Polynomial interpolation

-> barycentric formula

-> Chebyshev basis

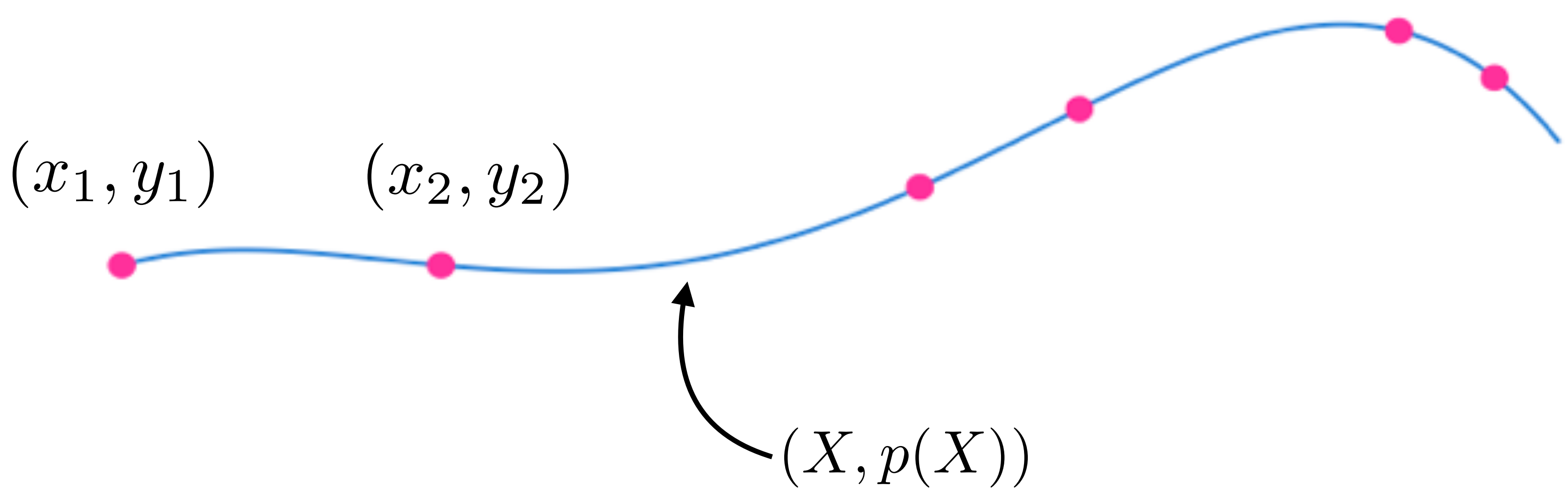
notes based on Trefethen's exposition in fifth chapter of ATAP:
https://people.maths.ox.ac.uk/trefethen/ATAP/ATAPfirst6chapters.pdf

problem: interpolation

-> Given: points $(x_1, y_1), (x_2, y_2), \cdots, (x_n, y_n)$

-> there is a unique polynomial $p(x)$ of degree $n-1$ interpolating these points

-> Goal: evaluate $p(X)$ where X is a number or a (possibly very long) vector



bad solution: get monomial coefficients

```
41 n <- 9 # this method breaks well before n=50.
42 x <- cos(0:n*pi/n)   # grid/data x-values
43 y <- cos((x+1)^2*pi)   # data y-values
44 A <- outer(x,0:(length(x)-1),"^") # Vandermonde matrix
45 c <- qr.solve(A,y) # c contains monomial coefficients
46 Xp <- -50:50/50 # plotting xvals
47 Yp <- outer(Xp,0:(length(x)-1),"^") %*% c # vectorized eval of poly
48 plot(Xp,Yp,'l')      # curve
49 points(x,y,pch=20)  # points
```

Vandermonde matrix of line 44 is very ill-conditioned for n > about 10

Do not use this widespread method.

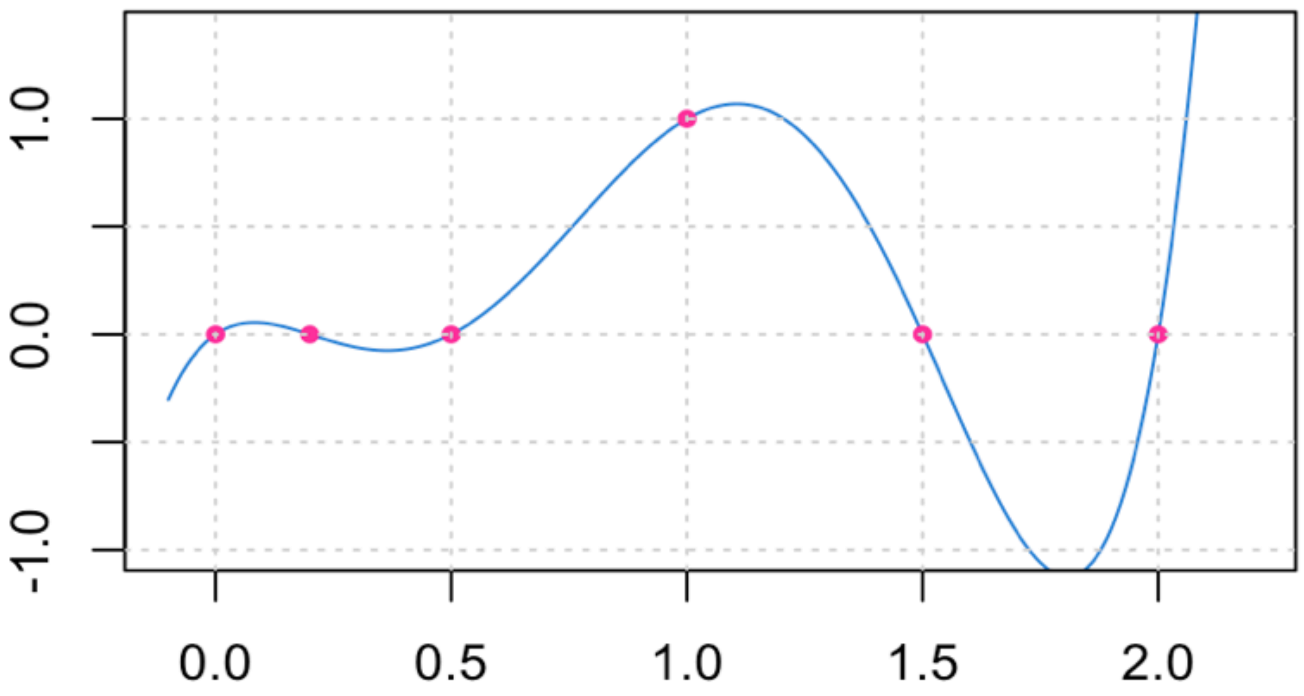
good solution 1: barycentric formula

Given: points $(x_1, y_1), (x_2, y_2), \cdots, (x_n, y_n)$

define ℓ_j as poly of degree $n-1$ such that:

$$\ell_j(x_k) = \begin{cases} 1 & j = k \\ 0 & j \neq k \end{cases}$$

ℓ_j is called a "Lagrange polynomial"



if the x_j-values are $(0, 0.2, 0.5, 1, 1.5, 2)$, then this curve is ℓ_4

good solution 1: barycentric formula

Given: points $(x_1, y_1), (x_2, y_2), \cdots, (x_n, y_n)$

define ℓ_j as poly of degree $n-1$ such that:

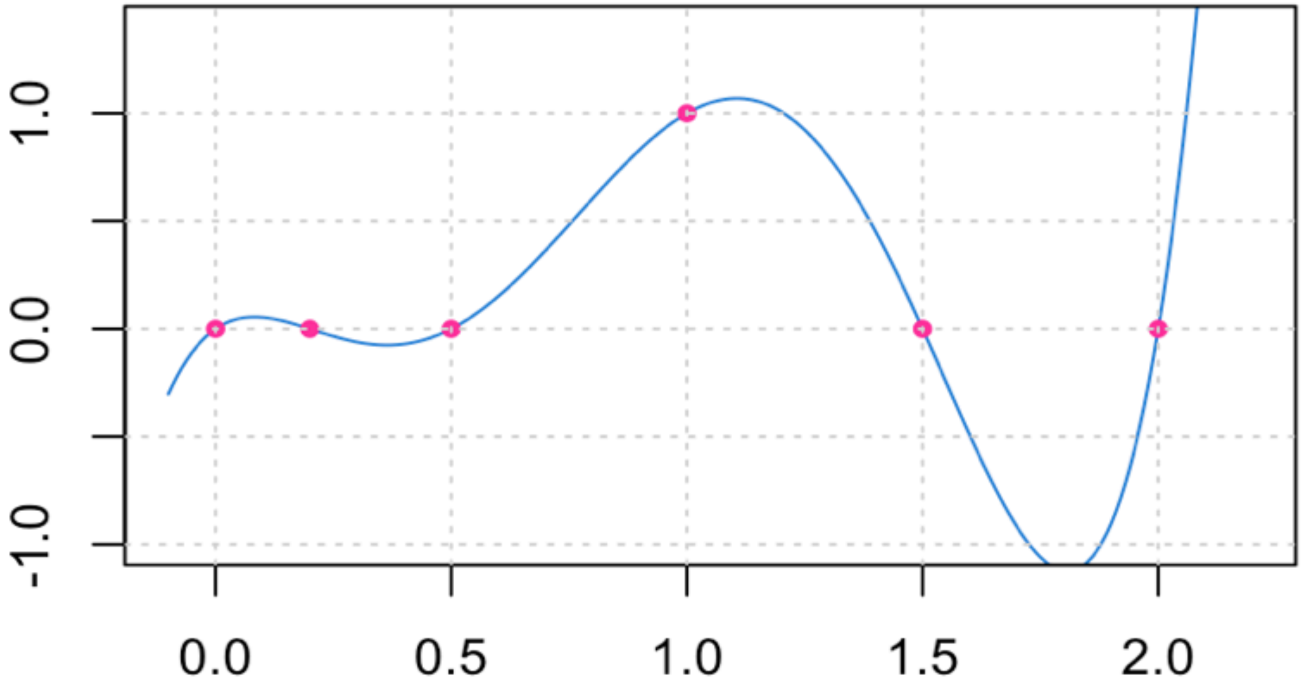
$$\ell_j(x_k) = \begin{cases} 1 & j = k \\ 0 & j \neq k \end{cases}$$

ℓ_j is called a "Lagrange polynomial"

neat formula for
Lagrange polynomial:

$$\ell_j(X) = \frac{\prod_{k\neq j}(X - x_k)}{\prod_{k \neq j}(x_j - x_k)}$$

numerator is poly of deg (n-1)
denominator is constant wrt X
check: equals one if X=x_j
check: equals zero if X=____



if the x_j-values are $(0, 0.2, 0.5, 1, 1.5, 2)$, then this curve is ℓ_4

good solution 1: barycentric formula

Given: points $(x_1, y_1), (x_2, y_2), \cdots, (x_n, y_n)$

define ℓ_j as poly of degree $n-1$ such that:

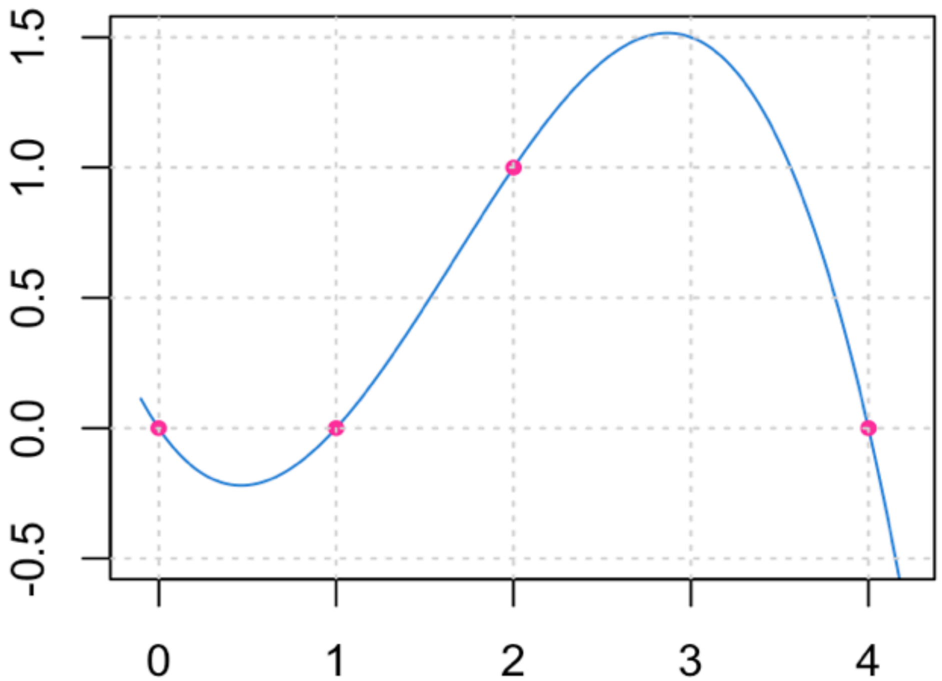
$$\ell_j(x_k) = \begin{cases} 1 & j = k \\ 0 & j \neq k \end{cases}$$

ℓ_j is called a "Lagrange polynomial"

neat formula for
Lagrange polynomial:

$$\ell_j(X) = \frac{\prod_{k \neq j}(X - x_k)}{\prod_{k \neq j}(x_j - x_k)}$$

numerator is poly of deg (n-1)
denominator is constant wrt X
check: equals one if X=x_j
check: equals zero if X=____



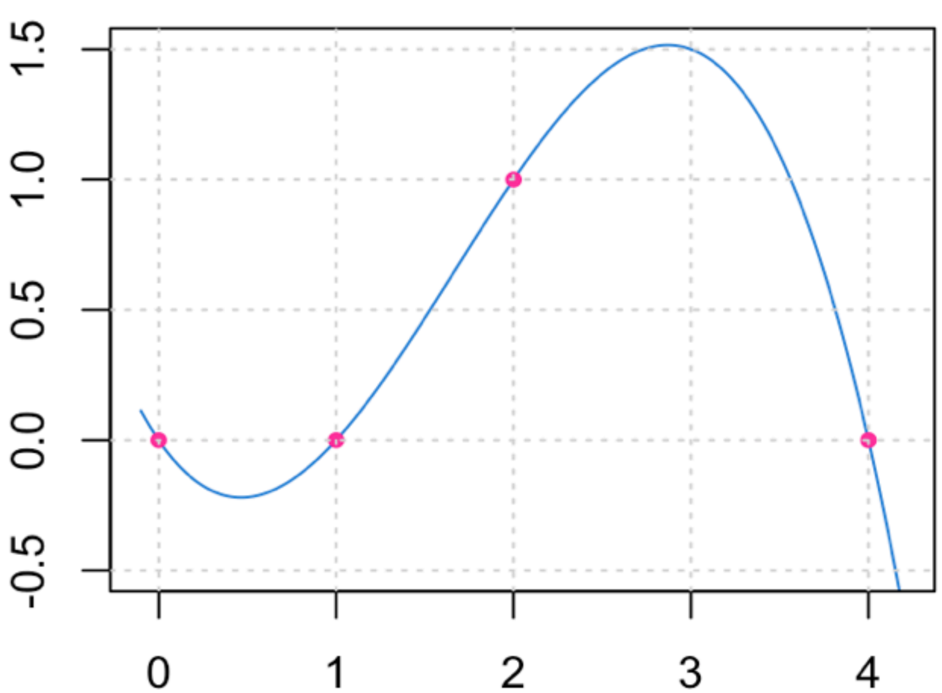
Exercise: reproduce this figure using R

````
```{r}
x <- c(0,1,2,4)
X <- -5:405/100
num <- (X-x[1]) *(X-x[2])*(X-x[4])
den <- (x[3]-x[1])*(x[3]-x[2])*(x[3]-x[4])
plot(X,num/den,'l')
points(x,c(0,0,1,0))
grid()
```
````

neat formula for
Lagrange polynomial:

$$\ell_j(X) = \frac{\prod_{k \neq j} (X - x_k)}{\prod_{k \neq j} (x_j - x_k)}$$

numerator is poly of deg (n-1)
denominator is constant wrt X
check: equals one if X=x_j
check: equals zero if X=____



Exercise: reproduce this figure using R

Full problem, with general x_j and y_j

$$p(X) = \sum_{j=1}^{n} y_j \ell_j(X)$$

Why?

Full problem, with general x_j and y_j

$$p(X) = \sum_{j=1}^{n} y_j \ell_j(X)$$
$$= \sum_{j=1}^{n} y_j \frac{\prod_{k\neq j}(X - x_k)}{\prod_{k\neq j}(x_j - x_k)}$$

define "barycentric weights" $\lambda_j = \dfrac{1}{\prod_{k\neq j}(x_j - x_k)}$

"node polynomial" $\ell(X) = \prod_{k=1}^{N}(X - x_k)$

can now evaluate! $p(X) = \sum_{j=1}^{n} y_j \ell_j(X) = \boxed{\ell(X) \sum_{j=1}^{n} \frac{y_j \lambda_j}{(X - x_j)}}$

First barycentric formula

$$p(X) = \sum_{j=1}^{n} y_j \ell_j(X) = \ell(X) \sum_{j=1}^{n} \frac{y_j \lambda_j}{(X - x_j)}$$

this perfectly hits every point (x_i, y_i)

$$\mathcal{O}(X) = \sum_{j=1}^{n} \ell_j(X) = \ell(X) \sum_{j=1}^{n} \frac{\lambda_j}{x_j - x_k}$$

this perfectly hits every point $(x_i, 1)$

$\mathcal{O}(x)$ is a super sneaky alter ego of... 1.

$$p(X) = \frac{p(X)}{\mathcal{O}(x)} = \boxed{\frac{\sum_{j=1}^{n} \frac{y_j \lambda_j}{X - x_j}}{\sum_{j=1}^{n} \frac{\lambda_j}{X - x_j}}}$$

"Second form of the barycentric formula"

note: if X is one of the original data points, X=x_i,
return y_i instead of dividing by zero

Polynomial interpolation

-> barycentric formula
-> Chebyshev basis
-> and beyond

notes based on Trefethen's exposition in ATAP:
https://people.maths.ox.ac.uk/trefethen/ATAP/ATAPfirst6chapters.pdf

Review Slide

Recall: the 'node polynomial' is $\ell(x) = \prod_{j=1}^{\infty} (x - x_j)$

Recall: the 'barycentric weights' are $\lambda_j = 1/\ell'(x_j) = 1/\left(\prod_{k \neq j}^{\infty} (x_j - x_k)\right)$

Plan: to evaluate the poly through $(x_1, y_1), \cdots, (x_n, y_n)$ at X:

-> compute barycentric weights

-> use one of the barycentric formulas:

$$\ell(X) \sum_{j=1}^{n} \frac{y_j \lambda_j}{(X - x_j)}$$

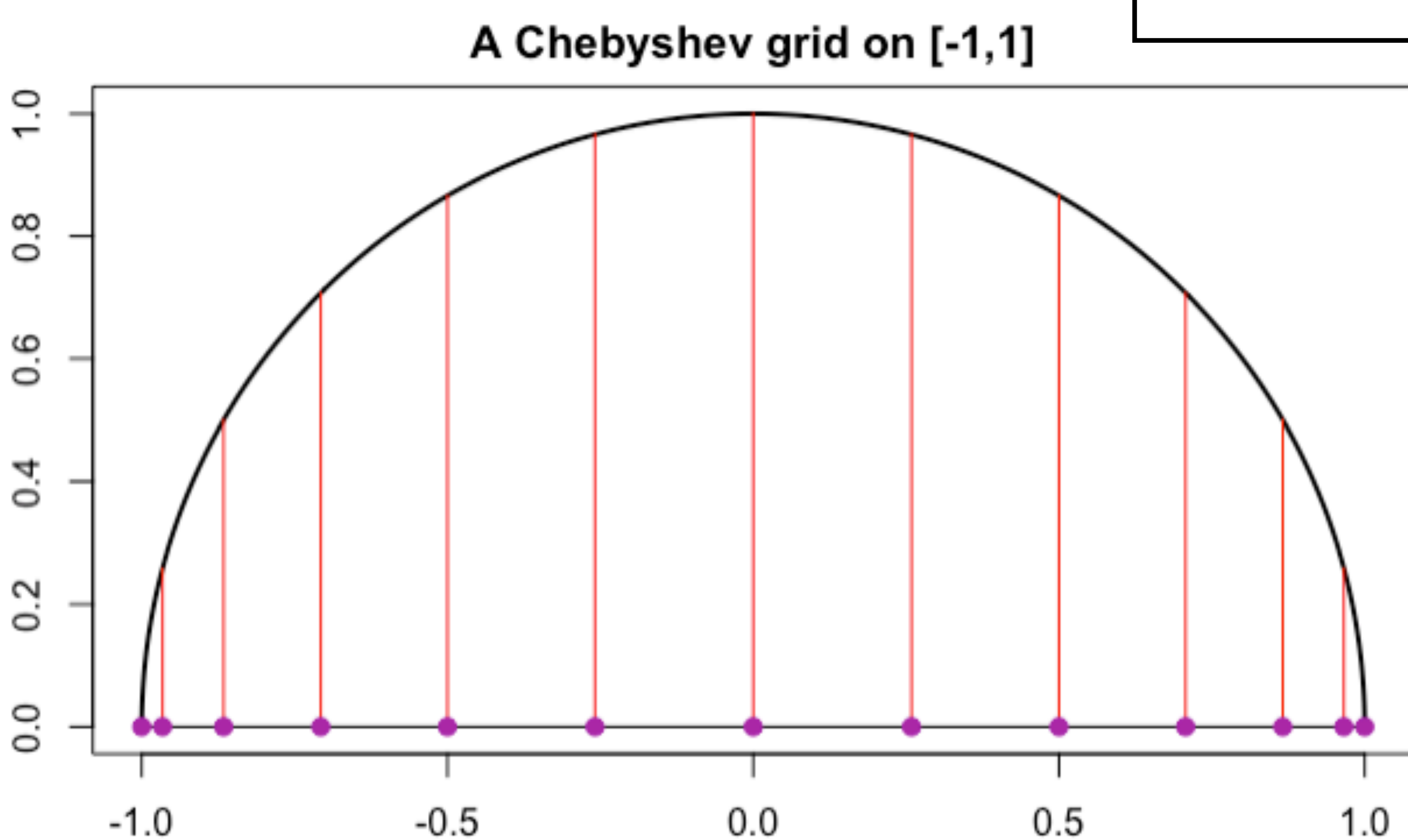
First BF

$$\frac{\sum_{j=1}^{n} \frac{y_j \lambda_j}{X - x_j}}{\sum_{j=1}^{n} \frac{\lambda_j}{X - x_j}}$$

Second BF

Barycentric weights are known in advance for some famous grids

```r
# make an illustration showing Chebyshev grids
n <- 12
x <- cos(0:n/n*pi)
y <- sin(0:n/n*pi)
xf <- cos(0:400/400*pi)
yf <- sin(0:400/400*pi)
plot(xf,yf,'l',lwd=2,asp=1,xlab="",ylab="",
      main="A Chebyshev grid on [-1,1]")
lines(c(-1,1),c(0,0),'l')
for (j in 1:(1+n)){
  lines(c(x[j],x[j]),c(0,y[j]),col="#ff0000")
  points(x[j],0,col="#aa00aa",pch=19)
}
```



Pink points are the "Chebyshev points"

aka "Chebyshev points of the second kind"

$$x_j = \cos\left(\frac{j\pi}{n}\right), \quad j = 0 \cdots n$$

note: we're counting from zero, so our data points are (x_0,y_0), …,(x_n, y_n)

Theorem (Riesz 1916): for these x_j, the barycentric weights are:

$$\lambda_j = (-1)^j \frac{2^{n-1}}{n} h_j$$

$$h_j = \begin{cases} 1/2 & j = 0 \text{ or } j = n \\ 1 & 0 < j < n \end{cases}$$

2nd barycentric formula for Chebyshev grid

$$p(X) = \frac{\sum_{j=0}^{n} \frac{y_j \lambda_j}{X - x_j}}{\sum_{j=0}^{n} \frac{\lambda_j}{X - x_j}}$$

Second BF

$$p(X) = \frac{\sum_{j=0}^{n} \frac{(-1)^j y_j h_j}{X - x_j}}{\sum_{j=0}^{n} \frac{(-1)^j h_j}{X - x_j}}$$

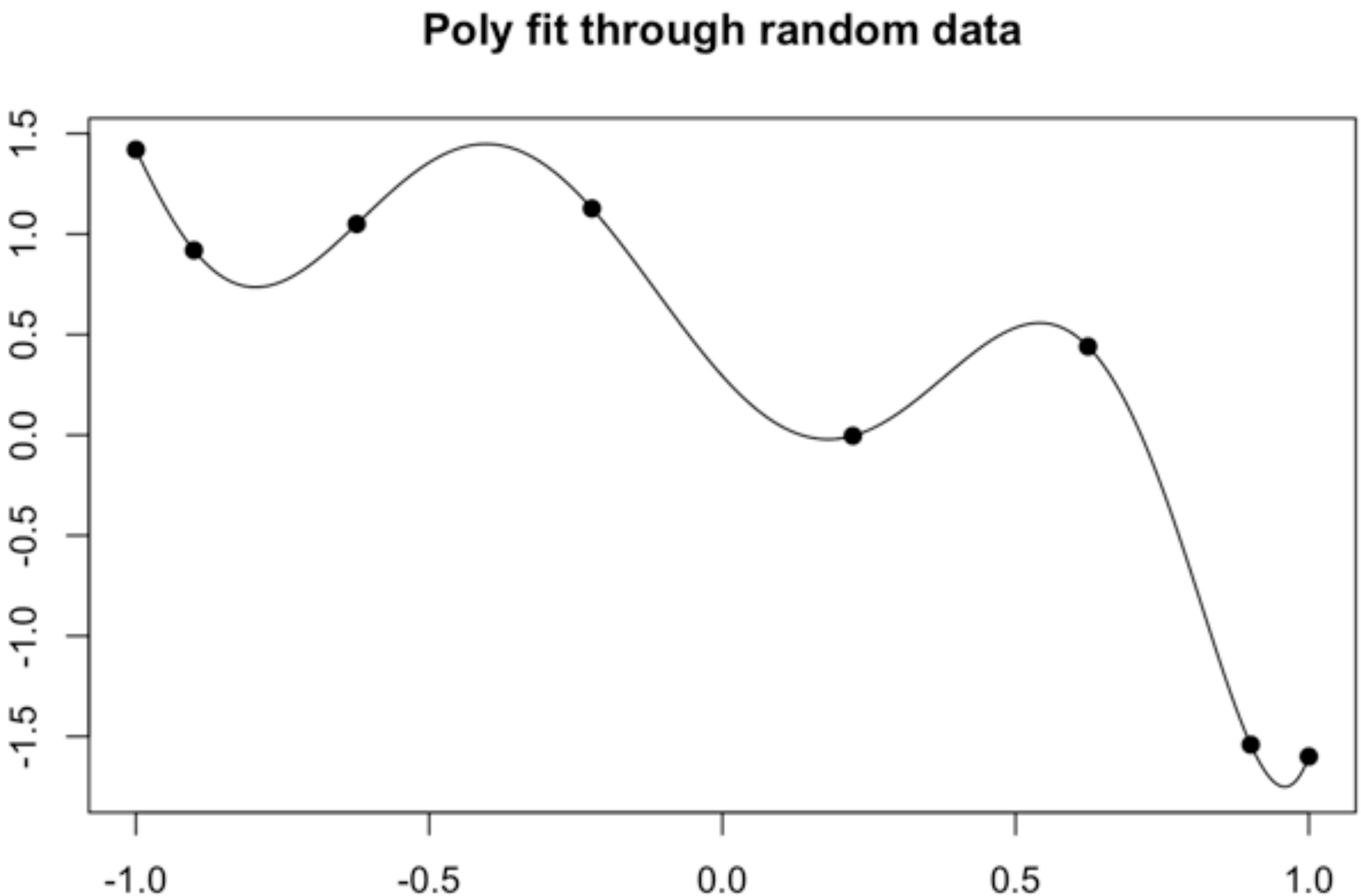
Second BF for Chebyshev grid
(Salzer 1972)

$$\lambda_j = (-1)^j \frac{2^{n-1}}{n} h_j$$

$$h_j = \begin{cases} 1/2 & j = 0 \text{ or } j = n \\ 1 & 0 < j < n \end{cases}$$

Green formula is wonderful: no problems on machine, and cheap.

Usual caveats:
(1) if X=x_j, the output should be y_j. Don't divide by 0
(2) Only use for interpolation, not extrapolation. Require $-1 \leq X \leq 1$


Poly fit through random data
1.5
1.0
0.5
0.0
-0.5
-1.0
-1.5
-1.0
-0.5
0.0
0.5
1.0