Source code for project 2
FYS4150

Fredrik E Pettersen

October 4, 2012

.cpp - file

```
/*
 * File:   main.cpp
 * Author: Fredrik E Pettersen
 * Program description: This program is for project 1 in FYS4150.
 * The aim is to solve a second order differential equation by linear algebra.
 * Created on 30. august 2012, 08:31
 */


#include "jacobi.h"

int main(int argc, char** argv) {

//------ Initialization --------

    int n,choose;
    double h,rho_0,rho_n, start, stop;
    rho_0 = 0.0;
//    cout << "Enter the size of the NxN matrix" << endl;
//    cin >> n;
//    cout<<"Enter rho_max"<<endl;
//    cin >> rho_n;
    n = atoi(argv[1]);
    rho_n = atof(argv[2]);
    choose = 1;//choose which potential to use 0 is without repulsive coloumbforce
    double omega = atof(argv[3]);
    h = (rho_n - rho_0)/(n+1);
    vec rho= linspace<vec>(rho_0+h,rho_n-h,n);
    mat A = make_A(n, rho_0, rho_n, rho, choose,omega);
    mat R(n,n); R.eye();
    double eps = 1e-9;//atof(argv[3]);
    int k=0;
    int l=0;
    int rotation_counter = 0;
    int max_it = 10*n*n;
    double max_offdiag = maxoffdiag(A, &k, &l, n);
//------ Solving the equations ----------
    cout<<"Init ok"<<endl;

    vec eigval(n); eigval(0) = A(0,0);
    vec nondiag(n);
    mat eigvec(n,n); eigvec.eye();
    for(int i=1;i<n;i++){
        eigval(i) = A(i,i);
        nondiag(i) = A(i-1,i);
```

```
    }
    start = clock();
    eigvec = tqli(eigval,nondiag,n,eigvec);
    stop = clock();
    cout<<"tqli function done. "<<timediff(start,stop)<<" ms. "<<endl;
    /*
    start = clock();     //start timing the computation
    while(max_offdiag>eps && rotation_counter<max_it){
        max_offdiag = maxoffdiag(A, &k, &l,n);
        rotate(A,R,k,l,n);
        rotation_counter ++;
    }
    stop = clock();
    */
    //-------- Print some messages to screen-------------
    /*
    cout<<"Yarrr! We be done! We been runnin' "<<rotation_counter<<" rounds in circle. Thi
    cout<< timediff(start,stop)<<" ms" <<endl;
    cout<<n<<" by "<<n<<" matrix, rho_max= "<<rho_n<<", eps= "<<eps<<", largest element: "<
    */
    cout<<"Done! "<<n<<" by "<<n<<" matrix. rho_max = "<<rho_n<<" eps = "<<eps<<endl;
    /*
    double eigval1 = 0;
    double eigval2 = 0;
    double eigval3 = 0;
    */
    //print_diag(A,3,n, &eigval1, &eigval2, &eigval3);
    vec min  =sort_eigenvalues(eigval, 3, n);
    vec eigenvec0=eigvec.col(min(0));
    vec eigenvec1=eigvec.col(min(1));
    vec eigenvec2=eigvec.col(min(2));
    ofstream myfile;
    myfile.open(make_filename(n,rho_n,omega,0));
    for(int i=0;i<n;i++){
        myfile <<eigenvec0(i)<<"  "<<rho(i)<<" "<<eigenvec1(i)<<" "<<eigenvec2(i)<<endl;
    }
    myfile.close();

    return 0;
}
```

.h - file

```
/*
 * File:   jacobi.h
 * Author: fredrik
 *
 * Created on 14. september 2012, 13:26
 */


#ifndef NEWFILE_H
#define NEWFILE_H
double sqrarg;
#define   SQR(a) ((sqrarg = (a)) == 0.0 ? 0.0 : sqrarg * sqrarg)
#define   SIGN(a,b) ((b)<0 ? -fabs(a) : fabs(a))

#include <cstdlib>
#include "armadillo"
#include <fstream>
#include <iostream>
```

```
#include <cmath>
#include <time.h>
using namespace arma;
using namespace std;

double maxoffdiag(mat &A, int *k, int *l, int n );
void rotate(mat &A, mat &R, int k, int l, int n);
double Potential(double r,int choose,double omega);
mat make_A(int n, double rho_0, double rho_n, vec rho,int choose,double omega);
int print_diag(mat A, int stop, int n,double* eigval1,double* eigval2,double* eigval3);
double timediff(double time1, double time2);
char* make_filename(int n, double rho_n,double omega,int power);
mat tqli(vec &d, vec e, int n, mat z);
double pythag(double a, double b);
vec sort_eigenvalues(vec A, int stop,int n);
#endif  /* JACOBI_H */

double maxoffdiag(mat &A, int *k, int *l, int n ){
    //Returns the sum/norm of the nondiagonal elements and updates k,l with the
    //index of the largest nondiagonal matrixelement. Requires a symmetric matix
    double max = 0.0;
    //double sum = 0.0;
    for (int i=0;i<n;i++){
        for(int j=i+1;j<n;j++){
            //sum += A(i,j)*A(i,j);
            if(max<fabs(A(i,j))){
                max = fabs(A(i,j));
                *l=i;
                *k=j;
            }
        }
    }
    return max;
}
void rotate(mat &A, mat &R, int k, int l, int n){
    //Does one Jacobi-rotation.
    double s,c;
    if(A(k,l)!=0.0){
        double tau,t;
        tau = (A(l,l)-A(k,k))/(2*A(k,l));
        if(tau>0){
            t = 1.0/(tau + sqrt(1.0 + tau*tau));
        }
        else{
            t = -1.0/(-tau +sqrt(1.0+tau*tau));
        }
        c = 1.0/sqrt(1.0+t*t);
        s = t*c;
    }
    else{
        c = 1.0;s = 0.0;
    }
    double a_kk,a_ll,a_il,a_ik,r_ik,r_il;
    a_kk = A(k,k);
    a_ll = A(l,l);
    A(k,k) = c*c*a_kk -2.0*c*s*A(k,l) +s*s*a_ll;
    A(l,l) = s*s*a_kk +2.0*c*s*A(k,l) +c*c*a_ll;

    for(int i=0;i<n;i++){
        if(i != k && i != l){
            a_ik = A(i,k);
```

```
            a_il = A(i,l);
            A(i,k) = c*a_ik - s*a_il;
            A(k,i) = A(i,k);
            A(i,l) = c*a_il + s*a_ik;
            A(l,i) = A(i,l);
        }
        r_ik = R(i,k);
        r_il = R(i,l);
        R(i,k) = c*r_ik - s*r_il;
        R(i,l) = c*r_il + s*r_ik;
    }
    A(k,l) = 0.0;
    A(l,k) = 0.0;
    return;
}

double Potential(double r,int choose, double omega){
    //Returns the potential.
    double pot;
    if(choose==0){
        pot = r*r;
    }else if(choose==1){
        pot = omega*omega*r*r +1.0/r;
    }
    return pot;
}

mat make_A(int n, double rho_0, double rho_n, vec rho,int choose,double omega){
    //Returns the tridiagonal matrix A
    mat A(n,n); A.fill(0.0);
    mat V(n,1); V.fill(0.0);

    double h = (rho_n-rho_0)/(n+1);
    double H = h*h;
    A(0,0) = 2/H - Potential(rho_0+h,choose,omega);
    A(0,1) = -1/H;
    A(n-1,n-2) = A(0,1);
    A(n-1,n-1) = 2/H + Potential(rho_n-h,choose,omega);
    for(int i=1;i<n;i++){
        V(i) = Potential(rho(i),choose,omega);
        for(int j=1;j<n-1;j++){
            if(i==j){
              A(i,j) = 2/H + V(i);
              A(i,j-1) = -1/H;
              A(i,j+1) = -1/H;
            }
        }
    }
    return A;
}
int print_diag(mat A, int stop,int n,double* eigval1,double* eigval2,double* eigval3){
    //Prints the diagonal elements of a matrix up to index stop sorted in ascending order.
    vec values(n);
    for(int j=0;j<n;j++){
        values(j)=A(j,j);
    }

    values = sort(values);
    for(int i=0;i<stop;i++){
        if(i==0){ cout<<"the "<<i+1<<"'st eigenvalue is: "<<values(i)<<endl;}
        else if(i==1){cout<<"the "<<i+1<<"'nd eigenvalue is: "<<values(i)<<endl;}
```

```
        else if(i==2){cout<<"the "<<i+1<<"'rd eigenvalue is: "<<values(i)<<endl;}
        else {cout<<"the "<<i+1<<"'th eigenvalue is: "<<values(i)<<endl;}
    }
    *eigval1 = values(0);
    *eigval2 = values(1);
    *eigval3 = values(2);
    return index;
}
double timediff(double time1, double time2){
    // This function returns the elapsed time in milliseconds
    return ((time2 - time1)*1000)/CLOCKS_PER_SEC;
}
char *make_filename(int n, double rho_n, double omega,int power){
        //Returns a filename saying something about the particular run.
        char* buffer = new char[60];
        sprintf(buffer,"coloumb_n%d_rhomax%g_omega_%ge%d.txt",n,rho_n,omega,power);
        return buffer;
}

mat tqli(vec &d, vec e, int n, mat z){
   //Modified version of tqli from file lib.cpp which uses armadillo matrices
   //and vectors.
   register int   m,l,iter,i,k;
   double         s,r,p,g,f,dd,c,b;
   for(i = 1; i < n; i++) {e(i-1) = e(i);}
   e(n-1) = 0.0;
   for(l = 0; l < n; l++) {
      iter = 0;
      do {
         for(m = l; m < n-1; m++) {
            dd = fabs(d(m)) + fabs(d(m+1));
            if((double)(fabs(e(m)+dd) == dd)) {break;}
         }
         if(m != l) {
            if(iter++ == 30) {
               printf("\n\nToo many iterations in tqli.\n");
               exit(1);
            }

            g = (d(l+1) - d(l))/(2.0 * e(l));
            r = pythag(g,1.0);
            g = d(m)-d(l)+e(l)/(g+SIGN(r,g));
            s = c = 1.0;
            p = 0.0;
            for(i = m-1; i >= l; i--) {
               f      = s * e(i);
               b      = c*e(i);
               e(i+1) = (r=pythag(f,g));
               if(r == 0.0) {
                  d(i+1) -= p;
                  e(m)    = 0.0;
                  break;
               }
               s      = f/r;
               c      = g/r;
               g      = d(i+1) - p;
               r      = (d(i) - g) * s + 2.0 * c * b;
               d(i+1) = g + (p = s * r);
               g      = c * r - b;
               for(k = 0; k < n; k++) {
                  f          = z(k,i+1);
```

```
                    z(k,i+1) = s * z(k,i) + c * f;
                    z(k,i)   = c * z(k,i) - s * f;
                } /* end k-loop */
            } /* end i-loop */
            if(r == 0.0 && i >= l) continue;
            d(l) -= p;
            e(l)  = g;
            e(m)  = 0.0;
         } /* end if-loop for m != 1 */
      } while(m != l); //?
   } /* end l-loop */
   return z;
} /* End: function tqli(), (C) Copr. 1986-92 Numerical Recipes Software )%. */

double pythag(double a, double b){
  double absa,absb;
  absa=fabs(a);
  absb=fabs(b);
  if (absa > absb) return absa*sqrt(1.0+SQR(absb/absa));
  else return (absb == 0.0 ? 0.0 : absb*sqrt(1.0+SQR(absa/absb)));
}

vec sort_eigenvalues(vec A, int stop,int n){
    //Sorts the entries in a vector in ascending order and returns their indeces prior to
    //Prints the n first diagonal elements of a matrix.
    vec index(n);
    vec values(n);
    double sugg = 0;
    for(int j=0;j<n;j++){
        sugg = 1e10;
        for(int i=0;i<n;i++){
            if(sugg>A(i)){
                sugg = A(i);
                index(j) = i;
            }
        }
        values(j) = sugg;
        A(index(j))=2e10;
    }
    //values.print("values:");
    for(int i=0;i<stop;i++){
        if(i==0){ cout<<"the "<<i+1<<"'st eigenvalue is: "<<values(i)<<endl;}
        else if(i==1){cout<<"the "<<i+1<<"'nd eigenvalue is: "<<values(i)<<endl;}
        else if(i==2){cout<<"the "<<i+1<<"'rd eigenvalue is: "<<values(i)<<endl;}
        else {cout<<"the "<<i+1<<"'th eigenvalue is: "<<values(i)<<endl;}
    }

    return index;
}
```

script

```
n = linspace(50,750,15);
rho_max = linspace(4,15,12);

for i=1:length(n)
    for rho=1:length(rho_max)
        call = sprintf('./goggen %d %f',n(i),rho_max(rho));
        system(call);
    end
end
```