>> a = ['King', 'ace'] → not possible

>> a = ["King"; "ace"] → possible

>> a = char(a) = ['King'; 'ace_'] → possible.

>> a = ["Key", "boaard", "ding!"] → 1×3 String array

"Key" "boaard" "ding!"

>> char(a)

1 × 5 × 3 char array

>> a = ['Key', 'boaard', 'ding!']

ans =

'Keyboaarding!' → Concatenating

>> b = [1, 2, 3]

>> c = [12 0 1 -2]

>> d = [7 7 7 7]

>> e = [b, c, d]

= [1 2 3 12 0 1 -2 7 7 7 7]

Concatenating

Sometimes — we need Concatenation.

Concatenation

>> str1 = "Key"

>> str2 = "boaard"

>> str3 = "ding!"

>> str1 + str2 + str3

"Keyboaarding!"

>> "three" - "one"

'undefined operation'

'+' only arithmetic operation to be used in strings

>> string(17)

"17"

>> string(-1.123456)

"-1.1235"

>> string(3.0e8)

"300000000"

>> b = uint16(477)

uint16

477

>> string(pi)

"3.1416"

4 digits - short precision

>> string(b)

"477"

string(true) → "true"

uses number format (not exp)

string([34 0 -4 18] > [0 0 0 0])

ans =

1 × 4 string array

"true" "false" "false" "true"

>> logical ("true") → error (not possible)

∴ matlab → don't know meaning of string.

>> double ("17"). → double knows.

ans = 17

>> double ("pi")

NAN

'matlab doesn't know meaning'

Non numeric string. ← >> double ("Hello") | >> double ('hello')

ans = NaN | 104 101 101 108 111

string (pi)

ans = "3.1416" → short format

>> format long

>> double ("-1.123456")

ans = -1.12345600000000

>> format short

>> double ('17') → ASCII

ans = 49 55

>> str2num ('17')

ans = 17

>> str2double ('17')

ans = 17

>> str2double ("17")

17

>> double (string (unknown)) → convert to string then double

>> int8 ("17") → conversion to int8 from string is not possible.

↓

>> int8 (double ("17"))

ans = int8 17

>> string ({1776, logical (1776), 'independence'})

1x3 string array

"1776" "true" "independence"

cell to string ←

Access individual characters

>> a = 'MATLAB for smarties';

>> a (1:6)

ans = 'MATLAB' ↓ Not happen with strings.

>> temp = char ("independence")

>> string (temp (3:8))

"depend" ↓

>> extract Between ("independence", "in", "ence")

>> depend

↓

Now b/w

>> For string

s = "independence"

>> extract Between (s, 3, 8)

ans = "depend" → Not b/w, since includes also

58 functions - working with strings

>> doc → matlab documentation page
'language fundamentals' ← → Data types → characters & strings.

numerical functions → help page

datetime

>> datetime

ans =

datetime

28 - Jul - 2018 10:22:43 → Based on Computer time & date.

↳ pop up window (documentation)

>> class (ans)

ans =

'datetime'

Preferences

Command window → default → time format.

(we can get resolution - ms, μs, ns)

>> datetime ("yesterday"), datetime ("today"), datetime ("tomorrow")

27- Jul - 2018 28 - Jul - 2018 29 - Jul - 2018

'computer clock' ↓ Not specific to include time

>> datetime ("now")

open webpage

'function can do automatic documentation with date & time'

[~, weekday_name] = weekday (Search_time, 'long');

↓ doesn't use this argument ↓ Full name (no need for abbreviation)

>> open_webpage

Enter the url : cs103.net

At 11 - Aug - 2018 09:26:50, you opened the webpage at

cs103.net

Have a great Saturday!

open webpage

returns 0 if successful ← web('url')

webpage_log.m

>> hours (2.5)

ans =

duration

2.5 hr

>> days (2.5)

ans =

duration

2.5 days

>> datetime

11-Aug-2018 09:27:48

>> three_years_from_now = ans + years (3)

11-Aug-2021 02:55:24

>> three_days_ago = rightnow − days(3)

(datetime)

>> datetime (1989, 11, 9)

09-Nov-1989

>> datetime (1918, 11, 11, 11, 0, 0)

11-Nov-1918 11:00:00

>> timeZones → opens webpage (we can search)

'we can do addition & subtraction'

timeZones

>> timeZones

>>

'Set the time zone property of the datetime variable to the string Europe/London'

↓

dot operators

>> agreement = datetime (1918, 11, 11, 11, 0, 0);

>> agreement.TimeZone = "Europe/London"

>> agreement =

datetime

11-Nov-1918 11:00:00 → No change (∵ London)

make a copy

>> Agreement_India = agreement → copies everything

>> agreement_India.TimeZone

ans =

'Europe/London' → Just copied everything.

>> agreement = datetime(1947, 8, 15, 1, 0, 0)

agreement =

datetime

15-Aug-1947 01:00:00

>> agreement.TimeZone = 'Asia/calcutta'

15-Aug-1947 01:00:00

'changing timezone' ⟶

>> ind_ame = agreement

>> ind_ame.TimeZone ?

ans =

'Asia/calcutta'

>> ind_ame.TimeZone = 'America/New_York'

ind_ame =

14-Aug-1947 15:30:00

---

datetime(1947, 8, 15, 11)

↳ 'If you are giving hours - must give sec, min'

else: error message

3 or 6 or 7 columns array i/p

ind_ame.TimeZone → used as structs.

See field names

>> fieldnames(ind_ame)

ans =

8×1 cell array

{'Format'}
{'TimeZone'}
{'year'}
{'month'}
{'Day'}
{'Hour'}
{'minute'}
{'second'}

we can't ← add new fields (fixed)

↓

use struct to properties

↓

So its datetime not Struct.

>> class(ind_ame)

ans =

'datetime'

(London)

>> Treaty_Of_Versailles = Armistice_WW1 + day(228)

27-Jun-1919 12:00:00 ⟶ change(11:00:00)

(Day light saving time) - In European Countries (From 1916)

US - started in 1919/2007(stopped)

need only date alone

>> Treaty_of_versailles • format = 'dd-MMMMM-yyyy'

27-June-1919 → June

(or) dd/MM/yyyy → month (mm → minutes)

(or) See properties.

'eeee' – day of the week.

'eeee, MMMM, dd, yyyy' → Friday, June 27, 1919.

'Takes cares of leap years/daylight hours' – etc...

>> Treaty_of_versailles + years(100)

ans =

wednesday, June 26, 2019 → not 27

→ Reason: leap years (365.2425 – MATLAB – All years has same days)

→ length of year averaged over four centuries.

Some variations with calendars

our calendar – 365 – leap year – 366

>> Treaty_of_versailles + calyears(100)

By calendar.

Thursday, June 27, 2019

---

year, hour (Singular version) → very old functions

>> day(Treaty_of_versailles)

ans =

27

>> month(Treaty_of_versailles)

ans = 6

>> hour(")

ans =

12

>> quarter (Treaty_of versailles) → which part of year (which quarter)

2

(Financial functions)

'Financial toolbox' — kolerate — handy (billions of array elements) They won't fits in comp memories.

>> ind_ame.year

fieldnames (ind_ame)

1918

>> ind_ame.day

>> duration (3,7,43)

03:07:43 → duration

>> half_duration = duration (3,7,43)/2

1 : 33: 51

>> 2* half_duration

3:07:43

>> long_duration – half_duration

01:33:51

>> long_movie + 1

↳ added as day.

ans =

27:07:43

→ mixed mode arithmetic

hours + double = duration.

(went up by 24 hours)

'Basic time unit of duration = day' — like microsoft Excel Spreadsheet

>> double (long_movie) → not possible (use seconds, minutes, hours, etc..)

↓

matlab doesn't know format (seconds or minutes/etc).

>> seconds(long_movie)

ans = 11263

>> days(long_movie)

ans =

0.1304.

subtract two datetime

↓

duration will be the answer.

```
>> a = datetime(1756,5,17) - datetime(1763,2,15)
```

59160:00:00

```
>> years(a)
```

6.7489

```
>> b = days(datetime(1764,5,17) - datetime(1764,2,15))
b = 92
```

'Takes care of leap years'

days ⟶ double (not duration)

```
>> days(2.5)
```

ans = duration

2.5 days

⟶ polymorphism

why — Sometimes we need double, duration.

- duration — when i/ps are numbers
- number — when i/ps are duration

* double() → doesn't know unit
* hours, minutes, seconds, days, milliseconds, years → for conversions
  6 functions

double ⟷ durations → No need for 12 functions.

Points in time

```
>> Start = datetime(2018,9,10)
```

10-sep-2018

```
>> End = datetime(2018,11,19)
```

end = 19-Nov-2018

```
>> rng(0)
>> dates = Start - days(30) + days(randi(100,5,1))
```

↓ 5 days

dates =

5x1 datetime array

01-Nov-2018 00:00:00
10-NOV-2018 "
24-Aug-2018 "
11-NOV-2018 "
14-OCT-2018 "

```
>> Start <= dates & dates <= End
```

5x1 logical array

1
1
0
1
1.

```
>> Start = datetime(2002,5,18)
>> End = datetime(2002,7,18)
>> dates = Start + days(30) + days(randi(100,5,1))
>> Start <= dates & dates <= End
```

1
1
1
0
1

>> make_calendar (8, 2018)

ans =

11 x 1 String array

August 2018

Su	mo	Tu	we	Th	fr	sa
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	

datetime (2018, 3, 27) : days(14) : datetime(2018, 5, 13)

ans :

1 x 4 datetime array

27-mar-2018 10-Apr-2018 24-Apr-2018 08-May-2018

Colon operation

>> datetime (2018, 3, 27) : 14 : datetime (2018, 5, 13)

27-mar-2018 10-Apr-2018 24-Apr-2018 8-May-2018

54:08

>> days(1) : 3 : days(19)

1 day 4 days 7 days 10 days 13 days 16 days 19 days

>> calendar (2021, 9)

sep 2021

S	M	T	W	Th	F	S
0	0	0	1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	0	0
0	0	0	0	0	0	0

→

a =

0	0	0	1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	0	0
0	0	0	0	0	0	0

6x7 double array

Problems

If a(x,y) == 0

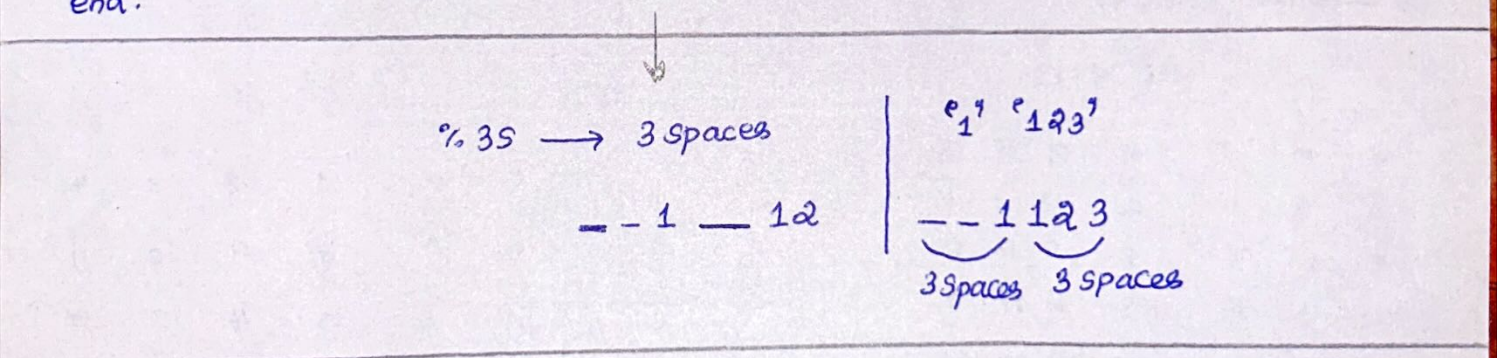
a(x,y) = ' ' → double → 32 will be replaced

↓ Convert to string then do the process.

```
function cal_str = make_calendar(n_month, n_year)

dt = datetime(n_year, n_month, 1);   01-May-2002
                                     ----→ 26
dt.format = 'MMMM yyyy'  → May 2002

cal_num = calendar(dt)
title   = string(dt)     → "May 2002"
cal_str = strings(11,1)  → preallocation
left    = blanks(floor(14 - strlength(title_str)/2));
right   = blanks(ceil(14 - strlength(title_str)/2));
cal_str([1,3,11]) = "----------------------";
cal_str(2) = sprintf('%s', left, title_str, right);
cal_str(4) = "Su Mo Tu We Th Fr Sa";
for ii = 1:6
    temp = sprintf("%3s", string(cal_num(ii,:)));
    cal_str(ii+4) = strrep(temp, "0", " ");  → replace 0 with " "
end.
```



Aim - change to a string (1×1)

%s → print as string

```
>> a =
     0   0   0   0   1   2   3

>> string(a)
    "0"  "0"  "0"  "0"  "1"  "2"  "3"

>> b = sprintf("%s", string(a))

b =
    "0000123"
```

my-cal.m

① User → month, year (9, 2021) ⟩ conversion.
calendar (2021, 9) ⟨

dat.format = 'MMMM yyyy' ⟩ calendar created
cal_number = Calendar(dat);

title = "September 2021"

>> ceil(

"September 2021" → 14

26 − len(title) = 12
6 ↙ ↘ 6

Suppose 13
6 ↙ ↘ 7

floor(12.6) → 12
ceil(12.6) → 13

>> R = sprintf("%.5", ans, "hello", ans)

R =

" hello "

%.3S → 3spaces must

'Build the function'

"0 1 2 10 20"

strrep(temp, "0", " ") ⟶ "0 1 2 1_ 2_"

All zeros are replaced

Solution

10 → nogap

"make 0 in single zeros"

↓

temp = sprintf("%3f", (1, :))

temp =

" _0 _0 _0 10 20 30"

∴ strrep(temp, '_0', ' ') ✓

Points : "1" "2" "3" → can't combine into single one...!

way

%s , temp(1,:)

---

we can use date

5-August-2018 12:49:0 → AS x axis

plot(hide_datetimes, hide_heights)

or

```
>> a = datetime("now");
>> b = a + hours(13);
>> c = a : hours(1) : b ;   → 1x14 matrix
>> d = randi(4, , 14);
>> plot(c, d)
```

↓ datetime, ↳ matrix → plotted.

Lesson:8 : I/O File

Read & write data: managed by OS

* File: exchange data b/w programs & o/p.
* File input/output.
* write a doc - generate pdf. (example)
* for permanent storage.

handle : text, binary, excel spreadsheets

---

change current folder

'pwd'

```
>> pwd
```

→ unix OS (directories)

print working directory

```
>>  C:\Users\fiza pajm   → windows
    /Users/fizapajm      → mac
```

* strrep(temp, old, new) → replace string

>> list of files in sub/current folders.

>> ls (list)

data.m
hello.m
→ array of chars. [1×198]

>> whos ans → display details.

---

change path

>> cd ("path") → change directory.

>> cd('Lesson08')

↓

> Tutorials > Lesson 08

Lesson 8 in the Tutorials (parent)

>>

Other directory - 'other files in some other parent can't be opened'

>>

Back to parent

>> cd ('..') → back to parent.

Tutorials → Lesson 01 to Lesson 08

>> we are in tutorial now

>> cd('Lesson 07') → changed (since we are in its parent)

---

* Functions take strings as an argument
* built in functions - don't need parenthesis - single word - no need quotes

>> ls 'Lesson 07'

>> help svrk
>> help('svrk') } same.

>> cd('Lesson 08')

» cd ('../..') → to grandparent. .. → parent

» mkdir ('new_folder') → make directory

» rmdir ('new-folder') → Removes only empty directories

↳ if it has something: error

Preference: Send it to trash / permanently delete.

---

* » whos
  : All variables in workspace

* save → save workspace. (Current) ⟩ preserve variables
* load → load workspace (Current)

Caution: existing variables will be overwritten

---

we can save particular / load particular variables.

save my_data_file data s a → particular variables

filename | variables

load my_data_file — data
— s
— a

» load my_data_file s a
↓
only s & a

Excel files

* xlsread
* xlswrite

» [num, txt, raw] = xlsread ('Nashville_climate.xlsx');

num → numbers from text
txt → text
raw → everything

num =

46	28	3.98
51	31	3.7
61	39	4.88
:	:	:

→ matlab identifies smallest rectangle to be extracted.

'num - smaller size than spreadsheet'

Some cells may not have numerical data : NaN'

: : :

Nan Nan Nan → See slides

: : :

>> txt = all the texts   txt → cell array

txt =

[1x30 char] " " " → like in cells order

[1x40 char] " "

" 'High temp'

'Jan' "

Too long strings to print in window (matlab prints type) alone

text → diff no. of chars

∴ rect array - not possible

---

* Type : txt → cell array.

>> raw

raw =

union to num & txt

>> temps = xlsread('Nashville-climate-data.xlsx')

temps =

. . . . –

. . . .

'Loaded'

[temps txt]

(or)

[~ txt]

→ alone

>> num = xlsread('Nashville-climate-data.xlsx', 1, 'D15')

1 → No. of sheet (or) Name

'D15' → Name of cell

61

(or)

>> num = xlsread('Nashville-climate-data.xlsx', 1, 'D15:E17')

D15:E17 → up to (rectangular)

num =

61.000 . . .

. . . . . .

. . . . . .

* xlswrite → write in xls files
* But on MAC, matlab provides a limited alternative
  * xlswrite → writes text files (Instead of excel)
  * only numerical arrays can be written
  * Each line is wrote on a seperate line by comma

Seperated Values (numbers on a row seperated by commas.)

↓

* Comma seperated values (CSV)

caution: we can overwrite: without even getting any warning.

'only modify the cell - that actually you write'

other cells won't be affected

	A	B	C	D	E	F
1		Abilene, TX	Akron, OH	Abbu	Alexa	Allentown, PA
2	Abi	0	1481	724	1615	1730
3	Akron, OH	1481	0	2185	382	552
4	Albuquerque, NM	724	2185	0	2331	2447
5	Alexandria, VA	1615	382	2331	0	195
6	Allentown, PA	1730	552	2447	195	0

Problem

get_destance → Two City names

→ distance (B/w two cities)

one (or) both cities - not in file → returns -1

Analyzing problem

num → doesn't have cities

txt → doesn't have values

raw → fit for our purpose.

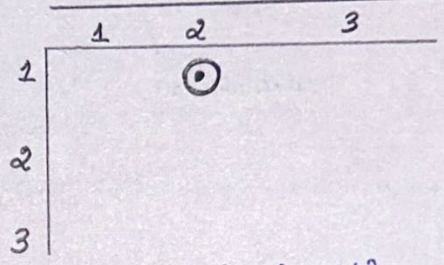
Sample

```
>> [~, ~, raw] = xlsread('Distances 1.xlsx', 1, 'A1:E5');
                                  ↓          ↓        ↓
                                 File.     1Sheet   Cells
```

```
>> raw =

{ [  NAN] }  . . . -
{'Abilene, TX' }  . . . -        ⟶ cell type.
{'Akron, OH' }  . . . -
{  . .  }  . . . -
```

Take row & col number from city



1, 2 → Concentrate on col number ∴ Row & col are same

main function

```
function distance = get_distance(city1, city2)

[~, ~, data] = xlsread('Distances.xlsx');

city = string(data(1, :));  ⟶ For comparing city

city1 = city_valid(city1, city);
                   ↓        ↓
                city 1   city names list

city2 = city_valid(city2, city);  ⟶ city name list
                   ↓
                city 2

if (city1 > 1 && city2 > 1)

    distance = data{city2, city1};
                        ↓      ↓
                      row    column   ⟩ replacable

else

    distance = -1;  ⟶ city not in list

end
end
```

city1, city2 → Index of city 1 & 2

helping function

```
function index = city_valid(cit, city)
for x = 1: length(city)
    if strcmp(city(1, x), cit)
        index = x; return;
    else
        index = 0;
    end
end
end
```

city → list of cities (string)

→ Compare with each elements of column array of cities match → That index is the index eg city1(cit)

---

text files

* Strings of characters: encoded: Matlab takes care of those.
* open/close - permission from OS.

* `fid = fopen(filename, permission)`
  `fclose(fid)` → opened file
* permission: read, write, overwrite.

'rt' - open text file - read
'wt' - write (open + discard existing contents)
'at' - append data to end (open/create)
'r+t' - open - don't create - read & write
'w+t' - open/create - read/write - discard existing content
'a+t' - open/create - read/write - append data to end.

r+ → both read & write (don't create)

t → text file

fopen() → if we don't have write permission/error

↓

returns negative number

Climate data for Nashville, TN
(Average highs (F), lows (F), and Precip (in)

	High	Low	Precip
Jan :	46.00	28.00	3.98
Feb :	51:00	31.00	3.70
March :	61:00	39.00	4.88
April :	70:00	47.00	3.94
May :	78:00	57.00	5.08
June :	85:00	65.00	4.09
	High	Low	Precip
July :	89:00	69.00	3.78
Aug :	88.00	68.00	3.27
Sep :	82.00	61.00	3.58
Oct :	71.00	49.00	2.87
Nov :	59.00	40.00	4.45
Dec :	49.00	31.00	4.53

>> fprintf('%e %g %.8 %.5f', string([1 2 4])

1 2 4

%.2 → 2 zeros (or)

%.5.00 → 2 zeros.

write_text.m

file = fopen(file_name, 'w+t')

If file < 0 → error while opening
return;

%.5s → formatting

Text files – Broken into lines – Reading text files

→ one line at a time
→ type points a text file in the command window

fgets → read a line & returns a string. (one time)
for multiple lines → loop it

when nothing to return → −1

So while (fgets(oneline))
use fgets()

Point_line_by_line.m

Reading - easy (txt files) → converting in to numerical data is much harder

recognize digits, decimal points etc.. → Tedious job.

'cumbersome'

↓

'Poor way to store data'

Binary files

* 'Any file: not a text file – Binary file'
* Represent numbers (have all data)
* we need to type..

> fopen, fclose
>
> 'r', 'w', 'a', 'r+', 'w+', 'a+'

```
function write_array_bin(A, filename)
fid = fopen(filename, 'w+');
if fid < 0
    error('error opening file %s\n', filename);
end.
fwrite(fid, A, 'double') → 'write an array(double) A'
fclose(fid)
```

Reading the array – read a double array from file

* A = fread(fid, inf, data_type)
  ↓
  no. of items (All - inf) – read everything.
* fclose(fid); → else it can't be opened by other functions.

'write_binaryfile.m'

>> write_binary(a, 'datafile.dat') ; a = randn(10,12)

open the written file

'write_binary.mat' – using textread

(read_line_by_line.m)

Problem:

>> writebinaryfiles(a, ''write_binary.dat'')

>> read_line_by_line(''write_binary.dat'') → Garbage lines

(Not a text file – So Garbage encoded

>> read_binaryfile('double', 'write_binary.dat')

ans =

120x1 double array.

∴ Binary files – not encoded – will be encoded by text encoding scheme.

(String of doubles)

>> read_binaryfile( ) → recovers but not in 10x12 (120x1) ✓

↳ original.

fwrite → writes the each element in a long – one long stream of numbers with nothing written about the dimensions of the array.

↓

120x1 double vector

* read_binaryfile() → simply read as it is

Preserve dimensions: using binary files

↙ Remember that file with a certain name has an array such & such dimensions

↘ write the dimension info into the file as well.

(Better option)

(reader wishes to know format of the file)

write_dims_array_bin.m

```
function write_dims_array_bin (A, filename)
fid = fopen (filename, 'w+');
if fid < 0
    error ('error opening file %s \n', filename);
end
dims = size(A);
fwrite (fid, length (dims), 'double' ); → In case dimension is alway 1×2 (no problem) no need
fwrite (fid, A, 'double'); ⟶ dimension
fclose (fid);
```

(A×2)

2 4 2

↓

take 2 numbers

Problems

```
1  2  3  4
5  6  7  8   ⟶ fwrite ⟶
9  10 11 12
```

A

.dat file

{1 5 9 2 6 10 3 7 11 4 8 12}

↓

one long line.

B

∴

A(1,1) = B(1)	A(1,2) = B(4)	A(1,3) = B(7)
A(2,1) = B(2)	A(2,2) = B(5)	A(2,3) = B(8)
A(3,1) = B(3)	A(3,2) = B(6)	A(3,3) = B(9)

'written Columnwise'

mod_read_binaryfile.m

mod_write_binaryfiles.m

* save dim→2×3 → as 1 & 2 nd element of .mat file
* using loop → mod_write.m → create the reoriented matrix.

>> isequal (1, 1) → 1 (only both are true)

Efficient method

reshape( ) → reshapes the matrix (vectors)

Take dim

$$a = \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \end{matrix} \quad (2)$$

$$b = \begin{bmatrix} 1 & 3 & 5 & 7 \\ 2 & 4 & 6 & 8 \end{bmatrix}$$

reshape(a, [2,4]) →

1	3	5	7
2	4	6	8

Single data type – matlab

* y = single(10); → 4-byte (32-bit) – floating point values

fread returns → column vectors

Possible to return any type: – Altern 3rd argument

```
01 = char(fread(fid, nums(1), 'char => char'))';
```

i/p o/p

Same (in this case)

'single => int64'

To know:

'int16', 'single'

fread(file, int, 'int16')

Keeping file sizes small: important when handling huge data

↓

Allot according

(need – single or int8 instead of double)

* Find real problem – Solve them – give real solutions

matlab inbuilt programs

>> edit numgstr [popup] → mathworks programmers.

(or)

Matlab central

Saddle points

Saddle → M (saddle point: whose element is greater or equal to every element in its row, & less than or equal to every element in its column

↳ Indices (matrix of saddle points)

Indices

Indices: 2 columns → row, column) index

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$, Compare 1 with all elements of row >= , 1 with all elements of col <=) saddle

row = 51, col = 41

my idea

1 for loop x = 1 : row → row

for loop y = 1 : col → col

r = 0, c = 0;

for s = 1 : col

if M(x,y) >= M(x,s) → saddle point

r = r + 1;

r = col

end

end

for k = 1 : row

if M(x,y) <= M(k,y) → saddle point

c = c + 1;

c = row

end

end

both True → saddle point.

'saddle_point_me.m'

```
>> min([2 1 3])
1
>> max([1 3 2])
3
```

>> Saddle_min_max_method.m

Logical indexing

```
>> min(2,3)
2
```

```
>> a = randi(10,4,4)
min(a)
```

→ default.

2 1 2 2 → In each column

>> a = min(a,[],1) → 2 1 2 2 (default case 1, [])

↓ No input (2nd one)

↳ Columnwise

>> a = min(a,[],2)

↳ Row wise.

7
1
2
2

---

* Compare row wise greatest with each element in row

```
a =
    9    7   10   10
   10    1   10    5
    2    3    2    9
   10    6   10    2
```

b = 10 10 9 10

↓ Row wise greatest

c = 2 1 2 2

↓ Columnwise Smallest.

>> a ≥ b

```
[ 0  0  1  1
  0  0  1  0
  0  0  0  1
  1  0  1  0 ]
```

→ 10 ≥ 10

>> a ≤ c

```
[ 0  0  0  0
  0  1  0  0
  1  0  1  0   ← 2 ≤ 2
  0  0  0  1 ]
```

* Number both greatest of row, Smallest of column has 1 in it.

* S & L = $\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$ → no saddle points.

---

* isempty([]) → 1

* isrow([1 2]) → 1 (row vector)

[4 3 5 12 ; -1 0 -2 0 -1 ; -4 1 4 3 5 ;
-3 0 -1 0 -2 ; 3 2 -7 3 8]

Some cases answer will be like

[2 2 3 1 4 4] Instead of

2	1
2	4
3	4

Solution

[Row' col'] = [Row' col']

Row'	col'
2	1
2	4
3	4

when even isrow([row col]) → safety measure (if needed)

ans = [row' col']

Image blur

function output = blur(img, w) img → 2d array (0, 255 values)

Blurring → Average the pixel values in the vicinity of every pixel.

outer pixel value = mean of the pixels in a square submatrix (2w+1)

★ when $w = 1 \rightarrow$ use 3×3 matrix

$w = 1$ at index (1, 1)

mean of (1,1), (1,2), (2,1), (2,2)

I/p & output are units

1	2	3
4	5	6
7	8	9

>> nan (1,4)

NaN NaN NaN NaN

$2w + 1 \rightarrow$ 3×3 sub-matrix

11	12
21	22

when row = 1, Col = 1

If $w = 1 \rightarrow$

11	12
13	14

r1 =

r2 =

r3 =

r4 = Col + 1

11	12	13
21	22	23

1 : 2 , 2 : 3

Problem: 11 (12 / 21 (22 → only 1×2) → So take front & back

when w = 1

11	12	13
21	22	23

max: 3×3

?

```
function output = blur(img, w)
img = double(img);
[row  col] = size(img);
output = zeros(row, col); ⟶ Initialize matrix for speed
for i = 1: row
    for j = 1:col
        r1 = i - w; ⟶ sub matrix may be upto 3×3 matrix for w = 1
        r2 = i + w;                                  d = 2(1) + 1 = 3
        c1 = j - w;
        c2 = j + w;
        if r2 > row
            r2 = row;
        end
        if c2 > col
            c2 = col;
        end
        if c1 < 1
            c1 = 1;
        end
        if r1 < 1
            r1 = 1;
        end
        mat = img(r1:r2, c1:c2);
        output(i, j) = mean(mat(:));
    end
end
output = uint8(output);
end
```

Echogen

Echogen - add an echo effect to the audio recording. An echo is the original signal delayed & attenuated. First compute echo then add to the original signal with the correct offset.

output = echo_gen (input, fs, delay, amp);

input → column vector [values b/w −1 and 1] - representing a time series of digitized sound data

fs - sampling rate - CD player uses 44,100 samples/second.

delay - delay of the echo in seconds

* Echo should start after the delay' seconds passed from the start of the start of the audio signal.
* amp - amplification of echo [As values < 1 → not loud, amp > 1]

* o/p - column vector having orig sound with echo superimposed
* o/p vector - no longer the same size (As delay ≠ 0)

* (round to the nearest value) : no of points - to get the delay → ceil / floor
* Sound → values b/w −1 & 1. So if the echo causes values to be outside of this range - scale down the entire vector, so that all values are within the range while retaining their relative amplitudes.

audiofiles : splat, gong & handel

* load gong %. loads two variables, y and Fs
* sound (y, Fs) %. outputs sound.

audio → 2 columns - stereo file
1 column - mono audio file.

10
5
20
30

→ scale all to 1
÷ max
(30)

$\begin{pmatrix} 1/3 \\ 1/6 \\ 2/3 \\ 1 \end{pmatrix}$ → scaled.

function output = echo_gen (s, FS, delay, amp)

dt = 1/FS ;

N = round (delay/dt) → calculate no. of points.

∴ 1 second FS samples are required, for delay seconds we need to add delay * FS sample points.

FS * delay (or) $\frac{delay}{dt}$ = delay * FS

(or)

$\frac{delay}{time\ per\ sample}$ = Total sample req.

output = superimpose (echo signal + original signal)

echosignal → delay (0's) up to delay time then original signal.

Add = echo_sig + org signal

↳ due to add of delay in echo_sig (size - add zeros at end - delay)

Scale:

div by maximum

'echo_gen.m'