**נספח**

התוכנית הבאה מבצעת אינטרפולציה לינארית על נקודות של הפונקציה

f(x) = x2

ומשווה תוצאות לערך האמיתי. הפעם הדבר נעשה לפי שיטת ניוטון.

גם הפעם האינטרפולציה היא מדויקת ונקבל את אותם תוצאות כמו בשיטת לגרנז.

/\* ni1.java - implement newton interpolation \*/

class Point

{

public double x;

public double fx;

} // Point

class Ni

{

int n; /\* no\_of\_points \*/

Point point\_arr[];

double [] coeffs;

void compute\_coeffs()

{

int i, j;

double a[], x[], y[], x\_sub;

a = new double [n+1];

x = new double [n+1];

y = new double [n+1];

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

{

x[i] = point\_arr[i].x;

y[i] = point\_arr[i].fx;

} /\* for \*/

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

{

a[i]=y[i];

x\_sub = 1;

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

{

a[i]= a[i] - a[j]\*x\_sub;

x\_sub = x\_sub\*(x[i]-x[j]);

} /\* for \*/

a[i] = a[i]/x\_sub;

} /\* for \*/

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

coeffs[i] = a[i];

} /\* compute\_coeffs \*/

double interpolate(double x)

{

int i, j;

double t1, ni;

ni = 1.0;

t1 = 0.0;

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

{

t1 = t1 + coeffs[i]\*ni;

ni = ni\*(x - point\_arr[i].x);

} /\* for \*/

return t1;

} /\* interpolate \*/

} // Ni

public class ni1

{

static double my\_sqr( double x)

{

return x\*x;

} /\* my\_sqr \*/

static public void main(String args[])

{

int i;

Ni ni\_v = new Ni();

double x;

ni\_v.n = 7;

ni\_v.point\_arr = new Point [8];

ni\_v.coeffs = new double [8];

for(i=0; i < 8; i++)

ni\_v.point\_arr[i] = new Point();

ni\_v.point\_arr[0].x = 0.0;

ni\_v.point\_arr[0].fx = 0.0;

ni\_v.point\_arr[1].x = 1.0;

ni\_v.point\_arr[1].fx = 1.0;

ni\_v.point\_arr[2].x = 2.0;

ni\_v.point\_arr[2].fx = 4.0;

ni\_v.point\_arr[3].x = 3.0;

ni\_v.point\_arr[3].fx = 9.0;

ni\_v.point\_arr[4].x = 4.0;

ni\_v.point\_arr[4].fx = 16.0;

ni\_v.point\_arr[5].x = 5.0;

ni\_v.point\_arr[5].fx = 25.0;

ni\_v.point\_arr[6].x = 6.0;

ni\_v.point\_arr[6].fx = 36.0;

ni\_v.point\_arr[7].x = 7.0;

ni\_v.point\_arr[7].fx = 49.0;

ni\_v.compute\_coeffs();

x = 1.5;

System.out.println("x = " + x + " real value = " +

my\_sqr(x) + " interpolation = " +

ni\_v.interpolate(x));

x = 2.5;

System.out.println("x = " + x + " real value = " +

my\_sqr(x) + " interpolation = " +

ni\_v.interpolate(x));

x = 3.5;

System.out.println("x = " + x + " real value = " +

my\_sqr(x) + " interpolation = " +

ni\_v.interpolate(x));

x = 4.5;

System.out.println("x = " + x + " real value = " +

my\_sqr(x) + " interpolation = " +

ni\_v.interpolate(x));

x = 5.5;

System.out.println("x = " + x + " real value = " +

my\_sqr(x) + " interpolation = " +

ni\_v.interpolate(x));

x = 5.5;

System.out.println("x = " + x + " real value = " +

my\_sqr(x) + " interpolation = " +

ni\_v.interpolate(x));

x = 6.5;

System.out.println("x = " + x + " real value = " +

my\_sqr(x) + " interpolation = " +

ni\_v.interpolate(x));

x = 6.5;

System.out.println("x = " + x + " real value = " +

my\_sqr(x) + " interpolation = " +

ni\_v.interpolate(x));

x = 3.9;

System.out.println("x = " + x + " real value = " +

my\_sqr(x) + " interpolation = " +

ni\_v.interpolate(x));

x = 3.1;

System.out.println("x = " + x + " real value = " +

my\_sqr(x) + " interpolation = " +

ni\_v.interpolate(x));

} /\* main \*/

} // ni1

פלט ריצה:

x = 1.5 real value = 2.25 interpolation = 2.25

x = 2.5 real value = 6.25 interpolation = 6.25

x = 3.5 real value = 12.25 interpolation = 12.25

x = 4.5 real value = 20.25 interpolation = 20.25

x = 5.5 real value = 30.25 interpolation = 30.25

x = 5.5 real value = 30.25 interpolation = 30.25

x = 6.5 real value = 42.25 interpolation = 42.25

x = 6.5 real value = 42.25 interpolation = 42.25

x = 3.9 real value = 15.209999999999999 interpolation = 15.209999999999999

x = 3.1 real value = 9.610000000000001 interpolation = 9.610000000000001