**נספח**

התוכנית הבאה משתמשת בשיטת ניוטון בכדי למצוא את המינימום של הפונקציה f(x) = -sin(x).

/\* newton2.java - use newton's method for optimization \*/

class Function

{

static double f( double x)

{

return -Math.sin(x);

} /\* f \*/

static double fd( double x)

{

return -Math.cos(x);

} /\* fd \*/

static double fdd( double x)

{

return Math.sin(x);

} /\* fdd \*/

} //Functions

public class newton2

{

static double newton(double x0, double eps)

{

double fdv, f0;

int i;

do {

f0 = Function.fd(x0);

fdv = Function.fdd(x0);

x0 = x0 - f0/fdv;

} while (Math.abs(f0) > eps);

return x0;

} /\* newton \*/

public static void main (String args[])

{

double x;

x = newton(1.0, 0.0000001);

System.out.println("Solution to min -sin(x), x = " + x +

", f(" + x + ") = " + Function.f(x) +

", fd(" + x+ ") = " + Function.fd(x));

} /\* main \*/

} // newton2

פלט ריצה:

Solution to min -sin(x), x = 1.5707963267948966, f(1.5707963267948966) = -1.0, fd(1.5707963267948966) = -6.123233995736766E-17

התוכנית הבאה משתמשת בשיטת המיתר למציאת המינימום של –sinx:

/\* secant\_op1.java - use secant\_op's method for optimization \*/

class Function

{

static double f( double x)

{

return -Math.sin(x);

} /\* f \*/

static double fd( double x)

{

return -Math.cos(x);

} /\* fd \*/

} //Functions

public class secant\_op1

{

static double secant\_op(double x0, double x1, double eps)

{

double f1, f0, x;

int i;

do {

f1 = Function.fd(x1);

f0 = Function.fd(x0);

x = x1 - f1\*(x1 - x0)/(f1 - f0);

x0 = x1;

x1 = x;

} while (Math.abs(f1) > eps);

return x;

} /\* secant\_op \*/

public static void main (String args[])

{

double x;

x = secant\_op(1.0, 2.000, 0.0000001);

System.out.println("Solution to min -sin(x), x = " + x +

", f(" + x + ") = " + Function.f(x) +

", fd(" + x+ ") = " + Function.fd(x));

} /\* main \*/

} // secant\_op1

פלט ריצה:

Solution to min -sin(x), x = 1.5707963267948966, f(1.5707963267948966) = -1.0, fd(1.5707963267948966) = -6.123233995736766E-17

התוכנית הבאה מוצאת את המינימום של f(x) = -sin(x) באינטרוול (1,2) ע"י שיטת ה-quadratic fit

/\* quad1.java \*/

class Function

{

static double f( double x)

{

return -Math.sin(x);

} /\* f \*/

} // Function

public class quad1

{

static double quad( double x1, double x3, double eps)

{

double fx1, x2, fx2, fx3, x4, fx4;

double a12, a31, a23, b31, b23, b12;

double oldx4;

x2 = (x1 + x3)/2;

fx1 = Function.f(x1);

fx2 = Function.f(x2);

fx3 = Function.f(x3);

x4 = x2;

do {

a12 = x1 - x2;

a23 = x2 - x3;

a31 = x3 - x1;

b12 = x1\*x1 - x2\*x2;

b31 = x3\*x3 - x1\*x1;

b23 = x2\*x2 - x3\*x3;

oldx4 = x4;

x4 = 0.5\*(b23\*fx1 + b31\*fx2 + b12\*fx3)/(a23\*fx1 + a31\*fx2 +

a12\*fx3);

fx4 = Function.f(x4);

if (x4 < x2)

if (fx4 < fx2 ) /\* (x1, x4, x2) \*/

{

x3 = x2;

x2 = x4;

fx3 = fx2;

fx2 = fx4;

} /\* if \*/

else /\* (x4, x2, x3) \*/

{

x1 = x4;

fx1 = fx4;

} /\* else \*/

else /\* x2 <= x4 \*/

if (fx4 < fx2 ) /\* (x2, x4, x3) \*/

{

x1 = x2;

fx1 = fx2;

x2 = x4;

fx2 = fx4;

} /\* else \*/

else /\* (x1, x2, x4) \*/

{

x3 = x4;

fx3 = fx4;

} /\* else \*/

} while ( Math.abs(oldx4 - x4) > eps);

return x4;

} /\* quad \*/

public static void main(String args[])

{

double x;

x = quad(1.0, 2.0, 0.0001);

System.out.println("Min solution x = " + x +

", min value = " + Function.f(x));

} /\* main \*/

} // quad1

פלט ריצה:

Min solution x = 1.570796326362451, min value = -1.0