Numerical Solutions of Boundary Value Problems

1 Linear Shooting

For linear shooting, we will consider problems of the form

$$\begin{aligned} y'' &= p(x)y' + q(x)y + r(x)\,, \quad a \le x \le b\,, \\ y(a) &= \alpha\,, y(b) = \beta\,. \end{aligned}$$

To solve this problem, we will consider the 2 initial value problems

$$\begin{aligned} y_1'' &= p(x)y_1' + q(x)y_1 + r(x)\,, \quad a \le x \le b\,, \\ y_1(a) &= \alpha\,, y'(a) = 0\,, \end{aligned}$$

and

$$\begin{aligned} y_2'' &= p(x)y_2' + q(x)y_2\,, \quad a \le x \le b\,, \\ y_2(a) &= 0\,, y'(a) = 1\,. \end{aligned}$$

Notice that the y_2 equation does not have the $r(x)$ term. We can approximate the solutions to the two initial value problems with any of our solvers. The solution to the boundary value problem is then approximated by

$$y = y_1 + \frac{\beta - y_1(b)}{y_2(b)} y_2\,.$$

In class we considered the example

$$y'' = y' + 2y + \cos(x)\,, \quad 0 \le x \le \frac{\pi}{2}\,, \quad y(0) = -.3\,, \quad y(\pi/2) = -.1\,.$$

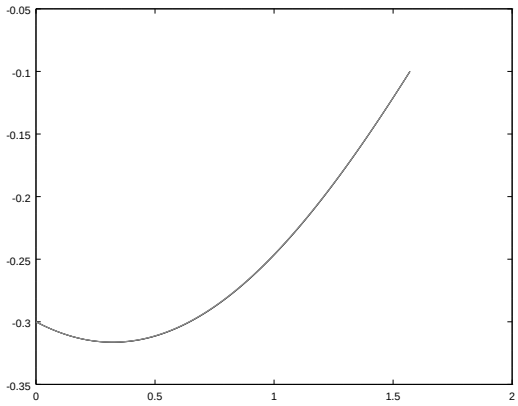
The exact solution is given by $y = -.1(\sin(x) + 3\cos(x))$. To use linear shooting I first need to set up the equations for y_1 and y_2. The following Matlab functions are used:

```
function dy=bvp1(t,y)

dy(1)=y(2);
dy(2)=y(2)+2*y(1)+cos(t);

end

function dy=bvp2(t,y)

dy(1)=y(2);
dy(2)=y(2)+2*y(1);
```

Here is the matlab session used to approximate the solution to the boundary value problem. I use a 4th order Runge-Kutta method, but any will do.

```
octave:28> [t,y1]=rk4('bvp1',[-.3 0],0,pi/2,100);
octave:29> [t,y2]=rk4('bvp2',[0 1],0,pi/2,100);
octave:30> y=y1+(-.1-y1(end,:))./y2(end,:).*y2;
warning: product: automatic broadcasting operation applied
octave:31> plot(t,y(:,1),'r')
octave:32> hold on
octave:33> plot(t,-.1*(sin(t)+3*cos(t)),'g')
octave:34> diary off
```

Here is the graph that was produced.



2 Nonlinear Shooting

In this section we will consider the solution to boundary value problems of the form

$$y'' = f(x, y, y')\,, \quad a < x < b\,, \tag{1}$$
$$y(a) = A\,, \tag{2}$$
$$y(b) = B\,. \tag{3}$$

In the shooting method, we consider the boundary value problem as an initial value problem and try to determine the value $y'(a)$ which results in $y(b) = B$. As in class I will apply these methods to the problem

$$y'' = -\frac{(y')^2}{y}\,, \quad y(0) = 1\,, \quad y(1) = 2\,.$$

The exact solution is given by $y = \sqrt{3x+1}$.

3 Shooting - Secant Method

For the shooting method, we consider the problem

$$y'' = f(x, y, y')\,, \tag{4}$$
$$y(a) = A\,, \tag{5}$$
$$y'(a) = t\,, \tag{6}$$

We let

$$m(t) = f(b; t) - B$$

where $f(b;t)$ is the solution to (4) using the value t. We wish to find a zero of $m(t)$ to solve the boundary value problem. In this case we use the secant method to locate the zero. Recall, to solve $f(x) = 0$ using the secant method we make successive approximations using the iteration

$$x_{n+1} = x_n - \frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})} f(x_n)\,.$$

We need two guess to start the process, then we just proceed until the iterations converge. Here is the code I used to approximate the solution to (1) using the shooting secant method:

```
function [x,y]=bvpsec(t1,t2)

[x1,y1]=rk23(@odes,[1 t1],0,1,0.000001);
[x2,y2]=rk23(@odes,[1 t2],0,1,0.000001);
i=1;
```

```
m1=y1(end,1)-2;
m2=y2(end,1)-2;
while(abs(t2-t1)>0.0001)
    tmp=t2;
    t2=t1-(t1-t2)/(m1-m2)*m1;
    t1=tmp;
    [x1,y1]=rk23(@odes,[1,t1],0,1,0.000001);
    [x2,y2]=rk23(@odes,[1,t2],0,1,0.000001);
    m1=y1(end,1)-2;
    m2=y2(end,1)-2;
    i=i+1;
    if (i>100)
       error("Did not converge")
    end
end
i
x=x2;
y=y2;
end

function yp=odes(t,y)

  yp=zeros(1,2);

      yp(1)=y(2);
      yp(2)=-y(2)^2/y(1);
end
```

If we run the code with input parameters 1 and 2, it takes 6 iterations to get the solution shown in figure 1

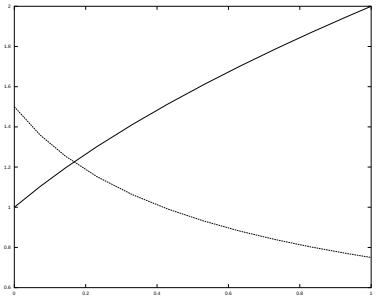


Figure 1: Approximation to the solution of (1) using the shooting method in combination with the secant method. The plot includes $y(x)$ as well as $y'(x)$.

4 Shooting Method - Newton's Method

Newton's root finding method is much faster and can produce more accurate results then the secant method. The iteration used to find a solution to $f(x) = 0$ is given by

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}\,.$$

However to apply this method to find a root of $m(t)$, we must know $m'(t)$. Here $m(t)$ is defined in terms of the solution to differential equation evaluated at a point b. In general, we can't solve the differential equation or we wouldn't need the numerical approximation. So we must find a method of approximating $m'(t)$.

We first note that

$$m(t) = f(b;t) - B$$

so that

$$m'(t) = \left.\frac{\partial f(x;t)}{\partial t}\right|_{x=b} .$$

Now we will consider f as a function of the three variables x, y and y', for the moment ignoring the relationship between y and y'. We can then differentiate the following relation by t

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

to get

$$\frac{\partial y''}{\partial t} = \frac{\partial f}{\partial y}\frac{\partial y}{\partial t} + \frac{\partial f}{\partial y'}\frac{\partial y'}{\partial t} .$$

We can let $z(x,t) = \frac{\partial}{\partial t} f(x,t)$. Now z satisfies the boundary value problem

$$z'' = \frac{\partial f}{\partial y} z + \frac{\partial f}{\partial y'} z' , \quad z(0) = 0 , \quad z'(0) = 1 .$$

Now $m'(t) = z(b)$. So if we wish to apply this method to our sample problem (1), we must solve the system

$$\begin{aligned} y' &= u , \\ u' &= -\frac{u^2}{y} , \\ z' &= v , \\ v' &= \frac{u^2}{y^2} v - 2\frac{u}{y} v , \\ y(0) &= 1 , \\ u(0) &= t , \\ z(0) &= 0 , \\ v(0) &= 1 . \end{aligned}$$

Then we can approximate $m'(t)$ by $z(1)$ and use Newton's formula to update the solution. The code is given below:

```
function [x,y]=bvpnewt(t1)

[x,y]=rk23(@odes,[1,t1,0,1],0,1,0.000001);
i=1;
m=y(end,1)-2;
t2=t1-m/y(end,3);
while(abs(t2-t1)>0.00001)
   t1=t2;
   [x,y]=rk23(@odes,[1,t1,0,1],0,1,0.000001);
   m=y(end,1)-2;
   t2=t1-m/y(end,3);
   i=i+1;
   if i>100
      error("Did not converge")
   end
end
y=y(:,1:2);
i
end

function yp=odes(t,y)

    yp=zeros(1,4);
```

```
    yp(1)=y(2);
    yp(2)=-y(2)^2/y(1);
    yp(3)=y(4);
    yp(4)=y(2)^2/y(1)^2*y(3)-2*y(2)/y(1)*y(4);
end
```

Running this code with an initial $t = 1$ takes 4 iterations to get to the same accuracy as the secant method. This method must solve a larger system, so each iteration is more work. However fewer iterations are required. The output is almost identical to the shooting method, so there is no need provide a graph.