Department of Mathematics and Statistics, University of Helsinki
Numerical methods and the C language, fall 2010

Workshop 1, solutions in C++/NR
Mon 13.9. at 16-18 B322

Exercise 1

The formula to calculate a Celsius wind chill is:

$$T(wc) = 0.045(5.27V^{0.5} + 10.45 - 0.28V)(T - 33) + 33$$

Where: $T(wc)$ = the wind chill, V = the wind speed in kilometers per hour and, T = the temperature in degrees Celsius. Write a program to compute the wind chill. *Hint.* Use the program `h1p011.c(pp)` on the www-page as a starting point.

Solution

```
// FILE: h011.cpp begins.

#include <iostream>
#include <cmath>

using namespace std;

double WindChill(double T, double V)
{
  return 0.045*(5.27*pow(V,0.5) + 10.45 - 0.28*V)* (T - 33.0) + 33.0;
}

void Prompt_TV()
{
  double T,V;
  cout<<"Enter temperature in Celsius:" <<endl;
  cin>>T;
  cout<<"Enter wind speed in m/s:" <<endl;
  cin>>V;
  cout<<"For T = "<<T<<", V = "<<V;
  cout<<", wind chill is: "<< WindChill(T,V*3.6)<<endl;
}

int main()
{
  Prompt_TV();
}

// FILE: h011.cpp ends.

```

Exercise 2

Use the function in problem 1 to print the values of wind chill factor for the wind speeds $2 * jm/s, j = 0, 1, 2, 3, 4$ and temperatures $10 - j * 5, j = 0, 1, 2, 3, 4$ in the following format

```
0   10  5  0  -5 -10
2   ....
4   ....
6   ....
8   ....
```

Hint. You may compare the results with a table the www-page h012.eps.

Solution

```
// FILE: h012.cpp begins.

#include <iostream>
#include <cmath>

using namespace std;

double WindChill(double T, double V)
{
  return  0.045*(5.27*pow(V,0.5) + 10.45 - 0.28*V)* (T - 33.0) + 33.0;
}

int main()
{
  double T,V;
  printf("%5d ", 0);
  for(int k = 0; k < 5; k++)
    printf("%5d ", (int)(10.0-k*5));
  for(int j = 1; j < 6; j++)
    for (int i = 0; i < 6; i++)
    {
      T = 10.0-(i-1)*5.0;
      V = j*2.0;
      double WC = WindChill(T, V*3.6);
      if(i==0)
        printf("\n% 5d ",(int)V);
      else
        printf("% 5d ", (int)WC);
    }
  printf("\n");
}
```

```

// FILE: h012.cpp ends.

```

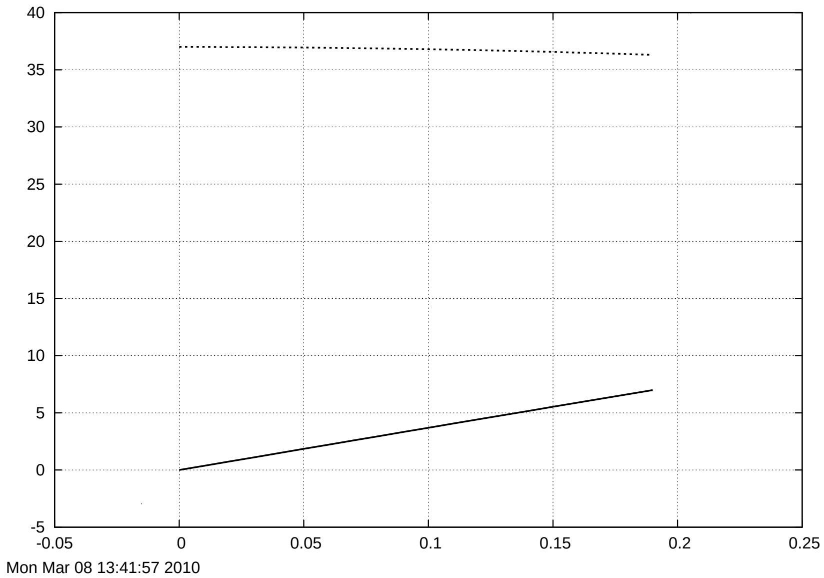
Exercise 3

The file h013.dat on the www-page contains 21 (x, y)-pairs, one pair per line. Use this data to numerically approximate dy/dx and write the approximations, 20 $(x, y'(x))$-pairs, on the screen or into a file.

Solution

```
// FILE: h013.cpp begins

#include <iostream>
#include <cmath>
#include "nr.h"
#include "plot.h"
#include "matutl02.h"

using namespace std;

int main()
{
  Mat_DP data = getmat("h013.dat");
  double d;
  FILE *fp1,*fp2;
  int n=data.nrows();

  fp1 = fopen("h013a.dat", "w");
  fp2 = fopen("h013b.dat", "w");

  for(int i=0; i < n-1; i++)
  {
    d = (data[i+1][1]-data[i][1]) / (data[i+1][0]-data[i][0]);
    fprintf(fp1,"%25.16e %25.16e\n", data[i][0], data[i][1]);
    fprintf(fp2,"%25.16e %25.16e\n", data[i][0], d);
  }

  fclose(fp1);
  fclose(fp2);

  setplotprint(1);
  plot("h013a.dat", "r-3", "h013b.dat", "b-3", NULL);
  system("rm h013a.dat h013b.dat plot.cmd mnmx.dat");
  system("mv plot.ps h013.ps");
```

```
}

// FILE: h013.cpp ends

```



Exercise 4

The following table gives the euro exchange rate in US dollars at 6 consecutive Mondays. Use this information to fit a least-squares line $ax + b = y$ to the data $(x_i, y_i), i = 1, \ldots, 6$, where $x_i = i$ is the ordinal of the given date and y_i the corresponding exchange rate. Use vectors to store the data.

Table 1: Average exchange rates, 2001

Date	22.10.	29.10.	5.11.	12.11.	19.11.	26.11.
1 EUR in USD	0.8969	0.9005	0.8961	0.8919	0.8793	0.8818

Hint: Generally, for $(x_i, y_i), i = 1, \ldots, n$, the formulas of the coefficients a and b are

$$a = \frac{\sum x_i y_i - \frac{1}{n} \sum x_i \sum y_i}{\sum (x_i - \bar{x})^2}, \quad b = \frac{\sum y_i - a \sum x_i}{n},$$

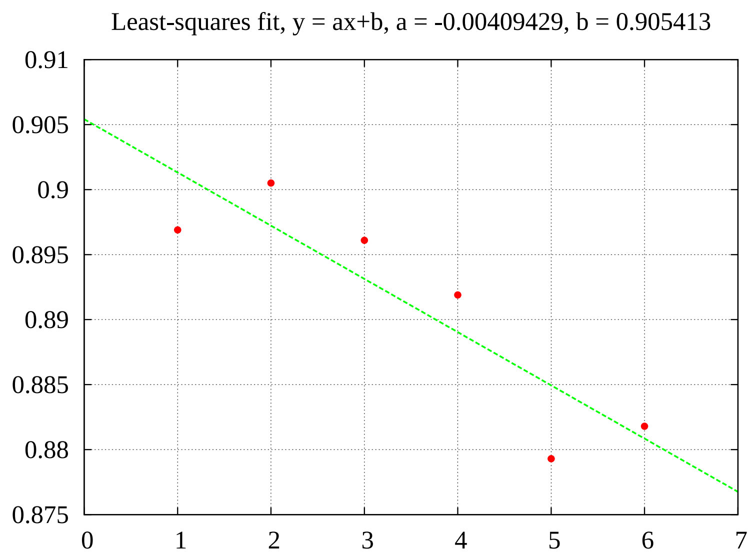
where $\bar{x} = \frac{1}{n} \sum x_i$ is the mean value.

Solution

```
// FILE: h014.cpp begins

```

```
#include <iostream>
#include <cstdlib>
#include <cmath>
#include "nr.h"

using namespace std;

double elem_sum(const Vec_DP &v)
{
  int i;
  double sum;
  for (i = 0, sum = 0.0; i < v.size(); i++)
    sum = sum + v[i];
  return sum;
}

Vec_DP operator*(const Vec_DP &a, const Vec_DP &b)
{
  Vec_DP c(a.size());
  for(int i = 0; i < a.size(); i++)
    c[i] = a[i]*b[i];
  return c;
}

void lsq_fit(const Vec_DP &x, const Vec_DP &y,
  double &a, double &b)
{
  int n = x.size();
  Vec_DP temp(x);
  double sumx, sumy, sumxy, sumxx;

  sumx = elem_sum(x);
  sumy = elem_sum(y);

  temp=temp*y;
  sumxy = elem_sum(temp);
  temp=x*x;
  sumxx = elem_sum(temp);

  a = (sumxy - sumx * sumy / n) / (sumxx - sumx*sumx / n);
  b = (sumy - a * sumx) / n;
}

int main()
{
  Vec_DP data(6);
```

```
    Vec_DP ordinal(6);
    double a, b;
    int i;

    data[0] = 0.8969;
    data[1] = 0.9005;
    data[2] = 0.8961;
    data[3] = 0.8919;
    data[4] = 0.8793;
    data[5] = 0.8818;

    for (i = 0; i < 6; i++)
      ordinal[i] = (double)(i+1);

    lsq_fit(ordinal, data, a, b);

    ofstream fout;
    fout.open("h014.dat");
    if(!fout.good())
    {
      cerr<<"Cannot open file\n";
      exit(1);
    }
    for (i = 0; i < 6; i++)
      fout<<i + 1<<" "<<data[i]<<endl;
    fout.close();
    fout.open("gnuplot.cmd");
    if(!fout.good())
    {
      cerr<<"Cannot open file\n";
      exit(1);
    }

    fout<<"set title ’Least-squares fit, y = ax+b, a = "<<a<<", ";
    fout<<"b = "<<b<<"’\nset grid \nset xrange [0:7] \n";
    fout<<"plot ’h014.dat’ t ’’ pt 7, "<<a<<"*x+"<<b<<" t ’’ w l lw 3 \n";
    fout<<"set terminal postscript color ’Times-Roman’ 22\n";
    fout<<"set output ’h014.ps’ \n";
    fout<<"replot \n pause -1\n";
    fout.close();
    system("gnuplot gnuplot.cmd");

    return 0;
  }

  // FILE: h014.cpp ends
```



Exercise 5

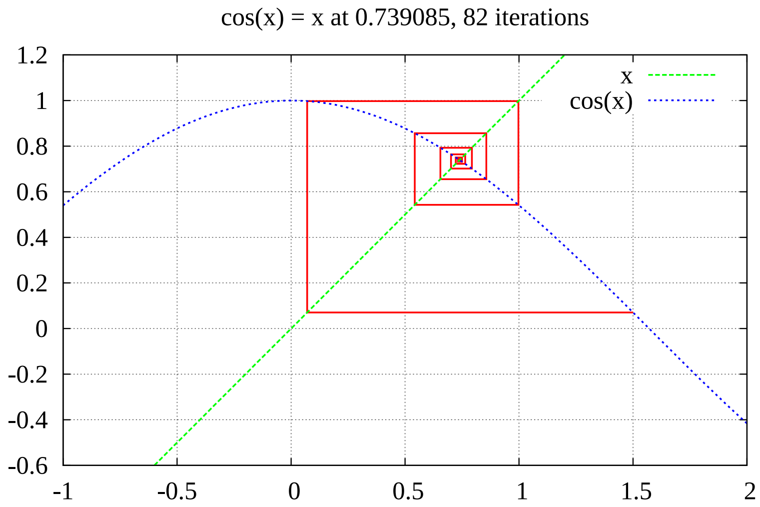
Use the fixed point iteration to solve the equations (a) $\cos(x) = x$, (b) $e^{-x} = x$, (c) $1 - \cosh(x) = x$.

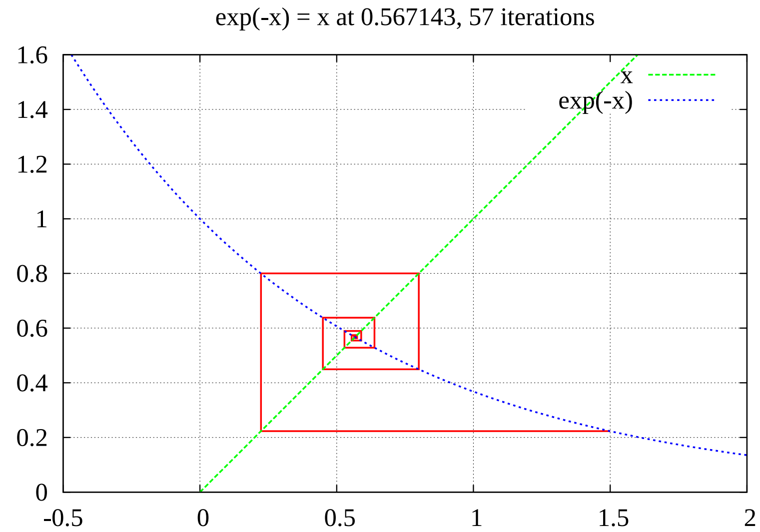
Solution

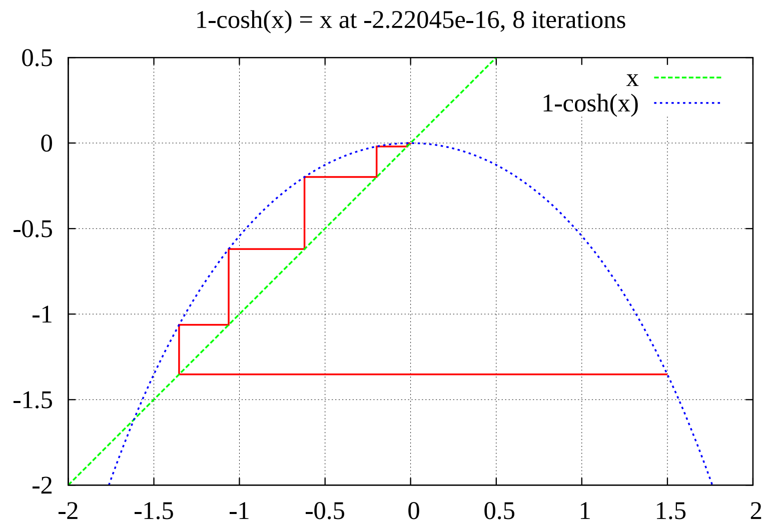
```
// FILE: h015.cpp begins

#include <cmath>
#include <cstdlib>
#include <fstream>
#include <iostream>

#define EPS 1e-14

using namespace std;

double f1 (double x)
{
  return cos(x);
}

double f2 (double x)
{
  return exp(-x);
}
```

```

double f3 (double x)
{
  return 1.0 - cosh(x);
}

int main()
{
  double (*f)(double);
  double x, y;
  int i, niter;
  const char *fun, *dat, *ps, *xrange, *yrange;

  cout<<"\nFixed point iterations. Initial point: x = 1.5 \n\n";
  ofstream fout_cmd("gnuplot.cmd");
  for (i = 1; i <= 3; i++)
  {
    switch (i)
        {
            case 1:
        f = &f1; xrange = "[-1:2]";
        yrange = "[-0.6:1.2]";
        fun = "cos(x)";
        dat = "h015a.dat";
        ps = "h015a.ps";
        break;
            case 2:
        f = &f2;
        xrange = "[-0.5:2]";
        yrange = "[0:1.6]";
              fun = "exp(-x)";
        dat = "h015b.dat";
        ps = "h015b.ps";
        break;
        case 3:
        f = &f3;
        xrange = "[-2:2]";
        yrange = "[-2:0.5]";
              fun = "1-cosh(x)";
        dat = "h015c.dat";
        ps = "h015c.ps";
        break;
    }

    x = 1.5; // the initial point
    y = (*f)(x);
```

```

    ofstream fout_data(dat);
    fout_data<<x<<" "<<y<<endl;

    for (niter = 0; fabs(x - y) > EPS; niter++)
    {
      x = y;
        y = (*f)(x);
        fout_data<<x<<" "<<y<<endl;
    }
    fout_data.close();

    cout<<fun<<" = x  at "<<x<<", "<<niter<<" iterations\n";
    fout_cmd<<"set terminal X11 \n set output \n set grid \n";
    fout_cmd<<"set title '"<<fun<<" = x at "<<x<<", "<<niter<<" iterations' \n";
    fout_cmd<<"set xrange "<<xrange<<" \n set yrange "<<yrange<<" \n set size ratio -1 \
    fout_cmd<<"plot '"<<dat<<"' t '' with steps lw 3, x w l lw 3, "<<fun<<" w l lw 3\n"
    fout_cmd<<"set terminal postscript color 'Times-Roman' 22\n";
    fout_cmd<<"set output '"<<ps<<"'\n replot \n";
    fout_cmd<<"pause -1 'Enter: ' \n";
  }

  fout_cmd.close();
  cout<<endl;
  system("gnuplot gnuplot.cmd");
  system("rm h015a.dat h015b.dat h015c.dat");

  return 0;
}

// FILE: h015.cpp ends

/* Output:
Fixed point iterations. Initial point: x = 1.5

cos(x) = x  at 0.739085, 82 iterations
exp(-x) = x  at 0.567143, 57 iterations
1-cosh(x) = x  at -2.22045e-16, 8 iterations
*/

```


cos(x) = x at 0.739085, 82 iterations
x
cos(x)
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-1
-0.5
0
0.5
1
1.5
2


exp(-x) = x at 0.567143, 57 iterations
x
exp(-x)
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
-0.5
0
0.5
1
1.5
2


1-cosh(x) = x at -2.22045e-16, 8 iterations
x
1-cosh(x)
0.5
0
-0.5
-1
-1.5
-2
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2

Exercise 6

The arithmetic-geometric mean $\mathrm{ag}(a, b)$ of two positive numbers $a > b > 0$ is defined as $\mathrm{ag}(a, b) = \lim a_n$, where $a_0 = a, b_0 = b$, and

$$a_{n+1} = (a_n + b_n)/2, \quad b_{n+1} = \sqrt{a_n b_n}, \quad n = 0, 1, 2, \dots$$

(a) Write a function, which takes two arguments (`double`), computes ag and returns the value (`double`).

(b) The hypergeometric function ${}_2F_1(a, b; c; x)$ is defined as a sum of the series,

$$ {}_2F_1(a, b; c; x) = 1 + \frac{ab}{c}\frac{x}{1!} + \frac{a(a+1)b(b+1)}{c(c+1)}\frac{x^2}{2!} + \dots + \frac{a(a+1)\dots(a+j-1)b(b+1)\dots(b+j-1)}{c(c+1)\dots(c+j-1)}\frac{x^j}{j!} + \dots \quad .$$

This hypergeometric series converges for $|x| < 1$. Gauss proved in 1799 that there is a connection between the hypergeometric function and the arithmetic-geometric mean,

$$ {}_2F_1(\tfrac{1}{2}, \tfrac{1}{2}; 1; r^2) = \frac{1}{\mathrm{ag}(1, \sqrt{1 - r^2})}$$

for $0 < r < 1$. Tabulate the difference of the two sides of this identity for $r = 0.05k, k = 1, \dots, 19$. Use a library routine to calculate the values of the ${}_2F_1$.

Solution

part a)

```
// FILE: h016a.cpp begins

#include <cmath>

#define EPS 1e-15
#define MAXITER 100

using namespace std;

double ag(double a, double b)
{
  double a1, b1;
  int n = 0;

  while (fabs(a - b) > EPS && n < MAXITER)
  {
    a1 = (a + b) / 2;
    b1 = sqrt(a * b);
    a = a1;
```

```
    b = b1;
    n++;
  }

  return a;
}

// FILE: h016a.cpp begins
```

part b)

```
// FILE: h016b.cpp begins

#include <iostream>
#include <iomanip>
#include <cmath>
#include <complex>
#include "nr.h"
#include "h016a.cpp"

using namespace std;

int main()
{
  int k;
  const char *chd[] = {
    "r", "2F1 (1/2, 1/2; 1; r^2)",
                "1/ag(1, sqrt(1-r^2))",
    "difference"
  };

  cout<<endl<<setw(3)<<chd[0]<<setw(28)<<chd[1]<<setw(23);
  cout<<chd[2]<<setw(14)<<chd[3]<<endl<<endl;
  for (k = 1; k <= 19; k++)
  {
    double r = 0.05 * k;
    double hyperg = (NR::hypgeo(0.5, 0.5, 1.0, pow(r, 2))).real();
    double aginv = 1.0 / ag(1.0, sqrt(1.0 - pow(r, 2)) );
    cout.precision(10);
    cout<<setw(4)<<r<<setw(23)<<hyperg<<setw(23)<<aginv;
    cout<<setw(20)<< fabs(hyperg - aginv)<<endl;
  }
  return 0;
}

// FILE: h016b.cpp ends
```

```

/* Output:
  r      2F1 (1/2, 1/2; 1; r^2)   1/ag(1, sqrt(1-r^2))    difference

0.05          1.00062588            1.00062588     2.220446049e-16
 0.1         1.002514161           1.002514161     2.220446049e-16
0.15         1.005697323           1.005697323     2.220446049e-16
 0.2         1.010231448           1.010231448                   0
0.25          1.01619936            1.01619936     2.220446049e-16
 0.3         1.023715546           1.023715546     2.220446049e-16
0.35         1.032933472           1.032933472     2.220446049e-16
 0.4         1.044056341           1.044056341     2.220446049e-16
0.45         1.057353019           1.057353019                   0
 0.5         1.073182007           1.073182007                   0
0.55         1.092028589           1.092028589     2.220446049e-16
 0.6         1.114564487           1.114564487     6.661338148e-16
0.65         1.141748341           1.141748341                   0
 0.7         1.175005293           1.175005293                   0
0.75         1.216573879           1.216573879     2.220446049e-15
 0.8           1.2702492             1.2702492     8.881784197e-16
0.85         1.343226637           1.343226637     5.329070518e-15
 0.9         1.451842673           1.451842673     7.993605777e-15
0.95          1.64885236            1.64885236     2.886579864e-15
*/
```