To populate D with the correct entries, you will need to form a system of linear algebraic equations to be solved for the interior elements. Since there are 8 elements whose value are unknown, 8 equations are needed. Put these equations into the $Az = b$ form such that z is an 8-by-1 array of the unknown element values and solve the system of linear equations using the backslash operator.

The preceding analysis corresponds to a simplified version of the finite difference (or finite element) method applied to solving a classical partial differential equation called the Laplacian. You may think of the values of D as the steady state distribution of temperature in a rectangular domain for which the boundary temperature is fixed.

3. Vertical and horizontal loads are applied to a simple truss as shown in Figure 1 (Note that this truss is nearly identical to that shown in the lecture slides).

   The geometry of the truss is:

   - members B_1, B_4, B_6 and the vertical wall form a rectangle;
   - $B_1 : 4.0m$
   - $B_4 : 3.0m$
   - $B_5 : 5.0m$
   - $B_6 : 4.0m$
   - angle between B_4 and B_2: 120^o
   - angle between B_4 and B_3: 30^o
   - angle between B_3 and B_2: 30^o

   Suppose you are to calculate the forces in the members of the truss for multiple different loading scenarios. So that you don't have to repeat the calculations for each different loading scenario you will write a function named trussAnalyze with the following declaration line

   begin code
   ```
   function T = trussAnalyze(Px, Pz)
   ```
   end code

   where the input arguments, Px and Pz, are the vertical and horizontal loads applied to the truss as shown in Figure 1. The function should return the force in each member of the truss as a 6-by-1 array where each element is the force in the corresponding member (i.e. T(1) is the force in member $B_1, \ldots$).

   Note: As in the lecture slides Pz is defined to be positive in the downward direction and Px is positive to the left as the arrow in Figure 1 indicates. As in the lecture slides the force into a pin is defined as positive. Hence a member with a positive force is in compression and a member with a negative force is in tension.

   As shown in the video lectures you should form a system of 6 linear equations that contain 6 unknowns. Then using the backslash operator you can solve for the unknown forces in the truss members.

4. Suppose there are a set of n points in the $x-y$ plane. Write a function with the following function declaration line

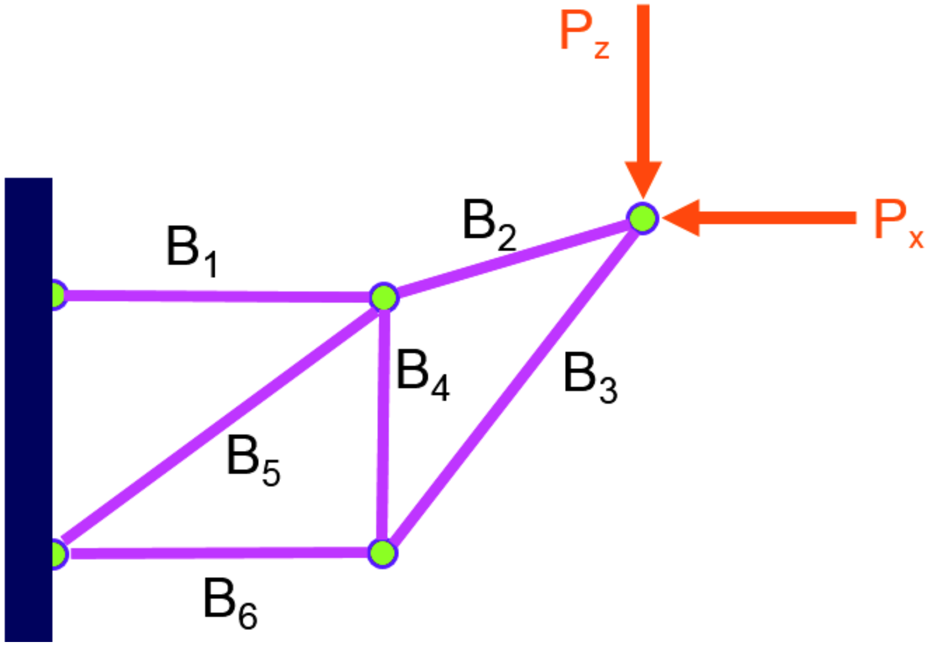

Figure 1: Truss 1 with vertical and horizontal loading

begin code
```
function poly = findpoly(xVal,yVal)
```
end code

where `xVal` and `yVal` are `n-by-1` arrays of the x and y locations of the points. The function `findpoly` finds a $n-1$ degree polynomial whose graph passes through all n of the points and returns a `1-by-n` array of the polynomial coefficients.

For each point, (x_i, y_i), a polynomial of the following form can be created.

$$y_i = p(x_i) = a_{n-1}x_i^{n-1} + a_{n-2}x_i^{n-2} + \cdots + a_1 x_i + a_0$$

where $a_{n-1}, \ldots, a_0$ are the coefficients of the polynomial. We want to find the value of the coefficients so that the polynomial passes through all the points.

In order to do this you should put the problem into $Az = b$ form such that z is an `n-by-1` array of the unknown polynomial coefficients, b is an `n-by-1` array of the values of y, and A is an `n-by-n` matrix that relates b and z through the values of x. Using the backslash operator you can solve for z, the polynomial coefficients. Make sure that the polynomial coefficients returned by this function are a `1-by-n` array and represent the polynomial correctly.

You can check your function by evaluating the polynomial at many different x values using `polyval` and plotting it with the associated data points as shown in the following code. The resulting figure from this code is shown in Figure 2.

begin code
```
xPt = [-2; 1; -4; 5; 2];
yPt = [2; 2.7; -7; 1; 2];

```