Programming in Excel and MATLAB

CBE 422

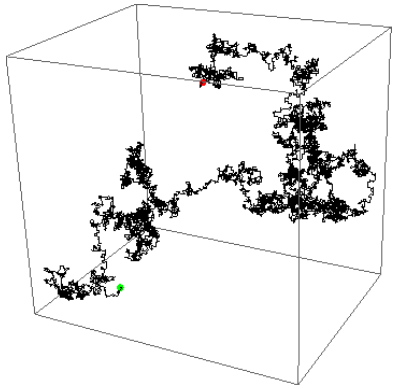
- Example 1: random walks
  - Excel: rand(), countif, vlookup
  - MATLAB: for loops, if statements, plotting
- Example 2: Particles on hexagonal lattice
- Example 3: Monte Carlo integration

Example 1: random walks

- Diffusion: molecules undergoing random walks
- Molecule takes *n* steps of size *l*, each in a random direction, over a time interval *t*
- What is the probability density, *p(x,t)*, of observing a molecule between *x* and *x* + *dx* after time *t*?
- **Answer** (1-D case, limit of large *n*):

$$p(x,t) = \frac{1}{\sqrt{4\pi Dt}} \exp\left(-\frac{x^2}{4Dt}\right)$$

$$D = \frac{nl^2}{2t}$$



- **Objective**: verify this through simulation of 1-D random walks

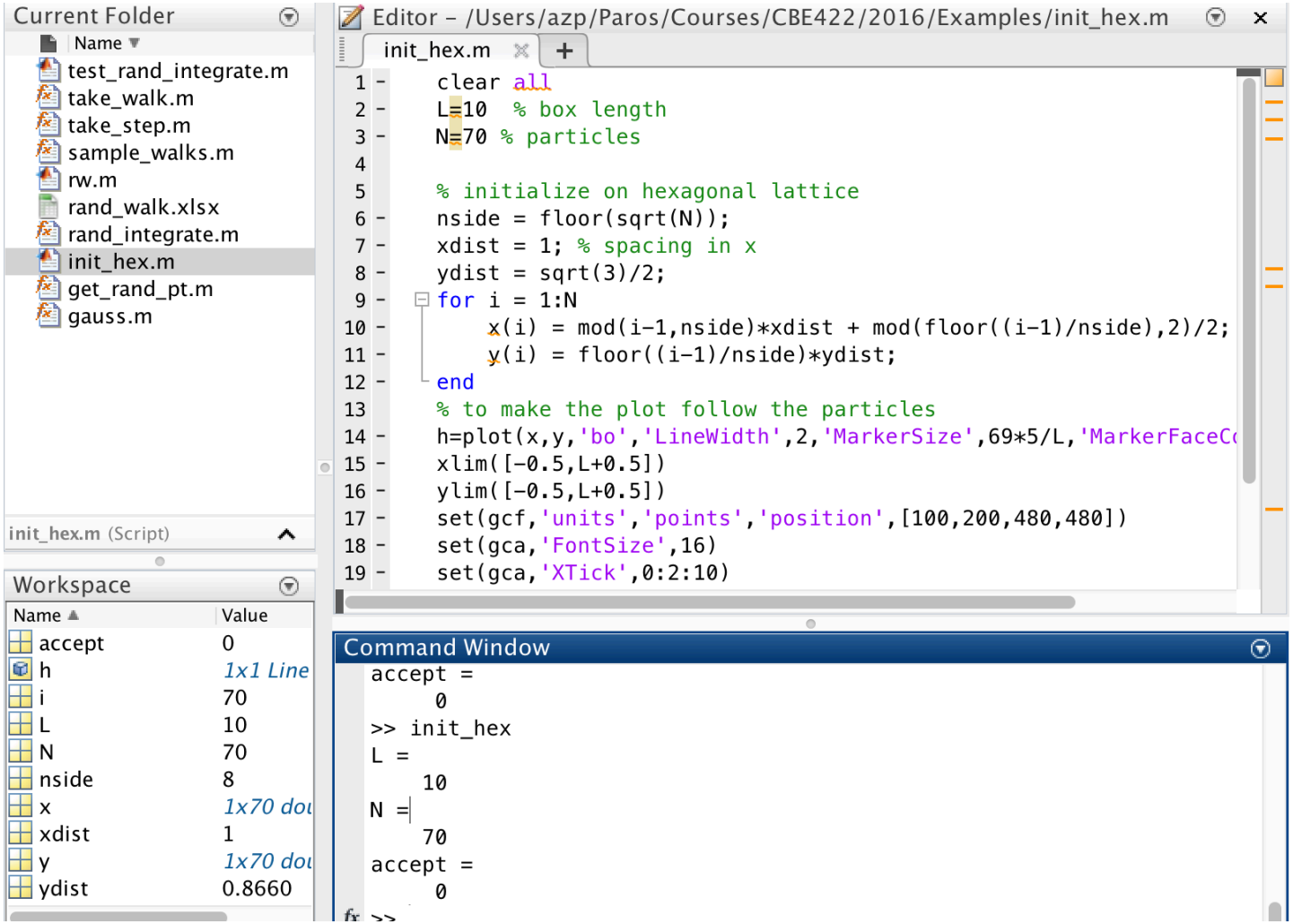
Excel implementation

- Use multiple sheets
- RAND() produces uniformly distributed random numbers
- VLOOKUP to select step to take based on random number
- FLOOR function to produce bin corresponding to the final position
- COUNTIF function to generate histogram
- F9 to recalculate (produce new set of runs)

Matlab

- MATLAB (MATtrix LABoratory) is a package used for technical computing. It integrates computation, visualization, and programming.
- can install on your own laptop:
  http://www.princeton.edu/software/licenses/software/matlab/
- Basic ML element: Matrices/Arrays
- Case sensitive (a ≠ A)
- Implicit type conversions (reals are "Double Precision," 64 bit)
- Semicolon (**;**) to suppress printing of results
- Colon (**:**) to construct a range of values
- Comments are preceded by %
- Relational operators: **<,>, <=,>=,==,~=**

Matlab windows



Editor: to work with script files

Command Window: to enter commands and data

Workspace: shows you all “local” variables and their type and size

Matlab simple scripts

- AKA m-files
- Equivalent to typing the commands one-by-one at the command prompt “>>”
- Require variables to be defined within the script file, or interactively prior to execution
- All variables created in a script file remain in workspace after execution

Matlab functions

- Take arguments and return one or more results
- First result is also the value of the function if used in an assignment statement
- Call is “By Value” (argument values in calling program are not changed even if changed within the function)

Simple Matlab functions

```
>> a=2; b=3;
>> e=testf(a,b)
a =
     7
d =
     5
e =
     7
>> b
b =
     3
>> a
a =
     2
>>
```

```
function[a]=testf(c,d)
a=2*c+d
d=d+2
```

Within function: **c** becomes 2, **d** becomes 3; **a** (the result of the function) becomes 2*2+3=7

d is changed within the function, but the new value does not get passed on to **b**; similarly, **a** (local to the function) does not change.

Function arguments

```
>>[e,f]=test2(3,4)
a =
     6
b =
     2
e =
     6
f =
     2
>>
```

```
function[a,b]=test2(c,d)
a=2*c
b=d/2
```

Getting two arguments back is simple.

Conditionals

```
function[a]=cond(x)
if x>5
     a=5;
elseif x<0
     a=0;
else
     a=x;
end
```

```
>> cond(10)
ans =
      5
>> cond(2.3)
ans =
    2.3000
>> cond(-2.3)
ans =
      0
>>
```

There is no "then"! Instead of "end" one can also use "end if"; ML is case sensitive so that "If" "Else" "End" do not work. Also "else if" does not work.

“For” loops

```
function[a]=MyLoop(N)
a=0;
for  i=1:2:N
     a=a+i;
end
```

```
>>
MyLoop(4)
ans =
      4
```

Remember that ML constructs a range of numbers as start:increment:finish

For increment=1, one can use start:finish

“While” loops

```
function[i]=MyWhile(x)
i=0; x=abs(x)
while x>0
    x=x/10;
    i=i+1;
end
```

```
>> MyWhile(1e300)
x =
   1.0000e+300
ans =
    624
```

This procedure counts how many divisions by 10 it takes to get a number to zero (in double precision)

Debugging Matlab functions

- Display intermediate results by not ending statement with ";"
- Can set formal "breakpoints" and "conditional breakpoints" in code and pause execution to examine variables

Example 2: particles on lattice

Example 3: Monte Carlo integration