Answers to Homework 7: Approximation: Polynomial Approximation

1. (a) Is the collection of functions $\phi_1(x) = 1$, $\phi_2(x) = x$, and $\phi_3(x) = \sin x$ orthogonal with respect to the inner product

$$< f, g >= \int_{-\pi}^{\pi} f(x) g(x) dx \quad ? \tag{1}$$

If not, find the corresponding orthogonal functions using the Gram-Schmidt orthogonalization process

$$\hat{\phi}_1(x) = \phi_1(x) ; \tag{2}$$

$$\hat{\phi}_k(x) = \phi_k(x) - \sum_{i=1}^{k-1} \frac{<\phi_k, \hat{\phi}_i>}{<\hat{\phi}_i, \hat{\phi}_i>} \hat{\phi}_i(x) , \quad k = 2, 3, \ldots \tag{3}$$

Answer: These functions are not orthogonal, because

$$\begin{aligned} <\phi_2, \phi_3> &= \int_{-\pi}^{\pi} x \sin x \, dx = - x \cos x |_{-\pi}^{\pi} + \int_{-\pi}^{\pi} \cos x \, dx \\ &= (\sin x - x \cos x)|_{-\pi}^{\pi} = -2\pi \cos \pi = 2\pi \neq 0. \end{aligned}$$

Using the Gram-Schmidt orthogonalization process, we get

$$\hat{\phi}_1(x) = 1 ;$$

$$\hat{\phi}_2(x) = x - \frac{\int_{-\pi}^{\pi} x \, dx}{\int_{-\pi}^{\pi} dx} = x ;$$

$$\hat{\phi}_3(x) = \sin x - \frac{\int_{-\pi}^{\pi} sinx \, dx}{\int_{-\pi}^{\pi} dx} - \frac{\int_{-\pi}^{\pi} x \, sinx \, dx}{\int_{-\pi}^{\pi} x^2 dx} x = \sin x - \frac{2\pi}{\frac{2}{3}\pi^3} x = \sin x - \frac{3x}{\pi^2} .$$

(b) Using the Gram-Schmidt process, find the first three orthogonal polynomials with respect to the inner product

$$< f, g >= \int_0^\infty w(x)\, f(x) g(x) dx \,, \tag{4}$$

where $w(x) = e^{-a\,x}$ $(a > 0)$.

Hint: Use the equality (for integer n)

$$\int_0^\infty x^n\, e^{-a\,x}\, dx = \frac{n!}{a^{n+1}} \,.$$

Answer:

$$\begin{aligned}
\hat{\phi}_1(x) &= 1\,; \\
\hat{\phi}_2(x) &= x - \frac{\int_0^\infty w(x) x\, dx}{\int_0^\infty dx} = x - \frac{1}{a}\,; \\
\hat{\phi}_3(x) &= x^2 - \frac{\int_0^\infty w(x) x^2\, dx}{\int_0^\infty dx} - \frac{\int_0^\infty w(x) x^2 \left(x - \frac{1}{a}\right) dx}{\int_0^\infty \left(x - \frac{1}{a}\right)^2 dx} \left(x - \frac{1}{a}\right) \\
&= x^2 - \frac{2}{a^2} - \frac{\frac{6-2}{a^4}}{\frac{2-2+1}{a^3}} \left(x - \frac{1}{a}\right) = x^2 - \frac{4x}{a} + \frac{2}{a^2} \,.
\end{aligned}$$

2. (a) Prove that the constant function $f(x) = a$ that fits inconsistent measurements $f_1, f_2, \ldots, f_n$ in the least-squares sense corresponds to the mean value

$$a = \frac{1}{n} \sum_{k=1}^{n} f_k \,. \tag{5}$$

Answer:

We need to minimize the least-squares norm

$$F(a) = \sum_{k=1}^{n} (a - f_k)^2 \,.$$

Differentiating with respect to a leads to the linear equation

$$F'(a) = \sum_{k=1}^{n} (a - f_k) = 0 \,,$$

whose solution is

$$a = \frac{\sum_{k=1}^{n} f_k}{\sum_{k=1}^{n} 1} = \frac{\sum_{k=1}^{n} f_k}{n} \,.$$

(b) Prove that, if the measurements $f_1, f_2, \ldots, f_n$ are taken at the integer values $x_k = k$, $k = 1, 2, \ldots, n$, the linear function $f(x) = a\,x + b$ that fits the data in the least-squares sense corresponds to the values

$$a = \frac{6}{n(n^2-1)}\left[2\sum_{k=1}^{n} k\,f_k - (n+1)\sum_{k=1}^{n} f_k\right]; \tag{6}$$

$$b = \frac{2}{n(n-1)}\left[(2n+1)\sum_{k=1}^{n} f_k - 3\sum_{k=1}^{n} k\,f_k\right]. \tag{7}$$

Answer:

We are minimizing the least-squares norm

$$F(a,b) = \sum_{k=1}^{n}(a\,k + b - f_k)^2\,.$$

Differentiating with respect to a and b leads to the system of two linear equations

$$\begin{aligned}
\frac{\partial F}{\partial a} &= \sum_{k=1}^{n} k\,(a\,k + b - f_k) = a\sum_{k=1}^{n} k^2 + b\sum_{k=1}^{n} k - \sum_{k=1}^{n} k\,f_k \\
&= a\,\frac{n(n+1)(2n+1)}{6} + b\,\frac{n(n+1)}{2} - \sum_{k=1}^{n} k\,f_k = 0\,, \\
\frac{\partial F}{\partial b} &= \sum_{k=1}^{n}(a\,k + b - f_k) = a\sum_{k=1}^{n} k + b\sum_{k=1}^{n} 1 - \sum_{k=1}^{n} f_k \\
&= a\,\frac{n(n+1)}{2} + b\,n - \sum_{k=1}^{n} f_k = 0\,.
\end{aligned}$$

In the matrix form, these equations are

$$\begin{bmatrix} \dfrac{n(n+1)(2n+1)}{6} & \dfrac{n(n+1)}{2} \\[2ex] \dfrac{n(n+1)}{2} & n \end{bmatrix}\begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} \displaystyle\sum_{k=1}^{n} k\,f_k \\[2ex] \displaystyle\sum_{k=1}^{n} f_k \end{bmatrix}.$$

The determinant of the system matrix is

$$n\,\frac{n(n+1)(2n+1)}{6} - \left(\frac{n(n+1)}{2}\right)^2 = \frac{n^2(n^2-1)}{12}\,,$$

and the solution of the system is

$$\begin{aligned}
a &= \frac{12}{n^2(n^2-1)}\left[n\sum_{k=1}^{n} k\,f_k - \frac{n(n+1)}{2}\sum_{k=1}^{n} f_k\right] = \frac{6}{n(n^2-1)}\left[2\sum_{k=1}^{n} k\,f_k - (n+1)\sum_{k=1}^{n} f_k\right]; \\
b &= \frac{12}{n^2(n^2-1)}\left[\frac{n(n+1)(2n+1)}{6}\sum_{k=1}^{n} f_k - \frac{n(n+1)}{2}\sum_{k=1}^{n} k\,f_k\right] \\
&= \frac{2}{n(n-1)}\left[(2n+1)\sum_{k=1}^{n} f_k - 3\sum_{k=1}^{n} k\,f_k\right].
\end{aligned}$$

3. Chebyshev polynomials $T_k(x)$ can be defined by the recursive relationship

$$\begin{aligned} T_0(x) &= 1 && (8)\\ T_1(x) &= x && (9)\\ T_{k+1}(x) &= 2x\,T_k(x) - T_{k-1}(x)\,, \quad k = 1,2,\ldots && (10) \end{aligned}$$

One can evaluate the Chebyshev polynomial representation

$$f(x) = \sum_{k=0}^{n} c_k\,T_k(x) \tag{11}$$

efficiently with the algorithm

CHEBYSHEV SUM$(x, c_0, c_1, \ldots, c_n)$
1 $\hat{c}_1 \leftarrow 0$
2 $\hat{c}_0 \leftarrow c_n$
3 **for** $k \leftarrow n-1, n-2, \ldots, 0$
4 **do**
5 $\quad t \leftarrow \hat{c}_1$
6 $\quad \hat{c}_1 \leftarrow \hat{c}_0$
7 $\quad \hat{c}_0 \leftarrow c_k + 2x\,\hat{c}_0 - t$
8 **return** $(\hat{c}_0 - x\,\hat{c}_1)$

Design an analogous algorithm for the *Hermite* polynomial representation

$$f(x) = \sum_{k=0}^{n} c_k\,H_k(x)\,. \tag{12}$$

Hermite polynomials $H_k(x)$ satisfy the recursive relationship

$$\begin{aligned} H_0(x) &= 1 && (13)\\ H_1(x) &= 2x && (14)\\ H_{k+1}(x) &= 2x\,H_k(x) - 2k\,H_{k-1}(x)\,, \quad k = 1,2,\ldots && (15) \end{aligned}$$

Answer:

Define a family of coefficients $\hat{c}_k(x)$ such that $\hat{c}_{n+1} = 0$, $\hat{c}_{n+2} = 0$, and

$$c_k = \hat{c}_k(x) - 2x\,\hat{c}_{k+1}(x) + 2(k+1)\hat{c}_{k+2}(x)\,, \quad k = 0,1,\ldots,n$$

Then

$$f(x) = \sum_{k=0}^{n} c_k\,H_k(x) = \sum_{k=0}^{n} \hat{c}_k\,H_k(x) - 2x\sum_{k=0}^{n} \hat{c}_{k+1}\,H_k(x) + 2\sum_{k=0}^{n}(k+1)\hat{c}_{k+2}\,H_k(x)\,.$$

Shifting the indeces in the last two sums, we obtain

$$\begin{aligned} f(x) &= \sum_{k=0}^{n} c_k\, H_k(x) = \sum_{k=0}^{n} \hat{c}_k\, H_k(x) - 2x \sum_{k=1}^{n} \hat{c}_k\, H_{k-1}(x) + 2 \sum_{k=2}^{n} (k-1)\, \hat{c}_k\, H_{k-2}(x) \\ &= \hat{c}_0\, H_0(x) + \hat{c}_1\, H_1(x) - 2x\, \hat{c}_1\, H_0(x) + \sum_{k=2}^{n} \hat{c}_k \left[H_k(x) - 2x\, H_{k-1}(x) + 2(k-1)\, H_{k-2}(x) \right] . \end{aligned}$$

The last term is zero because of the recursive property of Hermite polynomials. Finally,

$$f(x) = \hat{c}_0\, H_0(x) + \hat{c}_1\, H_1(x) - 2x\, \hat{c}_1\, H_0(x) = \hat{c}_0 + 2x\, \hat{c}_1 - 2x\, \hat{c}_1 = \hat{c}_0 .$$

The recursion for computing $\hat{c}_0$ is

$$\begin{aligned} \hat{c}_{n+1} &= 0 \\ \hat{c}_n &= c_n \\ \hat{c}_k &= c_k + 2x\, \hat{c}_{k+1} - 2(k+1)\, \hat{c}_{k+2} \qquad k = n-1, n-2, \ldots, 0 , \end{aligned}$$

which results in the following algorithm:

HERMITE SUM$(x, c_0, c_1, \ldots, c_n)$
1 $\hat{c}_1 \leftarrow 0$
2 $\hat{c}_0 \leftarrow c_n$
3 **for** $k \leftarrow n-1, n-2, \ldots, 0$
4 **do**
5 $\quad t \leftarrow \hat{c}_1$
6 $\quad \hat{c}_1 \leftarrow \hat{c}_0$
7 $\quad \hat{c}_0 \leftarrow c_k + 2x\, \hat{c}_0 - 2(k+1)t$
8 **return** $\hat{c}_0$

Note: Hermite polynomials are orthogonal with respect to the inner product

$$< f, g >= \int_{-\infty}^{\infty} e^{-x^2}\, f(x)\, g(x)\, dx .$$

4. (Programming)

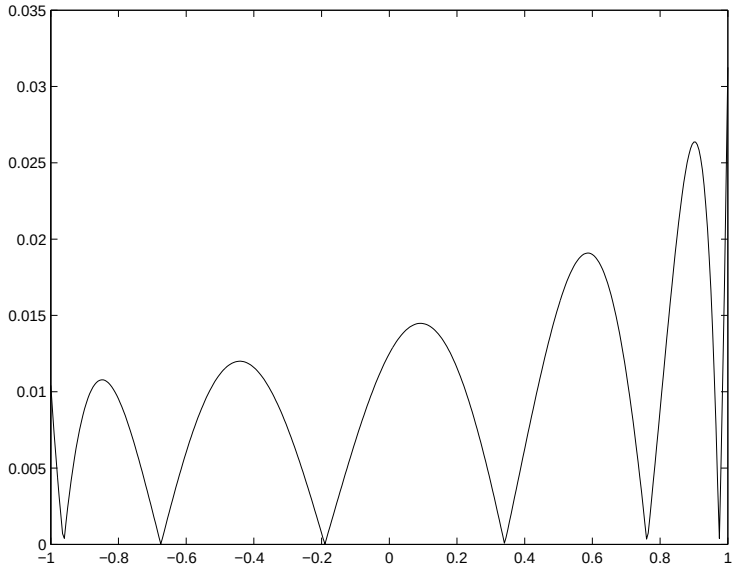
   (a) Write a program for evaluating Chebyshev polynomial representation using the algorithm above. Test your program by approximating the infinite sum

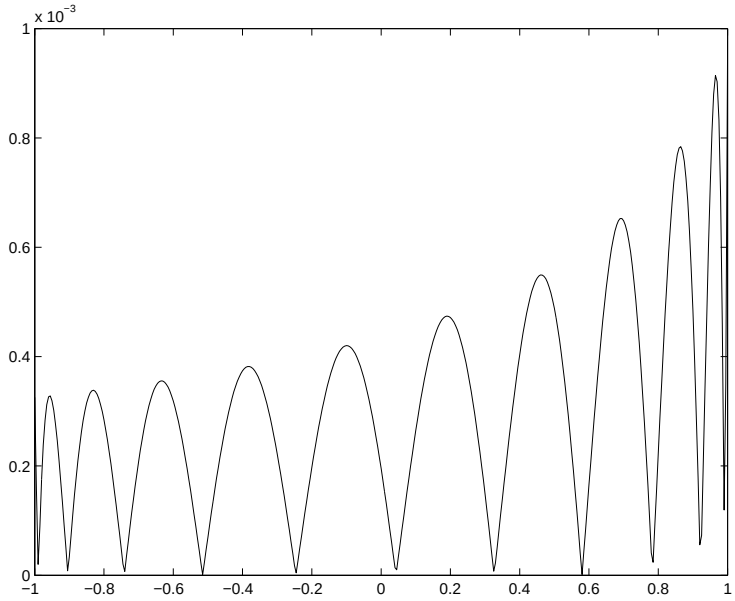
$$\frac{1 - t\,x}{1 - 2t\,x + t^2} = \sum_{k=0}^{\infty} t^k\, T_k(x) \tag{16}$$

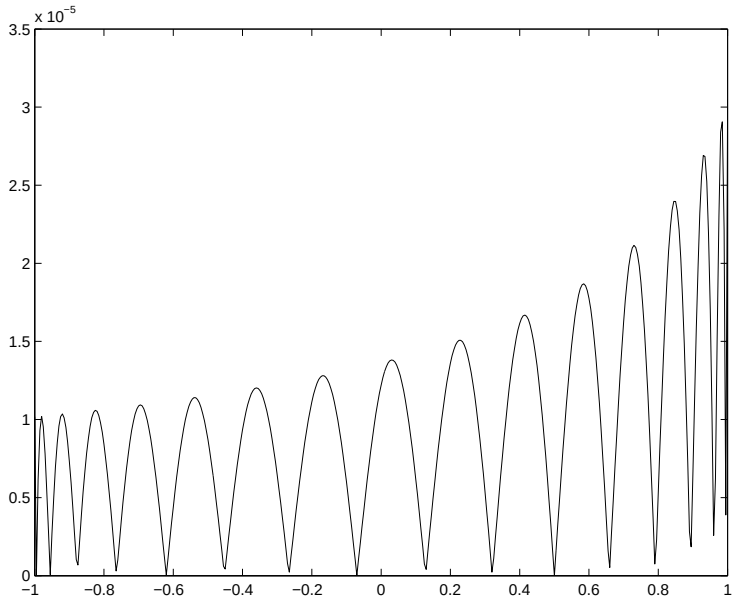
   with the finite sum

$$\sum_{k=0}^{n} t^k\, T_k(x) . \tag{17}$$

   Plot (or tabulate) the absolute error on the interval $-1 \le x \le 1$ for $t = 1/2$ and $n = 5, 10, 15$.
   Answer:


0.035
0.03
0.025
0.02
0.015
0.01
0.005
0
−1
−0.8
−0.6
−0.4
−0.2
0
0.2
0.4
0.6
0.8
1


x 10^−3
1
0.8
0.6
0.4
0.2
0
−1
−0.8
−0.6
−0.4
−0.2
0
0.2
0.4
0.6
0.8
1


x 10^−5
3.5
3
2.5
2
1.5
1
0.5
0
−1
−0.8
−0.6
−0.4
−0.2
0
0.2
0.4
0.6
0.8
1

Solution:

C program:

```
#include <stdlib.h> /* for allocation */
#include <stdio.h>  /* for output */
#include <assert.h> /* for assertion  */
#include <math.h>   /* for math functions */

/* Function: chebyshev_sum
   ----------------------
   Evaluates Chebyshev polynomial representation
   f(x) = sum(k=0,n-1) C_k*T_k(x)
   n    - number of coefficients
   x    - where to evaluate
   c[n] - coefficients
*/
double chebyshev_sum (int n, double x, double c[])
{
    int k;
    double t, c0, c1;

    c1 = 0;
    c0 = c[n-1];
    for (k=n-2; k >= 0; k--) {
        t = c1;
        c1 = c0;
        c0 = c[k] + 2.*x*c0 - t;
    }

    return (c0 - x*c1);
}

int main (void) {
    int i, k, n[] = {5,10,15}, nx=401;
    const double t=0.5;
    double *c, *x, *f, e;

    x = (double*) malloc (nx*sizeof(double));
    f = (double*) malloc (nx*sizeof(double));
    assert (x != NULL && f != NULL);

    for (k=0; k < nx; k++) {
        /* regular grid for plotting */
        x[k] = -1. + 2.*k/(nx-1.);
        /* true function values */
        f[k] = (1.-t*x[k])/(1.-2.*t*x[k] + t*t);
    }

    c = (double*) malloc ((n[2]+1)*sizeof(double));
    assert (c != NULL);

    /* chebyshev coefficients c_k = t^k */
    for (e=1., k=0; k <= n[2]; k++, e *= t) {
        c[k] = e;
    }

    /* three cases */
    for (i=0; i < 3; i++) {
        for (k=0; k < nx; k++) {
```

```
            /* absolute error */
            e = fabs(f[k] - chebyshev_sum (n[i]+1, x[k], c));
            /* output table */
            printf("%d %f %g\n",k,x[k],e);
        }
    }

    exit (0);
}
```

(b) Write a program for evaluating Hermite polynomial representation using your algorithm from problem 3. Test your program by approximating the infinite sum

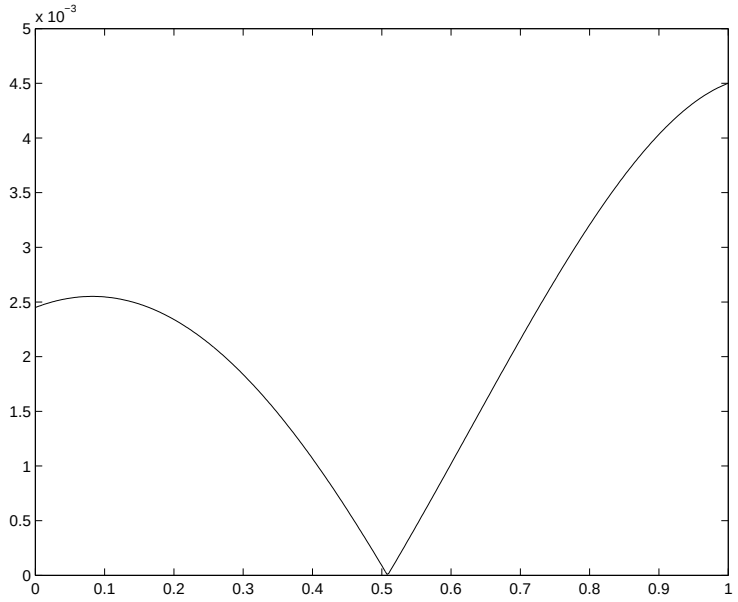
$$e^{t(2x-t)} = \sum_{k=0}^{\infty} \frac{t^k}{k!} H_k(x) \tag{18}$$

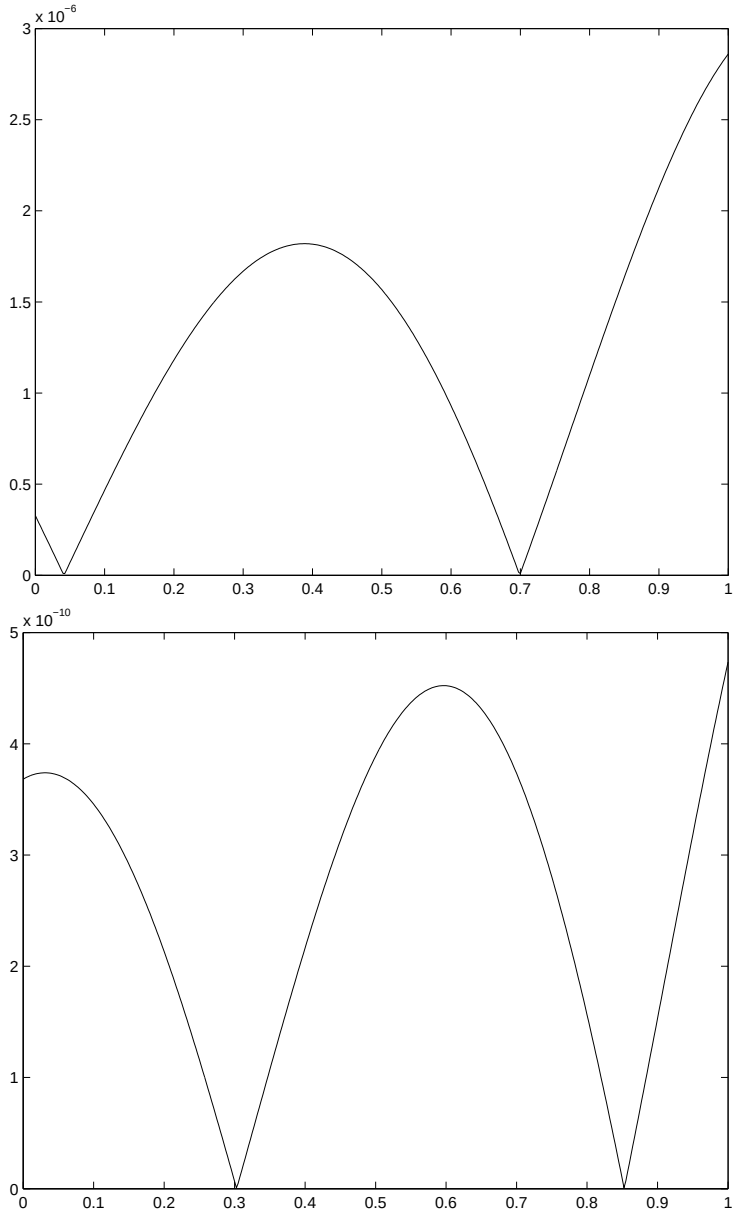
with the finite sum

$$\sum_{k=0}^{n} \frac{t^k}{k!} H_k(x)\,. \tag{19}$$

Plot (or tabulate) the absolute error on the interval $0 \le x \le 1$ for $t = 1/2$ and $n = 5, 10, 15$.

Answer:





Solution:

C program:

```
#include <stdlib.h> /* for allocation */
#include <stdio.h>  /* for output */
#include <assert.h> /* for assertion  */
#include <math.h>   /* for math functions */

/* Function: hermite_sum
   ---------------------
   Evaluates Hermite polynomial representation
   f(x) = sum(k=0,n-1) C_k*H_k(x)
   n    - number of coefficients
   x    - where to evaluate
   c[n] - coefficients
*/
double hermite_sum (int n, double x, double c[])
```

```
{
    int k;
    double t, c0, c1;

    c1 = 0;
    c0 = c[n-1];
    for (k=n-2; k >= 0; k--) {
        t = c1;
        c1 = c0;
        c0 = c[k] + 2.*(x*c0 - (k+1)*t);
    }

    return c0;
}

int main (void) {
    int i, k, n[] = {5,10,15}, nx=401;
    const double t=0.5;
    double *c, *x, *f, e;

    x = (double*) malloc (nx*sizeof(double));
    f = (double*) malloc (nx*sizeof(double));
    assert (x != NULL && f != NULL);

    for (k=0; k < nx; k++) {
        /* regular grid for plotting */
        x[k] = k/(nx-1.);
        /* true function values */
        f[k] = exp(t*(2.*x[k]-t));
    }

    c = (double*) malloc ((n[2]+1)*sizeof(double));
    assert (c != NULL);

    /* hermite coefficients c_k = t^k/k! */
    for (e=1., k=0; k <= n[2]; k++, e *= (t/k)) {
        c[k] = e;
    }

    /* three cases */
    for (i=0; i < 3; i++) {
        for (k=0; k < nx; k++) {
            /* absolute error */
            e = fabs(f[k] - hermite_sum (n[i]+1, x[k], c));
            /* output table */
            printf("%d %f %g\n",k,x[k],e);
        }
    }

    exit (0);
}
```

5. (Programming) The following table contains the number of medals won by the United States at the winter Olympic games:

Year	Location	Gold	Silver	Bronze	Total	Points
1924	CHAMONIX	1	2	1	4	8
1928	SAINT MORITZ	3	2	2	7	15
1932	LAKE PLACID	6	4	2	12	28
1936	GARMISH PARTENKIRCHEN	1	0	3	4	6
1948	SAINT MORITZ	3	5	2	10	21
1952	OSLO	4	6	1	11	25
1956	CORTINA D'AMPEZZO	2	3	2	7	14
1960	SQUAW VALLEY	3	4	3	10	20
1964	INNSBRUCK	1	2	4	7	11
1968	GRENOBLE	1	4	1	6	12
1972	SAPPORO	3	2	3	8	16
1976	INNSBRUCK	3	3	4	10	19
1980	LAKE PLACID	6	4	2	12	28
1984	SARAJEVO	4	4	0	8	20
1988	CALGARY	2	1	3	6	11
1992	ALBERTVILLE	5	4	2	11	25
1994	LILLEHAMMER	6	5	2	13	30
1998	NAGANO	6	3	4	13	28
2002	SALT LAKE CITY	10	13	11	34	67

The points are computed with the formula

$$\text{Points} = 3 \times \text{Gold} + 2 \times \text{Silver} + \text{Bronze}\,.$$

Using the method of least squares, find linear trends of the form $f(x) = a + b\,x$ for the functions

(a) Points(Year)

(b) Points(Gold)

In each case, find a, b and the Olympic games with the largest and smallest least-squares errors.

Answer:

$$\text{Points(Year)} \approx -532.218037 + 0.281527\,\text{Year}$$

The smallest least-squares error (0.181477) is in 1960 games. The largest error (1267.494312) is in 2002.

$$\text{Points(Gold)} \approx 1.736067 + 5.300210\,\text{Gold}$$

The smallest least-squares error (0.928761) is in 1924 games. The largest error (150.352466) is in 2002.

For both "dependencies", the 2002 games appear to be the least predictable.

Solution:

C program:

```
#include <stdio.h> /* for output */

/* Function: least_squares
   ----------------------
   Finds straight-line least-squares fit
   f(x) ~= a[0] + a[1]*x
   to a table of f(x)
   n    - number of points
   x[n] - table of variable
   f[n] - table of function
   a[2] - output line coefficients
*/
void least_squares(int n, const double *x, const double *f, double *a)
{
    int k;
    double sx, sx2, sf, sxf, det;

    sx=sx2=sf=sxf=0.;
    for (k=0; k < n; k++) {
        sx  += x[k];
        sx2 += x[k]*x[k];
        sf  += f[k];
        sxf += x[k]*f[k];
    }
    det = sx2*n - sx*sx;

    a[0] = (sx2*sf - sx*sxf)/det;
    a[1] = ( n*sxf - sx*sf )/det;
}

/* number of nodes */
#define NODES 19

/* main program */
int main (void)
{
  const int n = NODES;
  int k;
  double years[] = { 1924.,1928.,1932.,1936.,1948.,1952.,1956.,1960.,1964.,
                     1968.,1972.,1976.,1980.,1984.,1988.,1992.,1994.,1998.,
                     2002.};
  double points[] = { 8., 15.,28.,6. ,21.,25.,14.,20.,11.,
                      12.,16.,19.,28.,20.,11.,25.,30.,28.,
                      67.};
  double golds[] = { 1.,3.,6.,1.,3.,4.,2.,3.,1.,
     1.,3.,3.,6.,4.,2.,5.,6.,6.,
                     10.};
  double a[2], e;

  least_squares(n, years, points, a);
  printf("a=%f b=%f\n",a[0],a[1]);
  for (k=0; k < n; k++) {
      /* error */
      e = points[k] - a[0] - a[1]*years[k];
      /* output table */
```

```
        printf("%d \t %d \t %f \t %f\n",k,(int) years[k],e,e*e);
    }

    least_squares(n, golds, points, a);
    printf("a=%f b=%f\n",a[0],a[1]);
    for (k=0; k < n; k++) {
        /* error*/
        e = points[k] - a[0] - a[1]*golds[k];
        /* output table */
        printf("%d \t %d \t %f \t %f\n",k,(int) years[k],e,e*e);
    }

    return 0.;
}
```