Numerical Solutions of Boundary Value Problems

In these note we will consider the solution to boundary value problems of the form

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

We will consider two common methods, shooting and finite differences. 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 $y(b) = B$. Finite differences converts the continuous problem to a discrete problem using approximations of the derivative. 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 $y = \sqrt{3x+1}$.

1 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]=ode45(@odes,[0,1],[1,t1]);
[x2,y2]=ode45(@odes,[0,1],[1,t2]);
i=1;
m1=y1(end,1)-2;
m2=y2(end,1)-2;
while(abs(t2-t1)>0.000001)
    tmp=t2
    t2=t1-(t1-t2)/(m1-m2)*m1;
    t1=tmp;
    [x1,y1]=ode45(@odes,[0,1],[1,t1]);
    [x2,y2]=ode45(@odes,[0,1],[1,t2]);
    m1=y1(end,1)-2;
    m2=y2(end,1)-2;
    i=i+1;
end
i
x=x2;
y=y2;
```

```
end

function yp=odes(t,y)

    yp=zeros(2,1);
    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

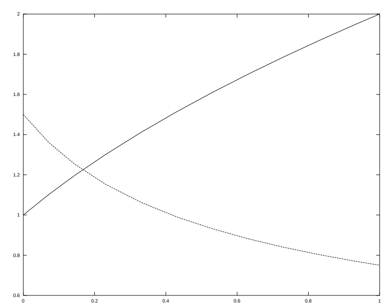

Figure 1: Approximation to the solution of (1) using the shooting method in combination with the secant method. The plot includes $y(x)$ as well as $y'(x)$.

2 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]=ode45(@odes,[0,1],[1,t1,0,1]);
i=1;
m=y(end,1)-2;
t2=t1-m/y(end,3);
while(abs(t2-t1)>0.000001)
   t1=t2;
   [x,y]=ode45(@odes,[0,1],[1,t1,0,1]);
   m=y(end,1)-2;
   t2=t1-m/y(end,3);
   i=i+1;
end
y=y(:,1:2);
i
end

function yp=odes(t,y)

    yp=zeros(4,1);
    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.

3 Finite Difference Method

For the finite difference method, we pick $N + 1$ discrete points in the interval $[a, b]$ by

$$x_i = a + i\Delta x\,, \quad i = 0 \ldots N\,, \quad \Delta x = \frac{b-a}{N}\,.$$

and we let y_i be our approximation of $y(x_i)$. In the solution to the heat equation notes we show that

$$y''(x_i) \sim \frac{y_{i+1} - 2y_i + y_{i-1}}{\Delta x^2}\,, \tag{7}$$

$$y'(x_i) \sim \frac{y_{i+1} - y_{i-1}}{2\Delta x}\,. \tag{8}$$

Using the boundary conditions $y_0 = A$ and $y_N = B$ together with (7), we get $N + 1$ equations in $N + 1$ unknowns. The equations are nonlinear and we must use Newton's method to find a solution. Recall Newton's method for systems is given by

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

We can now apply the finite difference approximation to (1).

$$\vec{F} = \begin{pmatrix} y_0 - 1 \\ \vdots \\ \frac{y_{i+1}-2y_i+y_{i-1}}{\Delta x^2} + \frac{\left(\frac{y_{i+1}-y_{i-1}}{2\Delta x}\right)^2}{y_i} \\ \vdots \\ y_N - 2 \end{pmatrix}$$

The nonzero entries of the Jacobian are then

$$\begin{aligned} J_{i,i-1} &= \frac{1}{\Delta x^2}\left(1 - \frac{2(y_{i+1} - y_{i-1}}{4y_i}\right) \,, \\ J_{i,i} &= -\frac{2}{\Delta x^2} \,, \\ J_{i,i+1} &= \frac{1}{\Delta x^2}\left(1 + \frac{2(y_{i+1} - y_{i-1}}{4y_i}\right) \,, \end{aligned}$$

for $i = 1, \ldots, N - 1$. Here is the code used to implement the method:

```
function [x,y]=findiff(n)

    x=linspace(0,1,n);
    y=zeros(n,1);
    for i=1:n
        y(i)=x(i)+1;
    end

    dy=-J(y)\f(y);
    while(norm(dy,2)>0.001)
        y=y+dy;
        dy=-J(y)\f(y);
    end
end


function y1=f(y)
    n=length(y);
    h=1/n;
    y1=zeros(n,1);
    y1(1)=y(1)-1;
    for i=2:n-1
      y1(i)=(y(i-1)-2*y(i)+y(i+1))/h^2+(y(i+1)-y(i-1))^2/y(i)/4/h^2;
    end
    y1(n)=y(n)-2;
end

function J1=J(y)
    n=length(y);
    h=1/n;
    J1=zeros(n,n);
    J1(1,1)=1;
    for i=2:n-1
```

```
        J1(i,i+1)=1/h^2*(1+(y(i+1)-y(i-1))/2/y(i));
        J1(i,i)=1/h^2*(-2-(y(i+1)-y(i-1))^2/4/y(i)^2);
        J1(i,i-1)=1/h^2*(1-(y(i+1)-y(i-1))/2/y(i));
    end
    J1(n,n)=1;
end
```

The resulting approximation is very close to those obtained with the shooting method. We will consider the error.

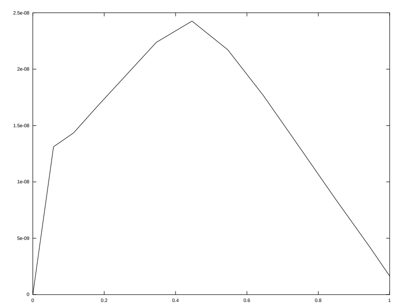

Figure 2: Error in shooting code. We note that the solution vector contains 13 points

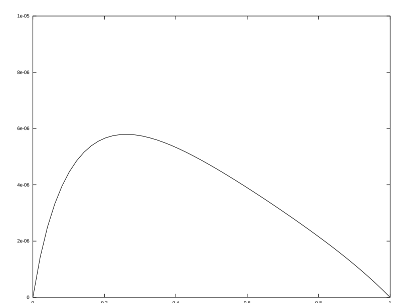

Figure 3: Error for finite difference code using 50 points.

We can see that we get much better results from the shooting code with much fewer points.

4 Difficult Boundary Value Problem

We will now consider a more difficult problem

$$y'' - y + y^2 = 0\,, \quad y'(0) = 0\,, y \to 0 \text{ as } x \to \infty\,, \quad y > 0\,. \tag{9}$$

This problem has the unique solution $y = \frac{3}{2}\text{sech}^2(x/2)$. Without the last restriction we also have the zero solution. So we must somehow deal with the non-uniqueness. The other problem we face is the boundary condition at infinity. The methods considered so far don't allow for a condition at ∞. If we try to enforce $y(R) = 0$ for some large R, we will find the solution is very sensitive to the choice or R and we will fail. Instead we note that if the solution decays, at $x = R \gg 1$, $y^2 \ll y$. So at R, the equation is approximately $y'' - y = 0$ and $y = Ce^{-x}$. So at $x = R$, we will impose the boundary condition

$$y'(R) = -y(R)$$

Here is the modified shooting method (secant version) code:

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

[x1,y1]=ode2r(@odes,[0,5],[t1,0]);
[x2,y2]=ode2r(@odes,[0,5],[t2,0]);
i=1;
m1=y1(end,1)+y1(end,2);
m2=y2(end,1)+y2(end,2);
while(abs(t2-t1)>0.000001)
    tmp=t2
    t2=t1-(t1-t2)/(m1-m2)*m1;
    t1=tmp;
    [x1,y1]=ode2r(@odes,[0,5],[t1,0]);
    [x2,y2]=ode2r(@odes,[0,5],[t2,0]);
    m1=y1(end,1)+y2(end,2);
    m2=y2(end,1)+y2(end,2);
    i=i+1;
end
i
x=x2;
y=y2;
end

function yp=odes(t,y)

    yp=zeros(2,1);

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

Notice the change to the initial conditions and the function $m(t)$. Now we set the initial value of y to t and set $y'(0) = 0$. We have defined $m(t) = f(5) - f'(5)$. The choice of $x = 5$ for our right endpoint is arbitrary, but if we make it too large, we will have stability problems and if we make it too small, our approximation $y'(x) = -y(x)$ will not be valid. If we run this code with bad initial choices of `t1` and `t2` we will find it converges to the wrong solution. Namely $y \equiv 0$ or $y \equiv 1$. We need a good approximation to $y(0)$ to get the right solution. If we multiply (9) by y' and perform the following manipulations,

$$
\begin{aligned}
y'y'' - yy' + y^2y' &= 0\,, \\
\frac{d}{dx}\left(\frac{1}{2}(y')^2 - \frac{1}{2}y^2 + \frac{1}{3}y^3\right) &= 0\,, \\
\frac{1}{2}(y')^2 - \frac{1}{2}y^2 + \frac{1}{3}y^3 &= C \text{ Some constant } C\,.
\end{aligned}
$$

Since $y(\infty) = y'(\infty) = 0$, the constant for the orbit we are interested in is 0. Now when $x = 0$, we have $y'(0) = 0$ so,

$$-\frac{1}{2}y(0)^2 + \frac{1}{3}y(0)^3 = 0$$

so $y(0) = 0$ or $y(0) = \frac{3}{2}$. For a non-zero solution, we use $y(0) = 1.5$ In the code we try `t1=1.4` and `t2=1.6` and we get a good approximation. It takes 113 iterations however. The finite difference code for solving (9) is given below:

```
function [x,y]=findiff2(n)

    x=linspace(0,5,n);
    y=zeros(n,1);
    for i=1:n
    y(i)=1.5/cosh(x(i)/2);
    end
```

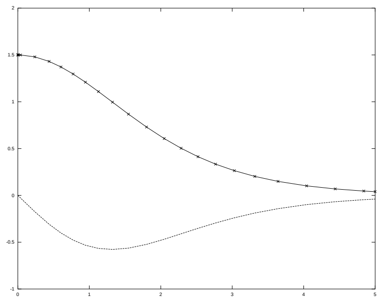

Figure 4: Output from the shooting code for (9). The solid lines represent the codes output. The crosses are the exact solution.

```
    dy=-J(y)\f(y);
    while(norm(dy,2)>0.001)
        y=y+dy;
        dy=-J(y)\f(y);
    end
end


function y1=f(y)
    n=length(y);
    h=5/n;
    y1=zeros(n,1);
    y1(1)=2*(y(2)-y(1))/h^2-y(1)+y(1)^2;
    for i=2:n-1
      y1(i)=(y(i-1)-2*y(i)+y(i+1))/h^2-y(i)+y(i)^2;
    end
    y1(n)=(2*y(n-1)-2*h*y(n)-2*y(n))/h^2-y(n)+y(n)^2;
end

function J1=J(y)
    n=length(y);
    h=5/n;
    J1=zeros(n,n);
    J1(1,1)=-2/h^2-1+2*y(1);
    J1(1,2)=2/h^2;
    for i=2:n-1
        J1(i,i+1)=1/h^2;
        J1(i,i)=-2/h^2-1+2*y(i);
        J1(i,i-1)=1/h^2;
    end
    J1(n,n-1)=2/h^2;
    J1(n,n)=-2/h-2/h^2-1+2*y(n);
end
```

The code does work, however I have given it the exact solution as a guess. So far it hasn't converged for any other guess. This may be a bug or it may be that this problem is very sensitive.