L20. More on 2D Arrays

Operations
Subscripting
Functions* & 2D Arrays

*Will see two new things.
Boolean-valued functions
Functions that have a function as a parameter.
Two Applications

A commercial setting that involves cost arrays, inventory arrays, and purchase orders.

A setting that requires the visualization of a function of two variables $f(x,y)$ via contour plotting.
A Cost/Inventory Setting

A company has 3 factories that make 5 different products.

The cost of making a product varies from factory to factory.

The inventory varies from factory to factory.
Problems

A customer submits a purchase order that is to be filled by a single factory.

1. How much would it cost a factory to fill the order?
2. Does a factory have enough inventory to fill the order?
3. Among the factories that can fill the order, who can do it most cheaply?
The value of $C(i, j)$ is what it costs factory $i$ to make product $j$. 

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>36</td>
<td>22</td>
<td>15</td>
<td>62</td>
</tr>
<tr>
<td>12</td>
<td>35</td>
<td>20</td>
<td>12</td>
<td>66</td>
</tr>
<tr>
<td>13</td>
<td>37</td>
<td>21</td>
<td>16</td>
<td>59</td>
</tr>
</tbody>
</table>
The value of $\text{Inv}(i,j)$ is the inventory in factory $i$ of product $j$. 

The table shows:

<table>
<thead>
<tr>
<th></th>
<th>38</th>
<th>5</th>
<th>99</th>
<th>34</th>
<th>42</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inv:</td>
<td>82</td>
<td>19</td>
<td>83</td>
<td>12</td>
<td>42</td>
</tr>
<tr>
<td></td>
<td>51</td>
<td>29</td>
<td>21</td>
<td>56</td>
<td>87</td>
</tr>
</tbody>
</table>
Purchase Order

PO: 1 0 12 29 5

The value of PO(j) is the number product j’s that the customer wants.
How Much Does it Cost for Each Factory to Process a Purchase order?
For factory 1:

\[ 1 \times 10 + 0 \times 36 + 12 \times 22 + 29 \times 15 + 5 \times 62 \]
For factory 1:

\[
\begin{align*}
s &= 0; \\
\text{for } j=1:5 \\
  s &= s + C(1,j) \ast PO(j) \\
\text{end}
\end{align*}
\]
For factory 1:

```matlab
s = 0;
for j=1:5
    s = s + C(1,j) * PO(j)
end
```
For factory 1:

```matlab
s = 0;
for j=1:5
    s = s + C(1,j) * PO(j)
end
```
For factory 1:

```matlab
s = 0;
for j=1:5
    s = s + C(1,j) * PO(j)
end
```
For factory 1:

\[
s = 0; \\
\text{for } j=1:5 \\
\hspace{1cm} s = s + C(1,j) \times PO(j) \\
\text{end}
\]
For factory 2:

```matlab
s = 0;
for j=1:5
    s = s + C(2,j)*PO(j)
end
```
For factory i:

```matlab
s = 0;
for j=1:5
    s = s + C(i,j)*PO(j)
end
```
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
Finding the Cheapest

<table>
<thead>
<tr>
<th></th>
<th>10</th>
<th>36</th>
<th>22</th>
<th>15</th>
<th>62</th>
<th>1019</th>
</tr>
</thead>
<tbody>
<tr>
<td>C:</td>
<td>12</td>
<td>35</td>
<td>20</td>
<td>12</td>
<td>66</td>
<td>930</td>
</tr>
<tr>
<td></td>
<td>13</td>
<td>37</td>
<td>21</td>
<td>16</td>
<td>59</td>
<td>1040</td>
</tr>
<tr>
<td>PO:</td>
<td>1</td>
<td>0</td>
<td>12</td>
<td>29</td>
<td>5</td>
<td></td>
</tr>
</tbody>
</table>

As computed by iCost
**Finding Cheapest: Initialization**

<table>
<thead>
<tr>
<th>PO:</th>
<th>1</th>
<th>0</th>
<th>12</th>
<th>29</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>C:</td>
<td>10</td>
<td>36</td>
<td>22</td>
<td>15</td>
<td>62</td>
</tr>
<tr>
<td></td>
<td>12</td>
<td>35</td>
<td>20</td>
<td>12</td>
<td>66</td>
</tr>
<tr>
<td></td>
<td>13</td>
<td>37</td>
<td>21</td>
<td>16</td>
<td>59</td>
</tr>
</tbody>
</table>

**iBest:** 0  
**minBill:** inf

Can we do better?
A Note on “inf”

A special value that can be regarded as + infinity.

\[ x = \frac{10}{0} \] assigns inf to \( x \)
\[ y = 1+x \] assigns inf to \( y \)
\[ z = \frac{1}{x} \] assigns zero to \( z \)
\[ w < \text{inf} \] is always true if \( w \) is numeric
**Improvement at i = 1**

<table>
<thead>
<tr>
<th>PO</th>
<th>1</th>
<th>0</th>
<th>12</th>
<th>29</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>36</td>
<td>22</td>
<td>15</td>
<td>62</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>35</td>
<td>20</td>
<td>12</td>
<td>66</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>37</td>
<td>21</td>
<td>16</td>
<td>59</td>
<td></td>
</tr>
</tbody>
</table>

**iBest**: 1  
**minBill**: 1019
**Improvement at i = 2**

| C:    |   |   |   |   | PO:    |
|-------|--|--|--|--|--|-------|
| 10    | 36| 22| 15| 62| 1     |
| 12    | 35| 20| 12| 66| 0     |
| 13    | 37| 21| 16| 59| 12    |
| 1019  |   |   |   |   | 1040  |
| 930   |   |   |   |   | 930   |

**iBest:** 2  
**minBill:** 930
No Improvement at $i = 3$

<table>
<thead>
<tr>
<th>C:</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>36</td>
<td>22</td>
<td>15</td>
<td>62</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>35</td>
<td>20</td>
<td>12</td>
<td>66</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>37</td>
<td>21</td>
<td>16</td>
<td>59</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PO:</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>12</td>
</tr>
<tr>
<td>29</td>
</tr>
<tr>
<td>5</td>
</tr>
</tbody>
</table>

$i_{\text{Best}}$: 2  
\text{minBill}: 930
Finding the Cheapest

\[ i\text{Best} = 0; \text{minBill} = \infty; \]
\[ \text{for } i=1:n\text{Fact} \]
\[ \quad i\text{Bill} = i\text{Cost}(i,C,PO); \]
\[ \quad \text{if } i\text{Bill} < \text{minBill} \]
\[ \% \quad \text{Found an Improvement} \]
\[ \quad i\text{Best} = i; \text{minBill} = i\text{Bill}; \]
\[ \quad \text{end} \]
\[ \text{end} \]
Inventory Considerations

What if a factory lacks the inventory to fill the purchase order?

Such a factory should be excluded from the find-the-cheapest computation.
### Who Can Fill the Order?

<table>
<thead>
<tr>
<th>Inv:</th>
<th>PO:</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>38</td>
<td>1</td>
<td>0</td>
<td>12</td>
<td>29</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>12</td>
<td>29</td>
<td></td>
<td></td>
</tr>
<tr>
<td>99</td>
<td>12</td>
<td>29</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>34</td>
<td>29</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>42</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Because $12 < 29$
Wanted: A True/False Function

B is “true” if factory i can fill the order.
B is “false” if factory i cannot fill the order.
Boolean Operations in Matlab

SO FAR we have indicated that expressions like

\[ a \leq x \quad \land \quad x \leq b \]

\[ \text{abs}(y) > 10 \]

are either TRUE or FALSE.
In reality, expressions like

\[ a \leq x \quad \&\& \quad x \leq b \]

\[ \text{abs}(y) > 10 \]

render the value “1” if they are TRUE and “0” if they are FALSE.
Example

>> x = 8; y = 7;

>> B = x<y
B =
  0

>> B = x>y
B =
  1
function B = Overlap(a,b,c,d)
% B is true if intervals [a,b]
% and [c,d] intersect.
% Otherwise B is false.
% Assume a<b and c<d.
abToLeft = b < c;
abToRight = d < a;
B = ~(abToLeft || abToRight);
Using Overlap

\[ S = 0; \]
\[ \text{for } k=1:100 \]
\[ \quad a = \text{rand}; \quad b = a + \text{rand}; \]
\[ \quad c = \text{rand}; \quad d = c + \text{rand}; \]
\[ \quad \text{if Overlap}(a,b,c,d) \]
\[ \quad \quad s = s+1; \]
\[ \quad \text{end} \]
\[ \text{end} \]
\[ \text{probOverlap} = s/100 \]
## Back to Inventory Problem

<table>
<thead>
<tr>
<th>Inv:</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>82</td>
<td>19</td>
<td>83</td>
<td>12</td>
<td>42</td>
<td></td>
</tr>
<tr>
<td>51</td>
<td>29</td>
<td>21</td>
<td>56</td>
<td>87</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PO:</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>12</td>
<td>29</td>
<td>5</td>
<td></td>
</tr>
</tbody>
</table>
# Initialization

<table>
<thead>
<tr>
<th>Inv:</th>
<th>38</th>
<th>5</th>
<th>99</th>
<th>34</th>
<th>42</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>82</td>
<td>19</td>
<td>83</td>
<td>12</td>
<td>42</td>
</tr>
<tr>
<td></td>
<td>51</td>
<td>29</td>
<td>21</td>
<td>56</td>
<td>87</td>
</tr>
<tr>
<td>PO:</td>
<td>1</td>
<td>0</td>
<td>12</td>
<td>29</td>
<td>5</td>
</tr>
</tbody>
</table>
Still True...

\[
\begin{array}{c|c|c|c|c|c}
\text{Inv:} & 38 & 5 & 99 & 34 & 42 \\
\hline
   82 & 19 & 83 & 12 & 42 \\
   51 & 29 & 21 & 56 & 87 \\
\end{array}
\]

\[
\begin{array}{c|c|c|c|c|c}
\text{PO:} & 1 & 0 & 12 & 29 & 5 \\
\end{array}
\]

\[
B = B \land ( \text{Inv}(2,1) \geq \text{PO}(1) )
\]
Still True...

\[
\begin{array}{cccccc}
38 & 5 & 99 & 34 & 42 \\
82 & 19 & 83 & 12 & 42 \\
51 & 29 & 21 & 56 & 87 \\
\end{array}
\]

**Inv:**

\[
\begin{array}{cccccc}
38 & 5 & 99 & 34 & 42 \\
82 & 19 & 83 & 12 & 42 \\
51 & 29 & 21 & 56 & 87 \\
\end{array}
\]

**PO:**

\[
\begin{array}{cccccc}
1 & 0 & 12 & 29 & 5 \\
\end{array}
\]

\[
B = B \&\& \left( \text{Inv}(2,2) \geq \text{PO}(2) \right)
\]
Still True...

$$B = B \land ( \text{Inv}(2,3) \geq \text{PO}(3) )$$
No Longer True...

\[ B = B \land (\text{Inv}(2,4) \geq \text{PO}(4)) \]
function B = iCanDo(i,Inv,PO)
% B is true if factory i can fill
% the purchase order. Otherwise, false
nProd = length(PO);
j = 1;
B = 1;
while j<=nProd && B
    B = B && ( Inv(i,j) >= PO(j) );
    j = j+1;
end
iBest = 0; minBill = inf;
for i=1:nFact
    \[iBill = iCost(i,C,PO)\];
    if iBill < minBill
        Found an Improvement
        iBest = i; minBill = iBill;
    end
end

Don't bother with this unless sufficient inventory.
Back To Finding the Cheapest

```matlab
iBest = 0; minBill = inf;
for i=1:nFact
    if iCanDo(i,Inv,PO)
        iBill = iCost(i,C,PO);
        if iBill < minBill
            % Found an Improvement
            iBest = i; minBill = iBill;
        end
    end
end
end
```
function [iBest, minBill] = Cheapest(C, Inv, PO)
    [nFact, nProd] = size(C);
iBest = 0; minBill = inf;
for i = 1:nFact
    if iCanDo(i, Inv, PO)
        iBill = iCost(i, C, PO);
        if iBill < minBill
            iBest = i; minBill = iBill;
        end
    end
end
end
# Finding the Cheapest

As computed by iCost

<table>
<thead>
<tr>
<th>C:</th>
<th>PO:</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>1</td>
</tr>
<tr>
<td>36</td>
<td>0</td>
</tr>
<tr>
<td>22</td>
<td>12</td>
</tr>
<tr>
<td>15</td>
<td>29</td>
</tr>
<tr>
<td>62</td>
<td>5</td>
</tr>
<tr>
<td>12</td>
<td>16</td>
</tr>
<tr>
<td>35</td>
<td>21</td>
</tr>
<tr>
<td>20</td>
<td>12</td>
</tr>
<tr>
<td>12</td>
<td>66</td>
</tr>
<tr>
<td>66</td>
<td>59</td>
</tr>
</tbody>
</table>

1019: Yes

930: No

1040: Yes

As computed by iCanDo
New Problem

Visualizing a function of the form

$$z = f(x, y).$$

Think of $z$ as an elevation which depends on the coordinates $x$ and $y$ of the location.
Sample Elevation Function

function z = Elev(x,y)

r1 =   (x-1)^2   +  3*(y-1.5)^2;

r2 = 2*(x+2)^2   +   (y-.5)^2;

r3 =   (x-.5)^2  + 7*y^2;

z = 100*exp(-.5*r1) + ...
    90*exp(-.3*r2) + ...
    80*exp(-.4*r3);

Three Hills at (1,1.5),(-2,.5), (.5,0)
Its Contour Plot
Making a Contour Plot

```matlab
x = linspace(-5,4,200);
y = linspace(-2.5,6.5,200);
A = zeros(200,200);
for i = 1:200
    for j = 1:200
        A(i,j) = Elev(x(j),y(i));
    end
end
contour(x,y,A,15)
```

Set up a matrix of function evals
function A = SetUp(f,xVals,yVals)
Nx = length(xVals);
Ny = length(yVals);
A = zeros(Ny,Nx);
for i=1:Ny
    for j=1:Nx
        A(i,j) = f(xVals(j),yVals(i));
    end
end
Calling SetUp

\[
x = \text{linspace}(-5,4,200);
\]
\[
y = \text{linspace}(-2.5,6.5,200);
\]
\[
F = \text{SetUp}(@\text{Elev},x,y);
\]

Not just ‘Elev’
The @ is required for function parameters.
Generating a Cross Section

Enter endpoints via ginput
Sample Elev(x,y) along the line segment
Mouse Input Via ginput

To draw a line segment connecting (a(1),b(1)) and (a(2),b(2)):

\[ [a, b] = \text{ginput}(2); \]
\[ \text{plot}(a, b) \]

\[ [a, b] = \text{ginput}(n) \] puts the mouseclick coords in length-\( n \) arrays \( a \) and \( b \).
n = 100;
t = linspace(0,1,n);
x = linspace(a(1),a(2),n);
y = linspace(b(1),b(2),n);
for i=1:n
    % At "time" t(i) we are at (x(i),y(i)).
    % Compute elevation at time t(i).
    f(i) = Elev(x(i),y(i));
end

figure
plot(t,f)