## Sand Pit Utilization

A construction company obtains sand, fine gravel, and coarse gravel from three different sand pits. The pits have different average compositions for the three types of raw materials as shown in Table 1. Historical data for operation of the three sand pits have been processed to obtain the monthly total cubic yards of material extracted from each pit averaged over the last ten years. These data are summarized in Table 2. As an engineer working for the construction company, you are asked to process the data in Tables 1 and 2 to obtain some further quantitative and qualitative information concerning the usage of the three sand pits.

**Table 1 Composition of sand, fine gravel, and coarse gravel in three different sand pits.**

| | Sand (%) | Fine Gravel (%) | Coarse Gravel (%) |
|---|---|---|---|
| **Pit #1** | 55 | 30 | 15 |
| **Pit #2** | 20 | 50 | 30 |
| **Pit #3** | 25 | 20 | 55 |

**Table 2 Monthly volume of material mined from each pit (thousands of cubic yards).**

| | Jan | Feb | Mar | Apr | May | June | July | Aug | Sept | Oct | Nov | Dec |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Pit #1** | 10 | 12 | 15 | 25 | 28 | 36 | 39 | 42 | 43 | 36 | 26 | 14 |
| **Pit #2** | 16 | 19 | 23 | 31 | 35 | 45 | 49 | 55 | 55 | 46 | 28 | 18 |
| **Pit #3** | 8 | 10 | 12 | 22 | 27 | 40 | 45 | 52 | 48 | 45 | 25 | 10 |

We will use Matlab to help in this task. This simple demo illustrates how to obtain different kinds of information from data stored within a couple of arrays. In this case, the data from the two tables are entered into Matlab arrays and several manipulations are performed to obtain the desired information.

The first step is to create the desired arrays from the tabular data, being careful to define precisely the data contained in the arrays. In this case, we will use two matrices, **C** and **V**, as follows:

C(i,j) = fractional composition of component j in pit i.

V(i,k) = cubic yards of material mined from pit i in month k.

where the discrete i, j, and k indices refer to pit i, sand component j, and month k, respectively. These data arrays can be created in Matlab with the follows statements:

```
>> C = [55 30 15; 20 50 30; 25 20 55]/100     % composition information
C =
    0.5500    0.3000    0.1500
    0.2000    0.5000    0.3000
    0.2500    0.2000    0.5500
>> V = [10 12 15 25 28 36 39 42 43 36 26 14;   % volume information
        16 19 23 31 35 45 49 55 55 46 28 18;
         8 10 12 22 27 40 45 52 48 45 25 10]*1000
V =
  Columns 1 through 10
   10000    12000    15000    25000    28000    36000    39000    42000    43000    36000
   16000    19000    23000    31000    35000    45000    49000    55000    55000    46000
    8000    10000    12000    22000    27000    40000    45000    52000    48000    45000
  Columns 11 through 12
   26000    14000
   28000    18000
   25000    10000
```

With these data available, we can now perform the following tasks:

1. Determine the yearly amount of material extracted from Pit #3.

This is given mathematically as

$$\text{yearly amount from Pit \#3} = \sum_{k=1}^{12} V_{3k}$$

and, to actually do this operation in Matlab, we simply need to sum the values in the 3rd row of the **V** matrix, or

```
>> sum(V(3,:))
ans =
      344000
```

2. Find the maximum and minimum amounts of fine gravel produced from all three pits and the month of the year where these occur.

Since we are interested in fine gravel, we will be working with column 2 of the **C** matrix. However, if we transpose this, it becomes a 1x3 vector. Now, a 1x3 matrix giving the fraction of fine gravel by pit times a 3x12 matrix containing the total volume extracted by pit and month will yield a 1x12 array containing the total fine gravel produced by month, or

```
>> A = C(:,2)'*V
A =
  Columns 1 through 10
   12600    15100    18400    27400    31300    41300    45200    50500    50000    42800
  Columns 11 through 12
   26800    15200
```

This operation is represented mathematically as

$$\text{amount of fine gravel by month} = \sum_{i=1}^{3} C_{i2} V_{ik}$$

And now, Matlab's ***min*** and ***max*** commands can be used to find the desired information,

```
>> [Amax,Imax] = max(A)
Amax =
       50500
Imax =
     8

>> [Amin,Imin] = min(A)
Amin =
       12600
```

```
Imin =
     1
```

Thus, the maximum monthly amount of fine gravel is about 50500 cubic yards and this occurs in the 8th month (August). Similarly, the minimum occurs in January with only about 12600 cubic yards of fine gravel produced. Of course, these data can be verified from the original tables…

3. Find the total amount of sand used in June, July, and August.

For this problem, we use the same logic as above, but use column 1 of the **C** matrix, since we are interested in the amount of sand. In addition, since we are only interested in the summer months, we only need to use a portion of the **V** matrix (months 6, 7, and 8). Now, doing the appropriate matrix multiplication directly within the ***sum*** command gives the desired result, or

```
>> sum(C(:,1)'*V(:,6:8))
ans =
      128400
```

This operation can be represented in discrete form as

$$\text{total amount of sand used during the summer} = \sum_{k=6}^{8}\left(\sum_{i=1}^{3} C_{i1} V_{ik}\right)$$

In addition to the above quantitative results, you were also asked to produce some qualitative plots that illustrate utilization of the three sand pits. Creating plots is definitely easier to do within a Matlab script file, so we will abandon the use of the interactive command window and simply write a script file to do the remainder of the tasks. In fact, all the above commands were also put within the **sand_pits.m** file, as shown in Table 3, so that we have a complete record of the analysis. Running the program produces the following quantitative results (which match our interactive analysis from above):

```
>> sand_pits
  Total amount of material from pit #3 (cubic yards):   344000
  Max amount of fine gravel produced was    50500 cubic yards in Aug
  Min amount of fine gravel produced was    12600 cubic yards in Jan
  Amount of sand used in summer (cubic yards):          128400
```

**Table 3 Program listing for sand_pits.m.**

```
%
%   SAND_PITS.M    Obtain various data for a construction company
%                       using array and matrix operations
%
%   This demo illustrates how to obtain several types of information from
%   data stored within a couple of arrays.  In this case, the data from two
%   tables are entered into Matlab arrays and several manipulations are performed
%   to obtain the desired information.
%     Table 1:  Composition of sand, fine gravel, and coarse gravel
%                       in three different sand pits
%        C(i,j) = composition of component j in pit i
%
%     Table 2:  Monthly volume of material mined from each pit
%        V(i,k) = cubic yards of material mined from pit i in month k
%
%   Note:  There are three components:
%          1 -> sand,   2 -> fine gravel,  and   3 -> coarse gravel
%
%   File produced by Prof. J. R. White, UMass-Lowell (last update: September 2017)
```

```
%

      clear all, close all, nfig = 0;
%
%   enter data from Tables 1 and 2
      C = [55 30 15; 20 50 30; 25 20 55]/100;     % composition information
      V = [10 12 15 25 28 36 39 42 43 36 26 14;   % volume information
           16 19 23 31 35 45 49 55 55 46 28 18;
            8 10 12 22 27 40 45 52 48 45 25 10]*1000;
%
%   1. total yearly amount of material from pit number 3
      fprintf(1,'  Total amount of material from pit #3 (cubic yards): %8.0f \n', ...
              sum(V(3,:)));
%
%   2. max and min amount of fine gravel produced and corresponding months
      A = C(:,2)'*V;           % gives vector by month of total fine gravel
      [Amax,Imax] = max(A);   [Amin,Imin] = min(A);
      months1 = ['Jan'; 'Feb'; 'Mar'; 'Apr'; 'May'; 'Jun';
                 'Jul'; 'Aug'; 'Sep'; 'Oct'; 'Nov'; 'Dec'];
      fprintf(1,'  Max amount of fine gravel produced was %8.0f cubic yards in %s \n', ...
                Amax,months1(Imax,:));
      fprintf(1,'  Min amount of fine gravel produced was %8.0f cubic yards in %s \n', ...
                Amin,months1(Imin,:));
%
%   3. total amount of sand used during June, July, and August
      fprintf(1,'  Amount of sand used in summer (cubic yards):        %8.0f \n', ...
              sum(C(:,1)'*V(:,6:8)));
%
%   4. plot of total volume for each pit by month
      nfig = nfig+1;  figure(nfig)
      bar(V'),grid
      title('Sand\_Pits:  Total Volume by Month for Each Pit')
      xlabel('Month of Year'), ylabel('Volume (cubic yards)')
      legend('Pit #1','Pit #2','Pit #3')
%
%   5. plot of total sand, fine gravel, and coarse gravel by month
      nfig = nfig+1;  figure(nfig)
      bar(V'*C),grid
      title('Sand\_Pits:  Total Volume by Month and Sand Type')
      xlabel('Month of Year'), ylabel('Volume (cubic yards)')
      legend('Sand','Fine Gravel','Coarse Gravel')
%
%   6. create pie chart showing the fractional total volume by pit
      totvol = sum(sum(V));
      nfig = nfig+1;  figure(nfig)
      pie(sum(V'),{'Pit #1' 'Pit #2' 'Pit #3'})
      title(['Sand\_Pits:  Fractional Usage of Each Sand Pit  (total ', ...
             num2str(totvol,'%10.2e'),' cubic yards)'])
%
%   7. create pie chart showing the fractional total volume by month
      months2 = {'Jan' 'Feb' 'Mar' 'Apr' 'May' 'June' ...
                 'July' 'Aug' 'Sept' 'Oct' 'Nov' 'Dec'};
      nfig = nfig+1;  figure(nfig)
      pie(sum(V),months2)
      title(['Sand\_Pits:  Fractional Usage by Month  (total ', ...
             num2str(totvol,'%10.2e'),' cubic yards)'])
%
%   end of program
```

As seen in Table 3, four different plots, using Matlab's ***bar*** and ***pie*** commands, are generated to show some general information about the utilization of the three sand pits. All four plots are included below in Fig. 1 and, collectively, they give a good picture of the sand pit usage by this particular construction company. You should type ***help bar*** and ***help pie*** to see how these graphing commands work…

Some interesting features in **sand_pits.m** that should be noted are identified as follows:

1. Notice that the transpose operator (the single quote) is used quite extensively in this file. This is often the easiest way to manipulate the arrays to give the desired result. However, to use it properly, you really need to understand the data contained in the arrays!!!

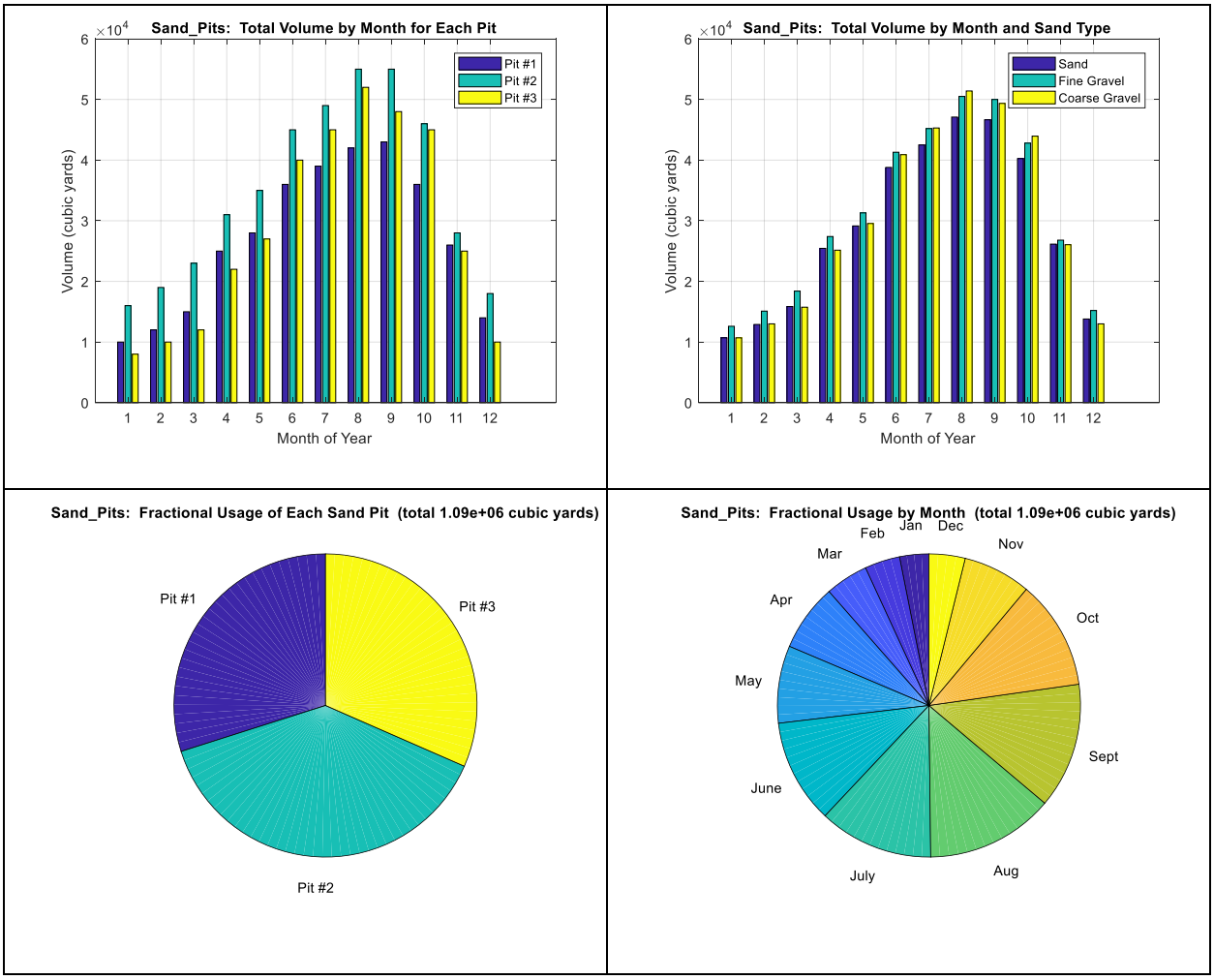


**Fig.1 Summary information concerning sand pit utilization.**

2. Note the different treatments for the **months1** and **months2** character arrays. The variable **months1** is a regular 12x3 array with each character occupying one element of the matrix. For this case, the abbreviation for each month must have the same number of characters so that each row has the same number of elements. Also note that when we used this variable to edit the month of interest, all the columns for a given row were used to get the month's name -- such as **months1(Imax,:)**. Now, the use of the **months2** cell array is completely different. A cell array is formed with the curly brackets ({ } instead of [ ]), and each string within single quotes makes up a single element. The array is a 1x12 row vector, where each element is a string that can have differing sizes. The ***pie*** command requires a cell array for its labels, so the format of **months2** was required here. The two forms, **months1** and **months2**, were used here to illustrate the differences, but the cell format for working with

character variables is often the best choice, since the string lengths do not have to be the same. The ***whos*** command after running **sand_pits.m** shows that **months1** and **months2** have a completely different structure, so clearly, they must be used appropriately:

```
>> whos
  Name          Size            Bytes  Class     Attributes

  A             1x12               96  double
  Amax          1x1                 8  double
  Amin          1x1                 8  double
  C             3x3                72  double
  Imax          1x1                 8  double
  Imin          1x1                 8  double
  V             3x12              288  double
  months1      12x3                72  char
  months2       1x12              1422  cell
  nfig          1x1                 8  double
  totvol        1x1                 8  double
```

Well, this completes our current relatively simple example. The goal here was to illustrate how to use array indexing and various matrix operations (matrix multiplication and the transpose operation) to assist in the analysis of data stored in a couple of 2-D arrays. As part of the example, we also used Matlab's ***sum***, ***max***, and ***min*** commands as well as the ***bar*** and ***pie*** plotting functions. And we even got a brief introduction to the difference between character and cell arrays! Hopefully this example will add to your developing familiarity with Matlab, and help you use it in a variety of creative ways in solving even more complex problems…