FITTING

Discussion points:

- Linear and nonlinear
- Quality criterion
- Given functions (physical) and other possibilities

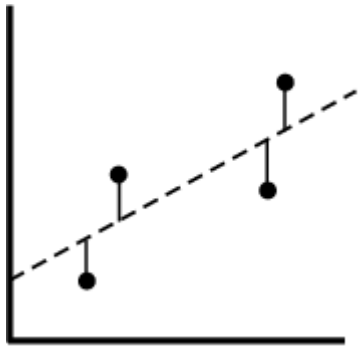
FITTING (SHORT):

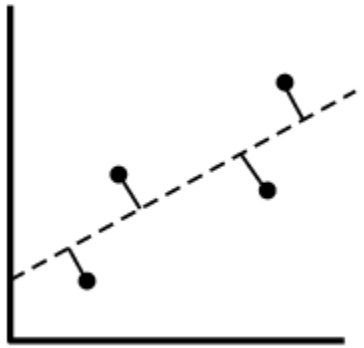
- **Linear and nonlinear**
  - Linear: The parameters c_i to be fitted (not the x_i !) must be simple weights of the terms containing the x_i .
    Example: $y= c_1 \sin(x) + c_2 \exp(-2x)$
  - Nonlinear: The parameters c_i to be fitted (not the x_i !) are not simple weights of the terms containing the x_i .
    Example:
    $y= \sin(c_1 x) + \exp(-2x/c_2)$

FITTING:

- **Quality criterion:**

1) Errors only in y or errors in x and y





2) How to balance ?

The extremes are:

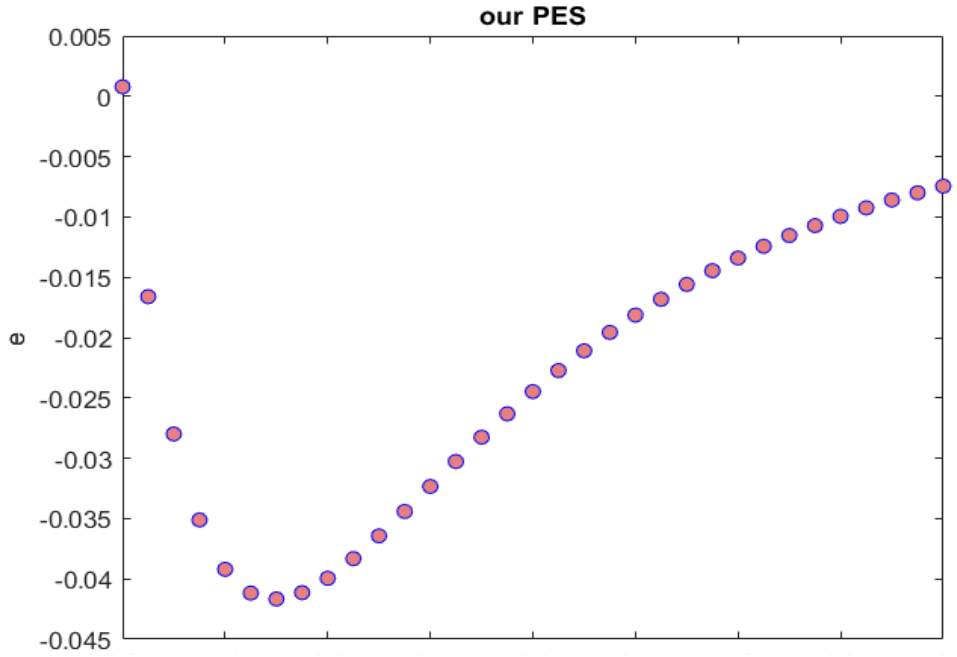
a) $\Sigma\ |y-f| \rightarrow$ minimum: "Points away from f are not very bad"

b) $\Sigma\ (y-f)^{\infty} \rightarrow$ minimum: "No point is allowed to be bad"

1. (Most) simple fitting (with Matlab code)

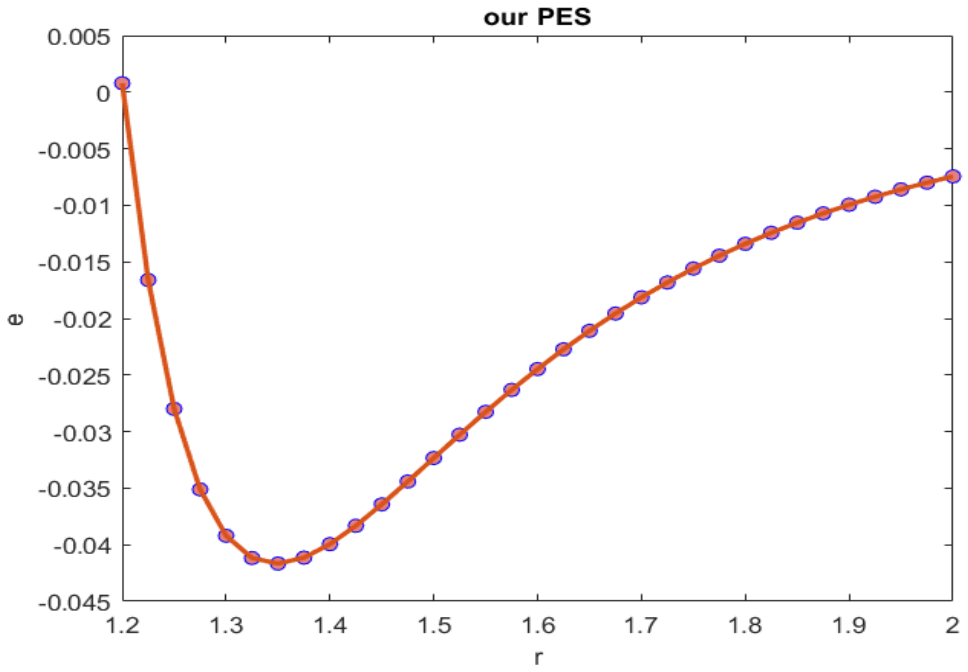
A) We have other (r,e) datapoints

```
r=(1.2:0.025:2)';                % distances
e=1.5./r.^12-0.5./r.^6;          % energies
plot(r,e,'o','MarkerEdgeColor','b','MarkerFaceColor',[0.9,0.5,0.5])
xlabel('r'),ylabel('e');title('our PES');
```



B) We fit them to a formula

```
dm=[r.^-12 -r.^-6];   % formula represented as 'design matrix'
p1=lscov(dm,e);       % matrix-based fitting (dm*p1=e -> p1=e/dm)
hold on; plot(r,dm*p1,'linewidth',2); hold off
```



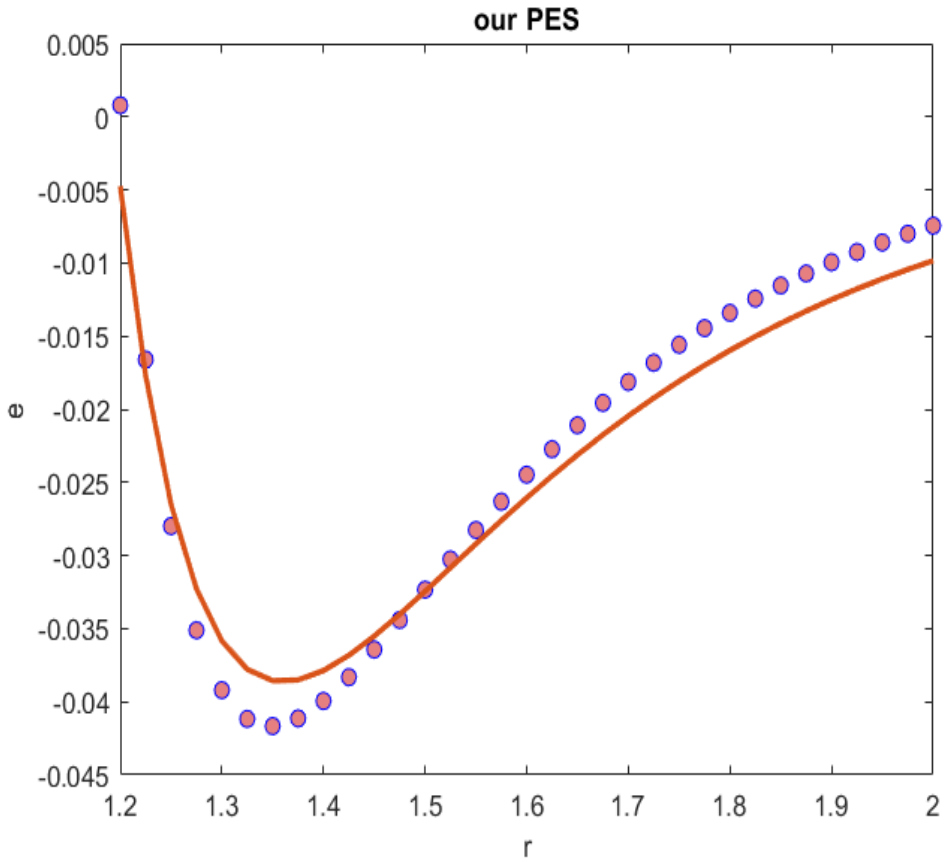
2. With a different formula

A) We have the same (r,e) datapoints

```
r=(1.2:0.025:2)';                % distances
e=1.5./r.^12-0.5./r.^6;          % energies
plot(r,e,'o','MarkerEdgeColor','b','MarkerFaceColor',[0.9,0.5,0.5])
xlabel('r'),ylabel('e');title('our PES');
```

B) We fit them to a different formula

```
dm=[r.^-11 -r.^-5];   % formula represented as 'design matrix'
p1=lscov(dm,e);       % matrix-based fitting (dm*p1=e -> p1=e/dm)
hold on; plot(r,dm*p1,'linewidth',2); hold off
```



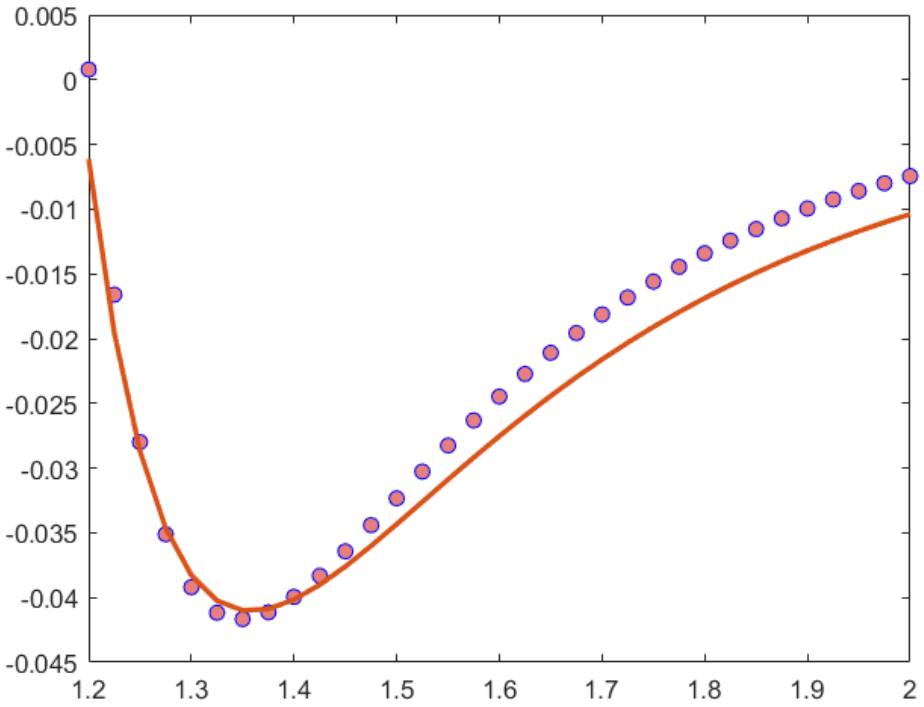
3. With an emphasis of the minumum

A) We have the same (r,e) datapoints

```
r=(1.2:0.025:2)';                % distances
e=1.5./r.^12-0.5./r.^6;          % energies
plot(r,e,'o','MarkerEdgeColor','b','MarkerFaceColor',[0.9,0.5,0.5])
xlabel('r'),ylabel('e');title('our PES');
```

B) We fit them to a different formula, with weigths

```
dm=[r.^-11 -r.^-5];   % formula represented as 'design matrix'
w=ones(size(r));
w(e<-0.04)=20;
p1=lscov(dm,e,w);   % matrix-based fitting (dm*p1=e -> p1=e/dm)
hold on; plot(r,dm*p1,'linewidth',2); hold off
```



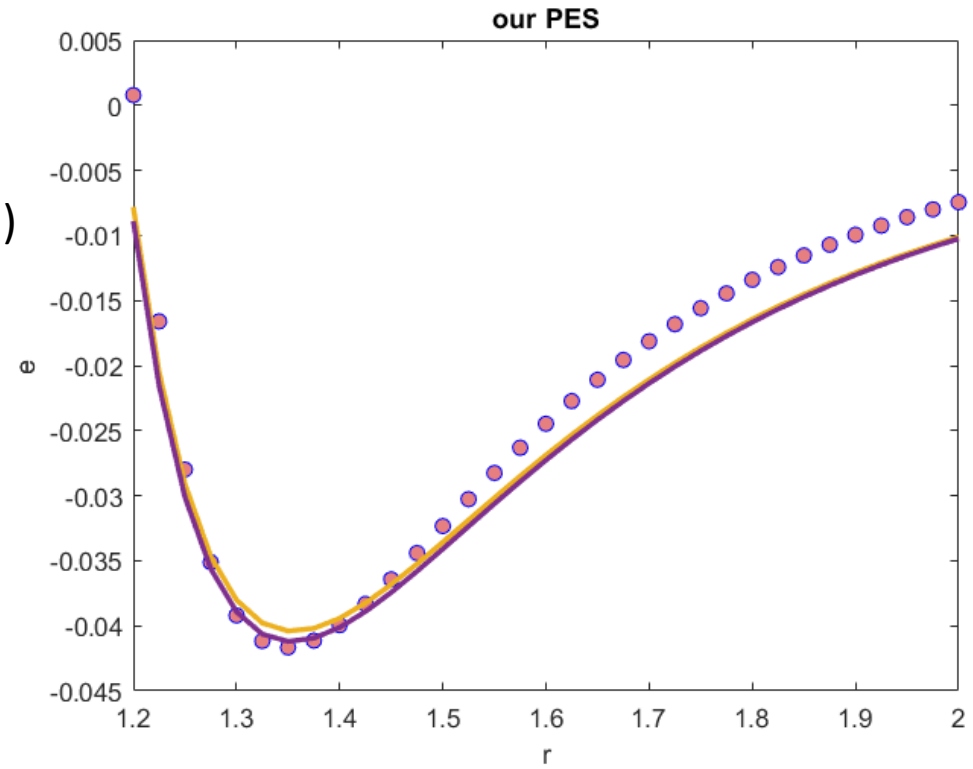
3. With an emphasis of the minumum

A) We have the same (r,e) datapoints

```
r=(1.2:0.025:2)';                % distances
e=1.5./r.^12-0.5./r.^6;          % energies
plot(r,e,'o','MarkerEdgeColor','b','MarkerFaceColor',[0.9,0.5,0.5])
xlabel('r'),ylabel('e');title('our PES');
```

B) We fit them to a different formula, with other weigths

```
dm=[r.^-11 -r.^-5];   % formula represented as 'design matrix'
w=1./(e+0.042);
p1=lscov(dm,e,w);   % matrix-based fitting (dm*p1=e -> p1=e/dm)
hold on; plot(r,dm*p1,'linewidth',2); hold off
```



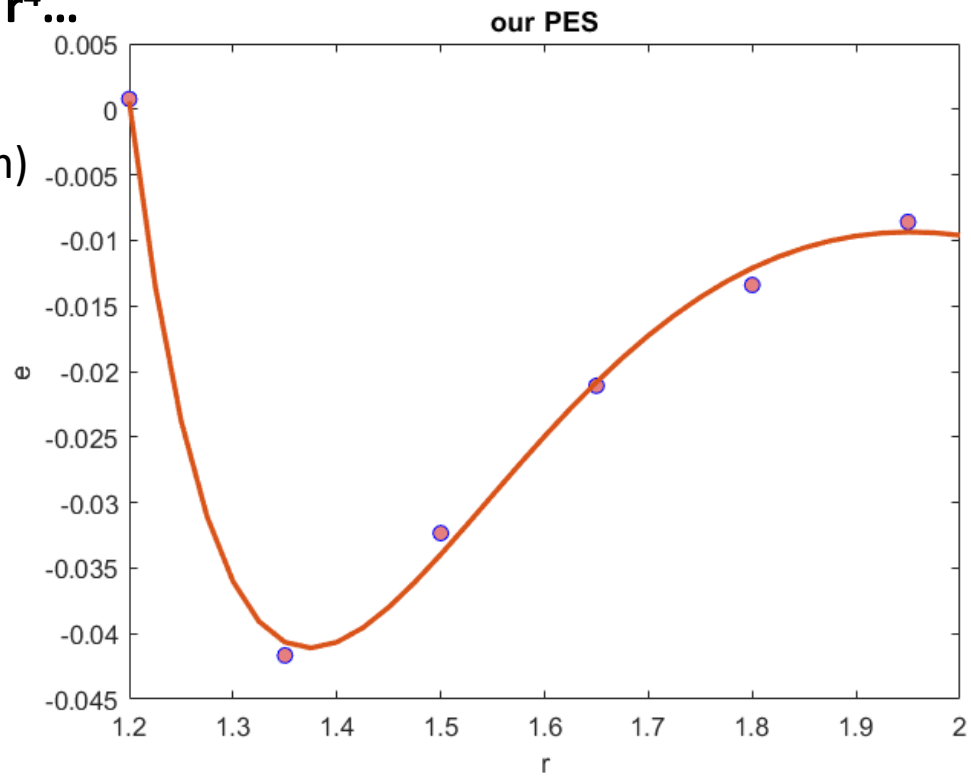
4. Overfitting (1)

A) We have the same LJ function, a bit less (r,e) datapoints

```
r=(1.2:0.15:2)';                % distances
e=1.5./r.^12-0.5./r.^6;         % energies
plot(r,e,'o','MarkerEdgeColor','b','MarkerFaceColor',[0.9,0.5,0.5])
xlabel('r'),ylabel('e');title('our PES');
```

B) Now we try to fit them to a polynomial in $1/r$, $1/r^2$, $1/r^3$, $1/r^4$…

```
dm=[r.^-(1:4)]; % formula represented as 'design matrix'
p1=lscov(dm,e);      % matrix-based fitting (dm*p1=e -> p1=e/dm)
r2=(1.2:0.025:2)'; dm2=[r2.^-(1:4)];
hold on; plot(r2,dm2*p1,'linewidth',2); hold off
```



4. Overfitting (2)

A) We have the same LJ function, a bit less (r,e) datapoints, and some noise

```
r=(1.2:0.15:2)';                    % distances
e=1.5./r.^12-0.5./r.^6; e=e+(rand(size(e))-0.5)*0.01;          % energies
plot(r,e,'o','MarkerEdgeColor','b','MarkerFaceColor',[0.9,0.5,0.5])
xlabel('r'),ylabel('e');title('our PES');
```

B) Now we try to fit them to a polynomial in 1/r, … , $1/r^6$…

```
dm=[r.^-(1:8)]; % formula represented as 'design matrix'
p1=lscov(dm,e);      % matrix-based fitting (dm*p1=e -> p1=e/dm)
r2=(1.2:0.025:2)'; dm2=[r2.^-(1:8)];
hold on; plot(r2,dm2*p1,'linewidth',2); hold off
```