(t, u(t)\big) = ku, \qquad a = 0, \qquad b \to \infty$$
- This problem is trivial; can be solved by separation of variables to get
$$u(t) = u_0 e^{kt}$$
- If $k > 0$ and $u_0 > 0$, this could apply to the early stages of population growth (e.g., cells in a petri dish)

Example IVPs

- Prob 2: solve is for $u(t)$ with
$$u'(t) - ku + ru^2 = 0, \qquad t > 0, \qquad u(0) = u_0$$
- In standard form, we have
$$f\big(t, u(t)\big) = ku - ru^2, \qquad a = 0, \qquad b \to \infty$$
- Can again be solved by separation of variables (not as easy!)
$$u(t) = \frac{k/r}{1 + \left(\frac{k}{ru_0} - 1\right) e^{-kt}}$$
- For $k > 0$ and $u_0 > 0$, this function tends to k/r as $t \to \infty$.
- This is a more realistic population model.

Some IVP theory

- We will still need to classify ODEs and use some theory
- The general linear first order ODE is
$$u'(t) = g(t) + h(t)u(t), \qquad t > a, \qquad u(a) = u_0$$
- We can use the integrating factor approach to get the solution
$$\rho(t)u(t) = u_0 + \int_a^t \rho(s)g(s)\, ds\,,$$

with the integrating factor given by
$$\rho(t) = \exp\left[\int h(t)dt\right]$$

- Thus for smooth enough *g* and *h*, we get a solution

IVPs

- Nonlinear problems can be trickier
- Let's solve the logistic model numerically

$$u'(t) = ku - ru^2, \qquad t > 0, \qquad u(0) = u_0$$

- In standard form, we have

$$f\big(t, u(t)\big) = ku - ru^2, \qquad a = 0, \qquad b \to \infty$$

- We can use Matlab's builtin solver `ode45.m` to solve it.
- For $k = 2, r = 2$ and $u_0 = 0.1$.
- The solution should tend to $\frac{k}{r} = 4$ as $t \to \infty$.

IVPs

- Define the functions and constants

```
f = @(t,u) 2*u - 0.5*u.^2;
a = 0;   b = 6;
u0 = 0.1;
```

- Call the solver and plot; 45 time levels were computed

```
[t,u] = ode45(f,[a,b],u0);
length(t)
```

```
ans =
    45
```

```
plot(t,u)
xlabel('t'), ylabel('u(t)'),
```

IVPs

- Define the functions and constants

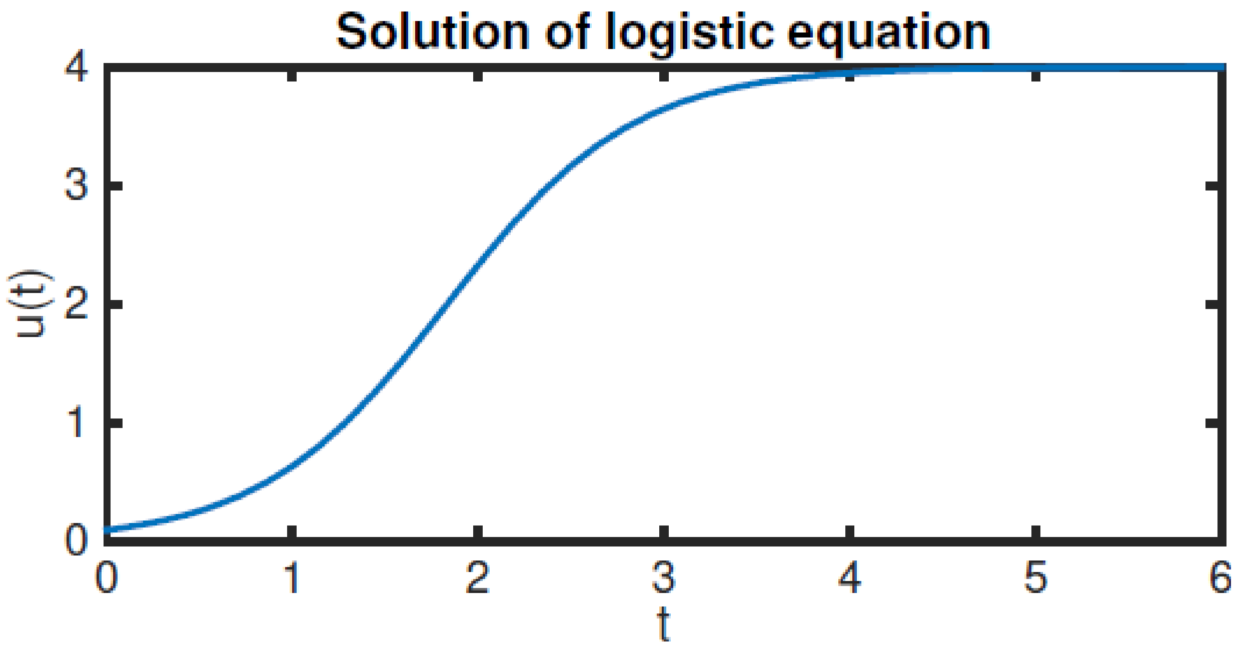
```
f = @(t,u) 2*u - 0.5*u.^2;
a = 0;   b = 6;
u0 = 0.1;
[t,u] = ode45(f,[a,b],u0);
length(t)
```

```
ans =
    45
```

- Call the solver and plot; 45 time levels were computed

```
plot(t,u)
xlabel('t'), ylabel('u(t)'),
```

- Plot the solution
- Solution $u \to 4$ as expected



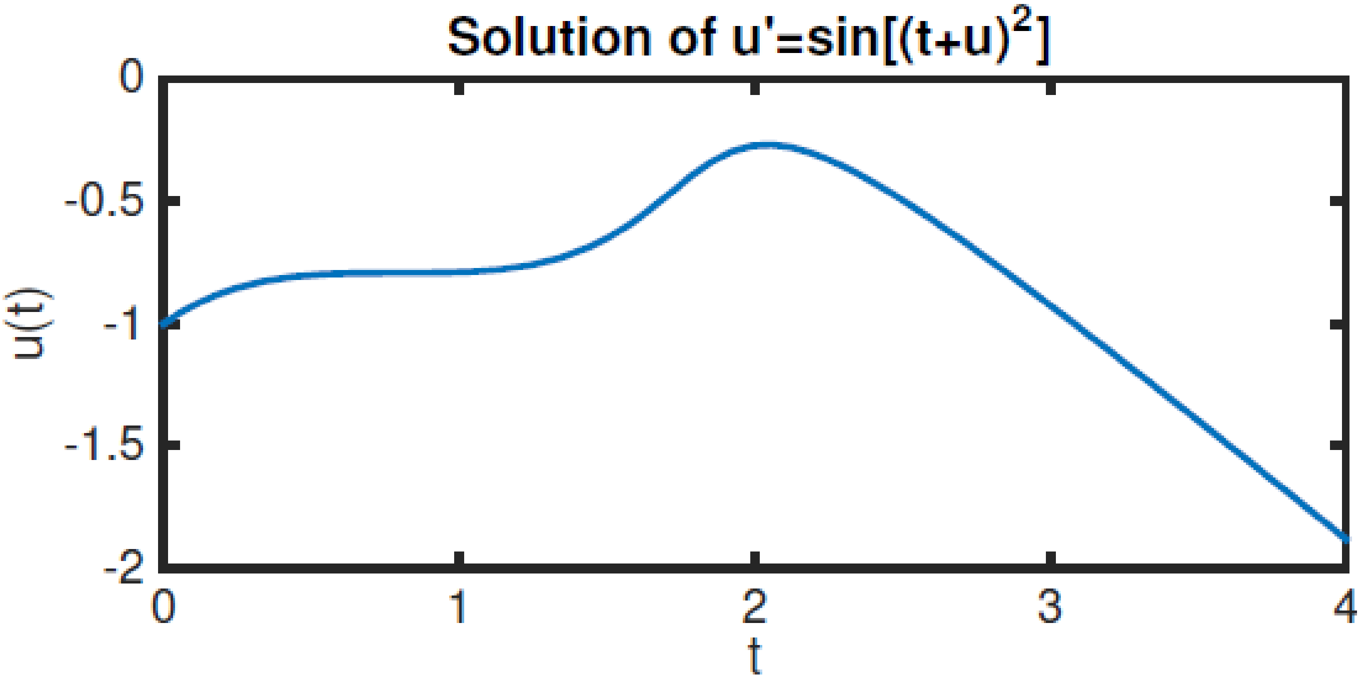
IVPs

- Now solve the nonlinear problem
$$u'(t) = \sin[(u+t)^2], \qquad t > 0, \qquad u(0) = u_0$$
- One could reason that the sin function never lets u' become large
- We have
$$f\big(t, u(t)\big) = \sin[(u+t)^2], \qquad a = 0, \qquad b \to \infty$$
- Use `ode45.m` again

IVPs

- Define the functions and constants, call the solver, plot the solution
- No worries here

```
f = @(t,u) sin( (t+u).^2 );
[t,u] = ode45(f,[0,4],-1);
plot(t,u)
xlabel('t'), ylabel('u(t)'), ti
```



IVPs

- Now solve the nonlinear problem
$$u'(t) = (u+t)^2, \qquad t > 0, \qquad u(0) = u_0$$
- No sin function now to keep u' small
- Will there be trouble?
- We have
$$f\big(t, u(t)\big) = (u+t)^2, \qquad a = 0, \qquad b \to \infty$$
- Use `ode45.m` again

IVPs

- Define the functions and constants, call the solver, plot the solution…
- No, wait… oops. The solver failed…

```
f = @(t,u) (t+u).^2;
[t,u] = ode45(f,[0,1],1);
semilogy(t,u)
xlabel('t'), ylabel('u(t)'), ti
```

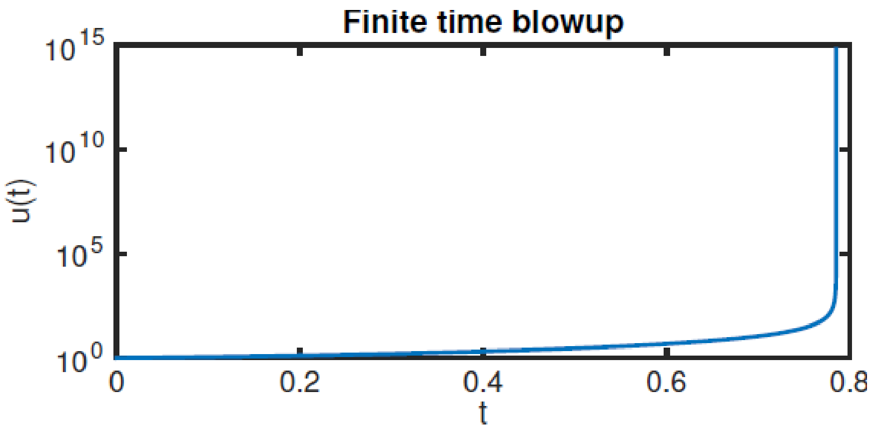
```
Warning: Failure at
t=7.853789e-01.  Unable to
meet integration tolerances
without reducing the step
size below the smallest value
allowed (1.776357e-15) at
time t.
```

IVPs

- Suppose we wanted to look at some values of the solution near blowup
- Here's one way
- We try evaluating the stored structure at certain requested points

```
sol = ode45(f,[0,1],1);
deval(sol,[0.78 0.785 0.7853])
```

```
Warning: Failure at
t=7.853789e-01.  Unable to
meet integration tolerances
without reducing the step
size below the smallest value
allowed (1.776357e-15) at
time t.
ans =
   1.0e+04 *
    0.0185     0.2638     1.2667
```



IVPs: some "theoretical" concerns

- Consider the nonlinear problem

$$u'(t) = 2\sqrt{u(t)}$$

- There are two solutions: $u = 0$ and $u = t^2$. This can be a problem sometimes
- Going back to our fundamental problem

$$u'(t) = f\big(t, u(t)\big), \qquad a < t \leq b, \qquad u(a) = u_0$$

It can be proven that if $\frac{\partial f}{\partial u}$ exists and if $\left|\frac{\partial f}{\partial u}\right| < L$, both for all $a \leq t \leq b$, then there is a unique solution to the IVP for $t \in [a, b]$.

- We will be interested in problems where there may be multiple solutions, but this can cause problems (more later)

Euler's method for IVPs

- Our first numerical method our fundamental problem
$$u'(t) = f\big(t, u(t)\big), \qquad a < t \leq b, \qquad u(a) = u_0$$
- We first convert the interval of interest into a set of evenly-spaced grid points
$$a = t_0, b = t_n, h = \frac{b-a}{n}, \qquad t_i = a + ih, \qquad i = 0,1,\dots,n$$
- h is the step size or grid step
- Approximate the equation at grid point t_i with a forward difference for the derivative and evaluate f there as well

Euler's method for IVPs

- One gets:

$$u'(t) \approx \frac{u(t_{i+1}) - u(t_i)}{t_{i+1} - t_i} \approx f\big(t_i, u(t_i)\big)$$

- But $t_{i+1} - t_i = h$, and let $u(t_i) = u_i$ so that

$$\frac{u_{i+1} - u_i}{h} \approx f(t_i, u_i), \qquad i = 0,1, \dots, n$$

- Now u_i is the exact solution at a grid point; let w_i satisfy the discrete problem:

$$\frac{w_{i+1} - w_i}{h} = f(t_i, w_i), \qquad i = 0,1, \dots, n$$

- What we really want is w_{i+1} …

Euler’s method for IVPs

- One gets:

$$w_{i+1} = w_i + hf(t_i, w_i), \qquad i = 0,1, \dots, n$$

- The initial value at t_0 is given, $w_0 = u_0$
- With that, we can compute w_1, then with that w_2, and so on
- “Time march” across the domain to get the solution at the grid points
- IVPs have this directionality
- Euler’s method has exceptionally easy algebra to do this, it is explicit

Euler solver function

```
function [t,w] = eulerivp(dydt,tspan,y0,n)
% EULERIVP  Euler's method for a scalar initial-value problem.
% Input:
%   dydt     Defines f in y'(t)=f(t,y). (callable function)
%   tspan    endpoints of time interval (2-vector)
%   y0       initial value
%   n        number of time steps (integer)
% Output:
%   t        selected mesh points  (vector, length N+1)
%   w        solution values   (vector, length N+1)

a = tspan(1);  b = tspan(2);
h = (b-a)/n;
t = a + (0:n)'*h;
w = zeros(n+1,1);
w(1) = y0;
for i = 1:n
  w(i+1) = w(i) + h*dydt(t(i),w(i));
end
```

Euler’s method: example

- Reconsider the nonlinear problem

$$u'(t) = \sin[(u+t)^2], \qquad 0 \le t \le 4, \qquad u(0) = u_0 = -1$$

- Then

$$f\big(t, u(t)\big) = \sin[(u+t)^2], \qquad a = 0, \qquad b = 4$$

- This time, Euler’s method uses

$$w_{i+1} = w_i + h\sin[(w_i + t_i)^2], \qquad i = 0,1,\dots,n-1, \qquad w_0 = -1$$

with

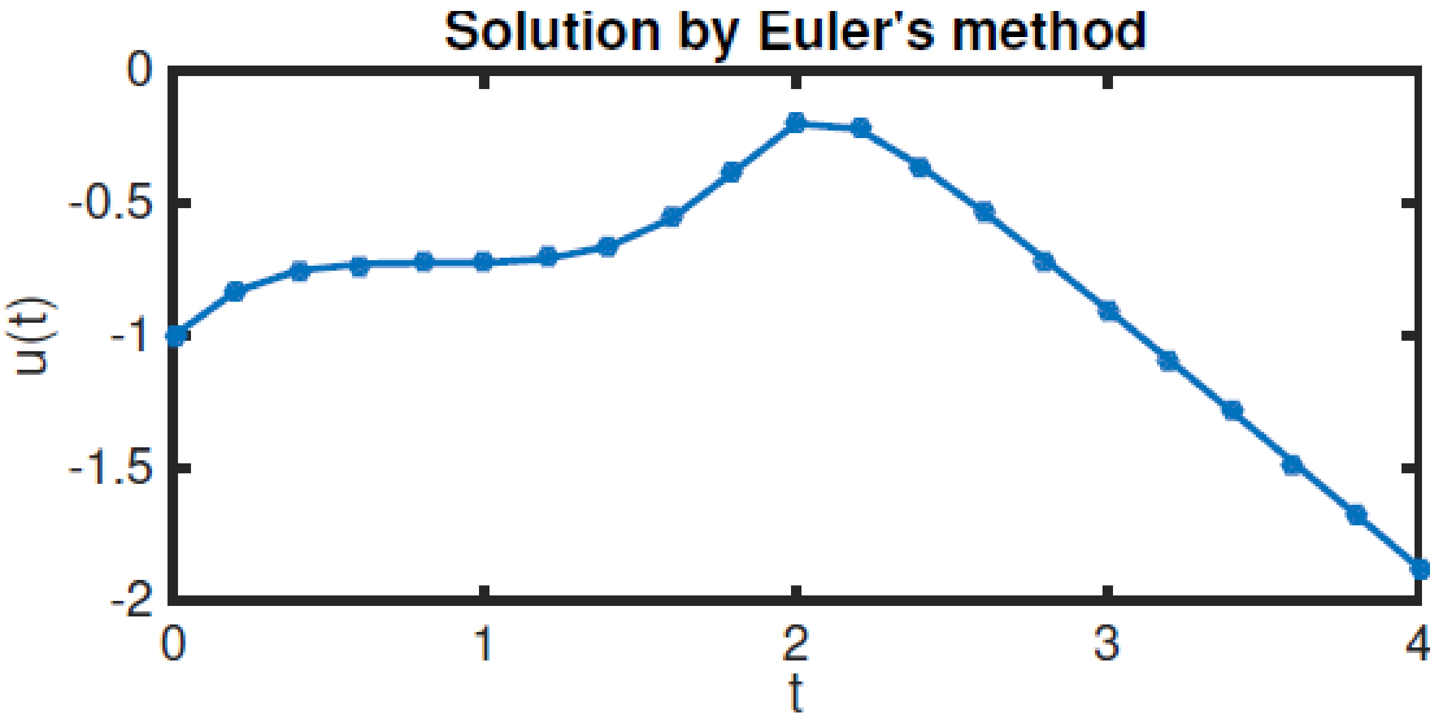
$$h = \frac{4-0}{n}, \qquad t_i = 0 + ih, \qquad i = 0,1,\dots,n$$

- Now for Matlab…

Euler Method ex

- Define the functions and constants, including $n = 20$ here, which makes $h = 0.2$
- Call the solver, plot the solution
- The curve is interpolated between the calculated points
- What's in the solver?

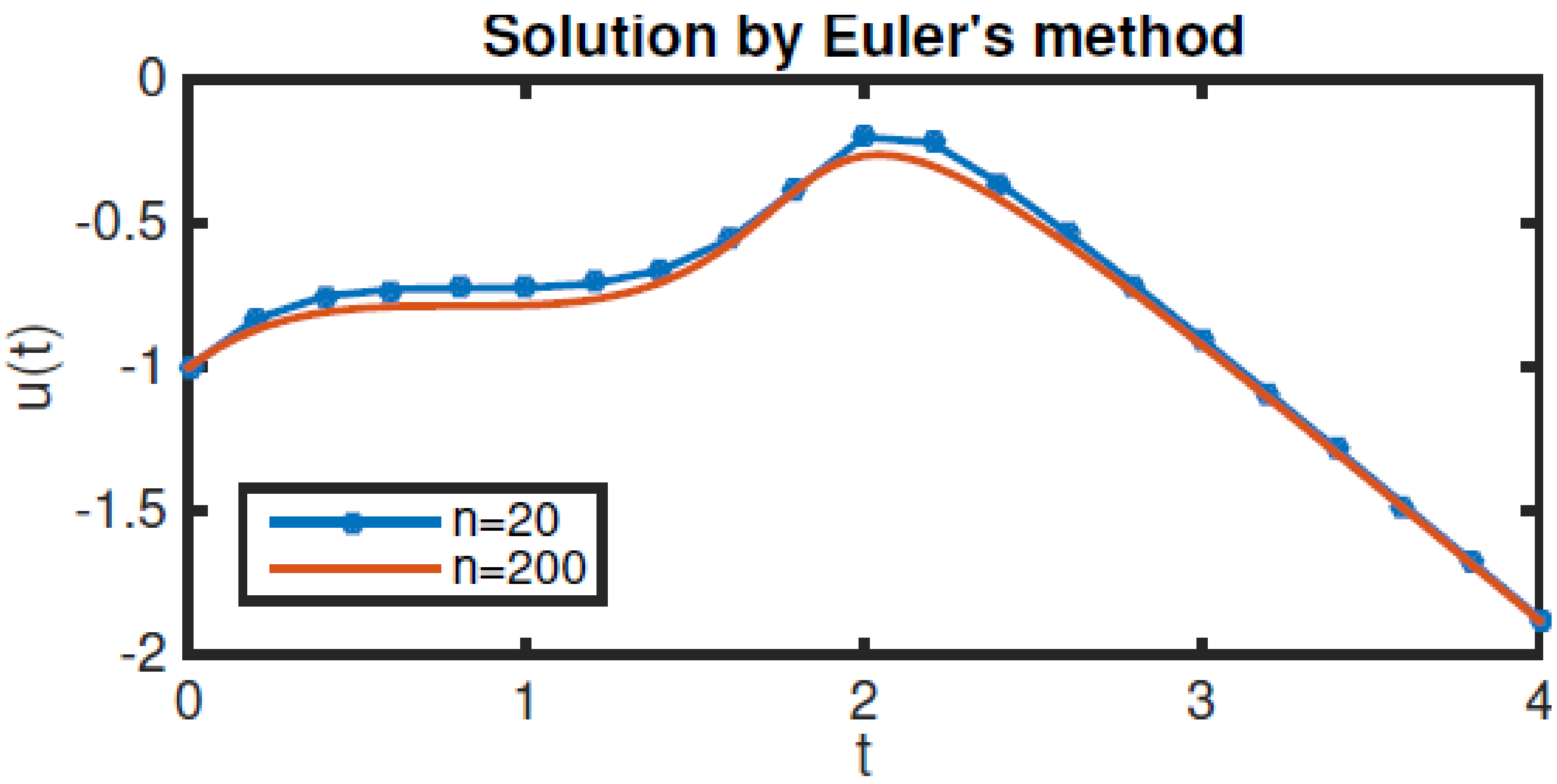
```
f = @(t,u) sin( (t+u).^2 );
a = 0;   b = 4;
u0 = -1;
[t,u] = eulerivp(f,[a,b],u0,20);
plot(t,u,'.-')
xlabel('t'), ylabel('u(t)'), title('Sol
```



Euler Method ex

- Using $n = 200$, which makes $h = 0.2$ and improves the error

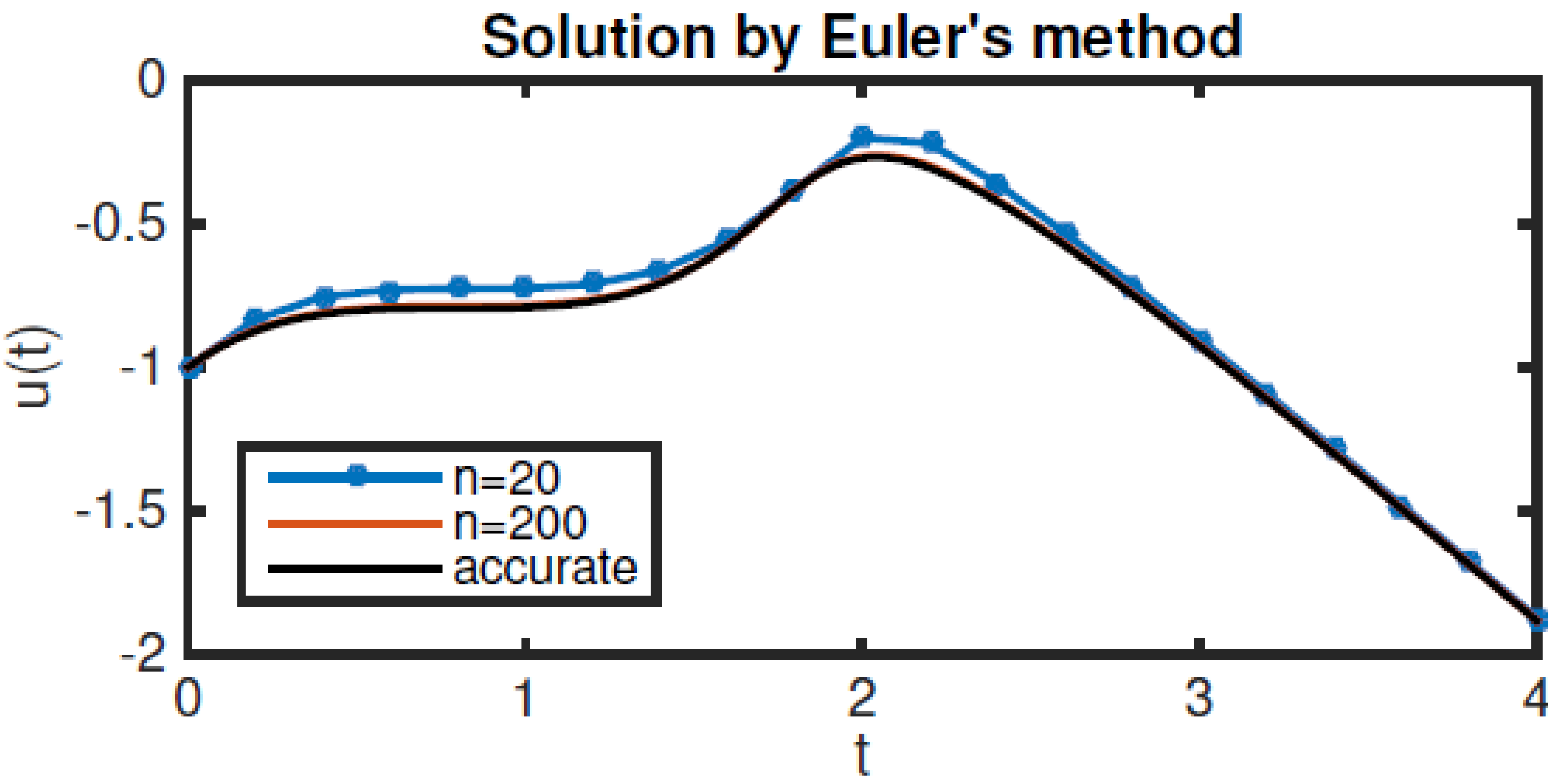
```
[t,u] = eulerivp(f,[a,b],u0,200);
hold on, plot(t,u,'-')
legend('n=20','n=200','location','southwest')
```



Euler Method ex

- Now use a builtin solver ode113 to get a really accuate answer
- Compare both Euler results
- The n=200 case compares well visually

```
u_exact = @(t) deval(uhat,t)';
fplot(u_exact,[a,b],'k-')
legend('n=20','n=200','accurate','location','southwest'
```



Euler Method ex

- Since the results appear to be the same, we need a convergence analysis to see what's going on
- Cut the error by a factor of two, and the (∞-norm) error roughly halves: $O(h)$ error

```
n_ = 50*2.^(0:5)';
err_ = [];
for n = n_'
    [t,u] = eulerivp(f,[a,b],u0,n);
    err_ = [ err_; max(abs(u_exact(t)-u)) ];
end
err_
```

```
err_ =
    0.0300
    0.0142
    0.0069
    0.0034
    0.0017
    0.0008
```

Euler Method analysis

- To analyze the method, we substitute the exact solution into the numerical method
- Then Taylor expand:

$$\begin{aligned}
\hat{u}(t_{i+1}) - [u_i + hf(t_i, u_i)] &= \hat{u}(t_{i+1}) - [\hat{u}(t_i) + hf(t_i, \hat{u}(t_i))] \\
&= [\hat{u}(t_i) + h\hat{u}'(t_i) + \tfrac{1}{2}h^2\hat{u}''(t_i) + \cdots] - [\hat{u}(t_i) + hf(t_i, \hat{u}(t_i))] \\
&= \tfrac{1}{2}h^2\hat{u}''(t_i) + O(h^3),
\end{aligned}$$

- We get second order error when we do this!
- This works for only a single step
- But we always do more than one…

Euler Method analysis

- To analyze the method, we substitute the exact solution into the numerical method
- Then Taylor expand (hatted u_i is exact solution):

$$\begin{aligned}
\hat{u}(t_{i+1}) - \left[u_i + hf(t_i, u_i)\right] &= \hat{u}(t_{i+1}) - \left[\hat{u}(t_i) + hf\left(t_i, \hat{u}(t_i)\right)\right] \\
&= \left[\hat{u}(t_i) + h\hat{u}'(t_i) + \tfrac{1}{2}h^2\hat{u}''(t_i) + \cdots\right] - \left[\hat{u}(t_i) + hf\left(t_i, \hat{u}(t_i)\right)\right] \\
&= \tfrac{1}{2}h^2\hat{u}''(t_i) + O(h^3),
\end{aligned}$$

- We get second order error when we do this!
- This works for only a single step
- But we always do more than one step…

Euler Method analysis

- It is useful to define a general form for single step methods (w_i here):

$$u_{i+1} = u_i + h\phi(t_i, u_i, h), \qquad i = 0, \ldots, n-1$$

- Consider the local truncation error (LTE) for t_{i+1}

$$\tau_{i+1}(h) := \frac{\hat{u}(t_{i+1}) - \hat{u}(t_i)}{h} - \phi(t_i, \hat{u}(t_i), h).$$

- The LTE is a better indicator for our computed error
- Note that in the limit $h \to 0$, we recover the ode: left hand term becomes derivative, which equals f
- Taylor expanding in the LTE will yield O(h) error, which is what we saw from numerical experiment

Euler Method analysis

- What is most desirable is the global error, which is the difference between what we compute and the exact solution at any time
- We can prove the following:
- Theorem: Suppose $|\tau_{i+1}(h)| = Ch^p$, $\left|\frac{\partial \phi}{\partial u}\right| < L \;\forall\; t \in [a.b]$ and $h > 0$, then

$$|\hat{u}(t_i) - u_i| \leq \frac{Ch^p}{L}\left[e^{L(t_i-a)} - 1\right] = O(h^p),$$

as $h \to 0$

- Proof follows…

Euler Method analysis

- Let the global error at t_i be $E\_i = u(t_i) - w_i \rightarrow \hat{u}(t_i) - u_i$
- Then

$$E_{i+1} - E_i = \hat{u}(t_{i+1}) - \hat{u}(t_i) - (u_{i+1} - u_i) = \hat{u}(t_{i+1}) - \hat{u}(t_i) - h\phi(t_i, u_i, h),$$

$$E_{i+1} = E_i + \hat{u}(t_{i+1}) - \hat{u}(t_i) - h\phi(t_i, \hat{u}(t_i), h) + h[\phi(t_i, \hat{u}(t_i), h) - \phi(t_i, u_i, h)].$$

Euler Method analysis

- Let the global error at t_i be $E\_i = u(t_i) - w_i \rightarrow \hat{u}(t_i) - u_i$
- Then
$$E_{i+1} - E_i = \hat{u}(t_{i+1}) - \hat{u}(t_i) - (u_{i+1} - u_i) = \hat{u}(t_{i+1}) - \hat{u}(t_i) - h\phi(t_i, u_i, h),$$

$$E_{i+1} = E_i + \hat{u}(t_{i+1}) - \hat{u}(t_i) - h\phi(t_i, \hat{u}(t_i), h) + h[\phi(t_i, \hat{u}(t_i), h) - \phi(t_i, u_i, h)].$$

- Take absolute, value, use triangle inequality, …

$$|E_{i+1}| \leq |E_i| + Ch^{p+1} + h|\phi(t_i, \hat{u}(t_i), h) - \phi(t_i, u_i, h)|,$$

Euler Method analysis

- Let the global error at t_i be $E\_i = u(t_i) - w_i \rightarrow \hat{u}(t_i) - u_i$
- Then

$$E_{i+1} - E_i = \hat{u}(t_{i+1}) - \hat{u}(t_i) - (u_{i+1} - u_i) = \hat{u}(t_{i+1}) - \hat{u}(t_i) - h\phi(t_i, u_i, h),$$

$$E_{i+1} = E_i + \boxed{\hat{u}(t_{i+1}) - \hat{u}(t_i) - h\phi(t_i, \hat{u}(t_i), h)} + \boxed{h[\phi(t_i, \hat{u}(t_i), h) - \phi(t_i, u_i, h)].}$$

- Take absolute, value, use triangle inequality, …

$$|E_{i+1}| \leq |E_i| + Ch^{p+1} + h|\phi(t_i, \hat{u}(t_i), h) - \phi(t_i, u_i, h)|,$$

Euler Method analysis

- Let the global error at t_i be $E\_i = u(t_i) - w_i \rightarrow \hat{u}(t_i) - u_i$
- Then

$$E_{i+1} - E_i = \hat{u}(t_{i+1}) - \hat{u}(t_i) - (u_{i+1} - u_i) = \hat{u}(t_{i+1}) - \hat{u}(t_i) - h\phi(t_i, u_i, h),$$

$$E_{i+1} = E_i + \hat{u}(t_{i+1}) - \hat{u}(t_i) - h\phi(t_i, \hat{u}(t_i), h) + h[\phi(t_i, \hat{u}(t_i), h) - \phi(t_i, u_i, h)].$$

- Take absolute, value, use triangle inequality, …

$$|E_{i+1}| \le |E_i| + Ch^{p+1} + \boxed{h|\phi(t_i, \hat{u}(t_i), h) - \phi(t_i, u_i, h)|,}$$

- Now use Lipschitz constant L to get rid of ϕ

$$|E_{i+1}| \le Ch^{p+1} + (1 + hL)|E_i|$$

Euler Method analysis

- Then
$$
\begin{aligned}
|E_{i+1}| &\le Ch^{p+1} + (1+hL)|E_i| \\
&\le Ch^{p+1} + (1+hL)\left[Ch^{p+1} + (1+hL)|E_{i-1}|\right] \\
&\vdots \\
&\le Ch^{p+1}\left[1 + (1+hL) + (1+hL)^2 + \cdots + (1+hL)^i\right]
\end{aligned}
$$

- Then replace sum:

$$|E_i| \le Ch^{p+1}\frac{(1+hL)^i - 1}{(1+hL)-1} = \frac{Ch^p}{L}\left[(1+hL)^i - 1\right]$$

- Now bound with exponential…

Euler Method analysis

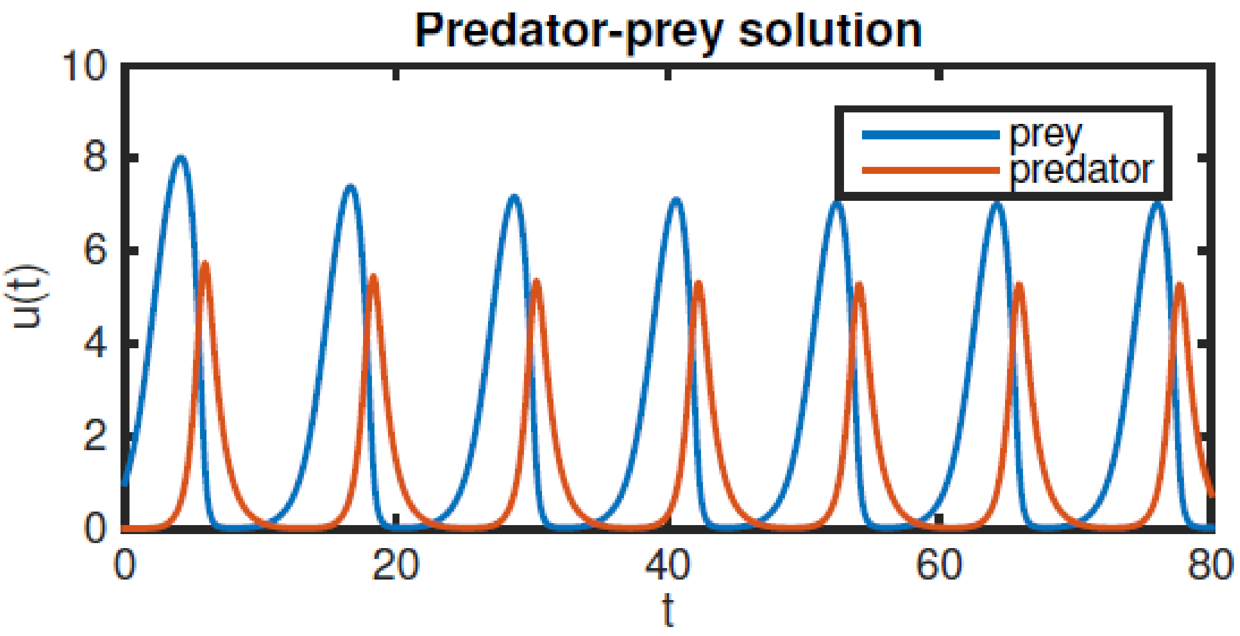
- Simplifying the sum gave

$$|E_i| \leq Ch^{p+1}\frac{(1+hL)^i - 1}{(1+hL)-1} = \frac{Ch^p}{L}\left[(1+hL)^i - 1\right]$$

- Bounding with an exponential gives

$$|\hat{u}(t_i) - u_i| \leq \frac{Ch^p}{L}\left[e^{L(t_i-a)} - 1\right] = O(h^p),$$

Systems of IVPs



IVP systems

- Much of the times, we don't have just a single ODE
- We need to generalize to systems
- Consider this example system:

$$\frac{dy}{dt} = y(1 - \alpha y) - \frac{yz}{1 + \beta y}$$
$$\frac{dz}{dt} = -z + \frac{yz}{1 + \beta y},$$

- There are two constants α and β
- This is the predator-prey model
- y is the prey, z is the predator
- Ex: rabbits and foxes…

IVP systems: predator-prey model

- Our previous approach of writing $u' = f(t, u)$ can be generalized by using vectors for $\boldsymbol{u}$ and $\boldsymbol{f}$
- Convert the system to indexed variables $u_1 = y$ and $u_2 = z$

$$u_1'(t) = f_1(t, \mathbf{u}) = u_1(1 - au_1) - \frac{u_1 u_2}{1 + bu_1}$$
$$u_2'(t) = f_2(t, \mathbf{u}) = -u_2 + \frac{u_1 u_2}{1 + bu_1}$$

- We'll need ICs $u_1(0)$ and $u_2(0)$ (assuming start at $t = 0$)
- Let's write a matlab function to solve the problem using built-in functions first

Pred-prey ex

- Define the constants
- Define the rhs function
- Call the solver
- Plot the output: one row for each time level
- Note the large number of steps

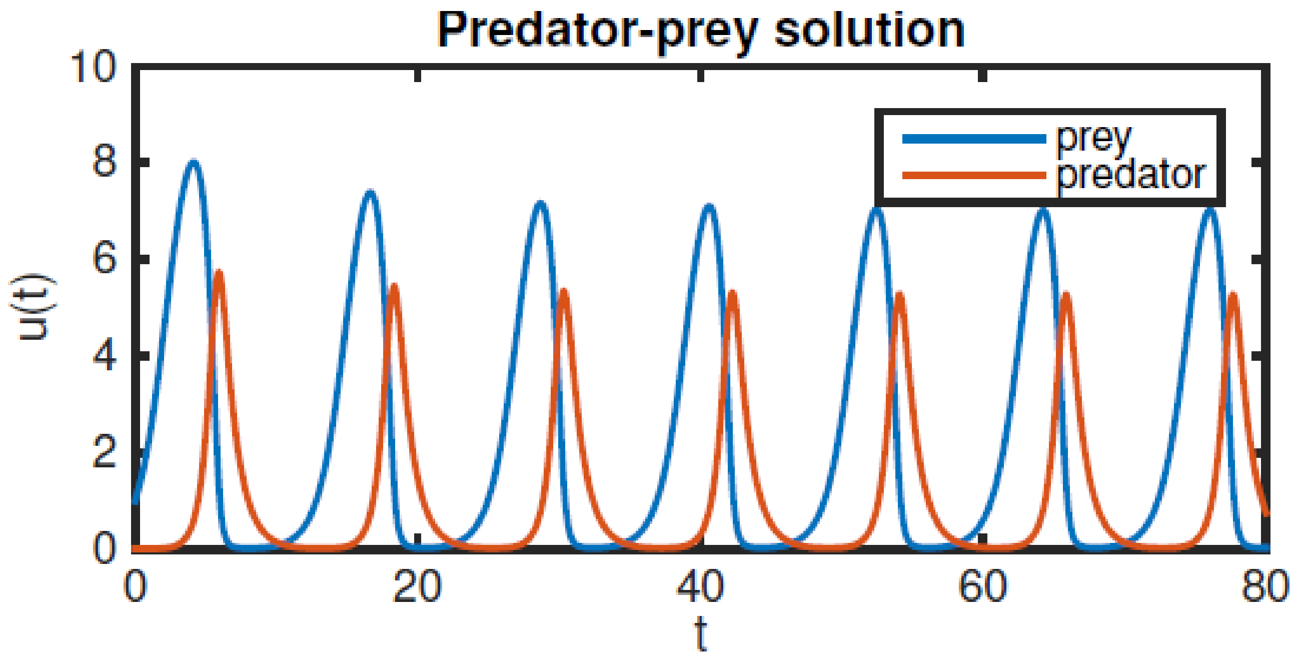
```
function predator

alpha = 0.1;
beta = 0.25;
    function dudt = f(t,u)
        y = u(1);  z = u(2);
        s = (y*z) / (1+beta*y);
        dudt = [ y*(1-alpha*y) - s;
                 -z + s ];
    end

u0 = [1;0.01];
t = linspace(0,80,2001)';
[t,u] = ode45(@f,t,u0);
```

```
size_u = size(u)
y = u(:,1);  z = u(:,2);
plot(t,y,t,z)
xlabel('t'), ylabel('u(t)'), title('Predator-prey solution')
legend('prey','predator')
```

```
size_u =
        2001            2
```

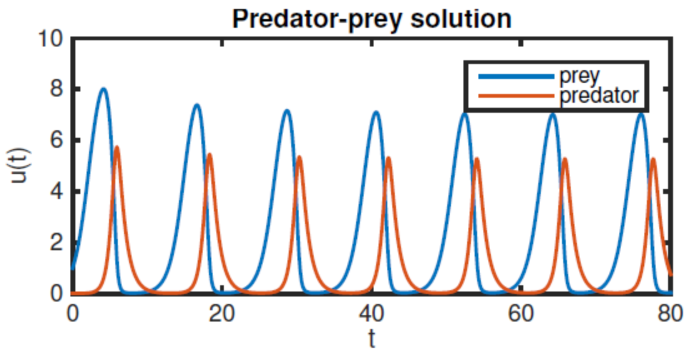
Pred-prey ex

- The solutions:
- At these parameters, the solutions oscillate
- Prey leads the predator
- Peaks decline a bit before possibly being periodic
- Rabbits in my neighborhood before the foxes; we don't see too many rabbits any more

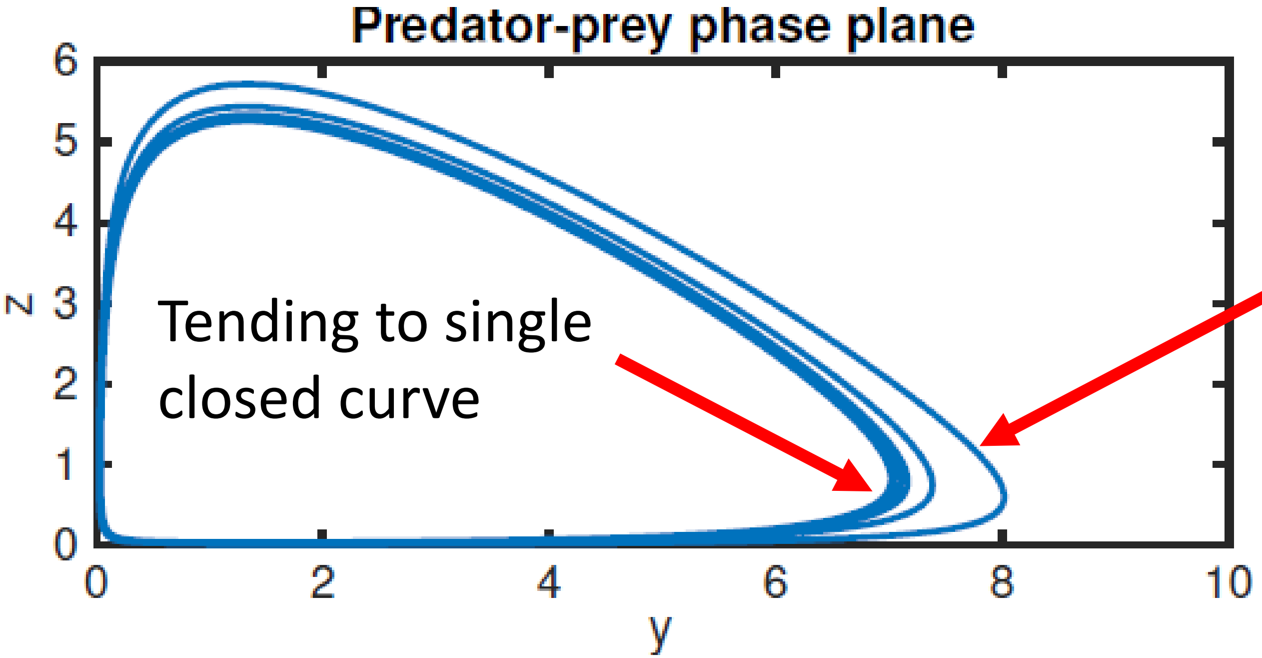


Pred-prey ex

- The qualitative nature of the solutions is often revealed by plotting the components against each other: the phase plane
- (Most useful for autonomous systems with two or three dependent variables)




```
plot(y,z)
xlabel('y'), ylabel('z'), title('Predator-prey phase plane')
```



Earlier peaks decrease as time goes on

IVP systems: higher order problems

- Not all problems come as one or more first order ODEs
- We can use changes of variable to reduce higher order problems to systems of first order ODEs.
- That will allow us to use a our fundamental form to solve even more kinds of problems
- Example: $y'' + (1+y')^3 y = 0, \qquad y(0) = y_0, \quad y'(0) = 0.$
- Use the change of variables: $u_1 = y$ and $u_2 = y'$.
- This gives the system:

$$\begin{aligned} u_1' &= u_2 \\ u_2' &= -(1+u_2)^3 u_1, \end{aligned} \qquad u_1(0) = y_0, \quad u_2(0) = 0.$$

IVP systems: higher order problems

- Consider this example of a system of two 2nd order ODEs

$$\theta_1''(t) - \gamma\theta_1' + \frac{g}{L}\sin\theta_1 + k(\theta_1 - \theta_2) = 0$$
$$\theta_2''(t) - \gamma\theta_2' + \frac{g}{L}\sin\theta_2 + k(\theta_2 - \theta_1) = 0,$$

- These represent two pendula suspended from the same support.
- The total order is 4 because there are two second derivatives
- Use the change of variables: $u_1 = \theta_1, \quad u_2 = \theta_1', \quad u_3 = \theta_2, \quad u_4 = \theta_2'.$
- This gives the system:
- (would need ICs too)
- The rhs forms the vector $\mathbf{f}$

$$\begin{aligned}
u_1' &= u_3 \\
u_2' &= u_4 \\
u_3' &= \gamma u_3 - \frac{g}{L}\sin u_1 + k(u_2 - u_1) \\
u_4' &= \gamma u_4 - \frac{g}{L}\sin u_2 + k(u_1 - u_2),
\end{aligned}$$

IVP systems: predator-prey model

- So we are working with $\mathbf{u}' = \mathbf{f}(t, \mathbf{u}), a \leq t \leq b, \mathbf{u}(a) = \boldsymbol{\alpha}$ now
- Consider approximating $\mathbf{u}$' as before, with
$$\mathbf{u}'_i \approx \frac{\mathbf{u}_{i+1} - \mathbf{u}_i}{h}$$
- Replacing the derivative and evaluate $\mathbf{f}$ as $\mathbf{f}(t_i, \mathbf{u}_i)$ results in Euler's method for a system
$$\mathbf{w}_{i+1} = \mathbf{w}_i + h\mathbf{f}(t_i, \mathbf{w}_i), \qquad i = 0,1, \dots, n-1$$
- This is really just a vectorized version of the Euler method for one equation
- Let's look at function…
- Output will have one row of dependent variables per time step, like Matlab's solvers: if n time steps and m unknowns, output is $n + 1 \times m$

Euler function for systems

- We calculate a new column of dependent variables for each time, and transpose at the end to stick with Matlab's conventions

```
function [t,u] = eulersys(dydt,tspan,u0,n)
% EULERSYS   Euler's method for a system initial-value problem.
% Input:
%   dydt    Defines f in y'(t)=f(t,y). (callable function)
%   tspan   endpoints of time interval (2-vector)
%   u0      initial value (vector, length m)
%   n       number of time steps (integer)
% Output:
%   t       selected mesh points  (vector, length n+1)
%   u       solution values   (array, (n+1)-by-m)

% Time discretization.
a = tspan(1);  b = tspan(2);
h = (b-a)/n;
t = a + (0:n)'*h;

% Initial condition and output setup.
m = length(u0);
u = zeros(m,n+1);
u(:,1) = u0(:);

% The time stepping iteration.
for i = 1:n
    u(:,i+1) = u(:,i) + h*dydt(t(i),u(:,i));
end

% This makes the output conform to MATLAB conventions.
u = u.';
```

Euler method for systems

- Example: $y'' + (1 + y')^3 y = 0, \qquad y(0) = y_0, \quad y'(0) = 0.$
- Converted to system:

$$\begin{aligned} u_1' &= u_2 \\ u_2' &= -(1 + u_2)^3 u_1, \end{aligned} \qquad u_1(0) = y_0, \quad u_2(0) = 0.$$

- Let $y_0 = 0.5, n = 1000$ time steps, $a = 0, b = 2\pi$
- Try it with $y_0 = 0.1, 0.75, 0.9$ too
- You will have to change n
- How to explain smaller y_0 vs larger ones
- Can’t get periodic solutions at 1 or larger

Runge-Kutta methods