

MP213

EIGENVALUES AND EIGENFUNCTIONS USING MAPLE

Consider the eigenvalue problem

$$\begin{aligned}y'' + \lambda y &= 0 \\2y(0) + y'(0) &= 0 \\y(1) &= 0\end{aligned}$$

We are seeking the values of λ for which it is possible to find a non-trivial solution for this problem, and the corresponding solutions.

If we attempt this analytically, we arrive at the equations

$$\begin{aligned}p &= 2 \tanh p \\ \omega &= 2 \tan \omega\end{aligned}$$

which we have to solve numerically in any case.

As an alternative, we can approach the problem numerically *ab initio*.

The procedure which we adopt is called a shooting method. We choose λ , and find a non-trivial solution which satisfies one of the boundary conditions. We then evaluate this solution at the second boundary point. Usually the second boundary condition is not satisfied, so that we have to keep choosing new values of λ until it is.

Provided that the original guess is not too bad, an approximately linear correction procedure can be used.

For example, in one such procedure where the second boundary condition was $y(1) = 0$, the following values appeared

$$\begin{aligned}\lambda = 1.5 & \quad y(1) = -.1648 \\ \lambda = 1.4 & \quad y(1) = -.0152\end{aligned}$$

Since a decrease of .1 in λ produced a decrease of .1496 in the error, a decrease of a further .01 should just about get it right. Indeed, at the next iteration the results were

$$\lambda = 1.39 \quad y(1) = .0007$$

This has overshoot by about 1/200 of the change arising from a decrease of .1, so we correct λ to 1.3905 for which $y(1)$ is now $-.00009$. The number of corrections you make depends on the accuracy that you are aiming to achieve.

At a more sophisticated level, it is possible to program the computer to make these choices and to perform the iterations until the required accuracy is attained.

For the example at the top of the page, a suitable choice of initial conditions is $y(0) = 1$, $y'(0) = -2$. (Any choice $y(0) = c$, $y'(0) = -2c$, $c \neq 0$, will do. Since the equation is linear, the multiple c merely scales the solution throughout by this same factor c . As you may remember from linear algebra, eigenvectors are only defined to within a non-zero multiple.)

The following sequence of maple commands can now be used to calculate the eigenvalues:

```
eq:= diff(y(x),x,x) + l*y(x) = 0;
```

This sets up the differential equation with a parameter **l** representing λ .

```
ini:= y(0) = 1, D(y)(0) = -2;
```

This specifies the initial conditions on the solution.

```
equ:= subs(l = *. ***,eq);
```

This substitutes whatever value ***. ***** you have chosen for λ into the original equation.

```
f:= dsolve({equ,ini},y(x),numeric);
```

This defines what is called a procedure on the system. The computer produces a program for the numerical solution of the differential equation **equ** with the initial conditions **ini**, but does not calculate any values at this stage. It returns a statement

```
f:= proc(rkf45_x) ... end
```

telling you this.

```
f(1);
```

This causes the computer to run the program **f** to calculate the solution when $x = 1$. The answer appears on the screen in the form

$$[x = 1, \quad y(x) = *. ** ** **, \quad \frac{d}{dx}y(x) = *. ** ** **]$$

If the calculated value of $y(1)$ is sufficiently close to 0, you can proceed to plot the eigenfunction. Otherwise, you press the up-cursor twice to retrieve the instruction

```
equ:= subs(l = *. ***,eq);
```

in which you amend your guess for **l**.

(Use the left and right cursor keys to position the **|**, the keyboard to enter values and the backspace key to delete.)

You now use the *ENTER* key to recall the previous instructions in sequence in order to repeat the calculations.

When you are satisfied with your estimate for the eigenvalue, you can draw the corresponding eigenfunction on the screen by issuing the commands

```
with(plots,odeplot);
```

```
odeplot(f,[x,y(x)],0..1);
```

The first instruction (which only needs to be used once in any session) loads the appropriate software.

It returns the prompt

```
[odeplot]
```

The second instruction plots the solution y of **f** as a function of x for $0 \leq x \leq 1$.

Because of the peculiar way in which **Maple** returns the numerical solution of a differential equation, the simple **plot** command cannot be used. An alternative approach to this problem will be considered later.

For the first three eigenvalues, approximate starting values are -4 , 20 and 60 . Using these, calculate the eigenvalues to three decimal place accuracy, and then plot the eigenfunctions.

Since we have the equations above for $p = \sqrt{-\lambda_1}$ and $\omega = \sqrt{\lambda_i}$ in terms of the eigenvalues, in this case you can check the accuracy of your calculations.

As a second exercise, you should calculate the first two eigenvalues and the corresponding eigenfunctions for the problem

$$\begin{aligned}y'' + \lambda x^2 y &= 0 \\y(0) &= 0 \\y(2) &= 0\end{aligned}$$

The equation can be entered as

eq:= diff(y(x),x,x) + 1*x*x*y(x) = 0;

and suitable initial conditions are

ini:= y(0) = 0, D(y)(0) = 1;

Starting approximations for λ are 2 and 9 .

In this case there is no analytic solution of the equation available to check the correctness of our numerical work.

However, we do know that the first eigenfunction has no zeros between 0 and 1 , while the second has one zero in this interval. Therefore, plotting the solutions provides a check that we have not overlooked an eigenvalue and eigenfunction.

Finally, consider the same differential equation

$$y'' + \lambda x^2 y = 0$$

but with the boundary conditions $y'(-1) = 0, y'(1) = 0$.

Suitable initial conditions are

ini:= y(-1) = 1, D(y)(-1) = 0;

and possible starting values for λ are 4 and 48 .

Are you happy with these results?

Approximating functions in terms of eigenfunctions

Let us consider in more detail the second problem treated; namely

$$\begin{aligned}y'' + \lambda x^2 y &= 0 \\y(0) &= 0 \\y(2) &= 0\end{aligned}$$

This equation arises in the solution by separation of variables of the partial differential equation

$$\frac{\partial^2 u}{\partial x^2} = x^2 \frac{\partial^2 u}{\partial t^2}$$

with the boundary conditions

$$u(0, t) = u(2, t) = 0 .$$

Suppose that we also have the initial conditions

$$u(x, 0) = f(x) , \quad u_t(x, 0) = 0 .$$

Then the solution has the form

$$u(x, t) = \sum_{i=1}^{\infty} a_i y_i(x) \cos(\sqrt{\lambda_i} t)$$

where the constants a_i are determined by $f(x)$.

If $y_i(x)$ is the eigenfunction associated with the eigenvalue λ_i , then

$$\begin{aligned}\lambda_i x^2 y_i(x) &= -y_i''(x) \\ \lambda_i \int_0^2 x^2 y_i(x) y_j(x) dx &= - \int_0^2 y_i''(x) y_j(x) dx \\ &= -y_i'(x) y_j(x) \Big|_0^2 \\ &\quad + \int_0^2 y_i'(x) y_j'(x) dx \\ &= \int_0^2 y_j'(x) y_i'(x) dx \\ &= \lambda_j \int_0^2 x^2 y_j(x) y_i(x) dx\end{aligned}$$

so that these eigenfunctions are orthogonal on the interval $[0, 2]$ with weight factor x^2 .

Given the function $f(x)$ defined on $[0, 2]$, (and preferably, but not necessarily, satisfying $f(0) = 0$, $f(2) = 0$) we can approximate it in terms of the eigenfunctions by

$$f(x) \simeq \sum_{i=1}^n a_i y_i(x)$$

where the coefficients a_i are given by

$$a_i = \int_0^2 x^2 f(x) y_i(x) dx / \int_0^2 x^2 y_i^2(x) dx$$

Since we do not have exact formulae for the eigenfunctions, these integrations need to be performed numerically.

For the purposes of this exercise we will divide the interval $[0, 2]$ into 100 equal steps and use the trapezoidal rule for integration.

Setting $h = 1/50$ and $x_r = r/50$, we have

$$\begin{aligned} \int_0^2 x^2 f(x) y_i(x) dx &\simeq \frac{h}{2} (0^2 f(0) y_i(0)) + \\ &h \sum_{r=1}^{99} x_r^2 f(x_r) y_i(x_r) + \\ &\frac{h}{2} (2^2 f(2) y_i(2)) \\ &= h^3 \sum_{r=1}^{99} r^2 f(x_r) y_i(x_r) \end{aligned}$$

since $y_i(2) = 0$, and $x_r = rh$.

Similarly

$$\int_0^2 x^2 y_i^2(x) dx \simeq h^3 \sum_{r=1}^{99} r^2 y_i^2(x_r)$$

so that

$$a_i \simeq \left(\sum_{r=1}^{99} r^2 f(x_r) y_i(x_r) \right) / \left(\sum_{r=1}^{99} r^2 y_i^2(x_r) \right)$$

The extent to which we can approximate $f(x)$ is limited by the number of eigenfunctions we can use.

For this problem, the first four eigenvalues are

$$\lambda_1 = 1.933334116$$

$$\lambda_2 = 8.720630338$$

$$\lambda_3 = 20.44117548$$

$$\lambda_4 = 37.09627022$$

so that we can find a four term approximation.

The first step is to determine $y_i(x_r)$.

To this end, we go:

```
eq:= diff(y(x),x,x) + l*x*x*y(x);
```

```
ini:= y(0)= 0, D(y)(0) = 1;
```

```
equ:= subs(l=1.933334116,eq);
```

```
f:= dsolve({equ,ini},y(x),numeric);
```

These are the same instructions as before for preparing to solve the equation numerically.

```
p1 := array(1..99);
```

This establishes an array **p1** in which the values of the first eigenfunction will be stored.

```
for i to 99 do p1[i] := rhs(op(2,f(i/50))) od;
```

This stores the appropriate values in the array.

Note:

(i) the construction **do od** which is used in **maple** for iterations.

(ii) the convoluted way in which a value for the eigenfunction must be extracted.

The instruction **op(2,f(i/50))** isolates the second term $y(x) = *.****$ from the output.

The instruction **rhs(...)** now takes the right hand side of this expression, which is the value we seek.

We now repeat the procedure for the other eigenvalues.

As before, you can use the cursor buttons to reduce the amount of typing you need to do.

```
equ:= subs(l=8.720630338,eq);  
f:= dsolve({equ,ini},y(x),numeric);  
p2 := array(1..99);  
for i to 99 do p2[i] := rhs(op(2,f(i/50))) od;
```

and similarly for the third and fourth eigenvalues.

Suppose for example that the function which we wish to approximate is $x(2-x)$. Then the denominator in the coefficient a_1 is given by

```
den:= sum(r*r*p1[r]*p1[r],r=1..99);
```

and the numerator by

```
num:= sum(r*r*p1[r]*(r/50)*(2-r/50),r=1..99);
```

giving

```
a1 := num/den ;
```

and this process can be repeated for the remaining three coefficients.

Finally we can determine the approximation to our function f which these calculations determine.

Firstly, define

```
app:= array(1..99);
```

and then calculate

```
for i to 99 do app[i] := a1*p1[i] + a2*p2[i] + a3*p3[i] + a4*p4[i] od;
```

and display the result by

```
plot(app[i],i=1..99);
```

Starting flow in a pipe

The problem of the motion of a fluid in a circular pipe when a uniform pressure gradient is applied leads to the partial differential equation

$$\frac{\partial u}{\partial t} = 1 + \left(\frac{\partial^2 u}{\partial r^2} + \frac{1}{r} \frac{\partial u}{\partial r} \right)$$

together with the boundary and initial conditions

$$\begin{aligned} u &= 0 & \text{at } r &= 1 & \text{for all } t, \\ u &= 0 & \text{at } t &= 0 & \text{for } 0 \leq r \leq 1. \end{aligned}$$

This equation is not homogeneous, so that the method of separation of variables cannot be applied directly.

The first step is to remove the inhomogeneous term by solving

$$0 = 1 + \left(\frac{\partial^2 u}{\partial r^2} + \frac{1}{r} \frac{\partial u}{\partial r} \right)$$

subject to the conditions $u = 0$ at $r = 1$, and the implicit requirement that the solution be finite at $r = 0$, which represents the centre of the pipe.

This is an Euler equation which you can solve to get

$$u = \frac{1}{4}(1 - r^2) .$$

If we now substitute

$$u = \frac{1}{4}(1 - r^2) - v$$

into the partial differential equation we obtain

$$\frac{\partial v}{\partial t} = \left(\frac{\partial^2 v}{\partial r^2} + \frac{1}{r} \frac{\partial v}{\partial r} \right)$$

together with the boundary and initial conditions

$$\begin{aligned} v &= 0 \quad \text{at} \quad r = 1 \quad \text{for all } t, \\ v &= \frac{1}{4}(1 - r^2) \quad \text{at} \quad t = 0 \quad \text{for } 0 \leq r \leq 1. \end{aligned}$$

This problem is now amenable to solution by separation of variables.

Since the problem has cylindrical symmetry, the solution involves Bessel functions, in terms of which the solution can be shown to be

$$u(r, t) = \frac{1}{4}(1 - r^2) - \sum_{n=1}^{\infty} c_n J_0(\lambda_n r) \exp(-\lambda_n^2 t)$$

Here the eigenvalues λ_n are the zeroes of $J_0(x)$, which were the subject of the assignment in week 8.

The first six eigenvalues are

2.4048
5.5201
8.6537
11.7915
14.9309
18.0711

and further eigenvalues can be approximated by $\lambda_n \sim (n - \frac{1}{4})\pi$ if required.

When $t = 0$, we obtain

$$\sum_{n=0}^{\infty} c_n J_0(\lambda_n r) = \frac{1}{4}(1 - r^2).$$

The orthogonality relation for the Bessel functions is

$$\int_0^1 r J_0(\lambda_i r) J_0(\lambda_j r) dr = 0 \quad i \neq j$$

so that

$$c_n \int_0^1 r J_0^2(\lambda_n r) dr = \frac{1}{4} \int_0^1 (r - r^3) J_0(\lambda_n r)$$

We can determine these coefficients using maple.

With

```
l:= 2.4048;
```

we can define

```
f:= BesselJ(0,l*r);
```

which is the way in which maple describes the function $J_0(lr)$, and then calculate the integrals

```
num:= evalf(int(0.25*r*(1-r*r)*f, r=0..1));
```

```
den:= evalf(int(r*f*f,r=0..1));
```

The required coefficient is then given by

```
a1:= num/den;
```

More compactly, we can create a procedure to calculate the coefficients.

```
l:= array(1..6);
```

```
l[1]:= 2.4048;
```

```
l[2]:= 5.5201;
```

```
l[3]:= 8.6537;
```

```
l[4]:= 11.7915;
```

```
l[5]:= 14.9309;
```

```
l[6]:= 18.0711;
```

This creates an array whose entries are the eigenvalues.

```
a:= array(1..6);
```

```
for i to 6 do
```

```
  f:= BesselJ(0,l[i]*r);
```

```
  num:= evalf(int(0.25*r*(1-r*r)*f,r=0..1));
```

```
  den:= evalf(int(r*f*f,r=0..1));
```

```
  a[i]:= num/den;
```

```
od;
```

This calculates the coefficients for each of the eigenvalues.

We can use these to plot the approximate solution for $0 \leq \rho \leq 1$, $0 \leq \tau \leq 1$.

We set

```
u:= 0.25*(1-r*r) - sum(a[j]*BesselJ(0,l[j]*r)*exp(-l[j]*l[j]*t),j=1..6);  
plot3d(u,r=0..1,t=0..1);
```

You can change the position and size of the plot using the mouse.

By clicking on the lefthand button, you get an arrow and the outline of a box where the plot had been. The mouse can now be used to alter the size and orientation of this box. When you have got it where you want it, click on the **R** sign to redraw the plot.

As an alternative, you could evaluate u for $t = 0, 0.1, 0.2, 0.5, 1$, and then plot these five expressions as functions of r .

```
u0:= subs(t=0,u);  
u1:= subs(t=0.1,u);  
u2:= subs(t=0.2,u);  
u3:= subs(t=0.5,u);  
u4:= subs(t=1,u);  
plot(u0,u1,u2,u3,u4,r=0..1);
```