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

ECE 204 *Numerical methods*

Approximating a point using least-squares best-fitting polynomials

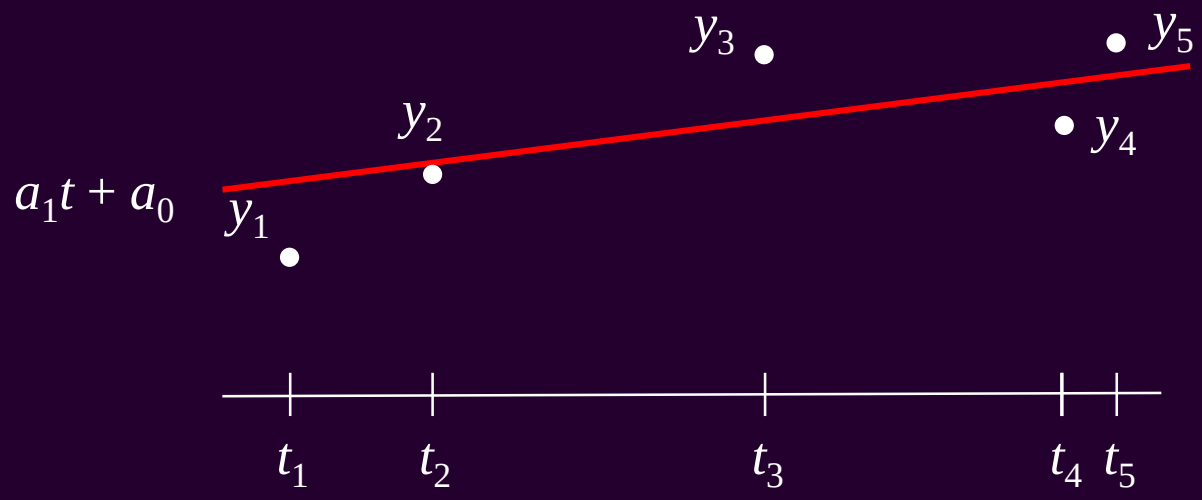
Douglas Wilhelm Harder, LEL, M.Math.
dwharder@uwaterloo.ca
dwharder@gmail.com

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

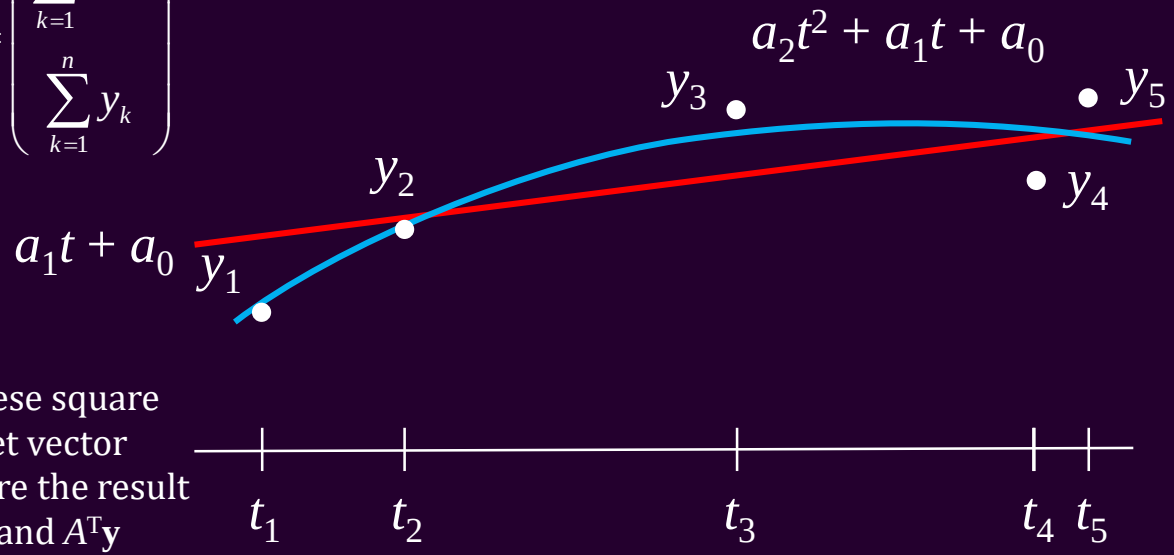


Review

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

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

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



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

- Suppose the t-values are equally spaced:

  $$t_k = t_0 + kh$$

- If the most recent reading is t_n and we are fitting the last $N + 1$ points, the t-values are $t_{n-N}, \ldots, t_n$
  - This requires new matrices and target vectors to be calculated with each time step:

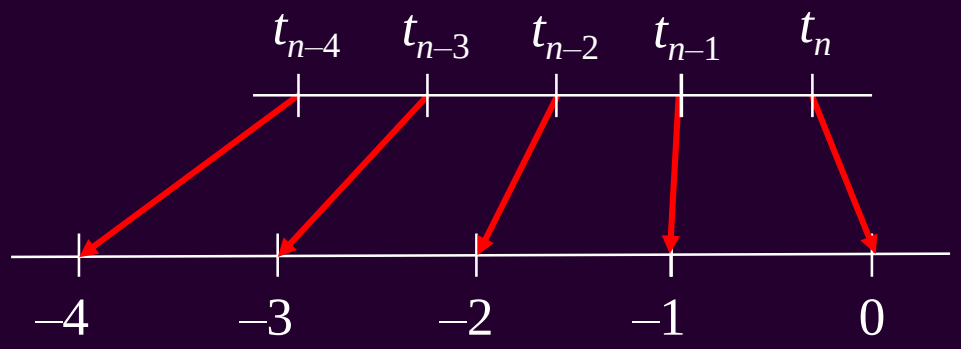
$$\begin{pmatrix} \sum_{k=0}^{N} t_{n-k}^2 & \sum_{k=0}^{N} t_{n-k} \\ \sum_{k=0}^{N} t_{n-k} & N+1 \end{pmatrix} \begin{pmatrix} a_1 \\ a_0 \end{pmatrix} = \begin{pmatrix} \sum_{k=0}^{N} t_{n-k} y_{n-k} \\ \sum_{k=0}^{N} y_{n-k} \end{pmatrix}$$

  - As t_n becomes large, the condition number grows ever larger

Equally spaced samples

- Suppose our most recent reading is t_n,
  so we are interested in the behavior around this point
- We will shift and scale time t_n to the origin with a time step of 1

$$t_n \longrightarrow 0$$
$$t_{n-1} \longrightarrow -1$$
$$\vdots$$
$$t_{n-N} \longrightarrow -N$$



- Thus, t will be mapped to $\delta \leftarrow \dfrac{t - t_n}{h}$ and specifically, $t_n + \delta h \mapsto \delta$
  - Because we are interested either in:
    - What happened in the most recent time step, between t_{n-1} and t_n , or
    - What will happen in one or time steps, between t_n and t_{n+1}

  our values our discrete time are $-1 < \delta \leq 1$

Equally spaced samples

- There are many benefits:
  - We will be evaluating least-squares best-fitting polynomials with small values of δ
  - The matrices become fixed integer matrices
    - They are only dependent on N:

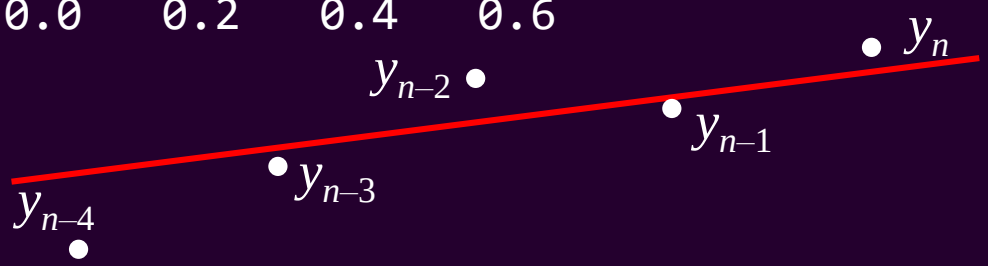
$$\begin{pmatrix} \sum_{k=0}^{N} k^2 & \sum_{k=0}^{N} -k \\ \sum_{k=0}^{N} -k & N+1 \end{pmatrix} = \begin{pmatrix} \frac{N(N+1)(2N+1)}{6} & -\frac{N(N+1)}{2} \\ -\frac{N(N+1)}{2} & N+1 \end{pmatrix}$$

$$\begin{pmatrix} \sum_{k=0}^{N} k^4 & \sum_{k=0}^{N} -k^3 & \sum_{k=0}^{N} k^2 \\ \sum_{k=0}^{N} -k^3 & \sum_{k=0}^{N} k^2 & \sum_{k=0}^{N} -k \\ \sum_{k=0}^{N} k^2 & \sum_{k=0}^{N} -k & N+1 \end{pmatrix} = \begin{pmatrix} \frac{N(N+1)(2N+1)(3N^2+3N-1)}{30} & -\frac{N^2(N+1)^2}{4} & \frac{N(N+1)(2N+1)}{6} \\ -\frac{N^2(N+1)^2}{4} & \frac{N(N+1)(2N+1)}{6} & -\frac{N(N+1)}{2} \\ \frac{N(N+1)(2N+1)}{6} & -\frac{N(N+1)}{2} & N+1 \end{pmatrix}$$

Equally spaced samples

- Consider this example with $N = 4$:

```
>> N = 4;
>> A = vander( -N: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

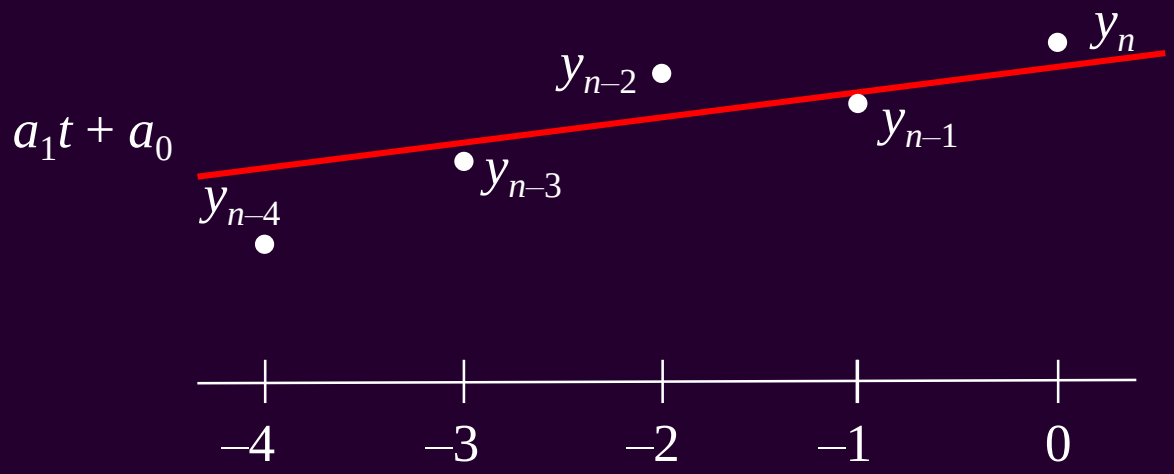
- Thus, we have that

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

  - More simply, we have that

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

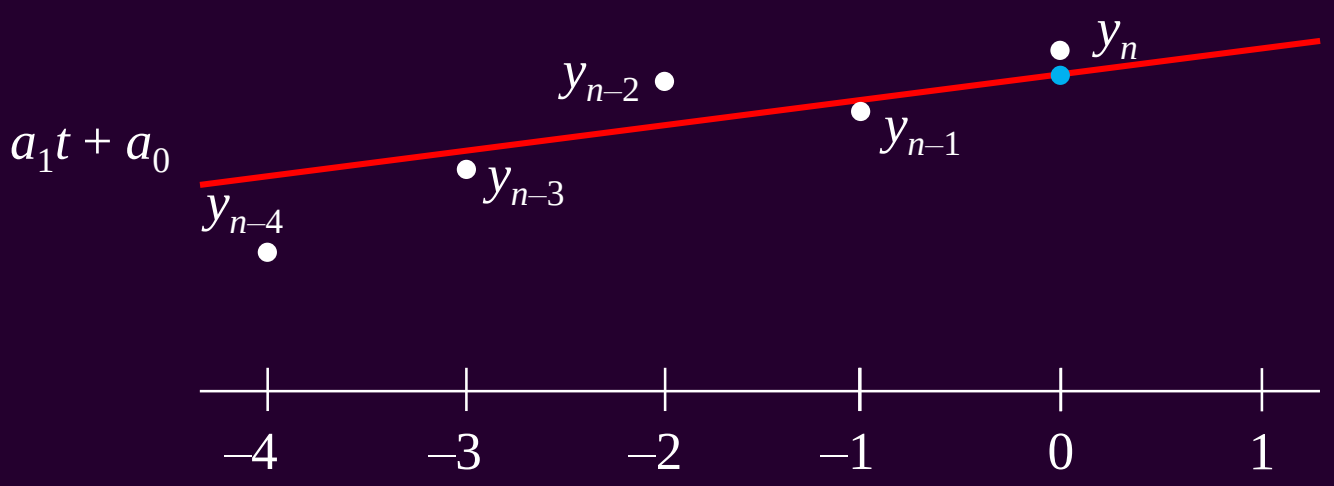
$$a_0 = -0.2y_{n-4} + 0.2y_{n-2} + 0.4y_{n-1} + 0.6y_n$$



Equally spaced samples

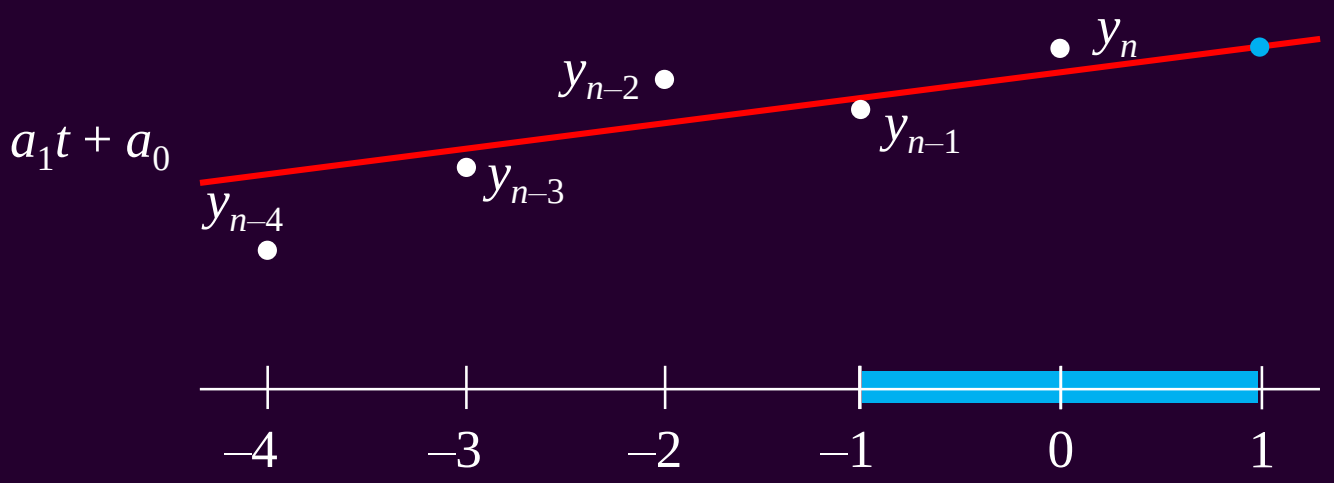
- If the data is noisy, y_n is not even a good approximation of the current value $y(t_n)$
  - Instead, evaluate the least-squares linear polynomial at $t = 0$, so $y(t_n)$ is best approximated by a_0

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



Equally spaced samples

- We can also estimate the value around t_n:
  - Given $t_{n-1} < t \le t_{n+1}$, then if $\delta \leftarrow \frac{t - t_n}{h}$, then $-1 < \delta \le 1$
    - Recall that least-squares allows us to extrapolate into the future
  - Then, $y(t_n + \delta h)$ is best approximated by $a_0 + a_1\delta$
  - Specifically, $y(t_{n+1})$ is approximated by $a_0 + a_1$



Equally spaced samples

- Finding the coefficients is always fast:
  - These can be pre-calculated and programmed into the algorithms
  - In all cases, a_0 and a_1 are linear combinations of the y values

```
>> N = 8;                       # Nine points
>> A = vander( -N:0, 2 );
>> inv( A'*A )*A'               # Solving A'*A*a = A'*y, calculate
   ans =
      -0.06667  -0.05000  -0.03333  -0.01667   0.00000   0.01667   0.03333   0.05000   0.06667
      -0.15556  -0.08889  -0.02222   0.04444   0.11111   0.17778   0.24444   0.31111   0.37778

>> N = 9;                       # Ten points
>> A = vander( -N:0, 2 );
>> inv( A'*A )*A'               # Solving A'*A*a = A'*y, calculate
   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.

Equally spaced samples

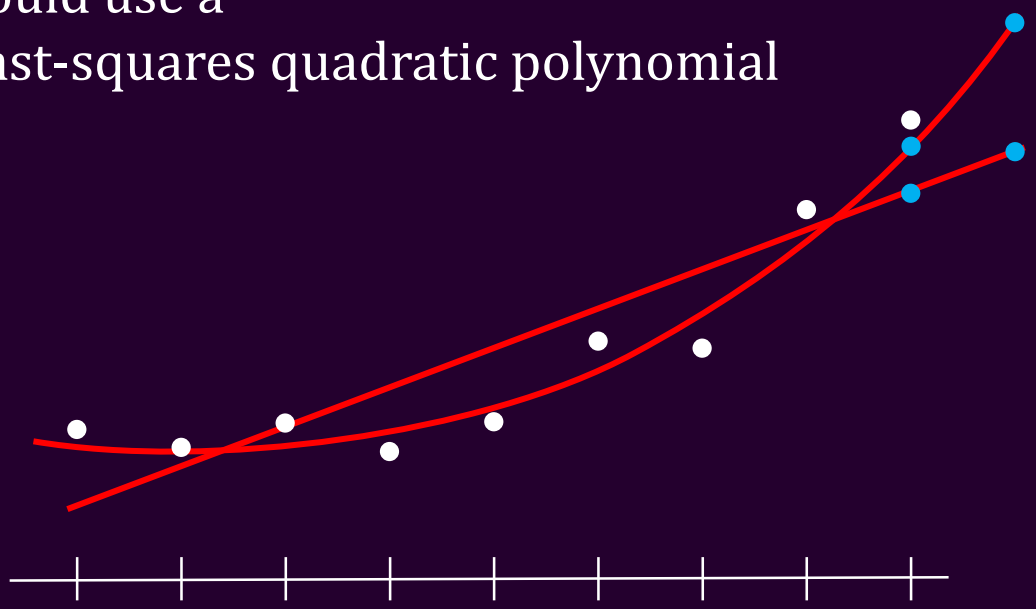
- Note that because these are integer matrices,
  we can use some of the properties of such matrices

```
>> N = 9;
>> A = vander( -N: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

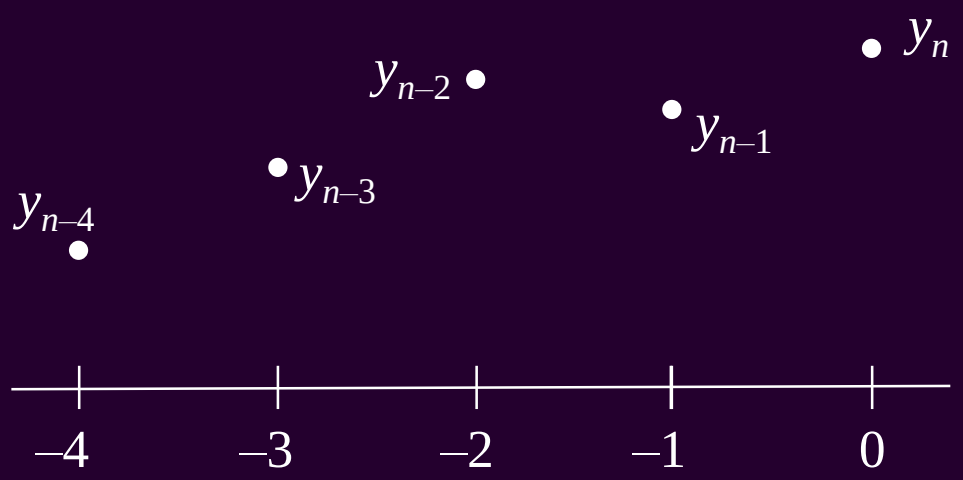
- Consider this data from a system that is clearly accelerating
  - Using a least-squares linear polynomial would be wrong
  - We should use a
    least-squares quadratic polynomial



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

This exemplifies an idea, it is not required for this course.

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

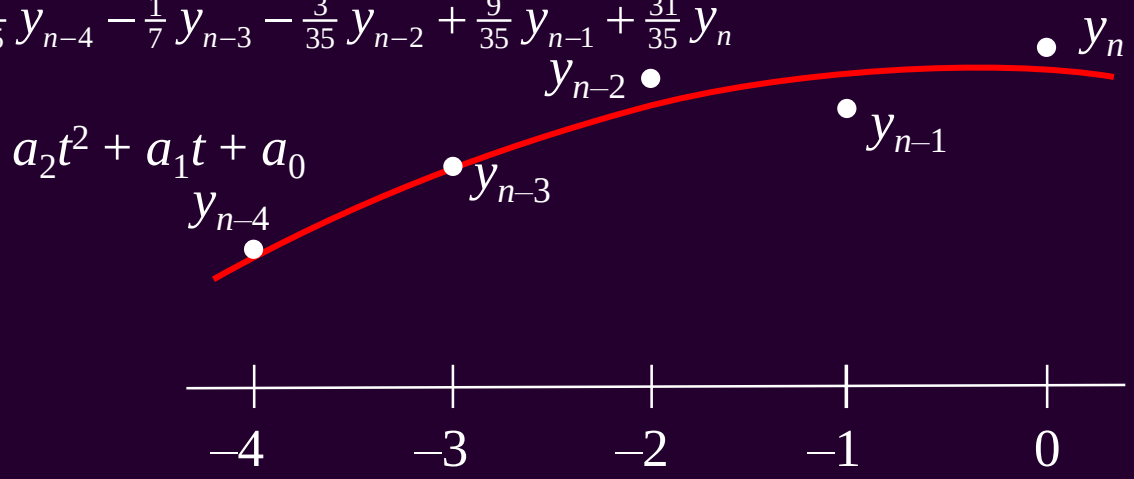
$$\begin{pmatrix} a_2 \\ a_1 \\ a_0 \end{pmatrix} = \frac{1}{\det\left(A^{\mathrm{T}}A\right)}\left(\det\left(A^{\mathrm{T}}A\right)\left(A^{\mathrm{T}}A\right)^{-1}A^{\mathrm{T}}\right)\mathbf{y} = \frac{1}{700}\begin{pmatrix} 100 & -50 & -100 & -50 & 100 \\ 260 & -270 & -400 & -130 & 540 \\ 60 & -100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

  - More simply, we have that

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

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

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

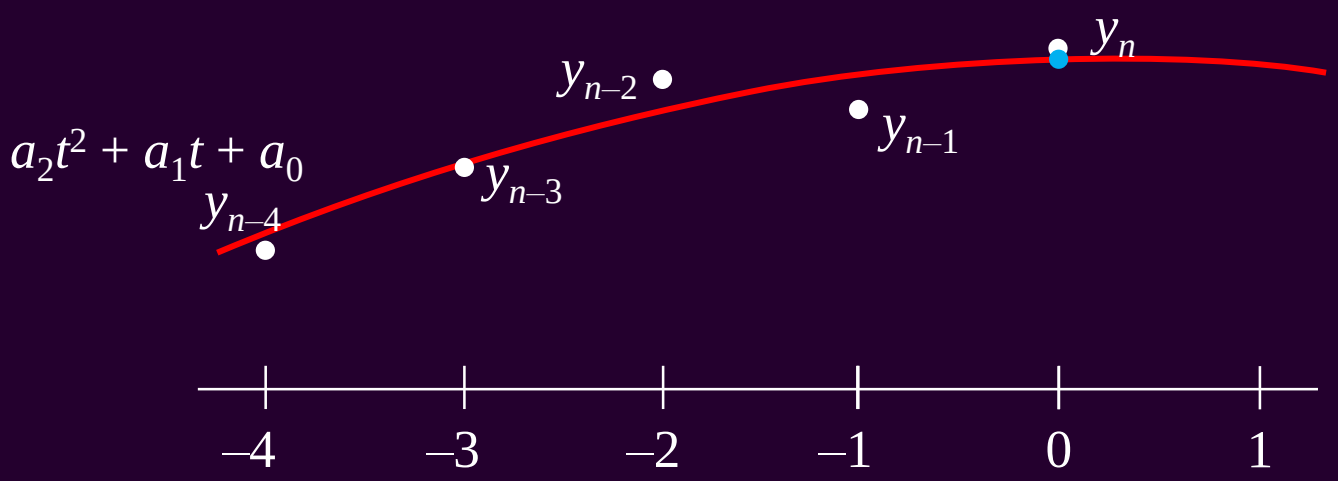


Equally spaced samples

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

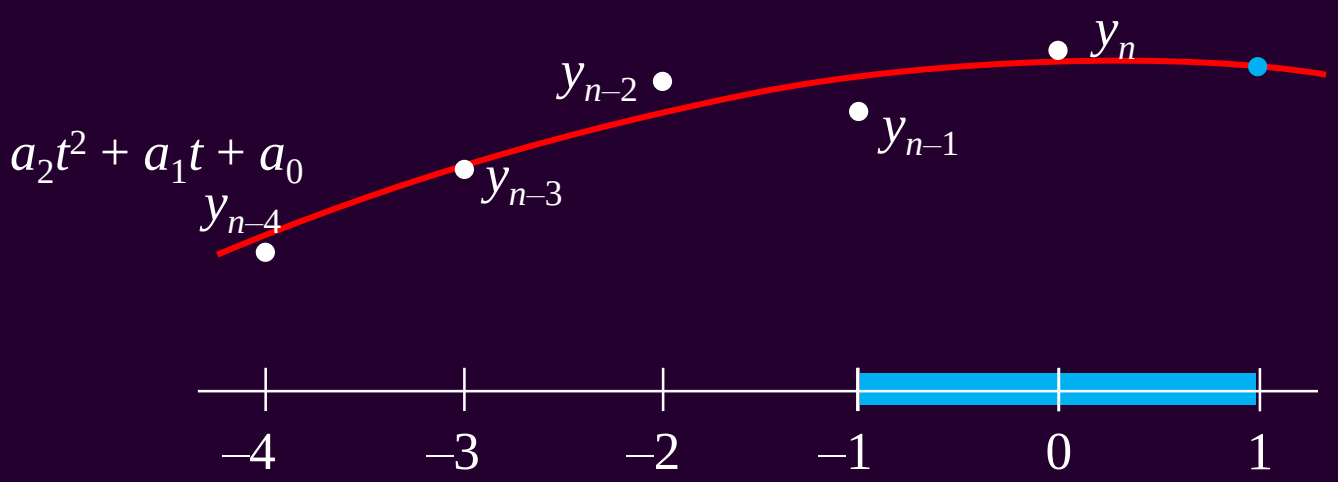
  $y(t_n)$ is best approximated by a_0

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



Equally spaced samples

- Again, we can estimate the value around t_n:
  - Given $t_{n-1} < t \le t_{n+1}$, then if $\delta \leftarrow \frac{t - t_n}{h}$, then $-1 < \delta \le 1$
    - Recall that least-squares allows us to extrapolate into the future
  - Then, $y(t_n + \delta h)$ is best approximated by $(a_2\delta + a_1)\delta + a_0$
  - Specifically, $y(t_{n+1})$ is approximated by $a_2 + a_1 + a_0$



O(1) run time?

This exemplifies an idea,
it is not required for this course.

- 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 in O(1) time:

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

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

$$s \leftarrow s - y_{n-3} + y_{n+1}$$

O(1) run time?

This exemplifies an idea,
it is not required for this course.

- Everything in this class runs in O(1) time with O(*n*) memory:

```
class Estimate {
    public:
        Estimate( double y0 );
        double operator()( double delta ) const;
        void next( double y );
    private:
        std::size_t curr_;
        double      ys_[4];
        double      a1_, a0_, sum_;
};

Estimate::Estimate( double y0 ):
curr_{ 0 },
ys_{ y0, y0, y0, y0 },
a1_{ 0.0 },
a0_{ y0 },
sum_ { 4*y0 } {
    // Empty
}
```

```
void Estimate::next( double y ) {
    curr_ = (curr_ + 1)%4;
    a1_ = a1_ + 0.2*ys_[curr_] - 0.1*sum_ + 0.2*y;
    a0_ = a0_ + 0.2*ys_[curr_] - 0.2*sum_ + 0.6*y;
    sum_ += y - ys_[curr_];
    ys_[curr_] = y;
}

double Estimate::operator()( double delta ) const {
    return a1_*delta + a0_;
}
```

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 $(A^{\mathrm{T}}A)^{-1}A^{\mathrm{T}}$
  - 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

These slides are provided for the ECE 204 *Numerical methods* course taught at the University of Waterloo. The material in it reflects the author's best judgment in light of the information available to them at the time of preparation. Any reliance on these course slides by any party for any other purpose are the responsibility of such parties. The authors accept no responsibility for damages, if any, suffered by any party as a result of decisions made or actions based on these course slides for any other purpose than that for which it was intended.