**נספח**

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

/\* euler.java - solve ordinary differential equations \*/

class Function

{

 static double f( double x, double y)

 {

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

 } /\* f \*/

 static double If( double x)

 {

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

 } /\* If \*/

}// Function

public class euler1

{

 static void euler(double x0, double y0,

 double h, int n, double yk[])

 {

 int i;

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

 {

 yk[i] = y0;

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

 x0 = x0 + h;

 } /\* for \*/

 } /\* euler \*/

 static public void main(String args[])

 {

 int i;

 double x0, h;

 double yk[] = new double[10];

 h= 0.1;

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

 x0 = 0;

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

 {

 System.out.println( "i = " +i +", xi = " + x0 + ", If(xi) = " +

 Function.If(x0) +

 " yi = " + yk[i]);

 x0 = x0 + h;

 } /\* for \*/

 } /\* main \*/

} // euler1

פלט ריצה:

i = 0, xi = 0.0, If(xi) = 1.0 yi = 1.0

i = 1, xi = 0.1, If(xi) = 0.9900990099009901 yi = 1.0

i = 2, xi = 0.2, If(xi) = 0.9615384615384615 yi = 0.98

i = 3, xi = 0.30000000000000004, If(xi) = 0.9174311926605504 yi = 0.941584

i = 4, xi = 0.4, If(xi) = 0.8620689655172413 yi = 0.88838917425664

i = 5, xi = 0.5, If(xi) = 0.8 yi = 0.8252503482617284

i = 6, xi = 0.6, If(xi) = 0.7352941176470589 yi = 0.757146534531118

i = 7, xi = 0.7, If(xi) = 0.6711409395973155 yi = 0.6883540295608203

i = 8, xi = 0.7999999999999999, If(xi) = 0.6097560975609756 yi = 0.6220176517590537

i = 9, xi = 0.8999999999999999, If(xi) = 0.5524861878453039 yi = 0.5601126983030781

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

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

class Function

{

 static double If( double x)

 {

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

 } // If

 static double f( double x, double y)

 {

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

 } /\* f \*/

 static double fdy( double x, double y)

 {

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

 } /\* fdy \*/

} //Function

public class back\_euler1

{

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

 double f, double yn1)

 {

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

 } //back\_euler\_formula

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

 {

 return (1.0 - h\*fdy);

 } //back\_euler\_formula

 static void back\_euler(double x0, double y0,

 double h, int n,

 double yn[])

 {

 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]/1000000000;

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

 /\* newton method - Corrector \*/

 do {

 f0 = Function.f(t,yn[i]);

 fdy0 = Function.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 (Math.abs(objf) > eps);

 t = t + h;

 } /\* for \*/

 } /\* back\_euler \*/

public static void main(String args[])

{

 int i;

 double x0, h;

 double yk[] = new double[10];

 h= 0.1;

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

 System.out.println("i xk If(xk) yk\n");

 x0 = 0;

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

 {

 System.out.println("i = " + i + " xi = " + x0 +

 " If(xi) = " + Function.If(x0) + " yk[i] = " + yk[i] );

 x0 = x0 + h;

 } /\* for \*/

 } /\* main \*/

} // back\_euler1

פלט ריצה:

i = 0, xi = 0.0, If(xi) = 1.0 yi = 1.0

i = 1, xi = 0.1, If(xi) = 0.9900990099009901 yi = 1.0

i = 2, xi = 0.2, If(xi) = 0.9615384615384615 yi = 0.98

i = 3, xi = 0.30000000000000004, If(xi) = 0.9174311926605504 yi = 0.941584

i = 4, xi = 0.4, If(xi) = 0.8620689655172413 yi = 0.88838917425664

i = 5, xi = 0.5, If(xi) = 0.8 yi = 0.8252503482617284

i = 6, xi = 0.6, If(xi) = 0.7352941176470589 yi = 0.757146534531118

i = 7, xi = 0.7, If(xi) = 0.6711409395973155 yi = 0.6883540295608203

i = 8, xi = 0.7999999999999999, If(xi) = 0.6097560975609756 yi = 0.6220176517590537

i = 9, xi = 0.8999999999999999, If(xi) = 0.5524861878453039 yi = 0.5601126983030781

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

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

class Function

{

 static double f( double x, double y)

 {

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

 } /\* f \*/

 static double If( double x)

 {

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

 } /\* If \*/

} // Function

public class adams1

{

 static void adams(double x0, double y0,

 double x1, double y1,

 double h, double yk[], int n)

 {

 int i;

 double yn, xn, xn1;

 yk[0] = y0;

 yk[1] = y1;

 xn1 = x0;

 xn = x1;

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

 {

 yk[i+1] = yk[i] + h\*((1.5\*Function.f(xn,yk[i])) -

 (0.5\*Function.f(xn1,yk[i-1])));

 xn1 = xn;

 xn = xn + h;

 } /\* for \*/

 } /\* adams \*/

 static public void main(String args[])

 {

 int i;

 double x0, h, yk[] = new double[10];

 h= 0.1;

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

 System.out.println("i xk If(xk) yk\n");

 x0 = 0;

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

 {

 System.out.println("i = " + i +", xi = " + x0 +

 ", If(xi) = " + Function.If(x0) +

 ", yi = " + yk[i]);

 x0 = x0 + h;

 } /\* for \*/

 } /\* main \*/

} // adams1

פלט ריצה:

i xk If(xk) yk

i = 0, xi = 0.0, If(xi) = 1.0, yi = 1.0

i = 1, xi = 0.1, If(xi) = 0.9900990099009901, yi = 0.990099

i = 2, xi = 0.2, If(xi) = 0.9615384615384615, yi = 0.9606901191059699

i = 3, xi = 0.30000000000000004, If(xi) = 0.9174311926605504, yi = 0.9151175491071093

i = 4, xi = 0.4, If(xi) = 0.8620689655172413, yi = 0.8582064476245239

i = 5, xi = 0.5, If(xi) = 0.8, yi = 0.7949474546757215

i = 6, xi = 0.6, If(xi) = 0.7352941176470589, yi = 0.7296169685911824

i = 7, xi = 0.7, If(xi) = 0.6711409395973155, yi = 0.6653926756218392

i = 8, xi = 0.7999999999999999, If(xi) = 0.6097560975609756, yi = 0.6043561741912605

i = 9, xi = 0.8999999999999999, If(xi) = 0.5524861878453039, yi = 0.5476893606173006

להלן תוכנית המתודדת עם אותה בעיה בשיטת RK4:

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

class Function

{

 static double f( double x, double y)

 {

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

 } /\* f \*/

 static double If( double x)

 {

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

 } /\* If \*/

} // Function

public class rk4a

{

 static void rk4( double x0, double y0,

 double h, double yk[], int n)

 {

 int i;

 double k1, k2, k3, k4;

 double tn, yn;

 tn = x0;

 yn = y0;

 yk[0] = y0;

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

 {

 k1 = Function.f(tn, yn);

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

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

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

 tn = tn + h;

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

 } /\* for \*/

 } /\* rk4 \*/

 public static void main(String args[])

 {

 int i;

 double x0, h, yk[]= new double[10];

 h= 0.1;

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

 System.out.println("i xk If(xk) yk\n");

 x0 = 0;

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

 {

 System.out.println("i = " + i + ", xi = " + x0 +

 ", If(xi) = " + Function.If(x0) +

 ", yi = " + yk[i]);

 x0 = x0 + h;

 } /\* for \*/

 } /\* main \*/

} // rk4a

פלט ריצה:

i xk If(xk) yk

i = 0, xi = 0.0, If(xi) = 1.0, yi = 1.0

i = 1, xi = 0.1, If(xi) = 0.9900990099009901, yi = 0.9900989249501665

i = 2, xi = 0.2, If(xi) = 0.9615384615384615, yi = 0.961538143658087

i = 3, xi = 0.30000000000000004, If(xi) = 0.9174311926605504, yi = 0.9174305975195712

i = 4, xi = 0.4, If(xi) = 0.8620689655172413, yi = 0.8620681834882847

i = 5, xi = 0.5, If(xi) = 0.8, yi = 0.7999992090185385

i = 6, xi = 0.6, If(xi) = 0.7352941176470589, yi = 0.7352935002790218

i = 7, xi = 0.7, If(xi) = 0.6711409395973155, yi = 0.6711406195178697

i = 8, xi = 0.7999999999999999, If(xi) = 0.6097560975609756, yi = 0.6097561191882678

i = 9, xi = 0.8999999999999999, If(xi) = 0.5524861878453039, yi = 0.552486529856882