UNIVERSITY OF WATERLOO
FACULTY OF ENGINEERING
Department of Electrical &
Computer Engineering

ECE 204 *Numerical methods*

Approximating a point using least-squares best-fitting polynomials

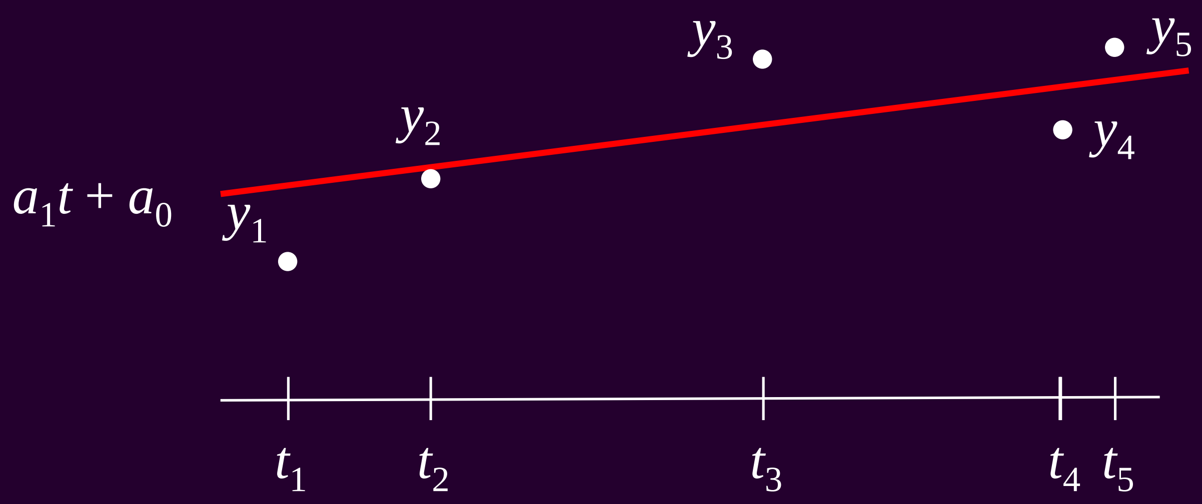
Douglas Wilhelm Harder, LEL, M.Math.
dwharder@uwaterloo.ca
dwharder@gmail.com

Introduction

- In this topic, we will
  - Discuss evaluating a least-squares best-fitting polynomial at a point
  - Describe how to find the coefficients of that polynomial
  - Look at the change in run time
    - We'll reduce the run time to O(1)!
  - Observing the differences between linear and quadratic interpolating polynomials

Review

- From the main discussion:
  - Suppose we have found a least-squares best-fitting linear polynomial passing through a set of given noisy points
    - We can thus evaluate the linear polynomial at any point on the line

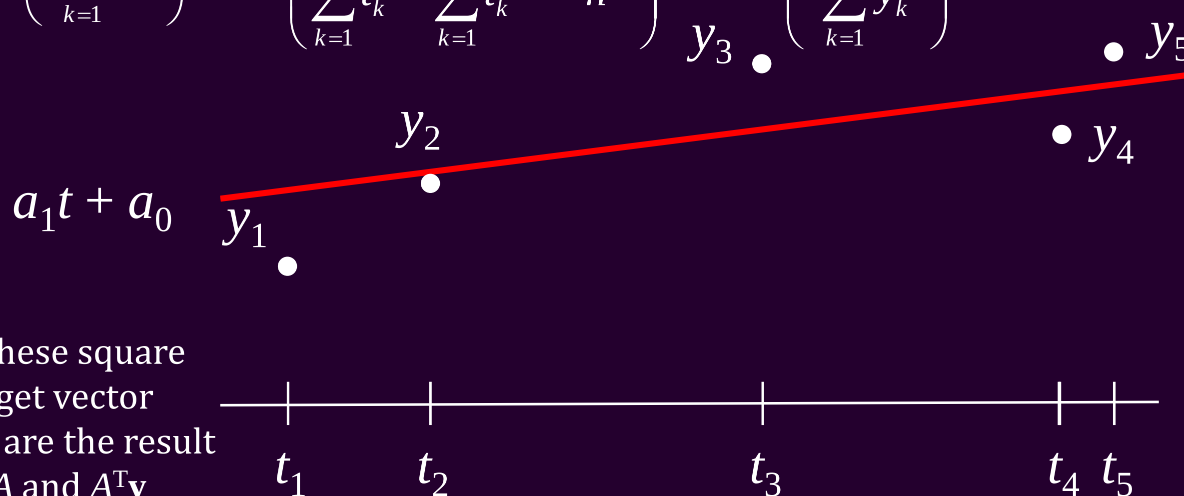


Review

- Problem:
  - Finding the least-squares best-fitting polynomial requires first calculating and then solving these systems of linear equations

$$\begin{pmatrix} \sum_{k=1}^{n} t_k^2 & \sum_{k=1}^{n} t_k \\ \sum_{k=1}^{n} t_k & n \end{pmatrix} \begin{pmatrix} a_1 \\ a_0 \end{pmatrix} = \begin{pmatrix} \sum_{k=1}^{n} t_k y_k \\ \sum_{k=1}^{n} y_k \end{pmatrix}$$

$$\begin{pmatrix} \sum_{k=1}^{n} t_k^4 & \sum_{k=1}^{n} t_k^3 & \sum_{k=1}^{n} t_k^2 \\ \sum_{k=1}^{n} t_k^3 & \sum_{k=1}^{n} t_k^2 & \sum_{k=1}^{n} t_k \\ \sum_{k=1}^{n} t_k^2 & \sum_{k=1}^{n} t_k & n \end{pmatrix} \begin{pmatrix} a_2 \\ a_1 \\ a_0 \end{pmatrix} = \begin{pmatrix} \sum_{k=1}^{n} t_k^2 y_k \\ \sum_{k=1}^{n} t_k y_k \\ \sum_{k=1}^{n} y_k \end{pmatrix}$$

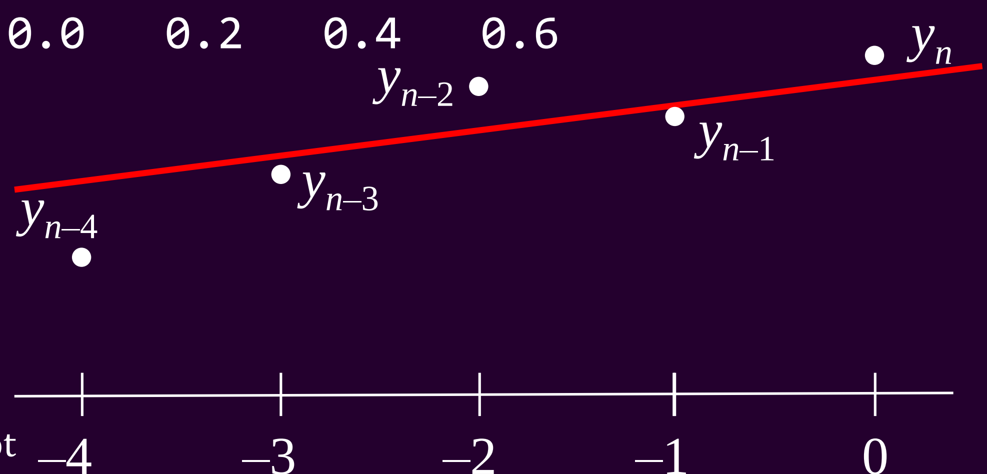


Do not memorize these square matrices or the target vector
- Understand they are the result of calculating A^TA and $A^T\mathbf{y}$

Equally spaced samples

- Fortunately, recall that data tends to be read periodically
  - Let us use the previous practice of shifting and scaling

```
>> A = vander( -4:0, 2 );
>> cond( A )
    ans = 4.738720018687270
>> inv( A'*A )*A'
    ans =
          -0.2   -0.1   0      0.1   0.2
          -0.2    0.0   0.2    0.4   0.6
```



This Matlab code is provided for demonstration purposes and is not required for the examination.

Equally spaced samples

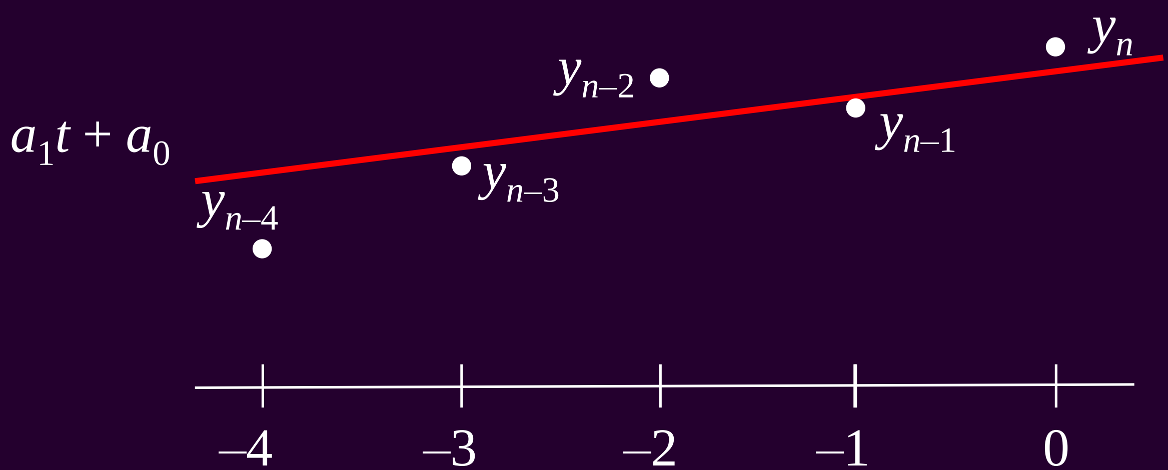
- Thus, we have that

$$\begin{pmatrix} a_1 \\ a_0 \end{pmatrix} = \left(A^{\mathrm{T}}A\right)^{-1} A^{\mathrm{T}}\mathbf{y} = \begin{pmatrix} -0.2 & -0.1 & 0 & 0.1 & 0.2 \\ -0.2 & 0 & 0.2 & 0.4 & 0.6 \end{pmatrix} \begin{pmatrix} y_{n-4} \\ y_{n-3} \\ y_{n-2} \\ y_{n-1} \\ y_n \end{pmatrix}$$

  - More simply, we have that

$$a_1 = -0.2y_{n-4} - 0.1y_{n-3} + 0.1y_{n-1} + 0.2y_n$$

$$a_0 = -0.2y_{n-4} + 0.2y_{n-2} + 0.4y_{n-1} + 0.6y_n$$

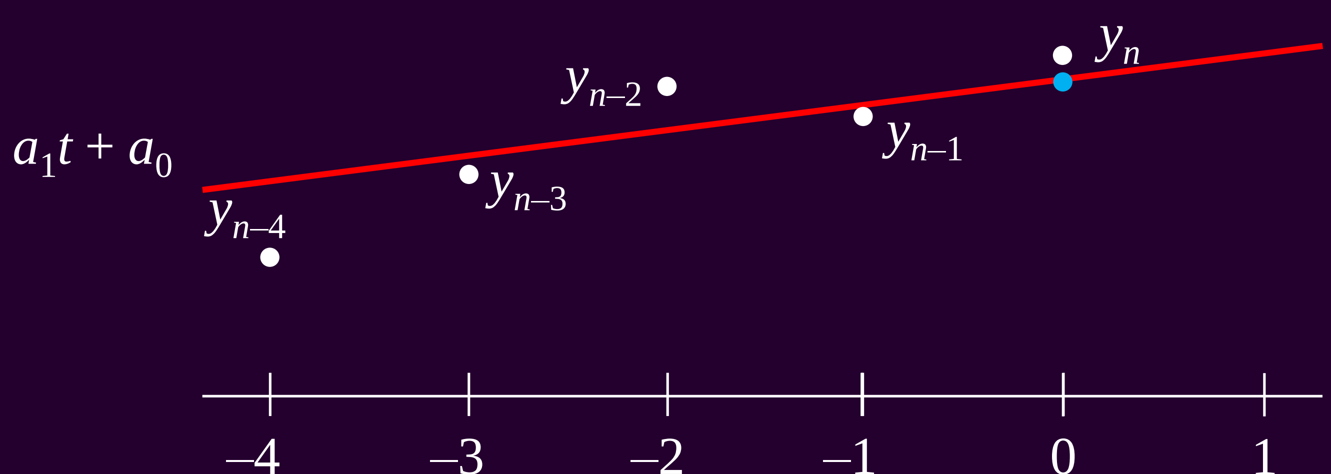


Equally spaced samples

- If the data is noisy, y_n is not even a good approximation of the current value $y(t_n)$
  - Instead, evaluate the least-squares linear polynomial at $t = 0$

    $y(t_n)$ is best approximated by a_0

$$-0.2y_{n-4} + 0.2y_{n-2} + 0.4y_{n-1} + 0.6y_n$$



Equally spaced samples

- We can also estimate the value in the future or around t_n
  - Extrapolate one step into the future by evaluating the least-squares linear polynomial at $t = 1$

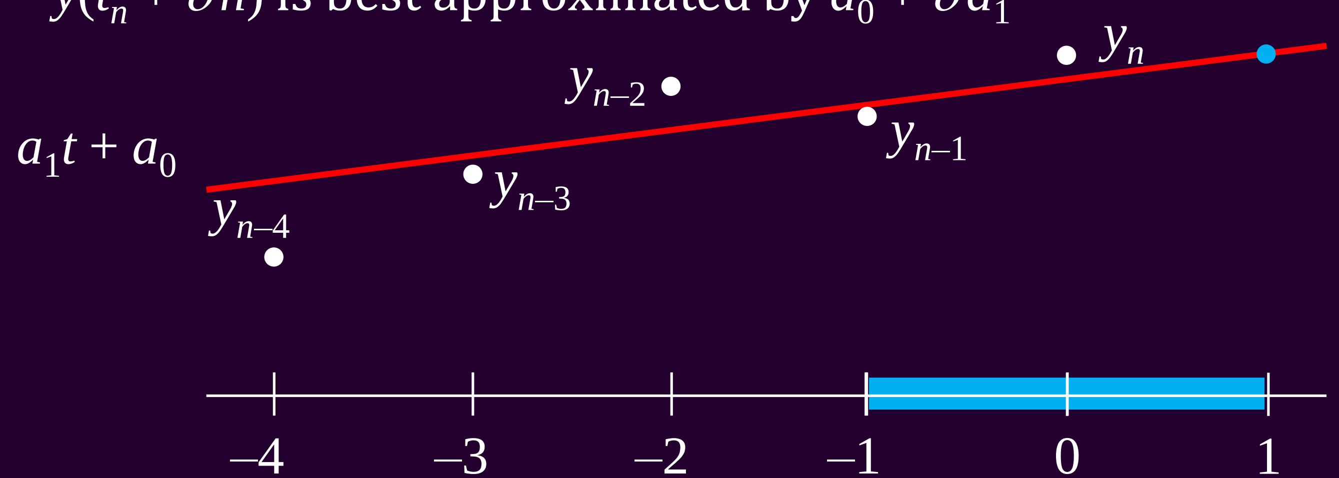
    $y(t_n + h)$ is best approximated by $a_0 + a_1$

    $$-0.4y_{n-4} - 0.1y_{n-3} + 0.2y_{n-2} + 0.5y_{n-1} + 0.8y_n$$

  - More generally, we can estimate the value at $t_n + \delta h$ by evaluating the least-squares linear polynomial at $t = \delta$

    $y(t_n + \delta h)$ is best approximated by $a_0 + \delta a_1$

$$\delta \leftarrow \frac{t - t_n}{h}$$



Equally spaced samples

- Our example uses five points
  - We could choose fewer or more points to find a least-squares line
  - In all cases, a_0 and a_1 are linear combinations of the y values

```
>> A = vander( -9:0, 2 );   # Ten points
>> inv( A'*A )*A'
   ans =
     -0.054545 -0.042424 -0.030303 -0.018182  -0.0060606 0.0060606 0.018182 0.030303 0.042424 0.054545
     -0.14545  -0.090909 -0.036364  0.018182   0.072727  0.12727   0.18182   0.23636 0.29091  0.34545
```

– Having found a_0 and a_1,
our estimators of $y(t_n)$, $y(t_n + h)$ and $y(t_n + \delta h)$ remain unchanged

This Matlab code is provided for demonstration purposes and is not required for the examination.

Equally spaced samples

- Note that because these are integer matrices, we can use some of the properties

```
>> A = vander( -9:0, 2 );   # Ten points
>> detAtA = round( det( A'*A ) )
    detA = 825
>> round( detAtA*inv( A'*A )*A' )
    ans =
       -45   -35   -25   -15    -5     5    15    25    35    45
      -120   -75   -30    15    60   105   150   195   240   285
>> ans/detAtA
    ans =
     -0.054545 -0.042424 -0.030303 -0.018182  -0.0060606 0.0060606 0.018182 0.030303 0.042424 0.054545
     -0.14545  -0.090909 -0.036364  0.018182   0.072727  0.12727   0.18182   0.23636 0.29091  0.34545
```

This Matlab code is provided for demonstration purposes and is not required for the examination.

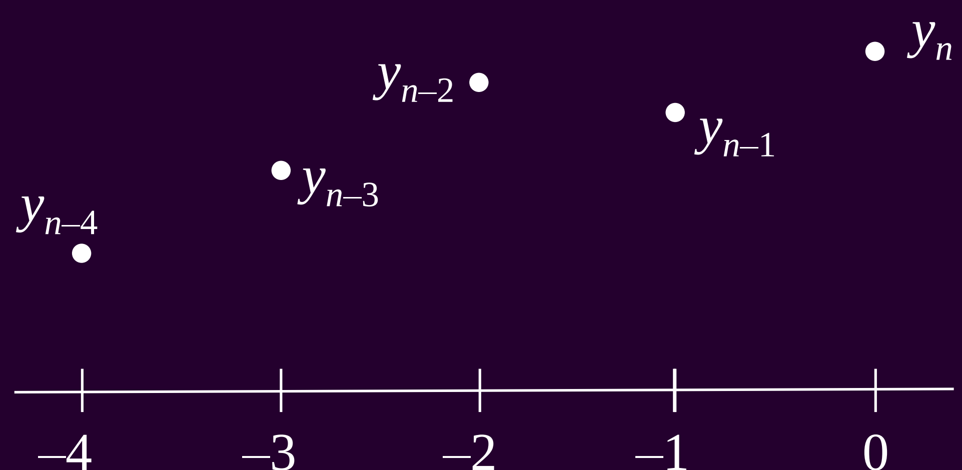
Linear or quadratic least-squares

- Consider this data from a system that is clearly accelerating
  - Using a least-squares linear polynomial would be wrong
  - We should use a least-squares quadratic polynomial

Equally spaced samples

- We can do the same for a least-squares quadratic:

$$\begin{pmatrix} a_2 \\ a_1 \\ a_0 \end{pmatrix} = \left(\left(A^{\mathrm{T}} A \right)^{-1} A^{\mathrm{T}} \right) \mathbf{y}$$



Equally spaced samples

- We can do the same for a least-squares quadratic:

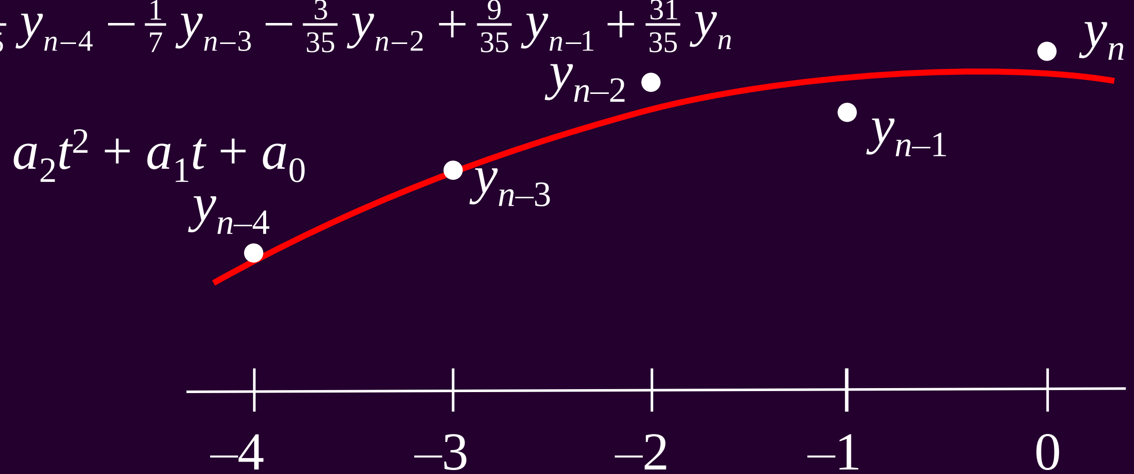
$$\begin{pmatrix} a_2 \\ a_1 \\ a_0 \end{pmatrix} = \frac{1}{\det\left(A^{\mathrm{T}}A\right)}\left(\det\left(A^{\mathrm{T}}A\right)\left(A^{\mathrm{T}}A\right)^{-1}A^{\mathrm{T}}\right)\mathbf{y} = \frac{1}{700}\begin{pmatrix} 100 & -50 & -100 & -50 & 100 \\ 260 & -270 & -400 & -130 & 540 \\ 60 & -100 & -60 & 180 & 620 \end{pmatrix}\begin{pmatrix} y_{n-4} \\ y_{n-3} \\ y_{n-2} \\ y_{n-1} \\ y_n \end{pmatrix}$$

  – More simply, we have that

$$a_2 = \tfrac{1}{7}y_{n-4} - \tfrac{1}{14}y_{n-3} - \tfrac{1}{7}y_{n-2} - \tfrac{1}{14}y_{n-1} + \tfrac{1}{7}y_n$$

$$a_1 = \tfrac{13}{35}y_{n-4} - \tfrac{27}{70}y_{n-3} - \tfrac{4}{7}y_{n-2} - \tfrac{13}{70}y_{n-1} + \tfrac{27}{35}y_n$$

$$a_0 = \tfrac{3}{35}y_{n-4} - \tfrac{1}{7}y_{n-3} - \tfrac{3}{35}y_{n-2} + \tfrac{9}{35}y_{n-1} + \tfrac{31}{35}y_n$$

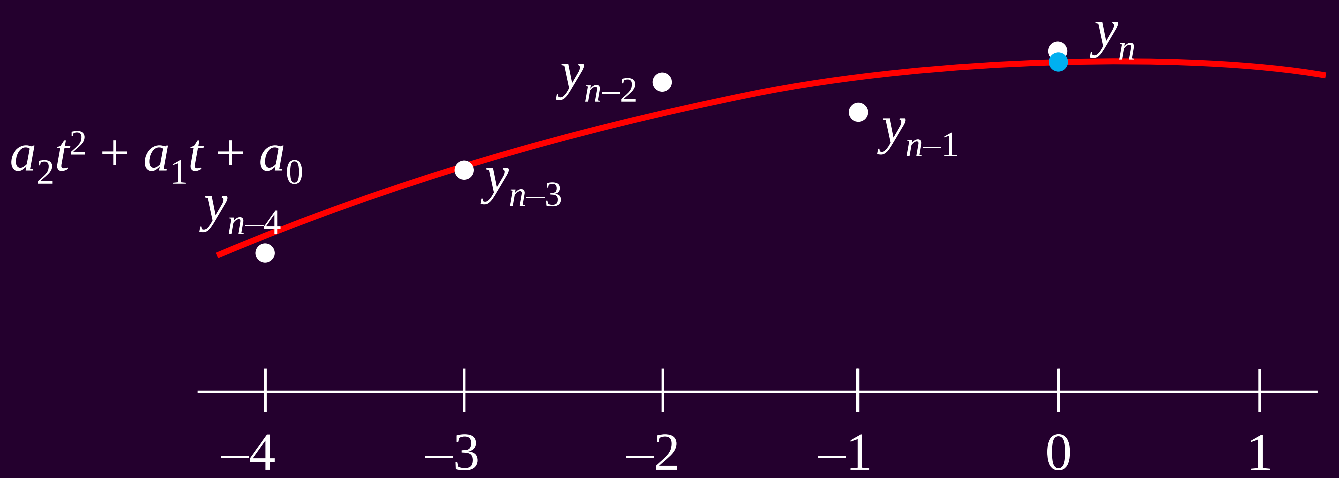


Equally spaced samples

- As before, our best approximation of the actual current value is evaluating this least-squares quadratic at $t = 0$

  $y(t_n)$ is best approximated by a_0

$$\frac{3}{35}y_{n-4} - \frac{1}{7}y_{n-3} - \frac{3}{35}y_{n-2} + \frac{9}{35}y_{n-1} + \frac{31}{35}y_n$$



Equally spaced samples

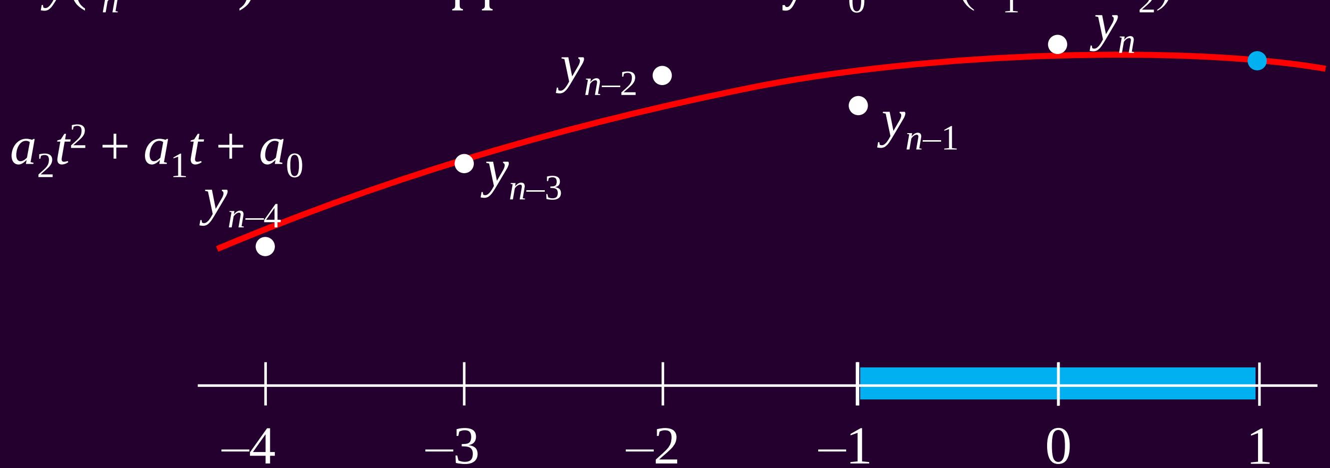
- We can also estimate the value in the future or around t_n
  - Extrapolate one step into the future by evaluating the least-squares quadratic polynomial at $t = 1$

    $y(t_n + h)$ is best approximated by $a_0 + a_1 + a_2$

    $$0.6y_{n-4} - 0.6y_{n-3} - 0.8y_{n-2} + 1.8y_n$$
  - We also estimate the value at $t_n + \delta h$ by evaluating the least-squares quadratic polynomial at $t = \delta$

    $y(t_n + \delta h)$ is best approximated by $a_0 + \delta(a_1 + \delta a_2)$

$$\delta \leftarrow \frac{t - t_n}{h}$$



O(1) run time?

- Issue:
  - This is still a single O(n) calculation with each step
  - You may note that there is a particular pattern

$$a_1 = -0.2y_{n-4} - 0.1y_{n-3} + 0.1y_{n-1} + 0.2y_n$$
$$a_0 = -0.2y_{n-4} + 0.2y_{n-2} + 0.4y_{n-1} + 0.6y_n$$

  - With the next step, the coefficients are now

$$a_1 = -0.2y_{n-3} - 0.1y_{n-2} + 0.1y_n + 0.2y_{n+1}$$
$$a_0 = -0.2y_{n-3} + 0.2y_{n-1} + 0.4y_n + 0.6y_{n+1}$$

  - Let $s \leftarrow y_{n-3} + y_{n-2} + y_{n-1} + y_n$, and so we update

$$a_1 \leftarrow a_1 + 0.2y_{n-4} - 0.1s + 0.2y_{n+1}$$
$$a_0 \leftarrow a_0 + 0.2y_{n-4} - 0.2s + 0.6y_{n+1}$$
$$s \leftarrow s - y_{n-3} + y_{n+1}$$

Summary

- Following this topic, you now
  - Understand that we can easily find formulas for least-squares best-fitting polynomials if the t-values are equally spaced
  - Are aware that with the integer matrices we defined, it is reasonable to calculate $(A^T A)^{-1} A^T$
  - Understand that this allows us to find least-squares best-fitting polynomial coefficients very quickly
  - Know that we can use these coefficients to estimate the value of the function around the current time t_n
  - Are aware that we can even do this constant run time

References

[1] https://en.wikipedia.org/wiki/Least_squares

Acknowledgments

None so far.

Colophon

These slides were prepared using the Cambria typeface. Mathematical equations use Times New Roman, and source code is presented using `Consolas`. Mathematical equations are prepared in MathType by Design Science, Inc.

Examples may be formulated and checked using Maple by Maplesoft, Inc.

The photographs of flowers and a monarch butter appearing on the title slide and accenting the top of each other slide were taken at the Royal Botanical Gardens in October of 2017 by Douglas Wilhelm Harder. Please see

https://www.rbg.ca/

for more information.

Disclaimer