Ordinary differential equations (ODE)

Given a problem of the kind

$$y' = f(x,y)$$

Example : $y' = x+y$

The analytical solution y(x) is normally not found. Use some numerical method to approximate $y(x_i)$ for some x-values.

$x_1 = x_0+h$, $x_2 = x_1+h$, and so on.

example

For some simple ODEs we can find the analytical solution

$y' = x + y$ has the solution

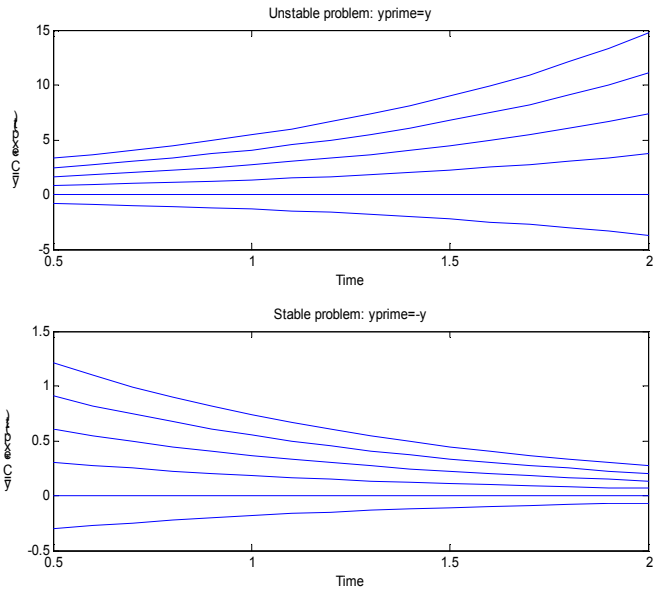
$$y(x) = -x - 1 - C*e^x$$

for any constant C

There are many solutions!

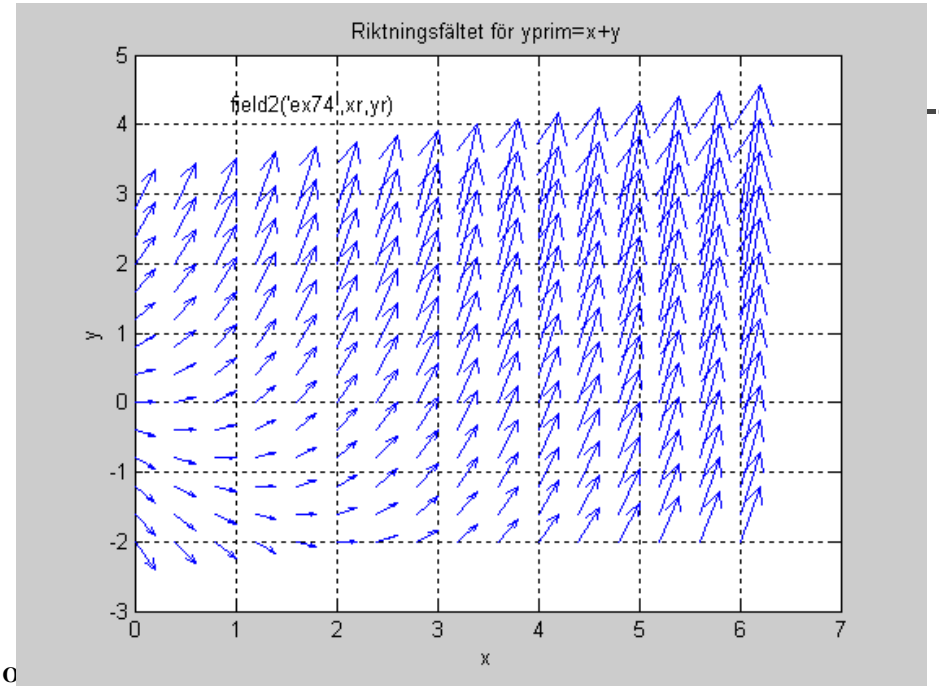
An initial condition $y(x_0)=y_0$ (known value) fixes the solution to a certain curve among the ones in the bundle

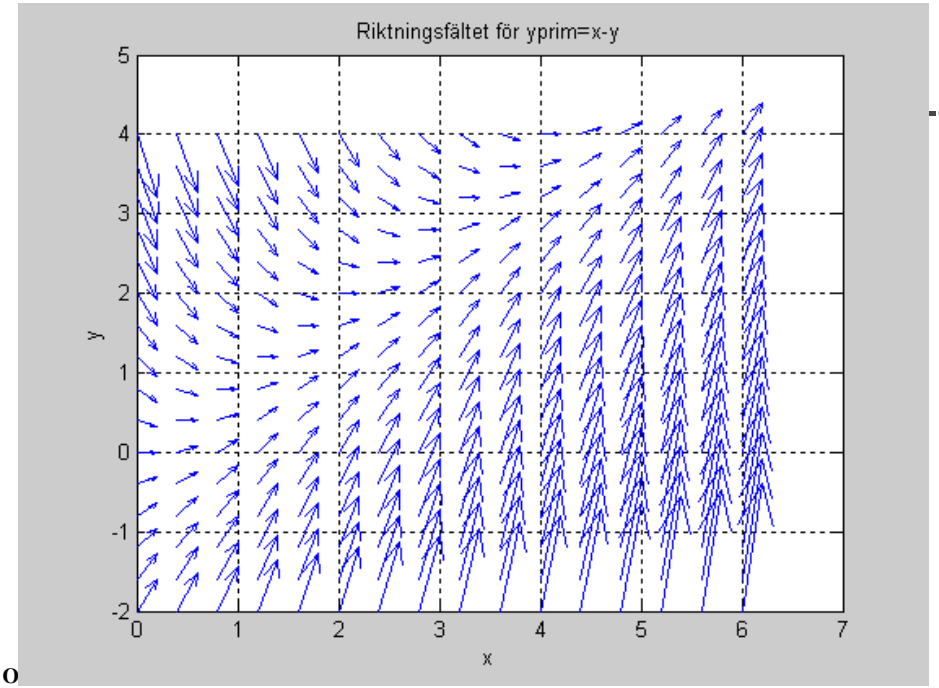
```
>> type odeEX2.m
%plot different solutions to yprime=y
% (an unstable problem)
%I.e. the solution y(t)=C*exp(t)
%Also plot different solutions to yprime=-
y
% (a stable problem)
%I.e. the solution y(t)=C*exp(-t)
%
t=0.5:0.1:2;
subplot(2,1,1)
for C=-0.5:0.5:2
    y=C*exp(t);
    plot(t,y)
    hold on
end
title('Unstable problem: yprime=y')
xlabel('Time')
ylabel('y=C*exp(t)')
subplot(2,1,2)
for C=-0.5:0.5:2
    y=C*exp(-t);
    plot(t,y)
    hold on
end
title('Stable problem: yprime=-y')
xlabel('Time')
ylabel('y=C*exp(-t)')
>>
```

The field of directions

- Since we know y(x)'=f(x,y) for all x it is possible to plot the value of y(x)', the direction.
- Predefined `quiver()` is suitable to use for drawing the field of directions






```
>> help field2
Riktningsfält för en ode
y'=f(t,y)
 OBS! Funkar bara för skalär
ode OBS!
 Call:
field('fkn',xrange,yrange)
>> type ex75
function yprim = ex75(x,y)
%Evaluate the ODE y' = x-y
%Call: yprim=ex75(x,y)
%
yprim=x-y;
>> xr=0:0.4:6;
>> yr=-2:0.4:4;
>> field2('ex75',xr,yr)
>> title('Riktningsfältet för
yprim=x-y')
>> xlabel('x')
>> ylabel('y')
```

```
function field2(fkn,xrange,yrange)
%Riktningsfält för en ode y'=f(t,y)
%OBS! Funkar bara för skalär ode OBS!
%Call: field('fkn',xrange,yrange)
%
hold off
clf
x=xrange(:);
y=yrange(:);
[X Y]=meshgrid(x,y);
%Beräkna lutningskoefficienten i varje punkt
K=feval(fkn,X,Y)
%Beräkna ny punkt på varje linje
h=min(min(diff(X')))/2;
X2=zeros(size(X))+h;
Y2=K.*X2
scale=0;
quiver(X,Y,X2,Y2,scale)
grid on
```



```
%Simulate Conway's game of life
%
format compact
format short
maxgen=input('How many generations: ');
dim=input('Give no. of rows: ');
gen=1;
A=round(rand(dim,dim));
equal=0;
%Zero the border cells
%
A(1,:)=zeros(1,dim);
A(dim,:)=zeros(1,dim);
A(:,1)=zeros(dim,1);
A(:,dim)=zeros(dim,1);
%for i=1:dim
%    A(1,i)=0;
%    A(i,1)=0;
%    A(dim,i)=0;
%    A(i,dim)=0;
%end %of for
disp('First generation');
A
%Simulate for maxgen generations
%
while (gen<maxgen)&(~equal)
   B=nextgen(A);
   gen=gen+1;
   if B==A
      equal=1;
   end
   disp('Next generation');
   A=B
   pause
end %of while
disp('Number of generations and maximum allowed…
 generations until stop')
[gen maxgen]
```

```
function B = nextgen(A)
%Call: B = nextgen(A)
%where A represents the current generation and
%B should represent the comming one in Conway's game of life
%
B=zeros(size(A));
[m, n]=size(A);
for i=2:(m-1)
   for j=2:(n-1)
      %grannar=neighb(A,i,j);
      grannar=sum(sum(A(i-1:i+1,j-1:j+1)))-A(i,j);
      if ((A(i,j)==0) & (grannar==3))
         B(i,j)=1;
      end %of if
      if ((A(i,j)==1) & ((grannar==2)|(grannar==3)))
         B(i,j)=1;
      end
      if ((A(i,j)==1) & ((grannar>=4)|(grannar<2)))
         B(i,j)=0;
      end
      %B(i,j)=((A(i,j)==1) & ((grannar==2)|(grannar==3)));
   end %of inner for
end %of outer for
```



```
function s = neighb(A,r,k)
%Call: s = neighb(A,r,k)
%Calculate s = the number of neighbours
surrounding the (r,k)-element in A
%
sum=0;
for i=r-1:r+1
   for j=k-1:k+1
      sum=sum+A(i,j);
   end
end
s=sum-A(r,k);
```