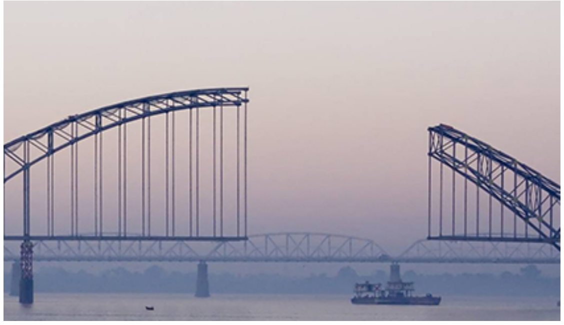
Boundary Value Problems

MATH2601: Control of Partial Differential Equations

Location: https://people.sc.fsu.edu/∼jburkardt/classes/control_2019/bvp/bvp.pdf



Fill in the middle to connect the edges.

Much like an initial value problem, a boundary value problem (BVP) over the interval $[a, b]$ uses an ordinary differential equation (ODE) to describe the shape of a curve $y(x)$ over that interval. However, the side conditions that accompany the ODE are now split, with one imposed at $x = a$ and one at $x = b$. While a typical second order IVP will specify the solution and derivative at the initial point a, a corresponding BVP may specify the solution at both ends, the solution at one end and the derivative at the other, or ever various algebraic combinations of solution and derivative.

Computational Solution of Boundary Value Problems

Given a second order linear BVP: $y'' = f(t, y, y')$ over $[a, b]$, with one condition at $x = a$ and another at $x = b$, produce an estimate for y(x).

Because the conditions are separated in space, the straighforward explicit marching method used for an IVP is not immediately applicable. Two simple techniques for dealing with a BVP are:

- the shooting method, which adopts a modified, iterative IVP approach;
- the finite difference method, which uses a mesh to create a linear system;

1 Rewriting a second order system

The typical boundary value problem involves a second order differential equation. If we plan to use an initial value problem solver, we need to rewrite this as a pair of first order equations. This may be done using the transformation $y \to u, y' \to v$, so that we have:

$$\begin{aligned} u' &= v \\ v' &= f(x, u, v) \end{aligned}$$

which now becomes a pair of coupled first order ODE's.

Thus, if our original problem is

$$x^2 y\text{''} + xy' + (x^2 - 0.4)y = \cos(x)$$

our transformed problem becomes:

$$u' = v$$

$$v' = \cos(x) - \frac{xv + (x^2 - 0.4)u}{x^2}$$

and the corresponding MATLAB function file would be

```
function uvprime = rhs ( x, uv )
  nu = 0.4
  uvprime(1) = uv(2);
  uvprime(2) = cos(x) - ( x .* uv(2) + (x.^2 - nu ) .* uv(1) ) ./ x^2;
  return
end
```

2 The shooting method

Suppose we need to solve a BVP, but we have software to solve an IVP. Let us suppose that our BVP specifies the solution values ya and yb at the endpoints. Then the reason we can't use the IVP software is that we are missing the prescribed starting value $y'(a)$. Suppose that we just guess a value α for $y'(a)$ and solve the resulting problem, which will result, among other things, in computing a corresponding value at the right endpoint. Since this result depended on our choice α, we might denote it by $yb(\alpha)$. Our computed answer and the prescribed value yb will differ. Consider the function $f(\alpha) = yb(\alpha) - yb$. We can use zero-finding software (such as the secant method) to search for values of α that drive $f(\alpha)$ towards zero. This will give us an approximate solution to our discretized BVP.

We have an initial condition for u, namely $u = ya$. Assuming there is a solution to the BVP, then there is a corresponding value for v at $x = a$; we don't know what it is, but let's call it α^*.

If we simply set $v(a)$ to a value α, then we can use an initial value solver to compute the solution all the way to $x = b$,. We note the value of v at this point using the notation $vb\alpha$. The correct solution will have the value yb, and so our discrepancy is $f(\alpha) = yb - vb(\alpha)$.

But now we can use a zero-finding technique to try to find a value of α for which $f(\alpha)$ is zero. But that would be a solution to the BVP. Obviously, this process will involve iteration as the zero finder seeks a solution, and it is possible that there is no solution, or that our zero finder is unable to find one to acceptable accuracy.

3 Example: $y" = \frac{3}{2}y^2$

Here is a MATLAB script to implement the shooting method for the equation $y" = \frac{3}{2}y^2$, over the interval $[0, 1]$, with conditions $y(0) = 4, y(1) = 1$. A secant method will be used, so the first two steps set α to 0, and then -1, before starting the iteration:

```
  a = 0.0;
  b = 1.0;

  ya = 4.0;
  yb = 1.0;

  fprintf ( 1, '\n' );
  fprintf ( 1, '  Step   Alpha   YB(Alpha)   F(ALPHA)\n' );
  fprintf ( 1, '\n' );

  for i = -1 : 10
```

```
    if ( i == -1 )
      alpha = 0.0;
    elseif ( i == 0 )
      beta = alpha;
      f_beta = f_alpha;
      alpha = -1.0;
    else
      gamma = alpha - f_alpha * ( beta - alpha ) / ( f_beta - f_alpha );
      beta = alpha;
      f_beta = f_alpha;
      alpha = gamma;
    end

    [ x, y ] = ode45 ( @myode, [a;b], [ya; alpha ] );
    yb_alpha = y(end,1);
    f_alpha = yb_alpha - yb;
    fprintf ( '  %d  %g  %g  %g\n', i, alpha, yb_alpha, f_alpha );

  end

  return
end
function dydx = myode ( x, y )
  dydx(1,:) = y(2,:);
  dydx(2,:) = 1.5 * y(1,:).^2;
  return
end
```

4 Exercise: Shooting for $y" + \frac{2}{x}y' + \frac{1}{x^4}y = 0$

Consider the BVP defined over the interval $[\frac{0.5}{\pi}, \frac{2.0}{\pi}]$, with ODE $y" + \frac{2}{x}y' + \frac{1}{x^4}y = 0$, with conditions $y(\frac{0.5}{\pi})$=0 and $y(\frac{2.0}{\pi}) = 1$.

EXERCISE:

1. Implement a shooting method approach to solve this problem.
2. Monitor the error in your nonlinear equation solver, and increase the number of steps appropriately. What number of steps did you need to drive the error very small?
3. Plot your final solution versus the function $y(x) = \sin(\frac{1}{x})$.

5 A finite difference approach

Suppose that the BVP is linear with respect to y, y' and $y"$. Choose a mesh of n points in $[a, b]$, equally spaced by $h = \frac{b-a}{n-1}$. At each interior point, write a discretized version of the ODE in which y' and $y"$ are replaced by centered differences:

$$y'(x_i) \approx \frac{y_{i+1} - y_{i-1}}{2 * h}$$
$$y"(x_i) \approx \frac{y_{i+1} - 2 * y_i + y_{i-1}}{h^2}$$

The first and last equations express the boundary conditions. If either condition involves the derivative y', then a forward difference or backward difference is used at $x = a$ or $x = b$ respectively. By evaluating right hand sides of the conditions and the ODE at each mesh point, we have the matrix of coefficients and the right hand side for a system of n linear equations, which then can be solved for the values y_i.

6 Example: $y'' = y + 4e^x$

```
a = 0.0;
b = 0.5;
ya = 1.0;
yb = 2.0 * exp ( 0.5 );

n = 11;
x = linspace ( a, b, n );
h = ( b - a ) / ( n - 1 );

A = zeros ( n, n );
rhs = zeros ( n, 1 );

A(1,1) = 1.0;
rhs(1) = ya;

for i = 2 : n - 1
  A(i,i-1) =  1.0 / h^2;
  A(i,i  ) = -2.0 / h^2 - 1.0;
  A(i,i+1) =  1.0 / h^2;
end
rhs(2:n-1) = 4.0 * exp ( x(2:n-1) );

A(n,n) = 1.0;
rhs(n) = yb;

y = A \ rhs;
```

7 Exercise: Finite Differences for $y” + \frac{2}{x}y' + \frac{1}{x^4}y = 0$

Consider the BVP defined over the interval $[\frac{0.5}{\pi}, \frac{2.0}{\pi}]$, with ODE $y” + \frac{2}{x}y' + \frac{1}{x^4}y = 0$, with conditions $y(\frac{0.5}{\pi})$=0 and $y(\frac{2.0}{\pi}) = 1$.

EXERCISE:

1. Implement the finite difference method approach to solve this problem.
2. If n is too small, your solution will not be accurate. By using plots, seek a value of n large enough that the solution “settles down”;
3. Plot your final solution versus the function $y(x) = \sin(\frac{1}{x})$.