**נספח**

התוכנית הבאה מבצעת אינטרפולציה לינארית על נקודות של הפונקציה

 f(x) = x2

ומשווה תוצאות לערך האמיתי:

/\* li.c - implement linear interpolation \*/

typedef struct point

{

 long double x;

 long double fx;

} POINT, \*POINT\_PTR;

typedef struct li\_rec

{

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

 POINT\_PTR point\_arr;

} LI\_REC, \*LI\_REC\_PTR;

int bin\_search(LI\_REC\_PTR li\_r, long double x)

{

 int low, high, mid;

 low = 0;

 high = li\_r->n-1;

while(1)

{

 mid = (low+high)/2;

 if (li\_r->point\_arr[mid].x == x)

 return mid;

 if (li\_r->point\_arr[mid+1].x == x)

 return mid+1;

 if ( (li\_r->point\_arr[mid].x < x) &&

 (li\_r->point\_arr[mid+1].x > x))

 return mid;

 if( li\_r->point\_arr[mid+1].x < x )

 low = mid+1;

 else /\* li\_r->point\_arr[mid].x > x \*/

 high = mid-1;

} /\* while \*/

} /\* bin\_search \*/

long double interpolate(LI\_REC\_PTR li\_rec, long double x)

{

 int i;

 long double t;

 i = bin\_search(li\_rec, x);

 t = (li\_rec->point\_arr[i+1].fx - li\_rec->point\_arr[i].fx);

 t = t/(li\_rec->point\_arr[i+1].x - li\_rec->point\_arr[i].x);

 t = t\*(x - li\_rec->point\_arr[i].x);

 t = li\_rec->point\_arr[i].fx + t;

 return t;

} /\* interpolate \*/

long double my\_sqr(long double x)

{

 return x\*x;

} /\* my\_sqr \*/

int main()

{

 LI\_REC li\_r;

 long double x;

 li\_r.n = 8;

 li\_r.point\_arr = (POINT\_PTR)malloc(8\*sizeof(LI\_REC));

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

 x = 1.5;

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

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

 x = 2.5;

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

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

 x = 3.5;

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

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

 x = 4.5;

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

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

 x = 5.5;

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

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

 x = 5.5;

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

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

 x = 6.5;

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

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

 x = 6.5;

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

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

 x = 3.9;

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

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

 x = 3.1;

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

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

} /\* main \*/

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

C;\> li.exe

x = 1.500000, real value = 2.250000, interpolation = = 2.500000

x = 2.500000, real value = 6.250000, interpolation = = 6.500000

x = 3.500000, real value = 12.250000, interpolation = 12.500000

x = 4.500000, real value = 20.250000, interpolation = 20.500000

x = 5.500000, real value = 30.250000, interpolation = 30.500000

x = 5.500000, real value = 30.250000, interpolation = 30.500000

x = 6.500000, real value = 42.250000, interpolation = 42.500000

x = 6.500000, real value = 42.250000, interpolation = 42.500000

x = 3.900000, real value = 15.210000, interpolation = 15.300000

x = 3.100000, real value = 9.610000, interpolation = 9.700000

C:\>

התוכנית הבאה מבצעת אינטרפולציה לגרנז על נקודות של הפונקציה

 f(x) = x2

ומשווה תוצאות לערך האמיתי. מכיוון ש f(x) הוא פולינום, האינטרפולציה היא **מדויקת**:

/\* lagrange.c - implement lagrange interpolation \*/

#include <stdio.h>

#include <math.h>

typedef struct point

{

 long double x;

 long double fx;

} POINT, \*POINT\_PTR;

typedef struct li\_rec

{

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

 POINT\_PTR point\_arr;

} LI\_REC, \*LI\_REC\_PTR;

long double interpolate(LI\_REC\_PTR li\_rec, long double x)

{

 int i, j, n;

 long double t1, t2;

 n = li\_rec->n;

 t1 = 0.0;

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

 {

 t2 = li\_rec->point\_arr[i].fx;

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

 if(i != j)

 {

 t2 = t2\*(x - li\_rec->point\_arr[j].x);

 t2 = t2/(li\_rec->point\_arr[i].x - li\_rec->point\_arr[j].x);

 } /\* if \*/

 t1 = t1 + t2;

 } /\* for \*/

 return t1;

} /\* interpolate \*/

long double my\_sqr(long double x)

{

 return x\*x;

} /\* my\_sqr \*/

int main()

{

 LI\_REC li\_r;

 long double x;

 li\_r.n = 7;

 li\_r.point\_arr = (POINT\_PTR)malloc(8\*sizeof(LI\_REC));

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

 x = 1.5;

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

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

 x = 2.5;

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

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

 x = 3.5;

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

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

 x = 4.5;

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

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

 x = 5.5;

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

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

 x = 5.5;

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

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

 x = 6.5;

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

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

 x = 6.5;

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

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

 x = 3.9;

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

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

 x = 3.1;

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

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

} /\* main \*/

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

C:\>lagrange.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 מבצעת אינטרפולציה לגרנז על נקודות של הפונקציה

 f(x) = sin(x)

ומשווה תוצאות לערך האמיתי.

כפי שאנחנו רואים, הדיוק של האינטרפולציה איננו רע.

/\* lagrange1.c - implement lagrange interpolation \*/

#include <stdio.h>

#include <math.h>

typedef struct point

{

 long double x;

 long double fx;

} POINT, \*POINT\_PTR;

typedef struct li\_rec

{

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

 POINT\_PTR point\_arr;

} LI\_REC, \*LI\_REC\_PTR;

long double interpolate(LI\_REC\_PTR li\_rec, long double x)

{

 int i, j, n;

 long double t1, t2;

 n = li\_rec->n;

 t1 = 0.0;

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

 {

 t2 = li\_rec->point\_arr[i].fx;

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

 if(i != j)

 {

 t2 = t2\*(x - li\_rec->point\_arr[j].x);

 t2 = t2/(li\_rec->point\_arr[i].x - li\_rec->point\_arr[j].x);

 } /\* if \*/

 t1 = t1 + t2;

 } /\* for \*/

 return t1;

} /\* interpolate \*/

int main()

{

 LI\_REC li\_r;

 long double x;

 li\_r.n = 8;

 li\_r.point\_arr = (POINT\_PTR)malloc(9\*sizeof(LI\_REC));

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

 x = 0.1;

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

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

 x = 0.3;

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

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

 x = 0.5;

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

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

 x = 0.7;

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

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

 x = 0.9;

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

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

 x = 1.1;

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

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

 x = 1.3;

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

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

 x = 1.5;

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

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

 x = 1.7;

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

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

} /\* main \*/

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

C:\> lagrange1.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:\>