Messdaten:

```
x = [1 2 3 4]'
```

```
x = 4×1
     1
     2
     3
     4
```

```
y = [10 8 9 11]'
```

```
y = 4×1
    10
     8
     9
    11
```

Modellgleichung:

$p(t) = \alpha_1 t + \alpha_0$

Systemmatrix:

$$\begin{pmatrix} 1 & 1 \\ 2 & 1 \\ 3 & 1 \\ 4 & 1 \end{pmatrix} \begin{pmatrix} \alpha_1 \\ \alpha_0 \end{pmatrix} \approx \begin{pmatrix} 10 \\ 8 \\ 9 \\ 11 \end{pmatrix}$$

```
vs = [x ones(size(x))]
```

```
vs = 4×2
     1     1
     2     1
     3     1
     4     1
```

QR-Zerlegung

```
[Q, R] = qr(vs);
```

Löse:

$$R \begin{pmatrix} \alpha_1 \\ \alpha_2 \end{pmatrix} = Q^T y$$

```
right_side = Q' * y;
alphas_manual = R \ right_side
```

```
alphas_manual = 2×1
    0.4000
    8.5000
```

Mit Funktion:

```
alphas_func = fit_linear(x, y, 1)
```

```
alphas_func = 2×1
    0.4000
    8.5000
```

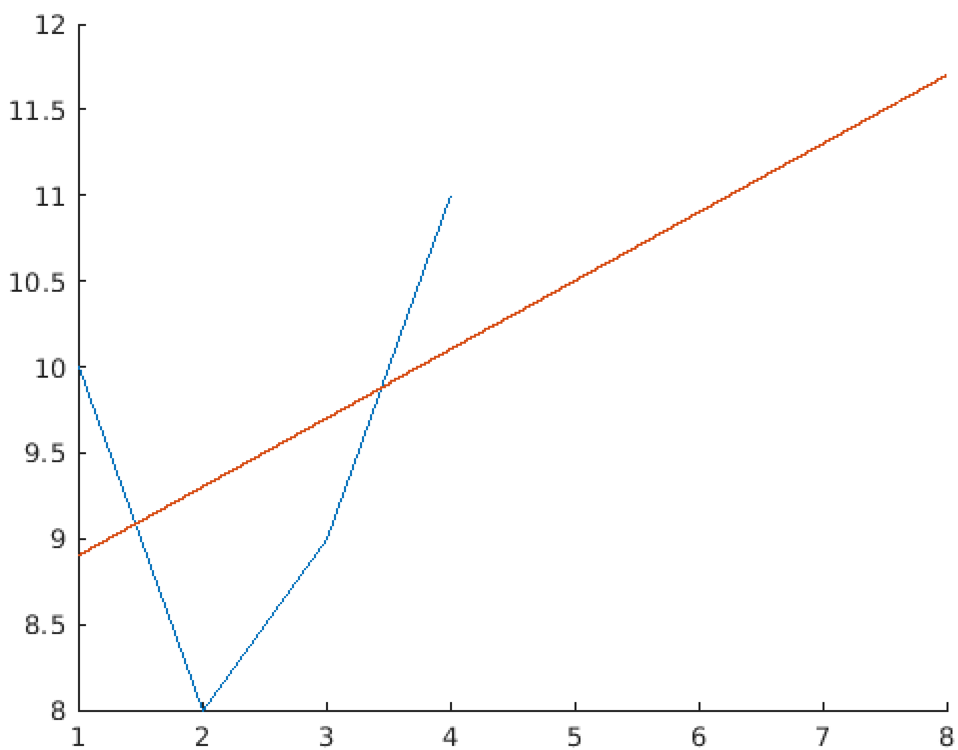
Vorhersage mit Modellfunktion:

```
model = fit_linear_model(x, y, 1);
model(8)
```

```
ans = 11.7000
```

```
xs = linspace(1, 8, 50000);
ys = arrayfun(model, xs);

clf
hold on
plot(x, y)
plot(xs, ys)
hold off
```



Allgemein:

```
function params = fit_linear(x, y, degree)
```

$d := degree$

Modellgleichung: $p(t) = \sum_{k=0}^{d} \alpha_k t^k$

```
    vs = x .^ (degree:-1:0);
```

Systemgleichung: $\begin{pmatrix} x_1^d & \cdots & x_1^0 \\ \vdots & & \vdots \\ x_m^d & \cdots & x_m^0 \end{pmatrix} \begin{pmatrix} \alpha_d \\ \vdots \\ \alpha_0 \end{pmatrix} \approx y$

```
    [Q, R] = qr(vs);
```

Löse: $R \begin{pmatrix} \alpha_d \\ \vdots \\ \alpha_0 \end{pmatrix} = Q^T y$

```
    params = R \ (Q' * y);
end

function model = fit_linear_model(x, y, degree)
    params = fit_linear(x, y, degree);
    model = @(t) (t .^ (degree:-1:0)) * params;
end
```