D A S U N A N K A
R A B B I N A K A
F A R A K O M A I

Computer Techniques in Physics

Shehu S. AbdusSalam

Department of Physics,
Shahid Beheshti University

2nd semester 1395/1396 (2016/2017)

Aims, overview, & outlook

- Introduce concepts and techniques to boost students' confidence and abilities in using computers for Physics.
- Address: Emacs; C programming language; Random number & Monte Carlo methods; numerial integration, differentiation/finite differencing; Physics modelling/differential equations & simulations;
- Sundays 13:00 to 15:00 & 15:00 to 17:00
  Exams: 1396/01/27 and 1396/03/27
- attendance(1), quiz(1), projects(8) & exams(10)

Possible References:

- http://publications.gbdirect.co.uk/c_book/thecbook.pdf
- B.W. Kernighan and D.M. Ritchie, "The C programming Language"
- N.J. Giordano and H. Nakanishi, "Computational Physics"
- The world wide web

کاربرد کامپیوتر در فیزیک

شماره درس: ۴۱۱

تعداد واحد: ۳ واحد (۲ واحد نظری + ۱ واحد عملی)

نوع واحد: نظری و عملی

پیشنیاز: مبانی کامپیوتر و برنامه سازی و ترمودینامیک و مکانیک آماری

سرفصل: درس شامل دو قسمت نظری و کار عملی با کامپیوتر به شرح زیر است

الف: قسمت نظری (۳۴ ساعت):

۱- آشنائی با بعضی از جنبه های یکی از زبانهای برنامه سازی

۲- مروری بر روشهای عددی مورد نیاز (حل دستگاههای معادلات - حل عددی معادلات دیفرانسیل - محاسبه انتگرالها - روشهای ماتریسی و ....)

۳- شبیه سازیهای کامپیوتری با روش مونته کارلو و دینامیک مولکولی با ذکر مثالهایی از کاربرد آنها در مطالعه مواردی از قبیل: مواد چگال، پلاسما، قطعات نیمه رسانا، اختر فیزیک و محاسبات کوانتومی

۴- آشنائی با چند بسته نرم افزاری مورد نیاز در محاسبات علمی

ب: کار عملی با کامپیوتر (۳۴ ساعت)

شامل ۲-۳ پروژه عملی کامپیوتری از موارد قسمت الف است که به دانشجویان واگذار می شود تا بطور فردی یا در دسته های چند نفره انجام دهند. (توصیه می شود اگر امکانات اجازه دهد حداقل یک پروژه با کامپیوتر اصلی و یک پروژه با کامپیوتر شخصی انجام گیرد). توصیه می شود یک پروژه در ساختن Interface بین آزمایش و کامپیوتر ساخته و مطالعه شود.

مأخذ درس :

1- M. Metcalf, Effective FORTRAN 77, Clarendon Press, 1986

2- R.W. Hockney & J.W. Eastwood, "Computer Simulation Using Particles", Adam Hilger 1988.

3- N.P. Allen and D.J. Tildesley, " Computer Simulation of Liquids", Clarendon Press, Oxford, 1987

4- Steven E. Koonin " Computational Physics " Addison-Wesely, 1985

توصیه می شود حداقل در این درس ۷۰ درصد مطالب ذکر شده پوشیده شود و بقیه مطالب با انتخاب مدرس باشد.

Feed back

For making a better Computational Physics course

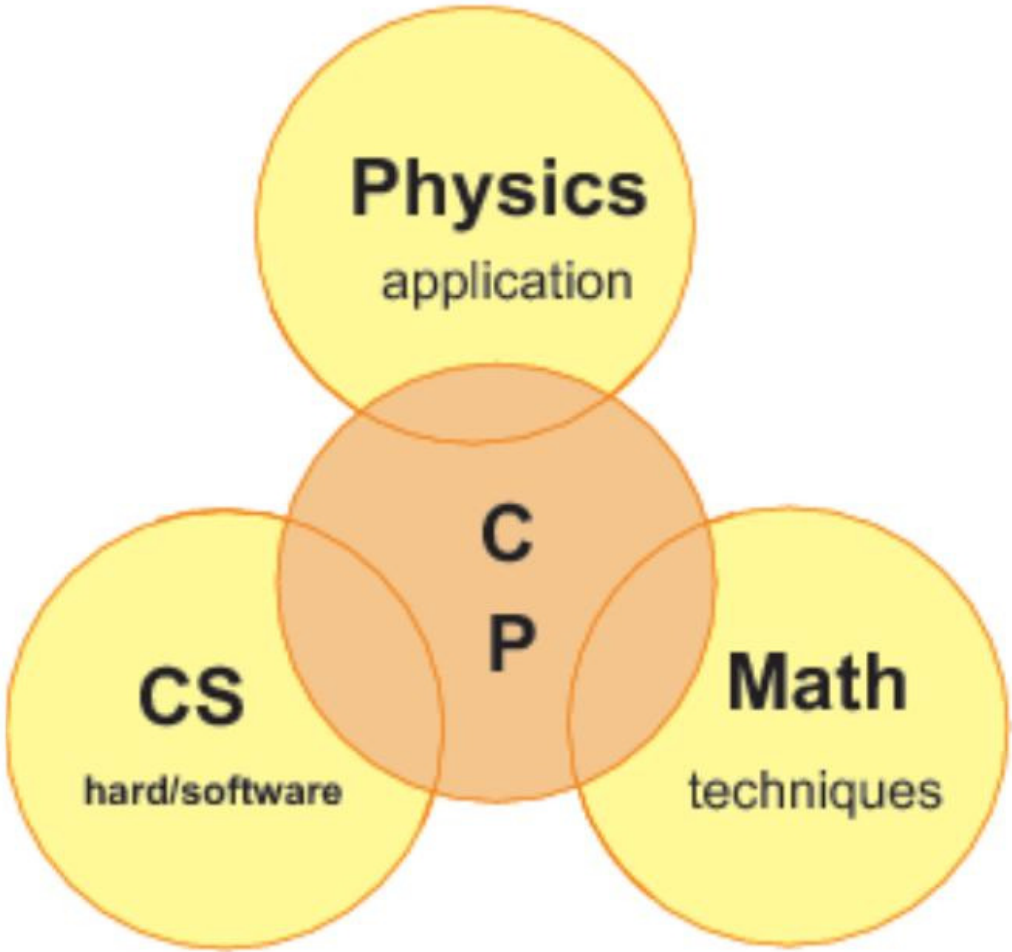
Say what you don't like & how to improve, etc

Introduction

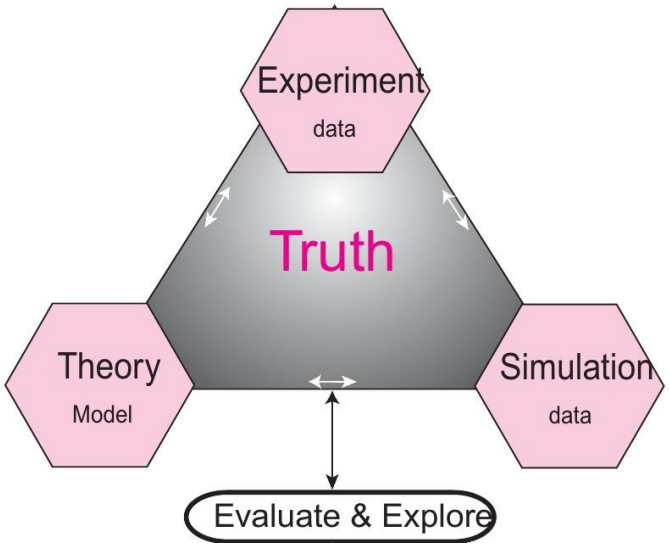
Introduction, Ref[0]

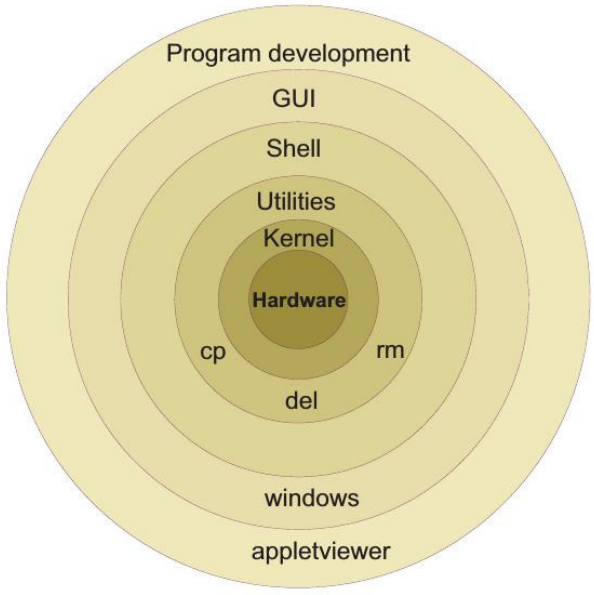
Computers are incredibly fast, accurate, and stupid; humans are incredibly slow, inaccurate, and brilliant; together they are powerful beyond imagination.

— *Albert Einstein*



Introduction, Ref[0]





Ref[0]: R.H. Landau, M.J. PAEZ, & C.C. Bordeianu,
"Computational Physics – Problem Solving with Computers"

Emacs/gedit/nano & Intro. Linux Command Line

Emacs & Intro. Linux commands

The topics to be address:

- Introduction to Emacs, gedit and nano text editors
- Files, directories, file system; viewing content, creating, moving, deleting
- Introduce commands: ls, mkdir, cd, rm, rm -r, man, info, file, echo, echo $?, gcc
- Writing & compiling simplest C program using Emacs

References:

1) https://www.gnu.org/software/emacs/tour/index.html

```c
/* Simplest C program - empty */

/* ------------ (1) ------------- */
/* show use of return value
   compile the source code  : gcc hello_empty.c
   execute the program      : ./a.out
   on the command line, type: echo $?
*/


/* ------------ (2) ------------- */
/* change the return value from 0 to something else
   and then repead (1) above
*/

int main(){
  return 0;
}
```

```c
/* Salam Dunya va Akhirat */

/* ------------ (1) ------------- */
/* show use of return value
   compile the source code  : gcc hello.c
   execute the program      : ./a.out
   see the printed message:
   on the command line, type: echo $?
*/


/* ------------ (2) ------------- */
/* change the message to be printed to something else
   and then repead (1) above
*/

#include <stdio.h>
int main(){
  printf("hello, world\n");
  return 1110;
}
```

Exercise 01

- (a) Read through Ref[1] for Emacs hands-on practise, especially the "Basic editing commands" part.
- (b) Practise the use of Emacs or nano/pico text editor: Write and compile a C program that should print on the screen the following: "Man Beheshti Hastam." Compile using the command "gcc filename.c" and run the program using "./a.out"

Program compilation & linking

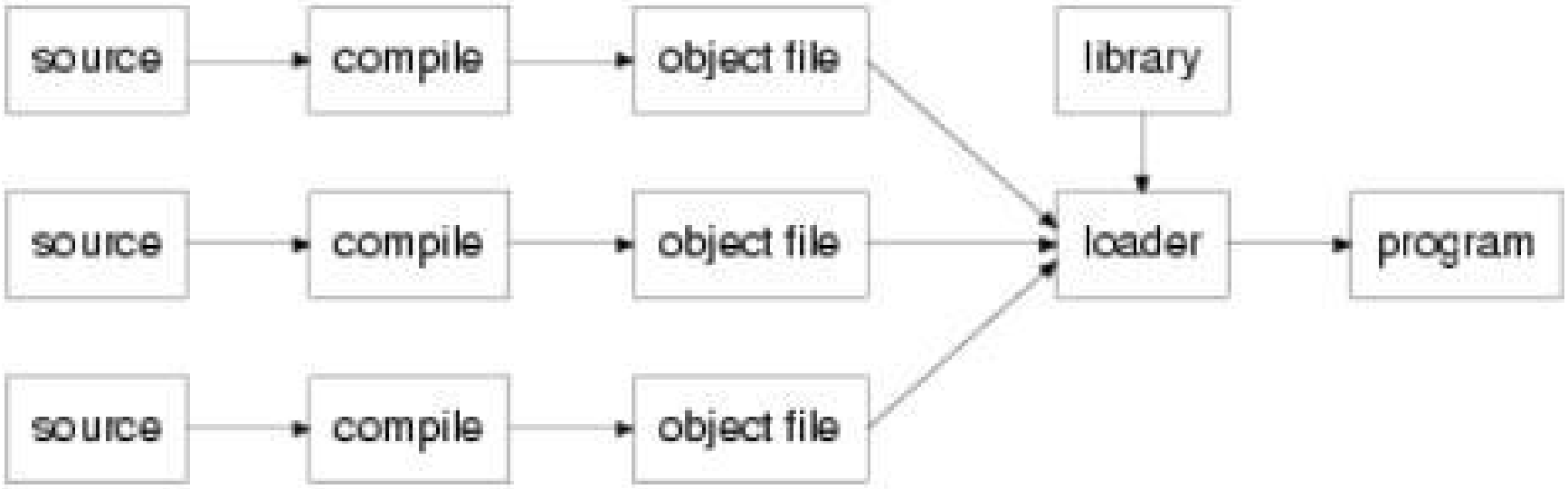


Figure 1.1: Diagram showing multiple files going from source, through compilation, to object files, and being combined with libraries by the loader to produce a program.

From: publications.gbdirect.co.uk/c_book/thecbook.pdf

C programming language

C programming language

- Introduction to C
- Variables and Arithmetic
- Flow Control and Logical Expressions
- Printing to, & Reading from Files; argc & argv
- Functions
- Arrays and Pointers
- Structures

C: main(), printf(), #include directive

An empty C program:

```
int main(){
  return 0;
}
```

A simple non-empty C program:

```
#include <stdio.h>
int main(){
  printf("hello, world\n");
  return 1110;
}
```

C: variables, while-loop, function

```c
// preprocessor directive/statement
#include <stdio.h>

// function declaration
void show_message(void);

// main function definition
int main(){
  int count = 0;          // variable declaration & initialisation
  while(count < 10){      // while-loop
    show_message();       // function call
    count = count + 1; // counter increment
  }
  return 0;               // return statement
}

// function definition
void show_message(void){
  printf("Salam, Salam, Salam ...\n");
}
```

C: if-statement, getchar()

```c
#include <stdio.h>
#include <stdlib.h>

int main(){
  char ch;
  ch = getchar();
  while(ch != 'a'){
    if(ch != '\n') // if-statement
      printf("ch was %c, value %d\n", ch, ch);
    ch = getchar();
  }
  return EXIT_SUCCESS;
}
```

C: Arrays, nested while-loop, sorting

arrays.c

```c
#include <stdio.h>
#include <stdlib.h>

#define ARSIZE 10

int main(){
  int ch_arr[ARSIZE], i, j, temp;
  int stop = 0, k = 0;

  // Read characters into array, stops when ENTER is pressed
  // or when array is full
  while(stop != 1){
    ch_arr[k] = getchar();
    if(ch_arr[k] == '\n') stop = 1;
    else k = k + 1;
    if(k == ARSIZE) stop = 1;
  }

  k = k - 1; // resetting k
```

```
  i = 0;          //bubble sort.
  while(i < k){
    j = i + 1;
    while(j <= k){
      if(ch_arr[i] > ch_arr[j]){ // swap
        temp = ch_arr[i];
        ch_arr[i] = ch_arr[j];
        ch_arr[j] = temp;
      }
      j = j + 1;
    }
    i = i + 1;
  }

  i = 0; // resetting i
  while(i <= k){
    printf("%c\n", ch_arr[i]);
    i = i + 1;
  }

  return EXIT_SUCCESS;
}
```

C: for-loop, printf(), pointers

```c
#include <stdio.h>
#include <stdlib.h>

int main(){
  int i, j;
  int *m, *n, ar[10];

  printf("i : j \n");
  for(i = 0; i < 10; i++){
    for(j = 0; j < 10; j++){
      if( i < 2 && j < 2) printf("%d : %d \n", i, j);
    }
    ar[i] = i + j;
  }

  // & operator gives the address of a variable
  printf("\nThe address of i is %p\n", &i);
  printf("The address of j is %p\n", &j);
```

```
  // m is assigned the address of i
  m = &i;
  n = &j;

  // indirection or dereference operator (*)
  printf("\nm contains %d\n", *m);
  printf("n contains %d\n", *n);

  for(i = 0; i < 4; i++)
    printf("ar[%d] = %d and its address is: %p \n",
           i, *(ar + i), (ar + i) );
  // double %f or %e; char %c; int %d; pointer %p

  return EXIT_SUCCESS;
}
```

C: main argc & argv

```c
#include <stdio.h>

int main (int argc, char *argv[]){
  int count;
  printf ("This program was called with \"%s\".\n", argv[0]);

  if (argc > 1){
    for (count = 1; count < argc; count++)
      printf("argv[%d] = %s\n", count, argv[count]);
  }
  else {
    printf("The command had no other arguments.\n");
  }

  return 0;
}
```

structures.c

```c
// This program reads characters from standard input
// into an array and sorts them when a newline has is
// passed or when the array is full (10 characters),
// the characters are then printed in increasing order

#include <stdio.h>
#include <stdlib.h>

#define ARSIZE 10

struct wp_char{
  char wp_cval;
  int wp_font;
  int wp_psize;
} ar[ARSIZE];


struct wp_char infunction(void);
```

```
int main(){

  int i, lo_indx, hi_indx;

  // read in the characters
  for(i = 0; i < ARSIZE; i++){
    ar[i] = infunction();
    if(ar[i].wp_cval == '\n') break;
  }

  // now a simple exchange sort
  for(lo_indx = 0; lo_indx <= i-2; lo_indx++)
    for(hi_indx = lo_indx + 1; hi_indx <= i-1; hi_indx++){
      if(ar[lo_indx].wp_cval > ar[hi_indx].wp_cval){
        // Swap the two structures.
        struct wp_char wp_tmp = ar[lo_indx];
        ar[lo_indx] = ar[hi_indx];
        ar[hi_indx] = wp_tmp;
      }
    }
```

```
  // now print
  for(lo_indx = 0; lo_indx < i; lo_indx++){
    printf("%c %d %d\n", ar[lo_indx].wp_cval,
           ar[lo_indx].wp_font,
           ar[lo_indx].wp_psize);
  }

  return EXIT_SUCCESS;
}


struct wp_char infunction(void){
  struct wp_char wp_char;

  wp_char.wp_cval = getchar();
  wp_char.wp_font = 2;
  wp_char.wp_psize = 10;

  return wp_char;
}
```

Exercise 02

(1) Change the program array.c above to use for-loop instead of a while loop.

(2) Modify array.c and structures.c so that the characters are read by using main() arguments.

Project 01 (to be submitted by 1396-01-15)

The Fourier series representation of the function,

$$f(x) = \begin{cases} -1 & \text{if } -\pi < x < 0, \\ 1 & \text{if } 0 < x < \pi \end{cases}$$

is given by

$$f(x) = \frac{4}{\pi} \sum_{n=1,3,5,\ldots}^{\infty} \frac{\sin(nx)}{n}.$$

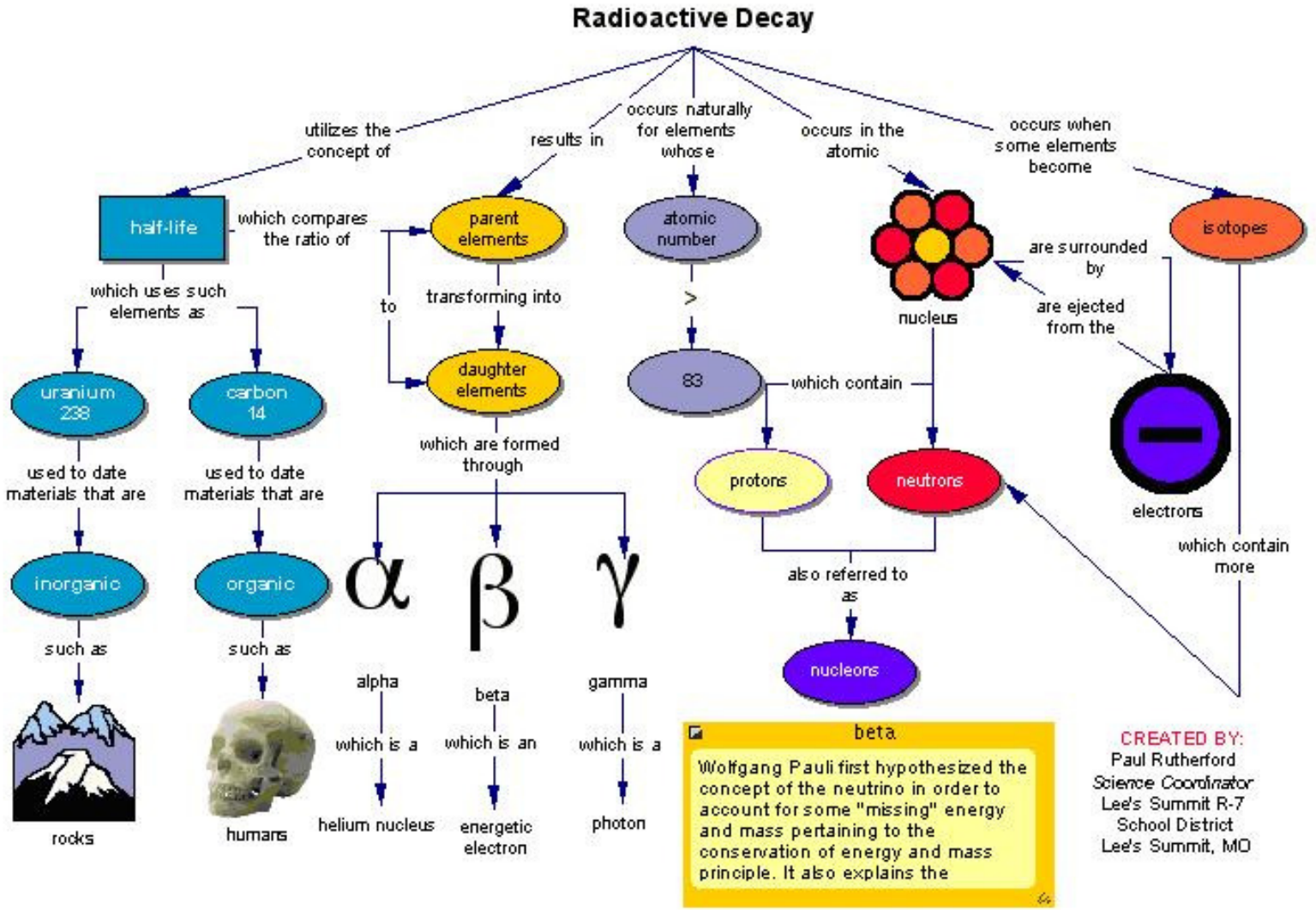
Write a C program for computing f(x). Plot your results to compare the function and its Fourier approximation.

Ordinary Differential Equations

Ordinary Differential Equations

- Generic form
- Example physics: radio-active decay, projectile, pendulum, etc
- Numerical solution, Euler method
- Solving nuclear decay, projectile and pendulum ODEs
- Instability of Euler method; Euler-Cromer method
- Sun-Earth & Sun-Earth-Jupiter systems

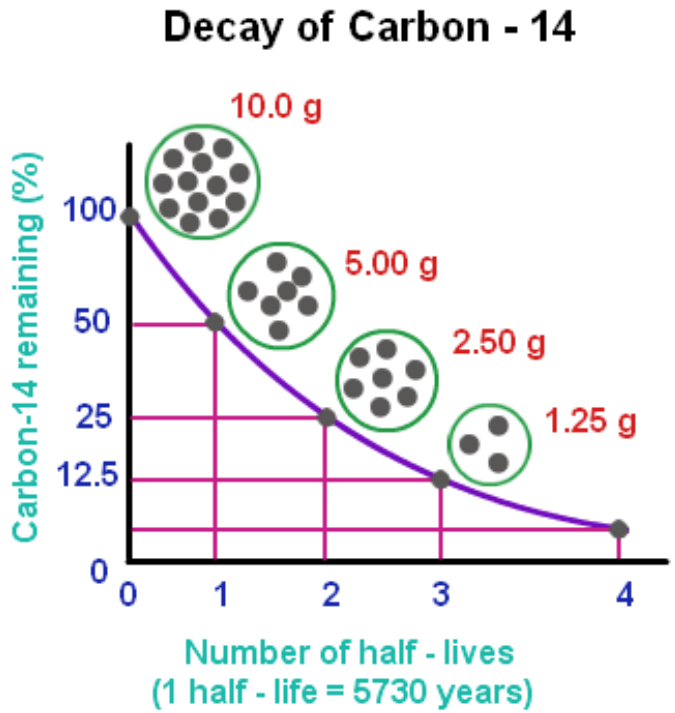
Radioactive Decay



http://schools.wikia.com/wiki/Radioactive_decay

Radioactive Decay, ODE

- $N(t) =$ no. of ^{14}C in a sample at time, t
- Decay is governed by $\frac{dN(t)}{dt} = -\frac{N(t)}{\tau}$



- Task: solve numerically the ODE $\frac{dN(t)}{dt} = -\frac{N(t)}{\tau}$, given that $N(t=0) = 10000$ adnd $\tau = 0.0005$

Radioactive Decay, Euler Method

```c
#include <stdio.h>
#include <math.h>
#include "mheader.h"

#define MAX 100

int main(){
  double nsample[MAX];  /* no. of atoms in sample*/
  double t[MAX];        /* time values           */
  double tau;           /* half life  of atom    */
  double dt;            /* time steps            */

  initialise(nsample, t, & tau, & dt);
  calculate_nt(nsample, t, & tau, & dt);
  store_result(nsample, t);

  return 0;
}

void initialise(double *nsamp, double *time, double *tau, double *delta_t){
  printf("Enter initial number of atoms: ");
  scanf("%lf", nsamp);
  printf("Enter time constant for the atom: ");
  scanf("%lf", tau);
  printf("Enter time step: ");
  scanf("%lf", delta_t);
  time[0] = 0.0;
}
```

Radioactive Decay, Euler Method

```
void  calculate_nt(double *n_i, double *time, double *tau, double *delta_t){
  int i;
  for(i=0; i < MAX-1; i++){
    n_i[i+1] = n_i[i] - (n_i[i]/(*tau)) * *delta_t;
    time[i+1] = time[i] + *delta_t;
  }
}

void store_result(double *n_i, double *time){
  FILE *file_out;
  int i;

  file_out = fopen("RadioActiveDecay.txt", "w");
  for(i=0; i < MAX; i++){
    fprintf(file_out, "%g %g\n", time[i], n_i[i]);
  }
  fclose(file_out);
}
// gnuplot
// plot "RadioActiveDecay.txt" using 1:2
```

mheader.h:

```
void initialise(double *, double *, double *, double *);
void  calculate_nt(double *, double *, double *, double *);
void store_result(double *, double *);
```

Radioactive Decay, Euler method

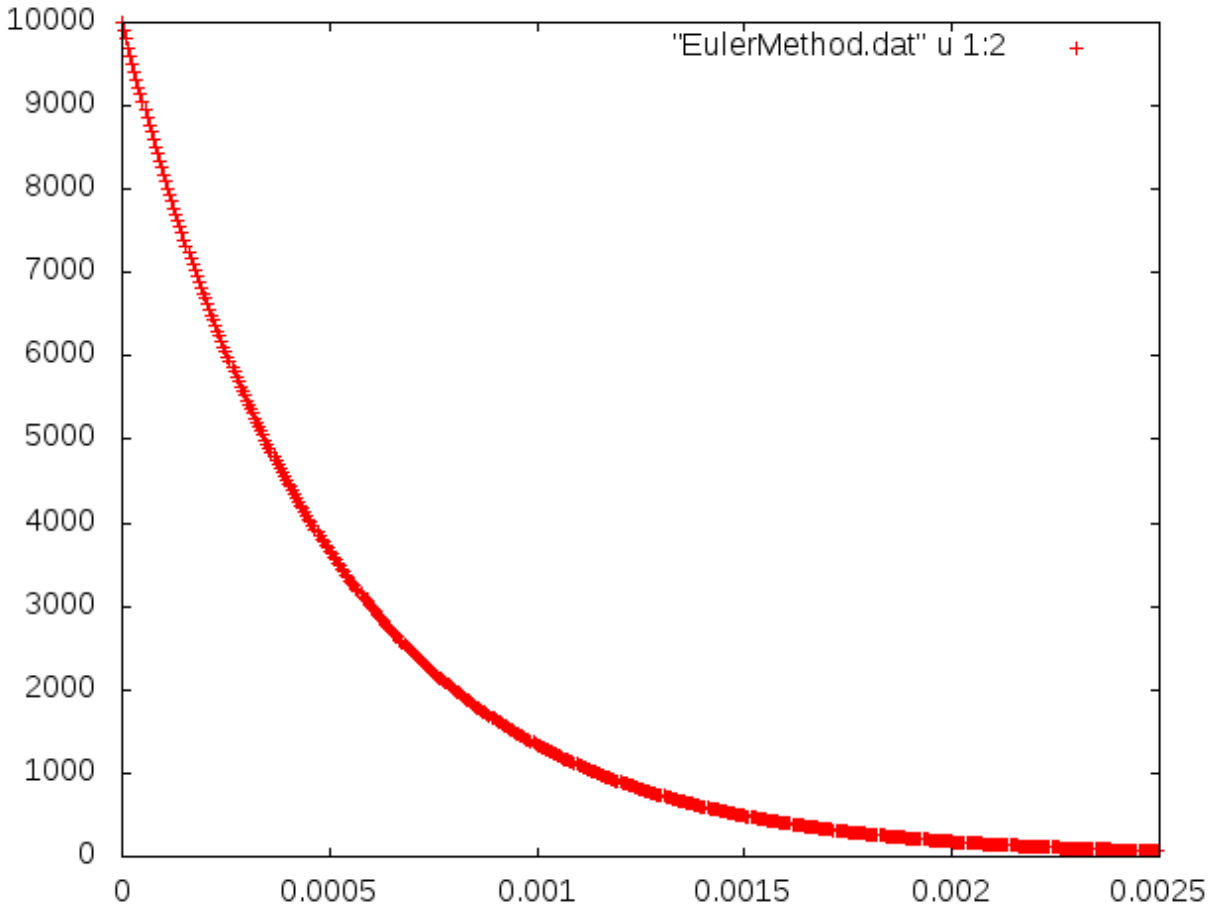


Figure : Numerical solution using Euler method, for the radio-active ODE $\frac{dN(t)}{dt} = -\frac{N(t)}{\tau}$ showing the number of nuclei $N(t)$ against time t. $N(t=0) = 10000$, $\tau = 0.0005$.

Free Pendulum, Euler method

```c
#include <stdio.h>
int main(){

  double thip1=0.2, wip1=0.0, dlt=0.04, zaman=0, tmp;
  int i;
  FILE *file = fopen("pendulum", "w");

  fprintf(file, "%e %e\n", zaman, thip1);
  for(i=0; i<500-1; i++){
    tmp = thip1;
    thip1 = thip1 + wip1*dlt;
    wip1 = wip1 + (-9.8*tmp - wip1)*dlt;
    zaman = zaman + dlt;
    fprintf(file, "%e %e %e\n", zaman, thip1, wip1);
  }

  fclose(file);
  return 0;
}
```

For plotting, use the file gplot.txt with content:

```
set term png

set output "pendulum_free.png"
plot "pendulum" using 1:2 title "theta" , "pendulum" using 1:3 title "velocity"

set output "pendulum_damped.png"
plot "damped.dat" u 1:2 t "theta" , "damped.dat" u 1:3 t "velocity"
```

Then execute on the terminal: $ gnuplot gplot.txt

Free Pendulum, Euler & Euler-Cromer methods

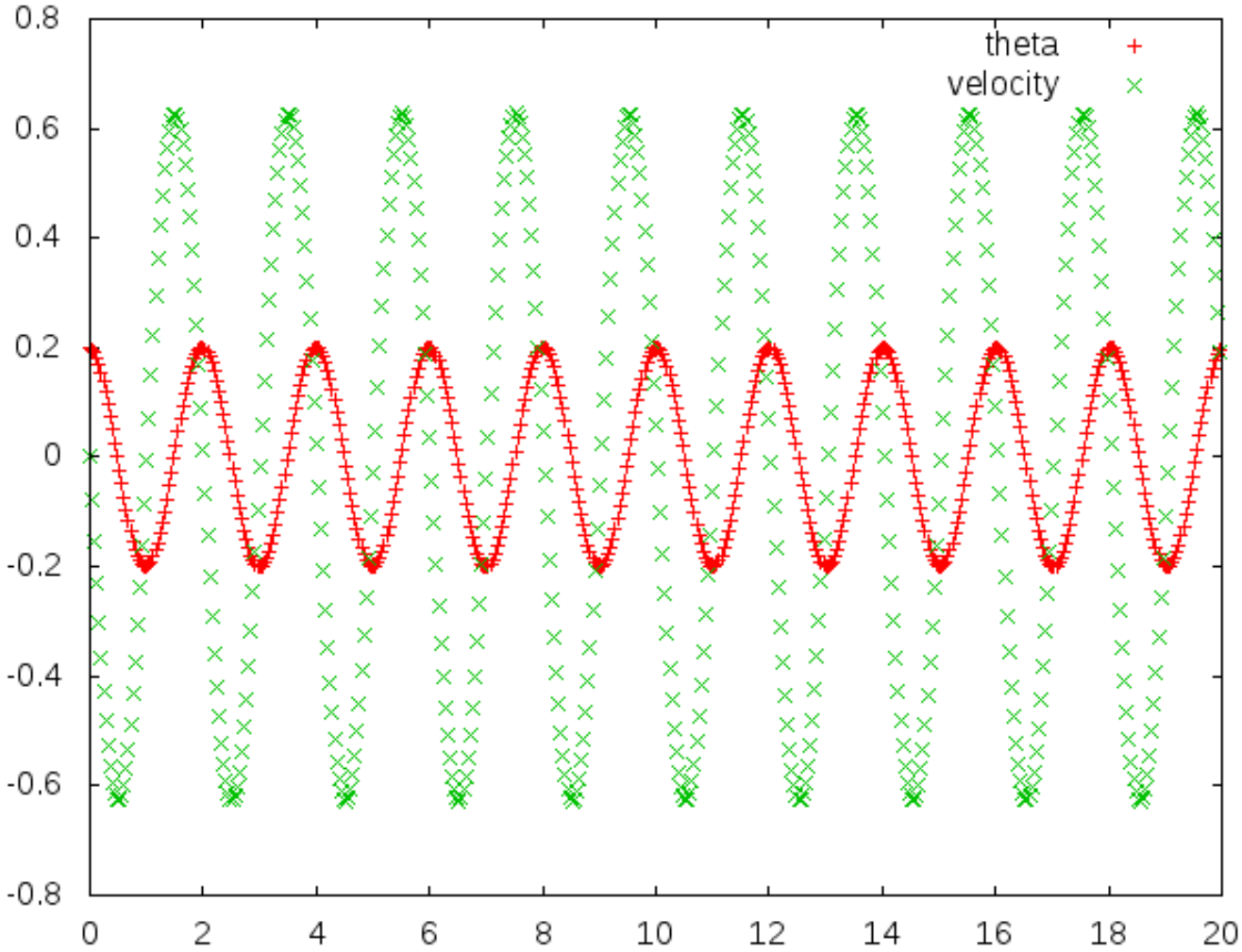


Figure : The displacement and speed with time of a simple pendulum.

Dampted Pendulum, Euler-Cromer method

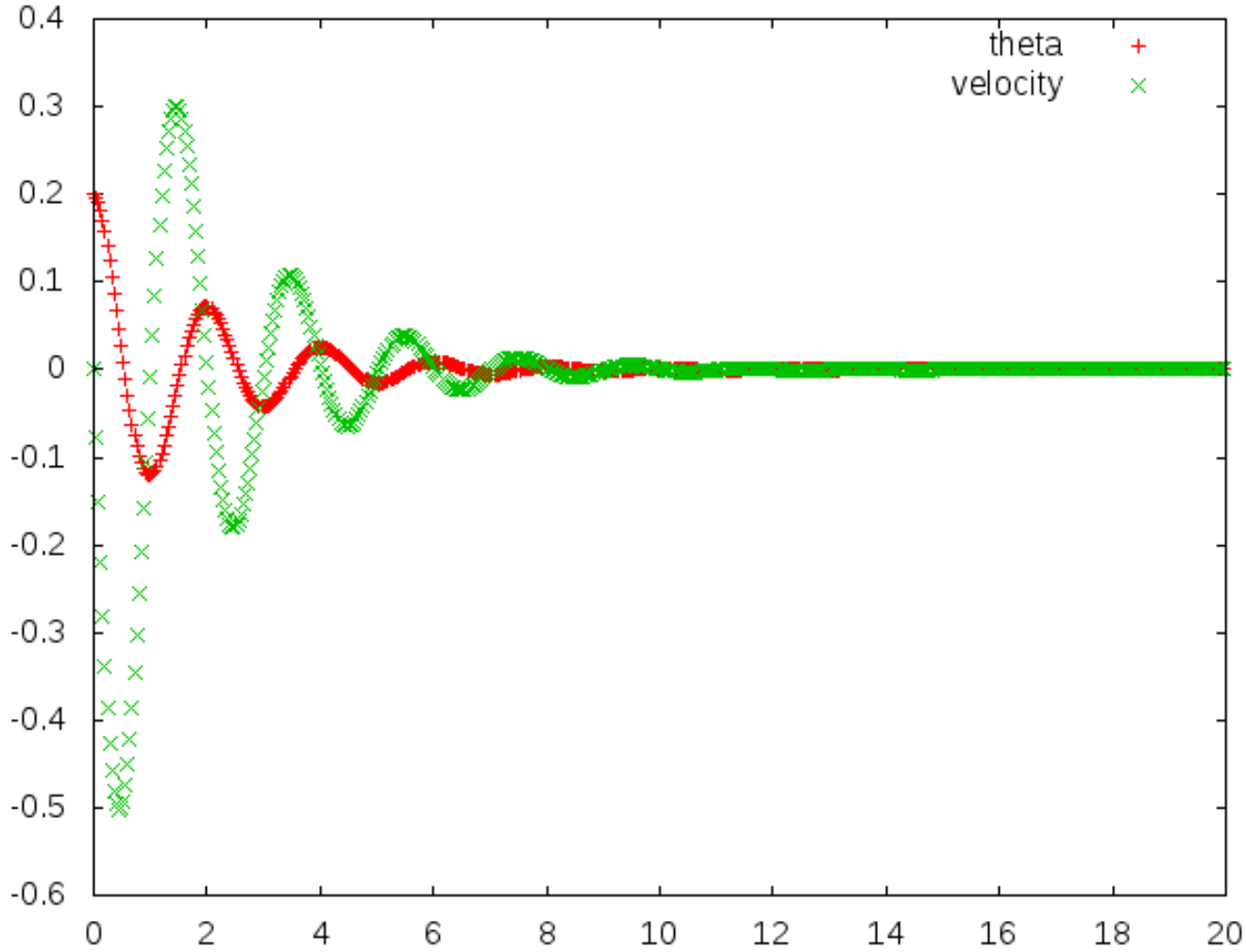


Figure : **Damped pendulum.**

Exercise 03

Use Euler-Cromer method to numerically solve for:
(1) a Sun-Earth system, and (2) a Sun-earth-Jupiter system. Let the Sun be stationary at the origin of a cartesian cordinate system. Use Newton's law in x- and y-directions for deriving the equations of motion. Then write C programs for solving the equations of motions. Plot your results to show the trajectories.

Sun-Earth-Jupiter

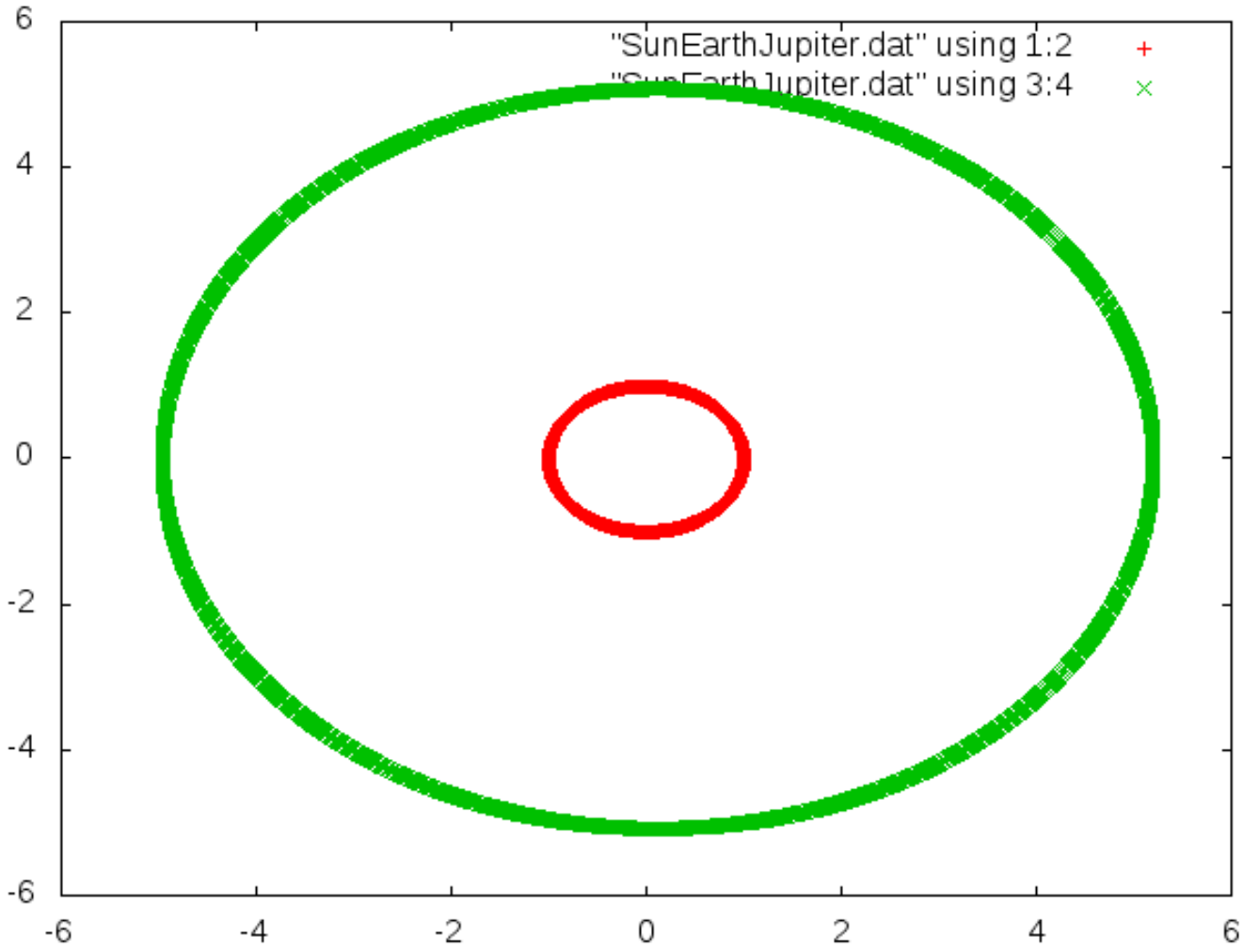


Figure : The Sun-Earth-Jupiter system.

Numerical Differentiation

Numerical Differentiation

Differentiation as a limit:

$$\frac{dy(t)}{dt} = \lim_{\Delta t \to 0} \frac{y(t+\Delta t) - y(t)}{\Delta t} \tag{1}$$

is not good for computing differentiation numerically or from a table of numbers. The numerator fluctuates between 0 and the computer's precision ϵ.

Given a table of projectile positions $y(t)$ taken at various time t during it's flight, the velocity can be estimated by the forward difference approximation via Taylor series:

$$y(t+\Delta t) = y(t) + \Delta t \frac{dy(t)}{dt} + \frac{(\Delta t)^2}{2!}\frac{d^2y(t)}{dt^2} + \dots \tag{2}$$

$$\left.\frac{dy}{dt}\right|_{fd} \approx \frac{y(t+\Delta t) - y(t)}{\Delta t} = \frac{dy(t)}{dt} + \frac{(\Delta t)}{2}\frac{d^2y(t)}{dt^2} + \dots \tag{3}$$

Numerical Differentiation

Similarly,

$$\left.\frac{dy}{dt}\right|_{bd} \approx \frac{y(t) - y(t - \Delta t)}{\Delta t} \tag{4}$$

Order $(\Delta t)^2$ error improvement is obtained via a central difference method which makes half-step forward and half-step backward

$$\left.\frac{dy}{dt}\right|_{cd} \approx \frac{y(t + \Delta t) - y(t - \Delta t)}{2\Delta t} \tag{5}$$

These numerical differentiation methods are important for solving partial differential equations.

Partial Differential Equations, Electrostatics & Quantum Mechanics

Project 2 (submit by 1396-02-17)

Task:

Consider a point-charge at the centre of a 3D box. Using Poisson's equation

$$\frac{\partial^2 V}{\partial x^2} + \frac{\partial^2 V}{\partial y^2} + \frac{\partial^2 V}{\partial z^2} = -\frac{\rho}{\epsilon_0}$$

and the boundary condition that $V = 0$ on the sides of the box, write a C program to compute and then plot the electric potential, V, and field, $\vec{E}$, inside the box.

Reference:

Chapter 5, "Computational Physics" by Nicholas J. Giordano and Hisao Nakanishi.

Partial Differential Equations

- Laplace & Poisson equations

- Relaxation algorithm (Jacobi method)

- Examples:
(1) Square box with potential overtwo sides,
(2) Hollow, square prism with metallic bar at centre
(3) Particle in a box, shooting method
(4) Particle in a Lennard-Jones potential, matching method

- Projects 2 and 3

Partial Differential Equations (PDE)

- Solving particle differential equations (PDEs) involves finding a function of many variables given a system and partial differential equations of the function and a set of boundary conditions.

- Applications in electrostatics, electrodynamics, heat, sound, fluid dynamics and quantum mechanics.

- We shall consider examples from:
(a) Electrostatics (relaxation algorithm, also called Jacobi method),
(b) Quantum mechanics (shooting and matching methods)

PDE: Electrostatics

- Possoin/Laplace equation,

$$\nabla^2 V(x,y,z) = -\frac{\rho(x,y,z)}{\epsilon_0} : \tag{6}$$

the problem is to find $V(x,y,z)$ which satisfies the PDF and a given boundary condition which specifies $V(x,y,z)$ on a surface in x-y-z-space. (Laplace equation, when $\rho = 0$.)

- There are various algorithms for solving partial differential equations, each depending on the problem at hand. Here, Jacobi method, also called relaxation algorithm will be introduced:
- Discretise (x,y,z) and specify points in space by integaers (i,j,k) such that $x = i\,\Delta x$, $y = j\,\Delta y$, and $z = k\,\Delta z$.

$$\frac{\partial^2 V}{\partial x^2} \approx \frac{1}{\Delta x}\left[\frac{\partial V}{\partial x}\Big|_{i+\frac{1}{2}} - \frac{\partial V}{\partial x}\Big|_{i-\frac{1}{2}}\right] = \frac{V(i+1,j,k) - 2V(i,j,k) + V(i-1,j,k)}{(\Delta x)^2}. \tag{7}$$

so that assuming $\Delta x = \Delta y = \Delta z$

PDE: Electrostatics. Jacobi/relaxation method

$$V(i,j,k) = \frac{1}{6}[V(i+1,j,k) + V(i-1,j,k) + V(i,j+1,k) + V(i,j-1,k) + V(i,j,k+1) + V(i,j,k-1)] + \frac{\rho(i,j,k)\,(\Delta x)^2}{6\epsilon_0} : \quad (8)$$

Numerical strategy:

- take an initial guess to solution. Let's call it $V_0(i,j,k)$
- Use V_0 on the r.h.s. of Eq.(8) to obtain an improved solution, V_1.
- Iterate until the solution satisfies convergence condition: e.g., difference between V at grid points is very small compared to the boundary values.

- Once V is determined, the electric field can be computed numerically: $\vec{E} = -\vec{\nabla} V$

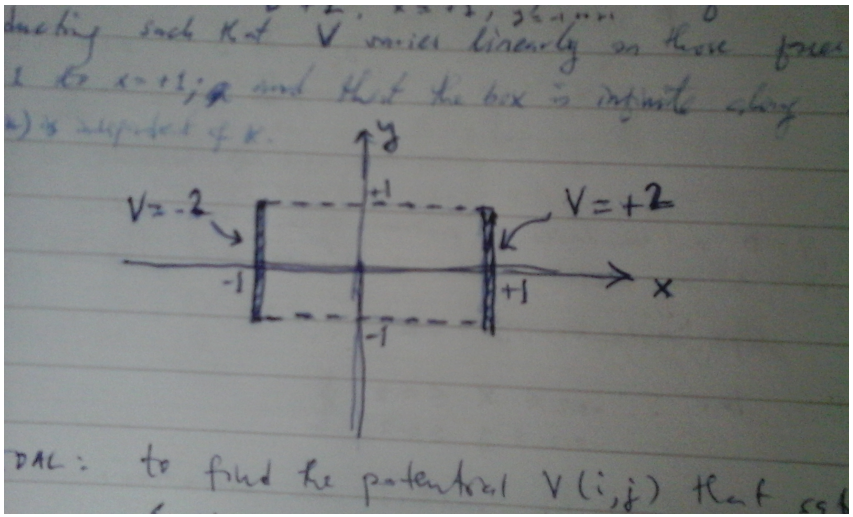
PDE: Electrostatics

Example(1): electric field inside a square box.

Compute the electric potential inside the square box shown below. The boundary conditions are

$$V(x,y) = \begin{cases} -2 & \text{if } x = -1, \\ 2 & \text{if } x = +1. \end{cases}$$

The sides of the box at $y = \pm 1$ are non conducting but assumes V varies linearly from $x = -1$ to $x = 1$. Assume also that the box is infinte along z-direction so that $V(i,j,k)$ is independent of k.



PDE: Electrostatics

Example(1): electric field inside a square box.

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

#define NPOINTS 7
#define prec NPOINTS * NPOINTS * 1.0e-9
double V[NPOINTS][NPOINTS];
double Vnp1[NPOINTS][NPOINTS];

void Initialise(void);
double UpdateV(void);
void ElectricField(void);

int main(){
  long int iterations = 1;
  double deltaV;

  Initialise();
  while( iterations !=0 ) {
    iterations = iterations + 1;
    deltaV = UpdateV();
    if(deltaV <= prec) break;
  }

  ElectricField();
  printf("FINISHED: deltaV = %e  prec = %e   iterations = %ld\n",
         deltaV, prec, iterations);
  return 0;
}
```

PDE: Electrostatics

```
void Initialise(void){
  int i, j;

  for( i = 0; i < NPOINTS; i++)
    for( j = 0; j < NPOINTS; j++){
      if(j == 0){
        V[i][j] = -2.0;
        Vnp1[i][j] = -2.0;
      } else if(j == NPOINTS-1){
        V[i][j] = 2.0;
        Vnp1[i][j] = 2.0;
      }
      else V[i][j] = 0.0;
    }

  for( i = 0; i < NPOINTS; i++)
    for( j = 0; j < NPOINTS; j++){
      if(i != 0 && i != NPOINTS-1){
        V[0][i] = V[0][i-1] + 0.66;
        V[NPOINTS-1][i] = V[NPOINTS-1][i-1] + 0.66;
        Vnp1[0][i] = V[0][i-1] + 0.66;
        Vnp1[NPOINTS-1][i] = V[NPOINTS-1][i-1] + 0.66;
      }
    }
}
```

PDE: Electrostatics

```
double UpdateV(void){
  double deltaV = 0.0;
  int i, j;
  for( i = 1; i < NPOINTS-1; i++)
    for( j = 1; j < NPOINTS-1; j++){
      Vnp1[i][j] = ( V[i+1][j] + V[i-1][j] + V[i][j+1] + V[i][j-1] )/4.0;
      deltaV = deltaV + fabs( Vnp1[i][j] - V[i][j] );
    }

  for( i = 1; i < NPOINTS-1; i++)
    for( j = 1; j < NPOINTS-1; j++) V[i][j] = Vnp1[i][j];

  return deltaV;
}
```

PDE: Electrostatics. Example (1)

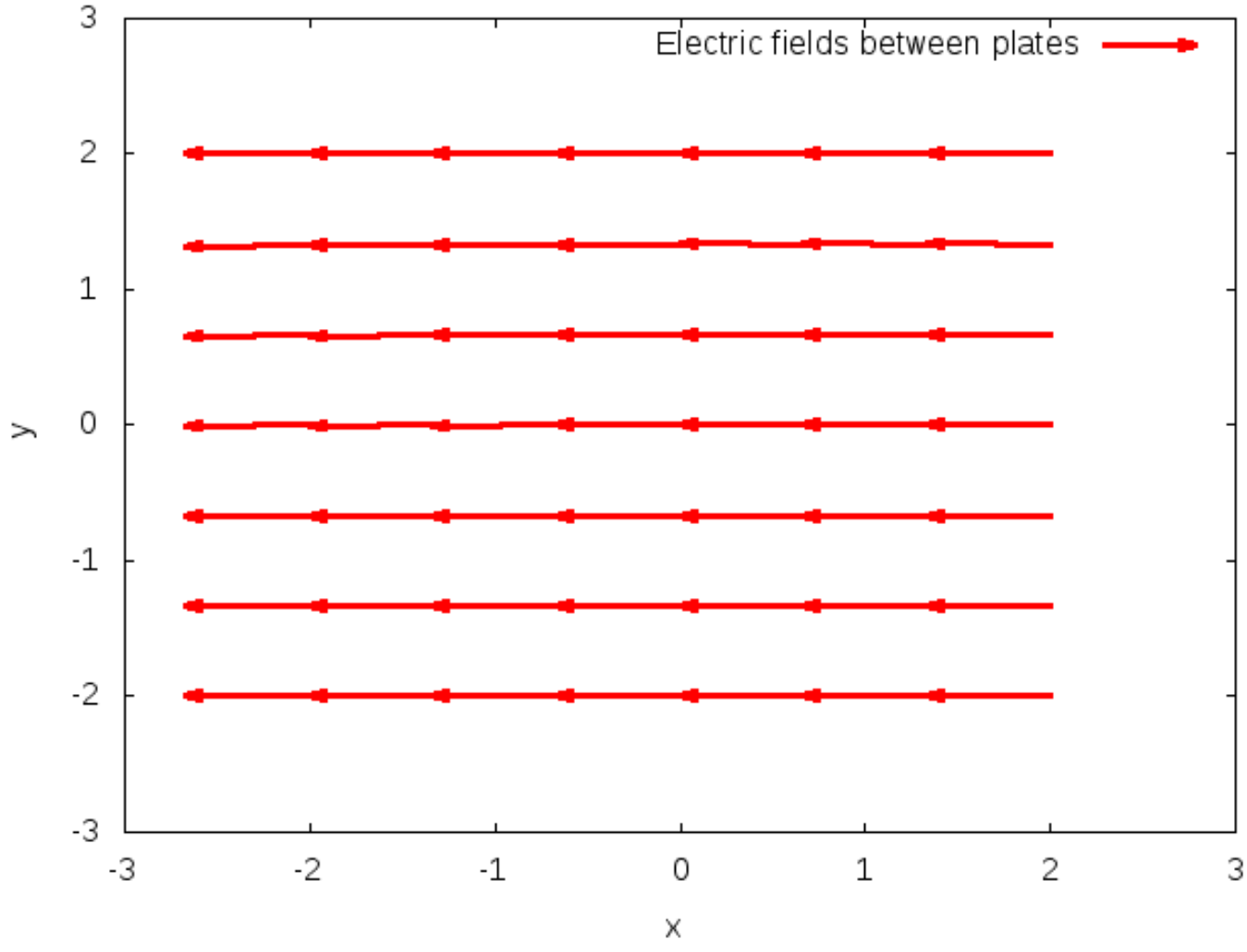
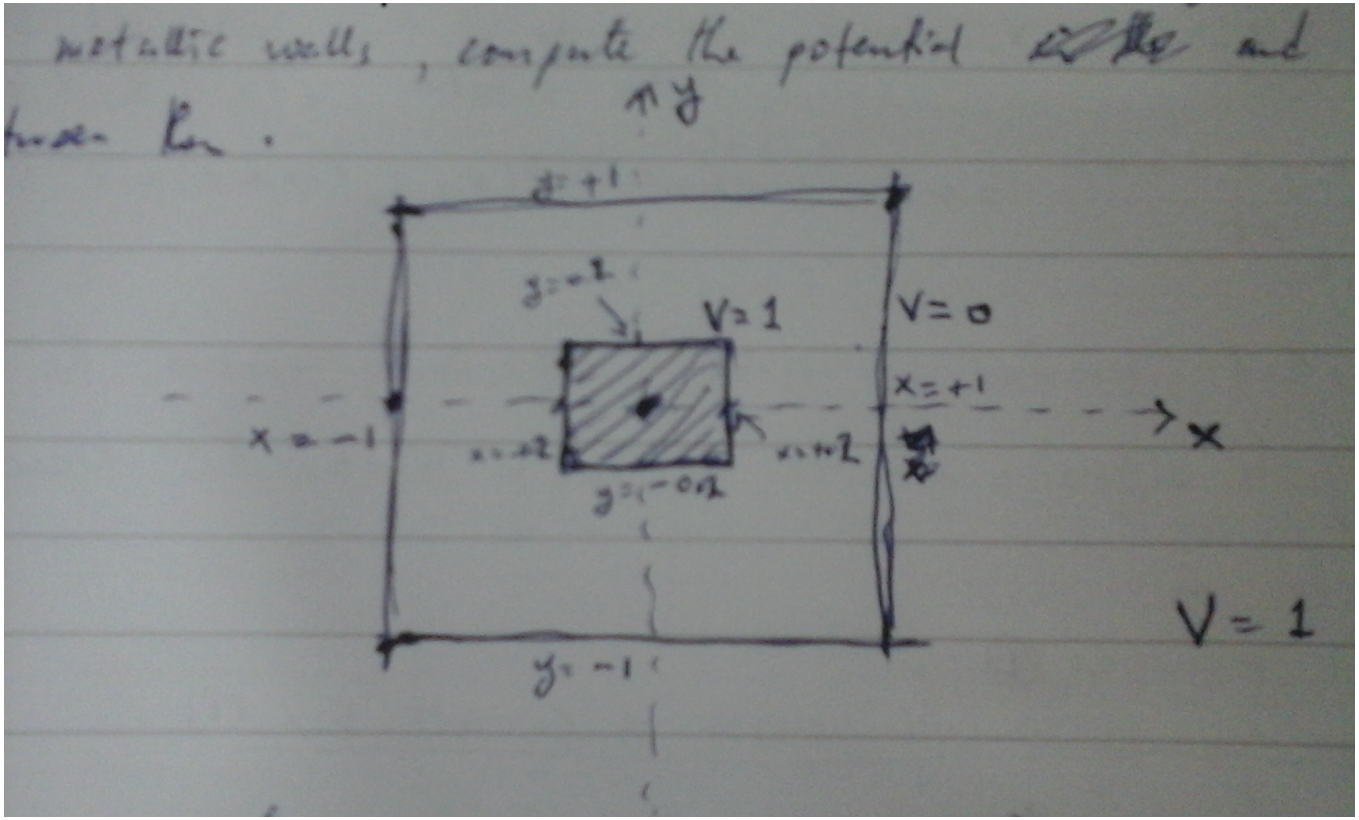


Figure : Electric fields between plates.

PDE: Electrostatics

Example(2): electric field inside a hollow square box with metallic bar at the centre.
Compute the electric potential in the region, as sketched below, between the walls of the hollow square box and the metallic bar.



PDE: Electrostatics. Example (2)

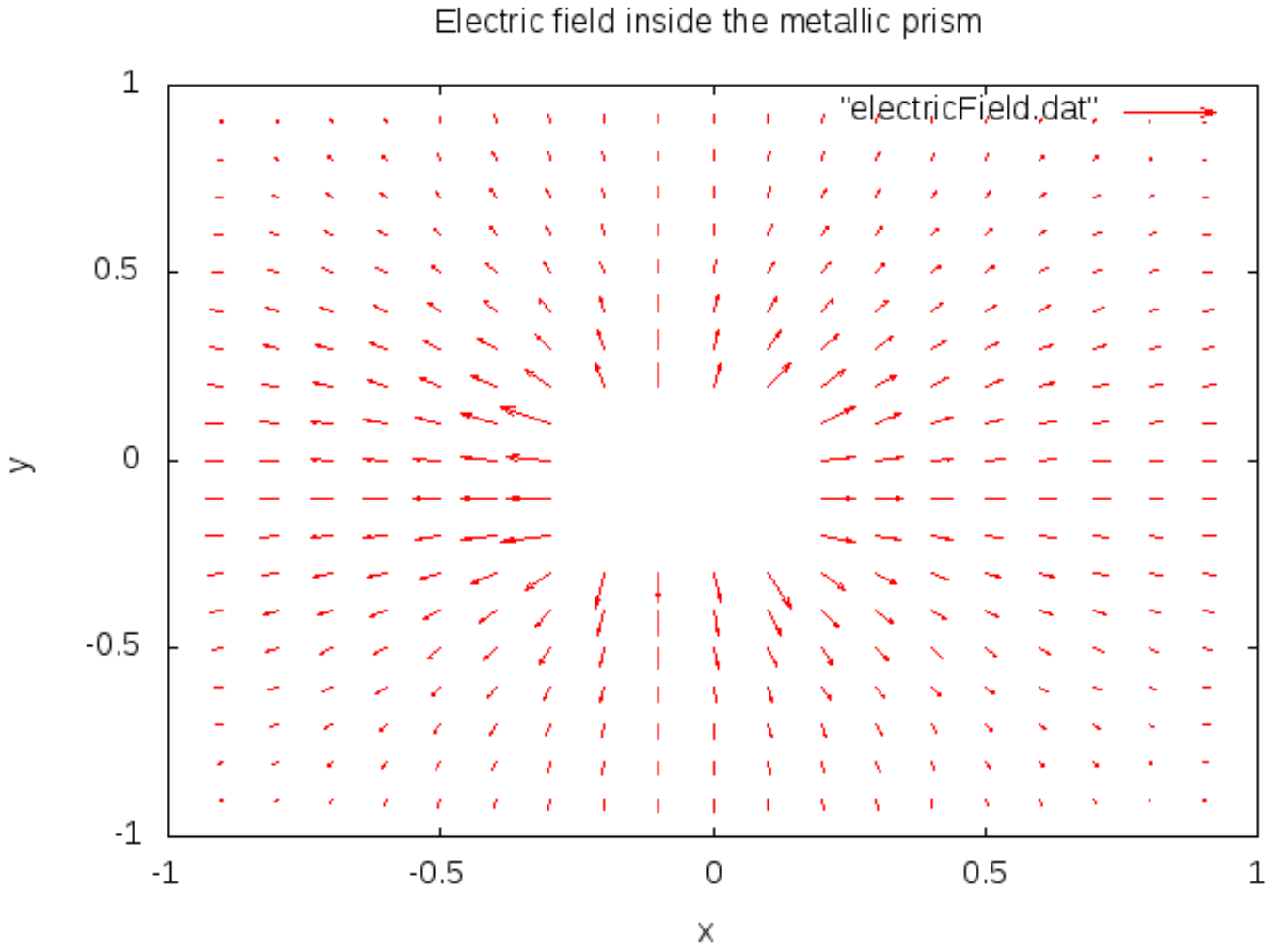


Figure : Electric fields inside hollow prism with metallic bar at centre.

PDE: Quantum Mechanics

- The time-independent Schrödinger equation

$$-\frac{\hbar^2}{2m}\nabla^2\Psi + V(r)\Psi = E\Psi \tag{9}$$

is a PDE for the wavefunction Ψ of a particle within a potential $V(r)$. The task is to find Ψ and E given the potential energy, $V(r)$.

- We shall consider two examples for illustrating how the PDE, eq.(9) can be solved numerically. The first and second example will respectively be for a particle in a box using shooting method and the matching method for a particle within a Lennard-Jones potential.

PDE: Quantum Mechanics

Example(1): Shooting Method. Find the wavefunction and the corresponding energy for the ground state of a particle in one-dimensional box[1]:

$$V(x) = \begin{cases} 0, & |x| \leq L, \\ \infty, & \text{otherwise} \end{cases} \tag{10}$$

- Descritise the independent variable x, $x_n = n\Delta x, n = 0, 1, 2, \ldots$.
- Descritise the $V(x)$ such that $V_n = V(n\Delta x)$ and PDE, eq.(9):

$$-\frac{\hbar^2}{2m}\frac{d^2\Psi}{dx^2} \approx -\frac{\hbar^2}{2m}\left[\frac{\Psi_{n+1} - 2\Psi_n + \Psi_{n-1}}{(\Delta x)^2}\right] \approx (E - V_n)\Psi_n$$

or assuming $m = 1, \hbar = 1$,

$$\Psi_{n+1} = 2\Psi_n - \Psi_{n-1} - 2(\Delta x)^2(E - V_n)\Psi_n \tag{11}$$

- Use symmetry of potential, $V(x)$ to find initial conditions.

[1] The purpose here is to show/learn the computational techniques starting with simple case.

PDE: Quantum Mechanics

Example(1) Initial conditions:
- $V(x)$ is symmetric, $\Psi(x)$ can be either and even or odd.
- $\Psi(x) = \Psi(-x) \Rightarrow \frac{d\Psi(x)}{dx}\Big|_{x=0} = 0 \quad \& \quad \Psi(x)\Big|_{x=0} = 1$
- $\Psi(x) = -\Psi(-x) \Rightarrow \frac{d\Psi(x)}{dx}\Big|_{x=0} = 1 \quad \& \quad \Psi(x)\Big|_{x=0} = 0.$
- For the example here, the even solution is addressed.

Example(1) Boundary conditions:
$V(x = \pm L) = \infty \Rightarrow \Psi(x = \pm L) = 0$

- shooting method: solving eq.(9) with the boundary conditions that must be satisfied at both ends of x-interval. This method for cornering the correct energy (eigen value) and wavefunction (eigen function) simultaneously is shown below.

PDE: Quantum Mechanics

Example(1) Pseudocode:

- Initialise Ψ and descretise the potential on a grid.
- Function for computing Ψ using eq.(13) after initialisation.
- main code: initialise, call the function for computing Ψ and then implement the shooting method:

```
void Initialise(void );
int ComputePsi(double );
int SignOf(double );

int main(){
  // initial guess for ground state energy, E = 0.0
  double E = 0., dE = 0.001, dE_precision = 0.00005;
  int i, diverge_direction, signPsi;
  FILE *file = fopen("PsiE0.txt", "w");

  Initialise();
  diverge_direction = ComputePsi(E);

  while( fabs(dE) > dE_precision){
    signPsi = ComputePsi(E);
    if(diverge_direction != signPsi)
      dE = - dE/2.0;
    E = E + dE;
    diverge_direction = signPsi;
  }

  printf("FINISHED: E = %e  Psinp1[99] = %e\n", E, Psi[N/2-1]);
  for(i = 0; i < N/2; i++) fprintf(file, "%e %e\n", x0+i*dx, Psi[i]);
  fclose(file);
  return 0;
}
```

PDE: Quantum Mechanics. Example (1)

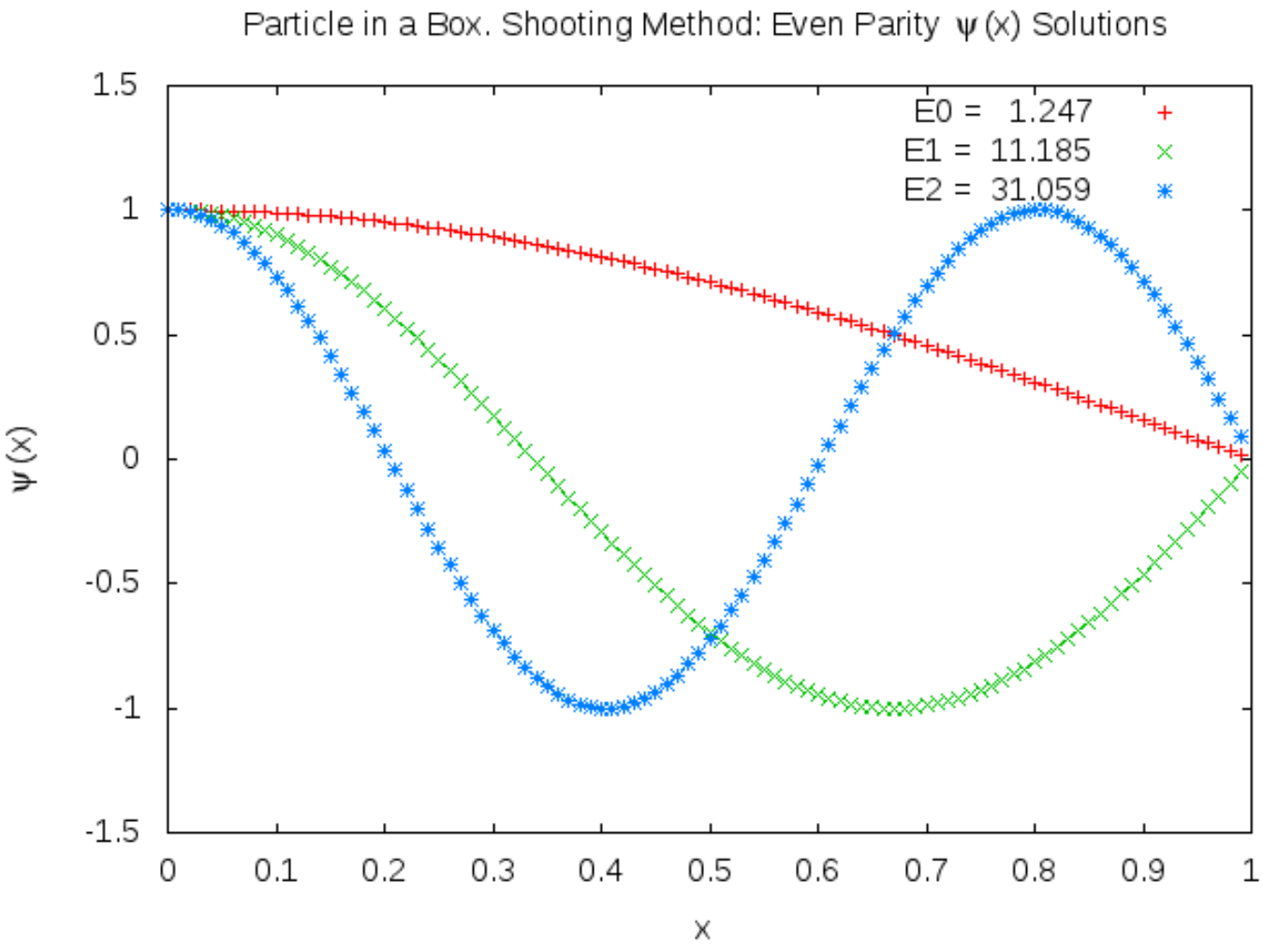


Figure : First three energy levels webfunctions (not normalised) for a particle confined in a box.

PDE: Quantum Mechanics

Example(2): Matching Method. The task is same as for example(1) only that here we use Lennard-Jones potential,

$$V(x) = 4\epsilon\left[\left(\frac{\sigma}{x}\right)^{12} - \left(\frac{\sigma}{x}\right)^{6}\right] \quad \text{use:}\epsilon = 10, \sigma = 1, \qquad (12)$$

instead of the "box".

- Descritise the independent variable x, $x_n = n\Delta x, n = 0, 1, 2, \ldots$.
- Descritise the $V(x)$ such that $V_n = V(n\Delta x)$ and PDE to:

$$\Psi_{n+1} = 2\Psi_n - \Psi_{n-1} - 2(\Delta x)^2(E - V_n)\Psi_n \qquad (13)$$

- For matching method, two different trial wavefunctions, Ψ_L and Ψ_R are considered. Ψ_L is computed by starting from say $x = x_0 = x_L = 0.9$ with $x_{n+1} = x_n + \Delta x$. While Ψ_L is computed by starting from $x = x_0 = x_R = 2.5$ with $x_{n+1} = x_n - \Delta x$. $\Psi_L = \Psi_R$ at matching point $x = x_{match} = 1.1$.

Example(2) Matching conditions:

- $\Psi_L(x = x_{match}) = \Psi_R(x = x_{match})$. If necessary, scale one of the wavefunctions so they match since constant scaling of a solution is another solution.

- The final wavefunction to be taken as solution should be continuous at the chosen matching/intersection point:
$\frac{d\Psi_L(x)}{dx}\Big|_{x=x_{match}} = \frac{d\Psi_R(x)}{dx}\Big|_{x=x_{match}}$

Example(2) Matching method for cornering the correct energy, E:

(a) Choose a value, ΔE, for increasing/decreasing the initial choise of energy, E. Compute Ψ_L, Ψ_R and check the matching conditions.

(b) Let a variable *match* be equal to $+1$ or -1 depending on whether $\frac{d\Psi_L(x)}{dx}\Big|_{x=x_{match}} > \frac{d\Psi_R(x)}{dx}\Big|_{x=x_{match}}$ or not.

(c) If *match* is different compared to its value from a previous trial of E, then let ΔE be changed to $-\frac{1}{2}\Delta E$. Otherwise leave ΔE.

(d) Change E to become $E + \Delta E$.

(e) Repeat the above steps until ΔE becomes less than a chosen precision.

PDE: Quantum Mechanics. Example (2)

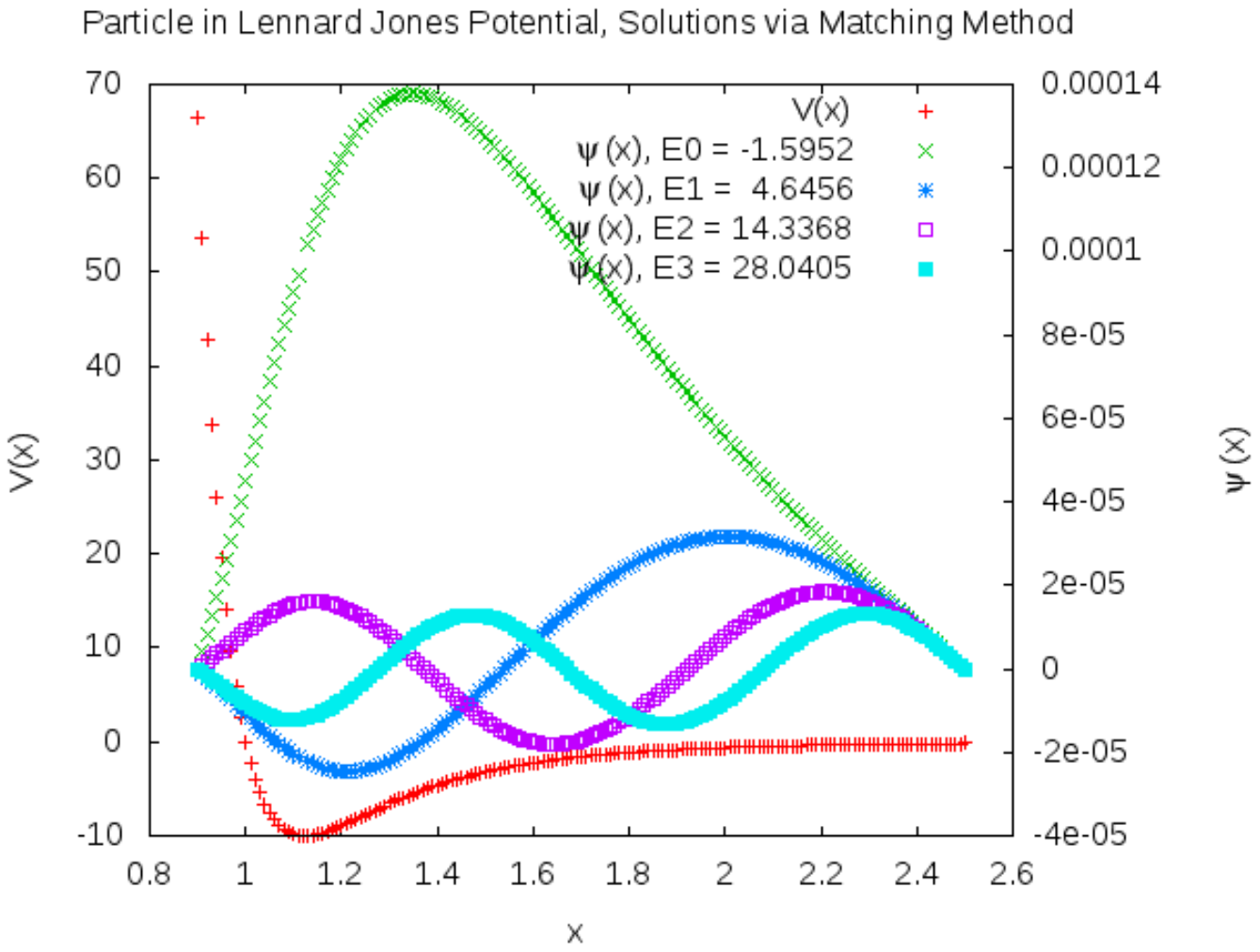


Figure : First four energy levels webfunctions (not normalised) for a particle confined in a Lennard-Jones potential $V(x)$.

Project 3 (submit by 1396-02-31)

Task:

Write a C programming for numerically computing the ground state wavefunction and energy for the quantum mechanics PDE example (2).

Reference:

Chapter 10, "Computational Physics" by Nicholas J. Giordano and Hisao Nakanishi.

Numerical Integration

Numerical Integration

The integral of $f(x)$ from a to b is the area under the curve $f(x)$ to the x-axis. It can be approximated as sum of the area of n rectangular panels with width $\Delta x = (b - a)/n$:

$$\int_a^b f(x)\, dx \approx \sum_{i=0}^{n-1} f(x_i)\, \Delta x. \tag{14}$$

Using trapeziodal panels instead, get a better estimate:

$$\int_a^b f(x)\, dx \approx \sum_{i=1}^{n-1} f(x_i)\, \Delta x + [f(a) + f(b)]\, \Delta x/2. \tag{15}$$

Instead of using equal-width panels, even better approximations can be achieved by better approximation of the panels' upper edges.

Numerical Integration

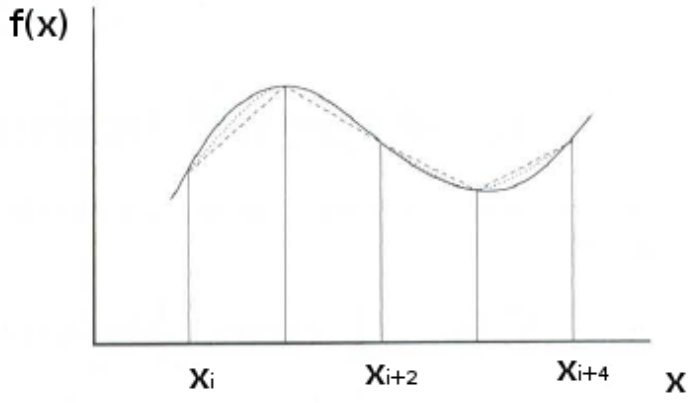


Figure : Integration algorithms using Trapezoid (dashed lines) or Simpson's (dotted line) rule.

Consider $n+1$ evenly-spaced points x_i, $i = 0, 1, 2, \ldots, n$ over the integration region $[a, b]$. These make n panels of width $\Delta x = \frac{b-a}{n}$ such that $x_i = a + i\,\Delta x$.

$$\text{Trapezoid:} \quad \int_{x_i}^{x_{i+1}} f(x)\,dx \approx \frac{\Delta x}{2} f_i + \frac{\Delta x}{2} f_{i+1} \qquad (16)$$

The integral is thus approximated as weighted sum over values of the function at two points.

Numerical Integration

Simpson'r rule employs a parabolic approximate to $f(x)$: $f(x) \approx \alpha x^2 + \beta x + \gamma$ such that

$$\int_{x_i}^{x_{i+1}} f(x)\, dx \approx \left(\frac{\alpha x^3}{3} + \frac{\beta x^2}{2} + \gamma x \right) \Big|_{x_i}^{x_{i+1}} \tag{17}$$

For relating α, β, γ to the integrand, $f(x)$, consider the interval -1 to $+1$ for which it can be shown that

$$\int_{-1}^{+1} (\alpha x^2 + \beta x + \gamma)\, dx = \frac{f(-1)}{3} + \frac{4f(0)}{3} + \frac{f(1)}{3}. \tag{18}$$

That is, the integration is approximated as weighted sum over values of the function at three points. This generalises to:

Numerical Integration

$$\int_{x_{i-1}}^{x_{i+1}} f(x)\,dx = \int_{x_{i-1}}^{x_i} f(x)\,dx + \int_{x_i}^{x_{i+1}} f(x)\,dx \tag{19}$$

$$\approx \frac{\Delta x}{3} f_{i-1} + \frac{4\Delta x}{3} f_i + \frac{\Delta x}{3} f_{i+1}. \tag{20}$$

The integration is over pairs of panels as such total number of panels $(n-1)$ should be an even number.

There are other higher orders algorithms such as Gaussian quadrature. Here we move next to the integration via Monte Carlo method.

Introduction to Monte Carlo Methods

Monte Carlo Integration

- Compute π, using random numbers

- Probabilities that randomly thrown points land within a region are proportional to the area of the region

- Therefore, get area under a curve from ratio of random points that fall below it over total number of points thrown.

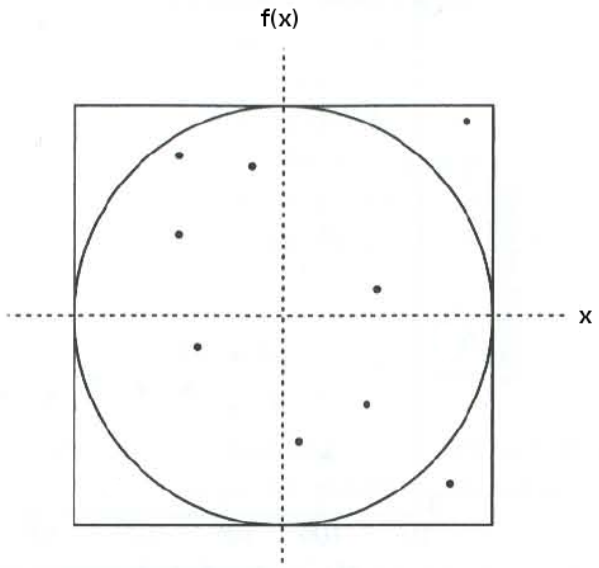


Figure : Computing area under $f(x)$ within interval $x \in [-1, 1]$ by using random numbers.

Monte Carlo Integration

The standard Monte Carlo integration algorithm is based on mean value theorem:

$$I = \int_a^b f(x)\,dx = (b-a)\,\langle f \rangle \tag{21}$$

where the average of $f(x)$ over region of interest, $\langle f \rangle = \frac{1}{n}\sum_{i=1}^{n} f_i$.

Therefore by generating n points randomly in $[a, b]$ then

$$\int_a^b f(x)\,dx \approx \frac{b-a}{n}\sum_{i=1}^{n} f(x_i) \pm \frac{\sigma_f}{\sqrt{n}}, \quad \sigma_f = \left\langle f^2 \right\rangle - \langle f \rangle^2 \tag{22}$$

Monte Carlo Integration

```c
#include <stdio.h>
#include <math.h>
#include <stdlib.h>

double a = -1.0, b = 1.0;

double f(double x){ return 1.0 - x*x; }

double trapezoid(double f(double), double xi, double xip1){
  return (xip1 - xi) * (f(xi) + f(xip1))/2.0;
}

double simpson(double f(double), double xi, double xip2){
  double xip1 = xi + (xip2 - xi)/2.0;
  return (xip1 - xi) * (f(xi) + 4.0 * f(xip1) + f(xip2)) / 3.0;
}

double montecarlo(double f(double), double a, double b, int n){
  double integ = 0.0;
  double random_0_to_1, xi;
  int i;
  srand(time(0));
  for (i = 0; i < n; i++) {
    random_0_to_1 = rand()/(double)RAND_MAX;
    xi = a + (b - a) * random_0_to_1;
    integ = integ + f(xi);
  }
  integ = integ * (b - a) / n;
  return integ;
}
```

Monte Carlo Integration

```
33  int main(){
34    int n=100001, i;
35    double integ = 0.0;
36    double xi, xip1, xip2;
37    double deltax = (b - a) / n;
38
39    printf(" (1) Trapezoid:\t\tinteg = ");
40    xi = a;
41    for(i = 1; i <= n-1; i++){
42      xip1 = xi + deltax;
43      integ = integ + trapezoid(f, xi, xip1);
44      xi = xip1;
45    }
46    integ = integ + (f(a)+f(b))*deltax/2.0;
47    printf("%e \n", integ);
48
49    printf(" (2) Simpson's:\t\tinteg = ");
50    integ = 0.0;
51    for(i = 0; i < n; i++){
52      xi = a + i*deltax;
53      xip2 = xi + deltax;
54      integ = integ + simpson(f, xi, xip2);
55    }
56    printf("%e \n",integ);
57
58    printf(" (3) Monte Carlo:\tinteg = ");
59    printf("%e \n", montecarlo(f, a, b, n));
60
61    printf("\n\n\n");
62    printf(" Tamam. Khoda hafez.\n\n");
63    return 0;
64  }
```

Quantum Mechanics, Variational-Monte Carlo

- The shooting and matching methods are not good for solving 2-d or 3-d Schroëdinger equations.

- A better approach is the variational principle on which Monte Carlo methods can be applied.

- Variational principle: given a trial wavefunction, ψ_t, the corresponding energy

$$E = \frac{\int \psi_t^* \mathcal{H} \psi_t dx}{\int \psi_t^* \psi_t dx} \tag{23}$$

is always greater than the system's true ground state energy, $E_0 \leq E$. The Hamiltonian, $\mathcal{H} \equiv -\frac{\hbar^2}{2m}\frac{\partial^2}{\partial x^2} + V(x)$.

- The task here is to find the wavefunction, ψ, which minimises E.

- Monte Carlo integration methods can be applied for computing E.

- Variational-Monte Carlo 1-d example: particle bounded by Lennard-Jones potential.

- Let the trial webfunction be

$$\psi_t(x) = \begin{cases} 0 & \text{if } 0.9 \leq x < 1, \quad \& \quad 2.4 \leq x \leq 2.5 \\ a & \text{if } 1 \leq x < 2.4 \end{cases} \tag{24}$$

where a is such that $\int \psi_t^* \, \psi_t dx = 1$. Improved trial wavefunctions can be obtained as follows. Compute $E = \int \psi_t^* \, \mathcal{H} \, \psi_t dx$.

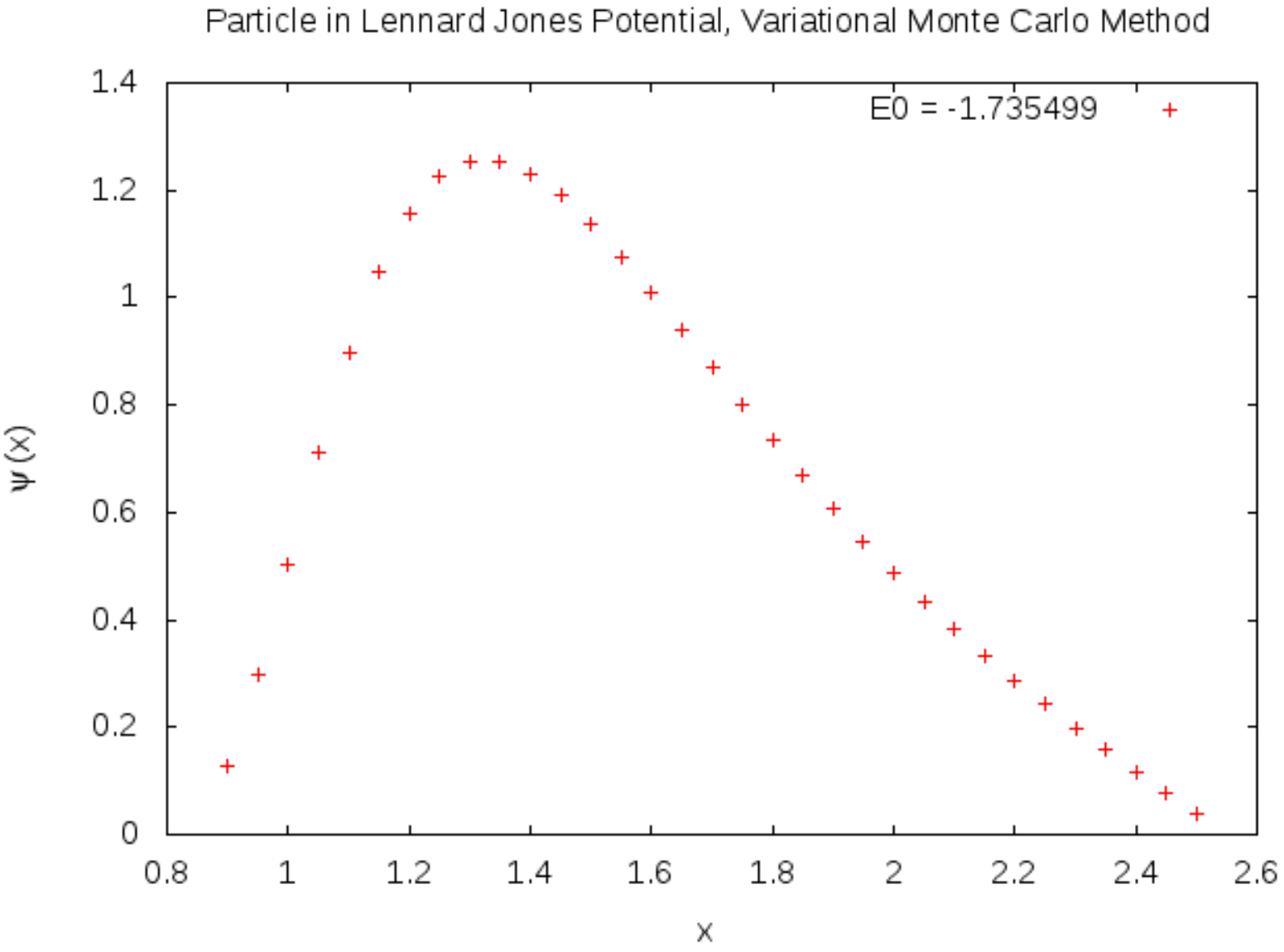
- Randomly pick a point $x = x'$ and let $\psi_t^{new}(x')$ be changed by adding a number chosen randomly in $-d\psi$ to $+d\psi$ with $d\psi = 0.05\, a$. Normalise ψ_t^{new} and compute $E_{new} = \int \psi_t^* \, \mathcal{H} \, \psi_t dx$. If $E_{new} < E$ then let $\psi_t = \psi_t^{new}$. Repeat the procedure until convergence, say: $(E - E_{new})/E_{first\_trial} < 10^{-14}$.

Quantum Mechanics, Variational-Monte Carlo

```
60  double normalisePsi(void){
61    double sum = 0.;
62    int i;
63    for(i = 0; i < N; i++)
64      sum = sum + Psi[i] * Psi[i] * dx;
65
66    if(sum != 0){
67      for(i = 0; i < N; i++)
68        Psi[i] = Psi[i] / sqrt(sum);
69      return 1.0/sqrt(sum);
70    } else {printf("Error: division by zero.\n"); exit(12);}
71  }

73  double computeEstar(void){
74    double energy = 0., Psi_ip1, Psi_im1;
75    int i;
76    for(i = 0; i < N; i++){
77      if(i==0) Psi_im1 = 0;
78      else Psi_im1 = Psi[i-1];
79      if(i==N-1) Psi_ip1 = 0;
80      else Psi_ip1 = Psi[i+1];
81      energy = energy
82        - dx * 0.5*Psi[i]*(Psi_ip1 + Psi_im1 - 2*Psi[i])/(dx*dx)
83        + Psi[i] * V[i] * Psi[i] * dx;
84    }
85    return energy;
86  }
```

Quantum Mechanics, Variational-Monte Carlo



Quantum Mechanics, Variational-Monte Carlo

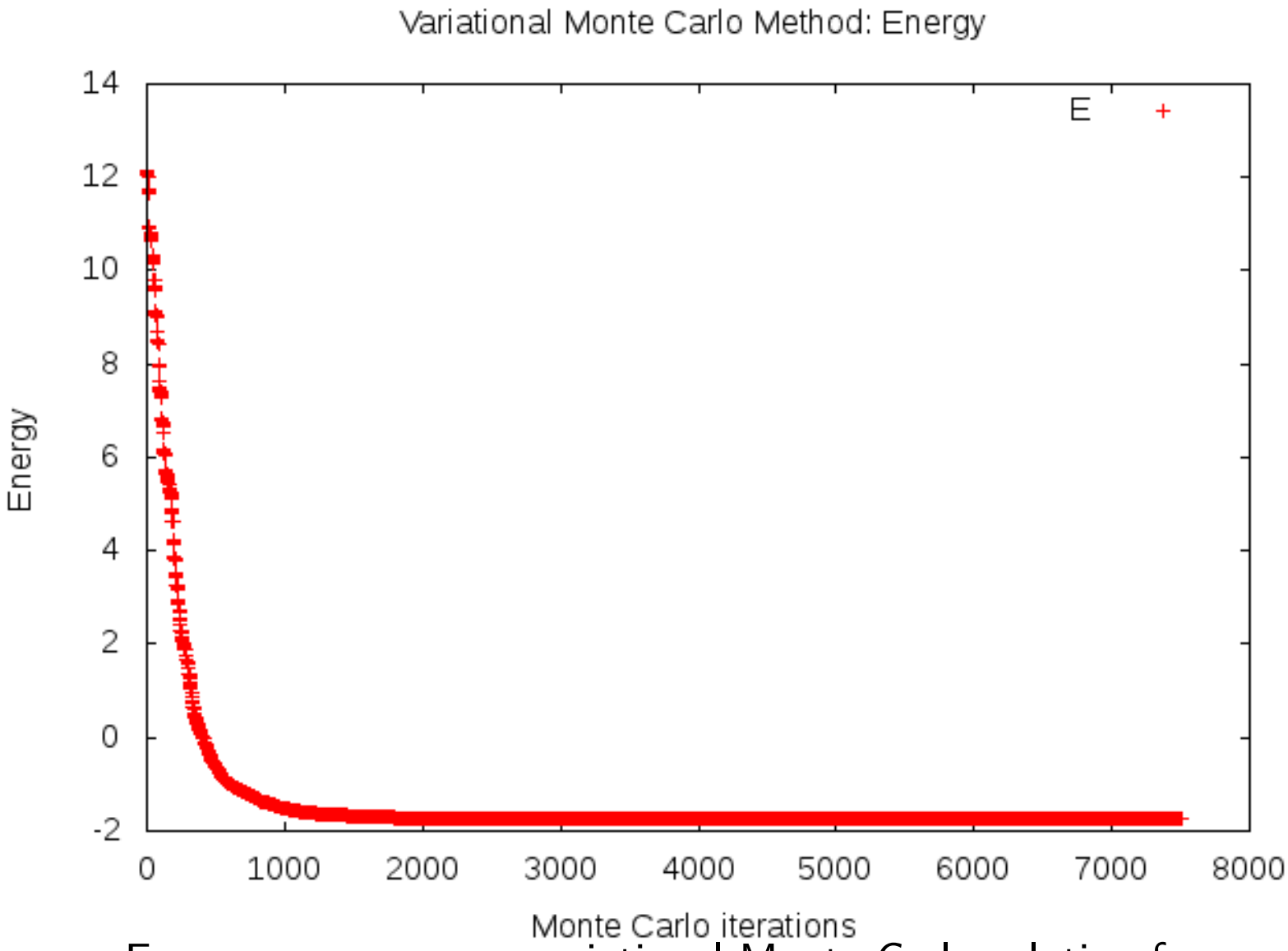


Figure : Energy convergence: variational Monte Carlo solution for a particle bound in Lennard-Jonnes potential.

Quantum Mechanics, Variational-Monte Carlo

- Real strength of Variational-Monte Carlo is for application to two- or three-dimensional problems.

- For illustration, consider the two-dimensional harmonic oscillator potential, $V(x,y) = \frac{1}{2}k_x\,x^2 + \frac{1}{2}k_y\,y^2$

- Let $x = i\,\Delta x$ and $y = j\,\Delta y$ (take $\Delta y = \Delta x$=0.2) with $-N\Delta x \le x \le N\Delta x$ and $-N\Delta y \le y \le N\Delta y$ with total of $(2N+1)^2$ grid-points. For example consider $k_x = 10$, $k_y = 40$, $|x|$, $|y|$ upto 2.0.

- Let the first trial wavefunction be a simple

$$\psi_t(x,y) = \begin{cases} a & \text{if } |x| < 1.6, \quad \& \quad |y| < 1.6 \\ 0 & \text{otherwise} \end{cases} \tag{25}$$

where a is such that $\int \psi_t^*\,\psi_t dx = 1$.

- The Monte Carlo algorithm can be implemented as in the previous example with Lennard-Jones potential.

Random Walker & Diffusion

Random Walk Monte Carlo

Consider an equally spaced lattice points on a line. At each point a walker can move right or left $s = \pm 1$ at equal probabilities.

Walker's position, x_n, after n steps is $x_n = \sum_{i=1}^{n} s_i$ which averages to zero over a large number of steps. But $\langle x_n^2 \rangle = n$.

An algorithm: generate a random number r between 0 and 1 and then compare the generated values to 0.5. If $r < 0.5$ the walker moves left ($s = -1$). If $r > 0.5$, it moves right ($s = +1$).

To write a code:
- seed a random number generator
- declare a file where to save x_n and $< x_n^2 >$
- loop over number of walkers; another loop over number of steps
- inner loop: perform the Monte Carlo algorith above
- after both loops, compute the averages and print to file

Random Walk Monte Carlo

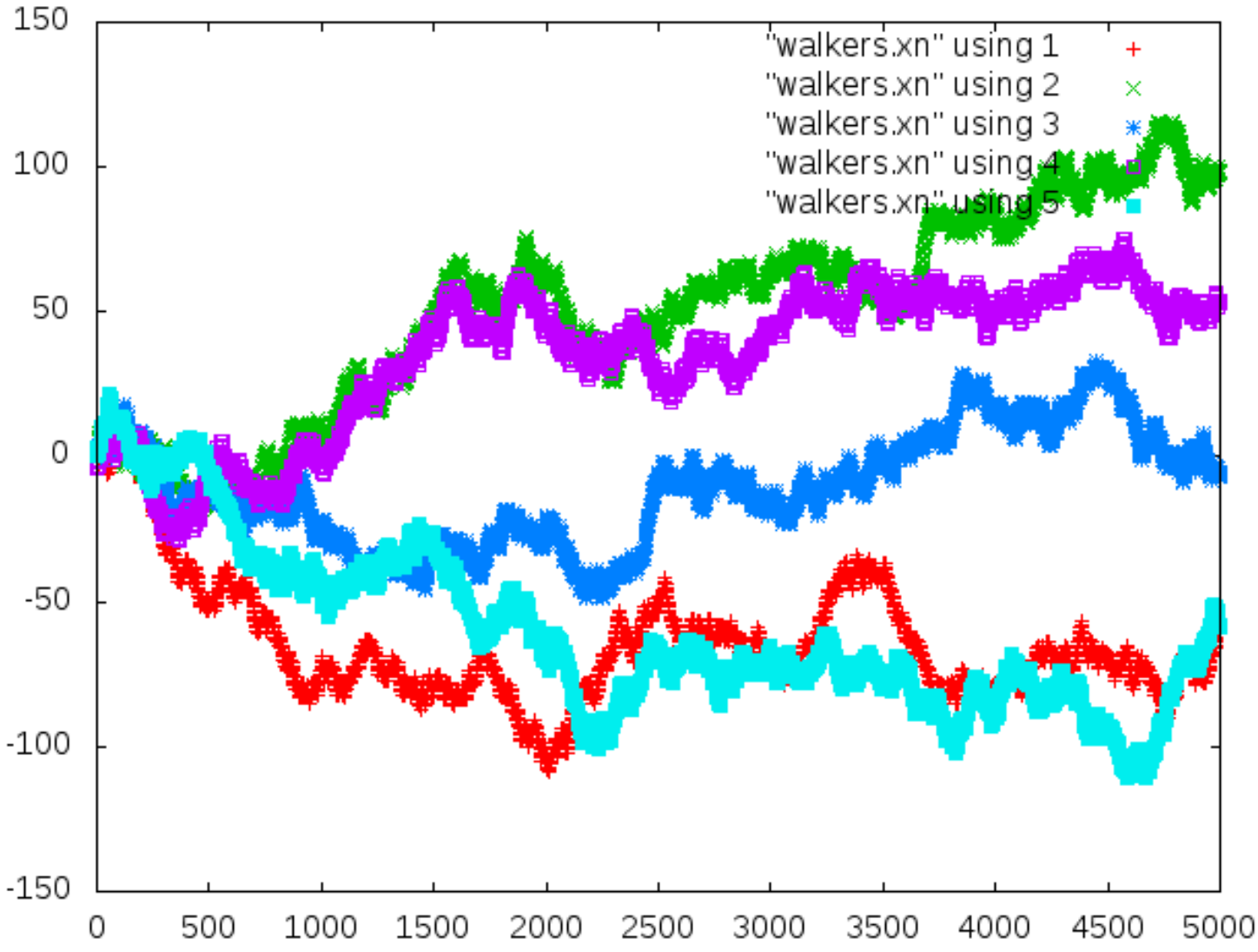


Figure : The positions of 5 random walkers with time.

Random Walk Monte Carlo

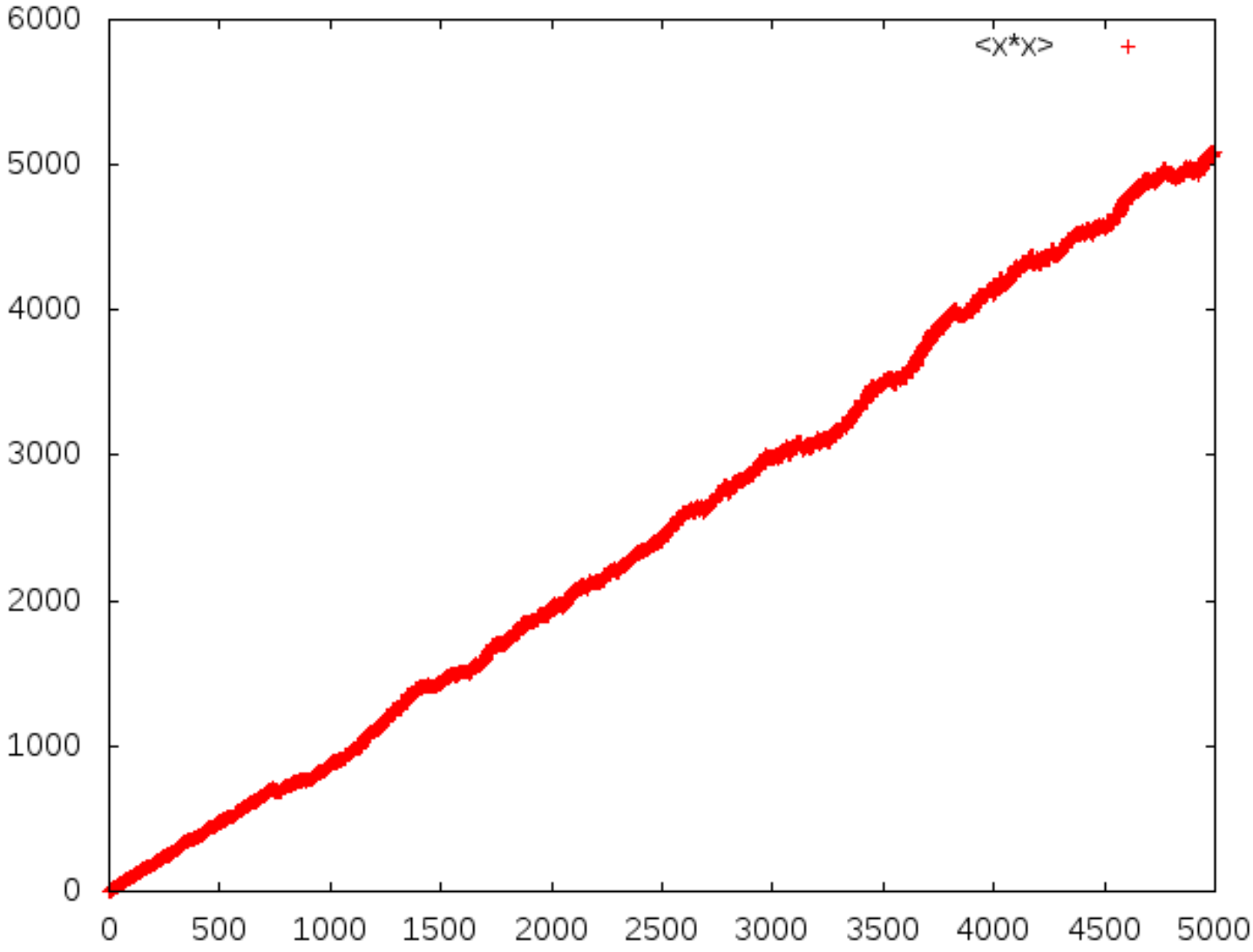


Figure : $< x_n^2 >$ versus the step numbers, averaged over 500 one dimensional random walkers.

Random Walk versus Diffusion

- Walker's alternative steps are equally likely. Thus, the average displacement of a walker is zero: $\langle x_n \rangle = 0$, where the average is over different walkers.

- But $\langle x_n^2 \rangle \sim n$ compared to the case for a free walker $x = vt$ where time, $t = n$ implies that a random walker will move away from its starting point much slower.

- Random walk: focus is on the behaviour of a single particle.
- Diffusion: the density of particles (large number of random walkers) in space at a given time, $\rho(x, y, z, t)$, is considered.

- If probability of finding a particle at point (x, y, z) at time t is $P(x, y, z, t)$, then both $P(x, y, z, t)$ and $\rho(x, y, z, t)$ should have the same dynamical equation which can be found by considering an individual random walker.

Random Walk versus Diffusion

- Assume that the random walker takes steps on a cubic lattice, one step each time. Let $P(i,j,k,n)$ be probability to find walker at (i,j,k) at time n. Walker has 6 nearest neighbours.

- If walker was at one of the 6 nearest neighbours at time $(n-1)$, then the probability to arrive at (i,j,k) is:

$$P(i,j,k,n) = \frac{1}{6}\,[P(i+1,j,k,n-1) + P(i-1,j,k,n-1) \\ P(i,j+1,k,n-1) + P(i,j-1,k,n-1) \\ P(i,j,k+1,n-1) + P(i,j,k-1,n-1)]. \quad (26)$$

Adding $-P(i,j,k,n-1)$ to both sides gives and using backward difference on the l.h.s. and central difference on the r.h.s. gives

$$\frac{\partial P(x,y,z,t)}{\partial t} = D\,\nabla^2 P(x,y,z,t), \quad D \equiv \frac{(\Delta x)^2}{6\,\Delta t} \qquad (27)$$

Random Walk versus Diffusion

- The diffusion equation derived above, shows the relationship between random walk and diffusion.

- The density of particles (or walkers) have the same equation $\frac{\partial \rho}{\partial t} = D\,\nabla^2 \rho$.

- Consider a one-dimensional case with $\rho(x,t) \equiv \rho(i,n)$. Applying finite difference gives

$$\rho(i,n+1) = \rho(i,n) + \frac{D\Delta t}{(\Delta x)^2}\left[\rho(i+1,n) - 2\rho(i,n) + \rho(i-1,n)\right] \quad (28)$$

- Given an initial condition, $\rho(x, t=0)$, one can solve for future ρ.

- Give relation of Diffusion equation to Jacobi method (used ealier.)