**נספח**

התוכנית הבאה מבצעת spline ריבועי על sin(x). כפי שאנחנו רואים הדיוק הוא של 2-3 ספרות משמעותיות בלבד, אבל זה לא כל כך רע על פולינום בדרגה כה נמוכה. בהחלט מדגימה את הפוטנציאל של spline.

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

#include <stdio.h>

#include <math.h>

typedef struct point

{

long double x;

long double fx;

} POINT, \*POINT\_PTR;

typedef struct si\_rec

{

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

POINT\_PTR point\_arr;

long double \*z;

} SI\_REC, \*SI\_REC\_PTR;

void compute\_z(SI\_REC\_PTR si\_r, long double z0)

{

long double zi, t;

int i;

si\_r->z[0] = z0;

zi = z0;

for(i=1; i <= si\_r->n; i++)

{

t = 2\*(si\_r->point\_arr[i].fx - si\_r->point\_arr[i-1].fx);

t = t/(si\_r->point\_arr[i].x -si\_r->point\_arr[i-1].x);

zi = -zi + t;

si\_r->z[i] = zi;

}

} /\* compute\_z \*/

long double interpolate(SI\_REC\_PTR si\_r, long double x)

{

int i, j, n;

long double t1, t2, a, b, c;

n = si\_r->n;

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

if (x >= si\_r->point\_arr[i].x && x < si\_r->point\_arr[i+1].x)

break;

b = si\_r->z[i];

c = si\_r->point\_arr[i].fx;

a = si\_r->z[i+1] - si\_r->z[i];

a = a/(2\*(si\_r->point\_arr[i+1].x - si\_r->point\_arr[i].x));

t1 = x - si\_r->point\_arr[i].x;

t2 = c + b\*t1 + a\*t1\*t1;

return t2;

} /\* interpolate \*/

int main()

{

int i;

SI\_REC si\_r;

long double x;

si\_r.n = 8;

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

si\_r.z = (long double \*)malloc(9\*sizeof(long double));

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

compute\_z(&si\_r, 1.0);

x = 0.1;

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

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

x = 0.3;

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

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

x = 0.5;

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

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

x = 0.7;

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

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

x = 0.9;

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

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

x = 1.1;

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

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

x = 1.3;

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

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

x = 1.5;

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

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

x = 1.7;

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

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

} /\* main \*/

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

C:\> spline.exe

x = 0.1000000000, real value = 0.0998334166, interpolation = 0.0996673327

x = 0.3000000000, real value = 0.2955202067, interpolation = 0.2956912491

x = 0.5000000000, real value = 0.4794255386, interpolation = 0.4792642154

x = 0.7000000000, real value = 0.6442176872, interpolation = 0.6443931030

x = 0.9000000000, real value = 0.7833269096, interpolation = 0.7831693979

x = 1.1000000000, real value = 0.8912073601, interpolation = 0.8913858735

x = 1.3000000000, real value = 0.9635581854, interpolation = 0.9632767741

x = 1.5000000000, real value = 0.9974949866, interpolation = 0.9970440326

x = 1.7000000000, real value = 0.9916648105, interpolation = 0.9920145521

C:\>

הגרסה הבאה של אותה תוכנית מדפיסה את ערכי ה-z והמקדמים של הפולינומים:

/\* spline1.c - implement spline interpolation \*/

#include <stdio.h>

#include <math.h>

typedef struct point

{

long double x;

long double fx;

} POINT, \*POINT\_PTR;

typedef struct si\_rec

{

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

POINT\_PTR point\_arr;

long double \*z;

} SI\_REC, \*SI\_REC\_PTR;

void compute\_z(SI\_REC\_PTR si\_r, long double z0)

{

long double zi, t;

int i;

si\_r->z[0] = z0;

zi = z0;

for(i=1; i <= si\_r->n; i++)

{

printf("z%d = %Lf\n",i-1, zi);

t = 2\*(si\_r->point\_arr[i].fx - si\_r->point\_arr[i-1].fx);

t = t/(si\_r->point\_arr[i].x -si\_r->point\_arr[i-1].x);

zi = -zi + t;

si\_r->z[i] = zi;

}

printf("z%d = %Lf\n",i-1, zi);

} /\* compute\_z \*/

long double interpolate(SI\_REC\_PTR si\_r, long double x)

{

int i, j, n;

long double t1, t2, a, b, c;

n = si\_r->n;

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

if (x >= si\_r->point\_arr[i].x && x < si\_r->point\_arr[i+1].x)

break;

b = si\_r->z[i];

c = si\_r->point\_arr[i].fx;

a = si\_r->z[i+1] - si\_r->z[i];

a = a/(2\*(si\_r->point\_arr[i+1].x - si\_r->point\_arr[i].x));

t1 = x - si\_r->point\_arr[i].x;

t2 = c + b\*t1 + a\*t1\*t1;

printf("a = %Lf, b = %Lf,c = %Lf, t1 = %Lf, t2 = %Lf\n",

a, b, c, t1, t2);

return t2;

} /\* interpolate \*/

int main()

{

int i;

SI\_REC si\_r;

long double x;

si\_r.n = 8;

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

si\_r.z = (long double \*)malloc(9\*sizeof(long double));

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

compute\_z(&si\_r, 1.0);

x = 0.1;

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

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

x = 0.3;

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

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

x = 0.5;

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

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

x = 0.7;

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

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

x = 0.9;

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

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

x = 1.1;

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

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

x = 1.3;

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

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

x = 1.5;

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

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

x = 1.7;

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

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

} /\* main \*/

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

C:\> spline1.exe

z0 = 1.000000

z1 = 0.986693

z2 = 0.920797

z3 = 0.831445

z4 = 0.695692

z5 = 0.545457

z6 = 0.360224

z7 = -0.022551

z8 = -0.234709

a = -0.033267, b = 1.000000,c = 0.000000, t1 = 0.100000, t2 = 0.099667

x = 0.1000000000, real value = 0.0998334166, interpolation = 0.0996673327

a = -0.164741, b = 0.986693,c = 0.198669, t1 = 0.100000, t2 = 0.295691

x = 0.3000000000, real value = 0.2955202067, interpolation = 0.2956912491

a = -0.223381, b = 0.920797,c = 0.389418, t1 = 0.100000, t2 = 0.479264

x = 0.5000000000, real value = 0.4794255386, interpolation = 0.4792642154

a = -0.339382, b = 0.831445,c = 0.564642, t1 = 0.100000, t2 = 0.644393

x = 0.7000000000, real value = 0.6442176872, interpolation = 0.6443931030

a = -0.375586, b = 0.695692,c = 0.717356, t1 = 0.100000, t2 = 0.783169

x = 0.9000000000, real value = 0.7833269096, interpolation = 0.7831693979

a = -0.463084, b = 0.545457,c = 0.841471, t1 = 0.100000, t2 = 0.891386

x = 1.1000000000, real value = 0.8912073601, interpolation = 0.8913858735

a = -0.478469, b = 0.360224,c = 0.932039, t1 = 0.100000, t2 = 0.963277

x = 1.3000000000, real value = 0.9635581854, interpolation = 0.9632767741

a = -0.478469, b = 0.360224,c = 0.932039, t1 = 0.300000, t2 = 0.997044

x = 1.5000000000, real value = 0.9974949866, interpolation = 0.9970440326

a = -0.530394, b = -0.022551,c = 0.999574, t1 = 0.100000, t2 = 0.992015

x = 1.7000000000, real value = 0.9916648105, interpolation = 0.9920145521

C:\>

התוכנית הבאה מממשת spline ריבועי ל-f(x) = x2 כולל הדפסות ביניים. כפי שאנחנו רואים, החישוב הוא מדויק.

/\* spline3.c - implement spline interpolation \*/

#include <stdio.h>

#include <math.h>

typedef struct point

{

long double x;

long double fx;

} POINT, \*POINT\_PTR;

typedef struct si\_rec

{

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

POINT\_PTR point\_arr;

long double \*z;

} SI\_REC, \*SI\_REC\_PTR;

void compute\_z(SI\_REC\_PTR si\_r, long double z0)

{

long double zi, t;

int i;

si\_r->z[0] = z0;

zi = z0;

for(i=1; i <= si\_r->n; i++)

{

printf("z%d = %Lf\n",i-1, zi);

t = 2\*(si\_r->point\_arr[i].fx - si\_r->point\_arr[i-1].fx);

t = t/(si\_r->point\_arr[i].x -si\_r->point\_arr[i-1].x);

zi = -zi + t;

si\_r->z[i] = zi;

}

printf("z%d = %Lf\n",i-1, zi);

} /\* compute\_z \*/

long double interpolate(SI\_REC\_PTR si\_r, long double x)

{

int i, j, n;

long double t1, t2, a, b, c;

n = si\_r->n;

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

if (x >= si\_r->point\_arr[i].x && x < si\_r->point\_arr[i+1].x)

break;

b = si\_r->z[i];

c = si\_r->point\_arr[i].fx;

a = si\_r->z[i+1] - si\_r->z[i];

a = a/(2\*(si\_r->point\_arr[i+1].x - si\_r->point\_arr[i].x));

t1 = x - si\_r->point\_arr[i].x;

t2 = c + b\*t1 + a\*t1\*t1;

printf("a = %Lf, b = %Lf,c = %Lf, t1 = %Lf, t2 = %Lf\n",

a, b, c, t1, t2);

return t2;

} /\* interpolate \*/

int main()

{

int i;

SI\_REC si\_r;

long double x;

si\_r.n = 8;

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

si\_r.z = (long double \*)malloc(9\*sizeof(long double));

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

si\_r.point\_arr[8].x = 8.0;

si\_r.point\_arr[8].fx = 64.0;

compute\_z(&si\_r, 0.0);

x = 0.5;

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

x, x\*x, interpolate(&si\_r, x));

x = 1.5;

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

x, x\*x, interpolate(&si\_r, x));

x = 2.4;

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

x, x\*x, interpolate(&si\_r, x));

x = 3.2;

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

x, x\*x, interpolate(&si\_r, x));

x = 4.9;

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

x, x\*x, interpolate(&si\_r, x));

x = 5.6;

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

x, x\*x, interpolate(&si\_r, x));

x = 6.3;

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

x, x\*x, interpolate(&si\_r, x));

x = 7.7;

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

x, x\*x, interpolate(&si\_r, x));

x = 8.1;

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

x, x\*x, interpolate(&si\_r, x));

} /\* main \*/

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

C:\> spline3.exe

z0 = 0.000000

z1 = 2.000000

z2 = 4.000000

z3 = 6.000000

z4 = 8.000000

z5 = 10.000000

z6 = 12.000000

z7 = 14.000000

z8 = 16.000000

a = 1.000000, b = 0.000000,c = 0.000000, t1 = 0.500000, t2 = 0.250000

x = 0.5000000000, real value = 0.2500000000, interpolation = 0.2500000000

a = 1.000000, b = 2.000000,c = 1.000000, t1 = 0.500000, t2 = 2.250000

x = 1.5000000000, real value = 2.2500000000, interpolation = 2.2500000000

a = 1.000000, b = 4.000000,c = 4.000000, t1 = 0.400000, t2 = 5.760000

x = 2.4000000000, real value = 5.7600000000, interpolation = 5.7600000000

a = 1.000000, b = 6.000000,c = 9.000000, t1 = 0.200000, t2 = 10.240000

x = 3.2000000000, real value = 10.2400000000, interpolation = 10.2400000000

a = 1.000000, b = 8.000000,c = 16.000000, t1 = 0.900000, t2 = 24.010000

x = 4.9000000000, real value = 24.0100000000, interpolation = 24.0100000000

a = 1.000000, b = 10.000000,c = 25.000000, t1 = 0.600000, t2 = 31.360000

x = 5.6000000000, real value = 31.3600000000, interpolation = 31.3600000000

a = 1.000000, b = 12.000000,c = 36.000000, t1 = 0.300000, t2 = 39.690000

x = 6.3000000000, real value = 39.6900000000, interpolation = 39.6900000000

a = 1.000000, b = 14.000000,c = 49.000000, t1 = 0.700000, t2 = 59.290000

x = 7.7000000000, real value = 59.2900000000, interpolation = 59.2900000000

a = 1.000000, b = 16.000000,c = 64.000000, t1 = 0.100000, t2 = 65.610000

x = 8.1000000000, real value = 65.6100000000, interpolation = 65.6100000000

C:\>

התוכנית הבאה מבצעת spline על פונקצית runge:

/\* spline4.c - implement spline interpolation

on the runge function \*/

#include <stdio.h>

#include <math.h>

typedef struct point

{

long double x;

long double fx;

} POINT, \*POINT\_PTR;

typedef struct si\_rec

{

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

POINT\_PTR point\_arr;

long double \*z;

} SI\_REC, \*SI\_REC\_PTR;

void compute\_z(SI\_REC\_PTR si\_r, long double z0)

{

long double zi, t;

int i;

si\_r->z[0] = z0;

zi = z0;

for(i=1; i <= si\_r->n; i++)

{

t = 2\*(si\_r->point\_arr[i].fx - si\_r->point\_arr[i-1].fx);

t = t/(si\_r->point\_arr[i].x -si\_r->point\_arr[i-1].x);

zi = -zi + t;

si\_r->z[i] = zi;

}

} /\* compute\_z \*/

long double interpolate(SI\_REC\_PTR si\_r, long double x)

{

int i, j, n;

long double t1, t2, a, b, c;

n = si\_r->n;

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

if (x >= si\_r->point\_arr[i].x && x < si\_r->point\_arr[i+1].x)

break;

b = si\_r->z[i];

c = si\_r->point\_arr[i].fx;

a = si\_r->z[i+1] - si\_r->z[i];

a = a/(2\*(si\_r->point\_arr[i+1].x - si\_r->point\_arr[i].x));

t1 = x - si\_r->point\_arr[i].x;

t2 = c + b\*t1 + a\*t1\*t1;

return t2;

} /\* interpolate \*/

long double f(long double x)

{

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

} /\* f \*/

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;

SI\_REC si\_r;

long double x;

long double \*runge\_nodes;

n = 8;

si\_r.n = n-1;

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

si\_r.z = (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++)

{

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

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

} /\* for \*/

compute\_z(&si\_r, 0.0);

x = -0.9;

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

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

x =runge\_nodes[2];

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

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

x = -0.5;

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

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

x = -0.3;

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

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

x = runge\_nodes[3];

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

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

x = runge\_nodes[4];

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

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

x = 0.3;

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

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

x = 0.5;

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

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

x = runge\_nodes[6];

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

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

x = 0.9;

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

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

} /\* main \*/

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

C:\> spline4.exe

x = -0.900000, real value = 0.047059, interpolation = 0.042656

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

x = -0.500000, real value = 0.137931, interpolation = 0.145239

x = -0.300000, real value = 0.307692, interpolation = 0.312294

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 = 0.312294

x = 0.500000, real value = 0.137931, interpolation = 0.145239

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

x = 0.900000, real value = 0.047059, interpolation = 0.042656

C:\>