Lecture 4

Symbolic Algebra Solver

Last week we solved a mesh circuit problem by setting up a system of equations for each loop in the circuit.

Matlab can assist with the algebra to factor the equations. We first need to declare the variables as symbols using the keyword **syms**

```
syms R1 R2 R3 R4 R5 I1 I2 I3;
currents=[I1 I2 I3];
resistors=[R1 R2 R3 R4 R5];
```

The problem consisted on specifying three equations for the voltage drop around the loops

```
loop1 = R1*I1+R3*(I1-I2)+R2*(I1-I3);
loop2 = R4*I2+R5*(I2-I3)+R3*(I2-I1);
loop3 = R2*(I3-I1)+R5*(I3-I2);
```

We can ask Matlab to refactor the equations above in terms of the currents using **collect**

```
equ1=collect(loop1,currents)
```

```
equ1 = 
```

$$(R_1 + R_2 + R_3)\, I_1 + (-R_3)\, I_2 + (-R_2)\, I_3$$

```
equ2=collect(loop2,currents)
```

```
equ2 = 
```

$$(-R_3)\, I_1 + (R_3 + R_4 + R_5)\, I_2 + (-R_5)\, I_3$$

```
equ3=collect(loop3,currents)
```

```
equ3 = 
```

$$(-R_2)\, I_1 + (-R_5)\, I_2 + (R_2 + R_5)\, I_3$$

From the above we collect the coefficients that multiply the currents using **coeffs** and arrange them into the lhs matrix

```
coeffs1=coeffs(loop1,currents);
coeffs2=coeffs(loop2,currents);
coeffs3=coeffs(loop3,currents);
lhs=[coeffs1;coeffs2;coeffs3]
```

```
lhs = 
```

$$\begin{pmatrix} -R_2 & -R_3 & R_1 + R_2 + R_3 \\ -R_5 & R_3 + R_4 + R_5 & -R_3 \\ R_2 + R_5 & -R_5 & -R_2 \end{pmatrix}$$

Finally, with we **subs**titute the actually values for the resistors and use **double** to convert the symbolic lhs matrix to a numerical type

```
lhs_subs=double(subs(lhs,resistors,[50 150 100 250 300]))
```

```
lhs_subs = 3×3
  -150  -100   300
```

```
  -300   650  -100
   450  -300  -150
```

With this we can proceed as before to solve last weeks mesh loop problem by inverting the matrix

```
V1=24;
inv(lhs_subs)*[0;0;V1]
```

```
ans = 3×1
    0.1361
    0.0772
    0.0938
```

Finally, we **clear** the resister and current variables from the symbolic algebra system

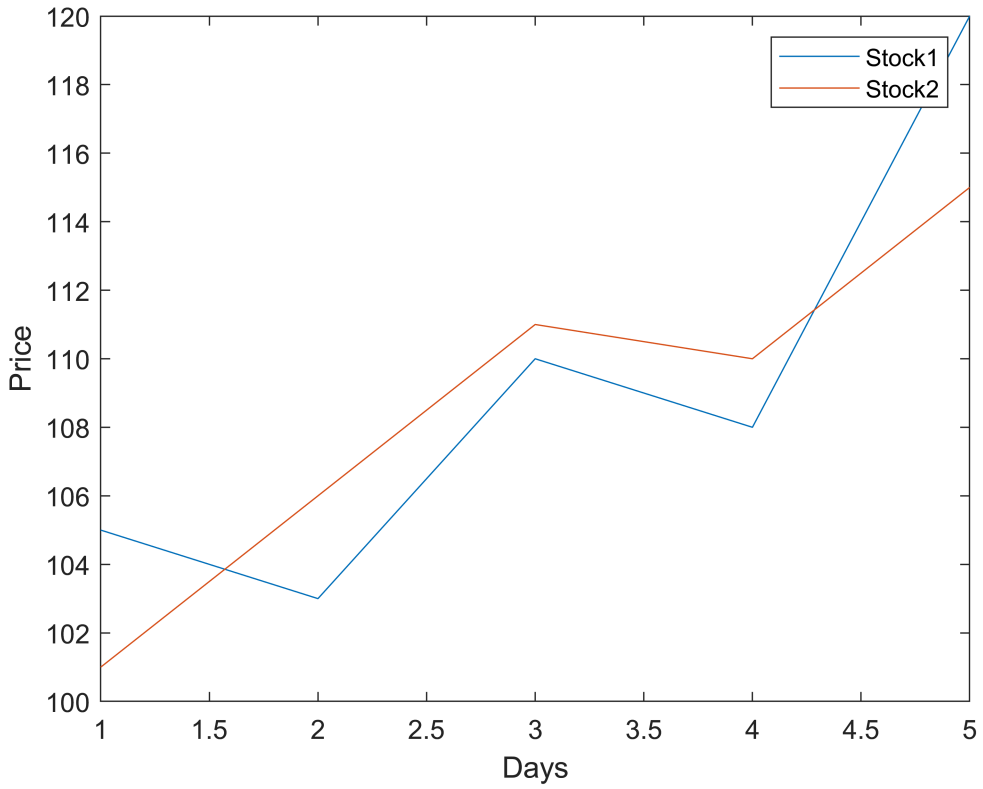
```
clear R1 R2 R3 R4 R5 I1 I2 I3;
```

Principal Component Analysis (PCA)

Using linear algebra concepts as statistical tools to analyse data.

For example, hypothetical end of day closing price data for two different stocks over one week period arranged as columns in the data matrix D

```
D = [105 103 110 108 120;
     101 106 111 110 115]';
plot(D); legend("Stock1", "Stock2");
xlabel("Days");ylabel("Price");
```



For further analysis we center the data by removing the mean and compute the correlation matrix C. Each entry (i,j) of the correlation matrix consists of the product of stock i with stock j. By construction the correlation matrix is real and symmetric.

```
d = D-mean(D,1);
C = d'*d  % correlation matrix
```

```
C = 2×2
  174.8000  117.4000
  117.4000  113.2000
```

Matlab's function **eig** allows to compute the eigenvectors and eigenvalues of the correlation matrix. The eigenvectors are returned as columns of V.

```
[V_,E_]=eig(C)
```

```
V_ = 2×2
    0.6108   -0.7918
   -0.7918   -0.6108
E_ = 2×2
   22.6270         0
         0  265.3730
```

Use sort to order the eigenvalues and vectors

```
[E__,ind] = sort(diag(E_),"descend");
E=E_(ind,ind)
```

```
E = 2×2
  265.3730         0
```

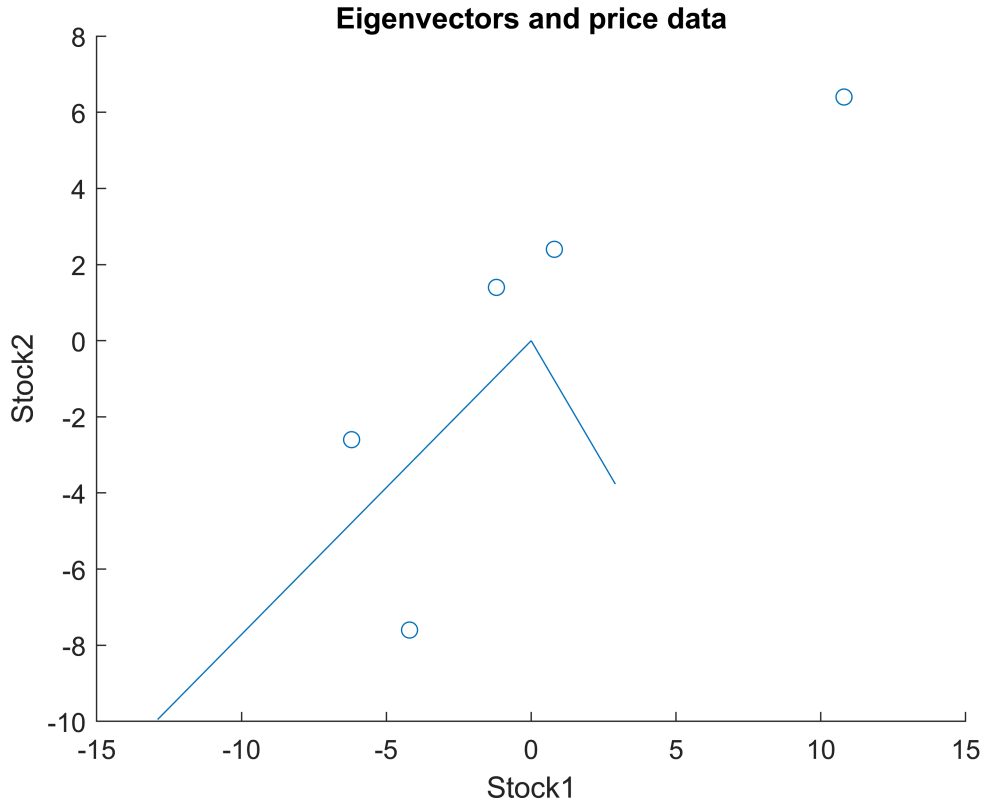
```
         0   22.6270
```

```
V=V_(:,ind)
```

```
V = 2×2
   -0.7918    0.6108
   -0.6108   -0.7918
```

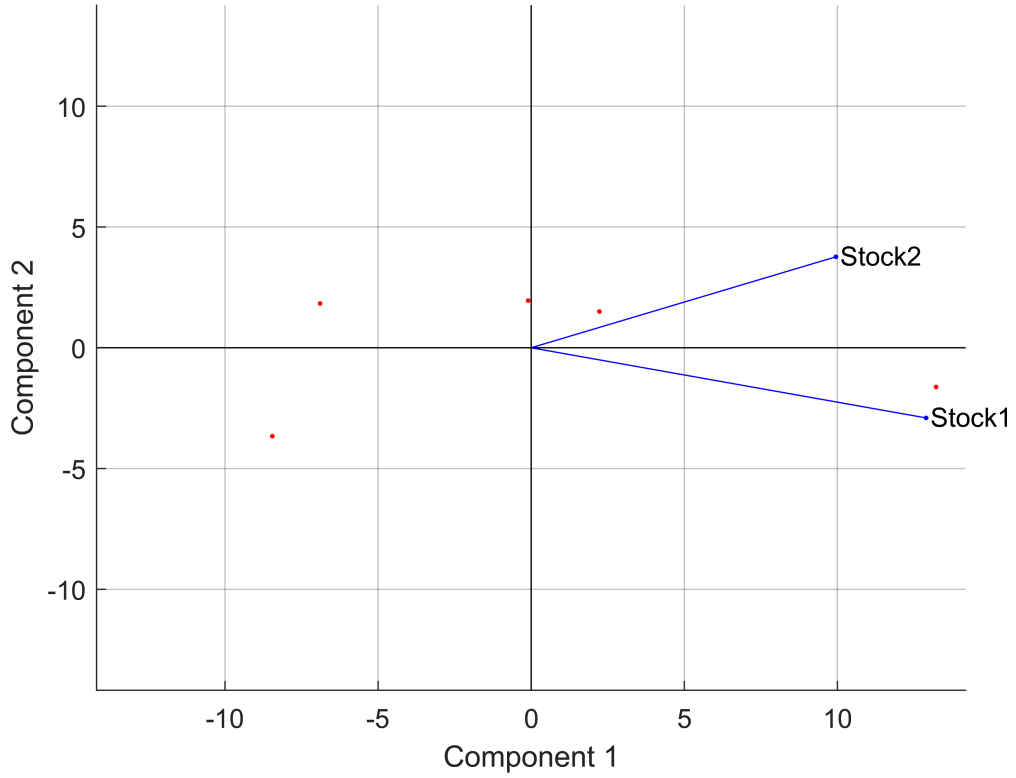
Visiualizing the data points as scatter plot shows how the data is dispered according the directions of the eigenvectors according to the magnitude of the eigenvalues (variance along each direction).

```
EV=(V*E.^.5);  % scale eigenvectors by standard derivation
scatter(d(:,1),d(:,2));
hold on;
xlabel("Stock1");ylabel("Stock2");
title("Eigenvectors and price data");
line([0 EV(1,1)],[0 EV(2,1)]);
line([0 EV(1,2)],[0 EV(2,2)]);
hold off;EV
```




```
EV = 2×2
  -12.8980    2.9056
   -9.9507   -3.7662
```

```
biplot(EV,'Scores',d*V,'Varlabels',["Stock1","Stock2"]);
```




```
%axis([-.26 0.6 -.51 .51]);
```

Projecting the data matrix d onto the eigenvector directions V gives a score matrix U. It shows the contribution of the principal eigenvector directions to each data point

```
U=d*V  % score
```

```
U = 5×2
    7.9677    3.4519
    6.4971   -1.7286
   -2.0994   -1.4116
    0.0949   -1.8415
  -12.4603    1.5298
```

The variation of the score matrix rows show how much of the signal is reflected in each direction of the eigenvectors

```
var(U)
```

```
ans = 1×2
   66.3432    5.6568
```

The build in function PCA gives the same results (up to signs) as the manual calculations above

```
[coeff,score,latent] = pca(d);

coeff  % eigenvectors
```

```
coeff = 2×2
    0.7918   -0.6108
```

```
    0.6108    0.7918
```

```
score  % projections of data on the EV
```

```
score = 5×2
   -7.9677   -3.4519
   -6.4971    1.7286
    2.0994    1.4116
   -0.0949    1.8415
   12.4603   -1.5298
```

```
latent % variance of the projected data
```

```
latent = 2×1
   66.3432
    5.6568
```

Each dimension of the projected space represents different aspects of the data. We can visualize this by zeroing out individual dimensions and projecting back to the data space

```
U1 = [U(:,1)';linspace(0,0,5)]
```

```
U1 = 2×5
    7.9677    6.4971   -2.0994    0.0949  -12.4603
         0         0         0         0         0
```

```
d1= V*U1
```

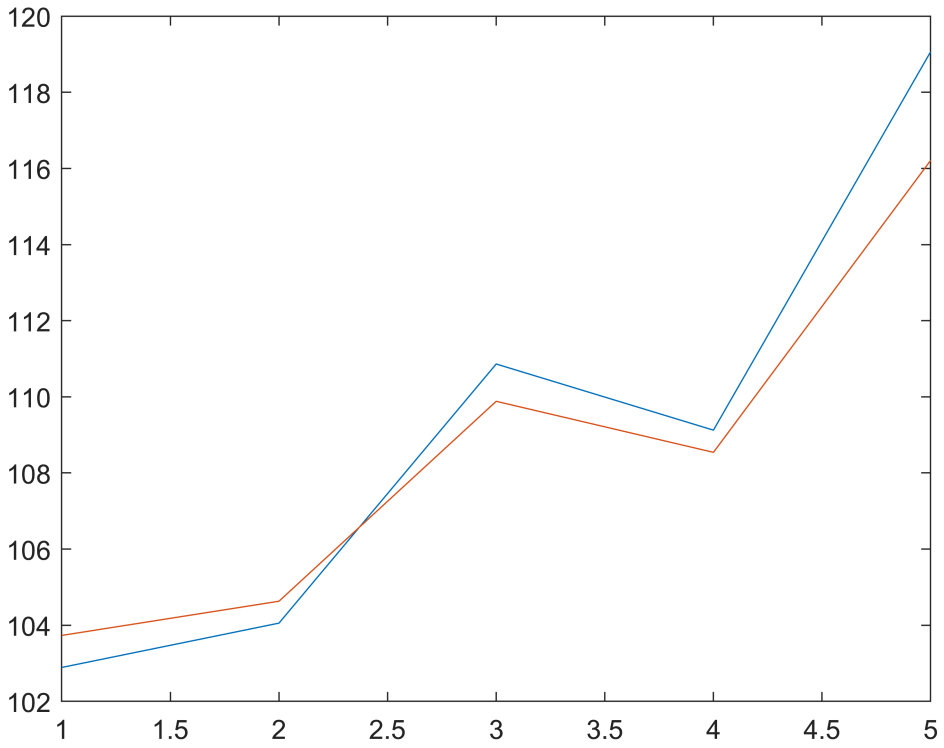
```
d1 = 2×5
   -6.3085   -5.1441    1.6622   -0.0752    9.8656
   -4.8670   -3.9686    1.2824   -0.0580    7.6112
```

```
D1=d1+mean(D)'
```

```
D1 = 2×5
  102.8915  104.0559  110.8622  109.1248  119.0656
  103.7330  104.6314  109.8824  108.5420  116.2112
```

The first dimension represents the dominant paralell move.

```
plot(D1')
```




```
U2 = [linspace(0,0,5); U(:,2)']
```

```
U2 = 2×5
         0         0         0         0         0
    3.4519   -1.7286   -1.4116   -1.8415    1.5298
```

```
d2=V*U2
```

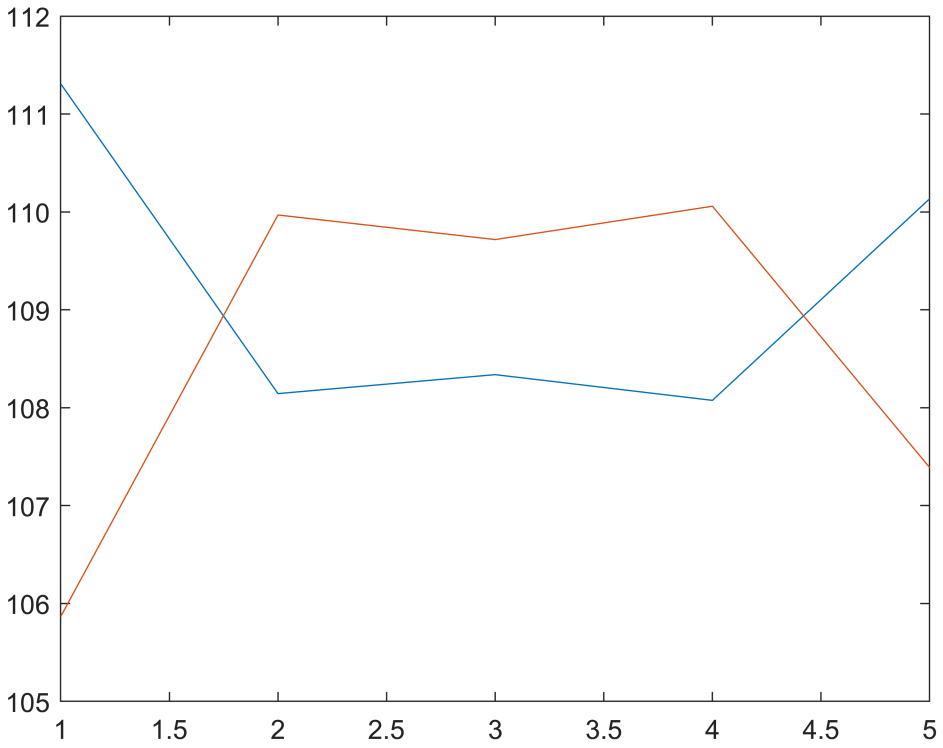
```
d2 = 2×5
    2.1085   -1.0559   -0.8622   -1.1248    0.9344
   -2.7330    1.3686    1.1176    1.4580   -1.2112
```

```
D2=d2+mean(D)'
```

```
D2 = 2×5
  111.3085  108.1441  108.3378  108.0752  110.1344
  105.8670  109.9686  109.7176  110.0580  107.3888
```

The second dimension represents the opposing moves

```
plot(D2')
```



One can use PCA as a compression or filtering method by using only a the factors with the highest importants (variances) to store and reconstruct the data.

Application: Simple Recommendation Engine

Given Matrix of Users and Movie Ratings

```
D = [1     5    NaN;
     2     4    5  ;
     1     5    4  ;
     NaN   4    5  ;
     4     2    1  ;
     5     1    2  ;
     4     2   NaN ;
     5     NaN   2];


 [coeff1,score1,latent1]= pca(D,'algorithm','als','centered','off');
 coeff1
```

```
coeff1 = 3×3
    0.4519    0.8896   -0.0668
    0.6019   -0.2488    0.7588
    0.6584   -0.3831   -0.6479
```

```
score1
```

```
score1 = 8×3
    7.7139   -2.8291   -0.4574
    6.6034   -1.1318   -0.3378
    6.0950   -1.8870    1.1357
    6.4825   -1.3697   -0.3200
    3.6699    2.6775    0.6024
    4.1783    3.4327   -0.8711
    4.0607    2.4501    0.2179
    5.0958    3.0534    0.2856
```

```
 latent1
```

```
latent1 = 3×1
   31.9581
    6.1250
    0.3708
```

Projecting the scores via the coefficients (eigenvector matrix) back to the ratings show us the predictions for previously unknown NaN ratings

```
 score1*coeff1'
```

```
ans = 8×3
    1.0000    5.0000    6.4589
    2.0000    4.0000    5.0000
    1.0000    5.0000    4.0000
    1.7325    4.0000    5.0000
    4.0000    2.0000    1.0000
    5.0000    1.0000    2.0000
    4.0000    2.0000    1.5935
    5.0000    2.5244    2.0000
```