Generating the plots

The following program written in Microsoft® Small Basic will ask the user for several input choices for an advection approximation, and then runs and plots the results. This program can be downloaded from the internet at *http://smallbasic.com/program/?LWSB363.000*.

```
Advection Tester in Small Basic
3/02/2021  CWH
(share with friends at http://smallbasic.com/program/?MXDC144.000)



TextWindow.Write("  ********ADVECTION TESTER ********  ")
TextWindow.Write("        FLOW SCIENCE, INC           ")
TextWindow.Write("      Written in Small Basic        ")
TextWindow.Write("  ")
TextWindow.Write("Input End Time Cycle=")
C=TextWindow.Read()
TextWindow.Write( " ")
TextWindow.Write("Donor/Centered (D=1, C=0)=")
A=TextWindow.Read()
TextWindow.Write("  ")
TextWindow.Write("Implicit/Explicit (I=1, E-0)=")
T=TextWindow.Read()
TextWindow.Write("  ")
TextWindow.Write("Second Derivative (Y=1, N=0)")
B=TextWindow.Read()
TextWindow.Write("  ")
TextWindow.Write("UDT/DX=")
D=TextWindow.Read()
TextWindow.Write("  ")
K=0
M1=20
L1=70
For I=1 To L1
  E[I]=0
  F[I]=0
  S[I]=0
  Q[I]=1
  If (I>=L1/2) then'
    Q[I]=0
  Endif
Endfor
C1=0.5*D*T*(1-A)
C2=1+D*A*T
C3=C1-T*D
C4=0.5*(1-T-D*B-A+A*T)*D
C5=-1+D*(D*B+A-A*T)
C6=C4-D*(1-T)
Repeat:
For I=2 To L1-1
  S[I]=C4*Q[I+1]+C5*Q[I]+C6*Q[I-1]
Endfor
E[2]=-1/(C2+C3)
F[2]=S[2]*E[2]
```

```
For I=3 To L1-1
  E[I]=-1/(C2+C1*C3*E[I-1])
  F[I]=E[I]*(C3*F[I-1]+S[I])
Endfor
Q[L1]=F[L1-1]/(1-E[L1-1]*C1)
For I=2 To L1-1
  J=L1+1-I
  Q[J]=C1*E[J]*Q[J+1]+F[J]
Endfor
K=K+1
If K<C Then
  goto Repeat
Endif
GraphicsWindow.Title="Advection Tester"
GraphicsWindow.Height=650
GraphicsWindow.Width=450
GraphicsWindow.Show()
GraphicsWindow.PenWidth=4
GraphicsWindow.DrawRectangle(0,0,450,400)
GraphicsWindow.PenWidth=3
GraphicsWindow.DrawRectangle(50,50,350,300)
GraphicsWindow.PenWidth=2
MAG=5
X1=50+2*MAG
Y1=50+(40-Q[2]*M1)*MAG
Y0=40
For I =3 To L1-1
  X2=50+I*MAG
  Y2=50+(Y0-Q[I]*M1)*MAG
  GraphicsWindow.Drawline(X1,Y1,X2,Y2)
  X1=X2
  Y1=Y2
Endfor
GraphicsWindow.Drawline(50,250,400,250)
For I=2 To L1-1
  YB=50+39.5*MAG
  YT=50+40.5*MAG
  X=50+I*MAG
  GraphicsWindow.Drawline(X,YB,X,YT)
EndFor
GraphicsWindow.Penwidth=1
X2=50+(L1/2+C*D)*MAG
Y2=50+40*MAG
For J=1.To M1
  GraphicsWindow.DrawLine(X2,Y2,X2,Y2-0.5*MAG)
  Y2=Y2-MAG
EndFor
```

The above program has also been written in the programming language Python and is listed below.

```
#!/usr/bin/env python

# ADVECTION TESTER
# FLOW SCIENCE, INC.

import sys

try:
    import matplotlib.pyplot as plt
except:
    print()
    print("Missing package! Please install the python package
matplotlib")
    print("using the following command:")
    print()
    print("pip install matplotlib")
    print()
    print("(Note: Installation instructions can vary across systems)")
    sys.exit(1)

# UNCOMMENT LINES BELOW IF YOU WANT INTERACTIVE INPUT
#C = input('CYCLE = ')
#A = input('DONOR/CENTERED (D=1, C=0) = ')
#T = input('IMPLICIT/EXPLICIT (I=1, E=0) = ')
#B = input('SECOND DERIVATIVE (Y=1, N=0) = ')
#D = input('DELT (0,1) = ')
C=100
A=0
T=1
B=1
D=0.2
Q = [1 for _ in range(100)]
E = [1 for _ in range(100)]
F = [1 for _ in range(100)]
S = [1 for _ in range(100)]

# SET INITIAL VALUES
K = 0
M1 = 20
L1 = 70
for I in range(100):
      Q[I]=1
      if I < L1/2+1:
            continue
      Q[I] = 0

# COMPUTE COEFFICIENTS
C1 = 0.5*D*T*(1-A)
C2 = 1+D*A*T
C3 = C1-T*D
C4 = 0.5*(1-T-D*B-A+A*T)*D
C5 = -1+D*(D*B+A-A*T)
C6=C4-D*(1-T)
```

```
# COMPUTE EXPLICIT SOURCES
while K < C:
        for I in range(1,L1-1):
            S[I] = C4*Q[I+1]+C5*Q[I]+C6*Q[I-1]

        E[1] = -1/(C2+C3)
        F[1] = S[1]*E[1]

# COMPUTE CYCLE
        for I in range(2,L1-1):
            E[I] = -1/(C2+C1*C3*E[I-1])
            F[I] = E[I]*(C3*F[I-1]+S[I])
        Q[L1-1] = F[L1-2] / (1-E[L1-2]*C1)

        for I in range(1,L1-1):
            J=L1-1-I
            Q[J] = C1*E[J]*Q[J+1]+F[J]

# UPDATE CYCLE COUNT AND TEST FOR PLOT
        K = K+1

plt.figure(1)

# PLOT THE NUMERICAL SOLUTION
X1 = 1
Y1 = 20
xplot1 = [0 for _ in range(L1-1)]
yplot1 = [0 for _ in range(L1-1)]
xplot1[0] = X1
yplot1[0] = Y1+Q[0]*M1
for I in range(1,L1-1):
      xplot1[I]=X1+I
      yplot1[I]=Y1+Q[I]*M1

plt.plot(xplot1, yplot1, 'r-')

# PLOT THE REFERENCE SOLUTION
X2=X1+int(L1/2)+C*D
Y2=Y1+M1
Z1=int(C*D)
plt.plot([X2-Z1, X2, X2], [Y2, Y2, Y1], 'b--')

# PLOT THE BOTTOM LINE
plt.plot([X1, X1+L1], [Y1, Y1], 'k-', linewidth=2.0)
plt.ylim([10, 45])

# ADD INPUTS TO PLOT
Y_LOC = M1 - 2
X_LOC = L1/2-5
Y_SPACE = 1.5

plt.text(X_LOC, Y_LOC, 'INPUTS:', size=9)

Y_LOC -= Y_SPACE
plt.text(X_LOC, Y_LOC, '1. CYCLE = {0}'.format(C), size=8)
```

```
Y_LOC -= Y_SPACE
if A==1:
    plt.text(X_LOC, Y_LOC, '2. DONOR/CENTERED = DONOR', size=8)
else:
    plt.text(X_LOC, Y_LOC, '2. DONOR/CENTERED = CENTERED', size=8)

Y_LOC -= Y_SPACE
if T==1:
    plt.text(X_LOC, Y_LOC, '3. IMPLICIT/EXPLICIT = IMPLICIT', size=8)
else:
    plt.text(X_LOC, Y_LOC, '3. DONOR/CENTERED = EXPLICIT', size=8)

Y_LOC -= Y_SPACE
if B==1:
    plt.text(X_LOC, Y_LOC, '4. SECOND DERIVATIVE', size=8)
else:
    plt.text(X_LOC, Y_LOC, '4. FIRST DERIVATIVE', size=8)

Y_LOC -= Y_SPACE
plt.text(X_LOC, Y_LOC, '5. UDT/DX = {0}'.format(D), size=8)

plt.legend(('Numerical Solution', 'Reference Solution'), loc='best')

plt.savefig('result.png')

# IF YOU DON'T WANT THE PLOT TO POP UP, COMMENT THE LINE BELOW
plt.show()
```