Lecture 13: Other Concepts and Utilities in Wolfram Mathematica.

A8B17CAS

Jozef Lukáč

Department of Radio Engineering
Czech Technical University in Prague
Czech Republic
lukacjo1@fel.cvut.cz

December 19
Winter semester 2023/24

Outline

1. `Export` to file.
2. Plotting + `Options`.
3. Conditions
4. Procedural Programming.
5. Applying Functions Repeatedly
6. `Module`
7. Trace – Debugging in MATHEMATICA
8. Calculus
9. Array shaping
10. Array/Matrix Broadcasting (Python/MATLAB vs MATHEMATICA).
11. Logical Indexing and multiple assignment
12. `Manipulate`
13. Example of Code.

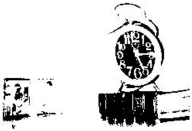
Export to file.

- To save a part of an expression (intended to be used only in MATHEMATICA), you can use Put, resp. >>. To read the saved expression, use Get, resp. <<, *e.g.*

```
SetDirectory[NotebookDirectory[]];
expr = Table[RandomInteger[], 5, 2];
expr >> "outFile"; (*Put[expr,"outFile"]*)
<< "outFile"; (*Get["outFile"]*)
a1 = % (*set a1 to the expression from file*)
```

- Note: % returns previously evaluated expression, %% – one before last evaluated expression, %*n* – returns (evaluated) expression from the output cell Out[n].
- To export an expression using a specific format, use `Export["dest.ext",expr]`, *e.g.*

```
SetDirectory[NotebookDirectory[] <> "data"];
im = Import["im03_desk.tiff"];
imData = ImageData[im]; (*2D matrix or Reals*)
thr[n_] := If[n > 0.34, 1, 0];(*threshold function*)
thrImData = Map[thr, imData, {2}];
Image[thrImData] (*create image from the thresholded data*)
Export["im03_desk_thr.png", %]
```

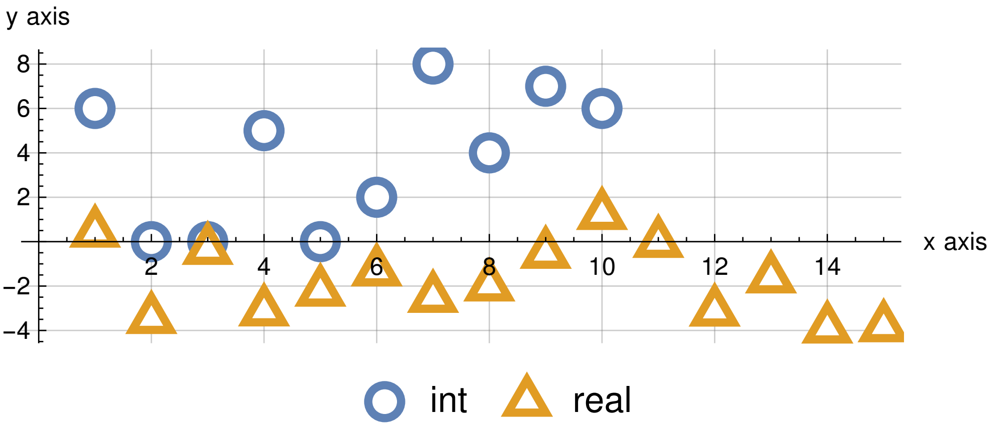


Plotting + Options.

- There are many plot commands: `Plot`, `Plot3D`, `CountourPlot`, `ListPlot`, `ListLinePlot`, `ParametricPlot`, `ParametricPlot3D`, `DensityPlot`, `DiscretePlot`, `ArrayPlot`, `MatrixPlot`, `Histogram`, `PolarPlot`, `RegionPlot`, `VectorPlot`, ...
- Some commands, especially plotting commands, usually have options. To get them, use `Options[`*symbolName*`]`, *e.g.* `Options[Plot]`
- Options are in the form of a rule. And have default values unless explicitly changed.
- Example of a plot:

```
ListPlot[{RandomInteger[{0, 8}, 10], RandomReal[{-5, 3}, 15]},
  PlotLegends -> {"int", "real"}, PlotMarkers -> {"OpenMarkers", 12},
  AspectRatio -> 1/3, AxesLabel -> {"x axis", "y axis"},
  GridLines -> Automatic]
```

Plotting + `Options`.

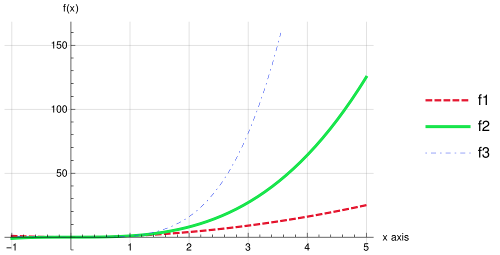


Example of using `ListPlot` from the previous slide. (exported to pdf with `Export`)

Plotting + Options.

Exercise

- Try to get similar figure of functions $f_1(x) = x^2$, $f_2(x) = x^3$, $f_3(x) = x^4$ in range of $x \in (-1, 5)$. Use Plot command.
- Usuful options are: PlotLegends, AxesLabel, GridLines, PlotStyle.
- PlotStyle substitutes for a list of line options, *e.g.* `PlotStyle->{lineOptionList1, lineOptionList2, lineOptionList3}`, where `lineOptionList` is a list and can contain (a) color specification (*e.g.* `RGBColor[r,g,b]`, or name of a color *e.g.* Red); (b) line thickness (*e.g.* Thick, Thin, `Thickness[ratio]`); (c) line style (*e.g.* Dotted, DotDashed, Full, Dashed)



Plotting + Options.

- Try to get similar figure of functions $f_1(x) = x^2$, $f_2(x) = x^3$, $f_3(x) = x^4$ in range of $x \in (-1, 5)$. Use `Plot` command.
- Usuful options are: `PlotLegends`, `AxesLabel`, `GridLines`, `PlotStyle`.
- `PlotStyle` substitutes for a list of line options, *e.g.* `PlotStyle->{lineOptionList1, lineOptionList2, lineOptionList3}`, where `lineOptionList` is a list and can contain (a) color specification (*e.g.* `RGBColor[r,g,b]`, or name of a color *e.g.* `Red`); (b) line thickness (*e.g.* `Thick`, `Thin`, `Thickness[ratio]`); (c) line style (*e.g.* `Dotted`, `DotDashed`, `Full`, `Dashed`)

```
Plot[{x^2, x^3, x^4}, {x, -1, 5}, GridLines -> Automatic,
  PlotStyle -> {{RGBColor[0.9, 0.1, 0.2], Dashed,
  Thick}, {RGBColor[0.1, 0.9, 0.3], Thickness[0.008],
  Dotted}, {RGBColor[0.2, 0.3, 0.95], Thin, DotDashed}},
  PlotLegends -> {"f1", "f2", "f3"}, AxesLabel -> {"x axis", "f(x)"}]
```

Exercise

Conditions

- `If[condition,t,f]` – evaluates `t` if `condition` is true otehrwise evaluates `f`.
- `Which[test1,val1,test2,val2,...]` – returns first $\texttt{val}_i$ for which $\texttt{text}_i$ evaluates true. Useful *e.g.* for classification.
- `Switch[expr,form1,val1,form2,val2,...]` – evaluates expression and checks a match with each $\texttt{form}_i$, and returning $\texttt{val}_i$ of the first matched form.
- Composition of boolean expressions, see `And` (`&&`) and `Or` (`||`).
- Examples:

```
classif[x_] :=
  Which[x<1,range1,1<=x<3,range2,3<=x<5,range3,True,otherRange];
RandomInteger[{-3, 9}, 10]
classif /@ %
expr = x^(-1 + 4);
Switch[expr,_^2,"square power",_^_,"power",_Plus, "sum",_,"dont know"]
```

Procedural Programming.

- MATHEMATICA has also commands for procedural programming, *e.g.* `For`, `While`, `Do`, `Break`, `Continue`, `Return`, (`Print`). They are not used much – usually, there is a simple, functional way to do the same, *e.g.*

```
v=RandomReal[{0,1},10]
For[i=1,i<Length[v],i++,If[v[[i]]>0.6,Print["OK"],Print["notOK"]]]
```

vs.

```
Scan[If[#>0.6,Print["OK"],Print["notOK"]]&,v]
```

Applying Functions Repeatedly

Besides `Map` and `Apply`, other commands are used to apply functions/symbols repeatedly/multiple times.

- `Nest[f,expr,n]` – n-times applies `f` to `expr`.
- `NestList[f,expr,n]` – gives all intermediate results of `Nest[f,expr,n]` in list.
- `NestWhile[f,expr,test]` – applies `f` to `expr` while `test` yields true.
- `NestWhileList[f,expr,test]` – same as above with all intermediate results in list.
- `FoldList[f,x,{a,b,...}]` – gives `{x,f[x,a],f[f[x,a],b],...}`.
- `Fold[f,x,list]` – last element of `FoldList[f,x,list]`.
- `FixedPoint[f,expr]` – similar to `Nest[f,expr,n]`, but nesting stops when the result no longer changes.
- Examples:

```
Nest[f,x,3]                          ↦ f[f[f[x]]]
NestList[f,x,3]                      ↦ {x,f[x],f[f[x]],f[f[f[x]]]}
NestWhile[2#+1&,6,Mod[#,5]!=0&]      ↦ 55
FoldList[#1^2-15#2&,2,{2,-3,1}]      ↦ {2, -26, 721, 519826}
```

Applying Functions Repeatedly. Examples.

- Find minimum from each column of a matrix

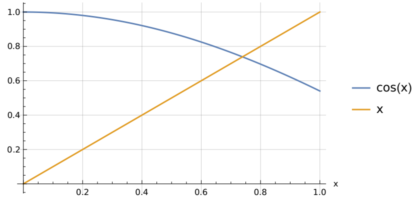
```
mtx=RandomInteger[{-4,5},{4,5}];mtx//MatrixForm
min[l1_List,l2_List]:=Inner[Min,l1,l2,List];
FoldList[min,mtx]//MatrixForm
Last[%]
```

  - ...other way with Map (/@):

```
Min/@Transpose[mtx]
```

- Fixed point[1]. Solution to $\cos(x) = x$:

```
Plot[{Cos[x],x},{x,0,1}]
NestList[Cos[#]&,0.,20]
FixedPoint[Cos[#]&,0.]
```



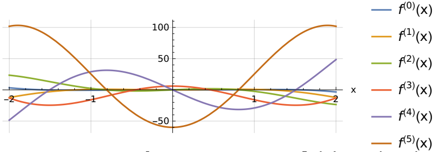
Plot of the functions $\cos(x)$ and x.

[1] https://en.wikipedia.org/wiki/Fixed_point_(mathematics)

Applying Functions Repeatedly. Examples.

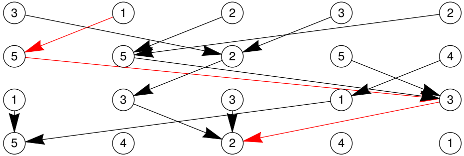
Exercise

- Compute first 5 derivatives of the function $x^3\cos(x)$ and plot the result. (Hint: use `NestList`, then use `Plot` with reference `%` to the previous result)



- Assume a matrix of indices `mtx=RandomInteger[{1,n},{m,n}]` for `m=4,n=5`.

$$\texttt{mtx=}\begin{bmatrix} 3 & 1 & 2 & 3 & 2 \\ 5 & 5 & 2 & 5 & 4 \\ 1 & 3 & 3 & 1 & 3 \\ 5 & 4 & 2 & 4 & 1 \end{bmatrix},$$



Each row corresponds to indices in the next row. Start at row 1, at position 2: $\texttt{mtx}_{1,2} = 1$, go to next row at the position 1 – there is number 5, and so on. What is the sequence of indices for a randomly generated matrix? (in this case it would be `{2, 1, 5, 3, 2}`) Start at the first row at position 2. You can use `FoldList`.

Applying Functions Repeatedly. Examples.

- Compute first 5 derivatives of the function $x^3 \cos(x)$ and plot the result. (Hint: use `NestList`, then use `Plot` with reference `%` to the previous result)

```
NestList[D[#, x] &, x^3 Cos[x], 5]
Plot[%, {x, -5, 5}]
```

- Assume a matrix of indices `mtx=RandomInteger[{1,n},{m,n}]` for `m=4,n=5`. Each row corresponds to indices in the next row. Start at row 1, at position 2: $\mathtt{mtx}_{1,2} = 1$, go to next row at the position 1 – there is number 5, and so on. What is the sequence of indices for a randomly generated matrix? (in this case it would be `{2, 1, 5, 3, 2}`) Start at the first row at position 2. You can use `FoldList`.

```
iStart = 2; ff[i_, lst_List] := Part[lst, i];
FoldList[ff, iStart, mtx]
```

Exercise

Module

- More complicated functions can be specified using `Module[{x,y,...}, expr]`, variables/symbols `x,y,...` are local. *E.g.*

```
g[vec_List,n_]:=Module[{idc},
 idc=Position[vec,n];
 Print[idc];
 Join[vec,Flatten@idc]];
g[RandomInteger[{0,6},10],2]
```

`Trace` – Debugging in MATHEMATICA.

- When writing a code, we iteratively adjust it to do what we originally intended. The process of looking for and correcting errors is called *debugging*[2].
- Steps in the evaluation of an expression can be revealed by `Trace[expr]`, *e.g.*

  `Trace[2^3+4^2+1]` $\mapsto$ `{{2^3,8},{4^2,16},8+16+1,25}`

[2]Debugging https://en.wikipedia.org/wiki/Debugging

Calculus

- MATHEMATICA allows users to solve problems in calculus, linear algebra, general algebra, and others, all symbolically and/or numerically.
- Usefull commands are: `N` (numeric evaluation), `D` (derivative), `Integrate`, `NIntegrate` (numeric integration), `FindRoot`, `Roots`, `Solve`, `Limit`.
- The Final result can sometimes be simplified using additional rules. See `Simplify`, `FullSimplify`, ...
- In solutions we can help MATHEMATICA by assumptions (use `Assuming`), *e.g.*

```
Integrate[x Exp[x^n], x]
Assuming[Element[n, Integers] && x > 0, % // Simplify]
```

Calculus

Calculate following:

$$\int_0^{2\pi} \mathrm{e}^{x\cos(\alpha)}\,\mathrm{d}\alpha, \tag{1}$$

$$\int_0^1 \cos(\sin(x))\,\mathrm{d}x, \quad \text{numerically}, \tag{2}$$

$$\lim_{n\to\infty}\left(1+\frac{1}{n}\right)^n, \tag{3}$$

$$\cos\left(\tan\left(\mathrm{e}^x\right)\right)'. \tag{4}$$

Exercise

Calculus

Exercise

Calculate following:

$$\int_0^{2\pi} \mathrm{e}^{x\cos(\alpha)}\,\mathrm{d}\alpha, \tag{1}$$

```
Integrate[Exp[x Cos[a]], {a, 0, 2 Pi}]
```

```
2 Pi BesselI[0, x]
```

$$\int_0^{1} \cos(\sin(x))\,\mathrm{d}x, \quad \text{numerically}, \tag{2}$$

```
NIntegrate[Cos[Sin[x]], {x, 0, 1}]
```

```
0.86874
```

$$\lim_{n\to\infty}\left(1+\frac{1}{n}\right)^n, \tag{3}$$

```
Limit[(1 + 1/n)^n, n -> Infinity]
```

```
E
```

$$\cos\left(\tan\left(\mathrm{e}^x\right)\right)'. \tag{4}$$

```
D[Cos[Tan[Exp[x]]], x]
```

```
-E^x Sec[E^x]^2 Sin[Tan[E^x]]
```

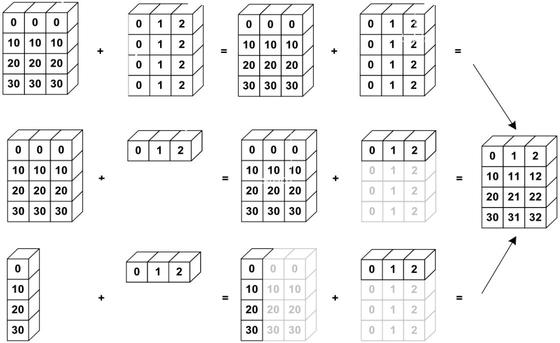
Array shaping

- When working with multidimensional arrays/expressions, we often need to rearrange shapes. In MATLAB, we can use `reshape(mat,dims)`. In MATHEMATICA, we have `ArrayReshape[mat,dims]`.
- Additionally, very useful are `Flatten` and `Partition`, *e.g.*

```
Table[{i, j, i - j}, {i, 3}, {j, 4}]
Flatten[%, 1]
RandomInteger[{1, 5}, 10]
Partition[%, 3] // MatrixForm
Partition[%%, 3, 1] // MatrixForm
```

Array/Matrix Broadcasting (Python/Matlab vs Mathematica)

- Schematic illustration of broadcasting arrays in Python (numpy)/Matlab[3]:



[3] source: https://mathematica.stackexchange.com/questions/99171/how-to-implement-the-general-array-broadcasting-method-from-numpy

Array/Matrix Broadcasting (Python/MATLAB vs MATHEMATICA)

- Define `mat=Table[10 i,i,0,3,,3]; v1=0,1,2=; v2=0,1,2,3=;`. In MATLAB define `mat=repmat(10*(0:3)',1,3), v1=0:2, v2=0:3`

MATLAB	MATHEMATICA
`v1 + mat`	`(# + v1) & /@ mat`
`v1 .* mat`	`(# v1) & /@ mat`
`v2'+ mat`	`v2 + mat`
`v2'.* mat`	`v2 * mat` (note the difference with `v2 . mat`)
`v1 + v2'`	`(# + v1) & /@ v2`
`v1 .* v2'`	`(# v1) & /@ v2`

- or you can explicitly replicate array with help of `ConstantArray` and `Join`.

Logical Indexing and multiple assignment.

- MATHEMATICA doesn't have logical indexing as MATLAB.
- ...but similar behaviour can be achieved by ReplacePart and Position, *e.g.*[4]

```
mat = RandomReal[{0, 1}, {5, 3}];
mat2 = ReplacePart[mat, Position[mat, pos_ /; pos < 0.7] -> -1];
MatrixForm /@ {mat, mat2}
```

  in MATLAB it would be

```
mat=rand(5,3); mat2 = mat; mat2(mat<0.7)=-1;
mat,mat2
```

- Multiple assignment works, but the parts have to be of equal size, *e.g.*

```
mat = RandomInteger[5, {5, 3}]; mat // MatrixForm
mat[[1]] = Table[-1, 3];
mat[[2 ;; 5, 2]] = Table[10, 4]; {a1, a2} = {10, 11};
{a1, a2, mat // MatrixForm}
```

[4] https://mathematica.stackexchange.com/a/2823/74293

Manipulate

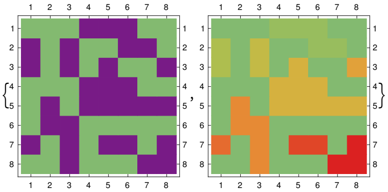
- Interactive manipulation with general expressions, *e.g.*

```
Manipulate[Plot[Sin[k x], {x, -10, 10}], {k, 0.1, 8.3}]
Manipulate[(x + y)^n // Expand, {{n, 2}, Range[10]}]
```

Example of Code.

Create a square matrix of 0s and -1s – the playground. -1s are ground, 0s are water. An island is a set of places that meet by at least one side (of square). Find islands and denote each by a different number. Then, plot them.

```
n = 8; plgrnd = RandomInteger[{0, -1}, {n, n}];
listMeetQ[lst1:{{_, _} ..}, lst2:{{_,_} ..}] :=
  AnyTrue[Flatten[Outer[List,lst1,lst2,1],1],
  (Plus@@Abs[Subtract@@ #]) ==1 &];
rule={head___List,x:{{_, _}..},fill___List,y:{{_, _}..},
  tail___List} /; listMeetQ[x, y] :> {head, Join[x, y], fill, tail};
pos = ({#} & /@ Position[plgrnd, -1]) //. rule;
plgrnd2 = ReplacePart[plgrnd, MapThread[Rule, {pos, Range[Length[pos]]}]];
MatrixPlot[#, ColorFunction -> "Rainbow"] & /@ {plgrnd, plgrnd2}
```



Questions?

A8B17CAS
lukacjo1@fel.cvut.cz

December 19
Winter semester 2023/24

Voluntary homework

- Assume you have a matrix (a bitmap picture) `n = 5; mat = RandomInteger[{-2, 4}, {n, n}];` Read the documentation of the command `Partition` and divide the matrix to submatrices of the size 3 x 3 with offset 1 in both directions. Compute the sums of the created submatrices.

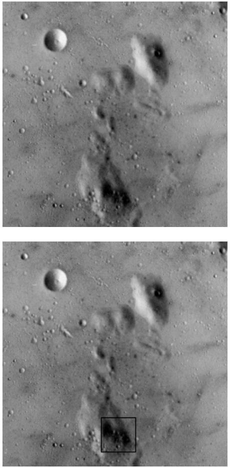
$$\begin{pmatrix} -2 & 3 & -1 & 2 & 4 \\ 4 & 3 & 2 & 0 & 4 \\ 0 & 2 & 3 & 4 & -1 \\ 0 & -2 & -1 & 2 & 1 \\ -1 & 2 & 0 & 4 & 4 \end{pmatrix} \begin{pmatrix} \begin{pmatrix} -2 & 3 & -1 \\ 4 & 3 & 2 \\ 0 & 2 & 3 \end{pmatrix} & \begin{pmatrix} 3 & -1 & 2 \\ 3 & 2 & 0 \\ 2 & 3 & 4 \end{pmatrix} & \begin{pmatrix} -1 & 2 & 4 \\ 2 & 0 & 4 \\ 3 & 4 & -1 \end{pmatrix} \\ \begin{pmatrix} 4 & 3 & 2 \\ 0 & 2 & 3 \\ 0 & -2 & -1 \end{pmatrix} & \begin{pmatrix} 3 & 2 & 0 \\ 2 & 3 & 4 \\ -2 & -1 & 2 \end{pmatrix} & \begin{pmatrix} 2 & 0 & 4 \\ 3 & 4 & -1 \\ -1 & 2 & 1 \end{pmatrix} \\ \begin{pmatrix} 0 & 2 & 3 \\ 0 & -2 & -1 \\ -1 & 2 & 0 \end{pmatrix} & \begin{pmatrix} 2 & 3 & 4 \\ -2 & -1 & 2 \\ 2 & 0 & 4 \end{pmatrix} & \begin{pmatrix} 3 & 4 & -1 \\ -1 & 2 & 1 \\ 0 & 4 & 4 \end{pmatrix} \end{pmatrix} \begin{pmatrix} 14 & 18 & 17 \\ 11 & 13 & 14 \\ 3 & 14 & 16 \end{pmatrix}$$

Example of the matrix, partitions/submatrices, and sums of submatrices.

Voluntary homework

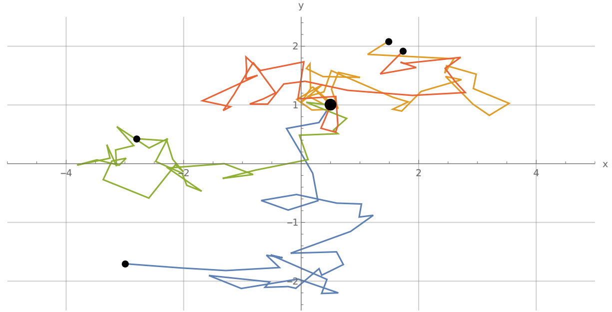
Your code can be used to detect a part of an image. *E.g.* To detect a darkest p x p block.

```
p = 40; (*partition size*)
SetDirectory[NotebookDirectory[] <> "data"];
mat = Import["im04_surface.tiff", "Data"]/255.;
n = Length@mat;
(*your code *)
(*goes here*)
ijFlat = First@Ordering[Flatten@%, 1]; (*find index of minimum*)
ij = With[{m=n-p+1}, {Quotient[#-1,m]+1, Mod[#-1,m]+1}]& @ ijFlat;
idc = Module[{i, j, iRng, jRng},
    i = First@%;
    j = Last@%;
    jRng = Range[j, j + p - 1];
    iRng = Range[i + 1, i + p - 2];
    Join[{i, #} & /@ jRng, {i + p - 1, #} & /@ jRng , {#, j} & /@
      iRng, {#, j + p - 1} & /@ iRng]
    ];
mat2 = ReplacePart[mat, idc -> 0];
GraphicsGrid[{{Image[mat], Image[mat2]}}, ImageSize -> Medium]
```



Voluntary homework

Generate a random walk[6] in 2 dimensions ($\mathbb{R}^2$). Assume the step in the x direction from $\mathcal{N}(0, 0.5)$; the step in the y direction from $\mathcal{N}(0, 0.25)$[7] Start from the point $[0.5, 1]$ and generate a list of 2-D points (2-element lists) – the path. Try to use `NestList` even though using `Accumulate` could be faster. To generate more paths, you can use `Table[(*code for one path*),{nPaths}]`. Plot the path(s) using `ListLinePlot`.



[6] https://en.wikipedia.org/wiki/Random_walk

[7] $\mathcal{N}(\mu, \sigma)$ denotes a random number from normal distribution with mean μ and standard deviation σ.