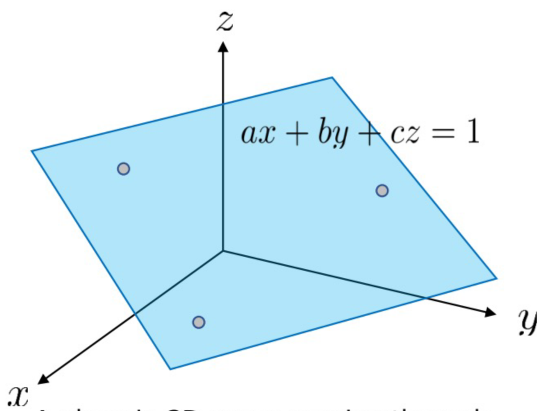
# Report 3

- Suppose we have three points in 3D space and their coordinates are $(x,y,z)$=(0.2+$r_{x1}$, -0.1+$r_{y1}$, 1.0+$r_{z1}$), (3.0+$r_{x2}$, 0.1+$r_{y2}$, -1.0+$r_{z2}$), and (1.0+$r_{x3}$, -2.0+$r_{y3}$, -0.5+$r_{z3}$), respectively. $r$ is a random number between -0.1 and 0.1. Find a plane passing through these three points. Note that the equation of a plane that does not pass through the origin (0,0,0) is given by

$$ax + by + cz = 1$$




A plane in 3D space passing through three points and not through the origin

Hint : Set up simultaneous linear equations and solve it to determine unknowns (a,b,c)

*Note: the problem was later modified so that the equation we should use to find the plane is* $ax + by + cz = (studentnumber)$*. My student number is C2TB1702, so I will be using 1702.*

## My Line of Thinking

We can break down the general plane equation $ax + by + cz = 1702$ and rewrite it as a matrix multiplication.

$$\begin{bmatrix} x & y & z \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} 1702 \\ 1702 \\ 1702 \end{bmatrix}$$

replacing x, y and z by values corresponding to the coordinates of the three points:

$$\begin{bmatrix} 0.2 + r_{x1} & 0.1 + r_{y1} & 1.0 + r_{z1} \\ 3.0 + r_{x2} & 0.1 + r_{y2} & -1.0 + r_{z2} \\ 1.0 + r_{x3} & -2.0 + r_{y3} & -0.5 + r_{z3} \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} 1702 \\ 1702 \\ 1702 \end{bmatrix}$$

as all random values are known, there are really only 3 unknowns in this system of linear equations, and since we have 3 equations, it is possible to get their values using Gaussian Elimination.

## Carrying over the math to Octave

For the purposes of the script, let's denote the matrix containing $xyz$ coordinates of the three points as $P$, the matrix containing $a, b$ and $c$ values as $Q$, and the matrix containing student numbers as capital $R$. I will use lowercase $r$ for the matrix containing random values.

$$PQ = R$$

The function `rand()` only generates values between 0 and 1, so in order to get values between -0.1 and 0.1, we need to modify the original function.

We can specify the size of the interval we want by multiplying the `rand()` function with the interval length we want. For example, if we want the length to be 5, we can do `5*rand()` and this will multiply the generated random value by 5, which means our original interval length of 1 is now multiplied to 5.

We can also specify where the interval starts and ends on the number line by adding the minimum point from the `rand()` function multiplied by the interval length.

Putting all of these together, we can conclude an explicit formula for generating random values.

$$r = min + rand(rows, columns) * length$$

and this will generate a matrix `r` that contains random values that exist in the interval $(min, min + length)$.

Conversely, to generate a *matrix* `r` that contains random values between -0.1 and 0.1, we set $min = -0.1$ and $length = 0.2$, and write the code:

```
% First, initialize random numbers
r = -0.1+(0.2)*rand(3,3);
% r is a 3x3 matrix containing random numbers we can use for rx1,rx2 and so on
```

To better visualize r:

$$\begin{bmatrix} r_{x1} & r_{y1} & r_{z1} \\ r_{x2} & r_{y2} & r_{z2} \\ r_{x3} & r_{y3} & r_{z3} \end{bmatrix}$$

Now we can initialize the matrix containing the coordinates of the three points.

```
% Next, initialize the points
P = [0.2,0.1,1.0;3.0,0.1,-1.0;1.0,-2.0,-0.5] + r;
```

To better visualize this line:

$$P = \begin{bmatrix} 0.2 & 0.1 & 1.0 \\ 3.0 & 0.1 & -1.0 \\ 1.0 & -2.0 & -0.5 \end{bmatrix} + \begin{bmatrix} r_{x1} & r_{y1} & r_{z1} \\ r_{x2} & r_{y2} & r_{z2} \\ r_{x3} & r_{y3} & r_{z3} \end{bmatrix}$$

which is equivalent to:

$$P = \begin{bmatrix} 0.2 + r_{x1} & 0.1 + r_{y1} & 1.0 + r_{z1} \\ 3.0 + r_{x2} & 0.1 + r_{y2} & -1.0 + r_{z2} \\ 1.0 + r_{x3} & -2.0 + r_{y3} & -0.5 + r_{z3} \end{bmatrix}$$

with the points initialized, we can then define the matrix containing the student number.

```
% Then we initialize the matrix containing student number
R = [1702;1702;1702];
```

With all the matrices containing known variables initialized, we can then do

```
% Do Gaussian elimination to find unknowns and collect in Q
Q = P\R;
```

Finally, print the results by assigning `Q(1)` to $a$, `Q(2)` to $b$ and `Q(3)` to $c$.

```
printf('a = %f, b = %f, c = %f\n',Q(1), Q(2), Q(3))
```

Output differs from one iteration of the program to the other, so we need to verify the program by checking if the points we are given are on the plane.

# Checking if the plane equation is correct

We can use the dot product to check if the points are on the plane we have. If the values of $a, b$ and $c$ are correct, we should receive our student number back.

```
correctRows = 0; % Initialize number of correct rows as 0
for i = 1:3 % Iterate from row 1 through 3
  isR = round(dot(P(i,:),Q)) % dot row i of P with Q
  if isR == R(i) % Check if the result of the dot is same as student number
    printf('Row %d is correct\n', i)
    correctRows += 1; % If this row is correct, add 1 to # of correct rows
  else
    printf('!! Row %d is INCORRECT !!\n', i)
  endif
endfor
```

First we initialize `correctRows` as 0. As we come across rows that the program deems correct, we add to this variable.

Next, we use a `for` loop to iterate through the three rows of the matrix $P$.

For each row of the matrix, we want to dot it with the $Q$ matrix, and check if the resulting number is consistent with out student number. I name that resulting number `isR`, as we are checking if it's the same as `R(i)`. Note that since `R` contains the same 3 numbers, it's arbitrary whether we use `R(i)` or just 1702 (my student number). However, I believe it is good practice to use the actual number inside the matrix, as we may come across other problems that have differing values in the `R` matrix.

Also note that I used `round()` to round the result of the dot multiplication, as we want to compare it with a round number.

After iterating through all rows of $P$, we can now check if `correctRows` equal the number of rows we have.

```
if correctRows == length(P) % Check if # of correct rows equal # of rows in P
  printf('All rows are correct. The program is working as intended.\n')
else
  printf('There are incorrect rows. Check the program again for bugs.\n')
endif
```

Note that as we are using variables from the first, main script, we have to run that main script firsts before we can run this script to check its validity.

# A Test

Running main script `CAPS_02_C2TB1702_coplanar.m`:

```
P =

   0.141138   0.058446   0.967838
   2.974780   0.027182  -1.097699
   1.079690  -2.003029  -0.555198

a = 1178.543150, b = -665.380413, c = 1626.875525
>>
```

Doing calculations manually to check the validity of the results:

$$\begin{bmatrix} 0.141138 & 0.058446 & 0.967838 \\ 2.974780 & 0.027182 & -1.097699 \\ 1.079690 & -2.003029 & -0.555198 \end{bmatrix} \begin{bmatrix} 1178.543150 \\ -665.380413 \\ 1626.875525 \end{bmatrix}$$
$$= \begin{bmatrix} 0.141138 \times 1178.543150 + 0.058446 \times (-665.380413) + 0.967838 \times 1626.875525 \\ 2.974780 \times 1178.543150 + 0.027182 \times (-665.380413) + (-1.097699) \times 1626.875525 \\ 1.079690 \times 1178.543150 + (-2.003029) \times (-665.380413) + (-0.555198) \times 1626.875525 \end{bmatrix}$$
$$= \begin{bmatrix} 1702.000354 \\ 1702.000612 \\ 1701.999493 \end{bmatrix}$$

The result is also validated by our checking program.

```
isR = 1702
Row 1 is correct
isR = 1702
Row 2 is correct
isR = 1702
Row 3 is correct
All rows are correct. The program is working as intended.
>>
```