**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.c - implement newton interpolation \*/

#include <stdio.h>

#include <math.h>

typedef struct point

{

 long double x;

 long double fx;

} POINT, \*POINT\_PTR;

typedef struct ni\_rec

{

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

 POINT\_PTR point\_arr;

 long double \*coeffs;

} NI\_REC, \*NI\_REC\_PTR;

void compute\_coeffs(NI\_REC\_PTR ni\_r)

{

 int i, j, n;

 long double \*a, \*x, \*y, x\_sub;

 n = (ni\_r->n)+1;

 a = (long double \*)malloc(n\*sizeof(long double));

 x = (long double \*)malloc(n\*sizeof(long double));

 y = (long double \*)malloc(n\*sizeof(long double));

 n--;

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

 {

 x[i] = ni\_r->point\_arr[i].x;

 y[i] = ni\_r->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++)

 (ni\_r->coeffs)[i] = a[i];

} /\* compute\_coeffs \*/

long double interpolate(NI\_REC\_PTR ni\_r, long double x)

{

 int i, j, n;

 long double t1, ni;

 n = ni\_r->n;

 ni = 1.0;

 t1 = 0.0;

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

 {

 t1 = t1 + ni\_r->coeffs[i]\*ni;

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

 } /\* for \*/

 return t1;

} /\* interpolate \*/

long double my\_sqr(long double x)

{

 return x\*x;

} /\* my\_sqr \*/

int main()

{

 int i;

 NI\_REC ni\_r;

 long double x;

 ni\_r.n = 7;

 ni\_r.point\_arr = (POINT\_PTR)malloc(8\*sizeof(POINT));

 ni\_r.coeffs = (long double \*)malloc(8\*sizeof(long double));

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

 compute\_coeffs(&ni\_r);

 x = 1.5;

 printf("x = %Lf, real value = %Lf, interpolation = %Lf\n",

 x, my\_sqr(x), interpolate(&ni\_r, x));

 x = 2.5;

 printf("x = %Lf, real value = %Lf, interpolation = %Lf\n",

 x, my\_sqr(x), interpolate(&ni\_r, x));

 x = 3.5;

 printf("x = %Lf, real value = %Lf, interpolation = %Lf\n",

 x, my\_sqr(x), interpolate(&ni\_r, x));

 x = 4.5;

 printf("x = %Lf, real value = %Lf, interpolation = %Lf\n",

 x, my\_sqr(x), interpolate(&ni\_r, x));

 x = 5.5;

 printf("x = %Lf, real value = %Lf, interpolation = %Lf\n",

 x, my\_sqr(x), interpolate(&ni\_r, x));

 x = 5.5;

 printf("x = %Lf, real value = %Lf, interpolation = %Lf\n",

 x, my\_sqr(x), interpolate(&ni\_r, x));

 x = 6.5;

 printf("x = %Lf, real value = %Lf, interpolation = %Lf\n",

 x, my\_sqr(x), interpolate(&ni\_r, x));

 x = 6.5;

 printf("x = %Lf, real value = %Lf, interpolation = %Lf\n",

 x, my\_sqr(x), interpolate(&ni\_r, x));

 x = 3.9;

 printf("x = %Lf, real value = %Lf, interpolation = %Lf\n",

 x, my\_sqr(x), interpolate(&ni\_r, x));

 x = 3.1;

 printf("x = %Lf, real value = %Lf, interpolation = %Lf\n",

 x, my\_sqr(x), interpolate(&ni\_r, x));

} /\* main \*/

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

C:\> ni1.exe

x = 1.500000, real value = 2.250000, interpolation = 2.250000

x = 2.500000, real value = 6.250000, interpolation = 6.250000

x = 3.500000, real value = 12.250000, interpolation = 12.250000

x = 4.500000, real value = 20.250000, interpolation = 20.250000

x = 5.500000, real value = 30.250000, interpolation = 30.250000

x = 5.500000, real value = 30.250000, interpolation = 30.250000

x = 6.500000, real value = 42.250000, interpolation = 42.250000

x = 6.500000, real value = 42.250000, interpolation = 42.250000

x = 3.900000, real value = 15.210000, interpolation = 15.210000

x = 3.100000, real value = 9.610000, interpolation = 9.610000

C:\>

התוכנית הבאה עושה אינטרפולציה ל-sin(x) בתחום 1.7 – 0.0 ברדיאנים.

התוצאות זהות לאלו שבשיטת לגרנז.

/\* ni2.c - implement newton interpolation \*/

#include <stdio.h>

#include <math.h>

typedef struct point

{

 long double x;

 long double fx;

} POINT, \*POINT\_PTR;

typedef struct ni\_rec

{

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

 POINT\_PTR point\_arr;

 long double \*coeffs;

} NI\_REC, \*NI\_REC\_PTR;

void compute\_coeffs(NI\_REC\_PTR ni\_r)

{

 int i, j, n;

 long double \*a, \*x, \*y, x\_sub;

 n = (ni\_r->n)+1;

 a = (long double \*)malloc(n\*sizeof(long double));

 x = (long double \*)malloc(n\*sizeof(long double));

 y = (long double \*)malloc(n\*sizeof(long double));

 n--;

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

 {

 x[i] = ni\_r->point\_arr[i].x;

 y[i] = ni\_r->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++)

 (ni\_r->coeffs)[i] = a[i];

} /\* compute\_coeffs \*/

long double interpolate(NI\_REC\_PTR ni\_r, long double x)

{

 int i, j, n;

 long double t1, ni;

 n = ni\_r->n;

 ni = 1.0;

 t1 = 0.0;

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

 {

 t1 = t1 + ni\_r->coeffs[i]\*ni;

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

 } /\* for \*/

 return t1;

} /\* interpolate \*/

int main()

{

 int i;

 NI\_REC ni\_r;

 long double x;

 ni\_r.n = 8;

 ni\_r.point\_arr = (POINT\_PTR)malloc(9\*sizeof(POINT));

 ni\_r.coeffs = (long double \*)malloc(9\*sizeof(long double));

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

 ni\_r.point\_arr[0].fx = sinl(0.0);

 ni\_r.point\_arr[1].x = 0.2;

 ni\_r.point\_arr[1].fx = sinl(0.2);

 ni\_r.point\_arr[2].x = 0.4;

 ni\_r.point\_arr[2].fx = sinl(0.4);

 ni\_r.point\_arr[3].x = 0.6;

 ni\_r.point\_arr[3].fx = sinl(0.6);

 ni\_r.point\_arr[4].x = 0.8;

 ni\_r.point\_arr[4].fx = sinl(0.8);

 ni\_r.point\_arr[5].x = 1.0;

 ni\_r.point\_arr[5].fx = sinl(1.0);

 ni\_r.point\_arr[6].x = 1.2;

 ni\_r.point\_arr[6].fx = sinl(1.2);

 ni\_r.point\_arr[7].x = 1.6;

 ni\_r.point\_arr[7].fx = sinl(1.6);

 ni\_r.point\_arr[8].x = 1.8;

 ni\_r.point\_arr[8].fx = sinl(1.8);

 compute\_coeffs(&ni\_r);

 x = 0.1;

 printf("x = %13.10Lf, real value = %13.10Lf, interpolation = %13.10Lf\n",

 x, sinl(x), interpolate(&ni\_r, x));

 x = 0.3;

 printf("x = %13.10Lf, real value = %13.10Lf, interpolation = %13.10Lf\n",

 x, sinl(x), interpolate(&ni\_r, x));

 x = 0.5;

 printf("x = %13.10Lf, real value = %13.10Lf, interpolation = %13.10Lf\n",

 x, sinl(x), interpolate(&ni\_r, x));

 x = 0.7;

 printf("x = %13.10Lf, real value = %13.10Lf, interpolation = %13.10Lf\n",

 x, sinl(x), interpolate(&ni\_r, x));

 x = 0.9;

 printf("x = %13.10Lf, real value = %13.10Lf, interpolation = %13.10Lf\n",

 x, sinl(x), interpolate(&ni\_r, x));

 x = 1.1;

 printf("x = %13.10Lf, real value = %13.10Lf, interpolation = %13.10Lf\n",

 x, sinl(x), interpolate(&ni\_r, x));

 x = 1.3;

 printf("x = %13.10Lf, real value = %13.10Lf, interpolation = %13.10Lf\n",

 x, sinl(x), interpolate(&ni\_r, x));

 x = 1.5;

 printf("x = %13.10Lf, real value = %13.10Lf, interpolation = %13.10Lf\n",

 x, sinl(x), interpolate(&ni\_r, x));

 x = 1.7;

 printf("x = %13.10Lf, real value = %13.10Lf, interpolation = %13.10Lf\n",

 x, sinl(x), interpolate(&ni\_r, x));

} /\* main \*/

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

C:\> ni2.exe

x = 0.1000000000, real value = 0.0998334166, interpolation = 0.0998334115

x = 0.3000000000, real value = 0.2955202067, interpolation = 0.2955202077

x = 0.5000000000, real value = 0.4794255386, interpolation = 0.4794255382

x = 0.7000000000, real value = 0.6442176872, interpolation = 0.6442176875

x = 0.9000000000, real value = 0.7833269096, interpolation = 0.7833269093

x = 1.1000000000, real value = 0.8912073601, interpolation = 0.8912073607

x = 1.3000000000, real value = 0.9635581854, interpolation = 0.9635581820

x = 1.5000000000, real value = 0.9974949866, interpolation = 0.9974949765

x = 1.7000000000, real value = 0.9916648105, interpolation = 0.9916648291

C:\>