```
julia> H4=[1/(i+j-1) for i=1:4, j=1:4]
4×4 Matrix{Float64}:
 1.0       0.5       0.333333  0.25
 0.5       0.333333  0.25      0.2
 0.333333  0.25      0.2       0.166667
 0.25      0.2       0.166667  0.142857
```

```
julia> cond(H4)
15513.738738892924
```

$\text{Cond} = \|H_4\| \, \|H_4^{-1}\|$

general rule... you can't solve $Ax=b$ to more than $15 - \log_{10} k$ digits of precision. Because you can't even check the answer that accurately...

```
julia> invH4=inv(H4)
4×4 Matrix{Float64}:
   16.0   -120.0    240.0   -140.0
 -120.0   1200.0  -2700.0   1680.0
  240.0  -2700.0   6480.0  -4200.0
 -140.0   1680.0  -4200.0   2800.0
```

$Ha = b$

```
julia> opnorm(H4)*opnorm(invH4)
15513.738738933602
```

recall finding H^{-1} is the same as solving $Ha = e_k$ for $k = 1, 2, 3, 4$

Math 330... Augmented matrix for finding inverses

$[H \mid I] \rightarrow$ reduced echelon for $[I \mid H^{-1}]$

$I = [e_1 | e_2 | \cdots | e_n]$ same as solving $Ha = e_k$ for $k = 1, 2, \ldots, n$

```
julia> H7=[1/(i+j-1) for i=1:7, j=1:7]
7×7 Matrix{Float64}:
 1.0       0.5       0.333333  0.25      0.2       0.166667  0.142857
 0.5       0.333333  0.25      0.2       0.166667  0.142857  0.125
 0.333333  0.25      0.2       0.166667  0.142857  0.125     0.111111
 0.25      0.2       0.166667  0.142857  0.125     0.111111  0.1
 0.2       0.166667  0.142857  0.125     0.111111  0.1       0.0909091
 0.166667  0.142857  0.125     0.111111  0.1       0.0909091 0.0833333
 0.142857  0.125     0.111111  0.1       0.0909091 0.0833333 0.0769231

julia> cond(H7)
4.753673567446793e8
```

← means that about 8 digit of precision are lost when solving $H_7 a = b$ using any method... because we can't even check the solution with any greater accuracy...

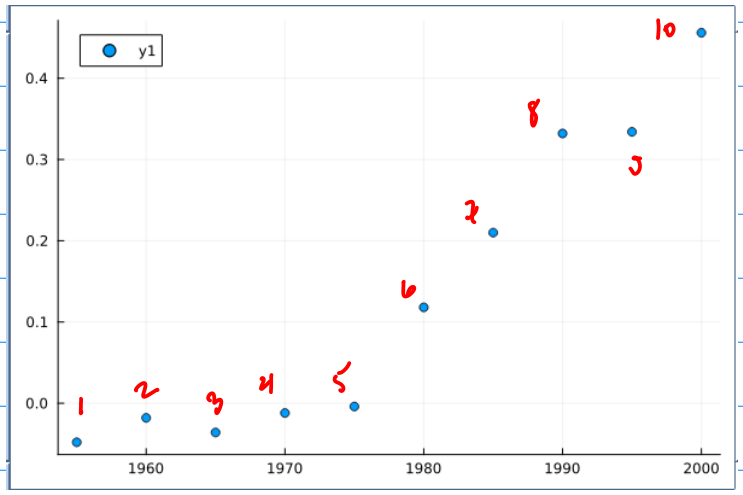
On one hand a 7×7 matrix doesn't seem large. On the other hand a 6^{th} degree polynomial has a fairly high degree...

Recall Lecture 5

$$V = \begin{bmatrix} 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \\ 1 & x_3 & x_3^2 \end{bmatrix}, \quad c = \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix}, \quad y = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}$$

```
year = 1955:5:2000
temp = [ -0.0480, -0.0180, -0.0360, -0.0120, -0.0040,
          0.1180, 0.2100, 0.3320, 0.3340, 0.4560 ]

scatter(year,temp,label="data",
    xlabel="year",ylabel="anomaly (degrees C)",leg=:bottomright)
```



fits a polynomial of degree 9.

$V =$ `julia> V=[float(year[i])^j for i=1:n, j=0:n-1]` (n=10)

Solve $Vc = temp$

$c = V \backslash temp$

```
julia> t=(year.-1950)/10
0.5:0.5:5.0

julia> V=[float(t[i])^j for  i=1:n, j=0:n-1]
10×10 Matrix{Float64}:
```

shift dates to avoid really big numbers and small ones...

make vandermond matrix

```
julia> c=V\temp
```

solve for coefficients

```
julia> p=Polynomial(c)
```

make a polynomial $p(x) = \sum_{i=1}^{n} c_i x^{i-1}$

```
julia> scatter(t,temp)
```

plot the points we will interpolate

```
julia> plot!(t,p.(t))
```

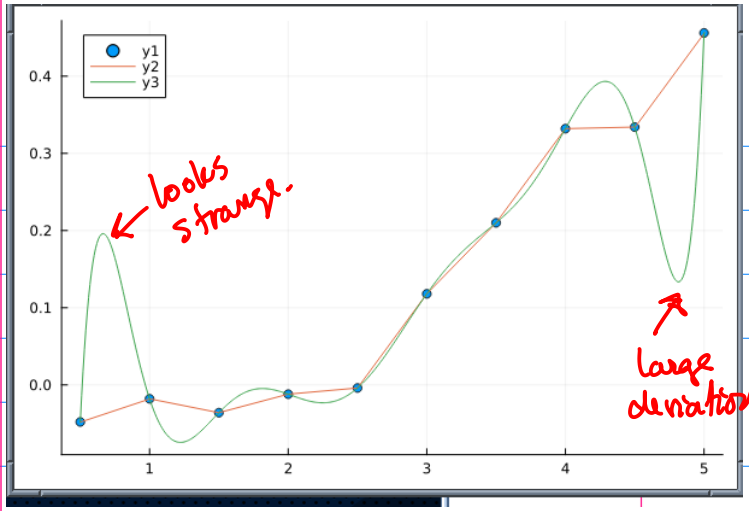
check that the polynomial passes through the points (polygonal line through the points)

```
julia> ts=0.5:0.01:5
0.5:0.01:5.0
```

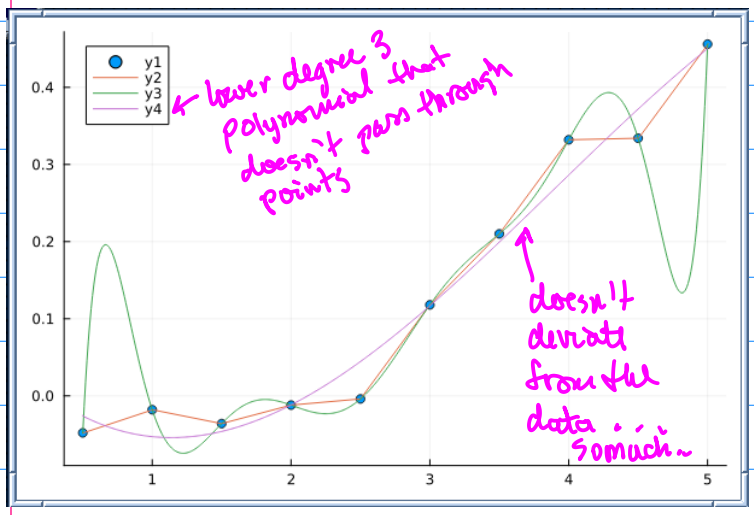
sample more times

```
julia> plot!(ts,p.(ts))
```

draw the interpolated fit.




```
julia> V=[float(t[i])^j for  i=1:n, j=0:3]
10×4 Matrix{Float64}:
 1.0  0.5   0.25    0.125
 1.0  1.0   1.0     1.0
 1.0  1.5   2.25    3.375
 1.0  2.0   4.0     8.0
 1.0  2.5   6.25   15.625
 1.0  3.0   9.0    27.0
 1.0  3.5  12.25   42.875
 1.0  4.0  16.0    64.0
 1.0  4.5  20.25   91.125
 1.0  5.0  25.0   125.0
```




```
julia> c=V\temp
4-element Vector{Float64}:
  0.03986666666666851
 -0.17520745920746172
  0.0901958041958051
 -0.007748251748251838

julia> p3=Polynomial(c)
Polynomial(0.039866666666668
251838*x^3)

julia> plot!(ts,p3.(ts))
```

purple line