Next: Continuation of limit cycles **Up:** Equilibrium continuation **Previous:** Equilibrium initialization
Contents

Bratu example

The first example we will look at is a 4-point discretization of the Bratu-Gelfand BVP [22]. This model is defined as follows:

$$x' = y - 2x + ae^x \tag{42}$$

$$y' = x - 2y + ae^y \tag{43}$$

The system is specified as

bratu.m

```
function out = bratu
out{1} = @init;
out{2} = @fun_eval;
out{3} = @jacobian;
out{4} = @jacobianp;
out{5} = @hessians;
out{6} = @hessiansp;
out{7} = [];
out{8} = [];
out{9} = [];
out{10}= @userf1;

end

% --------------------------------------------------------------------------
function dydt = fun_eval(t,kmrgd,a)
dydt = [ -2*kmrgd(1)+kmrgd(2)+a*exp(kmrgd(1));
kmrgd(1)-2*kmrgd(2)+a*exp(kmrgd(2)) ];

% --------------------------------------------------------------------------
function [tspan,y0,options] = init
tspan = [0; 10];
y0 = [0;0];handles = feval(@bratu)
options = odeset('Jacobian',handles(3),'JacobianP', 'handles(4)',...
...,'Hessians',handles(5), 'Hessiansp',handles(6));
% --------------------------------------------------------------------------
function jac = jacobian(t,kmrgd,a)
jac = [ -2+a*exp(kmrgd(1)) 1
1 -2+a*exp(kmrgd(2)) ];

% ------------------------------------------------------------------------
function jacp = jacobianp(t,kmrgd,a)

jacp = [ exp(kmrgd(1))
exp(kmrgd(2)) ];

% ------------------------------------------------------------------------
function hess = hessians(t,kmrgd,a)
hess1=[[a*exp(kmrgd(1)),0];[0,0]];
hess2=[[0,0];[0,a*exp(kmrgd(2))]];
hess(:,:,1) = hess1;
hess(:,:,2) = hess2;

% ------------------------------------------------------------------------
function hessp = hessiansp(t,kmrgd,a)
hessp1=[[exp(kmrgd(1)),0];[0,exp(kmrgd(2))]];
hessp(:,:,1) = hessp1;

%-------------------------------------------------------------------------
function userfun1 = userf1(t,kmrgd,a)
userfun1 = a-0.2;

```

bratu.m

As seen above, a user function is defined to detect all points where $a = 0.2$. This system has an equilibrium at $(x,y,a) = (0,0,0$ which we will continue with respect to a. We first compute 50 points and then extend the curve with another 50 points.

```
global cds
p=[0];ap=[1];
[x0,v0]=init_EP_EP(@bratu,[0;0],p,ap);
opt=contset;
opt=contset(opt,'MaxNumPoints',50);
opt=contset(opt,'Singularities',1);
opt=contset(opt,'Userfunctions',1);
UserInfo.name='userf1';
UserInfo.state=1;
UserInfo.label='u1';
opt=contset(opt,'UserfunctionsInfo',UserInfo);
[x,v,s,h,f]=cont(@equilibrium,x0,[],opt);
[x,v,s,h,f]=cont(x,v,s,h,f,cds);
cpl(x,v,s,[3 1 2]);
```

The above computations can be done by running `testbratu.m` in the directory `Testruns`. The output in the command window is as follows:

```
>> testbratu
first point found
tangent vector to first point found
label = u1, x = ( 0.259174 0.259174 0.200002 )
label = LP, x = ( 1.000001 1.000001 0.367879 )
```

```
a=3.535537e-01
label = NE , x = ( 2.000000 2.000000 0.270671 )
label = u1, x = ( 2.542639 2.542639 0.200000 )
elapsed time  = 0.4 secs
npoints curve = 50
start computing extended curve
label = BP, x = ( 3.000000 3.000000 0.149361 )
elapsed time  = 0.1 secs
npoints curve = 100
```

We note that in the first continuation two zeros of the user function (label u1) were detected, as well as a limit point (label LP) and a neutral equilibrium (label NE). A neutral equilibrium is an equilibrium with two real eigenvalues with sum zero.

In the extension of the run at $(x,y,a) \approx (3.0;3.0;0.15)$ the system has a branch point (label BP).

The script `cpl.m` allows to plot two- or three-dimensional curves. This routine automatically places labels at the singularity points. `cpl(x,v,s,[3 1 2])` plots a 3D-plot with the parameter a on the x-axis and the first and second state variable on the y- and z-axis. The labels of the plot are changed manually by the following commands:

- press 'Edit'$\rightarrow$'Axes Properties',
- go to labels,
- enter 'a' in the Xlabel window, 'x' in the Ylabel and 'y' in the Zlabel,
- close the subwindow.

The resulting curve is plotted in Figure 8.

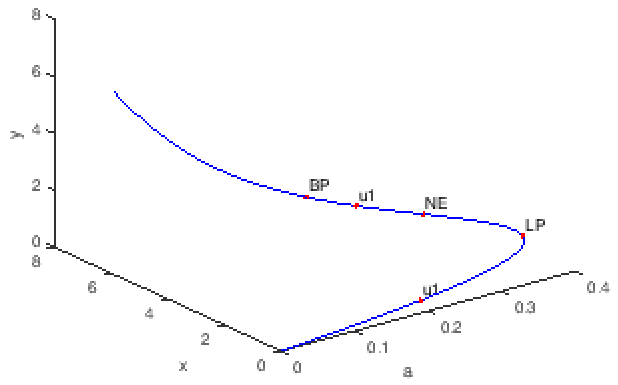

Figure 8: Equilibrium curve of `bratu.m` with a branch point

To select the branch point the output s is used. Since the first and last points are also treated as singular, the array of structures `s` has 7 components and the data concerning the branch point are at $s(6)$. To switch to another branch at the detected branch point, we select that branch point and we use the starter `init_BP_EP`. We start a forward and backward continuation from this point:

```
testbratu;
x1=x(1:2,s(6).index);
p(ap)=x(3,s(6).index);
[x0,v0]=init_BP_EP(@bratu,x1,p,s(6),0.01);
opt=contset(opt,'InitStepsize',0.0001);
[x1,v1,s1,h1,f1]=cont(@equilibrium,x0,v0,opt);
cpl(x1,v1,s1,[3 1 2]);
opt=contset(opt,'Backward',1);
```

```
[x2,v2,s2,h2,f2]=cont(@equilibrium,x0,v0,opt);
cpl(x2,v2,s2,[3 1 2]);
```

Note that p is a vector containing the initial values of all parameters. The above computations can be done by running `testbratu2.m` in the directory `Testruns`. The output in the command window is as follows:

```
>> testbratu2
first point found
tangent vector to first point found
label = u1, x = ( 0.259174 0.259174 0.200002 )
label = LP, x = ( 1.000001 1.000001 0.367879 )
a=3.535537e-01
label = NE , x = ( 2.000000 2.000000 0.270671 )
label = u1, x = ( 2.542639 2.542639 0.200000 )

elapsed time  = 0.1 secs
npoints curve = 50
start computing extended curve
label = BP, x = ( 3.000000 3.000000 0.149361 )

elapsed time  = 0.1 secs
npoints curve = 100
first point found
tangent vector to first point found

elapsed time  = 0.1 secs
npoints curve = 50
first point found
tangent vector to first point found
label = BP, x = ( 3.000000 3.000000 0.149361 )

elapsed time  = 0.1 secs
npoints curve = 50
```

We note that the branch point is "discovered" a second time during the backward continuation of the secondary branch that is rooted at the branch point. All computed curves are plotted together in Figure 9.

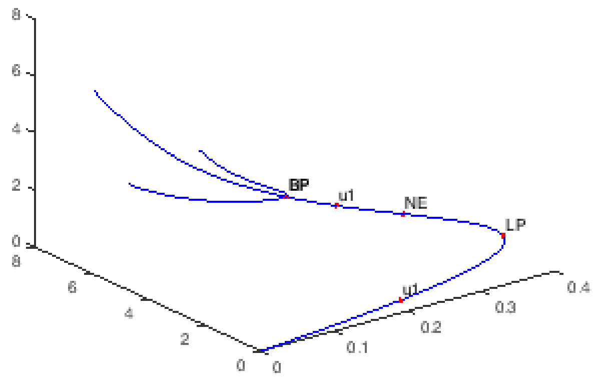

Figure 9: Equilibrium curve of `bratu.m` with new branches rooted in the branch point

Next: Continuation of limit cycles **Up:** Equilibrium continuation **Previous:** Equilibrium initialization
Contents