FF505/FY505

Computational Science

Example: Monte Carlo Simulation

Marco Chiarandini (marco@imada.sdu.dk)

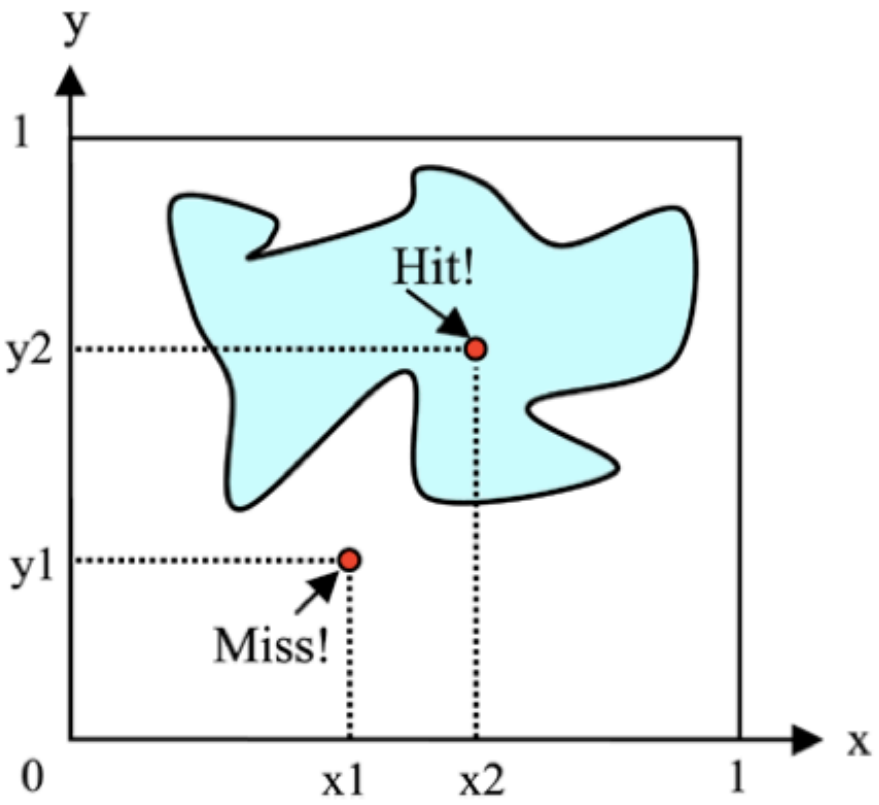
Department of Mathematics and Computer Science (IMADA)
University of Southern Denmark

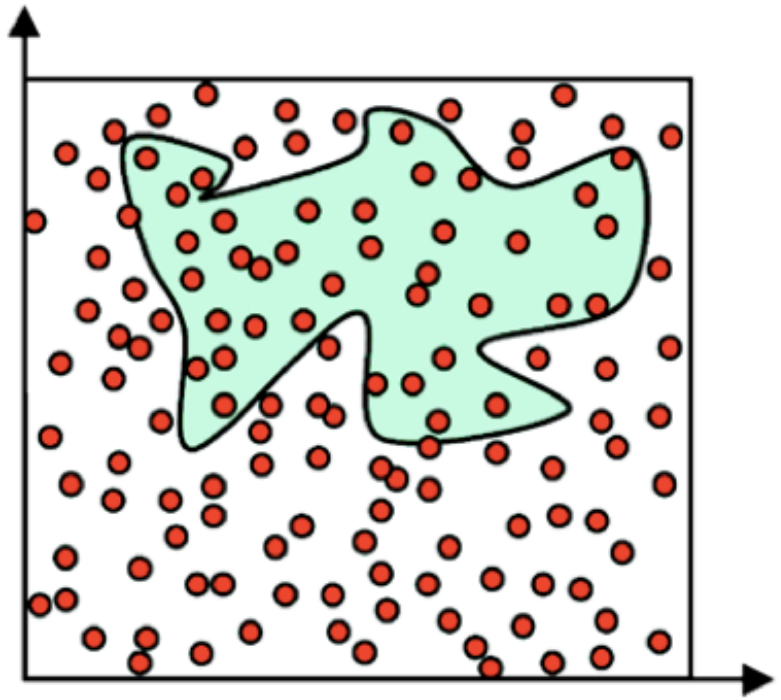
Outline

Outline

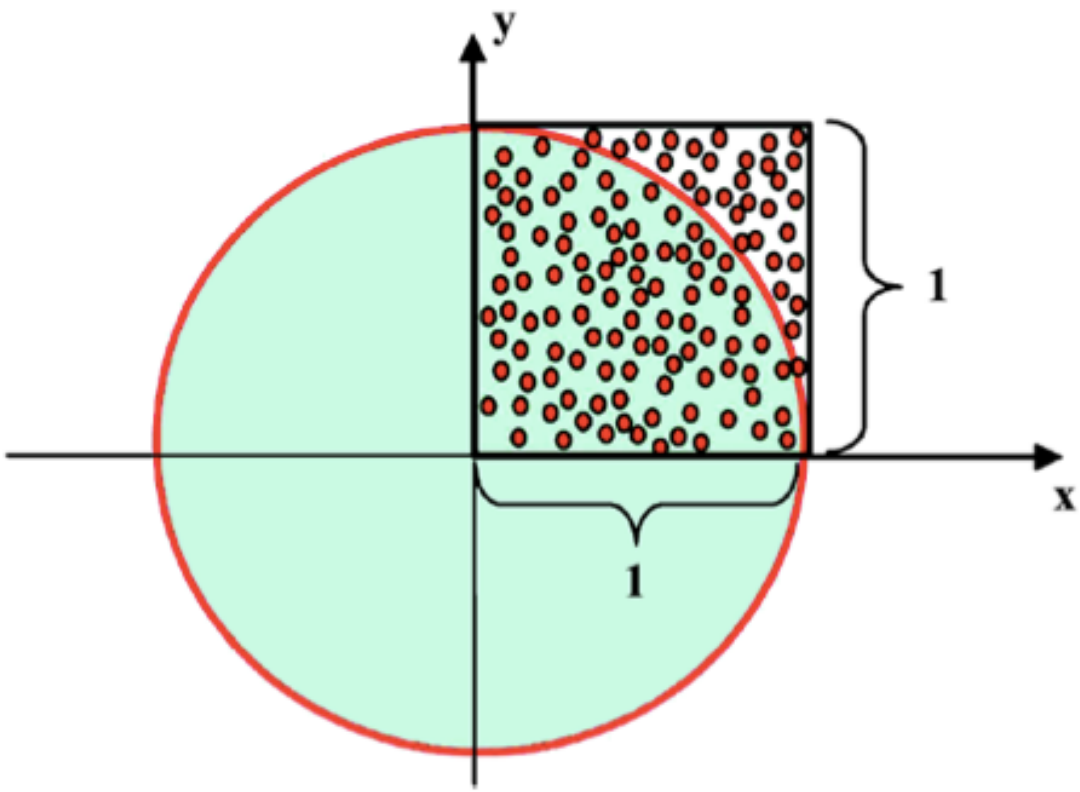
Monte Carlo Simulation

Calculate area by random rain:





Calculate π



Solution

Let A_s be the simulated area:

$$\frac{\pi}{4} = A_s$$

```
S=1000;
hits = 0;
for k = 1:S
    x = rand(1);
    y = rand(1);
    P = x^2+y^2;
    hits = P<1;
end
As=hits/S;
pi=4*As;
```

```
S=1000;
XY=rand(S,2);
P=sum(XY.^2,2);
hits=sum(P<1);
As=hits/S;
pi=4*As;
```

Script and Function Files (M-Files)

Script file

```
x=(1:1000)';
for k=1:5
  y(:,k)=k*log(x);
end
plot(x,y)
```

command line `simple`
does not take arguments
operates on data in the workspace

Function file

```
function y=simple(maxLoop)
        % (smart indent)
        x=(1:1000)';
        for k=1:maxLoop
        y(:,k)=k*log(x);
        end
        plot(x,y)
```

command line `g=simple(10)`
can take input arguments and return output arguments.
Internal variables are local to the function

Same name conventions for `.m` files as for variables.
Check if variables or functions are already defined.

```
exist("example1")
exist("example1.m","file")
exist("example1","builtin")
```

```
type fun
```

Script and Function Files (M-files)

- Modularize
- Make interaction clear
  make functions interact via arguments (in case structures) rather than via global variables
- Partitioning
- Use existing functions
  (http://www.mathworks.com/matlabcentral/fileexchange)
- Any block of code appearing in more than one m-file should be considered for packaging as a function
- Subfunctions
  packaged in the same file as their functions
- Test scripts

Efficient Code


```
function mypi=calculate_pi_1(S)
   hits = 0;
   for k = 1:S
       x = rand(1);
       y = rand(1);
       P = x^2+y^2;
       hits = hits + P<1;
   end
   As=hits/S;
   mypi=4*As;
```

```
function mypi=calculate_pi_2(S)
   S=1000;
   XY=rand(S,2);
   P=sum(XY.^2,2);
   hits=sum(P<1);
   As=hits/S;
   mypi=4*As;
```

```
tic,
for k=1:100
   calculate_pi_1(1000);
end
toc
```

```
tic,
for k=1:100
   calculate_pi_2(1000);
end
toc
```

Outline

Techniques for Improving Performance

Can you improve performance and use memory more efficiently for this code?

```
A=rand(1000,400)>0.7
s=[]
M=0
for j=1:400
    tmp_s=0
    for i=1:1000
        if A(i,j)>M
            M=A(i,j)
        end
        if A(i,j)>0
            tmp_s=tmp_s+A(i,j)
    end
    s=[s, tmp_s]
end
```

Use `tic ... toc` and `whos` to analyse your code.

`tic; bad; toc`

For inspiration look at User's Guide:

MATLAB > User's Guide > Programming Fundamentals > Software Development > Performance > Techniques for Improving Performance