Interpolation and Root-Finding Algorithms

Alan Grant and Dustin McAfee

THE UNIVERSITY OF TENNESSEE
KNOXVILLE

Questions

1. What is the rate of convergence of the Bisection Method?

1. What condition makes Newton’s Method fail?

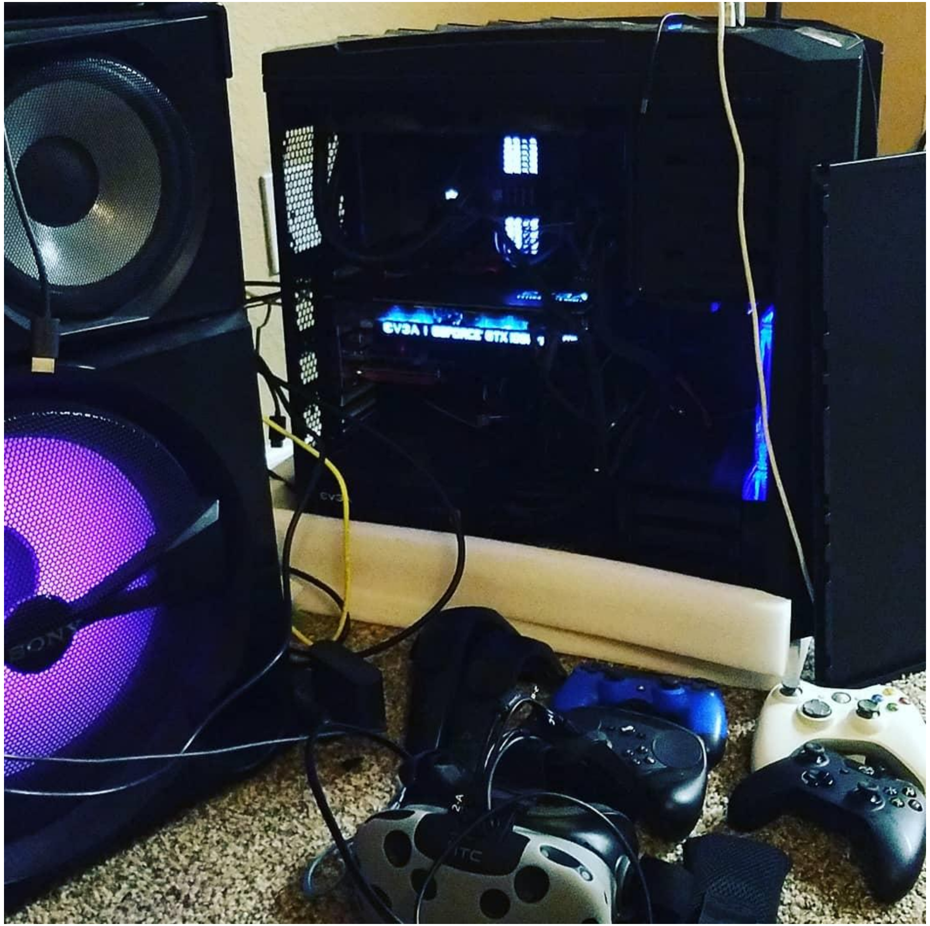
1. What civilization is the earliest known to have used some form of interpolation?

About Me: Dustin McAfee

- Hometown: Maryville, Tennessee
  - Great Smoky Mountains National Park
  - Maryville College
- B.A. Mathematics and Computer Science
- 1 Year at ORAU with FDA at White Oak.
  - Mostly Software Engineering and HPC
  - Some Signal Processing
- PhD. Candidate Computer Science emphasis HPC
  - Modifying the input event system in the Android Kernel for efficient power usage.
- Hobbies:
  - Longboarding, Live Music, PC Gaming, Cryptocurrency Markets.

My Specs:

- 7th gen 4-core 4.2GHz i7
- 64GB 3200MHz DDR4 RAM
- 8TB 7200 RPM 6.0Gb/s HDD
- 512GB 3D V-NAND SSD
- GTX 1080ti 11GB GDDR5X 11Gb/s
- 2400 Watt sound system
- HTC Vive with 5GHz wireless attachment

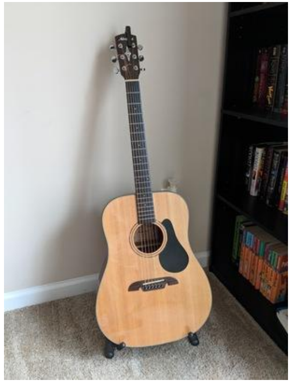


About Me: Alan Grant

- B.Sc. in CS from Indiana University Southeast
  - Research: Machine learning and image processing
- First year Ph.D. student
  - Adviser: Dr. Donatello Materassi
  - Research Areas: Graphical Models and Stochastic Systems

About Me: Alan Grant

- From New Pekin, IN
- Worked as a fabricator and machinist
- Hobbies
    - Reading, music, brewing beer, gaming
    - Recently wine making and photography

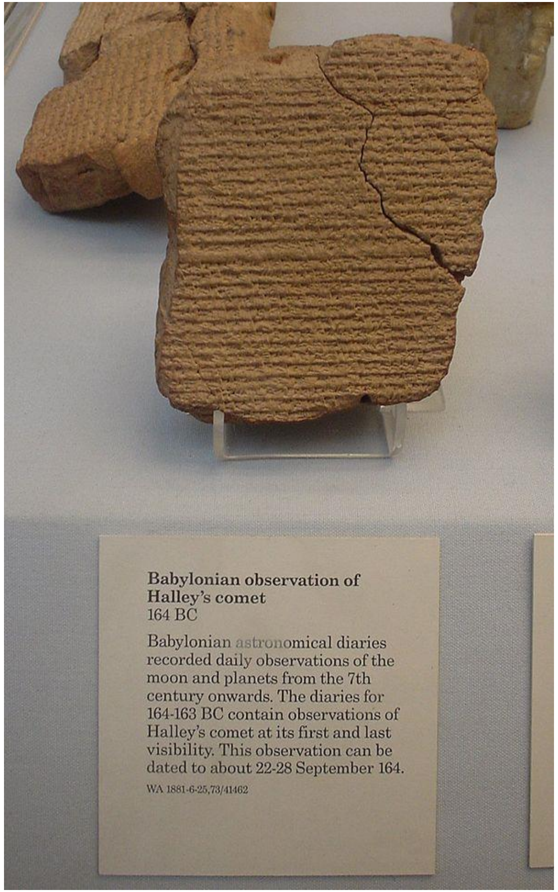


Overview

- Brief History
- Algorithms
- Implementations
  - Failures
  - Results
- Applications
- Open Issues
- References
- Test Questions

Brief History

- Earliest evidence are Babylonian ephemerides
  - Used by farmers to predict when to plant crops
  - Linear interpolation
- Greeks also used linear interpolation
  - Hipparchus of Rhodes' Table of Chords
  - A few centuries later Ptolemy created a more detailed version



Brief History

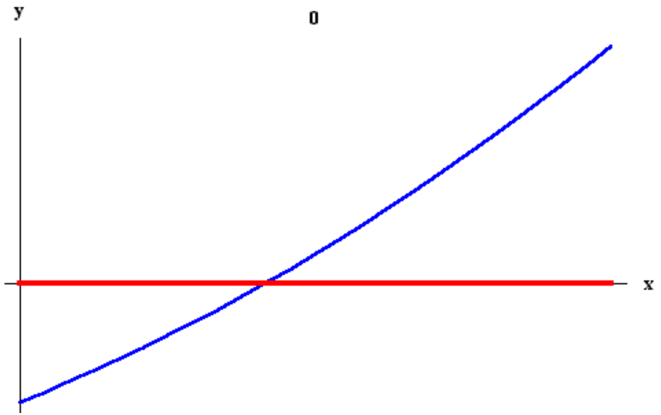
- Early Medieval Period
    - First evidence of higher-order interpolation
- Developed by astronomers to track celestial bodies
    - Second-order interpolation
        - Chinese astronomer Liu Zhuo
        - Indian astronomer Brahmagupta
- Later astronomers in both regions developed methods beyond second-order.

Brief History

- Scientific Revolution
- Calculus is cool
    - Allowed for a multitude of root finding algorithms and higher-order interpolation algorithms.
    - Newton-Rhapson method, Lagrange Polynomials, Müller’s Method

Bisection Method

- Suppose f is a continuous function defined on the interval [a, b], with f(a) & f(b) of opposite sign. By the Intermediate Value Theorem, there exists a p in (a, b), such that $f(p) = 0$.
  - Though this works when there are multiple roots, we assume that the root is unique.



Bisection Method

- The method: Halve the interval [a, b], locate the half containing the root. Repeat. Stop when within tolerance.
  - Set p = (a + b) / 2. If f(p) within tolerance, Stop.
  - Else, if f(p) and f(a) (or f(b)) have different signs, then the root is in (a, p) (or (p, b)).Converges linearly at a rate of ½.
  - Converges linearly at a rate of ½.
  - Then at each iteration, n, the error is $(b-a)(2^{-n})$.

Newton's Method

- Consider the first order Taylor Polynomial of f, and let x=p be the root. We create a Taylor Series about a point p_0, which we choose to be 'close' to p.

$$f(x) = f(p_0) + f^1(p_0)(x - p_0) + \frac{f^2(\xi(x))(x - p_0)^2}{2}$$

$$f(x) = 0 = f(p_0) + f^1(p_0)(p - p_0) + \frac{f^2(\xi(p))(p - p_0)^2}{2}$$

$$\frac{f^2(\xi(p))(p - p_0)^2}{2}$$

$$0 \approx f(p_0) + f^1(p_0)(p - p_0)$$

$$(p - p_0) \approx -(f(p_0)/f^1(p_0))$$

$$p \approx p_0 - (f(p_0)/f^1(p_0)) = p_1$$

$$p_n = p_{n-1} - f(p_{n-1})/f^1(p_{n-1})$$

- Since the error term is of $O((p-p_0)^2)$, Newton's Method converges quadratically with simple roots.

Newton's Method

- Using the root of the Tangent line to approximate the root of the function.
- Start with p_n and find the equation of the tangent line to f at $(p_n, f(p_n))$ using point slope form.

$$y - f(p_n) = f^1(p_n)(x - p_n)$$

$$y = f(p_n) + f^1(p_n)(x - p_n)$$

  - p_{n+1} (the root) is the x-intercept of the line, so plugging in p_{n+1} for x, we solve for p_{n+1}

$$0 = f(p_n) + f^1(p_n)(p_{n+1} - p_n)$$

$$p_{n+1} = p_n - f(p_n)/f^1(p_n)$$

- Does not work (fails) when $f'(p_0)$ is close to 0.

Fixed Point Problems

- Fixed point problem: $g(p) = p \rightarrow p - g(p) = 0$.
- Root finding problem: $f(p) = 0$.
- $g(x) = x - f(x)$.
- $\rightarrow g'(x) = 1 - f'(x)$, however this will not necessarily give us the root.
- Instead use function $\phi(x)$, such that $g(x) = x - \phi(x)f(x)$.

$$g^1(x) = 1 - (\Phi(x)f^1(x) + \Phi^1(x)f(x))$$
$$g^1(p) = 0 = 1 - \Phi(p)f^1(p)$$
$$\rightarrow \Phi(p) = f^1(p)^{-1}$$
$$\rightarrow g(x) = x - \frac{f(x)}{f^1(x)}$$

- This is the Newton's Method.

Multiplicity

- What if f'(p) = 0?
- If a root p has multiplicity of one (simple root), then the function crosses the root at p exactly once. Otherwise, the function "bounces off" the x-axis, and touches the root M times. We call M the multiplicity.
- If a root p of f(x) has multiplicity M, then we can define a related function, f(x), to be:
  - $f(x) = (x - p)^M q(x)$ for some polynomial q(x) such that $q(p) \neq 0$.
- Without further ugly derivations, We define:

$$\mu(x) = \frac{f(x)}{f^1(x)} = \frac{(x-p)q(x)}{(Mq(x) + (x-p)q^1(x))}$$

  - → μ(p) has multiplicity one → We can apply Newton's Method!

$$g(x) = x - \frac{\mu(x)}{\mu^1(x)} = x - \frac{\frac{f(x)}{f^1(x)}}{(f^1(x))^2 - f(x)f^2(x)}$$

False Position Method

- Newton's method uses tangent lines to approximate roots.
- Secant method is based on newton's method, and uses the secant of two points of opposite sign to approximate the root.
- Method of False position:
  - Start with $(p_0, f(p_0))$ and $(p_1, f(p_1))$ so that $f(p_0)f(p_1) < 0$.
  - Find the secant line. Assign p_2 = the root of the secant line.
  - If $f(p_1)f(p_2) < 0$, replace $p_0 = p_2$
  - Otherwise, $p_1 = p_2$.
  - Repeat
- Always converges. Linear convergence.

Müller's Method

- Use a 2nd order function to approximate our f(x) whose roots we are looking for.
- Input: 3 points p_0, p_1, and p_2,the tolerance TOL, and the max number of iterations N_0.
- Output: approximate solution p or message of failure.
- $P(x) = a(x - p_2)^2 + b(x - p_2) + c$
  - $f(p_0) = a(p_0 - p_2)^2 + b(p_0 - p_2) + c$
  - $f(p_1) = a(p_1 - p_2)^2 + b(p_1 - p_2) + c$
  - $f(p_2) = c$
- Solve for a and b to find the 2nd order polynomial approximation.

Müller's Method

- Apply the quadratic formula to find the next approximation, p_3.
  - $p_3 - p_2 = (-2c)/(b \pm \sqrt{}(b^2 - 4ac))$
  - $p_3 = p_2 - (2c)/(b \pm \sqrt{}(b^2 - 4ac))$
  - To avoid round off errors due to subtraction of nearly equal numbers, we use the larger determinant:
  - $p_3 = p_2 - (2c)/(b+\mathrm{sign}(b)\sqrt{}(b^2 - 4ac))$
- Replace $p_0 = p_1$; $p_1 = p_2$; $p_2 = p_3$.
- Repeat. Great for polynomials, Can find complex roots.

Lagrange Polynomials

- Given N + 1 data points and a function whose values are given at these numbers, then a unique polynomial P(x) of degree at most N exists with f(x) = P(x) at the given data points.
- This polynomial is:

$$P(x) = \sum_{k=0}^{N} f(x_k) L_{N,k}(x)$$

$$L_{N,k} = \prod_{i=0, i \neq k}^{N} \frac{x - x_i}{x_k - x_i}$$

- implemented in:
  - Neville’s Method
  - Hermite Polynomials

Neville's Method

- Recursive method for Lagrange Polynomials.
- Generates successive approximations to a function at a specific value.
- Error is estimated by the difference between successive approximations.
- Uses the recurrence relation:

$$P(x) = \frac{(x - x_j)P_{0,1,\ldots,j-1,j+1,\ldots,N}(x) - (x - x_i)P_{0,1,\ldots,i-1,i+1,\ldots,N}(x)}{x_i - x_j}$$

- $O(N^2)$

Hermite Polynomials

- The Taylor Polynomial matches a function at one value for n derivatives.
- The Lagrange Polynomial matches a function at n values.
- The Hermite Polynomial matches the function values and and a certain amount of derivatives at N points.
- Definition: Let $x_0, x_1, \ldots, x_N$ be N + 1 distinct numbers in [a, b] and f is continuously differentiable in [a, b], then the Hermite is a Polynomial of degree at most 2N + 1 such that:

$$H_{2N+1}(x) = \sum_{j=0}^{N} f(x_j) H_{N,j}(x) + \sum_{j=0}^{N} f^1(x_j) \hat{H}_{N,j}(x)$$

$$f(x_i) = H_{2N+1}(x_i)$$

$$f^1(x_i) = H^1_{2N+1}(x_i)$$

$$H_{N,j}(x) = [1 - 2(x - x_j) L^1_{N,j}(x_j)](L_{N,j}(x))^2$$

$$\hat{H}_{N,j}(x) = (x - x_j)(L_{N,j}(x))^2$$

Forward Divided Differences

- AKA Newton’s Method
- There is a backwards version, but focus is on this version.
- Define the polynomial P(x) to be:

$$P_N(x) = a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1) + \cdots + a_n(x - x_0)\cdots(x - x_{N-1})$$

$$a_0 = P_N(x_0) = f(x_0)$$

$$a_1 = \frac{f(x_1) - f(x_0)}{x_1 - x_0} = f[x_0, x_1]$$

$$a_k = f[x_0, x_1, \cdots, x_k]$$

$$P_N(x) = f[x_0] + \sum_{k=1}^{N} f[x_0, x_1, \cdots, x_k](x - x_0)(x - x_1)\cdots(x - x_{k-1})$$

Piecewise Splines

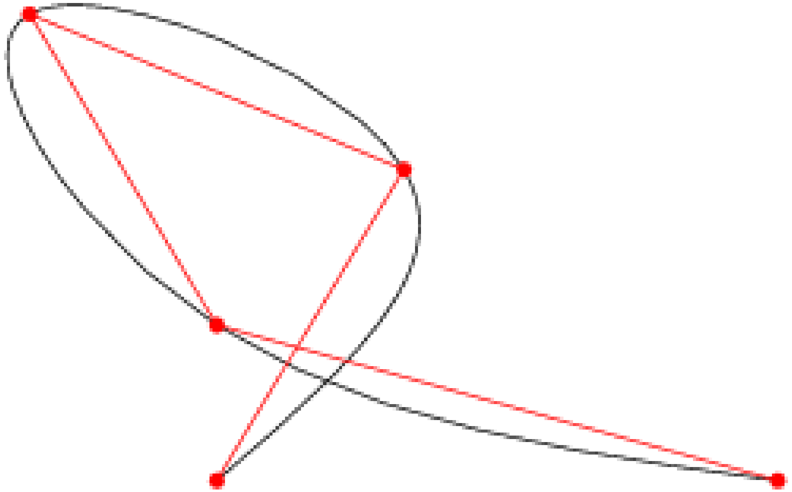
- Split the dataset up into pieces.
- Interpolate the separate partitions such that the function values and derivative values agree at the nodes, and that the second derivative is continuous.

- Very widely used technique:
  - Cubic Splines
  - Used in Modern Software Automated ECG Analysis

Cubic Splines

- Pick n data points: x_0, x_1, … , x_n such that $S_0(x)$ is in between x_0 and x_1, $S_1(x)$ is in between x_1 and x_2 … etc.
- S(x) is a piecewise cubic polynomial denoted $S_j(x)$ on the subinterval [x_j, x_{j+1}] for each j = 1, 0, … , n - 1.

$$S_j(x_j) = f(x_j)$$
$$S_j(x_{j+1}) = f(x_{j+1}) = S_{j+1}(x_{j+1})$$
$$S^1_{j+1}(x_{j+1}) = S^1_j(x_{j+1})$$
$$S^2_{j+1}(x_{j+1}) = S^2_j(x_{j+1})$$



Failures

- Serial and parallel implementations of Hermite Polynomial interpolation.
    - Look at speed up for randomized sets of *N* data points.
- Serial implementation is trivial.
    - Done using Divided Differences Method
    - Tried simply parallelizing this version.
- Tried implementing parallel versions from a few papers.
- CUDA implementations exist.

Implementations

- f(x) = x - cos(x)
- df/dx = 1 + sin(x)
  - p1 = 0,
  - p2 = 0.5 (for Muller's Method)
  - p3 = 1
  - Tolerance = 10e-15
- Use Bisection Method, Secant Method, False Position, Muller's Method, and Newton's Method to find the root in between 0 and 1.
- Compare number of iterations.

Results: Bisection Method

Iteration	a	b	p	error
1	0	1	0.5	0.5
5	0.6875	0.75	0.71875	0.03125
10	0.738281	0.740324	0.739258	0.000976562
20	0.739084	0.739086	0.739085	9.53674e-07
30	0.739085	0.739085	0.739085	9.31323e-10
40	0.739085	0.739085	0.739085	9.09495e-13
50	0.739085	0.739085	0.739085	8.88178e-16

Results: Secant Method

Iteration	a	b	p	error
1	0	1	0.685073	0.314927
2	1	0.685073	0.736299	0.0512256
3	0.685073	0.736299	0.739119	0.00282039
4	0.736899	0.739119	0.739085	3.42498e-05
5	0.739119	0.739085	0.739085	2.10875e-08
6	0.739085	0.739085	0.739085	1.59428e-13
7	0.739085	0.739085	0.739085	0

Results: False Position

Iteration	a	b	p	error
1	0	1	0.685073	0.314927
2	1	0.685073	0.736299	0.0512256
3	1	0.736299	0.739119	0.00264636
4	1	0.738945	0.739078	0.000132775
5	1	0.739078	0.739085	6.65157e-06
10	1	0.739085	0.739085	2.09799e-12
13	1	0.739085	0.739085	3.33067e-16

Results: Muller's Method

Iteration	p0	p1	p2	p	error
1	0	0.5	1	0.741502	0.258498
2	0.5	1	0.741502	0.739075	0.00242686
3	1	0.741502	0.739075	0.739085	1.01903e-05
4	0.741502	0.739075	0.739085	0.739085	4.60615e-10
5	0.739075	0.739085	0.739085	0.739085	0

Results: Newton's Method

Iteration	a	p	error
1	0	1	1
2	1	0.750364	0.249636
3	0.750364	0.739113	0.011251
4	0.739113	0.739085	2.77575e-05
5	0.739085	0.739085	1.70123e-10
6	0.739085	0.739085	0

Applications

- Many signal processing applications:
  - Electrocardiogram analysis
  - Upsampling

- Physical problems:
  - Zero points in motion or momentum
  - Area minimization problems

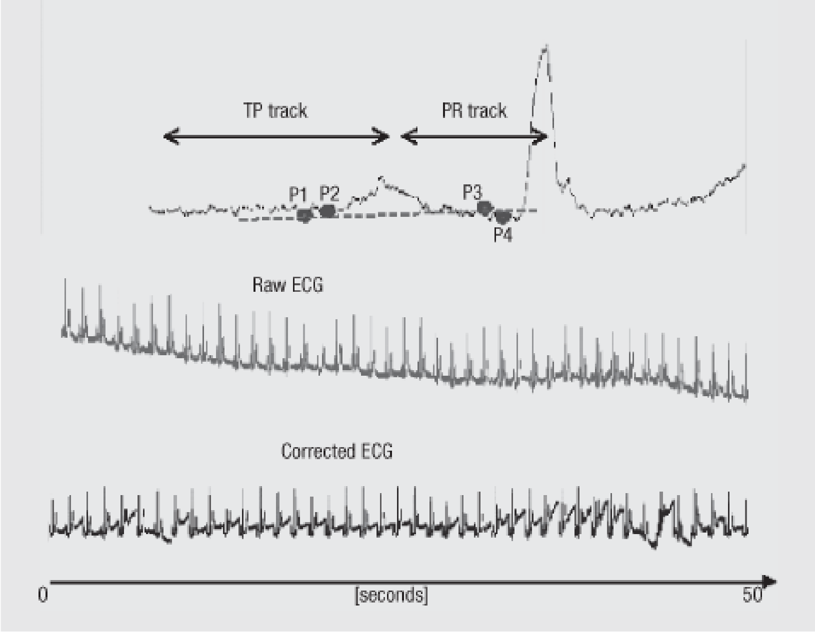
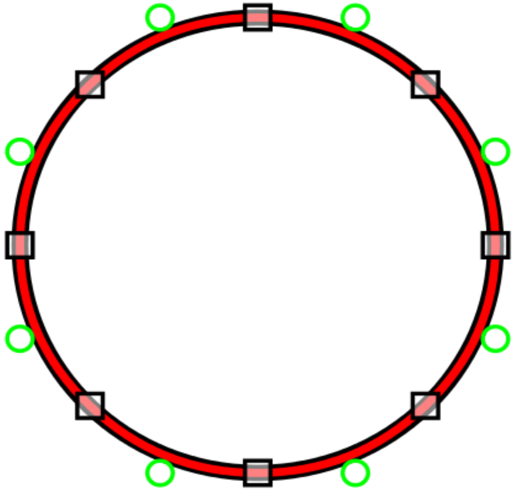


Fig. 3 | *Beat-to-beat linear piecewise interpolation (upper panel) and its effect of entire ECG (lower panel).*

Applications



- True Type Fonts
  - Outline information stored as straight lines and bezier curve segments.
- Curve sections are rendered by finding the roots with respect to horizontal rays.

Open Problems

- Most open problems are more based in the mathematics of interpolation or approximation theory
- Algorithms side is mostly about improving existing algorithms
  - Improving accuracy with less data
  - Better parallelization methods

References

Thomas H. Cormen, Charles Eric. Leiserson, Ronald L. Rivest, and Clifford Stein. 2009. *Introduction to Algorithms*3rd ed., Cambridge, MA: MIT Press.

Ö. Eğecioğlu, E. Gallopoulos, and Ç.K. Koç. 1989. Parallel Hermite interpolation: An algebraic approach. *Computing*42, 4 (December 1989), 291–307. DOI:http://dx.doi.org/10.1007/bf02243225

Ömer Eğecioğlu, E. Gallopoulos, and Çetin K. Koç. 1989. Fast computation of divided differences and parallel hermite interpolation. *Journal of Complexity*5, 4 (December 1989), 417–437. DOI:http://dx.doi.org/10.1016/0885-064x(89)90018-6

E. Meijering. 2002. A chronology of interpolation: from ancient astronomy to modern signal and image processing. *Proceedings of the IEEE*90, 3 (March 2002), 319–342. DOI:http://dx.doi.org/10.1109/5.993400

Otto Neugebauer. 1983. *Astronomical cuneiform texts: Babylonian ephemerides of the Seleucid period for the motion of the sun, the moon, and the planets*, New York u.a.: Springer.

Questions

1. What is the rate of convergence of the Bisection Method?

1. What condition makes Newton’s Method fail?

1. What civilization is the earliest known to have used some form of interpolation?