```
 1:
 2: ****************************************
 3: ./odemod.f90
 4:
 5: MODULE odemod
 6: !****
 7: !**** Module file to hold various numerical methods for
 8: !**** solving ordinary differential equations.
 9: !****
10:
11:   IMPLICIT NONE
12:
13:   CONTAINS
14:
15:  !***********************************************
16:
17:  SUBROUTINE euler(y,x,h)
18:   !******** Does one step of Euler's Method ********
19:
20:     ! *** Dummy declarations ***
21:     REAL, INTENT(IN) :: h
22:     REAL, INTENT(INOUT) :: y,x
23:
24:     y=y+h*ode(x,y)
25:     x=x+h
26:
27:   END SUBROUTINE euler
28:
29:  !***********************************************
30:
31:   SUBROUTINE heun(y,x,h)
32:   !******** Does one step of Heuns Method ********
33:
34:     ! *** Dummy declarations ***
35:     REAL, INTENT(IN) :: h
36:     REAL, INTENT(INOUT) :: y,x
37:
38:     ! *** Local declarations ***
39:     REAL yold
40:
41:     yold=y
42:     CALL euler(y,x,h)
43:     y=yold+h/2*(ode(x-h,yold)+ode(x,y))
44:
45:   END SUBROUTINE heun
46:
47:
48:   !***********************************************
49:
50:   SUBROUTINE rk4(y,x,h)
51:   !**** Does step of Runge-Kutta 4th Method ****
52:
53:     ! *** Dummy declarations ***
54:     REAL, INTENT(IN) :: h
55:     REAL, INTENT(INOUT) :: y,x
56:
57:     ! *** Local declarations ***
58:     REAL :: k1,k2,k3,k4
59:
60:    !**** Calculate k values ****
61:     k1=h*ode(x,y)
62:     k2=h*ode(x+h/2,y+k1/2)
63:     k3=h*ode(x+h/2,y+k2/2)
64:     k4=h*ode(x+h,y+k3)
65:
66:     !**** Calculate y(x) ****
67:     y=y+(k1+2*k2+2*k3+k4)/6
```



```
68:
69:     x=x+h
70:
71:   END SUBROUTINE rk4
72:
73:
74:   SUBROUTINE adam4(y,x,h)
75:   !******** Does one step of adams 4 step Method ********
76:
77:     ! *** Dummy declarations ***
78:     REAL, INTENT(IN) :: h
79:     REAL, INTENT(INOUT) :: y,x
80:
81:     ! *** Local declarations ***
82:     REAL :: y_in
83:     INTEGER, SAVE :: count=1
84:     REAL, SAVE :: oy1,oy2,oy3  !*** Store previous values
85:
86:     y_in=y              !** Make a copy of incoming y_{n}
87:
88:     If (count < 4) THEN !** If not enough starting values
89:       CALL rk4(y,x,h)   !** Do Runge-Kutta 4th order
90:       count=count+1     !** Increment count
91:     ELSE                !** Do the adams 4 step method
92:       y=y+h/24*(55*ode(x,y)-59*ode(x-h,oy1)+37*ode(x-2*h,oy2)-&
93:         9*ode(x-3*h,oy3))
94:       x=x+h             !** increment x
95:     ENDIF
96:
97:     oy3=oy2  !** Update y_{n-3}
98:     oy2=oy1  !** Update y_{n-2}
99:     oy1=y_in !** Update y_{n-1}
100:
101:   END SUBROUTINE adam4
102:
103:
104:   !***********************************************
105:
106: SUBROUTINE modmilne(y,x,h)
107:
108:   !******** Does one step modified Milne's Method ********
109:
110:     ! *** Dummy declarations ***
111:     REAL, INTENT(IN) :: h
112:     REAL, INTENT(INOUT) :: y,x
113:
114:     ! *** Local declarations ***
115:     REAL :: mpy,py,y_in
116:     INTEGER, SAVE :: count=1
117:     REAL, SAVE :: oy1,oy2,oy3,opy1
118:
119:
120:     y_in=y              !** Make a copy of y_{n}
121:
122:     If (count < 4) THEN !** If not enough starting values
123:       CALL rk4(y,x,h)   !** Do Runge-Kutta 4th order
124:       count=count+1     !** Increment count
125:     ELSE IF (count == 4) THEN !** Do the Milne's method
126:       py=oy3+4.0*h*(2*ode(x,y)-ode(x-h,oy1)+2*ode(x-2*h,oy2))/3
127:       y=oy1+h/3*(ode(x+h,py)+4*ode(x,y)+ode(x-h,oy1))
128:       x=x+h             !** increment x
129:       count=count+1     !** Increment count
130:     ELSE                !** Do the Modified Milne method
131:       py=oy3+4.0*h*(2*ode(x,y)-ode(x-h,oy1)+2*ode(x-2*h,oy2))/3
132:       mpy=py+28.0/29*(y-opy1)
133:       y=oy1+h/3*(ode(x+h,mpy)+4*ode(x,y)+ode(x-h,oy1))
134:       x=x+h             !** increment x
```

```
   ENDIF

    oy3=oy2  !** Update y_{n-3}
    oy2=oy1  !** Update y_{n-2}
    oy1=y_in !** Update y_{n-1}
    opy1=py  !** Update py_{n-1}

  END SUBROUTINE modmilne

  !*********************************************

  FUNCTION ode(x,y)
  !** Function to return the value of the the differential
  !** equation for a given x and y

    !*** Dummy declarations
    REAL, INTENT(IN) :: x,y

    !*** Function declaration ***
    REAL :: ode

    ode=-y+x+2

  END FUNCTION ode

  !*********************************************

END MODULE odemod
*****************************************


*****************************************
./results_2013.f90

======================================
                Euler
======================================
   x     yapprox      Y-True       Error
0.00    2.000000    2.000000    0.000000
0.10    2.000000    2.004838    0.004838
0.20    2.010000    2.018731    0.008731
0.30    2.029000    2.040818    0.011818
0.40    2.056100    2.070320    0.014220
0.50    2.090490    2.106531    0.016041
0.60    2.131441    2.148812    0.017370
0.70    2.178297    2.196585    0.018288
0.80    2.230467    2.249329    0.018862
0.90    2.287421    2.306570    0.019149
1.00    2.348679    2.367879    0.019201


======================================
                Heun
======================================

0.00    2.000000    2.000000    0.000000
0.10    2.005000    2.004838    0.000163
0.20    2.019025    2.018731    0.000294
0.30    2.041218    2.040818    0.000400
0.40    2.070802    2.070320    0.000482
0.50    2.107076    2.106531    0.000545
0.60    2.149404    2.148812    0.000592
0.70    2.197211    2.196585    0.000625
0.80    2.249975    2.249329    0.000646
0.90    2.307228    2.306570    0.000658
1.00    2.368541    2.367879    0.000662
```



```


======================================
                 RK4
======================================

   x     yapprox      Y-True       Error
0.00    2.000000    2.000000    0.000000
0.10    2.004838    2.004838    0.000000
0.20    2.018731    2.018731    0.000000
0.30    2.040818    2.040818    0.000000
0.40    2.070320    2.070320    0.000000
0.50    2.106531    2.106531    0.000000
0.60    2.148812    2.148812    0.000000
0.70    2.196585    2.196585    0.000000
0.80    2.249329    2.249329    0.000000
0.90    2.306570    2.306570    0.000000
1.00    2.367880    2.367879    0.000000

======================================
                ADAM4
======================================

   x     yapprox      Y-True       Error
0.00    2.000000    2.000000    0.000000
0.10    2.004838    2.004838    0.000000
0.20    2.018731    2.018731    0.000000
0.30    2.040818    2.040818    0.000000
0.40    2.070323    2.070320    0.000003
0.50    2.106536    2.106531    0.000005
0.60    2.148818    2.148812    0.000007
0.70    2.196594    2.196585    0.000008
0.80    2.249338    2.249329    0.000009
0.90    2.306580    2.306570    0.000010
1.00    2.367890    2.367879    0.000010

======================================
               MODMILNE
======================================

   x     yapprox      Y-True       Error
0.00    2.000000    2.000000    0.000000
0.10    2.004838    2.004838    0.000000
0.20    2.018731    2.018731    0.000000
0.30    2.040818    2.040818    0.000000
0.40    2.070320    2.070320    0.000000
0.50    2.106531    2.106531    0.000000
0.60    2.148811    2.148812    0.000000
0.70    2.196586    2.196585    0.000000
0.80    2.249329    2.249329    0.000000
0.90    2.306570    2.306570    0.000000
1.00    2.367879    2.367879    0.000000

*****************************************


*****************************************
./ode.f90

PROGRAM ode
!**
!** Driver main program for ode module
!**

  USE odemod, ONLY : euler,heun,rk4,adam4,modmilne

  IMPLICIT NONE
```

```
  269:
  270:   REAL, PARAMETER :: h=0.1
  271:
  272:   REAL :: x,&      !** Hold current "x" value of ode
  273:           y,&      !** Hold current "y" value of ode
  274:           ytrue,&  !** Hold current true solution
  275:           err=0   !** Hold the current error "ABS(y-ytrue)"
  276:
  277:   INTEGER :: i
  278:
  279:   x=0 ; y=2 !*** Initial Values
  280:
  281:   ytrue=EXP(-x)+x+1
  282:   PRINT '(a4,a12,a12,a12)',"x","yapprox","Y-True","Error"
  283:   PRINT '(F4.2,F12.6,F12.6,F12.6)',x,y,ytrue,err
  284:
  285:   DO i=1,10
  286:
  287:     !CALL euler(y,x,h)
  288:     !CALL heun(y,x,h)
  289:     !CALL rk4(y,x,h)
  290:     !CALL adam4(y,x,h)
  291:     CALL modmilne(y,x,h)
  292:
  293:     ytrue=EXP(-x)+x+1
  294:     err=ABS(y-ytrue)
  295:     PRINT '(F4.2,F12.6,F12.6,F12.6)',x,y,ytrue,err
  296:   ENDDO
  297:
  298: END PROGRAM ode
  299: ****************************************
  300:
```