**נספח**

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

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

#include <stdio.h>

#include <math.h>

long double newton(long double (\*fun)(long double),

long double (\*fd)(long double),

long double x0, long double eps)

{

long double fdv, f0;

int i;

do {

f0 = (\*fun)(x0);

fdv = (\*fd)(x0);

x0 = x0 - f0/fdv;

} while (fabsl(f0) > eps);

return x0;

} /\* newton \*/

long double f(long double x)

{

return -sinl(x);

} /\* f \*/

long double fd(long double x)

{

return -cosl(x);

} /\* fd \*/

long double fdd(long double x)

{

return sinl(x);

} /\* fdd \*/

int main ()

{

long double x;

x = newton(fd, fdd, 1.0, 0.0000001);

printf("Solution to min f(x) , x = %Lf, f(%Lf) = %Lf, fd(%Lf) = %Lf\n",

x, x, f(x), x, fd(x));

return 0;

} /\* main \*/

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

C:\> newton2.exe

Solution to min f(x), x = 1.570796, f(1.570796) = -1.000000, fd(1.570796) = 0.000000

C:\>

התוכנית הבאה משתמשת בשיטת המיתר למציאת המינימום של –sinx:

/\* secant\_op1.c - use scant method for optimization \*/

#include <stdio.h>

#include <math.h>

long double secant(long double (\*fun)(long double), long double x1,

long double x0, long double eps)

{

long double x=0, f1, f0;

int i;

do {

f1 = (\*fun)(x1);

f0 = (\*fun)(x0);

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

x0 = x1;

x1 = x;

} while (fabsl(f1) > eps);

return x;

} /\* secant \*/

long double f(long double x)

{

return -sinl(x);

} /\* f \*/

long double fd(long double x)

{

return -cosl(x);

} /\* fd \*/

int main ()

{

long double x;

x = secant(fd, 1.0, 2.000, 0.0000001);

printf("Solution to min f(x), x = %Lf, f(%Lf) = %Lf, fd(%Lf) = %Lf\n",

x, x, f(x), x, fd(x));

return 0;

} /\* main \*/

פלט ריצה:

E:\> secant\_op1.exe

Solution to min f(x), x = 1.570796, f(1.570796) = -1.000000, fd(1.570796) = 0.000000

E:\>

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

/\* quad.c \*/

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

long double quad(long double (\*fp)(long double), long double x1, long double x3, long double eps)

{

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

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

long double oldx4;

x2 = (x1 + x3)/2;

fx1 = (\*fp)(x1);

fx2 = (\*fp)(x2);

fx3 = (\*fp)(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 = (\*fp)(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 ( fabs(oldx4 - x4) > eps);

return x4;

} /\* quad \*/

long double f(long double x)

{

return -sin(x);

} /\* f \*/

int main()

{

long double x;

x = quad(f, 1.0, 2.0, 0.0001);

printf("Min solution x = %Lf, min value = %Lf\n", x, f(x));

} /\* main \*/

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

C:\> quad.exe

Min solution x = 1.570796, min value = -1.000000

C:\>