

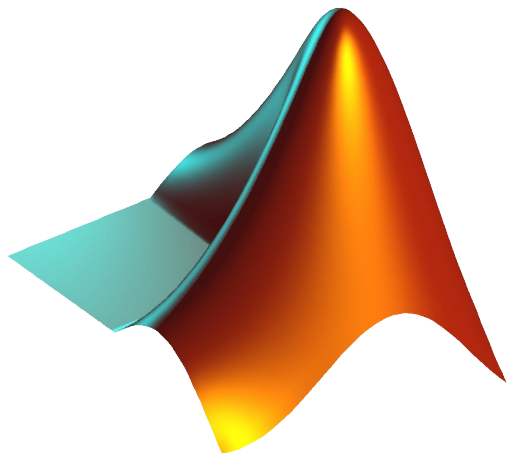
CS 1112 Introduction to Computing Using MATLAB

Instructor: Dominic Diaz

Website:

<https://www.cs.cornell.edu/courses/cs1112/2022fa/>

Today: Vectorized code + matrices

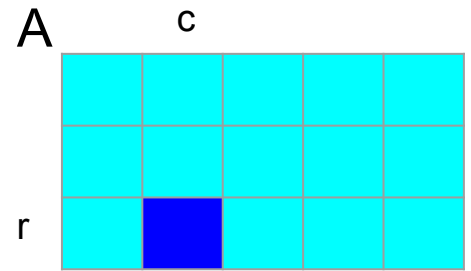


Agenda and announcements

- Last time
 - Vectorized computation
 - 2D arrays - matrix
- Today
 - More matrices
- Announcements
 - Discussion 08 next week (10/12) will have optional problems – problems to help you study for prelim 1
 - Project 3 late deadline TONIGHT 10/6 (only 5% deduction for late submission)
 - “Check your prelim 1 time/location” on CMS—read the “grading comment” to find exam time/location. **Any request for alternative arrangements (including conflicting exams) is due as a “regrade request” in CMS by 10/7 at 11 PM.**
 - Consultants will be holding tutoring
 - Wed & Thurs (10/12 - 10/13)
 - Sun & Mon (10/16 - 10/17)
 - NO OFFICE HOURS OVER BREAK

2D array: matrix

- A 2D array is like a table, and is also called a matrix
- Two indices identify the position of a value in a matrix



$A(r, c)$

First index: row index
Second index: column index

- If we set $[nr, nc] = \text{size}(A)$, then
 - $1 \leq r \leq nr$
 - $1 \leq c \leq nc$

Example: A cost/inventory problem

- A merchant has 3 different suppliers that stock 5 different products
- The cost of each product varies from supplier to supplier
- Inventory amount varies from supplier to supplier

Two matrices storing all relevant information:

C

10	36	22	15	62
12	35	20	12	66
13	37	21	16	59

$C(i, j)$ is what it costs
supplier i to supply product j

Inv

38	5	99	34	42
82	19	83	12	42
51	29	21	56	87

$Inv(i, j)$ is how many units
supplier i has of product j

Problem statement

A customer submits a purchase order that is to be shipped from a single supplier.

- Among the suppliers that can fill the order, who can do it most cheaply?
 - Which suppliers have enough inventory?
 - How much does it cost each supplier to fill the order?

C

10	36	22	15	62
12	35	20	12	66
13	37	21	16	59

$C(i, j)$: what it costs supplier i to supply product j

Inv

38	5	99	34	42
82	19	83	12	42
51	29	21	56	87

$Inv(i, j)$: how many units supplier i has of product j

PO

1	0	12	29	5
---	---	----	----	---

$PO(j)$: is the number of product j 's the client wants

Which supplier can fill order?

Supplier 2:

$1 < 82$, $0 < 19$, $12 < 83$, $29 > 12$, $5 < 42$

No, warehouse 2 cannot fill the order!

```
% Determine if supplier 2 can fill order
```

```
i = 2;
```

```
tf = true;
```

```
for j = 1:length(PO)
```

```
    tf = tf && Inv(i,j) >= PO(j);
```

```
end
```

C

10	36	22	15	62
12	35	20	12	66
13	37	21	16	59

$C(i, j)$: what it costs supplier i to supply product j

Inv

38	5	99	34	42
82	19	83	12	42
51	29	21	56	87

$Inv(i, j)$: how many units supplier i has of product j

PO

1	0	12	29	5
---	---	----	----	---

$PO(j)$: is the number of product j 's the client wants

Which supplier can fill order?

```
% determine if supplier i can fill order  
i = 2;  
tf = true;  
for j = 1:length(PO)  
    tf = tf && Inv(i,j) >= PO(j);  
end
```

Variable workspace	
i	2
tf	T
j	1

C

10	36	22	15	62
12	35	20	12	66
13	37	21	16	59

$C(i,j)$: what it costs supplier i to supply product j

Inv

38	5	99	34	42
82	19	83	12	42
51	29	21	56	87

$Inv(i,j)$: how many units supplier i has of product j

PO

1	0	12	29	5
---	---	----	----	---

$PO(j)$: is the number of product j 's the client wants

Which supplier can fill order?

```
% determine if supplier i can fill order  
i = 2;  
tf = true;  
for j = 1:length(PO)  
    tf = tf && Inv(i,j) >= PO(j);  
end
```

Variable workspace	
i	2
tf	T
j	2

C

10	36	22	15	62
12	35	20	12	66
13	37	21	16	59

$C(i,j)$: what it costs supplier i to supply product j

Inv

38	5	99	34	42
82	19	83	12	42
51	29	21	56	87

$Inv(i,j)$: how many units supplier i has of product j

PO

1	0	12	29	5
---	---	----	----	---

$PO(j)$: is the number of product j 's the client wants

Which supplier can fill order?

```
% determine if supplier i can fill order
i = 2;
tf = true;
for j = 1:length(PO)
    tf = tf && Inv(i,j) >= PO(j);
end
```

Variable workspace	
i	2
tf	T
j	3

C

10	36	22	15	62
12	35	20	12	66
13	37	21	16	59

$C(i,j)$: what it costs supplier i to supply product j

Inv

38	5	99	34	42
82	19	83	12	42
51	29	21	56	87

$Inv(i,j)$: how many units supplier i has of product j

PO

1	0	12	29	5
---	---	----	----	---

$PO(j)$: is the number of product j 's the client wants

Which supplier can fill order?

```
% determine if supplier i can fill order  
i = 2;  
tf = true;  
for j = 1:length(PO)  
    tf = tf && Inv(i,j) >= PO(j);  
end
```

Variable workspace	
i	2
tf	F
j	4

C

10	36	22	15	62
12	35	20	12	66
13	37	21	16	59

$C(i,j)$: what it costs supplier i to supply product j

Inv

38	5	99	34	42
82	19	83	12	42
51	29	21	56	87

$Inv(i,j)$: how many units supplier i has of product j

PO

1	0	12	29	5
---	---	----	----	---

$PO(j)$: is the number of product j 's the client wants

Which supplier can fill order?

```
% determine if supplier i can fill order
```

```
i = 2;
```

```
tf = true;
```

```
for j = 1:length(PO)
```

```
    tf = tf && Inv(i,j) >= PO(j);
```

```
end
```

Variable workspace	
i	2
tf	F
j	4

C

10	36	22	15	62
12	35	20	12	66
13	37	21	16	59

$C(i,j)$: what it costs supplier i to supply product j

Inv

38	5	99	34	42
82	19	83	12	42
51	29	21	56	87

$Inv(i,j)$: how many units supplier i has of product j

PO

1	0	12	29	5
---	---	----	----	---

$PO(j)$: is the number of product j 's the client wants

Which supplier can fill the order?

```
function tf = iCanFill(i, Inv, PO)
% tf is true if supplier i can fill the
% purchase order. Otherwise, false.

% determine if supplier i can fill order
tf = true;
for j = 1:length(PO)
    tf = tf && Inv(i,j) >= PO(j);
end
```

C

10	36	22	15	62
12	35	20	12	66
13	37	21	16	59

$C(i, j)$: what it costs supplier i to supply product j

Inv

38	5	99	34	42
82	19	83	12	42
51	29	21	56	87

$Inv(i, j)$: how many units supplier i has of product j

PO

1	0	12	29	5
---	---	----	----	---

$PO(j)$: is the number of product j 's the client wants

Which supplier can fill the order?

```
function tf = iCanFill(i, Inv, PO)
% tf is true if supplier i can fill the
% purchase order. Otherwise, false.

% determine if supplier i can fill order
nProd = length(PO);
j = 1;

while j <= nProd && Inv(i,j) >= PO(j)
    j = j + 1;
end
tf = _____;
```

Alternative solution!

C

10	36	22	15	62
12	35	20	12	66
13	37	21	16	59

$C(i, j)$: what it costs supplier i to supply product j

Inv

38	5	99	34	42
82	19	83	12	42
51	29	21	56	87

$Inv(i, j)$: how many units supplier i has of product j

PO

1	0	12	29	5
---	---	----	----	---

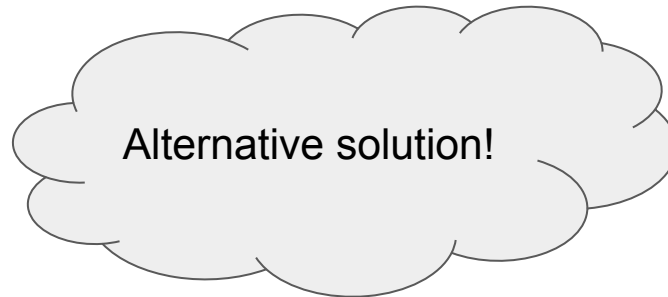
$PO(j)$: is the number of product j 's the client wants

Which supplier can fill the order?

```
function tf = iCanFill(i, Inv, PO)
% tf is true if supplier i can fill the
% purchase order. Otherwise, false.

% determine if supplier i can fill order
nProd = length(PO);
j = 1;

while j <= nProd && Inv(i,j) >= PO(j)
    j = j + 1;
end
tf = j > nProd;
```



C

10	36	22	15	62
12	35	20	12	66
13	37	21	16	59

$C(i, j)$: what it costs supplier i to supply product j

Inv

38	5	99	34	42
82	19	83	12	42
51	29	21	56	87

$Inv(i, j)$: how many units supplier i has of product j

PO

1	0	12	29	5
---	---	----	----	---

$PO(j)$: is the number of product j 's the client wants

How much for supplier i to fill order?

```
function theBill = iCost(i,C,PO)
% The cost when factory i fills
% the purchase order

nProd = length(PO);
theBill = 0;
for j=1:nProd
    theBill = theBill + C(i,j)*PO(j);
end
```

C

10	36	22	15	62
12	35	20	12	66
13	37	21	16	59

$C(i, j)$: what it costs supplier i to supply product j

Inv

38	5	99	34	42
82	19	83	12	42
51	29	21	56	87

$Inv(i, j)$: how many units supplier i has of product j

PO

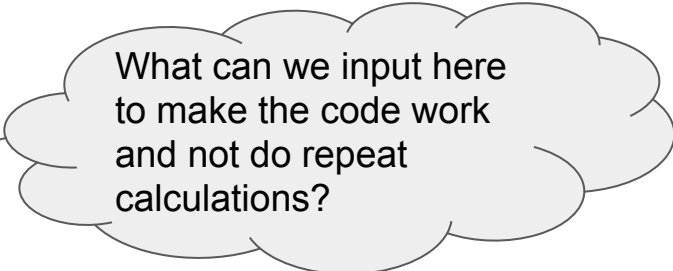
1	0	12	29	5
---	---	----	----	---

$PO(j)$: is the number of product j 's the client wants

Finding the cheapest

```
function [iBest,minBill] = Cheapest(C,Inv,PO)
% output the index (iBest) of the supplier that can
% complete the product order at the cheapest price
% (minBill).
```

```
[nFact,~] = size(C);
iBest = 0;
minBill = _____;
for i = 1:nFact
    iBill = iCost(i,C,PO);
    if iBill < minBill && iCanFill(i,Inv,PO)
        iBest = i;
        minBill = iBill;
    end
end
```



What can we input here to make the code work and not do repeat calculations?

C

10	36	22	15	62
12	35	20	12	66
13	37	21	16	59

$C(i,j)$: what it costs supplier i to supply product j

Inv

38	5	99	34	42
82	19	83	12	42
51	29	21	56	87

$Inv(i,j)$: how many units supplier i has of product j

PO

1	0	12	29	5
---	---	----	----	---

$PO(j)$: is the number of product j 's the client wants

Finding the cheapest

```
function [iBest,minBill] = Cheapest(C,Inv,PO)
% output the index (iBest) of the supplier that can
% complete the product order at the cheapest price
% (minBill).
```

```
[nFact,~] = size(C);
iBest = 0;
minBill = inf;
for i = 1:nFact
    iBill = iCost(i,C,PO);
    if iBill < minBill && iCanFill(i,Inv,PO)
        iBest = i;
        minBill = iBill;
    end
end
```

C

10	36	22	15	62
12	35	20	12	66
13	37	21	16	59

$C(i,j)$: what it costs supplier i to supply product j

Inv

38	5	99	34	42
82	19	83	12	42
51	29	21	56	87

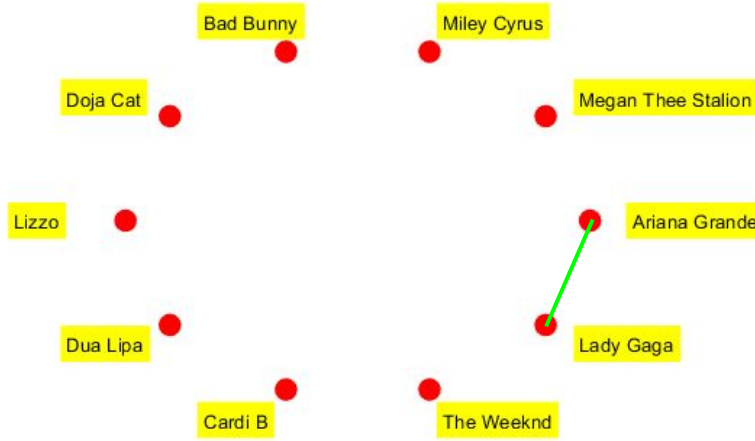
$Inv(i,j)$: how many units supplier i has of product j

PO

1	0	12	29	5
---	---	----	----	---

$PO(j)$: is the number of product j 's the client wants

Example: Plotting collabs between singers



Rain on me

Don't call me angel

0	1	1	0	1	0	1	0	1	1
1	0	0	0	1	0	1	1	0	0
1	0	0	0	0	0	1	0	1	0
0	0	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	1	0	0
1	1	1	0	0	0	0	0	0	0
0	1	0	1	0	1	0	0	0	0
1	0	1	0	1	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0

Say we have collaborations between singers stored in a matrix called `collabs`. $\text{collabs}(i, j) = 1$ if singer i has a song with singer j . Otherwise, 0 .

Singer 1: Ariana

Singer 2: Megan Thee Stallion

Singer 3: Miley Cyrus

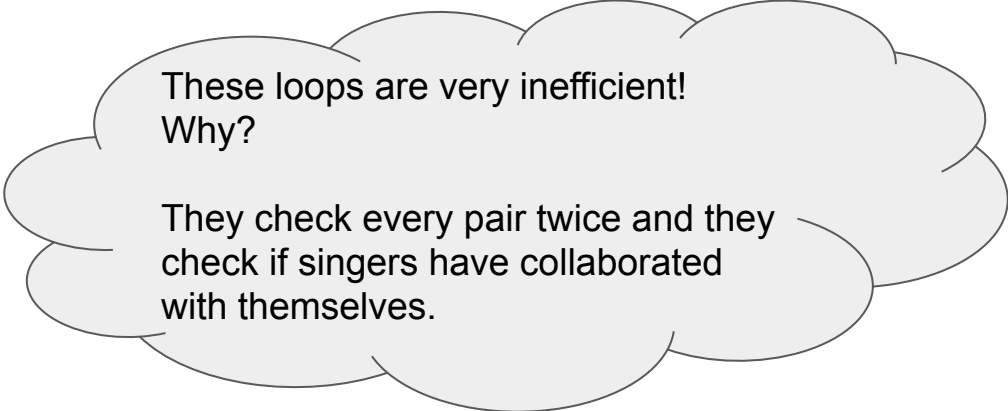
...

Singer 10: Lady Gaga

Plotting lines between singers who have a song together

```
[nr, nc] = size(collabs);
```

```
for r = 1:nr  
    for c = 1:nc  
        if collabs(r,c) == 1  
            % plot a line between celeb i and j  
        end  
    end  
end  
end
```



These loops are very inefficient!
Why?

They check every pair twice and they check if singers have collaborated with themselves.

Because we only need to check each pair once and we do not need to check if a singer collaborated with themselves, we only need to loop through the green cells.

0	1	1	0	1	0	1	0	1	1
1	0	0	0	1	0	1	1	0	0
1	0	0	0	0	0	1	0	1	0
0	0	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	1	0	0
1	1	1	0	0	0	0	0	0	0
0	1	0	1	0	1	0	0	0	0
1	0	1	0	1	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0

```
[nr, nc] = size(collabs);
```

```
for r = 1:nr
    for c = r+1:nc
        if collabs(r,c) == 1
            % plot a line between celeb i and j
        end
    end
end
```

```
[nr, nc] = size(collabs);
```

```
for r = 1:nr
```

```
    for c = r+1:nc
```

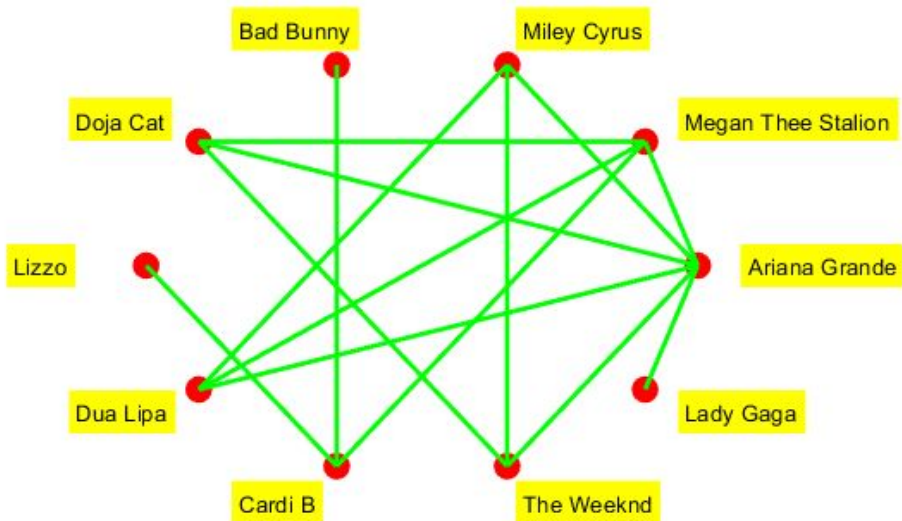
```
        if collabs(r,c) == 1
```

```
            % plot a line btwn celebs i,j
```

```
        end
```

```
    end
```

```
end
```



0	1	1	0	1	0	1	0	1	1
1	0	0	0	1	0	1	1	0	0
1	0	0	0	0	0	1	0	1	0
0	0	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	1	0	0
1	1	1	0	0	0	0	0	0	0
0	1	0	1	0	1	0	0	0	0
1	0	1	0	1	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0

See drawCelebNetwork.m for code. It uses constructs like cell arrays and char arrays that we have not covered in this class.... yet!

Poll Everywhere

0	1	1	0	1	0	1	0	1	1
1	0	0	0	1	0	1	1	0	0
1	0	0	0	0	0	1	0	1	0
0	0	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	1	0	0
1	1	1	0	0	0	0	0	0	0
0	1	0	1	0	1	0	0	0	0
1	0	1	0	1	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0

```
[nr, nc] = size(collabs);  
for r = 1:nr  
    for c = 1:r-1  
        if collabs(r,c) == 1  
            % plot line btwn celeb i, j  
        end  
    end  
end
```

Accessing more than just one element at a time

`a = M(1, :);`

0	1	5	9	7
---	---	---	---	---

`b = M(1, 2:4);`

1	5	9
---	---	---

`c = M(:, 2:4);`

1	5	9
-5	-1	20
-8	13	4

`d = M(1:2:3, 1:2:5);`

0	5	7
19	13	2

`e = M(M(1,2), 1:M(end,end):end);`

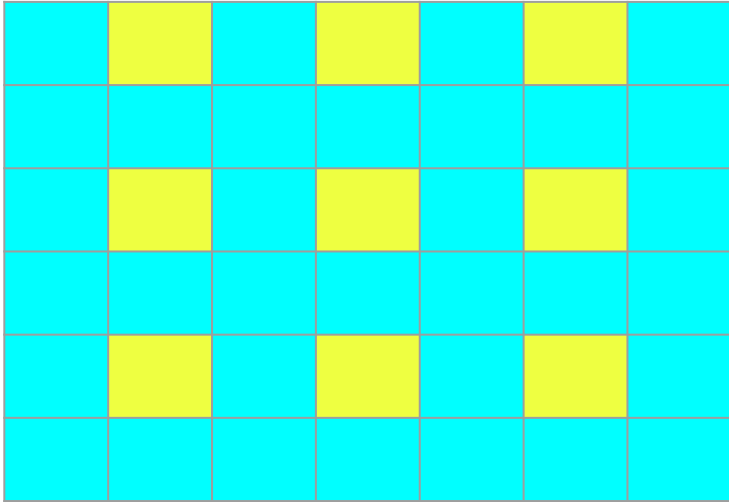
0	5	7
---	---	---

M

0	1	5	9	7
7	-5	-1	20	26
19	-8	13	4	2

How can you access just the yellow elements?

M



```
A = M(1:2:end, 2:2:end);
```