**נספח**

התוכנית הבאה משתמשת בנוסחת אוילר בכדי לחשב פתרון למשוואה דיפרנציאלית רגילה y' = -2xy2:

/\* euler1.c - solve ordinary differential equations \*/

#include <stdio.h>

#include <math.h>

void euler(long double (\*f)(long double, long double),

long double x0, long double y0,

long double h, long double yk[], int n)

{

int i;

for(i=0; i < n; i++)

{

yk[i] = y0;

x0 = x0 + h;

y0 = y0 + h\*(\*f)(x0,y0);

} /\* for \*/

} /\* euler \*/

long double f(long double x, long double y)

{

return -2.0\*(x\*y\*y);

} /\* f \*/

long double If(long double x)

{

return 1.0/(x\*x+1);

} /\* If \*/

int main()

{

int i;

long double x0, h, yk[10];

h= 0.1;

euler(f, 0.0, 1.0, h, yk, 10);

printf("i xk If(xk) yk\n");

x0 = 0;

for(i=0; i < 10; i++)

{

printf("%1d, %Lf, %Lf %Lf\n", i, x0, If(x0), yk[i]);

x0 = x0 + h;

} /\* for \*/

} /\* main \*/

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

C:\> euler1.exe

i xk If(xk) yk

0, 0.000000, 1.000000 1.000000

1, 0.100000, 0.990099 0.980000

2, 0.200000, 0.961538 0.941584

3, 0.300000, 0.917431 0.888389

4, 0.400000, 0.862069 0.825250

5, 0.500000, 0.800000 0.757147

6, 0.600000, 0.735294 0.688354

7, 0.700000, 0.671141 0.622018

8, 0.800000, 0.609756 0.560113

9, 0.900000, 0.552486 0.503642

C:\>

להלן תוכנית שמתמודדת עם אותה בעיה בשיטת אוילר ההפוכה, השיטה במשתמעת (implicit):

/\* back\_euler1.c - solve ordinary differential equations \*/

#include <stdio.h>

#include <math.h>

long double If(long double x)

{

return 1.0/(x\*x+1);

} /\* If \*/

double back\_euler\_formula(long double yn, long double h,

long double f, long double yn1)

{

return (yn1 - yn - h\*f);

} //back\_euler\_formula

double back\_euler\_formula\_deriv(long double fdy, long double h)

{

return (1.0 - h\*fdy);

} //back\_euler\_formula

void back\_euler(long double (\*f)(long double, long double),

long double (\*fdy)(long double, long double),

long double x0, long double y0,

long double h, int n,

long double yn[])

{

long double eps, d0, f0, fdy0, t, frac, objf;

int i;

yn[0] = y0;

t = x0+h;

for(i=1; i < n; i++)

{

eps = yn[i-1]/1000000000000;

yn[i] = yn[i-1]+ h\*(\*f)(t,yn[i-1]); /\* Predictor \*/

/\* newton method – Corrector \*/

do {

f0 = (\*f)(t,yn[i]);

fdy0 = (\*fdy)(t,yn[i]);

objf = back\_euler\_formula(yn[i-1],h, f0, yn[i]);

frac = objf/back\_euler\_formula\_deriv(fdy0, h);

yn[i] = yn[i] - frac;

} while (fabsl(objf) > eps);

t = t + h;

} /\* for \*/

} /\* back\_euler \*/

long double f(long double x, long double y)

{

return -2.0\*(x\*y\*y);

} /\* f \*/

long double fdy(long double x, long double y)

{

return - (4.0)\*(x\*y);

} /\* fdy \*/

int main()

{

int i;

long double x0, h;

long double yk[10];

h= 0.1;

back\_euler(f, fdy, 0 , 1, h, 10, yk);

printf("i xk If(xk) yk\n");

x0 = 0;

for(i=0; i < 10; i++)

{

printf("%1d, %Lf, %Lf %Lf\n", i, x0, If(x0), yk[i]);

x0 = x0 + h;

} /\* for \*/

} /\* main \*/

פלט ריצה:

E:\>back\_euler1.exe

i xk If(xk) yk

0, 0.000000, 1.000000 1.000000

1, 0.100000, 0.990099 0.980762

2, 0.200000, 0.961538 0.945038

3, 0.300000, 0.917431 0.896785

4, 0.400000, 0.862069 0.840297

5, 0.500000, 0.800000 0.779530

6, 0.600000, 0.735294 0.717716

7, 0.700000, 0.671141 0.657241

8, 0.800000, 0.609756 0.599699

9, 0.900000, 0.552486 0.546032

E:\>

להלן תוכנית המממשת את שיטת Adams-Bashforth הראשונה, על אותה משוואה כמו הקודמים:

/\* adams1.c - solve ordinary differential equations \*/

#include <stdio.h>

#include <math.h>

void adams(long double (\*f)(long double, long double),

long double x0, long double y0,

long double x1, long double y1,

long double h, long double yk[], int n)

{

int i;

long double yn, xn, xn1;

yk[0] = y0;

yk[1] = y1;

xn1 = x0;

xn = x1;

for(i=1; i < n; i++)

{

yk[i+1] = yk[i] + h\*((1.5\*(\*f)(xn,yk[i]))- (0.5\*(\*f)(xn1,yk[i-1])));

xn1 = xn;

xn = xn + h;

} /\* for \*/

} /\* adams \*/

long double f(long double x, long double y)

{

return -2.0\*(x\*y\*y);

} /\* f \*/

long double If(long double x)

{

return 1.0/(x\*x+1);

} /\* If \*/

int main()

{

int i;

long double x0, h, yk[10];

h= 0.1;

adams(f, 0.0, 1.0, 0.1, 0.990099, h, yk, 10);

printf("i xk If(xk) yk\n");

x0 = 0;

for(i=0; i < 10; i++)

{

printf("%1d, %Lf, %Lf %Lf\n", i, x0, If(x0), yk[i]);

x0 = x0 + h;

} /\* for \*/

} /\* main \*/

פלט ריצה:

E:\> adams1.exe

i xk If(xk) yk

0, 0.000000, 1.000000 1.000000

1, 0.100000, 0.990099 0.990099

2, 0.200000, 0.961538 0.960690

3, 0.300000, 0.917431 0.915118

4, 0.400000, 0.862069 0.858206

5, 0.500000, 0.800000 0.794947

6, 0.600000, 0.735294 0.729617

7, 0.700000, 0.671141 0.665393

8, 0.800000, 0.609756 0.604356

9, 0.900000, 0.552486 0.547689

E:\>

להלן תוכנית המתודדת עם אותה בעיה בשיטת RK4:

/\* rk4.c - solve ordinary differential equations \*/

#include <stdio.h>

#include <math.h>

void rk4(long double (\*f)(long double, long double),

long double x0, long double y0,

long double h, long double yk[], int n)

{

int i;

long double k1, k2, k3, k4;

long double tn, yn;

tn = x0;

yn = y0;

yk[0] = y0;

for(i=1; i < n; i++)

{

k1 = f(tn, yn);

k2 = f(tn+h/2.0, yn+ h\*k1/2.0);

k3 = f(tn+h/2.0, yn+ h\*k2/2.0);

k4 = f(tn+h, yn+ h\*k3);

tn = tn + h;

yn = yk[i] = yn + h\*(k1 + 2\*k2 + 2\*k3 +k4)/6.0;

} /\* for \*/

} /\* rk4 \*/

long double f(long double x, long double y)

{

return -2.0\*(x\*y\*y);

} /\* f \*/

long double If(long double x)

{

return 1.0/(x\*x+1);

} /\* If \*/

int main()

{

int i;

long double x0, h, yk[10];

h= 0.1;

rk4(f, 0.0, 1.0, h, yk, 10);

printf("i xk If(xk) yk\n");

x0 = 0;

for(i=0; i < 10; i++)

{

printf("%1d, %Lf, %Lf %Lf\n", i, x0, If(x0), yk[i]);

x0 = x0 + h;

} /\* for \*/

} /\* main \*/

פלט ריצה:

E:\>rk4.exe

i xk If(xk) yk

0, 0.000000, 1.000000 1.000000

1, 0.100000, 0.990099 0.990099

2, 0.200000, 0.961538 0.961538

3, 0.300000, 0.917431 0.917431

4, 0.400000, 0.862069 0.862068

5, 0.500000, 0.800000 0.799999

6, 0.600000, 0.735294 0.735294

7, 0.700000, 0.671141 0.671141

8, 0.800000, 0.609756 0.609756

9, 0.900000, 0.552486 0.552487

E:\>