Addison Euhus
Nathan Feldman
Xiaoyue Pi
Zachary Rom
Guillaume Toujas

Numerical Solutions to Second-Order Ordinary Differential Equation Boundary Value Problems

Overview of Second-Order ODE BVPs

References

- http://www.cs.elte.hu/~faragois/ODE_angol.pdf
- http://www.math.usm.edu/lambers/mat461/spr10/lecture25.pdf
- http://mathworld.wolfram.com/BoundaryConditions.html

Our problem that we are trying to solve is a boundary value problem of a second order ordinary differential equation. The problem is presented as

$$\begin{aligned} y'' &= f(x, y, y') \\ a &< x < b \\ y(a) &= \alpha \\ y(b) &= \beta \end{aligned}$$

where α, β, a, and b are all given constants and $f(x, y, y')$ is the given function representing y''. There are several theorems that guarantee the problem presented above has a unique solution. For reference please see Theorem 5.0.3 on page 125 in the first link under references for this section. The conditions needed for a unique solution to exist include the following

- f, f_y, and $f_{y'}$ are continuous on the domain $x \in [a,b]$
- $f_y > 0$
- $f_{y'}$ is bounded on the domain

Note that there can be a variety of different boundary value conditions. The type of BVP mentioned above is given by Dirichlet Boundary Conditions, where the function values at the two endpoints of the interval of interest are specified. There are three other common boundary constraints that show up in a variety of different applications: the Neumann, Mixed, and Robin Boundary Conditions. For the Neumann condition, the values of the functions derivatives are specified at the endpoints. Mixed conditions consist of a combination of both the Dirichlet and Neumann conditions, where one endpoint's value is specified and the other's derivative value is specified. Robin conditions are similar to the mixed conditions, where a system of two equations relates the functional value and derivatives at each endpoint.

Real World Applications

References

- http://www.dmmm.uniroma1.it/~agostino.prastaro/Agarwal-conf.pdf

These boundary value problems, although not very common in the real world, do exist. For example, they can be used to model a heat transfer problem involving conducting solids and how the heat is diffused throughout the solid. There are other examples of using differential equations involving boundaries across biology and chemistry that scientists use every day.

The first example that we looked at where we could use boundary ODE problems to model a real world problem was by examining the diffusion of heat from a positive temperature source and through an electrically conducting solid. The equation $y'' = \lambda e^{\mu y}$, where $y(0) = y(1) = 0$ as the boundary conditions can be used to solve a few different kinds of heat diffusion problems. If $\mu = 1$, then the problem describes the loss of Joules in the conducting solid where λ is related to the current and e^y is related to the resistance based on the temperature. It could also be an equation that describes friction heating where λ is related to the constant shear stress and e^y is related to the fluidity, once again based on the temperature.

The second application we looked at had to do with the modeling of a circular membrane that has a normal uniform pressure applied to it. The equation is $y'' + \frac{k}{y^2} + \frac{3}{x}y' = 0$, where $0 < x < 1$ since at the edge we have the condition $y(1) = \lambda > 0$ and at the center, for symmetry, we have $y'(0) = 0$. In this equation, k is a positive constant, x is the radial coordinate for a point on the circular membrane, and $y(x)$ is the radial stress that the membrane is receiving. This problem can applied to any kind of real world situation where there is pressure being applied to a plate and you want to determine what the distorted plate will look like when that pressure is being applied.

Shooting Method: Core Idea

The shooting method is one way of solving boundary value problems. The core concept of the shooting method is to take the two-point boundary value problem and represent it as an initial value problem. During this conversion, we have to take an initial guess at the value y'(a) since it is not given in our problem and it is needed to solve the IVP. Therefore we pick some constant θ and set y'(a) = θ. We then solve the IVP using this guess to obtain y and then check the value of y(b). If y(b) == β then we are done and the solution of our IVP is the solution to the BVP. If y(b) != β then we can choose another guess for y'(a) and try again. This nature of taking a guess and then evaluating whether that guess provided a solution such that we have y(b) = β is where the shooting method got its name. We now begin an explanation of how to perform the shooting method in a broad perspective to solve a BVP.

Shooting Method: Broad Process

References

- https://www.youtube.com/watch?v=cq3bPBePE8E

The first step in the shooting method is to convert the second order ODE from the BVP into a system of first order ODEs. This procedure can be done using the method briefly discussed in class; however, for reference on this reduction you can visit the youtube video in the references section. We will assume this reduction makes sense, and now we present the reduction. Given our BVP in the form presented:

$$y'' = f(x, y, y')$$
$$y(a) = \alpha$$
$$y(b) = \beta$$

We can reduce it to a system of first order ODEs:

$$y' = u$$
$$u' = f(x, y, u)$$
$$y(a) = \alpha$$
$$y'(a) = \theta$$

where θ is the constant that we initially guess. Now that the problem is in IVP form, we can solve the system of first order ODEs using numerical methods. Examples of methods that can be used include explicit Euler and the collection of Runge-Kutta methods. These methods were discussed in class, and a quick reference on Wikipedia will explain their implementation. Let $y(x; \theta)$ denote the solution to the IVP using guess $y'(a) = \theta$. Now that we have a solution, we need to check whether it appropriately represents our end boundary condition. Let function $F(\theta)$ represent the difference between our solutions value at the end boundary point and the value at the end boundary point as defined in our BVP.

$$F(\theta) = y(b; \theta) - \beta$$

If F contains a root with the value θ then the solution $y(b; \theta)$ to our IVP problem is also a solution to the BVP problem. Using this knowledge, our next step is to root find. If we have linear ODEs we can use a linear interpolation and quickly solve the BVP. For non-linear ODEs we will need to use a more expansive root finding numerical method, such as the Secant method or Newton's method. We continue the discussion by going into more detail about solving the case of linear ODEs and non-linear ODEs

Linear ODEs

References

- http://www.math.usm.edu/lambers/mat461/spr10/lecture25.pdf
- http://mathfaculty.fullerton.edu/mathews//n2003/shootingmethod/ShootingProof.pdf
- http://en.wikipedia.org/wiki/Shooting_method

Here I present a summary of the information contained in the above references. An important proof about the shooting method for linear systems can be found at the second link.

The ODE is in linear form if $f(x, y, y') = p(x)y'(x) + q(x)y(x) + r(x)$. In this case we solve the problem using linear interpolation. We start by solving two IVPs where we guess y'(a) = 0 for one and y'(a) = 1 for the second. This leaves us with the following two systems of IVP problems

Guess $y'(a) = 0$

$$y' = u$$
$$u' = f(x, y, u)$$
$$y(a) = \alpha$$
$$y'(a) = 0$$

Guess $y'(a) = 1$

$$y' = u$$
$$u' = f(x, y, u)$$
$$y(a) = \alpha$$
$$y'(a) = 1$$

Let $y(x; 0)$ be the solution for the first IVP system and $y(x; 1)$ be the solution for the second IVP system. We use the fact that any linear combination of solutions of the ODE also satisfies the ODE to write

$$y(x) = y(x; 0) + \theta y(x; 1)$$

and then evaluate at x = b to write

$$y(b) = y(b; 0) + \theta y(b; 1) = \beta$$
$$\theta = \frac{\beta - y(b; 0)}{y(b; 1)}$$

Therefore as long as y(b;1) != 0, then y(x) as written as a linear combination represents the solution to our BVP.

$$y(x) = y(x; 0) + \frac{\beta - y(b; 0)}{y(b; 1)} y(x; 1)$$

Note that we could also calculate the solution to the IVP with y'(a) = θ and that solution would be the solution to the BVP problem as well.

Non-linear ODEs

References

- http://www.math.usm.edu/lambers/mat461/spr10/lecture25.pdf
- http://www.mathstat.dal.ca/~iron/math3210/bvp.pdf

In the case that $f(x, y, y')$ is nonlinear, then we have to use numerical root finding methods, such as Secant Method or the Newton Method, to find the root of

$$F(\theta) = y(b; \theta) - \beta$$

where $y(b; \theta)$ is the solution of the IVP problem with initial guess $y'(a) = \theta$. We iteratively solve the problem until the root of $F(\theta)$ converges. We now discuss solving using the Secant Method and Newton's Method

Non-linear ODEs: Secant Method

The secant method is

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

Therefore we need two guesses in order to begin solving the BVP. The process is as follows

- Choose a numerical scheme for solving systems of first order ODEs, such as Euler or Runge-Kutta
- Convert the BVP into IVP form as a system of first order ODEs
- Choose two guesses θ_1 and θ_2
- Solve an IVP$_1$ where $y'(a) = \theta_1$ and IVP$_2$ where $y'(a) = \theta_2$
- Compute $F(\theta_1)$ and $F(\theta_2)$.
- While (we have not converged up to the tolerance we want)
    - Use the secant method to guess the next θ_1 and θ_2
    - Solve the IVP using these two new guesses.
    - Compute $F(\theta_1)$ and $F(\theta_2)$
    - Check for convergence of the root

Non-linear ODEs: Newton's Method

References

- http://www.math.usm.edu/lambers/mat461/spr10/lecture25.pdf
- http://www.mathstat.dal.ca/~iron/math3210/bvp.pdf

Newton's Method is

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

Therefore in order to use it to approximate the root of $F(\theta) = y(b;\theta) - \beta$ we must also know $F'(\theta)$.

$$F'(\theta) = \frac{dy(x;\theta)}{d\theta} \quad evaluated\ at\ x = b$$

We now consider ODE given in the problem and by the chain rule write

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

$$\frac{dy''}{d\theta} = \frac{df}{dy}\frac{dy}{d\theta} + \frac{df}{dy'}\frac{dy'}{d\theta}$$

Next we let

$$z(x,\theta) = \frac{dy(x,\theta)}{d\theta}$$

and notice that it satisfies the ODE

$$z'' = \frac{df}{dy} z + \frac{df}{dy'} z'$$
$$z(a,\theta) = 0$$
$$z'(a,\theta) = 1$$

which can be obtained by differentiating the original BVP with respect to θ. This leads to $F'(\theta) = z(b)$ Therefore each iteration requires two IVPs to be solved, but we gain by using the rapid convergence of Newton's Method. Lets rewrite both IVPS, one representing our BVP problem and the other representing Newton's Method, as systems of first order ODEs

Conversion

$$y_1 = y, \qquad y_2 = y', \qquad z_1 = z, \qquad z_2 = z'$$

First IVP

$$\begin{aligned} y_1{}' &= y_2 \\ y_2{}' &= f(x, y_1, y_2) \\ y_1(a) &= \alpha \\ y_2(a) &= \theta \end{aligned}$$

Second IVP

$$\begin{aligned} z_1' &= z_2 \\ z_2' &= f_y(x, y_1, y_2)z_1 + f_{y'}(x, y_1, y_2)z_2 \\ z_1(a) &= 0 \\ z_2(a) &= 1 \end{aligned}$$

The general procedure is as follows

- Choose a numerical scheme for solving systems of first order ODEs, such as Euler or Runge-Kutta
- Convert BVP into an IVP_1 problem and then convert the BVP representing Newton's method to the IVP_2. See the above conversions
- Until convergence do
  - Solve both IVP_1 and IVP_2
  - Use Newton's Method to approximate the next guess for θ where $F'(\theta) = z(b)$

Properties of the Single Shooting Method

References

- http://www.mech.utah.edu/~pardyjak/me6700/Lect15_BoundEigenvalueProblemsCh27.pdf
- https://www.tu-ilmenau.de/fileadmin/media/simulation/Lehre/div/Lec_Slides3.pdf
- http://en.wikipedia.org/wiki/Direct_multiple_shooting_method
- https://www.tu-ilmenau.de/fileadmin/media/simulation/Lehre/div/Lec_Slides3.pdf

In the above sections, we discussed one variant of the shooting method, the single shooting method. Below are the advantages and disadvantages.

- Advantages
  - Solves both linear and non-linear BVPs
  - Simple implementation
  - Effect when the interval [a,b] is short since its "easier" to shoot a target a short distance away

- Disadvantages
    - Unstable for some problems. Typically those of highly nonlinear or unstable ODEs.
    - Requires a good initial guess for the IVP as the numerical method does not guarantee convergence.
    - The length of the interval of the boundary values has a high impact on the results. A large interval requires a large number of iterations.

We will not discuss this in this paper, but a solution to the issue of the length of the interval is to use a multiple shooting method. In general, this method can be seen as similar to chopping the interval into pieces and then applying multiple shooting methods to each piece and then piecing the solutions together. References for this method can easily be found online, but a good starting point can be found at last link under the references for this section.

MATLAB Examples

References

- http://www.mathstat.dal.ca/~iron/math3210/bvp.pdf

We will go over the problem discussed in the above reference as it clearly illustrates both an interesting non-linear problem and good MATLAB implementation. Since the implementation is clear, I highly suggest you to go through the information on the link.

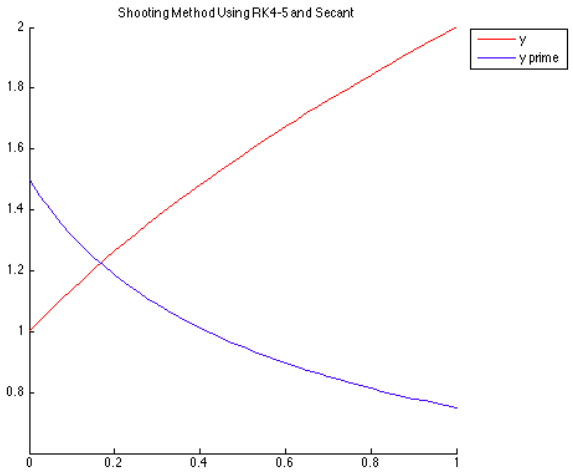
BVP Problem

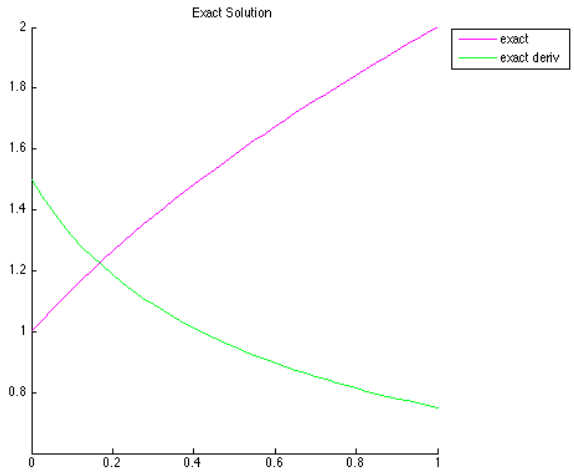
$$y'' = -\frac{(y')^2}{y}, \quad y(0) = 1, \quad y(1) = 2.$$

Exact Solution

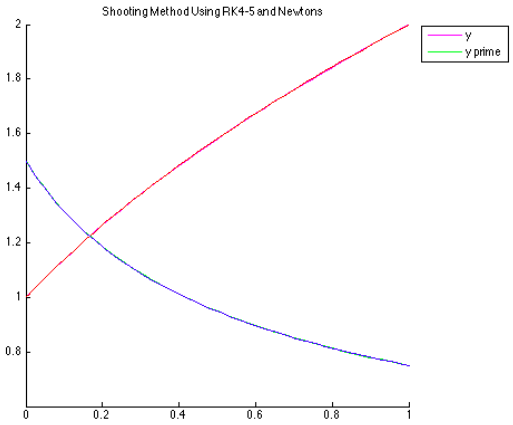
$$y = \sqrt{3x + 1}.$$

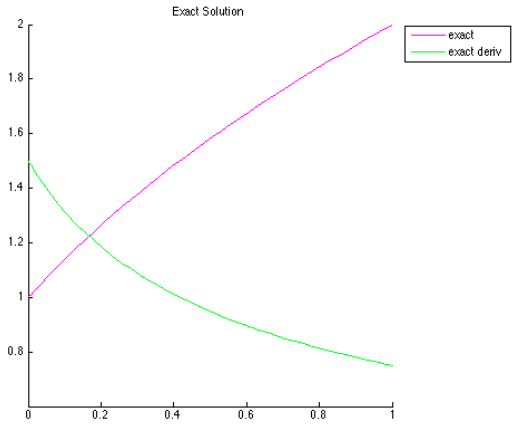
Shooting Using RK4-5 and Secant





Shooting Using RK4-5 and Newton's





As you can see the shooting method performs very well and closely matches the exact solution to this nonlinear ODE.

Code

Note: Much of the code can be credited towards the reference link for the problem. However, I will briefly explain it so that those without a strong MATLAB background can understand what is going on.

First we have to take the ODE and reduce it to the system of first order ODEs. We store these systems in a MATLAB function file so that we can pass them in as function handles to ode45, which is MATLAB's RK4-5 implementation. These functions can be found at the end of the code, and they are called ode_system and

ode_system2. Why are there two systems for this problem? Well its because one implementation uses the Secant Method and the other uses the Newton's Method. The Secant Method just requires the reduction of the BVP problem to a system of two first order ODEs. The Newton's method requires both the reduction of the BVP to a system of first order ODEs in addition to one more set of first order ODE equations due to the additional IVP problem that arises out of the necessity to perform the method. For more reference see the section above on Newton's method. Next we implement the shooting method through iteration. The necessary given information that we need in order to perform that method are a,b, alpha, and beta. For Secant Method, we need two guesses of the slope at point a. For Newton's method we only need one guess of the slope at point a. Then through iteration we continue to try and obtain a better solution until we have hit convergence.

```
function [ yp ] = ode_system( t,y )
    yp = zeros(2,1);
    yp(1) = y(2);
    yp(2) = -y(2)^2/y(1);
end

function [ yp ] = ode_system2( 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
```

```
function [ x, y ] = nonlinear_shooting( a, b, alpha, beta, ...
                                        theta1, theta2, ...
                                        method )
%UNTITLED2 Summary of this function goes here
%   Detailed explanation goes here

    if strcmp(method,'secant')
        % First we solve two IVP problems using two guesses at
        % the value of theta
        [x1,y1] = ode45(@ode_sys,[a,alpha],[b,theta1]);
        [x2,y2] = ode45(@ode_sys,[a,alpha],[b,theta2]);
        i = 1;
        F_theta1 = y1(end,1)-beta;
        F_theta2 = y2(end,1)-beta;
        while (abs(theta2-theta1)>eps)
            tmp = theta2;
            theta2=theta1-(theta1-theta2)/(F_theta1-F_theta2)*F_theta1;
            theta1 = tmp;
            [x1,y1]=ode45(@ode_system,[a,alpha],[b,theta1]);
            [x2,y2]=ode45(@ode_system,[a,alpha],[b,theta2]);
            F_theta1=y1(end,1)-beta;
            F_theta2=y2(end,1)-beta;
            i=i+1;
        end
        x = x2;
        y = y2;
    elseif strcmp(method,'newtons')
        [x1,y1] = ode45(@ode_system2,[a,alpha],[b,theta1,0,1]);
        i = 1;
        F_theta1 = y1(end,1)-beta;
        theta2 = theta1 - F_theta1/y1(end,3);
        while(abs(theta2-theta1)>0.000001)
            theta1 = theta2;
            [x1,y1] = ode45(@ode_system2,[a,alpha],[b,theta1,0,1]);
            F_theta1 = y1(end,1)-beta;
            theta2 = theta1 - F_theta1/y1(end,3);
            i = i+1;
        end
        x = x1;
        y = y1(:,1:2);
    end
end
```

```
clear
clc

a = 0;
b = 1;
alpha = 1;
beta = 2;
theta1 = 1; %guess
theta2 = 2; %guess

[x,y] = nonlinear_shooting(a,b,alpha,beta,theta1,theta2,'secant');

exact = sqrt(3*x+1);
exact_deriv = (3/2).*(1./exact);
hold on
title('Shooting Method Using RK4-5 and Secant')
plot(x,y(:,1),'r')
plot(x,y(:,2),'b')
legend('y','y prime', 'Location', 'northeastoutside');
hold off
figure
hold on
title('Exact Solution')
plot(x, exact, 'm')
plot(x, exact_deriv, 'g')
legend( 'exact','exact deriv', 'Location', 'northeastoutside');
hold off
```

```
clear
clc

a = 0;
b = 1;
alpha = 1;
beta = 2;
theta1 = 1; %guess
theta2 = 2; %guess
[x,y] = nonlinear_shooting(a,b,alpha,beta,theta1,theta2,'newtons');
exact = sqrt(3*x+1);
exact_deriv = (3/2).*(1./exact);
hold on
title('Shooting Method Using RK4-5 and Newtons')
plot(x,y(:,1),'r')
plot(x,y(:,2),'b')
legend('y','y prime', 'Location', 'northeastoutside');
hold off
figure
hold on
title('Exact Solution')
plot(x, exact, 'm')
plot(x, exact_deriv, 'g')
legend( 'exact','exact deriv', 'Location', 'northeastoutside');
hold off
```

Overview of the Finite Difference Method

References

- http://web.stanford.edu/~fringer/teaching/numerical_methods_02/handouts/lecture9.pdf
- http://www.ann.jussieu.fr/frey/cours/UdC/ma691/ma691_ch6.pdf

The finite difference method is one of the most intuitive and straightforward approaches for solving ODE BVPs. The basic idea behind it is to replace each derivative with their respective divided difference approximation. The most commonly used ones for the first derivative are the forward, backward, and central difference approximations shown below, all derived from algebraically relating the differences between the Taylor series of $u(x+h)$, $u(x)$ and $u(x-h)$. It is important to note that the central difference approximation has an error proportional to h^2 (order two) while the forwards and backwards have an error of order one. Furthermore, in the case of solving second-order ODEs, the consistent second-order approximation for u" also has an error of order two

Name	Value	Approximation	Error
Forward Difference	u’	$(u_{n+1} - u_n)/h$	O(h)
Backward Difference	u’	$(u_n - u_{n-1})/h$	O(h)
Central Difference	u’	$(u_{n+1} - u_{n-1})/(2h)$	$O(h^2)$
Consistent Second-Order	u’’	$(u_{n+1} - 2u_n + u_{n-1})/h^2$	$O(h^2)$

Procedure for Finite Difference Method

There are four main steps in order to solve a boundary value problem in one dimension using the finite difference method: discretize x, discretize the ODE, discretize the boundary conditions, and finally construct and solve the corresponding linear system.

1. **Discretize x**

For an ODE with Dirichlet conditions, this step is fairly simple. The x-values will be equally spaced throughout the interval starting at the first endpoint and ending at the other endpoint, where the amount of points is specified for the given approximation (generally, more points will yield more accurate results). With Neumann and Mixed conditions, this type of discretization of the independent variables will not work. This is because the derivative will need to be evaluated for

at least one of the endpoints, and the only way to guarantee that this will happen with second-order accuracy and result in a tridiagonal system later on is by having a value before and after the endpoints. By doing so, the derivative can be evaluated at x of ½ or x of N – ½ using the central difference scheme, which has order of degree two and therefore preserves the overall order. If this is confusing, it will make sense during the third step where the boundary conditions are discretized. In any case, with Dirichlet conditions we can use evenly spaced intervals starting and ending at the endpoints, whereas for Neumann, Mixed, and Robin conditions we will have to stagger the nodes to include one right before the first endpoint and one right after the second endpoint.

2. Discretize the ODE

In order to discretize the ODE, we will write out the equation and replace all of the derivatives with the schemes corresponding to second-order in order to preserve the accuracy of the method (using the central divided differences). Additionally, we will write all of the values of x in terms of x_i in order to properly discretize them. From there, we define the four values at each step – a_i, b_i, c_i, and d_i. The a values correspond to the coefficient of the u_{i-1} term, the b values correspond to the coefficient of the u_i term, the c values correspond to u_{i+1} and the d values correspond to the remaining values, where d_i is a function of x (although the d values can also be zero). An example is converting $u'' - 2u = x^2$ to $u_{n+1}/h^2 - (2 - 2/h^2)u_n + u_{n-1}/h^2 = x_i^2$. In this case, a and c will take on the values of $1/h^2$, b will take on the value $2 - 2/h^2$ and the d values will correspond to x_i^2.

3. Discretize and Embed the Boundary Conditions

In order to embed the boundary conditions into the method, we will adjust the b and d coefficients at the endpoints. If we are working under Dirichlet conditions, then the b values will not change, and we need to only subtract $\frac{u(a)}{h^2}$ from d_1 and $\frac{u(b)}{h^2}$ from d_N. However, if we have other boundary conditions, we will be working with a staggered interval around the endpoints.

If we are given the value of the function at the first endpoint, we use the centered interpolation (with second-order accuracy) to relate the first and second values. This will accordingly adjust both the value of b_1 and d_1. Analytically, we relate it as $u_{½} \cong (u_0+u_1)/2$, where $u_{½} = u(a)$

On the other hand, if we are given the value of a derivative at the second endpoint, for instance, then we can use the central difference scheme to relate the second to last and last values, again changing the values of b_{N-1} and d_{N-1}. This is done using the relationship $u'_{N-½} \cong (u_N - u_{N-1})/h$ given $u'_{N-½} = u'(b)$.

Of course, these could be switched depending on the specific boundary conditions.

4. Construct the Linear System and Solve

The last step is to solve for vector u, which will represent the discrete approximations of the solution to the BVP. This is done by creating a tridiagonal matrix A, where the diagonal elements are the values of b, with the values of a and c on the side. An example is pictured below (note that in the picture, the first value is b_2, whereas we begin at b_1 due to a discrepancy in indexing. Additionally, they use y instead of u to represent their approximations. In either case, it is a matter of personal preference).

$$\begin{pmatrix} b_2 & c_2 & & & & & \\ a_3 & b_3 & c_3 & & & & \\ & a_4 & b_4 & c_4 & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & a_{N-3} & b_{N-3} & c_{N-3} & \\ & & & & a_{N-2} & b_{N-2} & c_{N-2} \\ & & & & & a_{N-1} & b_{N-1} \end{pmatrix} \begin{pmatrix} y_2 \\ y_3 \\ y_4 \\ \vdots \\ y_{N-3} \\ y_{N-2} \\ y_{N-1} \end{pmatrix} = \begin{pmatrix} d_2 \\ d_3 \\ d_4 \\ \vdots \\ d_{N-3} \\ d_{N-2} \\ d_{N-1} \end{pmatrix}$$

This is very intuitive when the formulas are written out. Essentially, you take the discretized equation and create a large system relating the approximation values of u to what they equal from our discretized equation. It works very well because of the fact that this specific method results in a tridiagonal matrix A (with coefficients of a, b and c) and a corresponding vector d that the systems equal. From here, we can solve the corresponding system for u_1 to u_{N-1} using a variety of different methods we have learned in class. If we had Dirichlet conditions, then we already know the values of u_0 and u_N; on the other hand, with other conditions we have already related u_0 to u_1 and u_N to u_{N-1}, and we can solve for these once we have the values for u_1 and u_{N-1}.

Characteristics of the Finite Difference Method

The finite difference method is a very broad method that is very easy to implement for solving differential equations. This is its main strength – the problem intuitively makes sense, and results in computations that are not hard to understand, such as solving a tridiagonal linear system. However, it has its shortfalls. Like the shooting method, it does not work very well on large intervals. Also, accuracy is lost when solving large linear systems. This reduces the effectiveness of increasing the amount of nodes used to approximate the solution.

The finite difference method is such a broad technique that a variety of its characteristics depend on the specific scenario. Different equations and boundary conditions will yield different results when speaking about things such as convergence and error. There are a large variety of different possibilities that

require careful analysis in order to measure the strength of the method that are too in-depth to cover in such a short presentation. However, we can analyze a simple equation $u"(x) + c(x)u(x) = f(x)$ to get an idea of basic properties. We can see that if c > x for all values of x on the interval, then the corresponding matrix A (the one consisting of all of the a, b, and c coefficients) will be symmetric and positive definite, therefore A is invertible and a solution exists. It can also be shown that if the true solution has continuous derivatives of up to the fourth power, then the numerical scheme that we have provided is consistent with second-order accuracy. As you can see, these interpretations can easily change for various reasons. One thing that can change all of these conclusions is the introduction of a non-linear term. It is for this reason that the specific BVP must be analyzed to make more detailed conclusions.

Finite Difference Example

This next section will show an example of solving the second-order ODE BVP:

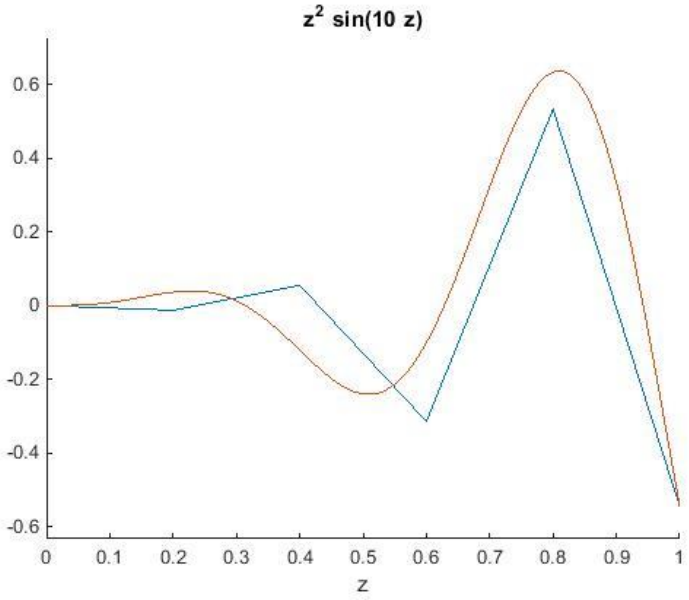
$$u"(x) + 100u(x) = 40\cos(10x) + 2\sin(10x)$$
$$u(0) = 0, u(1) = \sin(10)$$

This has exact solution

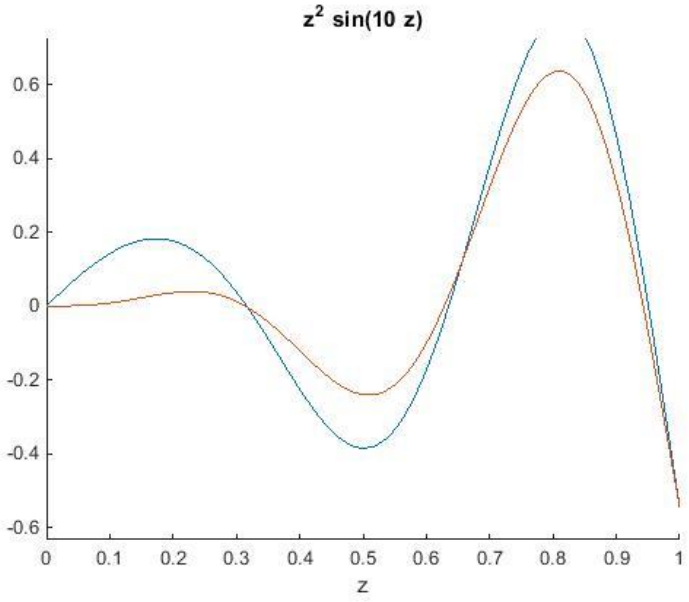
$$u(x) = x^2\sin(10x)$$

After the following graphs, we have included our MATLAB code used to construct the approximation. There are two pieces of the code: the first one solves a second-order ODE BVP similar to the one highlighted immediately before this section using Dirichlet conditions, while the second one uses mixed conditions given the function value at the left endpoint and the derivative value at the right endpoint. In either case, both converged very nice to the exact solution.

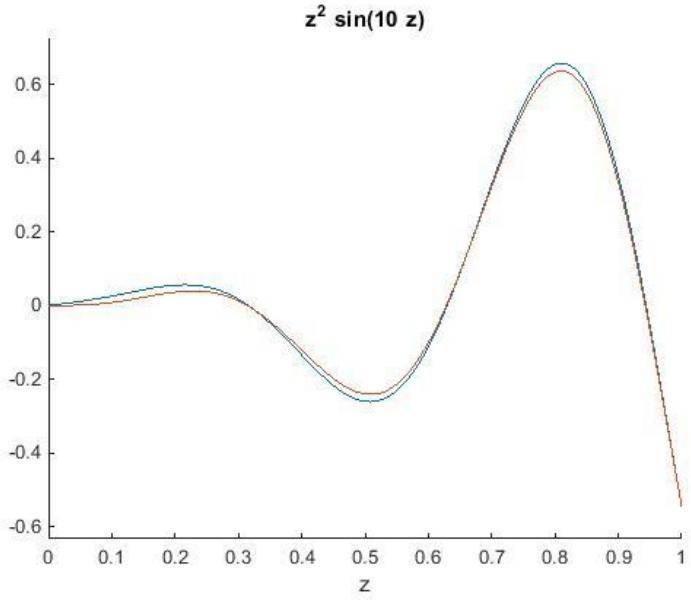
For N = 5:



For N = 50:



For N = 500:



Code (Following Two Pages)

```
%****************Finite Difference Dirichlet
clearvars
format long
N = 500;

%y" + g(x)y = f(x)
syms x
g = symfun(100, x);
f = symfun(40*x*cos(10*x) + 2*sin(10*x), x);
%[x0,x1] with u(x0) = u0, u(x1) = x1
x0 = 0;
x1 = 1;
u0 = 0;
u1 = sin(10);

%Change in X
x = linspace(x0,x1,N+1);
h = 1/(N-1);

%Creating Coefficients
a = zeros(N-1,1);
b = zeros(N-1,1);
c = zeros(N-1,1);
d = zeros(N-1,1);

for i=1:N-1
    a(i) = 1/h^2;
    b(i) = g(x(i)) - 2/h^2;
    c(i) = 1/h^2;
    d(i) = f(x(i));
end

%Adjustments
d(1) = d(1) - u0/h^2;
d(N-1) = d(N-1) - u1/h^2;

%Coefficient Matrix
A = zeros(N-1,N-1);
for i=2:N-2
    A(i,i-1) = a(i);
    A(i,i) = b(i);
    A(i,i+1) = c(i);
end
A(1,1) = b(1);
A(1,2) = c(1);
A(N-1,N-2) = a(N-1);
A(N-1,N-1) = b(N-1);

y = A\d;
u(2:N) = y(1:N-1);
u(1) = u0;
u(N+1) = u1;

%Exact Solution
syms z
j(z) = z^2*sin(10*z);

hold all
plot(x,u)
ezplot(j,[x0,x1])
```

```
%***************Finite Difference Mixed
clearvars
format long
N = 50;

%y" + g(x)y = f(x)
syms x
g = symfun(100, x);
f = symfun(40*x*cos(10*x) + 2*sin(10*x), x);
%[x0,x1] with u(x0) = u0, u'(x1) = x1
x0 = 0;
x1 = 1;
u0 = 0;
u1 = 10*cos(10)+2*sin(10);

%Change in X
h = 1/(N-2);
x = linspace(x0-h/2,x1+h/2,N+1);

%Creating Coefficients
a = zeros(N-1,1);
b = zeros(N-1,1);
c = zeros(N-1,1);
d = zeros(N-1,1);

for i=1:N-1
    a(i) = 1/h^2;
    b(i) = g(x(i)) - 2/h^2;
    c(i) = 1/h^2;
    d(i) = f(x(i));
end

%Adjustments
d(1) = d(1) - 2*u0/h^2;
d(N-1) = d(N-1) - u1/h;
b(1) = b(1) - 1/h^2;
b(N-1) = b(N-1) + 1/h^2;
%Coefficient Matrix
A = zeros(N-1,N-1);
for i=2:N-2
    A(i,i-1) = a(i);
    A(i,i) = b(i);
    A(i,i+1) = c(i);
end
A(1,1) = b(1);
A(1,2) = c(1);
A(N-1,N-2) = a(N-1);
A(N-1,N-1) = b(N-1);

y = A\d;
u(2:N) = y(1:N-1);
u(1) = 2*u0 - u(2);
u(N+1) = h*u1 + u(N);

plot(x,u)
```

Overview of the Finite Element Method

References

- http://www.researchgate.net/post/Why_cant_an_Initial_value_Problem_IVP_be_solvable_using_the_Finite_Element_method
- http://en.wikipedia.org/wiki/Self-adjoint_operator#Self-adjoint_operators

Finite Element Method (F.E. Method) can be used to solve a BVP by using the Galerkin approximation as we talked about in class. In addition, this Galerkin method can also be used to solve an IVP. We can achieve it by either using weak form combined with stabilizing techniques or using least-squares process.

For a F.E. method to be used successfully for a specific differential equation, the resulting coefficient matrix in [K]{u}={F} must be positive-definite (This means that [K] can be inverted and the obtained solution {u} will be unique).

The following three theorems discuss the positive-definite constraint:

1. When the differential operator is self-adjoint (an operator A is self-adjoint if and only if A = A*) the Galerkin method with weak form and least squares process are the only F.E. methods that guarantee positive-definite coefficient matrix [K].
2. When the differential operator is non-self adjoint or non-linear: least square process is the only F.E. method that guarantees a positive-definite coefficient matrix [K].
3. When considering an IVP, the differential operator is either non-self adjoint or nonlinear. It is never self-adjoint. Hence, the only F.E. method that can guarantee positive definite coefficient matrices for all IVPs is least squares process.

There is a lot of research being done to make the Galerkin method with weak form produce coefficient matrices that are positive-definite. These are known as stabilizing techniques. However, all of these techniques, without exception, end up changing the original IVP.

Practical Uses of the Finite Element Method

When we searched for the documents online, we found that many people are under the impression that Galerkin method with weak form is the only F.E. Method. This happens because almost all commercial F.E. software is based on this method. The reason for this is because the differential operators of differential equations which describe elastic solid mechanics are always self-adjoint; hence Galerkin method with weak form works very well. Least-squares processes also work well, however, least-squares processes require additional resources (in terms of interpolation theory) that didn't exist at the time when commercial F.E. software started to appear (in the 1960s). Hence, people didn't consider least-squares process. Around the late 1990s,

those additional resources needed for least-squares were invented, and least-squares started being used more in the research area. But since so many companies were built on Galerkin method with weak form and are only marketed towards solid mechanics applications, they still continue to use it.

Differences between the Finite Differences and Finite Element Methods
References

- http://www.researchgate.net/post/What_is_the_difference_between_FEM_and_FDM2

The Finite Element Method (FEM) and the Finite Difference Method (FDM) are two different ways of solving second-order ordinary differential equation boundary value problems. We have already discussed the Finite Difference Method in this paper and in class, but it is also interesting to see how it differs from the Finite Element Method.

We know that the Finite Difference Method uses difference formulation such as backwards and forwards methods on the current step to estimate the next step. On the other hand, the Finite Element Method divides the time steps into small sections and individually calculates those sections before putting them back together using the theory of superposition.

When examining the differences between the two methods, we can understand how the FEM method is more computationally expensive than the FDM method. FEM also tends to give more refined results, but in some cases FDM is actually preferred for accuracy. For example, in dynamic problems that depend on time the FDM will actually be more accurate. In general though, the results from using both methods will be relatively similar and choosing the method depends on the type of problem you are facing as well as your computational limitations.

Overview of the Quadrature Method
References

- http://www2.math.umd.edu/~dlevy/classes/amsc466/lecture-notes/integration-chap.pdf
- http://www.math.uakron.edu/~kreider/num1/quadrature.pdf
- http://people.cs.uchicago.edu/~lebovitz/Eodesbook/intro.pdf

The goal of the quadrature method for second-order ODE boundary value problems is to be able to approximate the solution to the differential equation. How is this done? First, assume that we are given an equation of the following style for a function:

$$y'' = f(x, y, y'); f(x_0) = y_0; f(x_M) = y_M;\ x_0 < x_M$$

This is done first by using a numerical method of choice to approximate the value of $y'(x_i)$ at strategically chosen x_i in $[x_0, x_M]$. For instance, the shooting method can be used to find an approximation of $y'(x_0)$, and some other method could be potentially used to approximate $y'(x_i)$ at other x_i. Now, given these approximations of y', a method of numerical integration can be chosen to approximate $y(x_j)$ for some suitable x_j in $[x_0, x_M]$.

Basic Example

As a very rudimentary example, let us imagine a general equation:

$$y'' = f(x, y, y');\ \mathrm{y}(0) = \mathrm{e},\ \mathrm{y}(1) = 0$$

To begin, say we first use a chosen method of approximation to find $\mathrm{y}'(0) = 4$, and from there, we use a chosen method of approximation to find $\mathrm{y}'\left(\frac{1}{4}\right) = .5$. Then, we can use the midpoint rule to approximate $\mathrm{y}\left(\frac{1}{2}\right) = y(0) + \int_{1/2}^{0} y'(x) = \frac{1}{2} y'\left(\frac{1}{4}\right) + y(0) = e + .25$. This isn't a fantastic means of approximating $\mathrm{y}\left(\frac{1}{2}\right)$ as the midpoint rule is rather inaccurate in most circumstances, but it illustrates the method. We can use this technique to approximate any values of y in (0,1).

Advantages and Disadvantages

A pro of this method is that it allows us to use any method of numerical integration- For instance, if I want to use Gaussian 2-point quadrature rules, I can use that instead of the midpoint method (which is obviously a vast improvement). In addition, the method allows us to approximate the solutions to difficult differential equations, where the actual value of the integral may be difficult to find.

However, this method has its drawbacks as well. The different values of x_i we find limit the method of numerical integration that we can use. For instance, if we have a differential equation over (0,1) and y'(0.00001), then using the midpoint method only approximates y(0.00002). This means it becomes more difficult to get accuracy near the endpoints of the interval without extremely small step sizes, which take large amounts of computational time. Furthermore, taking large step sizes isn't easy either. If we want to use our midpoint method again, we are stuck only using values of y'(x < .5) as otherwise the midpoint method would be approximating values of x outside of [0,1].

Conclusion

As you can see, there are a variety of different methods that can be used to determine numerical solutions to second-order ODE BVPs. Each method has strengths and weaknesses, and the selection of the method is greatly influenced by the specific problem that is being analyzed.

As we have seen throughout the presentation, many established numerical procedures – such as solutions to matrix equations and numerical integration techniques – are incorporated into the analysis and computation of the approximations to these ODEs.

The mathematical field concerned with discovering computational methods for all different types of differential equations is very vast. Although we briefly touched on them, there is still plenty more in-depth analysis of these methods' existence, rate of convergence, accuracy and stability.