Numerical Methods

Class website – www.cs.purdue.edu/homes/cs314

Goal

- You will learn the potential & limitations of the computer in solving mathematical problems
- You will learn how to use the computer to solve or approximate non linear & linear equations. Integrate, differentiate, interpolation and solving differential equations

Textbook

Numerical methods using MATLAB 4th edition – Mathews & Fink

Mailing List

It is important to add yourself to the mailing list. To do this

- Login to a CS machine (lore)
- Type – 'mailer add me to cs314-students
- If you don't have a CS account yet, send an email to grr@cs.purdue.edu

Grade Distribution

25% → Final Exam
25% → Mid-Term Exam
50% → Homework's & Projects

Syllabus

- Floating and fixed point representation of numbers
- Solutions of non linear equations of the form f(x) = 0
- Solutions of linear systems Ax=B
- Interpolation
- Curve fitting
- Numerical differentiation & integration
- Numerical optimization
- Solutions of differential equations

Binary Numbers

Computers use binary numbers

Example

$1011_b = 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 8 + 2 + 1 = 11$

$537.5_d = 5 \times 10^2 + 3 \times 10^1 + 7 \times 10^0 + 5 \times 10^{-1}$

$110.11_b = 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 + 1 \times 2^{-2}$

15/06/2010 – Lecture 2

Q) How to go from base 10 to base 2?

A) First we have to separate the integer part (part >=1) from the fraction (<1)

The integer part has the form $N_{10} = b_0 + (b_1 \times 2^1) + (b_2 \times 2^2) \ldots\ldots (b_k \times 2^k)$ and $N_b = b_k \ldots b_2 b_1 b_0$

We wish to find the digits b_i

Then we have, $\frac{N}{2} = \frac{b_0}{2} + b_1 + \ + b_2 x 2^1 + \ b_3 x 2^2 + \cdots + \ b_k x 2^{k-1}$

Where the first term of the right hand side is called the remainder and the rest of the term forms the integer part.

The remainder determines if b_0 is 0 or 1. Then to determine the rest of the b's we keep dividing by 2 similar to obtaining b_0.

Example

23 to binary

23/2 quotient = 11 & remainder = 1
11/2 quotient = 5 & remainder = 1
5/2 quotient = 2 & remainder = 1
2/2 quotient = 1 & remainder = 0
1/2 quotient = 0 & remainder = 1 ……. Stop here because quotient obtained is 0

Binary number = 10111

Fraction part

$F_{10} = b_{-1}2^{-1} + b_{-2}2^{-2} + \cdots \ldots + b_{-m}2^{-m}$
$F_b = 0.b_{-1}b_{-2} \ldots b_{-k}$

To get the binary number

$$\boldsymbol{2xF} = b_{-1} + b_{-2}2^{-1} + \cdots \ldots + b_{-m}2^{-m+1}$$

If the LHS >= 1 then b_{-1} =1 else it will equal 0

Example

Convert 0.23 from decimal to binary

$$0.23 * 2 = 0.46 < 1$$
$$0.46 * 2 = 0.92 < 1$$
$$0.92 * 2 = 1.84 > 1$$
$$0.84 * 2 = 1.68 > 1$$
$$0.68 * 2 = 1.36 > 1$$
$$0.36 * 2 = 0.72 < 1$$
$$.$$
$$.$$
$$.$$
$$.$$
$$.$$

This tells us that the binary number is 0.001110

A question that arises might be when do we stop?

Example

Convert 0.5 to binary

$$0.5 * 2 = 1.0 = 1$$
$$0 * 2 = 0 < 1 \rightarrow since\ we\ obtained\ a\ 0\ as\ the\ result\ we\ stop$$

Binary number is 0.10

Example

Convert 5.33 from decimal to binary

5/2 quotient = 2 & remainder = 1
2/2 quotient = 1 & remainder = 0
0/2 quotient = 0 & remainder = 1……stop here since quotient is 0

Fraction part

$$.33 * 2 = 0.66 < 1$$
$$0.66 * 2 = 1.32 > 1$$
$$0.32 * 2 = 0.64 < 1$$
$$.$$
$$.$$
$$.$$
$$.$$
$$.$$

Binary number obtained is 101.01010

Floating point number representation

How to represent numbers with integer part and fraction in memory?

NNN.FF

Two ways 1) Fixed Point 2) Floating Point

Fixed Point → The number of bits for the integer and fraction are fixed.

NNNN.FF

That is we have

4 bits for the integer
2 bits for the fraction
1 bit for the sign

For the sign, if we have a 0 that tells us its positive where as a 1 tells us it's negative.

The largest number that can be represented with the above configuration is 1111.11 = 15.75

The smallest number will just be the negative of the largest hence it is -15.75

The smallest difference between 2 consecutive numbers is

$$0000.01 - 0000.00 = 0000.01 \rightarrow 2^{-2} \rightarrow 0.25$$

Problem: little range, bad precision
Advantage: fast arithmetic operations using integer arithmetic

Floating Point → The decimal point moves around to keep the desired precision. Use scientific notation in binary.

In this representation the computer represents numbers as

$$x = \pm 1.q * 2^{\pm e}$$

Where q is the mantissa and e is the exponent value.

0 is represented as all 0's since it cannot be represented in the form shown above.

In floating point we allocate a number of bits for the mantissa and a number of bits for the exponent.

16/06/2010 – Lecture 3

Errors in computation of numbers

Absolute error → $|p - p^*|$

Relative error → $\frac{|p-p^*|}{|p|}$

Where p is a number and p^* is the approximate value of p

The number of bits used to represent a number are limited. So we need to "chop off" or "round off" a number.

For example π = 3.1415926

Chop off → limit to 3 digits we get 3.141
Round off → limit to 3 digits we get 3.142

	Absolute error	Relative error
Chop off	0.59e-3	1.9e-4 = 0.02%
Round off	0.41e-3	1.3e-4=0.01%

Propagation of errors

Assume that p and q are approximated by p^* and q^* such that

$$p^* = p + e_p$$
$$q^* = q + e_q$$

sum: $p^*+q^* = p + e_p + q + e_q = (p + q) + (e_p + e_q)$

In the sum, the new error is the sum of the initial errors.

Product: $p^* * q^* = (p + e_p) * (q + e_q) = pq + (qe_p + pe_q) + e_p e_q$

In the product, the new errors are the result of the amplification of the initial errors by q and p. That is the errors are amplified by the magnitudes of q and p.

- An algorithm is stable if the small initial errors stay small
- An algorithm is unstable if the small initial errors get larger & larger

Solution of nonlinear equations

f(x)= 0

we will use iteration to solve the equations starting with a value p_0 and a function $P_k = g(P_{k-1})$

we stop when $|p_k - p_{k-1}| < \varepsilon$ where epsilon is a small number

Definitions

Fixed point

A fixed point of *g(x)* is a real number P such that *P=g(P)*
Geometrically the fixed point of a function *y=g(x)* are the points of intersection between the line *y=g(x)* and *y=x.* This is shown in the graph below.

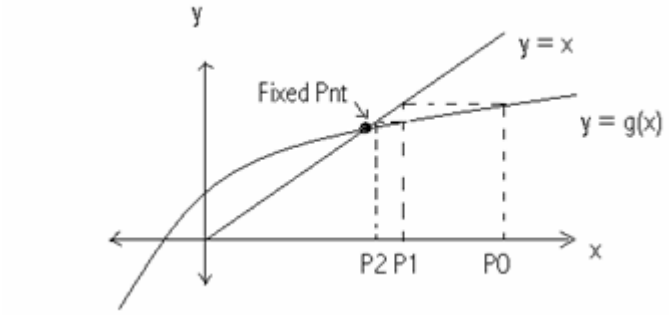


Image obtained from Karthik Rajasekar

Example

$$x^2 - 2x + 1 = 0$$
$$\Rightarrow x_k = \frac{{x_{k-1}}^2 + 1}{2}$$

$$x_0 = 0$$
$$x_1 = 0.5$$
$$x_2 = \frac{0.5^2 + 1}{2}$$
$$\vdots$$
$$x_6 = 0.79$$

$$g(x) = \frac{x^2 + 1}{2} \quad , \qquad g'(x) = x$$

X	*G(x)*
0	*0*
0.3	*0.3*
0.6	*0.6*
1	*1*

0<g'(x)<1 for x ranging from 0 to 1.

Analytic Solution

$$x^2 - 2x + 1 = 0 \Rightarrow x = 1$$

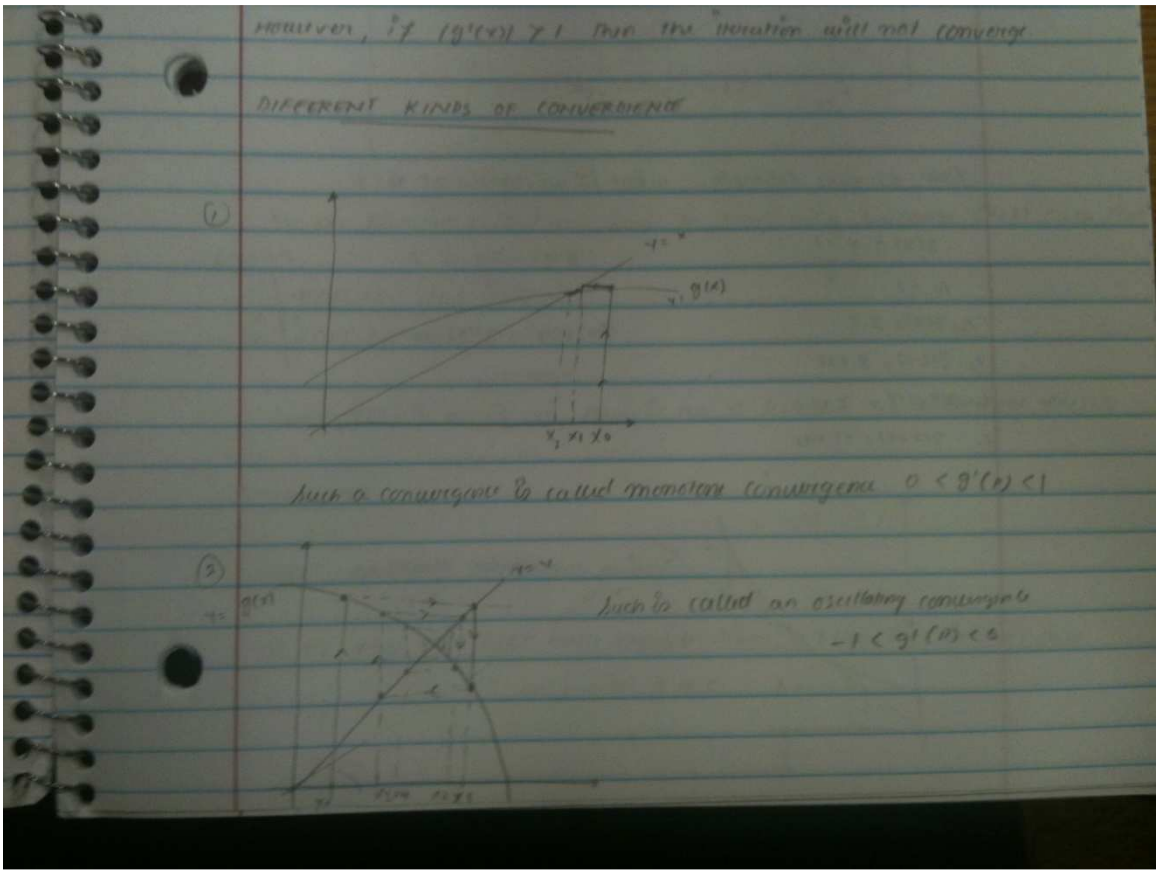
17/06/2010 – Lecture 4

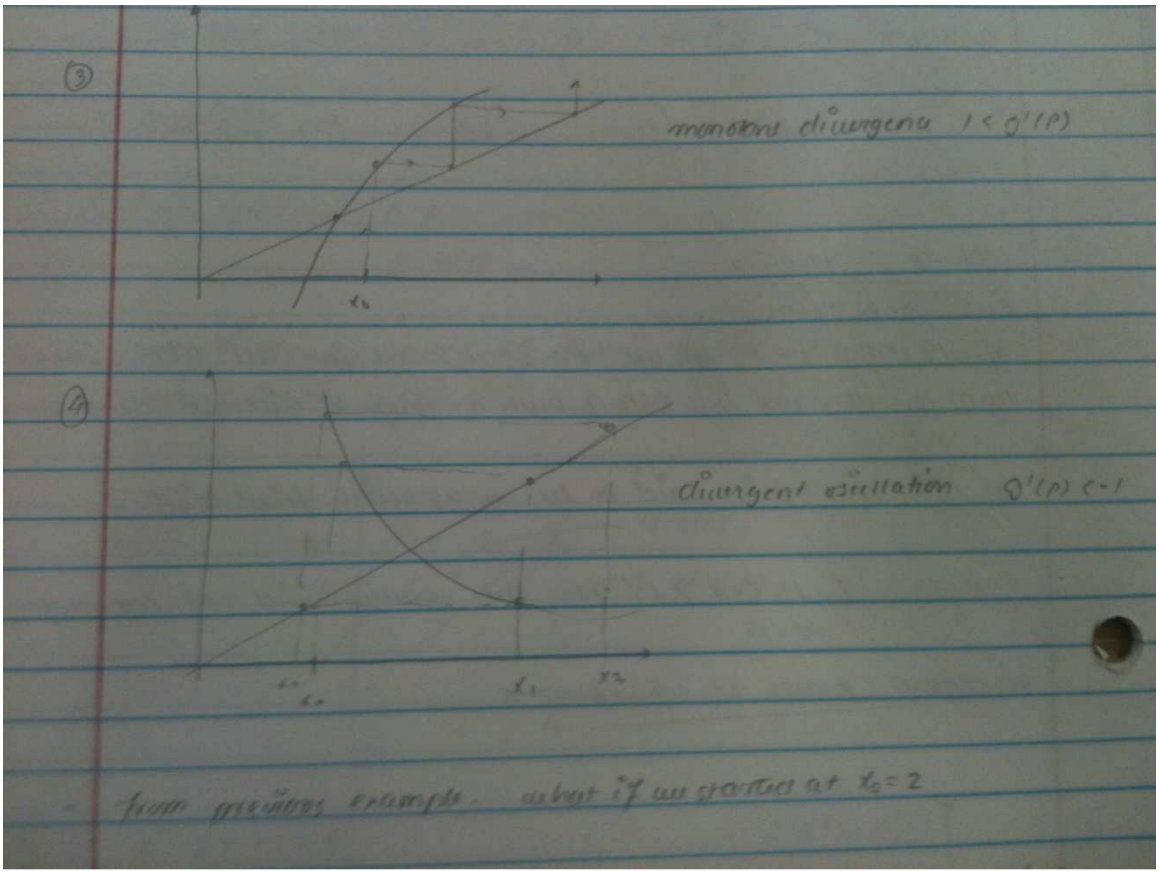
Fixed Point Theorem

If | g'(x) | ≤ k < 1, for x in [a,b] and g(x) in [a,b], then the iteration $P_n = g(p_n\text{-}1)$ will converge to a unique fixed point. In this case P is said to be an attractive fixed point

If | g'(x) | > 1, then the iteration $P_n = g(p_n\text{-}1)$ will not converge.

Different types of convergence





From the previous example, what if we started at 2.

$$g(x) = \frac{x^2 + 1}{2}, g'(x) = x$$
$$x_0 = 2$$
$$x_1 = g(2) = 2.5$$
$$x_2 = g(2.5) = 3.625$$
$$\vdots$$

$$for\ x > 2\ g'(x) > 1$$

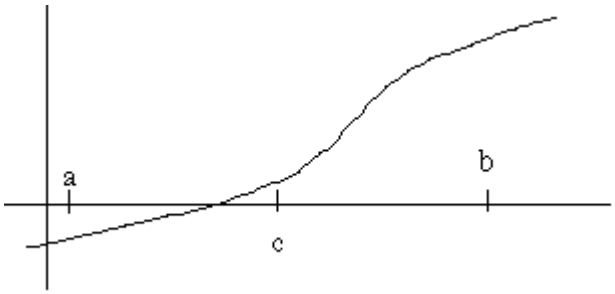
Monotone divergence

Bisection Method

Used to find the solution of *f(x) = 0.* This is similar to binary search

# of guesses = $\log_2$ (100)

When dividing an interval in halves until the range becomes 1.



In the bisection method you have to start with a subrange {a,b} such that

f(a) <0 and f(b)>0 or

f(a) >0 and f(b)<0

since *f(x)* is contigious, there is an x between a,b such that *f(x) =0*

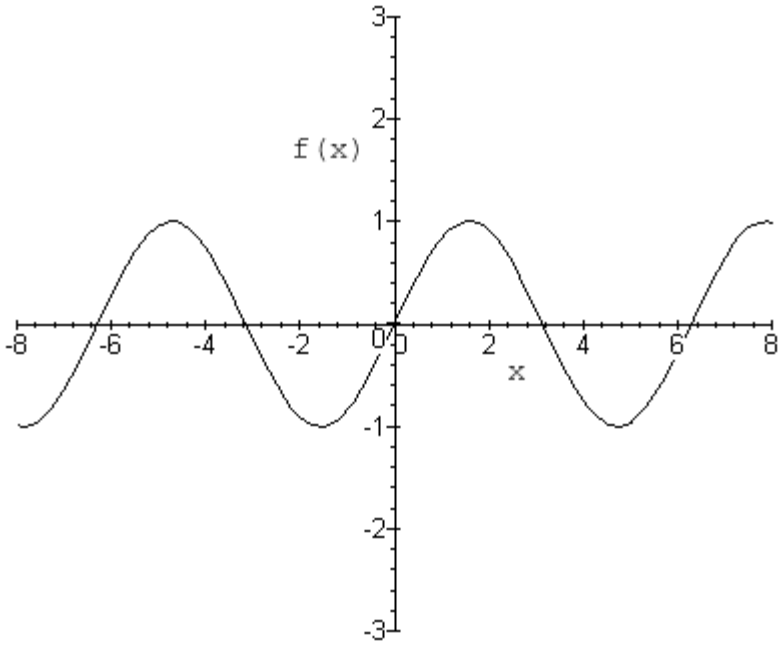
Algorithm

1. Get $c = \frac{a+b}{2}$ which is the mid point
2. If *f(a) and f(c)* have opposite signs, a zero lies in [a,c] then we assign b to c.
3. If *f(c) and f(b)* have opposite signs then a zero lies in [c,b] then assign a to c.
4. If *f(c)=0* then c is the solution to *f(x) =0.*

Usually step 4 will take a long time or may never happen, so we stop when $|f(c)| < \varepsilon$. Where epsilon is a small value.

Example

Sin(x) = 0.5



Graph of *f(x) = sin(x)*

We have *f(x) = sin(x) – 0.5* which simply shifts the graph above down by 0.5 units.

Initially,

A=0 b=50

f(0) = sin(0)-0.5 = -0.5
f(50)= sin(50)-0.5=0.266

c=(0+50)/2 = 25

A	*B*	*C*	*f(c)*
0	*50*	*25*	*-06677*
25	*50*	*37.5*	*0.1089*
25	*37.5*	*31.25*	*0.018*
.	.	.	.
.	.	.	.
.	.	.	.
.	*30*		*0*

18/06/2010 Lecture 5

How many iterations do we need for bisection?
At each step the range |a-b| is reduced by half. Therefore at step n we have

$|a_n-b_n| = \frac{|a-b|}{2^n}$

If we want an approximation error < epsilon

$$|a_n - b_n| = \frac{|a-b|}{2^n} < \varepsilon$$
$$\Rightarrow \frac{1}{\log\ 2} log \frac{|a-b|}{\varepsilon} < n$$

For the previous example using the formula above we get n>12 iterations.

False Position Method

Created because the bisection method was too slow. It also needs {a,b} where there is a zero between a and b.

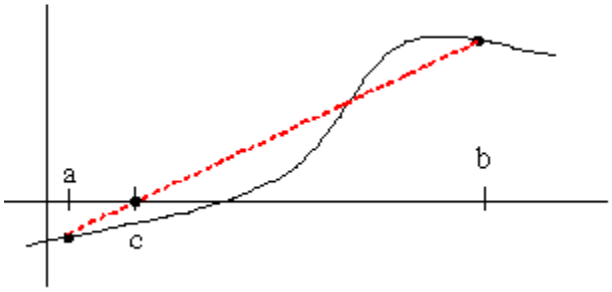


Image created by Emil Stefanov

How do we obtain the value of c

$$m = \frac{f(a)-f(b)}{a-b} \qquad m = \frac{f(b)-0}{b-c}$$

Combining these two and solving for c we obtain

$$c = b = f(b)\frac{a-b}{f(a)-f(b)}$$

A line is subtended between *(a,f(a)) and (b,f(b))* and then c is chose to be the intersection of the line with the x axis. As in bisection if *f(a) and f(c)* have different signs then assign b to c else assign a to c.

For the previous example

sin(x) = 1/2 f(x) = sin(x) - 1/2 = 0
Start with a=0, b=50

Iteration	a	f(a)	b	f(b)	c	f(c)
0	0	sin(0) - 0.5= -0.5	50	1.266	32.637	sin(32.637) = 0.0393
1	0	-0.5	32.637	0.0393	30.259	0.0039
2	0	-0.5	30.259	0.0039	30.0248	0.000375

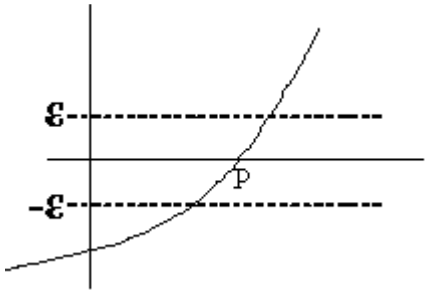
Once again we stop when the method produces a value that is lower than a predetermined epsilon.

Checking for convergence

1. **Horizontal convergence**

Stop when |*f(x)*| < epsilon

Geometrically this is shown below



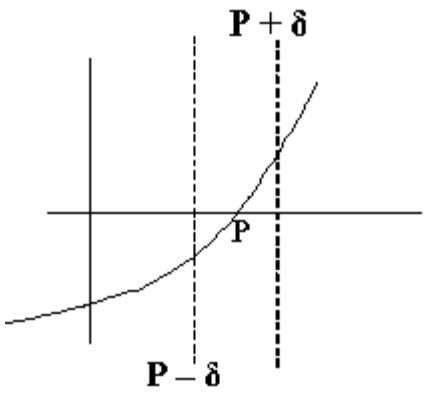
2. **Vertical convergence**

Stop when |x-p|<delta where P is the zero of the equation

The problem with this method is that we need to know p beforehand.

We can approximate the value of P by using the value of p in the previous iteration

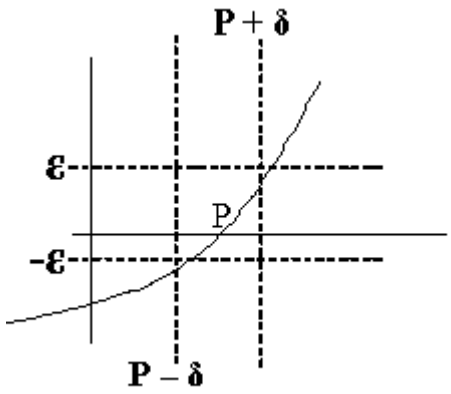
Geometrically this is shown below



3. **Both vertical and horizontal convergence**

We stop when both $|p_n-p_{n-1}| < \delta$ and $|f(x)| < \varepsilon$

Graphically this is shown below



Troublesome functions

1. If the graph of y=f(x) is steep near the root, then the root finding is called well conditioned. That is a solution is easy to obtain with good precision.
2. If the graph of y=f(x) is shallow near the root then the root finding is ill conditioned. That is the root finding may only have few significant digits.

Newton Rapson Method

If *f(x) f'(x) and f''(x)* are continuous then we can use this information to find the solution of *f(x)*

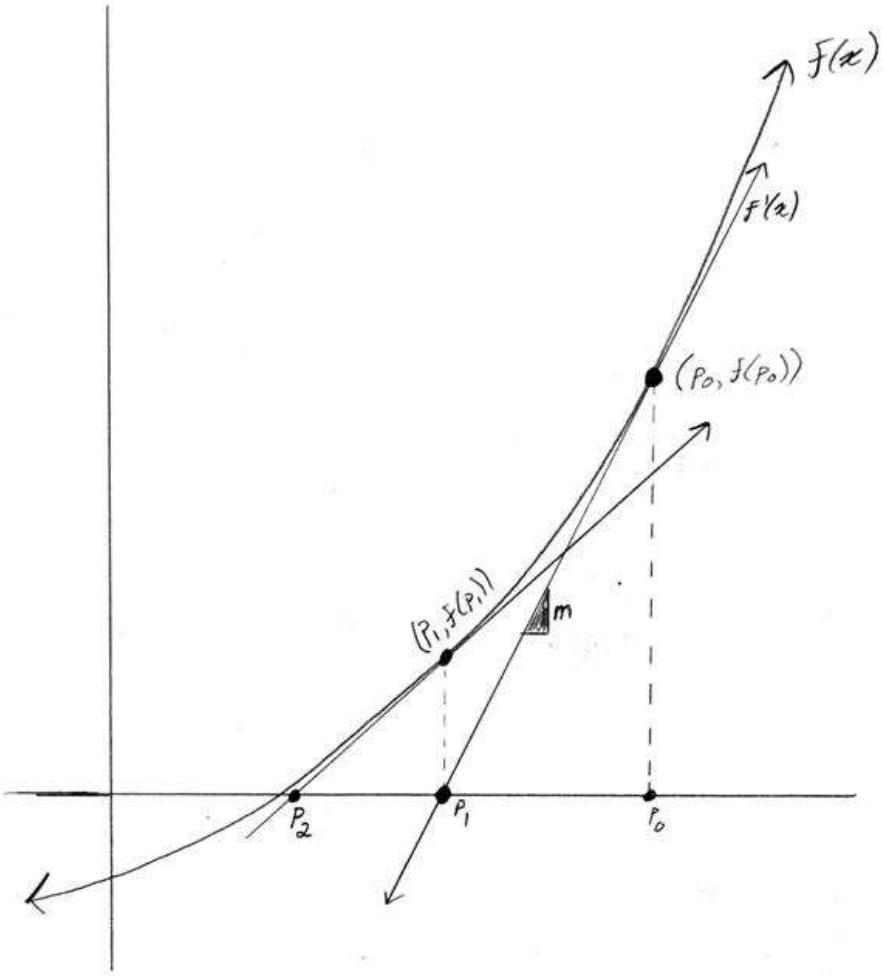


Image created by Charles Milutinovic