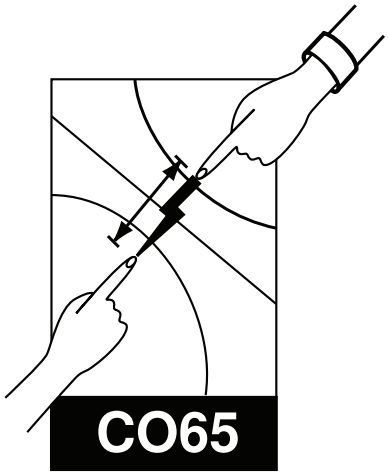
CO65: Monte Carlo integration



revised EG, MK October 2024

Preface

It is common practice in modern software engineering to write programs in a modular and standardised way. To help you get started, appendix A provides you with one or more specific function headers which you **MUST** use to write the functions around which your program should be built and in order to receive a satisfactory mark. Your comments in the header of each function must include:

- author and date,
- purpose (a brief description of what the function does),
- inputs and outputs,
- and, if appropriate, any constraints or limitations of use.

Do not forget that you will also need to keep good records of your progress during this practical in your logbook. If you make plots and/or write any notes electronically, ideally, you would print them and affix them into the pages of your logbook. You should have comments in function and script headers of your code, as well as comments within your code. These aspects may all be considered at marking time.

Assessment

Please see the Prelims Guide in Canvas for Computing Lab deadlines as well as availability (schedule) of Computing Lab Demonstrators who mark your work as well as offer help and advice. Before you meet with a demonstrator for marking, **you must** upload your work into Canvas (both your MATLAB code and your short summary). Your code must be uploaded in a single file (see instructions in Canvas) and must include any functions required by the script. Your short summary must use the format recommended in Canvas and uploaded into Canvas in valid pdf format.

1 Introduction

The Monte Carlo method is a simple yet powerful method that can be used to integrate functions, even when the functions are difficult or impossible to integrate analytically. The basis of the method is to surround the function to be integrated by a simple volume then generate sets of random numbers inside this volume and calculate the percentage of these sets that satisfy the equation to be integrated. The method is also used to calculate line and surface integrals by choosing random numbers along a line or surface rather than a volume.

The error in the Monte Carlo method is inversely proportional to the square root of the number of random numbers generated so to increase accuracy many random numbers need to be generated. Modern Monte Carlo methods modify the simple approach in the last paragraph to gain a lower error, these modern methods will not be investigated in this project.

In this script, students need to implement the functions in appendix A and write scripts that call the functions to calculate parameters asked in each section.

Copyright © 2024 University of Oxford, except where indicated.

2 Calculate the volume of the unit 10-dimensional sphere

Although physical space has only three dimensions, mathematically we sometimes need to think in more then three variables. It is easy to do higher dimensional integrals using the Monte Carlo method. In this problem we will calculate the volume of a 10-dimensional sphere (called a *hypersphere*) encased in a 10-dimensional *hypercube* of side 2 units.

The formula for the unit hypersphere is given by

$$x_1{}^2 + x_2{}^2 + x_3{}^2 + \ldots + x_{10}{}^2 = 1$$

We start by generating random numbers $x_1, x_2, x_3, \ldots x_{10}$ in the hypercube ($-1 \leq x_1, x_2, x_3, \ldots x_{10} \leq 1$). Count the ratio, R, of these random sets that lie inside the sphere. Another way of saying this is count the ratio of these random sets that satisfy

$$x_1{}^2 + x_2{}^2 + x_3{}^2 + \ldots + x_{10}{}^2 \leq 1$$

The volume of the sphere is the ratio R multiplied by the volume of the box ($2^{10} = 1024$).

Since the accuracy of the Monte Carlo method grows as $1/\sqrt{N}$ (where N is the number of random sets of $x_1, x_2, x_3, \ldots x_{10}$ you choose) you will need to choose a large N for your calculation ($N \sim 100000$).

3 Calculate the mass, centre of mass and moment of inertia of a complicated shape

We will use the Monte Carlo method as in the previous exercise to determine the mass of a complicated shape with a constant density, the centre of mass of this shape and its moment of inertia.

The mass of an object is defined as

$$\int \rho \, \mathrm{d}x \, \mathrm{d}y \, \mathrm{d}z \tag{1}$$

the centre of mass $(x_{\mathrm{cm}}, y_{\mathrm{cm}}, z_{\mathrm{cm}})$ is defined as

$$\frac{\int x\rho \, \mathrm{d}x \, \mathrm{d}y \, \mathrm{d}z}{\int \rho \, \mathrm{d}x \, \mathrm{d}y \, \mathrm{d}z} \qquad \frac{\int y\rho \, \mathrm{d}x \, \mathrm{d}y \, \mathrm{d}z}{\int \rho \, \mathrm{d}x \, \mathrm{d}y \, \mathrm{d}z} \qquad \frac{\int z\rho \, \mathrm{d}x \, \mathrm{d}y \, \mathrm{d}z}{\int \rho \, \mathrm{d}x \, \mathrm{d}y \, \mathrm{d}z} \tag{2}$$

and the moment of inertia is defined as

$$I_{xx} = \int (y^2 + z^2)\rho \, \mathrm{d}x \, \mathrm{d}y \, \mathrm{d}z \qquad I_{yy} = \int (z^2 + x^2)\rho \, \mathrm{d}x \, \mathrm{d}y \, \mathrm{d}z \qquad I_{zz} = \int (x^2 + y^2)\rho \, \mathrm{d}x \, \mathrm{d}y \, \mathrm{d}z \tag{3}$$

We will calculate the mass, centre of mass and moment of inertia of a torus given by

$$z^2 + \left(\sqrt{x^2 + y^2} - 3\right)^2 \leq 1 \tag{4}$$

bounded by the planes $x \geq 1$, and $y \geq -3$ as shown in figure 1.
Similar to the previous problem, you will need to calculate a set of random numbers (3 in this case) x, y, z in a suitable volume around the shape. The easiest volume is smallest box that totally encloses the part of a torus we're interested in and is given by

$$4 \geq x \geq 1 \qquad 4 \geq y \geq -3 \qquad 1 \geq z \geq -1$$

For each x, y, z that satisfies equation 4 calculate the following quantities, assuming that the density of the shape is unity i.e. $\rho = 1$.

$$\sum \rho, \quad \sum \rho^2, \quad \sum x\rho, \quad \sum (x\rho)^2, \quad \sum y\rho, \quad \sum (y\rho)^2, \quad \sum z\rho, \quad \sum (z\rho)^2, \quad \sum (y^2 + z^2)\rho,$$
$$\sum ((y^2 + z^2)\rho)^2, \quad \sum (z^2 + x^2)\rho, \quad \sum ((z^2 + x^2)\rho)^2, \quad \sum (x^2 + y^2)\rho, \quad \sum ((x^2 + y^2)\rho)^2$$

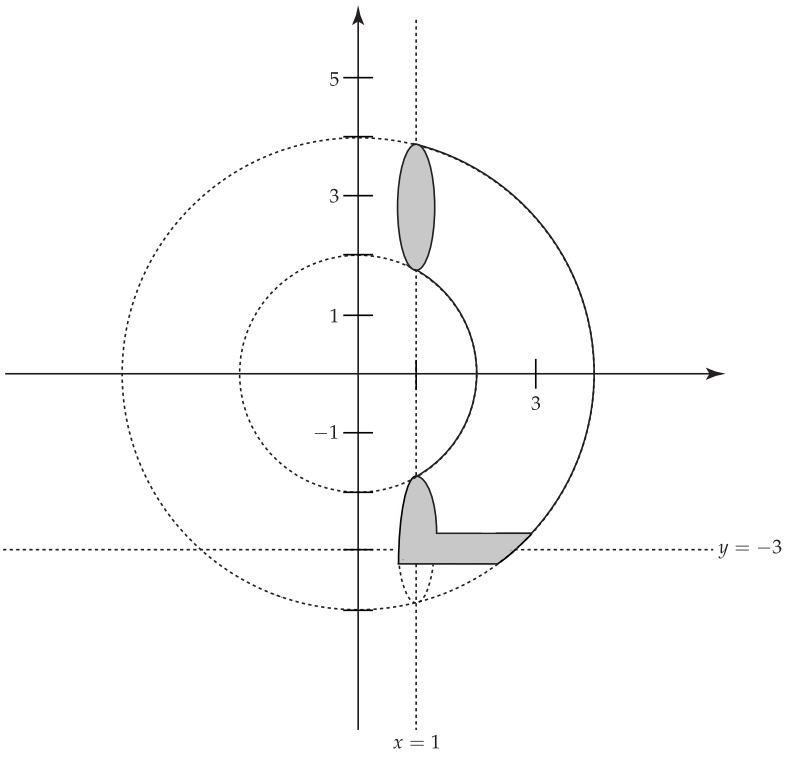


Figure 1: A complicated shape — a toroidal section.

We need to repeat this process a large number of times for reasonable accuracy: start with $N = 10000$.

▶ **Check your results with a demonstrator.**

The mass of the shape with associated error is

$$\text{volume of the bounding box} \times \Sigma\rho/N \pm \text{volume} \times \sqrt{\frac{\frac{\Sigma\rho^2}{N} - \left(\frac{\Sigma\rho}{N}\right)^2}{N}}$$

the centre of mass with associated error is given by

$$x_{\text{cm}} = \frac{\text{volume} \times \Sigma(x\rho)/N \pm \text{volume} \times \sqrt{\frac{\frac{\Sigma(x\rho)^2}{N} - \left(\frac{\Sigma(x\rho)}{N}\right)^2}{N}}}{\text{mass}} \quad \text{etc}$$

and the moment of inertia with associated error is

$$I_{xx} = \text{volume} \times \frac{\Sigma(y^2+z^2)\rho}{N} \pm \text{volume} \times \sqrt{\frac{\frac{\Sigma((y^2+z^2)\rho)^2}{N} - \left(\frac{\Sigma((y^2+z^2)\rho)}{N}\right)^2}{N}} \quad \text{etc}$$

4 Calculate an integral over a surface

Calculate the integral

$$I = \int_S \sin(\ln(x+y+1))\,\mathrm{d}S \qquad \text{(n.b. ln is natural logarithm)} \tag{5}$$

over the disk

$$S = x^2 + y^2 \leq 0.25 \tag{6}$$

We may approximate this by

$$i = \frac{\sum_{j=1}^{N} \sin(\ln(x+y+1))\Delta S}{N} \tag{7}$$

where ΔS is the area of an integration box (the area over which we are interested i.e. the disk).

As in the previous exercises we surround the area we're integrating over (i.e. S) by an area. Since this is a 2D problem we enclose our disk in with the square

$$-0.5 \leq x \leq 0.5 \qquad -0.5 \leq y \leq 0.5$$

and choose our N sets of random numbers x, y in this square. For each (x, y) that lies inside the disk (i.e. satisfies equation 6) we add the contribution $\sin(\ln(x+y+1))\Delta S$ to the sum in equation 7.

The area over which we integrated, ΔS, was equal to unity so the value of the integral is given by

$$i = \frac{\sum_{j=1}^{N} \sin(\ln(x+y+1))}{N} \tag{8}$$

5 Preparing for assessment

Please see Canvas Prelims Practicals Computing section where you will find all requirements, instructions, and the links for uploading your work. When you have finished writing your code (and have tested it completely) and your short summary is complete (double checking it meets all the requirements), you must upload your work (both the code and the report) electronically to Canvas. You must complete these uploads in advance of meeting with a demonstrator for marking. Deadlines and Computing Demonstrator schedules are also found in Canvas.

At marking time, you should have at hand your computer with your code, a copy of your short summary, and your logbook (where you wrote extra notes during your program development). Be prepared to describe your code and demonstrate its execution.

A Functions to be implemented

A.1 Points selection

```
function [accepted_pts] = get_accepted_points(pts, cond_func)
% Author: ??? , Date: ??/??/????
% Returns the points from pts that returns True in cond_func.
% Input:
% * pts: (N x D) matrix that contains N points with D dimension
% * cond_func: a function that accepts a point as a D-dimensional vector and returns true
%              if a certain condition is fulfilled and false otherwise (see hint).
%
% Output:
% * accepted_pts: (M x D) that only returns D-dimensional points that fulfills the conditions
%                 by the cond_func function
%
% Example use:
% >> pts = rand(10000, 4); % 10000 points with 4 dimensions
% >> cond_func = @(pt) (sum(pt.^2, 2) < 1); % only accepts with radius < 1
% >> acc_pts = get_accepted_points(pts, cond_func);
end
```

A.2 Sum of accepted points

```
function [val] = calculate_sum(acc_pts, func)
% Author: ??? , Date: ??/??/????
% Calculate the sum of func(pt) for every pt in acc_pts and returns the value.
% Input:
% * acc_pts: the list of points accepted for the calculation with the form of a (M x D) matrix,
%            and each row of the matrix corresponds to each point.
% * func: a function that accept a point and returns a value to be calculated.
%
% Output:
% * val: sum of func(pt) for every pt in acc_pts
%
% Example use:
% >> % Calculating the x-distance of centre of mass of the positive quadrant of a
% >> % 3-dimensional sphere with radius 1
% >> pts = rand(10000, 3); % 10000 points with 4 dimensions
% >> cond_func = @(pt) (sum(pt.^2, 2) < 1); % only accepts with radius < 1
% >> acc_pts = get_accepted_points(pts, cond_func);
% >> func = @(pt) pt(1); % get the x-coordinate of the point
% >> xcm = calculate_sum(acc_pts, func) / length(acc_pts);
end
```

Bibliography

[1] W. H. Press et al. *Numerical Recipes: the Art of Scientific Computing*, Cambridge University Press (2007).