2.2 Methodology

2.2.1 For Linear Second Order ODE

Let’s say we have a linear boundary value problem in the form:

A + B + C y + D x = 0 (1)

With boundary conditions

y(x1) = y1 (2)

And

y(x2) = y2 (3)

For example, we have equation (4)

- 2y- 8x (9-x) = 0 (4)

With boundary conditions

y(o) = 0(5)

And

y(9) = 0(6)

Let’s shoot an initial condition say,

y’(0) = 4(7)

2.2.1.1 Euler’s Method

The Euler’s method is a first-order numerical procedure for solving ordinary differential equations (ODE) with a given initial value. Euler’s method uses the first given value to give the next values using the general formula.

*yi*+1*= yi +h f(xi,yi)* (8)

where,

* *yi+1*is the next estimated solution value;
* *yi* is the current value;
* *h* is the interval between steps;
* *f* (*xi,yi*)is the value of the derivative at the current (*xi,yi*) point.

Since, Euler’s method is a first order method, we first convert the equation and make it a first order ODE.

*y*’’ = *f* (*x, y,*  *)* (9)

- 2*y-* 8*x* (9*-x*) *=* 0 (10)

1. Let’s take

=*z = f1* (*x,y,z*) (11)

for the given boundary condition *y*(0)=0.

2) Similarly,

*y’’=*  = *f2* (*x,y,z*) =*2y+8x(9-x)* (12)

for our initial condition *y’*(0)=z(0)=4.

For Euler’s method we have,

1. *yi*+1*= yi +f1(xi,yi,*,*zi)\*h* (13)
2. *zi+1= zi +f2(xi,yi,*,*zi)\*h* (14)

We are taking ***h=3****.*

**At *i=0***

*y*1*= y0 +f1(x0,y0,z0)\*h* (15)

*z*1*= z0 +f2(x0,y0,*,*z0)\*h* (16)

*x0=0; y0=0; z0=4; h=3*

Calculation with equation 15 we get,

*y1= y0 +f1(x0,y0,z0)\*h*

*= 0+f1(0,0,4)\*3=0+4\*3=12* (17)

*z*1*= z0 +f2(x0,y0,*,*z0)\*h*

*= 4+[2y+8x(9-x)]\*3=4+0=4* (18)

*y(x1)y1=12* (19)

*z(x1)z1=4* (20)

*x1=3* (21)

**At *i=1***

*y*2*= y1 +f1(x1,y1,z1)\*h* (22)

*z*2*= z1 +f2(x1,y1,*,*z1)\*h* (23)

*x1=x0+h=0+3=3; y1=12; z1=4; h=3*

Calculation with equation 22 we get,

*y*2*= y1 +f1(x1,y1,z1)\*h*

*= 12+f1(3,12,4)\*3=12+4\*3=24*  (24)

*z*2*= z1 +f2(x1,y1,*,*z1)\*h*

*= 4+[2y+8x(9-x)]\*3=4+68\*3=508* (25)

*y(x2)y2=24* (26)

*z(x2)z2=508* (27)

*x2=6* (28)

**At *i=2***

*y*3*= y2 +f1(x2,y2,z2)\*h* (29)

*z*3*= z2 +f2(x2,y2,*,*z2)\*h* (30)

*x2=x1+h=3+3=6; y1=24; z1=508; h=3*

Calculation with equation 29 we get,

*y*3*= y2 +f1(x2,y2,z2)\*h*

*= 24+f1(6,24,508)\*3=1548*  (31)

*y(x3)y3=1548* (32)

*x3=9* (33)

Equation 32 implies,

*y(x3)=y(9)=1548*  (34)

Since, our guess was wrong, lets take any other arbitrary value for initial value problem.

Say,

*y’*(0)=-24

After going through all the steps of the Euler’s Method we get,

|  |  |  |  |
| --- | --- | --- | --- |
| *i* | *xi* | *yi* | *zi* |
| 0 | 0 | 0 | -24 |
| 1 | 3 | -72 | -24 |
| 2 | 6 | -144 | -24 |
| 3 | 9 | -216 | \_ |

Table 1.1 Solutions of Euler’s Method at y’(0)=4

*y(x3)=y*(9)=-216

Interpolation Technique

Since, we can’t keep making random guesses. Fortunately, we can make are next value more accurate by using the formula to get straight line with two points.

guess 3=guess 2+m(target−solution 2) (35)

m=(guess 1−guess 2)/(solution 1−solution 2) (36)

Using formula 35 and 36 we get

*y’*(0)=-21.57

The solution we from this guess are:

|  |  |  |  |
| --- | --- | --- | --- |
| *i* | *xi* | *yi* | *zi* |
| 0 | 0 | 0 | -21.57 |
| 1 | 3 | 7 | -21.57 |
| 2 | 6 | -123.42 | -24 |
| 3 | 9 | 0.09 | \_ |

Table 1.2 Solutions of Euler’s Method at y’(0)=-20.57

*y(x3)=y*(9)=0.09

Thus, on the third guess, we get the solution close our target value.

2.2.1.2 Runge-Kutta 2nd Method

Using the Runge-Kutta second derivative method for

*y*’(0) = 4

we get,  
 *y*(9)=25200

Taking our second guess as

*y*’(0) = -24

we get,

*y*(9)=-1512

Using formula 35 and 36 we get

*y’*(0)=-18.8679

We got

*y*(9)= 3384.0234

As we can see that well have to use the method again a few times in order to get correct solution while using the Runge-Kutta Method.

2.2.1 For Non-Linear Second Order ODE

To get to a solvable ODE, we start with the conservation of momentum equation (i.e., Navier–Stokes equation) in the x-direction:

u+v +=ν

and the conservation of mass equation:

+ =0

where u is the velocity component in the xx-direction, v is the velocity component in the y-direction, and ν is the fluid’s kinematic viscosity. The boundary conditions are that *u*=*v*=0 at *y*=0, and that u=U∞=U∞ as y→∞ where U∞ is the free-stream velocity.

Blasius solved this problem by converting the PDE into an ODE, by recognizing that the boundary layer thickness is given by δ(x)∼√xν/U∞, and then nondimensionalizing the position coordinates using a similarity variable.

η=*y*

By introducing the stream function, ψ(x,y), we can ensure the continuity equation is satisfied:

*u*=, *v*=

Let’s check this, using SymPy (Appendix 1.4)

True

Using the boundary layer thickness and free-stream velocity, we can define the dimensionless stream function f(η):

f(η)=

which relates directly to the velocity components:

*u==*

*u=. f’().*

*u= f’(*

*v==-(+)= (*

We can insert these into the x-momentum equation, which leads to an ODE for the dimensionless stream function f(η):

*f′′′+ff′′=0*

with the boundary conditions *f=f′=0* at *η=*0, and*f′*=1 as  *η→∞*.

This is a 3rd-order ODE, which we can solve by converting it into three 1st-order ODEs:

*y1=f y′1=y2*

*y2=f’ y′2=y3*

*y3=f’’ y′3=-y1y3*

and we can use the shooting method to solve by recognizing that we have two initial conditions, *y1(0)=y2(0)= 0*, and are missing *y3*(0). We also have a target boundary condition: *y*2(∞)=1.

(Note: obviously we cannot truly integrate over 0≤*η*<∞. Instead, we just need to choose a large enough number. In this case, using 10 is sufficient.)

We created a MATLAB function to evaluate the derivative (appendix 1.1).

Trying the same three-step approach we used for the simpler example, taking two guesses and then using linear interpolation to find a third guess (appendix 1.2) we get:

|  |  |  |
| --- | --- | --- |
| *i* | y(xi) | *y’*(*xi*) |
| 1 | 1 | 1.6553 |
| 2 | 0.1 | 0.3566 |
| 3 | 0.54587 | 1.1056 |

Table 1.3 Solution for Non-Linear PDE using 3 iterations

Our target value was 1.00.

So, for this problem, using linear interpolation did not get us the correct solution on the third try. This is because the ODE is nonlinear. But, you can see that we are converging towards the correct solution—it will just take more tries.

Rather than manually take an unknown (and potentially large) number of guesses, we automate this with a while loop (appendix 1.3).