Numerical Analysis

UT Knoxville, Fall 2023, MATH 471

Ohannes Karakashian, Alex Zhang

September 28, 2023

Contents

1 Introduction

Numerical analysis studies algorithms that use numerical approximation, focused on practical use by computers rather than theoretical use by mathematicians. It covers:

- computation, instabilities and rounding,
- solution of nonlinear equations and systems,
- interpolation and approximation by polynomials,
- numerical differentiation and integration, and
- solution of initial and boundary value problems for ordinary differential equations.

To prime ourselves for numerical analysis, let's review some key concepts from previous math courses:

1.1 Calculus Review

Concepts from calculus play a key role in numerical techniques. Let's begin by reviewing some familiar theorems.

Theorem 1.1.1 ▸ Extreme Value Theorem

(Simplified) If f is continuous on a closed interval $[a, b]$, then f attains an absolute maximum value and an absolute minimum value at some values x_0 and x_1 in $[a, b]$.

(More formal) Suppose K is a nonempty and compact subset of $\mathbb{R}$, and suppose $f : K \to \mathbb{R}$ is continuous. Then:

(a) f is bounded on K (that is, $f[K]$ is bounded),
(b) there exists $x_0 \in K$ such that $f(x_0) = \sup(f[K])$, and
(c) there exists $x_1 \in K$ such that $f(x_1) = \inf(f[K])$.

Theorem 1.1.2 ▸ Intermediate Value Theorem

Suppose f is continuous on the closed interval $[a, b]$. Then for any function value y between the minimum and maximum function values from $[a, b]$, there exists some x in

$[a, b]$ where $f(x) = y$.

A notable application of the Intermediate Value Theorem is figuring out whether a polynomial function attains zero (i.e. whether a polynomial has a root). For example, let's consider the following polynomial:

$$f(x) = 2x^3 - 3x - 7$$

Using the Intermediate Value Theorem, all we need to do is find a value x_0 where $f(x_0) < 0$, and a value x_1 where $f(x_1) > 0$. Since polynomial functions are continuous, we can use the IVT to ensure that the polynomial has a root. For the above example, we can consider $f(-1) = -2$ and $f(0) = 3$. By the IVT, there must exist some $x \in [-1, 0]$ where $f(x) = 0$.

Moving on, let's recall the following two theorems related to derivatives:

Theorem 1.1.3 ▸ Rolle's Theorem

Let $a, b \in \mathbb{R}$ where $a < b$, and let $f : [a, b] \to \mathbb{R}$ be continuous on $[a, b]$ and differentiable on (a, b). If $f(a) = 0$ and $f(b) = 0$, then there exists $c \in (a, b)$ such that $f'(c) = 0$.

Theorem 1.1.4 ▸ Mean Value Theorem

Let $a, b \in \mathbb{R}$ where $a < b$, and let $f : [a, b] \to \mathbb{R}$ be continuous on $[a, b]$ and differentiable on (a, b). Then there exists $c \in (a, b)$ such that $f'(c) = \frac{f(b)-f(a)}{b-a}$.

Intuitively, the Mean Value Theorem states the following: given any two points on a differentiable function's curve, there exists some x value where the tangent line at $f(x)$ is the same slope as the slope between those two points.

One of the most important theorems in analysis is the following:

Theorem 1.1.5 ▸ Taylor's Theorem

Let n be a non-negative integer. Let $f : (a, b) \to \mathbb{R}$ be a function where $f^{(n+1)}(x)$ exists for any $x \in (a, b)$. Then, for any $c, x \in (a, b)$, there exists some number ζ between c and x such that:

$$f(x) = p_n + R_n(\zeta)$$

Here, $p_n(x)$ represents the Taylor polynomial of f:

$$p_n(x) = f(c) + f'(c)(x-c) + \cdots + \frac{f^{(n)}(c)}{n!}(x-c)^n$$

And, $R_n(\zeta)$ represents the remainder (i.e. how far off the Taylor polynomial was to approximating the actual answer):

$$R_n(\zeta) = \frac{f^{(n+1)}(\zeta)}{(n+1)!}(x-c)^{n+1}$$

The Taylor expansion is well-liked in numerical analysis since it can turn an incomputable function such as sine or cosine to a computable one (albeit, only an approximation). The above theorem gives us a formula for determining the inaccuracy of certain Taylor polynomial approximations of functions.

We use $|R_n(\zeta)|$ as the error term for how different the approximation is from the actual answer. In practical application, we often do not know the value for ζ. However, since the theorem states ζ is between c and x, it is easy to attain and upper and lower bound for ζ.

A Taylor polynomial's accuracy is good near the initial point c, but deteriorates farther from initial point. It's a consequence of the fact that all information about the approximated function f is coming from a single point of that function.

On the other hand, interpolation is not so accurate at any specific point, but it's often more accurate across a range of points.

1.2 Sequences

Many algorithms generate sequences. For example, consider Newton's method for calculating $\sqrt{3}$:

$$\begin{cases} x_{k+1} = \frac{x_k}{2} + \frac{3}{2x_k}, & k = 0, 1, 2, \ldots \\ x_0 = 2 \end{cases}$$

In this course, we will use 0 to index the first element of a sequence.

This method works as this sequence (x_k) converges to $\sqrt{3}$. We can notate this as:

$$\lim_{k\to\infty} x_k = \sqrt{3}$$

Intuitively, this means that the numbers of the sequences gets closer to the limit $\sqrt{3}$ as k gets bigger. We formalize this idea in the following definition:

Definition 1.2.1 ▸ Convergence

We say a sequence x_n converges if: for any $\epsilon > 0$, there exists $K \in \mathbb{N}_0$ such that, for all $k \geq K$, $|x_k - L| < \epsilon$.

In practical applications, we want the sequence to converge quickly. The "speed" of convergence matters, especially in numerical analysis. For example, consider the following two sequences:

$$x_n = \frac{n}{n+1} \quad \text{and} \quad y_n = 1 - 1/n^2$$

Both of these sequences converge to 1. However, it might be intuitive to think that (y_n) converges "faster" than (x_n). The problem is: how do prove that (y_n) converges "faster" than (x_n)? And how do we quantify this? We may look at the progress achieved at each iteration of the sequence. More specifically, we can calculate:

$$\frac{|x_{n+1} - L|}{|x_n - L|}$$

where L is the limit of this sequence. The lower this expression evaluates, the "faster" or "better" the sequence.

Definition 1.2.2 ▸ Sublinear, linear, superlinear

Let x_n be a sequence that converges to L, and suppose that $\lim_{n\to\infty} \frac{|x_{n+1}-L|}{|x_n-L|} = \lambda$ where $0 \leq \lambda \leq 1$.

- If $\lambda = 1$, then (x_n) is ***sublinear***.
- If $0 < \lambda < 1$, then (x_n) is ***linear***.
- If $\lambda = 0$, then (x_n) is ***superlinear***.

Although all three of these types are convergent, it is clear that a lower value of λ yields a more easily-computable sequence. Thus, a superlinear sequence is generally more ideal for computation than either linear or sublinear. This would not be true if the steps themselves are more difficult to compute. (TODO: wording)

(TODO: convergence of order, quadratic convergence)

Linear convergence is slow if λ is close to 1. There is a way to accelerate convergence known

as Aitken's Δ^2 method.

(TODO: definition 1.10 and 1.11, theorem 1.12)

Theorem 1.12: intuitively says that Aitken's method actually works in making the new sequences faster than the old

We can apply Aitken's method again to the result of Aitken's method to again achieve a more quickly converging sequence.

When we have a converging sequence, how do we know how far from the limit we actually are? Assuming we don't know the limit beforehand, one way can be to look at the difference between successive steps. This might be a bad metric for some sequences, where two consecutive numbers may be extremely small, but are still far from the limit.

(Theorem: test of "closeness" to limit (stopping criterion) for superlinear sequences)

1.3 Computational problems and conditioning

Just like how absolute value measures the "magnitude" of a real number, the norm measures the size of a vector.

We view a ***computational problem*** as a function that converts inputs to numbers. More specifically, it is a function $f : X \to Y$ where: (todo: normed vector space from notes)

Example 1.3.1 ▸ Computational problem

Given f, find root r.

$$g : C(\mathbb{R}) \to r$$

Definition 1.3.2 ▸ Condition number

For a square matrix A, we define the (TODO)

Definition 1.3.3 ▸ Well-conditioned, ill-conditioned

Intuitively, a problem is:

- ***well-conditioned*** if a small change in input causes only a small change in output.
- ***ill-conditioned*** if a small change in input may cause a large change in output.

For example, we may want to solve a linear system of the form $Ax = b$. more

If a problem is ill-conditioned, then this cannot be cured or fixed by an algorithm. That is, the problem is too unstable for an algorithm to produce a good solution. An ill-conditioned problem is hopeless; it's often better to reconsider the physical phenomenon being observed and reformulate it in a better way.

We can apply the same criteria to algorithms: ***stable*** and ***unstable***.

Definition 1.3.4 ▸ Stable, unstable

Intuitively, an algorithm is:

- ***stable*** if a small change in input causes only a small change in output.
- ***unstable*** if a small change in input may cause a large change in output.

Ideally, we work with well-conditioned problems and solve it using a stable algorithm.

2 Floating Point Numbers

Brief: How computers store numbers

2.1 Integers

Every integer can be represented as:

$$\alpha_n \beta^n + \alpha_{n-1} \beta^{n-1} + \cdots + \alpha_1 \beta^1 + \alpha_0 \beta^0$$

where $\beta \in \mathbb{Z}$ represents the base we choose, and $\alpha \in \mathbb{Z}$ must be $0 \leq \alpha < \beta$.

(TODO: binary representation for computers, two's complement for signed integers)

A negative integer $-y$, $1 \leq y \leq 2^{31}$ is stored as the binary representation of $2^{32} - y$.

2.2 Real Numbers

3 Non-linear Equations and Systems

3.1 Roots and Fixed Points

Definition 3.1.1 ▸ Root

Given a function $f : [a, b] \to \mathbb{R}$, we say that $x^* \in [a, b]$ is a ***root*** of f if $f(x^*) = 0$.

Intuitively, we can think of roots as places where the graphed function crosses the x-axis or $y = 0$ line. A common question in mathematics is finding the root of a given function.

ill-condition well-condition example

Definition 3.1.2 ▸ Fixed point

Given $g : [a, b] \to \mathbb{R}$, we say $p \in [a, b]$ is a ***fixed point*** of g if $g(p) = p$.

Intuitively, we can think of fixed points as places where the graphed function crosses the $y = x$ line (often called a ***bisecting*** line).

Root problems and fixed point problems are related in the sense that one can be "converted" into the other.

Example 3.1.3 ▸ Root problem to fixed point problem

For example, we may want to find the root(s) of the function $f(x) = x^2 + 2x$. We would solve for $f(x) = 0$, which means we want to solve the equation $x^2 + 2x = 0$ for x. We can convert this to a fixed point problem in several ways, including:

1. $f(x) = 0 \iff x + f(x) = x$
2. $f(x) = 0 \iff g(x) = x - \dfrac{f(x)}{f'(x)}$
3. This is the same as finding the fixed-point(s) of $g(x) = -x^2/2$.

Example 3.1.4 ▸ Root problem

$f(x) = \sqrt{x^2 + x - \cos x} + \sin x.$

Finding a root of f is equivalent to finding a fixed point of which function g?

Example from notes

To solve the root problem, there are several methods, including:

- Bisection
- Newton's method
- Variants of Newton's method

Technique 3.1.5 ▸ Calculating midpoint in fixed-precision computing

In fixed-precision computing, we calculate the midpoint m between two numbers a and b where $a < b$ by:

$$m := a + \frac{b - a}{2}$$

Why (a+b)/2 is bad for fixed-precision computing

Technique 3.1.6 ▸ Root problem: bisection method

This method is analogous to binary search. Assume $f : [a, b] \to \mathbb{R}$ has a root in some interval $[a, b]$. For simplicity, we will assume that there is only one root.

todo: rest of this

As with any iterative method, we need some criteria to know when to start and stop the method. We may stop the method once the interval reaches some small width.

Using IEEE-754 floating points, each iteration of the bisection method will yield one more bit of accuracy. Thus, for single-precision floats, we only need 22 iterations to achieve full accuracy. For double-precision floats, we only need 52 iterations.

3.2 Newton's Method

Notes on paper

Definition 3.2.1 ▶ Multiplicity of a root

Let $f : [a,b] \to \mathbb{R}$ be a function, and let $r \in [a,b]$ be a root of f. We say r is a ***root of multiplicity*** $m \in \mathbb{Z}^+$ of f if:

$$f(r) = \cdots = f^{(m-1)}(r) = 0 \quad \text{but} \quad f^{(m)}(r) \neq 0$$

For example:

- If $f'(r) \neq 0$, then r is a ***simple root*** or ***root of multiplicity*** 1.
- If $f'(r) = 0$ and $f''(r) \neq 0$, then r is a ***root of multiplicity*** 2.

Suppose r is a root of multiplicity m of f. Then there exists some function q such that:

$$f(x) = (x-r)^m q(x) \quad \text{with} \quad q(r) \neq 0$$

Newton's method only converges quickly if (1) we start with a guess "close enough" to the actual root, and (2) we have $f'(r) \neq 0$. If $f(r) = f'(r) = 0$, then our convergence is slow. This can be fixed by modifying to:

$$x_{k+1} = x_k - m\frac{f(x_k)}{f'(x_k)}$$

However, it is expensive to compute the function, and it is especially expensive to compute the derivative. Worse yet, there might not be a computable derivative function. For example, f itself may be unknown to us, perhaps operating purely on empirical data. Without access to the underlying function, there is no good way to compute the derivative.

The problem is further exacerbated in the case of vector functions in several variables. In this case, $f'(x)$ is an $n \times n$ matrix of partial derivatives, called a ***Jacobian matrix***. We may have:

$$J(x) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & & & \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \cdots & \frac{\partial f_n}{\partial x_n} \end{bmatrix}$$

How can we replace $f'(x_k)$ by something "cheaper" and/or more available? This brings us to the secant method. Recall that the derivative is defined by the following limit:

$$f'(x_k) = \lim_{x \to x_k} \frac{f(x) - f(x_k)}{x - x_k}$$

The limit is expensive and sometimes infeasible to compute. We can approximate $f'(x_k)$ by simply choosing some ζ close to x_k:

$$f'(x_k) \approx \frac{f(\zeta) - f(x_k)}{\zeta - x_k}$$

A common and good choice for ζ is simply x_{k-1}, the previous term in Newton's method.

Technique 3.2.2 ▸ Secant method

Taking the approximation $f'(x_k) \approx \frac{f(x_{k-1}) - f(x_k)}{x_{k-1} - x_k}$, we have:

$$\begin{cases} x_{k+1} = x_k - \frac{f(x_k)}{\frac{f(x_{k-1}) - f(x_k)}{x_{k-1} - x_k}}, & k \geq 1 \\ x_0, x_1 \text{ given.} \end{cases}$$

Note that whereas Newton's method requires one starting value x_0, the secant method requires two starting values x_0 and x_1.

Some properties of Secant method:

- Almost as fast as Newton's method. In fact, we can show Newton's method converges quadratically, meaning $|e_{n+1}| \approx c|e_n|^2$. Instead of converging by an exponent of 2, the secant method converges by an exponent of $\frac{1+\sqrt{5}}{2} \approx 1.6$. This difference in convergence speed is less noticeable when we start closer to the actual root.

3.3 Roots of Polynomials

Any polynomial can be written in the following standard form:

$$p(z) = a_n z^n + \cdots + a_z z + 1$$

Theorem 3.3.1 ▸ Fundamental Theorem of Algebra

A polynomial of degree $n \geq 1$ has at least one complex root.

It follows that p can have up to n roots

write the m m-1 m-2 thing

For example, we may a polynomial like:

$$p(x) = (x-1)^2(x+1)^3$$

This polynomial has two distinct roots but five total roots: 1 which has multiplicity 2, and -1 which has multiplicity 3.

Suppose all coefficients are real numbers. Then if $z = a+bi \in \mathbb{C}$ is a root, then so is $\bar{z} = a-bi$.

Descarte's rule of signs

This limits the possibilities for the number of positive roots. For example, if there are no sign changes, then we know there are no positive roots. If there is only one sign change, then we know there must be one positive root.

Consider $p(x) = 2x^3 + 2x^2 + 1$. As long as $x > 0$, then $p(x) > 0$. Any polynomial $p(x)$ might only be 0 if $x < 0$, which is not guaranteed.

All roots are contained within a disk in the complex plane $\mathbb{C}$ within $|z| \leq \alpha$. Here, $|z|$ denotes the Euclidean distance from 0 to z in the complex plane.

Horner's method

There is a specific way to compute the value of a polynomial. For example, to compute $p(x) = 2x^5 + 7x^4 + 3x^3 + 2x - 5$, we may do:

$$p(a) = -5 + a(2 + a(0 + a(3 + a(7 + a(2)))))$$

Technique 3.3.2 ▸ Horner's method

More about this?

Horner's method is the most efficient method for computing the value of a polynomial. Let $p(x)$ be a polynomial, and let a be given. We wish to calculate $p(a)$. Define $b_n, \ldots, b_0$ by:

$$b_n = a_n$$

$$b_k = ab_{k+1} + a_k \quad \text{where } k \in \{n-1, \ldots, 0\}$$

We get $b_0 = p(a)$, and moreover, $p(x) =$ TODO: MORE

Usefulness of horner's method on polynomial's derivative; usefulness when doing newton's method for polynomials

Moreover:

$$p(x) = (x - x_0)Q(x) + b_0$$
$$p'(x) = Q(x) + (x - x_0)Q'(x) + 0$$
$$p'(x_0) = Q(x_0) + 0$$

So:

$$a_n a_{n-1} \cdots a_1 a_0$$
$$b_n b_{n-1} \cdots b_1 b_0$$
$$Q(x) = b_n x^{n-1} + \cdots + b_2 x + b_1$$

3.4 Deflation

Say r is a root of p. Horner's method tells us that:

$$p(x) = (x - r)Q(x) + b_0 \quad \text{where } b_0 = 0 = p(r)$$

The remaining roots are in $Q(x)$ (the "quotient"). Whatever method we applied to find r, we can apply it to Q as well to find the other roots. This is a technique known as ***deflation***. Essentially, we "deflate" the original polynomial by one root at a time.

For any $p(x) = (x - r_1)Q_{n-1}(x)$, we have $Q_{n-1}(x) = (x - r_2)Q_{n-2}(x)$. We continue this until we get to $Q_1(x) = (x - r_n)$.

tecbox for deflation?

Cascading Error Consider the case r_1 is an approximate root. Our b_0 is hopefully quite small, in which case we might simply pretend that the b_0 isn't even there. Then apply root finding to Q to find another approximate root r_2, and drop the b_0. This can lead to a cascading series of errors, growing exponentially.

Our erroneous approximations can be "refined" by applying Newton's method to the **original** polynomial $p(x)$, using our approximated roots as starting values.

3.5 Eigenvalue Stuff

Definition 3.5.1 ▶ Eigenvector, eigenvalue

Given an $n \times n$ matrix A, if there exist $\vec{v}$ and scalar λ such that:

$$A\vec{v} = \lambda\vec{v}$$

then we call $\vec{v}$ the ***eigenvector*** and λ the ***eigenvalue***.

Moreover, for identity matrix I, we define the ***characteristic polynomial*** as:

$$P_A(\lambda) \coloneqq \det(\lambda I - A)$$

3.6 The Fixed-Point Problem

Recall that a fixed point p of a function $g : [a,b] \to \mathbb{R}$ is where $g(p) = p$. Intuitively, it's wherever the graph of g intersects the $y = x$ line.

> We will use f to denote functions in root-finding problems and use g to denote functions in fixed-point problems.

Theorem 3.6.1 ▶ Fixed Point Theorem

Let $g : [a,b] \to [a,b]$ be a function continuous on $[a,b]$, and suppose that g is differentiable on (a,b), satisfying:

$$-1 < g'(x) < 1 \quad \text{for any } x \in (a,b)$$

Then g has at least one fixed point in $[a,b]$.

Intuition: Since the function is bounded to this square $[a,b] \times [a,b]$ and is continuous, then there's no way for the function to avoid the line $y = x$.

Proof. Let $f(x) \coloneqq g(x) - x$. Since g is continuous, then f is continuous. Also:

$$f(a) = g(a) - a \geq 0$$

$$f(b) = g(b) - b \leq 0$$

There are three possibilities:

1. If $g(a) = a$, then a is the fixed point, and we are done.
2. If $g(b) = b$, then b is the fixed point, and we are done.
3. If $g(a) \neq a$ and $g(b) \neq b$, then $a < g(x) < b$. $g(x) > a$ implies $f(a) > 0$, and $g(x) < b$ implies $f(b) < 0$. Thus, by the **??**, f has a root in (a, b).

□

Fixed point to root finding and vice versa

uniqueness of fixed point

Fixed Point iteration To approximate a fixed point, define:

$$\begin{cases} p_{n+1} = g(p_n), & n \in \{0, 1, 2, \ldots\} \\ p_0, & \text{given} \end{cases}$$

Convergence of fixed point iteration

k is the upper bound on derivative

3.7 Systems of Nonlinear Equations

Now we consider vector equations, which map $\mathbb{R}^n$ to $\mathbb{R}^n$, denoted:

$$F : \mathbb{R}^n \to \mathbb{R}^n$$

Example 3.7.1 ▸ Root finding problem for vector functions

For example, we may have $F : \mathbb{R}^2 \to \mathbb{R}^2$ where $F(x, y) = \begin{bmatrix} x - y^2 \\ \cos x + \sin y \end{bmatrix}$.

Find $\vec{x} \in \mathbb{R}^2$ such that $F(\vec{x}) = 0$.

For the first component:

$$\cos x + \sin y = 0 \implies y = \pm x$$

$$\cos x + \sin y = 0 \implies \ldots$$

Newton's Method for Systems Newtons' method generalizes very well to vector functions. Recall the regular Newton's method for scalar values:

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$$

We instead write:

$$\vec{x}_{k+1} = x_k - \left[f'(\vec{x}_k)\right]^{-1} f(\vec{x}_k)$$

Note that matrix multiplication does not commute, so we must not switch the roles of the two matrices. We may also instead write:

$$f'(\vec{x}_k)\underbrace{(\vec{x}_{k+1} - \vec{x}_k)}_{\vec{\delta}_k} = -f(\vec{x}_k)$$

Or better yet:

$$\begin{cases} f'(\vec{x}_k)\vec{\delta}_k = -f(\vec{x}_k) \\ \vec{x}_{k+1} = \vec{x}_k + \vec{\delta}_k \end{cases}$$

We:

- Form matrix $f'(\vec{x}_k)$
- Form $f(\vec{x}_k)$ (vector)
- Solve linear system $f'(\vec{x}_k)\vec{\delta}_k = -f(\vec{x}_k)$.

How might we improve the efficiency of this method? One way may be to FINISH THIS

3.8 Broyden's Method

Broyden's method

Norms For a given vector $\vec{v} = (v_1, \ldots, v_n)$, there are various notions for measuring the "size" or "magnitude" of $\vec{v}$. The most common is the Euclidean norm:

$$\|\vec{v}\|_2 = \sqrt{\sum_{i=1}^{n} v_i^2}$$

There is also the "Manhattan norm" or "taxicab norm":

$$\|\vec{v}\|_1 = \sum_{i=1}^{n} |v_i|$$

In general, we can take the norm with respect to any non-negative real number r:

$$\|\vec{v}\|_r = \sqrt[r]{\sum_{i=1}^{n} |v_i^r|}$$

As r approaches infinity, we have:

$$\|\vec{v}\|_\infty = \sup\{|v_i| : 1 \leq i \leq n\}$$

This is called the ***infinity norm*** or ***supremum norm***.

Given two column vectors u and v, we have the inner product defined as:

$$u^T v = \begin{pmatrix} u_1 & \dots & u_n \end{pmatrix} \begin{pmatrix} v_1 \\ \vdots \\ v_n \end{pmatrix} = u_1 v_1 + \cdots + u_n v_n$$

The outer product is defined as:

$$uv^T = \begin{pmatrix} u_1 \\ \vdots \\ u_n \end{pmatrix} \begin{pmatrix} v_1 & \dots & v_n \end{pmatrix} = \text{todo: n by n matrix}$$

Fixed Point System (FPS) Recall that in fixed point iteration, we take:

$$x_{k+1} = G(x_k)$$

When we compute $G(x_k)$, the value x_k is not altered. However, in Nonlinear Gauss-Seidel, x_k is updated.

a lot

4

Approximation of Functions

We've already seen one way of approximating functions by their Taylor polynomial. Although the approximation is excellent for values "near" the initial point, error can quickly grow for more "distant" values.

Index

Definitions

Examples

Techniques

Theorems