**נספח**

התוכנית הבאה מבצעת אינטרפולציה פולינומיאלית על פונקצית רונגה בקטע (-1,1) כאשר הצמתים נבחרים במרווחים שווים כלומר:

xi = -1 +(i-1)2/n, i=1, 2, …, n+1

ההמשך הוא לפי גישת ניוטון לאינטרפולציה.

/\* runge1.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 f(long double x)

{

return 1.0/(1.0+25.0\*x\*x);

} /\* f \*/

long double pi = 3.1415926535898;

void compute\_runge\_nodes(int n, long double runge\_nodes[])

{

int i;

for (i=1; i <= n+1; i++)

runge\_nodes[i-1] = -1 + (i-1)\*2.0/(long double)n;

} /\* compute\_nodes \*/

int main()

{

int i, n;

NI\_REC ni\_r;

long double x;

long double \*runge\_nodes;

n = 8;

ni\_r.n = n-1;

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

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

runge\_nodes = (long double \*)malloc(n\*sizeof(long double));

compute\_runge\_nodes(n-1, runge\_nodes);

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

{

ni\_r.point\_arr[i].x = runge\_nodes[i];

ni\_r.point\_arr[i].fx = f(runge\_nodes[i]);

} /\* for \*/

compute\_coeffs(&ni\_r);

x = -0.9;

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

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

x =runge\_nodes[2];

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

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

x = -0.5;

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

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

x = -0.3;

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

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

x = runge\_nodes[3];

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

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

x = runge\_nodes[4];

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

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

x = 0.3;

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

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

x = 0.5;

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

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

x = runge\_nodes[6];

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

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

x = 0.9;

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

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

} /\* main \*/

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

C:\> runge1.exe

x = -0.900000, real value = 0.047059, interpolation = 0.218516

x = -0.428571, real value = 0.178832, interpolation = 0.178832

x = -0.500000, real value = 0.137931, interpolation = 0.084914

x = -0.300000, real value = 0.307692, interpolation = 0.406896

x = -0.142857, real value = 0.662162, interpolation = 0.662162

x = 0.142857, real value = 0.662162, interpolation = 0.662162

x = 0.300000, real value = 0.307692, interpolation = 0.406896

x = 0.500000, real value = 0.137931, interpolation = 0.084914

x = 0.714286, real value = 0.072700, interpolation = 0.072700

x = 0.900000, real value = 0.047059, interpolation = 0.218516

C:\>

התוכנית הבאה מיישמת את האינטרפולציה שבה נקודות ה-x נבחרים לפי שורשי צביצב בקטע הסטנדרטי (-1,1):

xci= cos((2i-1)π/2n), i =1, 2, …,n

/\* cheby1.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 f(long double x)

{

return 1/(1.0+25.0\*x\*x);

} /\* f \*/

long double pi = 3.1415926535898;

void compute\_cheby\_nodes(int n, long double cheb\_nodes[])

{

int i;

for (i=1; i <= n; i++)

cheb\_nodes[i-1] = cosl((2\*i-1)\*pi/(2\*n));

} /\* compute\_nodes \*/

int main()

{

int i, n;

NI\_REC ni\_r;

long double x;

long double \*cheb\_nodes, reg\_nodes;

n = 8;

ni\_r.n = n-1;

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

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

cheb\_nodes = (long double \*)malloc(n\*sizeof(long double));

compute\_cheby\_nodes(n, cheb\_nodes);

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

{

ni\_r.point\_arr[i].x = cheb\_nodes[i];

ni\_r.point\_arr[i].fx = f(cheb\_nodes[i]);

} /\* for \*/

compute\_coeffs(&ni\_r);

x = -0.9;

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

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

x =cheb\_nodes[6];

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

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

x = -0.5;

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

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

x = -0.3;

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

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

x = cheb\_nodes[4];

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

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

x = cheb\_nodes[3];

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

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

x = 0.3;

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

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

x = 0.5;

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

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

x = cheb\_nodes[2];

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

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

x = 0.9;

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

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

} /\* main \*/

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

C:\> cheby1.exe

x = -0.900000, real value = 0.047059, interpolation = 0.063523

x = -0.831470, real value = 0.054694, interpolation = 0.054694

x = -0.500000, real value = 0.137931, interpolation = 0.164948

x = -0.300000, real value = 0.307692, interpolation = 0.399568

x = -0.195090, real value = 0.512425, interpolation = 0.512425

x = 0.195090, real value = 0.512425, interpolation = 0.512425

x = 0.300000, real value = 0.307692, interpolation = 0.399568

x = 0.500000, real value = 0.137931, interpolation = 0.164948

x = 0.555570, real value = 0.114726, interpolation = 0.114726

x = 0.900000, real value = 0.047059, interpolation = 0.063523

C:\>

התוכנית הבאה מיישמת את שיטת צומתי צביצב על פני **קטע (a,b) כלשהו**:

/\* cheby2.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 f(long double x)

{

return 1/(1.0+25.0\*x\*x);

} /\* f \*/

long double pi = 3.1415926535898;

void compute\_cheby\_nodes(int n, long double a,

long double b, long double cheb\_nodes[])

{

int i;

for (i=1; i <= n; i = i++)

cheb\_nodes[i-1] = 0.5\*(a+b)+

0.5\*(b-a)\*cosl((2.0\*i-1.0)\*pi/(2.0\*n));

} /\* compute\_nodes \*/

int main()

{

int i, n;

NI\_REC ni\_r;

long double x;

long double \*cheb\_nodes, reg\_nodes;

n = 8;

ni\_r.n = n-1;

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

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

cheb\_nodes = (long double \*)malloc(n\*sizeof(long double));

compute\_cheby\_nodes(n, -3.0, 2.0,cheb\_nodes);

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

{

ni\_r.point\_arr[i].x = cheb\_nodes[i];

ni\_r.point\_arr[i].fx = f(cheb\_nodes[i]);

} /\* for \*/

compute\_coeffs(&ni\_r);

x = -2.9;

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

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

x =cheb\_nodes[6];

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

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

x = -0.5;

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

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

x = -1.3;

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

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

x = cheb\_nodes[4];

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

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

x = cheb\_nodes[3];

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

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

x = 1.3;

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

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

x = 1.7;

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

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

x = cheb\_nodes[2];

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

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

x = 1.9;

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

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

} /\* main \*/

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

C:\> cheby2.exe

x = -2.900000, real value = 0.004734, interpolation = -0.058504

x = -2.578674, real value = 0.005979, interpolation = 0.005979

x = -0.500000, real value = 0.137931, interpolation = 0.641169

x = -1.300000, real value = 0.023121, interpolation = -0.162736

x = -0.987726, real value = 0.039385, interpolation = 0.039385

x = -0.012274, real value = 0.996248, interpolation = 0.996248

x = 1.300000, real value = 0.023121, interpolation = -0.186741

x = 1.700000, real value = 0.013652, interpolation = 0.125908

x = 0.888926, real value = 0.048182, interpolation = 0.048182

x = 1.900000, real value = 0.010959, interpolation = 0.105869

C:\>