CHE501 Computer lab:
Least-squares fitting

A. *Standard example*
B. *Brief explaination*
C. *Exercises: Modifications to be performed*
D. *Solutions*

Fitting: A. *Standard example 1*

You have the data

% COPY START

```
x=[0   1   2   3   4    5   6   8   9   10]';
y=2*[0.5,1.4,1.2,0.4,0.4,-0.3,0.2,1.1,0.1,0.6]';
```

% The following code fits a quartic polynomial to the data:

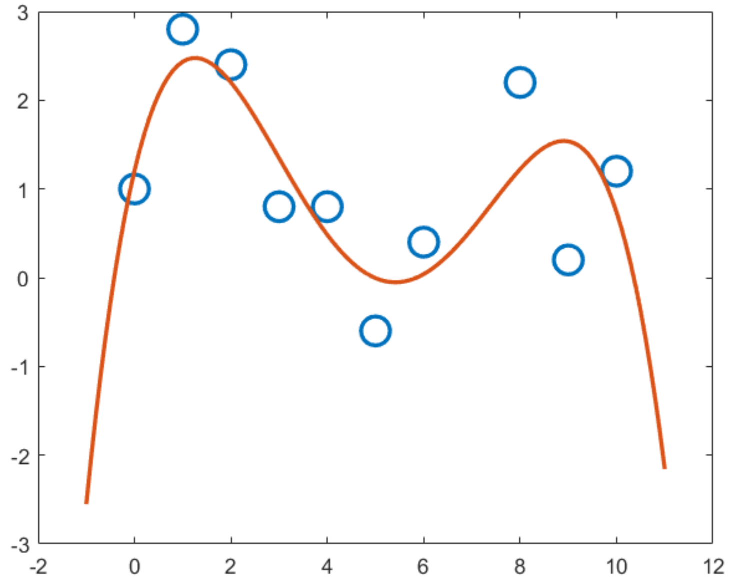
```
pexp=0:4;     % powers
dm=x.^pexp;   % construct the designmatrix
p=dm\y;       % solve y=p0+p1*x+…; find the parameters
```

%Plot the datapoints and the polynomial :

```
xx=((min(x)-1):0.1:(max(x)+1))';% smooth x-points
yy=xx.^pexp*p;                  % corresponding y
plot(x,y,'o',xx,yy,'-', ...
'linewidth',2,'markersize',14);
```

% COPY END

You should get a picture like the one to the right:



B. Brief explanation:

- x and y are the data points, two column vectors.
- pexp is a vector with the powers of the polynomial, here [0 1 2 3 4]. We fit to a polynomial of the form

  $\mathbf{y_{fit} = p_0+p_1*x+p_2*x^2+p_3*x^3+p_4*x^4d}$

  Fitting means that we want to find the p values that give the best y values in this formula. 'Best' means here that we minimize the standard deviation $\mathbf{\Delta=\Sigma(y-y_{fit})^2}$ (Other criteria are also possible but we use this one)
- Linear fitting is always done by constructing a design matrix (*https://en.wikipedia.org/wiki/Design_matrix*). The design matrix **dm** contains has the data points in the lines and the terms for which p is calculated in the columns.
- For obtaining p we have to solve the linear matrix equation **y=p*dm** with respect to **p**. If dm is a square matrix, $\mathbf{p=y*dm^{-1}}$. This is hardly the case, though. In the general case, the so-called pseudoinverse of **dm** is a generalization of $\mathbf{dm^{-1}}$. We ignore the details, and and just note that "dm\" (which is "/dm", but reversed) performs the operation. We have now **p**. , the best polynomial coefficients.
- For plotting the correspondence between the function and the data points we calculate yy from y_{fit}=p*dm but now with dm build from regularly spaced x values (xx) so that we get a smooth function. This is the red line.

Fitting: C. *exercises*

Could you reproduce the graph ? What are the values of p ?
Now you should modify the code according to the 5 points below. The necessary modifications are quite small.

1. What happens if the polynomial has only the powers 0,2,4 ? Do you get an acceptable fit ?
2. What happens if the polynomial has instead the powers 0 to 10 ? Do you get a good fit ? What is the problem ?
3. Can you also fit with polynomial powers that are not integers ? For example: 0,0.5,1 … 4 ?
   Is there a problem ?
4. Can you also fit with sin functions instead of powers ? For example: $y=p_1*\sin(x)+p_2*\sin(2x)+\ldots p_n*\sin(5x)$?
   Is there a problem ?
5. You have seen that the sin-terms do not reach the points ! Can you repair that ? Try it by allowing a shift :
   $y=p_1+p_2*\sin(x)+p_3*\sin(2x)+\ldots\ p_n*\sin(5x)$?

You can modify the code and directly copy your new code into the Octave window.

D. Solutions: Fitting *exercises 1-5*

- *To be shown at the end id problems arose or during discussions*

Fitting *exercise 1*

What happens if the polynomial has only the powers 0,2,4 ?
Do you get an acceptable fit ?

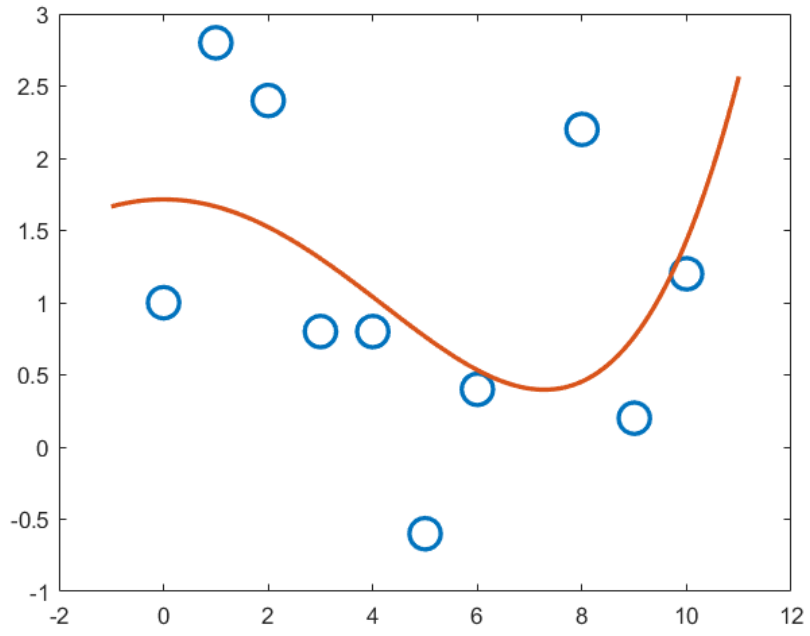
```
x=[0   1   2   3   4    5   6   8   9   10]';
y=2*[0.5,1.4,1.2,0.4,0.4,-0.3,0.2,1.1,0.1,0.6]';
```

%The following code fits a polynomial with even powers up to 4 to the data:

```
pexp=0:2:4;   % powers
dm=x.^pexp;   % construct the designmatrix
p=dm\y;       % solve y=p0+p1*x+…; find the parameters
```

%Plot the datapoints and the polynomial :

```
xx=((min(x)-1):0.1:(max(x)+1))';% smooth x-points
yy=xx.^pexp*p;                   % corresponding y
plot(x,y,'o',xx,yy,'-', ...
'linewidth',2,'markersize',14);
```



You should get a picture like the one to the right.
Clearly the fit is much worse. The answer is therefore 'no'

Fitting *exercise 2*

What happens if the polynomial has the powers 0 to 10 ?
Do you get a good fit ? What is the problem ?

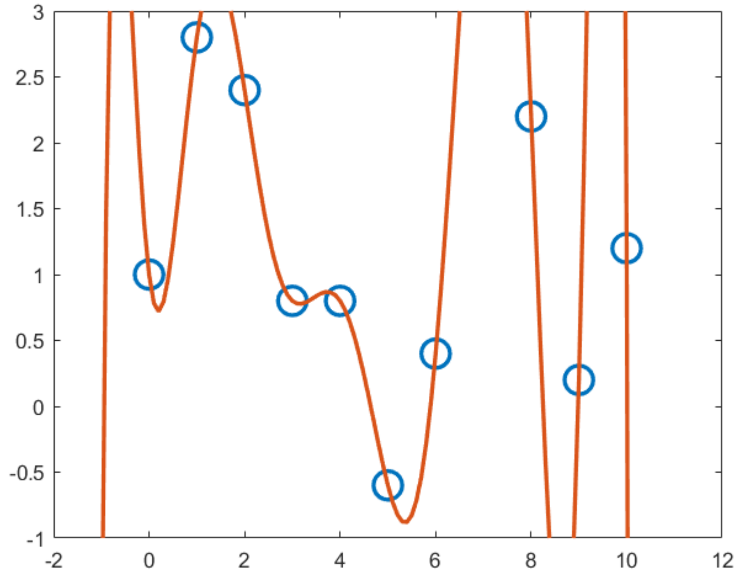
```
x=[0   1   2   3   4    5   6   8   9   10]';
y=2*[0.5,1.4,1.2,0.4,0.4,-0.3,0.2,1.1,0.1,0.6]';
```

%The following code fits a polynomial with even powers up to 4 to the data:

```
pexp=0:10;    % powers
dm=x.^pexp;   % construct the designmatrix
p=dm\y;       % solve y=p0+p1*x+…; find the parameters
```

%Plot the datapoints and the polynomial :

```
xx=((min(x)-1):0.1:(max(x)+1))';% smooth x-points
yy=xx.^pexp*p;                   % corresponding y
plot(x,y,'o',xx,yy,'-', ...
'linewidth',2,'markersize',14); set(gca,'ylim',[-1 3]);
```



You should get a picture like the one to the right.
Clearly the fit is strange. The problem is ‘overfitting’.

Fitting *exercise 3*

Can you also fit with polynomial powers that are not integer ?
For example: 0,0.5,1 … 4 ? Is there a problem ?

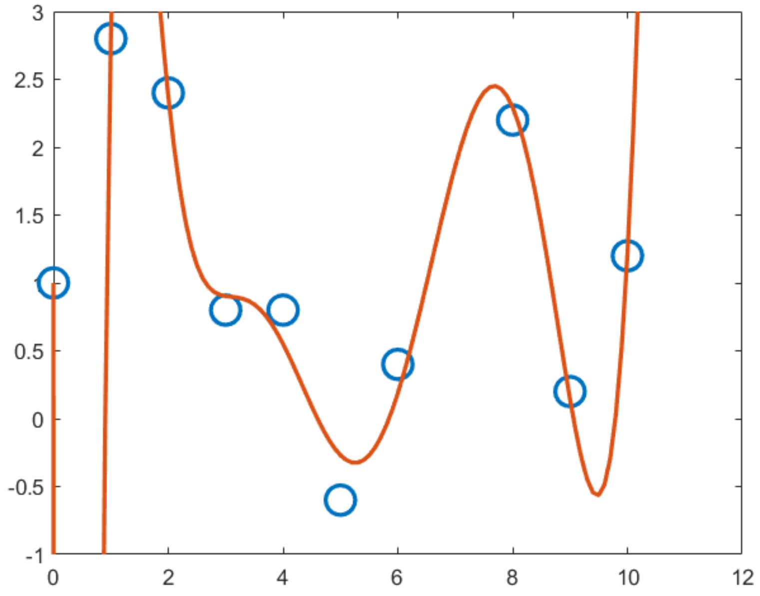
```
x=[0   1   2   3   4    5   6   8   9   10]';
y=2*[0.5,1.4,1.2,0.4,0.4,-0.3,0.2,1.1,0.1,0.6]';
```

%The following code fits a polynomial with even powers up to 4 to the data:

```
pexp=0:0.5:4; % powers
dm=x.^pexp;   % construct the designmatrix
p=dm\y;       % solve y=p0+p1*x+…; find the parameters
```

%Plot the datapoints and the polynomial :

```
xx=((min(x)):0.1:(max(x)+1))';% smooth x-points
yy=xx.^pexp*p;                 % corresponding y
plot(x,y,'o',xx,yy,'-', ...
'linewidth',2,'markersize',14); set(gca,'ylim',[-1 3]);
```



You should get a picture like the one to the right.

Answer: Yes ! – why not ? Any function linear in the parameters (!) is ok.
But you must take care that x is ≥ 0, though.

Fitting *exercise 4*

Can you also fit with sin functions ? For example:
$y=p_1*\sin(x)+p_2*\sin(2x)+\ldots\ p_n*\sin(5x)$? Is there a problem ?

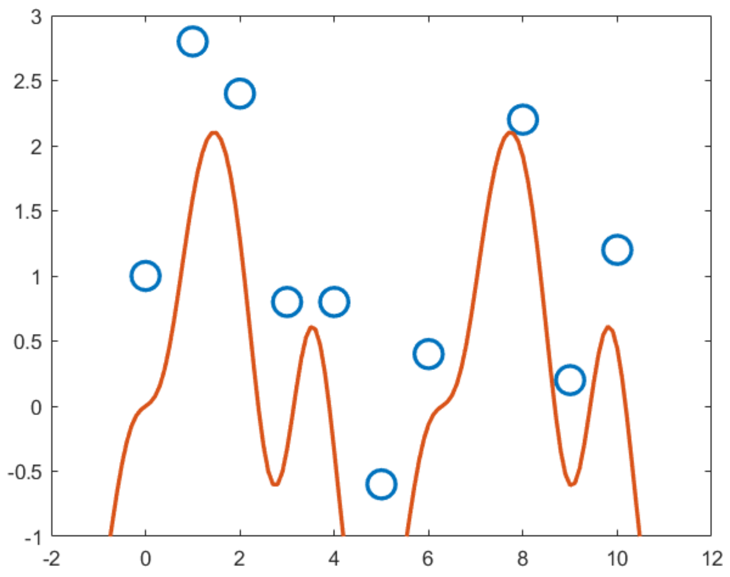
```
x=[0   1   2   3   4    5   6   8   9   10]';
y=2*[0.5,1.4,1.2,0.4,0.4,-0.3,0.2,1.1,0.1,0.6]';
```

%The following code fits a polynomial with even powers up to 4 to the data:

```
sinfac=1:5;          % factors in the sinus terms
dm=sin(x.*sinfac);   % construct the designmatrix
p=dm\y;       % solve y=p0+p1*x+…; find the parameters
```

%Plot the datapoints and the polynomial :

```
xx=((min(x)-1):0.1:(max(x)+1))';% smooth x-points
yy=sin(xx.*sinfac)*p;          % corresponding y
plot(x,y,'o',xx,yy,'-', ...
'linewidth',2,'markersize',14); set(gca,'ylim',[-1 3]);
```



You should get a picture like the one to the right

Answer again: yes ! – why not ? Any function linear in the parameters (!) is ok.
Here you see that the sin-terms do not reach the points – why ?

Fitting *exercise 5*

You have seen that the sin-terms do not reach the points !
Can you repair that ? Try it by allowing a shift : $y=p_1+p_2*\sin(x)+p_3*\sin(2x)+\ldots\ p_n*\sin(5x)$?

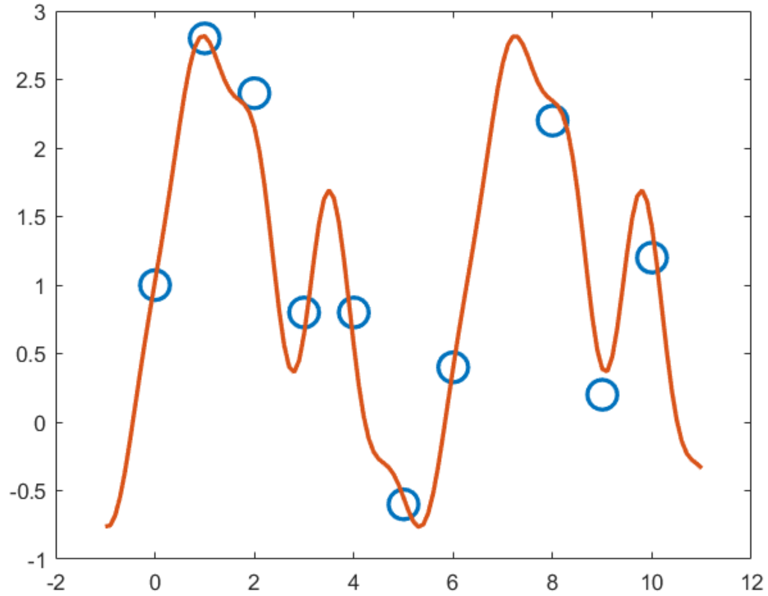
```
x=[0   1   2   3   4    5   6   8   9   10]';
y=2*[0.5,1.4,1.2,0.4,0.4,-0.3,0.2,1.1,0.1,0.6]';
```

%The following code fits a polynomial with even powers up to 4 to the data:

```
sinfac=1:5;          % factors in the sinus terms
dm=[x.^0 sin(x.*sinfac)];  % construct the designmatrix
p=dm\y;      % solve y=p0+p1*x+…; find the parameters
```

%Plot the datapoints and the polynomial :

```
xx=((min(x)-1):0.1:(max(x)+1))';  % smooth x-points
yy=[xx.^0 sin(xx.*sinfac)]*p;    % corresponding y
plot(x,y,'o',xx,yy,'-', ...
'linewidth',2,'markersize',14); set(gca,'ylim',[-1 3]);
```



You should get a picture like the one to the right

Yeah ! Now we can be happy ! But we should still check if less terms can also be used.

Fitting *exercises*

END (We have covered simple least-squares fitting).