Practicum Module

BIOMEDICAL COMPUTATION

Endah Purwanti, S.Si., M.T.

Osmalina Nur Rahma, S.T., M.Si.

Alfian Pramudita Putra, S.T., M.Sc.

BIOMEDICAL ENGINEERING STUDY PROGRAM
DEPARTMENT OF PHYSICS
FACULTY OF SCIENCE AND TECHNOLOGY
UNIVERSITAS AIRLANGGA

2019

TABLE OF CONTENT

Contents

Preface: Rules of the Class ii

Chapter 1. Introduction to Python: Core Python 1

Chapter 2. Finding-Root Method: Bisection 12

Chapter 3. Newton-Raphson 16

Chapter 4. System of Linear Equation: Gauss Elimination 18

Chapter 5. Jacobi & Gauss Seidel Iteration 25

Chapter 6. Regression: Linear & Polynomials 29

Chapter 7. Interpolation Lagrange 35

Chapter 8. Numerical Derivative: Finite Difference Approximations 37

Chapter 9. Numerical Integration: Trapezoid & Simpson's Rule 39

Chapter 10. Richardson Extrapolation: Derivatives 45

Chapter 11. Richardson Extrapolation: Integration 48

Chapter 12. Ordinary Differential Equation: Euler & Heun Method 53

Chapter 13. Ordinary Differentia Equation: Runge-Kutta Method 58

Chapter 14. Partial Differential Equation 64

PREFACE

Preface: Rules of the Class

Presence

1. The practical class should be followed by all student with 0% absence from all classes. If someone does not meet this requirement, so he/she failed this class.
2. The absence due to ill or leave should be equipped by an official letter and it should be handed in to the lecturer or lab assistants beforehand.
3. All the students should be in the class, by the latest 15 minutes after the schedule starts.

Requirement

1. The students should hand in the preliminary task that is given in this module before the practical class starts for every chapter.
2. All the gadgets should be turned off or in silent mode.

Practice

1. Before the practical starts, the student should prepare themselves by reading the designated module for each chapter.
2. The students should follow the explanation and algorithm in the module and implement it in the python programming software.
3. The students are prohibited to make noise, inappropriate behavior, going in and out the class without permission from lecturers or lab assistants.
4. The students may ask to the lecturer and lab assistants about their obstacles to understand the algorithm or tasks.

Additional Information

1. The students must not change their group/class/time without the permission of the lecturer.
2. **Plagiarism** would be taken into account seriously, such as copy-pasting from the other students, including senior students, putting information without citation and references, etc.
3. The students should wear polite clothes, no sandals, no T-shirts etc.
4. Another additional information would be announced in the class by the lecturer.

PREFACE

Grading System

Report

The completeness of the report	(10%)
Source code	(15%)
Result	(15%)
Discussion and Analysis	(25%)
Final Tasks	(35%)

Final Grade

Preliminary Task	(10%)
Activity in the Practical Class	(20%)
The Report	(40%)
Final Practical Test	(30%)

Report Structure

1. The practical report should contain cover page and main content.
2. The cover page should contain the chapter title and code, student's name student number (NIM), the date of the practical class, and the name of the lecturer.
3. The main content should contain literature review, source code, result, discussion and analysis, final tasks, conclusion, references.
4. Literature review: It contains brief explanation about the basic theory of the chapter and all information that could be used to explain the result obtained from the practical class, including the formula, algorithm etc. All of information should be followed by citations and mentioned the references in the last part of the report.
5. Source code: It contains the source code from python programming (copy-paste) and keep it in structural way to make it neat.
6. Result: It contains the figure, plot, or screen-capture of the python console with high-resolution quality.
7. Discussion and analysis: It contains the explanation about the result of the running source code and their comparison with the literature. It should also explain about the advantages and disadvantages of the method used in this chapter.
8. Final tasks: Every chapter has several final tasks that should be solved by

9. the students by using the method in each chapter in this module. Each task should be answered thoroughly with analysis as well.
10. Conclusion: After finishing the main content of the report, the students should tell the take home message from each chapter, such as the final statement about the method.
11. References: Only the references that are used in the report are allowed to be put in this part. It could be a book, scientific paper, proceeding from international conference, etc. References from untrusted website or blog is not encouraged.

Chapter 1. Introduction to Python: Core Python

Variables

Variable represent a value of given type in most computer languages, including Python. The following are examples in inputting variables in the Python (>>> is the Python prompt):

```
>>> a = 5          # a in integer type

>>> print (a)

5

>>> b = a*2.0    # b is flow type

>>> print (b)

10.0
```

All characters after the pound sign (#) will be ignored by the interpreter or denotes the beginning of a *comment.* In Python 3 and higher, print ( ) is a command to bring out the value of variables to the Python Console. While in Python 2.7 (and before) it prints the arguments with a space in between.

```
>>> print a      # for Python 2.7 (and before)
```

Strings

A string is a sequence of characters that is located inside the single or double quotation marks. Here is an example:

```
>>> text1 = 'Biomedical Engineers'          #using a single quote

>>> text1 = "Biomedical Engineers"          #using a double quote

>>> print (text1)

Biomedical Engineers
```

or it could be written as:

```
>>> print ('Biomedical Engineers')
```

```
Biomedical Engineers
```

Strings also could be *concatenated* with the plus operator (+) and *sliced* using (:) to extract a portion of character.

```
>>> text2 = ‘Universitas Airlangga’
>>> print (text1 + ‘ ’ + text2)              # Concatenation
Biomedical Engineers Universitas Airlangga
>>> print (text1 [0:10])                     # Slicing
Biomedical
```

Lists

A list is a sequence of arbitrary objects separated by commas and enclosed in brackets. Lists are mutable, so that its elements and length can be changed or modified. Here is the example that can be performed on lists:

```
>>> a = [4.0, 5.0, 6.0]              # Create a list
>>> a.append (7.0)                   # Append 4.0 to list
>>> print (a)
[4.0, 5.0, 6.0, 7.0]
>>> a.insert (0, 3.0)                # Insert 0.0 in position 0
>>> print (a)
[3.0, 4.0, 5.0, 6.0, 7.0]
>>> print len (a)                    # Determine length of list
5
>>> a [2:4] = [2.0, 2.0, 2.0]        # Modify selected elements
>>> print a
[3.0, 2.0, 2.0, 2.0, 2.0, 7.0]
```

Since list is a mutable object, if we want to create a new reference to the previous variable, such as b = a, any changes made to *b* will be reflected in *a*. However, an independent copy of the original list of *a* can be create use the statement c = a [:]. Thus, any changes made to *c* will not affected in *a*.

```
>>> a = [4.0, 5.0, 6.0]
>>> b = a                # 'b' is new reference of 'a'
>>> b [0] = 7.0          # Change 'b'
>>> print (a)
[7.0, 4.0, 5.0, 6.0]     # the change is reflected in 'a'
>>> c = a [:]            # 'c' is an independent copy of  'a'
>>> c [0] = 5.0          # Change 'c'
>>> print (a)
[7.0, 4.0, 5.0, 6.0]     # 'a' is not affected by the change
```

A nested list also can be implemented in matrices. Here is the example of 2 x 3 matrix *d* and how to print a certain element in the form of a list:

```
>>> a = [[1, 2, 3], [4, 5, 6]]
>>> print a [1]          # Print second row (element 1)
[1, 2, 3]
>>> print a [0][1]       # Print second element of first row
2
```

Operators

Python support some operators, such as arithmetic operators, comparison operators:

Arithmetic Operators		Augmented Assignment Operators		Comparison Operators	
+	Addition	a +=b	a = a + b	<	Less than
-	Subtraction	a -= b	a = a - b	>	Greater than
*	Multiplication	a *= b	a = a * b	<=	Less than or equal to
/	Division	a /= b	a = a / b	>=	Greater than or equal to
**	Exponentiation	a **= b	a = a ** b	==	Equal to
%	Modular division	a %= b	a = a % b	!=	Not equal to

Some of these operators are also used for strings and sequences, as shown below:

```
>>> text1 = ‘ Biomedical’
>>> print (3*text1)                # Repetition
Biomedical Biomedical Biomedical
>>> num = [1, 2, 3]
>>> print (2*num)                  # Repetition
[1, 2, 3, 1, 2, 3]
>>> print (num + [7, 8])           # Append elements
[1, 2, 3, 7, 8]
```

Conditionals

If statements are used to define a condition. If the condition returns true, then a block f statement will be execute. If the condition return false, the block is skipped or continued to next condition (else if). Here is the example of the function if_statement that illustrated a conditional statement:

```
def if_statement (a):              # Defining function
  if a < 0.0 :
    text = 'negative'
  elif a > 0.0:
    text = 'positive'
  else:
    text = 'zero'
  return text

a = -2.5
print ('a is '+ if_statement (a))
```

The result of running the program is:

a is negative

Loops

Looping can be execute using the *While construct* or the *For construct.* The *while construct* will execute a block of statements if the condition is true, then evaluate it. If it is still true, the block continuous to execute until the condition becomes false. The example of the *while construct* is shown as:

```
Target = 5
n = 0
a = [ ]                              # Create empty list
while n < Target :
    a.append ((2*n)/10)       # Append element to list
    n = n + 1
print (a)
```

The output of the program is

[0.0, 0.2, 0.4, 0.6, 0.8]

The *for* statement requires a target and a sequence over which the target loops. The example of the *for* construct is shown as:

```
Target = 5
a = [ ]                              # Create empty list
for n in range (1, Target) :
    a.append ((2*n)/10)       # Append element to list
print (a)
```

The *break* statement can terminate loops if the previous condition has been fulfilled, so the next condition will not be execute. Here is the example of *break* in looping:

```
list = [ ‘Ana , ‘Bona’, ‘Caca’, ‘Dona’ ]
name = eval (input ( 'Type a name: ' ))   # Python input prompt
for i in range (len (list)):
   if list [i] == name:
      print (name, 'is number' , i + 1, 'on the list' )
      break
else:
      print (name, 'is not on the list' )
```

The result of running the program twice using the different name:

```
Type a name: ‘Dona’
Dona is number 4 on the list

Type a name: ‘Toni’
Toni is not on the list
```

Loops can also skip a portion of the statements in an iterative loop by the *continue* statement. The *continue* statement will immediately returns to the beginning of the loop without executing the statements afterwards.

```
num = [ ]                          # Create an empty list
for i in range (1,10):
   if i % 2 != 0 : continue        # if not an even number, skip the rest of loop
      num.append (i)               # Append I to the list
print (num)
```

The printout from the program:

```
[ 2, 4, 6, 8 ]
```

Functions and Modules

Defining a function had been appeared in the previous example by using *def (param1, param2, ...)* statement, where *param1, param2*, ... are the parameters.

Modules contain of useful functions which can be loaded into a program by the statement:

```
from module_name import *
```

(*) means to import all function in the module

Python provided various modules contain function and methods for various task, including arithmetic operations and matrices. Some of modules that frequently used in this practicum are *math* module, *NumPy* module and *SciPy* module. Beside these module, there are a lot of other module that provided by Python. The contents of a module can be printed by write *dir* (*module_name*). Here is the example of how to obtain a list of *Math* and *NumPy* module:

```
>>> import math

>>> dir (math)

['__doc__', '__name__', 'acos', 'asin', 'atan', 'atan2', 'ceil', 'cos', 'cosh', 'e', 'exp', 'fabs', 'floor', 'fmod', 'frexp', 'hypot', 'ldexp', 'log', 'log10', 'modf', 'pi', 'pow', 'sin', 'sinh', 'sqrt', 'tan', 'tanh']

>>> import numpy

>>> dir (numpy)

['complex', 'float', 'abs','append','arccos', 'arccosh', 'arcsin', 'arcsinh', 'arctan', 'arctan2', 'arctanh', 'argmax', 'argmin', 'cos', 'cosh', 'diag', 'diagonal', 'dot', 'e', 'exp', 'floor', 'identity', 'inner, 'inv', 'log', 'log10', 'max', 'min', 'ones', 'outer', 'pi', 'prod' 'sin', 'sinh', 'size', 'solve', 'sqrt', 'sum', 'tan', 'tanh', 'trace', 'transpose', 'zeros', 'vectorize', ...more]
```

Creating an Array

Array is a function that located in *NumPy* module. To create an array we need to import the *array* function from *NumPy* module. Here is the statement:

```
>>> from numpy import array

>>> a = array ( [ [ 1, 2, 3 ], [ 1, 2, 3 ] ] )        # Creating an array
>>> print (a)
[ [ 1   2   3 ]
   [ 1   2   3 ] ]
```

Other way to create an array are using *zeros* (*dim1, dim2*) that creates a *dim1 x dim2* array filled with zeroes and *ones* (*dim1, dim2*) which fills the array with ones.

```
>>> from numpy import *
>>> a = ones ( ( 3, 3), dtype = int )
>>> print (a)
[ [ 1   1   1 ]
   [ 1   1   1 ]
   [ 1   1   1 ]]
>>> a [ 0 ] = [ 2, 5, 7]  # Change a row
>>> a [ 1, 1 ] = 3         # Change an element
>>> a [ 2, 0 : 2 ] = [ -2, 6 ]      #change part of a row
>>> print (a)
[[ 2  5   7 ]
 [ 1   3   1 ]
 [-2   6   1 ]]
```

Operating on Array

Arithmetic operators work differently on arrays than they do on list or list. In arrays, the operation is applied to each element in the array. Here are the illustration:

```
>>> from numpy import *
>>> a = array ([ 1.0,  2.0,  3.0,  4.0 ])
>>> print (a/2)
[ 0.5  1.  1.5   2. ]
>>> print (sqrt (a))
[ 1.     1.41421356     1.73205081      2. ]
>>> print (cos (a))
```

```
[ 0.54030231   -0.41614684   -0.9899925   -0.65364362 ]
>>> print (sqrt ( a [3] )
2.0
```

Array Function

NumPy provides numerous function that perform in array operations. The examples are shown as below:

```
>>> from numpy import *
>>> a = array ( [ [ 1, 2, 3 ], [ 1, 2, 3 ], [ 1, 2, 3 ] ] , dtype = float )
>>> b = array ( [ 1, 2, 3 ], dtype = float )
>>> print (diagonal (a))                        # Principal diagonal
[ 1.   2.   3. ]
>>> print (diagonal ( a, 1 ))           # First sub diagonal
[ 2.   3. ]
>>> print (trace (a))                   # Sum of diagonal elements
6.0
>>> print (argmax (b))                  # Index of largest element
2
>>> print (argmin (a, axis = 0))        # Index of smallest col. Elements
[ 0   0   0 ]
>>> print (identity (3))                # Identity matrix
[ [ 1.   0.   0. ]
  [ 0.   1.   0. ]
  [ 0.   0.   1. ] ]
```

NumPy also allow us to compute array product from its function, such as *Dot product, Inner product,* and *outer product*. Here are the illustration:

```
>>> from numpy import *
>>> x = array ( [ 2, 3 ] )
>>> y = array ( [ 5, 2 ] )
>>> A = array ( [ [ 1, 2 ] , [ 3, 4 ] ] )
>>> B = array ( [ [ 1, 1 ] , [ 2, 2 ] ] )
>>> print ( "dot (x , y) =  \n ", dot (x , y) )              # Dot product {x} . {y}
dot (x , y) =
 16
>>> print ( "dot (A , x) =  \n ", dot (A , x))               # Dot product [A] . {x}
dot (A , x) =
 [ 8   18 ]
>>> print ( "dot (A , B) =  \n ", dot (A , B) )              # Dot product [A] . [B]
dot (A , B) =
 [ [  5     5  ]
   [ 11   11 ] ]
>>> print ( "inner (x , y) =  \n ", inner (x , y) )          # Inner product {x} . {y}
inner (x , y) =
 16
>>> print ( "inner (A , x) =  \n ", inner (A , x))           # Inner product [A] . {x}
inner (A , x) =
 [ 8   18 ]
>>> print ( "inner (A , B) =  \n ", inner (A , B) )          # Inner product [A] . [B]
inner (A , B) =
 [ [  3     6 ]
   [  7   14 ] ]
>>> print ( "outer (x , y) =  \n ", outer (x , y) )          # Outer product {x} . {y}
outer (x , y) =
 [ [ 10     4 ]
   [ 15     6 ] ]
>>> print ( "outer (A , x) =  \n ", outer (A , x))           # Outer product [A] . {x}
outer (A , x) =
  [ [ 2      3 ]
    [ 4      6 ]
    [ 6      9 ]
    [ 8    12 ] ]
```

```
>>> print ( " outer (A , B) =  \n ", outer (A , B) )     # Outer product [A] . [B]
outer (A , B) =
 [ [ 1   1   2   2 ]
   [ 2   2   4   4 ]
   [ 3   3   6   6 ]
   [ 4   4   8   8] ]
```

NumPy comes with a linear algebra module called *linalg* that contains routine tasks such as matrix inversion and solution of simultaneous equations. Here is the example:

```
>>> from numpy import array
>>> from numpy.linalg import inv,solve
>>> A = array ( [ [ 4.0, -2.0, 1.0], \
                  [-2.0, 4.0, -2.0], \
                  [ 1.0, -2.0, 3.0] ] )
>>> b = array ( [ 1.0, 4.0, 2.0 ] )
>>> print inv (A)                          # Matrix inverse
[ [ 0.33333333   0.16666667      0. ]
  [ 0.16666667   0.45833333   0.25 ]
  [ 0.           0.25           0.5   ] ]
>>> print solve ( A, b )                   # Solve [A]{x} = {b}
[ 1. , 2.5,   2. ]
```

Task

1. Display the even number from zero to 100 and calculate the sum of it !

2. Calculate the value of 10! (10 factorial) !

References

1. Kiusalaas J, 2013, ”Numerical Methods in Engineering with Python 3. Vol. 51”, Cambridge University Press, New York.

Chapter 2. Finding-Root Method: Bisection & Regula-Falsi

Aims and Objectives

To determine the roots of function by using Bisection and Regula-Falsi method.

Preliminary Task

If there is a function as follows:

$$f(x) = x - 2\cos x \ \ [0, \pi/2]$$

Please Find the roots in the given intervals by using Bisection and Regula-Falsi method.

Literature Review

Root-finding problem is one of the most basic problems in numerical analysis. If we have a function $f(x)$, the process to find the roots of this function involves finding the value of x while $f(x) = 0$. When the function equals zero, the x value for that function is its root, so-called zero of the function $f(x)$. One of the method to find this root is bisection method or interval halving method or binary search method or dichotomy method.

If the function equals zero, x is the root of the function. A root of the equation $f(x) = 0$ is also called a zero of the function $f(x)$. The Bisection method also called the interval halving method, the binary search method, or the dichotomy method. This method is based on the Bolzano's theorem for continuous functions. Theorem (Bolzano): If a function f(x) is continuous on an interval $[a, b]$ and $f(a)\cdot f(b) < 0$, then a value $c \in (a, b)$ exist for which $f(c) = 0$.

This method is started by choosing two numbers, make interval between them by taking half of them, and make it smaller and smaller. Those two number should have result of the function with opposite sign. As example, there are two numbers, a and b. $f(a)$ and $f(b)$ should have opposite sign so that $f(a).f(b)<0$. Then, the median number r is taken from $(a+b)/2$. If $f(r)$ should be tested by using one of $f(a)$ or $f(b)$. If the result of $f(r).f(a)<0$ then they would be used as the new interval. This process is repeated until the interval reach 0 or near to 0 according to the error. This method is illustrated in Figure 1.

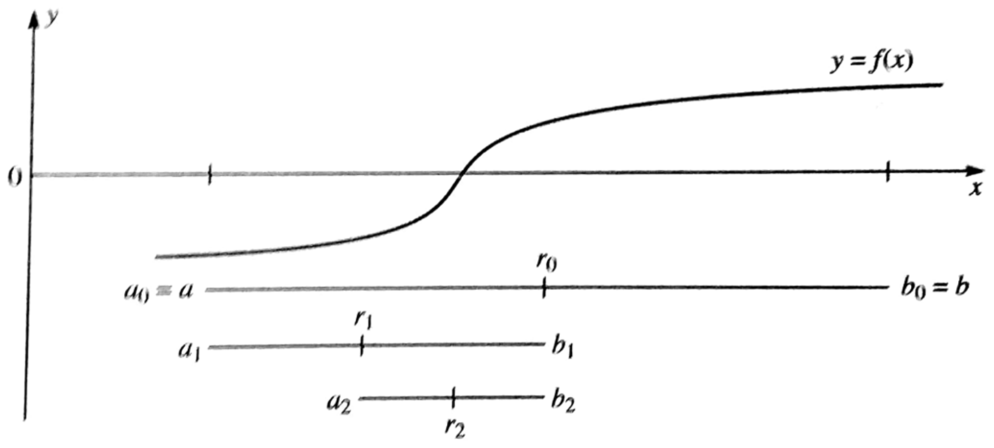


Figure 1. Bisection Method for Finding Roots of a Function

The other method is Regula-Falsi. This method starts with defining initial interval $[a, b]$ so that $f(a) \,.\, f(b) < 0$. In this method, the r point is not exactly in the middle. The r point is slightly close to a or b. The r point is calculated by using equation below:

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

After that, the new interval is determined by using the same method as bisection method which is by checking whether $f(a) \,.\, f(r) < 0$ or $f(b) \,.\, f(r) < 0$. If $f(a) \,.\, f(r) < 0$ then the new interval is a and r, if not, then the new interval is r and b. This process is repeated until the closest value is found by having the smallest error.

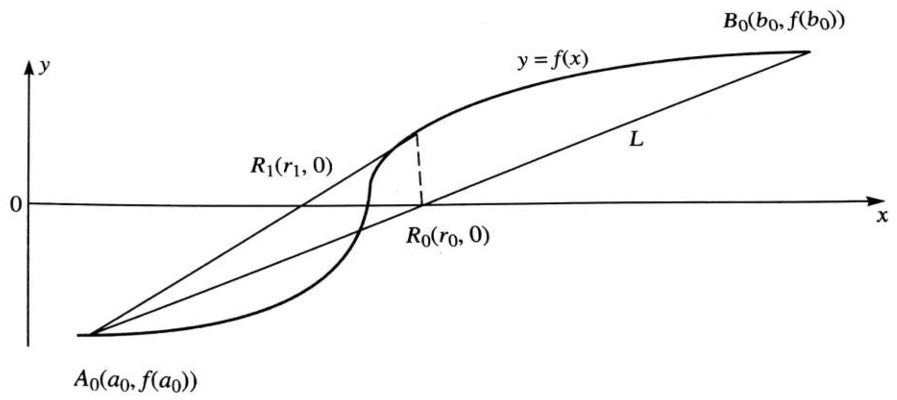


Figure 2. Regula-Falsi method for Finding Root of a Function

FINDING-ROOT METHOD: Bisection & Regula-Falsi

Algorithm

1. Bisection method

   a. Define the function $f(x)$, interval $[a, b]$, max iteration ($maxit$), and max error ($maxer$).
   b. Define n=1
   c. **Repeat** step d-h if $n \leq maxit$
   d. $r = (a+b)/2$
   e. Print n, r, and $f(r)$
   f. **If** $|f(r) \leq maxer|$ or $|a\text{-}b| \leq maxer$, **then** the iteration stop and exit.
   g. **If** $f(r)*f(a) < 0$, then b=r, otherwise $a = r$.
   h. $n = n+1$
   i. Print “algorithm fails: no convergence” and exit.

2. Regula-Falsi method

   a. Define the function $f(x)$, interval $[a, b]$, max iteration ($maxit$), and max error ($maxer$).
   b. Define $n = 1$
   c. **Repeat** step d-h if $n \leq maxit$
   d. $r = a - (f(a)(a - b))/(f(a) - f(b))$
   e. Print n, r, and $f(r)$
   f. **If** $|f(r) \leq maxer|$ or $|a\text{-}b| \leq maxer$, **the**n the iteration stop and exit.
   g. **If** $f(r)*f(a) < 0$, then b=r, otherwise $a = r$.
   h. $n = n+1$
   i. Print “algorithm fails: no convergence” and exit.

Tasks

1. Find the root of equation system in following problem!
   Packed bed column. A column packed with spherical particles provides high surface area geometry that is useful in isolating specific protein(s) or other biological molecules from a cell lysate mixture. The Ergun equation relates the pressure drop through a packed bed of spheres to various fluid and geometric parameters of the bed:

$$\frac{\Delta p}{l} = 150\frac{(1-\varepsilon)^2\mu u}{d_p^2} + 1.75\frac{(1-\varepsilon)\rho u^2}{d_p}$$

Where Δp is the pressure drop, l is the length of the column, ε is the porosity, μ is the fluid viscosity, u is the fluid velocity, d_p is the diameter of the spherical particles, and ρ is the fluid density. For a 20 cm column, packed with 1 mm spheres and perfused with a buffer of equal viscosity and density to water (μ = 0.01 P, ρ = 1 g/cm^3). By using the bisection and Regula-Falsi method, determine the column porosity if the pressure drop for a fluid flowing is 810.5 dyn/cm^2 with velocity u = 0.75 cm/s. Make sure that you use consistent units throughout your calculation.

Using a starting interval of $0.1 < \varepsilon < 0.9$, report the number of iterations necessary to determine the porosity to within 0.01 (tolerance).

2. Analyze the advantages and disadvantages of both methods!

References

1. Capra, Steven C and Canale, 1991, “Numerical Methods for Engineers with Personal Computer Applications”, MacGraw-Hill Book Company.
2. King M.R and Mody N.A, 2010, “Numerical and Statistical Methods for Bioengineering”, Cambridge University Press, New York.
3. Patel VA, 1994, “Numerical Analysis”, Saunders College Publishing.

Chapter 3. Finding-Root Method: Newton-Raphson

Aims and Objectives

To determine the roots of function by using Newton-Raphson method.

Preliminary Task

If there is a function as follows:

$$f(x) = x^3 + x^2 + x + 1$$

Please find the roots in the given intervals by using Newton-Raphson method.

Literature Review

Root-finding problem is one of the most basic problems in numerical analysis. This method uses an approximation method by using one initial point and derive it by taking the slope or gradient on that point. The approximation point is defined as follows:

$$x_{n+1} = x_n - \frac{f(x_n)}{f\prime(x_n)} \tag{3.1}$$

Algorithm

1. Define the function $f(x)$ and the first derivative $f'(x)$
2. Define the error tolerance (e)
3. Define the initial approximation x_0
4. Calculate $f(x_0)$
5. Calculate $x_{n+1} = x_n - \frac{f(x_n)}{f\prime(x_n)}$
6. Calculate $f(x_{n+1})$
7. If $|f(x_n)|$ < tolerance, then the process is stopped and the root is x_n, otherwise **repeat** step 5.

8. If $|x_2 - x_1|$ < tolerance or $|f(x_0)|$ < tolerance, then the process is stopped and the root is x_0, otherwise **repeat** step 5.

Task

1. Find the root of equation system in following problem!

Osteoporosis in Chinese Woman. Wu te la. (2008) studied the variations in age-related speed of sound (SOS) at the tibia and the prevalence of osteoporosis in native Chinese women. They obtained the following relationship between the SOS and the age in year, Y.

$$SOS = 3383 + 39.9Y - 0.78Y^2 + 0.0039Y^3$$

Where the SOS is expressed in unit of m/s. The SOS for one research subject is measured to be 3850 m/s. Use the Newton-Raphson to find the root of the above equation! Take Y=45 years as initial guess!

References

1. Capra, Steven C and Canale, 1991, “Numerical Methods for Engineers with Personal Computer Applications”, MacGraw-Hill Book Company.

2. King M.R and Mody N.A, 2010, “Numerical and Statistical Methods for Bioengineering”, Cambridge University Press, New York.

3. Patel VA, 1994, “Numerical Analysis”, Saunders College Publishing.

Chapter 4. System of Linear Equation: Gauss Elimination

Aims and Objectives

To determine the result of linear equation system by using Gauss Elimination method.

Preliminary Task

If there is a function as follows:

$$a + b + c = 6$$
$$a + 2b - c = 2$$
$$2a + b + 2c = 10$$

Please Find a, b, and c!

Literature Review

The solution of linear equation system is by solving each variable. Several method has been used to solve linear equation system, such as:

1. Gauss Elimination
2. Invers of Matrix

Elementary Row Operations

There are three operations that could be performed towards linear equation systems without changing the real solution, which are:

1. Rearrange the order of the equation.
2. Multiplication an equation by a non-zero number.
3. Change an equation by adding that equation with a multiplication of another equation.

All of them could be applied in the full matrix and is called *Elementary Row Operation* (ERO). By using ERO, the full matrix is changed to a matrix based on linear equation system that is easier to solve. A matrix that has this characteristic is called Echelon Matrix. A matrix is called an echelon matrix if it fulfills two characteristics, which are:

SYSTEM OF LINEAR EQUATION: GAUSS ELIMINATION

1. If there is a row that all of its element are zero, then that row should be placed under the row that has non-zero elements.
2. In the row that has non-zero element, that non-zero element should be placed on the right side of the non-zero element of the previous row (this non-zero element is called the principle element)

The method to solve linear equation system by using Gauss elimination method could be done as follow.

1. Forming a full matrix of linear equation system
2. Changing the full matrix to echelon matrix with several elementary row oerations
3. Obtaining the solution of linear equation system.

As example, if several equations like below are known, then:

a11x + a12y + a13z = b1

a21x + a22y + a23z = b2

a31x + a32y + a33z = b3

The initial matrix:

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

The linear equation system:

$$\left(\begin{array}{ccc|c} a_{11} & a_{12} & a_{13} & b_1 \\ a_{21} & a_{22} & a_{23} & b_2 \\ a_{31} & a_{32} & a_{33} & b_3 \end{array}\right)$$

By using elementary row operations, an echelon matrix is obtained.

z = b3’

y + a23z = b2’ → y = b2’ - a23z

x + a12y + a13z = b1’ → x = b1’ - a12y - a13z

SYSTEM OF LINEAR EQUATION: GAUSS ELIMINATION

Algorithm

1. Define matrix *a,* matrix *b* and the order of the matrix (*n*).

2. Take the coefficients of the linear equation as:

```
    Do for $k$ = 1 to $n$
        Do for $i$ = k+1 to $n$
            $a_{ik} = a_{ik} / a_{kk}$
        Do for $j$ = k+1 to $n$
            $a_{ij} = a_{ij} - a_{ik} a_{kj}$
```

3. Forward elimination:

```
    Do for $k$ = 1 to $n$
        Do for $i$ = k+1 to $n$
            $b_i = b_i - a_{ik} a_{ik}$
```

4. Backward solve:

```
    Do for i = n down to 1 do
        $s = b_i$
        Do for $j = i + 1$ to $n$
            $s = s - a_{ij} x_j$
        $x_i = s / a_{ii}$
```

Task

Find the solution for linear equation system in following problem!

Pharmacokinetic modelling for ”animal-on-chip”

A material balance is performed for naphthalene epoxide (NO) generation, consumption, and transport in the μCCA device described in Figure 4.1; NO is an intermediate formed during the metabolism of naphthalene.

Routes of generation of naphthalene epoxide:

(1) conversion of naphthalene into its epoxide.

Routes of consumption of naphthalene epoxide:

(1) conversion of epoxide to naphthalene dihydrodiol;

(2) binding to GSH to form epoxide–GSH conjugates;

(3) rearrangement to naphthol.

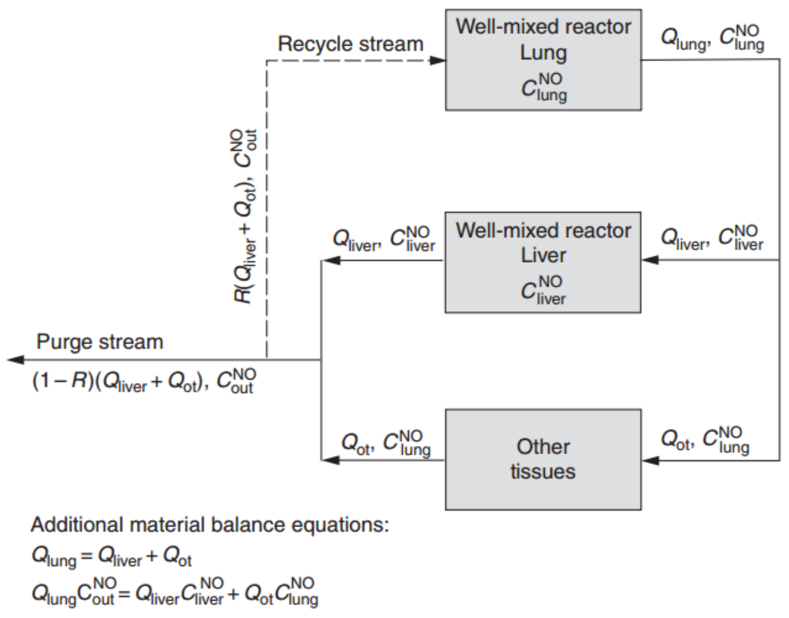

Recycle stream
Well-mixed reactor
Lung
C^{NO}_{lung}
Q_{lung}, C^{NO}_{lung}
$R(Q_{liver}+Q_{ot}), C^{NO}_{out}$
$Q_{liver}, C^{NO}_{liver}$
Well-mixed reactor
Liver
C^{NO}_{liver}
$Q_{liver}, C^{NO}_{liver}$
Purge stream
$(1-R)(Q_{liver}+Q_{ot}), C^{NO}_{out}$
Q_{ot}, C^{NO}_{lung}
Other tissues
Q_{ot}, C^{NO}_{lung}
Additional material balance equations:
$Q_{lung} = Q_{liver} + Q_{ot}$
$Q_{lung}C^{NO}_{out} = Q_{liver}C^{NO}_{liver} + Q_{ot}C^{NO}_{lung}$

Figure 4.1. Material balance diagram for naphtalene epoxide

The material balance diagram for naphthalene epoxide (NO) is shown in Figure 4.1. Since we are dealing with a multicomponent system, we use superscripts N, NO, and NOH for naphthalene, naphthalene epoxide, and naphthol, respectively, to differentiate between the concentration terms in various compartments.

A mass balance of NO is performed over the two chambers – lung and liver. This yields two linear equations in the unknowns C^{NO}_{lung} and C^{NO}_{liver}. Note that simplifications have been made to the original equations (Quick and Shuler, 1999) by assuming that C_{lung}^{NO} and C_{liver}^{NO} are small in comparison to relevant constants present in the equations.

Lung compartment:

$$R\left(Q_{liver}C_{liver}^{NO} + Q_{ot}C_{lung}^{NO}\right) + v_{max,P450-lung}V_{lung} - \frac{v_{max,P450-lung}\,C_{lung}^{NO}}{K_{m,EH-lung}}V_{lung} - V_{lung}\frac{v_{max,GST}C_{lung}^{NO}C_{lung}^{GSH}}{Kl_{lung}+K2_{lung}C_{lung}^{GSH}} - k_{NOH}\exp\left(l_{NOH}TP_{lung}\right)C_{lung}^{NO}V_{lung} - Q_{lung}C_{lung}^{NO} = 0 \qquad (1)$$

Liver compartment:

$$Q_{liver}C_{lung}^{NO} + v_{max,P450-lung}V_{liver} - \frac{v_{max,P450-liver}C_{liver}^{NO}}{K_{m,EH-liver}}V_{liver} - V_{liver}\frac{v_{max,GST}\,C_{liver}^{NO}\,C_{liver}^{GSH}}{Kl_{liver}+K2_{liver}C_{liver}^{GSH}} - k_{NOH}\exp(l_{NOH}TP_{liver})\,C_{liver}^{NO}V_{liver} - Q_{liver}C_{liver}^{NO} = 0 \qquad (2)$$

NO balance assumptions

1. Binding of naphthalene epoxide to proteins is comparatively less important and can be neglected.
2. The concentration of GSH in cells is constant. It is assumed that GSH is resynthesized at the rate of consumption.
3. Production of the RS enantiomer of the epoxide (compared to SR oxide) is dominant, and hence reaction parameters pertaining to RS production only are used.
4. The total protein content in the cells to which the metabolites bind remains constant.

The parametric values and definitions are provided below. The modeling parameters correspond to naphthalene processing in mice.

Flowrates

Q_{lung}: flowrate through lung compartment = 2 μl/min;

Q_{liver}: flowrate through liver compartment = 0.5 μl/min;

Q_{ot}: flowrate through other tissues compartment = 1.5 μl/min.

Compartment volumes

V_{lung}: volume of lung compartment = 2 mm ×2mm × 20 μm = 8 ×10l = 0.08μl;

V_{liver}: volume of liver compartment = 3.5 mm ×4.6 mm ×20 μm = 3.22 × 10-7l = 0.322 μl.

Reaction constants

(1) Naphthalene → naphthalene epoxide

$v_{max,P450\text{-}lung}$: maximum reaction velocity for conversion of naphthalene into napthalene epoxide by cytochrome P450 monooxygenases in lung cells = 8.75 μM/min;

$v_{max,P450\text{-}liver}$: maximum reaction velocity for conversion of naphthalene into napthalene epoxide by cytochrome P450 monooxygenases in liver cells = 118 μM/min.

(2) Naphthalene epoxide → naphthalene dihydrodiol

$v_{max,EH\text{-}lung}$: maximum reaction velocity for conversion of naphthalene epoxide to dihydrodiol by epoxide hydrolase in the lung = 26.5 μM/min;

$K_{m,EH\text{-}lung}$: Michaelis constant = 4.0 μM;

$v_{max,EH\text{-}liver}$: maximum reaction velocity for conversion of naphthalene epoxide to dihydrodiol by epoxide hydrolase in the liver = 336 μM/min;

$K_{m,EH\text{-}lung}$: Michaelis constant = 21 μM

(3) Naphthalene epoxide → naphthol

k_{NOH}: rate constant for rearrangement of epoxide to naphthol = 0.173 μM/μM of NO/min;

l_{NOH}: constant that relates naphthol formation rate to total protein content = − 20.2 ml/g protein

(4) Naphthalene epoxide → epoxide–GSH conjugates

$v_{max,\ GST}$: maximum reaction velocity for binding of naphthalene epoxide to GSH catalyzed by GST (glutathione S-transferase) = 2750 μM/min;

$K1_{lung}$: constant in epoxide–GSH binding rate = 310 000 μM2;

$K2_{lung}$: constant in epoxide–GSH binding rate = 35 μM;

$K1_{liver}$: constant in epoxide–GSH binding rate = 150 000 μM2;

$K2_{liver}$: constant in epoxide–GSH binding rate = 35 μM.

Protein concentrations

TP_{lung}: total protein content in lung compartment = 92 mg/ml;

TP_{liver}: total protein content in liver compartment = 192 mg/ml;

C_{lung}^{GSH}: GSH concentration in lung compartment = 1800 μM;

C_{liver}^{GSH}: GSH concentration in liver compartment = 7500 μM;

R: fraction of the exiting stream that reenters the microcircuit.

Your goal is to vary the recycle fraction from 0.6 to 0.95 in increasing increments of 0.05 in order to study the effect of reduced excretion of toxicant on circulating concentration values of naphthalene and its primary metabolite naphthalene epoxide.

a. Use the Gaussian elimination method to determine the napthalene epoxide concentrations at the outlet of the lung and liver compartments of the animalon-a-chip for the range of R specified.

b. Plot the concentration values of epoxide in the liver and lung chambers as a function of R.

References

1. Capra, Steven C and Canale, 1991, "Numerical Methods for Engineers with Personal Computer Applications", MacGraw-Hill Book Company.

2. King M.R and Mody N.A, 2010, "Numerical and Statistical Methods for Bioengineering", Cambridge University Press, New York.

3. Patel VA, 1994, "Numerical Analysis", Saunders College Publishing.

Chapter 5. System of Linear Equation: Jacobi & Gauss Seidel Iteration

Aims and Objectives

To determine the solutions of linear equation system by using Jacobi iteration and Gauss-Seidel iteration method.

Preliminary Task

There are several linear equations, as follows:

$2x_1 + x_2 - 5x_3 = 9$

$x_1 - 5x_2 - x_3 = 14$

$7x_1 - x_2 - 3x_3 = 26$

Define x1, x2, and x3 by using Jacobi iteration and Gauss-Seidel Method (x1(0)=1, x2(0)=x3(0)=2, do the iteration until the third iteration)

Literature Review

To minimize the error in the rounding process in Gauss elimination method, the linear equation system could be solved by using iteration method. There are two types of iteration method, which are Jacobi iteration method and Gauss-Seidel iteration method.

The general form of linear equations are shown below.

$a_{11} x_1 + a_{12} x_2 \ldots\ldots\ldots\ldots + a_{1n} x_n = b_1$

$a_{21} x_1 + a_{22} x_2 \ldots\ldots\ldots\ldots + a_{2n} x_n = b_2$

$a_{21} x_1 + a_{32} x_2 \ldots\ldots\ldots\ldots + a_{3n} x_n = b_3$

$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$

$a_{n1} x_1 + a_{n2} x_2 \ldots\ldots\ldots\ldots + a_{nn} x_n = b_n$

with $a_{kk} \neq 0$, k=1,2,……,n then the iteration equation could be written as follows.

$$x_n^{k+1} = \frac{b_n - (a_{n1}x_1^k + a_{n2}x_2^k + \cdots + a_{nn-1}x_{n-1}^k)}{a_{nn}} \tag{5.1}$$

As a stop iterative condition, the relative error equation below could be used.

$$\left|\frac{x_i^{k+1}-x_i^k}{x_i^{k+1}}\right| < \varepsilon \qquad (5.2)$$

The requirement to get a convergent iteration is $|a_{ii}| > \sum_{j=1,j\neq i}^{n} |a_{ij}|$

1. Jacobi Iteration Method

   If the initial guess is $X^{(0)}$,

   $X^{(0)} = (x_1^{(0)}, x_2^{(0)}, \dots, x_n^{(0)})$

   then the iteration procedure is defined by using the following equation.

$$x_i^{k+1} = \frac{b_i-\sum_{j=i,j\neq i}^{n} a_{ij}x_j^k}{a_{ii}} \text{ , } k=0, 1, 2, \dots \qquad (5.3)$$

2. Gauss-Seidel Iteration Method

   In the Gauss-Seidel iteration method, every new x that is just obtained is used for the next equation immediately. The iteration procedure is defined by using the following equation.

$$x_i^{k+1} = \frac{b_i-\sum_{j=i,j\neq i}^{n} a_{ij}x_j^{k+1} - \sum_{j=i+1}^{n} a_{ij}x_j^k}{a_{ii}} \text{ , } k=0, 1, 2, \dots \qquad (5.4)$$

Algorithm

1. Input the dimension of the matrix.
2. Input matrix A and B.
3. Define the error limit.
4. Define the initial value of x_i, for i=1 to n
5. K=1
6. As long as ($\left|\frac{x_i^{k+1}-x_i^k}{x_i^{k+1}}\right| > error$) **Do** step 6 to 15
7. For i = 1 to n, Do the steps 7 to 13

```
8.  Sum=0
9.  For $j$ = 1 to $n$, **do** step 9 to 10
10. If $j$ ~= I, **do** step 10
11. sum = sum + $a_{i,j}$ * $x_j$
12. $x_i$ = ($b_i$ - sum) /$a_{i,i}$
13. Calculate $\left|\frac{x_i^{k+1}-x_i^k}{x_i^{k+1}}\right|$
14. Print ($k$, $x_1$, $x_2$, … , $x_n$)
15. $k$=$k$+1
```

Task

1. Define a biomedical problem of "Drug development and toxicity studies: animal-on-a-chip". (Reference: King M.R and Mody N.A , 2010, "Numerical and Statistical Methods for Bioengineering", Cambridge University Press, New York , page 48) by using Jacobi iteration method and Gauss-Seidel Iteration method! Analyze the advantages and disadvantages of both methods!

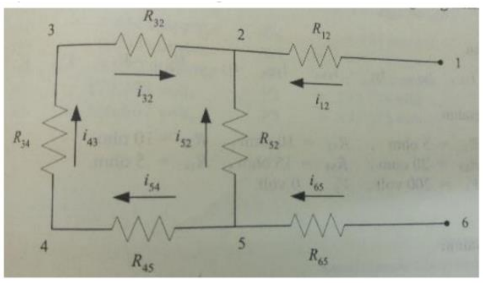
2. Construction of Diet

   A doctor suggest a patient to follow a diet program based on the table below.

Amounts (gr) supplied per 100 gr of ingredients				Amounts (gr) supplied by Cambridge Diet in One Day
Nutrient	Non-fat milk	Soy flour	Whey	
Protein	36	51	13	33
Carbohydrate	52	34	74	45
Fat	0	7	11	3

   Please find how much non-fat milk-soy flour, and whey that are needed to fulfill the amounts of protein, carbohydrate, and fat each day ideally?

3. Electrical Circuits



Please define i_{12}, i_{52}, i_{32}, i_{65}, i_{54}, i_{13}, V_2, V_3, V_4, V_5, if the following information is known.

$R_{12} = 5\ \Omega$; $R_{23} = 10\ \Omega$; $R_{34} = 5\ \Omega$;

$R_{45} = 15\ \Omega$; $R_{52} = 10\ \Omega$; $R_{65} = 20\ \Omega$;

$V_1 = 200$ V; $V_6 = 0$ V

References

1. Capra, Steven C and Canale, 1991, "Numerical Methods for Engineers with Personal Computer Applications", MacGraw-Hill Book Company.

2. King M.R and Mody N.A , 2010, "Numerical and Statistical Methods for Bioengineering", Cambridge University Press, New York

3. Munir R, 2003, "Metode Numerik", Informatika Bandung.

Chapter 6. Regression: Linear & Polynomials

Aims and Objectives

To define the correlation of variables in a function by using regression method.

Preliminary Task

There are data of variable x and y, such as:

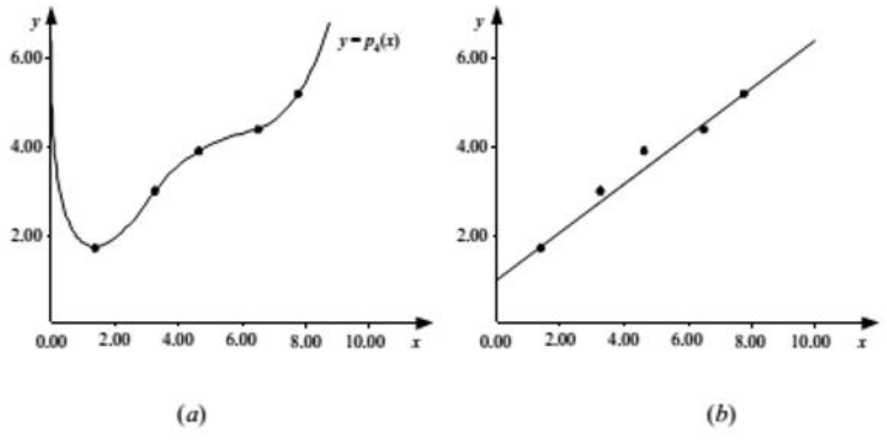
A. x=[1, 3, 5, 7, 10, 12, 13, 16, 18, 20]
   y=[3, 2, 6, 5, 8, 7, 10, 8, 12, 10]

B. x=[0.1, 0.3, 0.5, 0.7, 0.9, 1.1, 1.3]
   y=[0.003, 0.067, 0.148, 0.248, 0.370, 0.518, 0.697]

If y = f(x) is a linear function, define the equation of that function by using linear regression method!

Literature Review

Regression is a method to obtain an approximate mathematical model (function) of two dependent variables. It could produce a linear, polynomial and exponential function regarding the assumption of the user as shown in Figure 6.1.



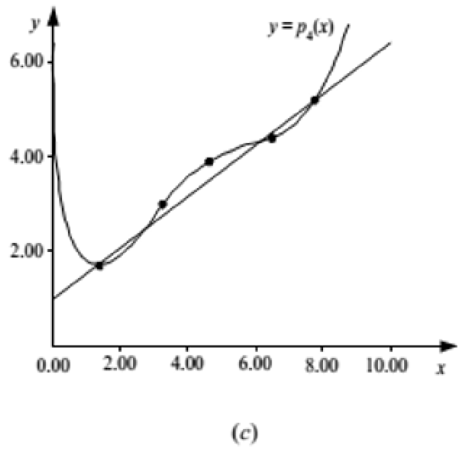


Figure 6.1. The example of regression method (a) the data was accustomed to Lagrangian polynomial function fourth order (b) the data was accustomed to a linear line, and (c) the comparison of both graphs

Linear regression is a regression method that produces a function as a correlation between two linear dependent variables. As example, xi and yi are the data from measurement. We would track those data by using a straight line. That straight line was made so that the error is relatively small. That process was illustrated in Figure 6.2.

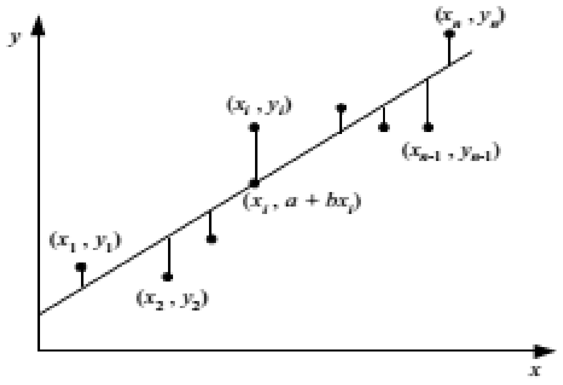


Figure 6.2. Linear Regression

Mathematically, the linear function is written as follows.

$$f(x) = a + bx$$

Thus, we need to find a and b that could be determined by solving normal equations shown below.

$$\begin{bmatrix} n & \sum x_i \\ \sum x_i & \sum {x_i}^2 \end{bmatrix} = \begin{bmatrix} a \\ b \end{bmatrix} \begin{bmatrix} \sum y_i \\ \sum x_i y_i \end{bmatrix}$$

By solving that matrix equation, we could obtain a and b, like the equation shown below.

$$b = \frac{n \sum x_i y_i - \sum x_i \sum y_i}{n \sum {x_i}^2 - (\sum x_i)^2}$$

$$a = y - bx$$

Algorithm

1. Input data x and y
2. Define the length of the data n of x and y
3. Calculate $\sum x_i$
4. Calculate $\sum y_i$
5. Calculate $\sum x_i^2$
6. Calculate $\sum x_i y_i$
7. Make matrix $M = \begin{bmatrix} n & \sum x_i \\ \sum x_i & \sum {x_i}^2 \end{bmatrix}$
8. Make matrix $N = \begin{bmatrix} \sum y_i \\ \sum x_i y_i \end{bmatrix}$
9. Calculate $\begin{bmatrix} a \\ b \end{bmatrix} = M^{-1} \cdot N$
10. Plot the linear regression with a function = $ax + b$

REGRESSION: Linear & Polynomials

Task

1. Make a linear regression program based on data as shown below.

X	8	17	20	25	31	42	50	59	65	72	80
Y	100	130	209	276	330	359	420	487	550	645	700

2. **Using hemoglobin as a blood substitute: hemoglobin–oxygen binding**
   Hemoglobin (Hb) is a protein present in red blood cells that is responsible for the transport of oxygen (O_2) from lungs to individual tissues throughout the body and removal of carbon dioxide (CO_2) from the tissue spaces for transport to the lungs. The hemoglobin molecule is a tetramer and consists of four subunits, two α chains, and two β chains. Each α or β polypeptide chain contains a single iron atom containing heme group that can bind to one O_2 molecule. Thus, a hemoglobin molecule can bind up to four O_2 molecules. The subunits work cooperatively with each other (allosteric binding), such that the binding of one O_2 molecule to one of the four subunits produces a conformational change within the protein that makes O_2 binding to the other subunits more favorable. The binding equation between hemoglobin and oxygen is as follows:

$$Hb(O_2)_n \leftrightarrow Hb + O_2; \text{ where n = 1, …, 4}$$

The exchange of O_2 and CO_2 gases between the lungs and tissue spaces via the blood occurs due to prevailing differences in the partial pressures of pO_2 and pCO_2, respectively. The atmospheric air contains 21% O_2. The O_2 in inspired air exerts a partial pressure of 158 mm Hg (millimeters of mercury), which then reduces to 100 mm Hg when the inspired air mixes with the alveolar air, which is rich in CO_2. Venous blood that contacts the alveoli contains O_2 at a partial pressure of 40 mm Hg. The large difference in partial pressure drives diffusion of O_2 across the alveolar membranes into blood. Most of the oxygen in blood enters into the red blood cells and binds to hemoglobin molecules to form oxyhemoglobin. The oxygenated blood travels to various parts of the body and releases oxygen from oxyhemoglobin. The partial pressure of oxygen in the tissue spaces depends on the activity level of the tissues and is lower in more active tissues. The high levels of CO_2 in the surrounding tissue drive entry of CO_2 into blood and subsequent reactions with water to produce bicarbonates (HCO_3^-). Some of the bicarbonates enter the red blood cells and bind to hemoglobin to form carbaminohemoglobin. At the lungs, the oxygenation of blood is responsible for the transformation of carbaminohemoglobin to oxyhemoglobin or the dissociation of CO_2

from hemoglobin, and conversion of bicarbonates into CO_2 and H_2O, resulting in CO_2 leaving the blood and entering into the lungs.

The Hill equation describes a mathematical relationship between the extent of oxygen saturation of hemoglobin and the partial pressure of O_2 in blood. This equation is derived from the application of the law of mass action to the state of chemical equilibrium of the reaction

$$Hb(O_2)_n \leftrightarrow Hb + O_2$$

$$S = \frac{[Hb(O_2)_n]}{[Hb(O_2)_n] + [Hb]} = \frac{(pO_2)^n}{P_{50}^n + (pO_2)^n} \quad (1)$$

Equation (1) is the Hill equation, and accounts for the observed cooperative binding that produces the characteristic sigmoidal shape of the hemoglobin–oxygen dissociation curve.

The partial pressure of oxygen that results in occupancy of half of the oxygen binding sites of hemoglobin is denoted by P_{50}.

Biochemical engineers at several universities have explored the use of polymerized bovine hemoglobin as a possible blood substitute (Levy et al., 2002). Monomeric hemoglobin is small enough to leak out of capillary pores; however, by chemically attaching several hemoglobin units together, the end product is large enough to be retained within the circulatory system. Data collected for tetrameric bovine hemoglobin binding to oxygen are given in Table 2.1.

We first examine the shape of the oxygen dissociation curve by plotting the data in Figure 2.5.

We wish to determine the best-fit values of P_{50} and n in the Hill equation for this data set and compare with values measured for human hemoglobin. Equation (1) is nonlinear but can be converted to linear form by first rearranging to:

$$\frac{S}{1-S} = \frac{(pO_2)^n}{P_{50}^n} \quad (2)$$

And then taking the logarithm of both sides to obtain:

$$ln\frac{S}{1-S} = n\, ln(pO_2) - n \ln P_{50} \quad (3)$$

Plotting ln $(S/(1-S))$ as a function of $\ln(pO_2)$ produces a line with slope n and intercept -n ln P_{50}.

Equation (3) is the functional form for the dependency of oxygen saturation of hemoglobin as a function of the oxygen partial pressure, and is linear in the regression parameters.

Tabel 1. Fractional saturation of hemoglobin as a function of partial pressure of oxygen in blood

pO_2 (mmHg)	10	20	30	40	50	60	70	80	90	100	110	120
S	0.18	0.40	0.65	0.80	0.87	0.92	0.94	0.95	0.95	0.96	0.96	0.97

Question:

a. Please derive equation (1) to equation (3).
b. Make a program of linear regression based on the data above by following the given information. (Remember: the equation is non-linear at first, so it is needed to be linearized first. Then, input the data in the linearized equation to get the unknown variables).
c. Find what is n and P_{50}
d. Plot the data before and after linearization.

3. Make a polynomial regression program to solve a nonlinear data problem.

References

1. Capra, Steven C and Canale, 1991, "Numerical Methods for Engineers with Personal Computer Applications", MacGraw-Hill Book Company.

2. King M.R and Mody N.A, 2010, "Numerical and Statistical Methods for Bioengineering", Cambridge University Press, New York.

Chapter 7. Interpolation Lagrange

Aims and Objectives

To determine an interpolation based on a given data.

Preliminary Task

Make an Lagrange interpolation of the following data: (0,1), (1,2), (3,4), (6,-1)

Literature Review

Lagrange interpolation is used to find several connecting dots of n dots $P_1(x_1, y_1)$, P_2(x2,y2), $P_3(x_3, y_3)$, … , $P_N(x_N, y_N)$ by using polynomial function approximation that is arranged in a row combination and defined as follows.

$$y = \sum_{i=1}^{N} y_i \prod_{j \neq i} \frac{(x - x_j)}{(x_i - x_j)}$$

Algorithm

1. Define the number of dots (n)

2. Define the dots Pi(xi,yi) with i=1, 2, 3, …, n

3. Define the input x

4. Calculate the y by using the Lagrange interpolation equation:

$$y = \sum_{i=1}^{N} y_i \prod_{j \neq i} \frac{(x - x_j)}{(x_i - x_j)}$$

5. Print (x, y)

INTERPOLATION

Task

There is a data as shown below.

n	1	2	3	4	5	6	7
$x(n)$	0.1	0.3	0.5	0.7	0.9	1.1	1.3
$y(n)$	0.030	0.067	0.148	0.248	0.320	0.518	0.697

1. Define the order of the polynomial Lagrange interpolation that exactly goes through the seven dots shown above and Plot the function!

2. Guess the y for each dot in the table below and show them as well in the graph that has been made!

x(n)	y(n)
0.365	
0.512	
0.621	
0.715	

3. What does happen if the *x* value is not in the range of the data of Lagrange interpolation? Elaborate your answer and show it in a plot!

4. What is the difference of linear regression and Lagrange interpolation?

5. Give one example of biomedical case that could use Lagrange interpolation to solve it! Provide the data and the result of the Lagrange interpolation!

References

1. Capra, Steven C and Canale, 1991, "Numerical Methods for Engineers with Personal Computer Applications", MacGraw-Hill Book Company.

2. King M.R and Mody N.A, 2010, "Numerical and Statistical Methods for Bioengineering", Cambridge University Press, New York.

Chapter 8. Numerical Derivative: Finite Difference Approximations

Aims and Objectives

To determine a numerical derivative using forward, backward and central difference approximation.

Preliminary Task

Define the analytical value of $f'(1)$ from a function $f(x) = x \sin(x)$!

Literature Review

If there are some points of x at $x_0 - h$, x_0, and $x_0 + h$ and also their result of a function as well, several points would be obtained, such as (x_{-1}, f_{-1}), (x_0, f_0) and (x_1, f_1), with $x_{-1} = x_0 - h$ and $x_1 = x_0 + h$. There are three approximations to calculate the $f'(x_0)$ as follows.

a. Forward difference approximation

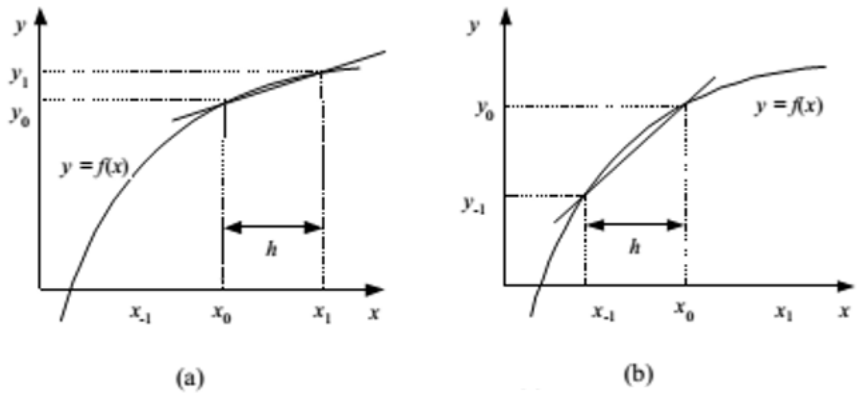
$$f'(x_0) = \frac{f(x_0 + h) - f(x_0)}{h} = \frac{f_1 - f_0}{h}$$

b. Backward difference approximation

$$f'(x_0) = \frac{f(x_0) - f(x_0 - h)}{h} = \frac{f_0 - f_1}{h}$$

c. Central difference approximation

$$f'(x_0) = \frac{f(x_0 + h) - f(x_0 - h)}{2h} = \frac{f_1 - f_{-1}}{2h}$$



NUMERICAL DERIVATIVE

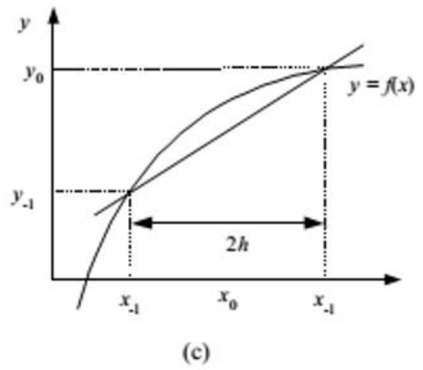


Figure 8.1. Three type of derivative approximation (a) forward difference approximation (b) backward difference approximation, and (c) central difference approximation

Algorithm

1. Define the function that would be derived, such as $f(x)$
2. Define the value of h
3. Define the value of x that would be calculated, like x_0
4. Calculate $f(x_0 - h)$, $f(x_0)$, and $f(x_0 + h)$
5. Calculate $f'(x_0)$ by using forward, backward and central difference approximation
6. Show the derivative value of every approximation.

Task

1. Explain the effect of the change of h towards the error of numerical derivative calculation and give the reason to that!
2. Modify the program that you make to solve the following problem. Calculate $f'(1.5)$ if the available points are (1.2, 0.8333), (1.4, 0.7143), (1.6, 0.6250), and (1.8, 0.5556).

References

1. Capra, Steven C and Canale, 1991, "Numerical Methods for Engineers with Personal Computer Applications", MacGraw-Hill Book Company.
2. King M.R and Mody N.A, 2010, "Numerical and Statistical Methods for Bioengineering", Cambridge University Press, New York.

Chapter 9. Numerical Integration: Trapezoid & Simpson's Rule

Aims and Objectives

To determine a numerical integration using Trapezium and Simpson 1/3 method!

Preliminary Task

If there is function as follows:

$f(x) = x^2 \cos x^2$, $1.5 \leq x \leq 2.5$ and h =0,1.

Calculate the following integration by using trapezium and Simpson 1/3 method!

$$\int_0^2 f(x)dx$$

Literature Review

The analytical integration calculation was performed by using discrete points that were divided into several parts called "segment". There are several method that use segment method, such as trapezoid method and rectangular method. The rectangular method divide the area below the curve into several rectangle segments as shown in Figure 9.1.

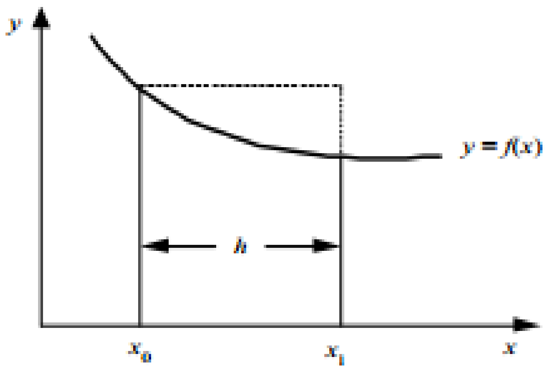


Figure 9.1. Rectangular method for numerical integration

The area of one rectangle is

$$\int_{x_0}^{x_1} f(x)dx \approx h\,f(x_0) \qquad \int_{x_0}^{x_1} f(x)dx \approx h\,f(x_1)$$

$$\int_{x_0}^{x_1} f(x)dx \approx \frac{h}{2}\,[f(x_0) + f(x_1)]$$

Those equations could be simplified as follows.

$$\int_{a}^{b} f(x)dx \approx \frac{h}{2}\,(f_0 + 2f_1 + 2f_2 + \ldots + 2f_{n-1} + f_n) = \frac{h}{2}(f_0 + 2\sum_{i=1}^{n-1} f_i + f_n)$$

The trapezoid method is almost the same as rectangular method. The shape that is used for dividing the area below the curve is a trapezium. By using this method, it is expected to minimize the error produced by the shape of rectangle in rectangular method. The equation for trapezoid method is illustrated in Figure 9.2.

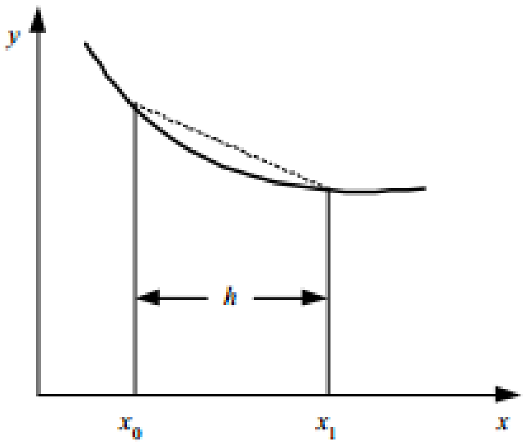


Figure 9.2. Trapezoid method for numerical integration

$$\begin{aligned}\int_a^b f(x)dx &\approx \int_{x_0}^{x_1} f(x)dx + \int_{x_1}^{x_2} f(x)dx + \ldots + \int_{x_{n-1}}^{x_n} f(x)dx \\ &\approx \frac{h}{2}[f(x_0)+f(x_1)] + \frac{h}{2}[f(x_1)+f(x_2)] + \ldots + \frac{h}{2}[f(x_{n-1})+f(x_n)] \\ &\approx \frac{h}{2}[f(x_0)+2f(x_1)+2f(x_2)+\ldots+2f(x_{n-1})+f(x_n)] \\ &\approx \frac{h}{2}\left(f_0 + 2\sum_{i=1}^{n-1} f_i + f_n\right)\end{aligned}$$

Besides the “segment” method, there is another method to calculate the integration of a function, such as Newton Cotes method. This method is used polynomial interpolation, such as trapezoid method, Simpson 1/3 method, and Simpson 3/8 method.

Simpson 1/3 method needs at least 3 points to determine the approximation of the integration of a function, as example $(0, f(0))$, $(h, f(h))$, dan $(2h, f(2h))$ as shown in Figure 9. 3.

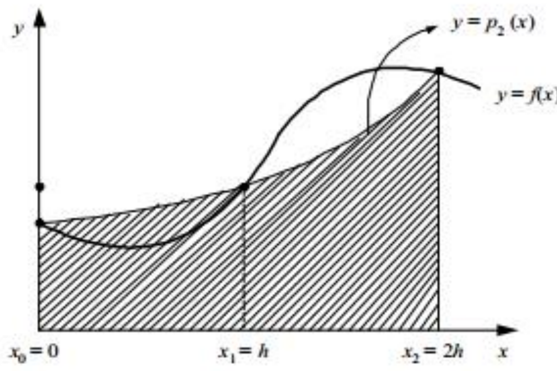


Figure 9.3. Simpson 1/3 Method for numerical integration

$$\begin{aligned}I_{tot} = \int_a^b f(x)dx &\approx \int_{x_0}^{x_2} f(x)dx + \int_{x_2}^{x_4} f(x)dx + \ldots + \int_{x_{n-2}}^{x_n} f(x)dx \\ &\approx \frac{h}{3}\left(f_0 + 4\sum_{i=1,3,5}^{n-1} f_i + 2\sum_{i=2,4,6}^{n-2} f_i + f_n\right)\end{aligned}$$

NUMERICAL INTEGRATION

Task

1. A sky-diver falls from a plane with a velocity as shown in equation below.

$$v(t) = \frac{gm}{c}\left(1 - e^{-(c/m)t}\right)$$

v = velocity (m/s)

g = gravitational acceleration = 9.8 m/s^2

m = sky-diver mass = 78.5 kg

c = air resistive coefficient = 12.5 kg/s

Please, calculate how far the sky diver is fell after 12 seconds with several numerical integration methods and compare the result of each method to each other. Define the error compared to the analytical result. Which method is the most accurate? Explain!

2. **IV Drip**

A cylindrical bag of radius 10 cm and height 50 cm contains saline solution (0.9% NaCl) that is provided to a patient as a peripheral intravenous (IV) drip. The method used to infuse saline into the patient's bloodstream in this example is gravity drip. The solution flows from the bag downwards under the action of gravity through a hole of radius 1 mm at the bottom of the bag. If the only resistance offered by the flow system is the viscous resistance through the tubing, determine the time it will take for the bag to empty by 90%. The length of the tubing is 36″ (91.44 cm) and its ID (inner diameter) is 1 mm. The viscosity of the saline solution μ = 0.01 Poise, and the density ρ = 1 g/cm^3. Let L be the length of the tubing, d the diameter of the tubing, and R the radius of the cylindrical bag. Then, L = 91.44 cm; d = 0.1 cm; R = 10 cm.

Mechanical energy balance

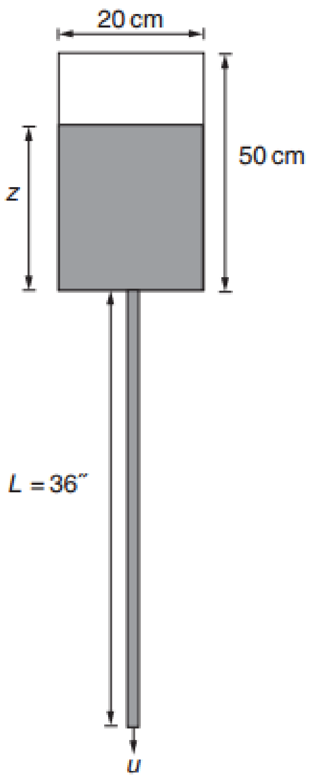
The mechanical energy balance is given by the Bernoulli equation corrected for loss of mechanical energy due to fluid friction. The mechanical energy at the liquid level in the bag is equal to the mechanical energy of the fluid flowing out of the tubing minus the frictional loss. We ignore the friction loss from sudden contraction of the cross-sectional area of flow at the tube entrance. The height at which the drip fluid enters the patient is considered the datum level. The height of the fluid in the bag is z cm with respect to the bottom of the bag, and is (z + L) cm with respect to the datum level (see
Figure 6.12).

The pressure drop in the tubing due to wall friction is given by the Hagen–Poiseuille equation:

$$\Delta p = \frac{32Lu\mu}{d^2} = 2926.08u$$

The Bernoulli equation for this system is as follows:

$$g(z+L) = \frac{u^2}{2} + \frac{\Delta p}{\rho} = \frac{u^2}{2} + \frac{32Lu\mu}{d^2\rho},$$



Where $\Delta p/\rho$ is the term for mechanical loss due to fluid friction and has units of energy per mass.

Steady state mass balance

There is no reaction, accumulation, or depletion within the system, and we ignore the initial unsteady flow when initiating the IV drip. The mass flow rate in the bag is equal to the mass flow rate in the tubing:

$$-\rho\pi R^2\frac{dz}{dt} = \rho\frac{\pi}{4}d^2u.$$

On rearranging, we get

$$dt = -\frac{4R^2}{d^2u}dz.$$

We can express u in terms of z by solving Equation (6.24) for u.

Letting a=64Lμ/d$^2\rho$

$$u = \frac{-a+\sqrt{a^2+8g(z+L)}}{2}.$$

Substituting the above into the steady state mass balance,

$$dt = -\frac{8R^2}{\left(-a+\sqrt{a^2+8g(z+L)}\right)d^2}dz.$$

Integrating the differential equation

We wish to integrate z from 50 cm to 5 cm. Integrating the left-hand side from 0 to t, we obtain:

$$t = \int_{50}^{5} -\frac{8R^2}{\left(-a+\sqrt{a^2+8g(z+L)}\right)d^2}dz$$
$$= \frac{8R^2}{d^2}\int_{5}^{50}\frac{1}{\left(-a+\sqrt{a^2+8g(z+L)}\right)}dz$$

Question:

a. Calculate the time that the IV Drip needed to be changed! (When the z = 5 cm). Use all integration methods for this problem!

3. **Polymer cyclization (Jacobson– Stockmayer theory)**

Polymer chains sample a large number of different orientations in space and time. The motion of a polymer chain is governed by stochastic (probabilistic) processes. Sometimes, the two ends of a linear polymer chain can approach each other within a reactive distance. If a bond forms between the two polymer ends, the reaction is termed as cyclization. By studying the probability with which this occurs one can estimate the rate at which a linear chain is converted to a circular chain. Consider a linear polymer with N links. The probability that the two ends of the chain come within a bond distance b of each other is given by the following integral:

$$\left[\frac{3}{2\pi N b^2}\right]^{3/2} \int_0^b \exp\left(\frac{-3r^2}{2Nb^2}\right) 4\pi r^2 dr.$$

If N = 20 links and b = 1, calculate the probability the chain ends come within a distance b of each other. Use a two-segment composite trapezoidal rule and then a four-segment composite trapezoidal rule. Use these two approximations to extrapolate to an even more accurate solution by eliminating the O(h2) error term, where h is the panel width.

References

1. Capra, Steven C and Canale, 1991, “Numerical Methods for Engineers with Personal Computer Applications”, MacGraw-Hill Book Company.

2. King M.R and Mody N.A , 2010, “Numerical and Statistical Methods for Bioengineering”, Cambridge University Press, New York

Chapter 10. Richardson Extrapolation: Derivatives

Aims and Objectives

To determine a numerical derivative using Richardson extrapolation and knowing its strengths and weaknesses compare to other numerical derivative methods.

Preliminary Task

Given the data points:

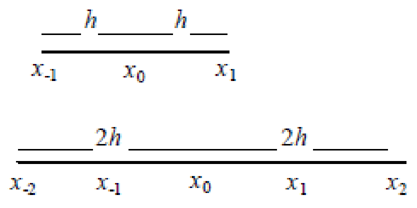
x	2.0	2.1	2.2	2.3	2.4	2.5
***f*(*x*)**	0.42298	0.40051	0.37507	0.34718	0.31729	0.28587

x	2.6	2.7	2.8	2.9	3.0
***f*(*x*)**	0.25773	0.22008	0.18649	0.15290	0.11963

Determine $f'(2.5)$ by extrapolating Richardson if $D(h)$ and $D(2h)$ are calculated by the formula of the center-difference order $O(h^2)$ to 5 significant figure!

Literature Review

Richardson extrapolation is a numerical method for boosting accuracy. Richardson extrapolation can be applied in numerical derivatives to obtain a more accurate solution. Suppose that $D(h)$ and $D(2h)$ are the approximation of $f'(x_0)$ by taking the points respectively a distance of h and $2h$. For example, to calculate $f'(x_0)$, the formula for center-difference approximation $O(h^2)$ is used:



$$D(h) = \frac{(f_1 - f_{-1})}{2h} + O\left(h^2\right)$$

$$= f_0' + Ch^2 + \cdots \tag{10.1}$$

$$D(2h) = \frac{(f_2 - f_{-2})}{2(2h)} + O((2h)^2)$$

$$= f_0' + C(2h)^2 + \cdots$$

$$= f_0' + 4Ch^2 + \cdots \tag{10.2}$$

From equation (10.1) and (10.2), we could obtain:

$$D(h) - D(2h) = -3Ch_2 \tag{10.3}$$

$$C = \frac{D(h)-D(2h)}{-3h^2} \tag{10.4}$$

Distribute (10.4) into equation (10.1):

$$D(h) = f_0' + \frac{[D(h)-D(2h)]h^2}{-3h^2}$$

$$= f_0' - {}^1/_3\,[D(h) - D(2h)] \tag{10.5}$$

or

$$f_0' = D(h) + {}^1/_3\,[D(h) - D(2h)] \tag{10.6}$$

Richardson extrapolation can be expanded to improve derivative functions. Based on equation (10.6) the above rules can be written:

$$f_0' = D(h) + \frac{1}{2^n-1}[D(h) - D(2h)] \tag{10.7}$$

Where n is the error order of the formula used. For example the formula used for the center-difference order $O(h^2)$ in calculating $D(h)$ and $D(2h)$, then $n = 2$, so the extrapolation formula for Richardson is as in equation (10.6).

Also note that each expansion of Richardson extrapolation will increase the error order from $O(h_n)$ to $O(h_{n+2})$.

Algorithm

1. Define the data points (x), function $f(x)$ and a desired value, which want to calculate the derivative.

2. Determine the length of data n.

3. Determine the order of desired value in the data x.

4. Define h.

5. Calculate $D(h) = \frac{(f_1 - f_{-1})}{2h}$

6. Define $2h$

7. Calculate $D(2h) = \frac{(f_2 - f_{-2})}{2(2h)}$

8. Calculate the length of $D(2h)$

9. **for** k in range 0 to length $D(h)$-1

    Calculate $f_0' = D(h)_k + \frac{1}{2^{n-1}}[D(h)_k - D(2h)_{k+1}]$

10. **Repeat** step 4-9, depend on the order error $O(h_n)$

Task

Let assume $D(2h)$ and $D(4h)$ are the approximate derivation of $f\,'(x_0)$ with the interval 2h and 4h using the formula of the order of center-order $O(h^4)$. By using the Richardson extrapolation, calculate the better estimate of $f\,'(x_0)$:

$$f_0' = D(2h) + \frac{[D(2h\,) - D\,(4h)]}{15}$$

Determine the approximate derivation of $f\,'(1.2)$ if the function is $f(x) = e^x$ in the interval [0.8, 1.6] with h = 0.1.

References

1. Capra, Steven C and Canale, 1991, “Numerical Methods for Engineers with Personal Computer Applications”, MacGraw-Hill Book Company.

2. King M.R and Mody N.A, 2010, “Numerical and Statistical Methods for Bioengineering”, Cambridge University Press, New York.

Chapter 11. Richardson Extrapolation: Integration

Aims and Objectives

To determine a numerical integration using Richardson extrapolation and knowing its strengths and weaknesses compare to other numerical integration methods.

Preliminary Task

Calculate the result of $\int_0^1 \frac{1}{1+x} dx$ using the Richardson extrapolation and Romberg integration. (the total of segment is 8)

Literature Review

Richardson extrapolation can also be applied in numerical integration to obtain a more accurate solution. Recall the Trapezoidal rule:

$$\int_a^b f(x)dx = \frac{h}{2}\left(f_0 + 2\sum_{i=1}^{n} f_i + f_n\right) - \frac{(b-a)f''(t)}{12}h^2 \tag{11.1}$$

which can be written as:

$$\int_a^b f(x)dx = I(h) + Ch^2 \tag{11.2}$$

where $I(h)$ is the integration result using trapezoidal rule with $C = \frac{(b-a)f''(t)}{12}$.

Generally, integration method can be written as:

$$\int_a^b f(x)dx = I(h) + Ch^q \tag{11.3}$$

if it assumes that C is constant regardless of step size h, then q is determined from the integration method's error order. For the example:

Trapezoidal method has the error order $O(h^2)$, so the q is 2

Midpoint method has the error order $O(h^2)$, so the q is 2

Simpson 1/3 method has the error order $O(h^4)$, so the q is 4

RICHARDSON EXTRAPOLATION: Derivatives & Integration

Richardson's extrapolation technique is possible to combine the numerical results of integration method for two different step sizes, h_1 and h_2, to obtain a third numerical result, $I_3 = f(I_1, I_2)$. Extrapolation technique is much more accurate than the two approximations

I_1 and I_2. The higher-order integral approximation I_3 obtained by combining the two low-accuracy approximations of the trapezoidal rule so it has an error order $O(h_4)$. Thus, we can reduce the error efficiently in a single step using Richardson extrapolation.

Suppose the trapezoidal rule is used to solve an integral over the interval $[a, b]$. If J is the better approximate value of the integral, $I(h)$ is the approximation from n-segment of trapezoidal rule with the step size $h_1 = h$ and $h_2 = 2h$, and the error is $C(h)$, then the numerical integration will be:

$$J = I(h) + C(h^q) \tag{11.4}$$

$$J = I(2h) + C(2h^q) \tag{11.5}$$

Eliminate C from both equation

$$I(h) + C(h^q) = I(2h) + C(2h^q) \tag{11.6}$$

Thus, we obtain:

$$C = \frac{I(h)-I(2h)}{(2^q-1)h^q} \tag{11.7}$$

If we substitute (11.7) into (11.4), the Richardson extrapolation formula becomes:

$$J = I(h) + \frac{I(h)-I(2h)}{2^q-1} \tag{11.8}$$

When Richardson extrapolation technique is applied to numerical integration **repeatedly** to improve the accuracy of each successive layer of approximation, the scheme is called **Romberg integration.**

Romberg Integration is an extended of Richardson extrapolation combined with Trapezoidal method to obtain a better integration result. In every application of Richardson will increase the order of errors in the solution by two:

$$O(h^{2N}) \rightarrow O(h^{2N+2})$$

RICHARDSON EXTRAPOLATION: Derivatives & Integration

Recall the Richardson extrapolation (11.8), if I is the exact value of the integral then it can be written as:

$$I = A_k + Ch^2 + Dh^4 + Eh^6 + \cdots \tag{11.9}$$

A_k is the approximate value of the integral using Trapezoidal method with total of segment $n = 2^k$ and the error order $O(h^2)$.

$A_0, A_1, \ldots A_k$ from the Richardson extrapolation then will be used to determine the sequences of B_k, which is:

$$B_k = A_k + \frac{A_k - A_{k-1}}{2^2 - 1} \tag{11.10}$$

So, from now on the better I will be $I = B_k + D'h^4 + E''h^6 + \cdots$ the error order $O(h^4)$.

Hereafter, $B_0, B_1, \ldots B_k$ will be used for obtaining the sequences of $C_0, C_1, \ldots C_k$

$$C_k = B_k + \frac{B_k - B_{k-1}}{2^4 - 1} \tag{11.11}$$

This process will be continue until the n-segment ($n = 2^k$) is fulfilled, Thus the order of error $O(h)$ and the layer of Richardson extrapolation ($A, B, C, \ldots, k$+1)will increase depends on k.

Overall, the Romberg integration is illustrated as below:

$O(h^2)$	$O(h^4)$	$O(h^6)$	$O(h^8)$	$O(h^{10})$	$O(h^{12})$	$O(h^{14})$
A_0						
A_1	B_1					
A_2	B_2	C_2				
A_3	B_3	C_3	D_3			
A_4	B_4	C_4	D_4	E_4		
A_5	B_5	C_5	D_5	E_5	F_5	
A_6	B_6	C_6	D_6	E_6	F_6	G_6

The better integration result

RICHARDSON EXTRAPOLATION: Derivatives & Integration

Algorithm

Richardson Extrapolation

1. Define the data points (x), function $f(x)$ and number of segment

2. Define the step size (h_1) = h

3. Calculate the integral results ($I(h)$) using integration method (such as Trapezoidal method) for step size h

4. Define the different step size (h_2) = $2h$

5. Calculate the integral results ($I(2h)$) using integration method (such as Trapezoidal method) for step size $2h$

6. Calculate the result of integration using Richardson extrapolation $J = I(h) + \frac{I(h)-I(2h)}{2^2-1}$

Romberg Integration

1. Define the data points (x), function $f(x)$ and number of segment (n)

2. Determine the total of layer (k) , since n-segment (n = 2^k) then $k = {\ln(n)}/{\ln(2)}$

2. Define the step size (h_1) = h

3. Calculate the integral results ($I(h)$) using integration method (such as Trapezoidal method) for step size h

4. Define the different step size (h_2) = $2h$

5. Calculate the integral results ($I(2h)$) using integration method (such as Trapezoidal method) for step size $2h$

6. Calculate the result of integration using Richardson extrapolation $J = I(h) + \frac{I(h)-I(2h)}{2^2-1}$

7. **Repeat** step 3 to 6 to as many as the number of layer (k)

Task

Using the same problem as in Chapter 9, a cylindrical bag of radius 10 cm and height 50 cm contains saline solution (0.9% NaCl) that is provided to a patient as a peripheral intravenous (IV) drip. The method used to infuse saline into the patient's bloodstream in this example is gravity drip. The solution flows from the bag downwards under the action of gravity through a hole of radius 1mm at the bottom of the bag. If the only resistance offered by the flow system is the viscous resistance through the tubing, **determine** the time it will take for the bag to empty by 90%. The length of the tubing is 36″ (91.44 cm) and its ID (inner diameter) is 1 mm. The viscosity of the saline solution μ = 0.01 Poise, and the density ρ = 1 g/cm3. Let L be the length of the tubing, d the diameter of the tubing, and R the radius of the cylindrical bag. Then, L =91.44 cm; d =0.1 cm; R =10 cm. (g = 981 cm^2/s)

$$t = \frac{8R^2}{d^2}\int_5^{50} \frac{1}{\left(-a + \sqrt{a^2 + 8g(z+L)}\right)} dz$$

Use Romberg integration four level (layer k =3, n-segment = 2^3 = 8) and compare the error with the previous Trapezoidal method! (Hint: The exact value of integral = 0.574948166362027)

References

1. Capra, Steven C and Canale, 1991, "Numerical Methods for Engineers with Personal Computer Applications", MacGraw-Hill Book Company.

2. King M.R and Mody N.A, 2010, "Numerical and Statistical Methods for Bioengineering", Cambridge University Press, New York.

Chapter 12. ODE: Euler's method & Heun's Method

Aims and Objectives

To determine the solution of ordinary differential equation using Euler and Heun's Method.

Preliminary Task

Use Euler's method and Heun's method to integrate $y' = 4e^{0.8t} - 0.5y$ from $t = 0$ to 4 with a step size of one. The initial condition at $t = 0$ is $y = 2$. Note that the exact solution can be determined analytically as: $y(t) = \frac{4}{1.3}(e^{0.8t} - e^{-0.5t}) + 2e^{-0.5t}$

Literature Review

Euler's Method

A differential equation $\left({}^{dy}/_{dt}\right)$ is expressed in a function$f(t,y)$, where $y(t)$ is an original equation.

$$\frac{dy}{dt} = f(t,y), \quad a \le t \le b, \quad y(a) = \propto \tag{12.1}$$

It is known that the value of t is in range a and b and if the initial conditions is $t = a$ then y is α. However, we do not know the form of the original equation formulation at all y (t). So the challenge is how we can get a solution of differential equations for each value of y (t), which t is in interval $[a, b]$. The initial step of the numerical approach is to specify points within the step size in the interval $[a, b]$ as:

$$h = \frac{b-a}{N} \tag{12.2}$$

where N is a positive integer and $t_i = a + ih$, $i = 0, 1, 2, \ldots, N$.

Euler method is derived from Taylor. For example, the function $y(t)$ is a continuous function and has derivatives in the interval $[a, b]$. In the Taylor series, the function $y(t)$ is formulated as:

$$y(t_{i+1}) = y(t_i) + (t_{i+1} - t_i)y'^{(t_i)} + \frac{(t_{i+1}-t_i)^2}{2}y''(\xi_i) \tag{12.3}$$

By entering $h = (t_{i+1} - t_i)_r$, then

$$y(t_{i+1}) = y(t_i) + (t_{i+1} - t_i)y'(t_i) + \frac{(t_{i+1}-t_i)^2}{2}y''(\xi_i) \qquad (12.4)$$

Euler method is built based on equation (12.4), which ignored the second derivative. Besides, in general notation for $y(t_i)$ is replaced by w_i. So, the Euler method is formulated as:

$$w_{i+1} = w_i + h\,f(t_i, w_i), \qquad \text{with initial condition } w_0 = \alpha \qquad (12.5)$$

Where i = 0, 1, 2, …, N-1.

Heun's Method

Heun's method is used to improve the estimation of differential solution by determining then averaging two derivatives for the entire interval. Recall the Euler's method (12.1 to 12.3):

$$w_{i+1}^0 = w_i + h\,f(t_i, w_i)$$

In Heun's method the w_{i+1} is not the final answer, but *a predictor equation*. It provides an estimate that allows the calculation of a slope at the end of the interval:

$$w_{i+1}' = f(t_{i+1}, w_{i+1}^0)$$

Then, this predictor equation corrected by averaging two slopes, which is called a *corrector equation*:

$$w_{i+1} = w_i + h\,\frac{f(t_i, w_i) + f(t_{i+1}, w_{i+1}^0)}{2}$$

Algorithm

Euler's Method

1. Import module and define the function.

2. Determine the interval and the initial condition.

3. Define the step size h

4. Calculate the Euler's equation: $w_{i+1} = w_i + h\,f(t_i, w_i)$

5. Calculate the exact value

6. Determine the error value

Heun's Method

1. Import module and define the function.

2. Determine the interval and the initial condition.

3. Define the step size h

4. Calculate the predictor equation:

$$w_{i+1}^{0} = w_i + h\, f(t_i, w_i)$$

$$w_{i+1}' = f(t_{i+1}, w_{i+1}^{0})$$

5. Calculate the corrector equation: $w_{i+1} = w_i + h\, \frac{f(t_i, w_i) + f\left(t_{i+1}, w_{i+1}^{0}\right)}{2}$

6. Calculate the exact value

7. Determine the error value

Task

1. A non-charged capacitor is connected in series with a resistor and battery (Figure 12.1). It is known that Q = 12 Volts, C = 5.00 μF and R = 8.00 × 10^5 Ohm. When the switch is connected (t = 0), the charge does not yet exist (q = 0).

$$\frac{dq}{dt} = \frac{\epsilon}{R} - \frac{q}{RC}$$

The exact solution is:

$$q_{exact} = q(t) = C\epsilon\left(1 - e^{-t/RC}\right)$$

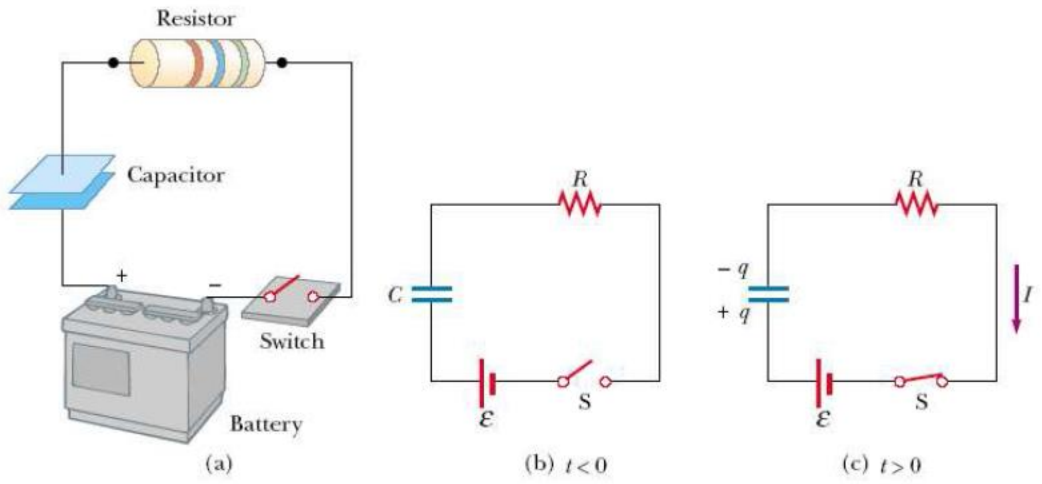


Figure 12.1 RC Circuit

If $t_0 = 0$ then $a = 0$ and at that time $q_0 = 0.0$. Step size $h = 0.1$ then $t_1 = 0.1$. Calculate the charge (q) at time $t = 2$ using the Euler's and Heun's methods and plot the curve of charging the q in t intervals 0 to 2. Compare the results!

2. HIV–1 virus particles attack lymphocytes and hijack their genetic machinery to manufacture many virus particles within the cell. Once the viral particles have been prepared and packaged, the viral genome programs the infected host to undergo lysis. The newly assembled virions are released into the blood stream and are capable of attacking new cells and spreading the infection. The dynamics of viral infection and replication in plasma can be modeled by a set of differential equations (Perelson et al.,1996). If T is the concentration of target cells, T* is the concentration of infected cells, and V1 is the concentration of infectious viral RNA in plasma, and VX is the concentration of noninfectious viral particles in plasma, we can write:

$$\frac{dT^*}{dt} = kV_1T - \delta T^*$$

$$\frac{dV_1}{dt} = -cV_1$$

$$\frac{dV_x}{dt} = N\delta T^* - cV_x$$

The experimental study yielded the following estimates of the reaction rate parameters (Perelson et al., 1996):

$\delta = 0.5/day$

$c = 03.0/day$

The initial concentrations of T, T^*, V_1, and V_X are as follows:

$V_1(t = 0) = 100/\mu l$;

$V_X\ (t = 0) = 0/\mu l$;

$T\ (t = 0)$ = 250 non-infected cells/µl (Haase et al., 1996);

$T^*(t = 0)$ = 10 infected cells/µl (Haase et al., 1996).

Based on a quasi-steady state analysis ($dT^*/dt = 0$, $dV/dt = 0$) before time $t = 0$, we calculate the following:

$k = 2x10^{-4}$ µl/day/virions, and

N = 60 virions produced per cell.

Calculate the T^*, V_1, V_X at time $t = 5$ days using the Euler's methods and plot the curve of T^*, V_1, V_X in t intervals 0 to 5.

References

1. Capra, Steven C and Canale, 1991, "Numerical Methods for Engineers with Personal Computer Applications", MacGraw-Hill Book Company.

2. King M.R and Mody N.A, 2010, "Numerical and Statistical Methods for Bioengineering", Cambridge University Press, New York.

Chapter 13. ODE: Runge-Kutta's Method

Aims and Objectives

To determine the solution of ordinary differential equation using Runge-Kutta's method and knowing its strengths and weaknesses compare to the previous methods.

Preliminary Task

Solve the following equation by using the thord order Runge-Kutta method.

$$\frac{dy}{dx} = -2x^3 + 12x^2 - 20x + 8{,}5.$$

from $x = 0$ to $x = 4$ by using an interval of $\Delta x = 0{,}5.$ The initial condition at $x = 0$ is $y = 1$.

Literature Review

Euler method ends up with a non-thorough result. Because of that, we need to consider higher order of Taylor series or by using small interval or Δx. Both of methods is not beneficial. The calculation of higher order needs higher derivative of a function while the use of small interval needs longer time of computation.

The Runge-Kutta method gives more precise result and does not need the derivative of the function. The common form of Runge-Kutta method is shown in equation (13.1).

$$y_{i+1} = y_i + \Phi(x_i, y_i, \Delta x)\, \Delta x \tag{13.1}$$

Where $\Phi(x_i, y_i, \Delta x)$ is a function of addition that is an average slope in the interval. The function of addition could be written as a common form shown in equation (2)

$$\Phi = a_1 k_1 + a_2 k_2 + \ldots + a_n k_n \tag{13.2}$$

where a is a constant and k is defined in equation (13. 3) to (13.6)

$$k_1 = f(x_i, y_i) \tag{13.3}$$

$$k_2 = f(x_i + p_1 \Delta x,\ y_i + q_{11} k_1 \Delta x) \tag{13.4}$$

$$k_3 = f(x_i + p_2 \Delta x,\ y_i + q_{21} k_1 \Delta x + q_{22} k_2 \Delta x) \tag{13.5}$$

$$\vdots$$

$$k_n = f(x_i + p_{n-1}\Delta x,\ y_i + q_{n-1,1}k_1\Delta x + q_{n-1,2}k_2\Delta x + \ldots + q_{n-1,n-1}k_{n-1}\Delta x) \qquad (13.6)$$

The equation shows that the *k* value has a sequential relationship. The value of k_1 appears in the equation to calculate k_2 that also appears in the equation to calculate k_3 and so on. This sequantial relationship makes the RungeKutta method becomes efficient in the calculation process.

There are several type of Runge-Kutta method that depends on the value of n that is used in the equation.

For *n* = 1, so-called first order Runge-Kutta, Equation (13.1) becomes

$$\Phi = a_1 k_1 = a_1\ f(x_i, y_i) \qquad (13.7)$$

For $a_1 = 1$, Equation (13.1) becomes

$$y_{i+1} = y_i + f(x_i, y_i)\,\Delta x \qquad (13.8)$$

That is similar with Euler method.

In the Runge-Kutta method, after the value of *n* is defined, the value of *a*, *p* and *q* are calculated by using Equation (1) by using the member of Taylor series.

Second Order Runge-Kutta Method

The second order Runge-Kutta method has a form shown in Equation (13.9).

$$y_{i+1} = y_i + (a_1 k_1 + a_2 k_2)\Delta x \qquad (13.9)$$

where:

$$k_1 = f(x_i, y_i) \qquad (13.10)$$

$$k_2 = f(x_i + p_1\Delta x,\ y_i + q_{11}k_1\Delta x) \qquad (13.11)$$

The value of a_1, a_2, p_1 and q_{11} were evaluated by equaling Equation (13.10) with the second order Taylor series that has form as shwon in Equation (13.12).

$$y_{i+1} = y_i + f(x_i, y_i)\frac{\Delta x}{1} + f'(x_i, y_i)\frac{\Delta x}{2} \qquad (13.12)$$

where $f'(x_i, y_i)$ could be defined from the chain rule as shown in Equation (13.13).

$$f'(x_i, y_i) = \frac{\partial f}{\partial x} + \frac{\partial f}{\partial y}\frac{dy}{dx} \tag{13.13}$$

The substitution of Equation (13.13) to Equation (13.12) produces Equation (13.14).

$$y_{i+1} = y_i + f(x_i, y_i)\frac{\Delta x}{1} + \left(\frac{\partial f}{\partial x} + \frac{\partial f}{\partial y}\frac{dy}{dx}\right)\frac{\Delta x}{2} \tag{13.14}$$

The value of a_1, a_2, p_1 and q_{11} was searched for so that the Equation (13.9) is equaivalent to Equation (13.14). Thus, the Taylor series is used to explpore Equation (13.11). The Taylor series as a function of two variables has a form of equatino shown below.

$$g(x+r, y+s) = g(x, y) + r\frac{\partial g}{\partial x} + s\frac{\partial g}{\partial y} + \ldots \tag{13.15}$$

Equation (13.11) could be re-written as follows.

$$f(x_i + p_1\Delta x,\ y_i + q_{11}k_1\Delta x) = f(x_i, y_i) + p_1\Delta x\frac{\partial f}{\partial x} + q_{11}k_1\Delta x\frac{\partial f}{\partial y} + 0(\Delta x^2) \tag{13.15}$$

The previous form and Equation (13.10) is substituted to Equation (13.9) and resulted in Equation below.

$$\begin{aligned} y_{i+1} = y_i &+ a_1\Delta x\, f(x_i, y_i) + a_2\Delta x\, f(x_i, y_i) + a_2 p_1\Delta x^2\frac{\partial f}{\partial x} \\ &+ a_2 q_{11}\Delta x^2\, f(x_i, y_i)\frac{\partial f}{\partial x} + 0(\Delta x^3) \end{aligned} \tag{13.16}$$

or

$$\begin{aligned} y_{1+1} = y_i &+ \left[a_1 f(x_i, y_i) + a_2 f(x_i, y_i)\right]\Delta x \\ &+ \left[a_2 p_1\frac{\partial f}{\partial x} + a_2 q_{11} f(x_i, y_i)\frac{\partial f}{\partial x}\right]\Delta x^2 + 0(\Delta x^3) \end{aligned} \tag{13.17}$$

By comparing Equation (13.14) and Equation (13.15), it could concluded that both equations would be equavalent if:

$$a_1 + a_2 = 1. \tag{13.18}$$

$$a_2 p_1 = \frac{1}{2}. \tag{13.19}$$

$$a_2 q_{11} = \frac{1}{2}. \tag{13.20}$$

The equation system above that has 3 equations with four unknown variables is unsolvable. Thus, one of the unknown variables should be defined and the other three variavbles could be calculated. If a_2 is stated, then Equation (9a), (9b), and (9c) could solved and produces:

$$a_1 = 1 - a_2 \tag{13.21}$$

$$p_1 = q_{11} = \frac{1}{2a_2} \tag{13.22}$$

Because the value of a_2 is chosen randomly, the second order Runge-Kutta method would be numerous. One of them is Heun method.

Fourth Order Runge-Kutta Method

The fourth order Runge-Kutta method is derived by using the same method as the second order Runge-Kutta method with n = 3. The result is six equations with eight unknown variables. Thus, two variables need to be defined first to obtain the other six unknown variables. The common result is shown in Equation (13.23).

$$y_{i+1} = y_i + \tfrac{1}{6}\,(k_1 + 2k_2 + 2k_3 + k_4) \tag{13.23}$$

Where

$$k_1 = \Delta x\,.\,f(x_i, y_i) \tag{13.24}$$

$$k_2 = \Delta x\,.\,f(x_i + \tfrac{1}{2}\Delta x, y_i + \tfrac{1}{2}k_1\Delta x) \tag{13.25}$$

$$k_3 = \Delta x\,.\,f(x_i + \tfrac{1}{2}\Delta x, y_i + \tfrac{1}{2}k_2\Delta x) \tag{13.26}$$

$$k_4 = \Delta x\,.\,f(x_i + \Delta x, y_i + k_3\Delta x) \qquad (13.26)$$

Algorithm

1. Import module and define the function.

2. Determine the interval and the initial condition.

3. Define the step size h

4. Calculate the predictor equation:

$$k_1 = h * f(x_i, y_i)$$

$$k_2 = h * f(x_i + \tfrac{1}{2}\Delta x, y_i + \tfrac{1}{2}k_1\Delta x)$$

$$k_3 = h * f(x_i + \tfrac{1}{2}\Delta x, y_i + \tfrac{1}{2}k_2\Delta x)$$

$$k_4 = h * f(x_i + \Delta x, y_i + k_3\Delta x)$$

5. Calculate the corrector equation: $y_{i+1} = y_i + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$

Task

1. By using the same problem as shown in Chapter 12, please calculate the load (q) at t=2 by using the fourth order Runge-Kutta method and plot the curve of loading process (charging) towards t between t = 0s and t = 2s. Compare the result!

2. A spherical cell or particle of radius a and uniform density ρ_s, is falling under gravity in a liquid of density ρ_f and viscosity μ. The motion of the sphere, when wall effects are minimal, can be described by the equation:

$$\frac{4}{3}\pi a^3 \rho_s \frac{dv}{dt} + 6\mu\pi a v - \frac{4}{3}\pi a^3(\rho_s - \rho_f)g = 0$$

With initial boundary conditions

$$z = 0; \quad v = \frac{dz}{dt} = 0,$$

Where z is the displacement distance of the sphere in the downward direction. Downward motion v is along the positive z direction. On the left-hand side of the equation, the first term describes the acceleration, the second describes viscous drag, which is proportional to the sphere velocity, and the third is the force due to gravity

acting downwards from which the buoyancy force has been subtracted. Two initial conditions are included to specify the problem completely. This initial value problem can be converted from a single second-order ODE into a set of two coupled first-order ODEs. Set $y_1 = z$ and $y_2 = dz/dt$. Then:

$$\frac{dy_1}{dt} = y_2, \qquad y_1 = 0$$

$$\frac{dy_2}{dt} = \frac{(\rho_s - \rho_f)}{\rho_s} g - \frac{9\mu}{2a^2\rho_s} y2, \qquad y_2 = 0$$

The constants are set to:

a = 10^{-4} cm; ρ_s = 1.1 g/cm^3; ρ_f = 1 g/cm^3; g = 981 cm/s^2; μ = 3.5 x 10^{-2} g/cm.s

Plotting the numerically calculated displacement z and the velocity dz/dt with time interval 0 to 0.001 s and find the minimum step size to obtain the steady state velocity (which it does not increase or decrease).

References

1. Capra, Steven C and Canale, 1991, "Numerical Methods for Engineers with Personal Computer Applications", MacGraw-Hill Book Company.

2. King M.R and Mody N.A, 2010, "Numerical and Statistical Methods for Bioengineering", Cambridge University Press, New York.

Chapter 14. Partial Differential Equation: Forward

Aims and Objectives

To determine the solution of partial differential equation (PDE) by reduce it to a system of ODE's using finite difference method.

Preliminary Task

It is known one-dimensional heat distribution (1D) as a function of time (t) on a metal meets the following equation:

$$\frac{\partial u}{\partial t}(x,t) - \frac{\partial^2 u}{\partial x^2}(x,t) = 0, \qquad 0 < x < 1 \qquad 0 \leq t$$

With boundary condition:

$$u(0,t) = u(1,t) = 0, \quad 0 < t,$$

and initial condition:

$$u(x,0) = sin(\pi x), \quad 0 \leq x \leq 1,$$

Solve that PDP parabolic equation by using finite difference method if the vertical axis shows changes over time with intervals k = 0,0005. Because α = 1, h = 0, 1 and k = 0,0005.

Literature Review

Previously, we have learn to solve the ordinary differential equation (ODE) and now we will determine the solution of partial differential equation (PDE) by reduce it to ODE's system so we could obtain the result easily. One important technique for achieving this, is based on finite difference discretization of spatial derivatives.

We shall focus on one of the most widely encountered partial differential equations: the diffusion equation, which in one dimension looks like:

$$\frac{\partial u}{\partial t} = \beta \frac{\partial^2 u}{\partial x^2} + g \tag{14.1}$$

Introduce a spatial mesh in Ω with *mesh points.*

$$x_0 = 0 < x_1 < x_2 < \cdots < x_N = L$$

The space between two mesh points x_i and x_{i+1}, i.e. the interval $[x_i, x_{i+1}]$, is call a *cell*. It could be assume that each cell has the same length ($\Delta x = x_{i+1} - x_i, i = 0, \dots N-1$).

The partial differential equation is valid at all spatial points $x \in \Omega$, but we may relax this condition and demand that it is fulfilled at the internal mesh points only, $x_1, \dots, x_{N-1}$:

$$\frac{\partial u(x_i,t)}{\partial t} = \beta \frac{\partial^2 u(x_i,t)}{\partial x^2} + g(x_i,t) \quad , \; i = 1, \dots, N-1 \qquad (14.2)$$

Now, at any point x_i we can approximate the second-order derivative by a finite difference:

$$\frac{\partial^2 u(x_i,t)}{\partial x^2} \approx \frac{u(x_{i+1},t) - 2u(x_i,t) + u(x_{i-1},t)}{\Delta x^2} \qquad (14.3)$$

It is common to introduce a short notation $u_i(t)$ for $u(x_i,t)$, i.e., u approximated at some mesh point x_i in space. By inserting (14.3) to (14.2), an approximation to the partial differential equation at mesh point (x_i, t) can be written as:

$$\frac{\partial u_i(t)}{\partial t} = \beta \frac{u_{i+1}(t) - 2u_i(t) + u_{i-1}(t)}{\Delta x^2} + g_i(t), \; i = 1, \dots, N-1 \qquad (14.4)$$

Note that we have adopted the notation $g_i(t)$ for $g(x_i,t)$.

Equation (14.4) is an ODE's system of PDE in equation (14.2) with aid of the finite difference approximation.

The initial condition $u(x,0) = I(x)$ translates to an initial condition for every unknown function $u_i(t)$: $u_i(0) = I(x_i), i = 0, \dots, N$. At the boundary $x = 0$ we need an ODE in our ODE system, which must come from the boundary condition at this point. The boundary condition reads $u(x,0) = s(t)$. We can derive an ODE from this equation by differentiating both sides: $u_0'(t) = s'(t)$. The ODE system above cannot be used for u_0' since that equation involves some quantity u_{-1}' outside the domain. Instead, we use the equation $u_0'(t) = s'(t)$ derived from the boundary condition. For this particular equation we also need to make sure the initial condition is $u_0(t) = s(0)$. The reason for including the boundary values in the ODE system is that the solution of the system is then the complete solution at all mesh points, which is convenient, since special treatment of the boundary values is then avoided.

The condition $\frac{\partial u}{\partial x} = 0$ at $x = L$ is a bit more complicated, but we can approximate the spatial derivative by a centered finite difference:

$$\left.\frac{\partial u}{\partial x}\right|_{i=N} \approx \frac{u_{N+1} - u_{N-1}}{2\Delta x} = 0 \tag{14.5}$$

This approximation involves a fictitious point x_{N+1} outside the domain. A common trick is to use (14.4) for $i = N$ and eliminate u_{N+1} by use of the discrete boundary condition ($u_{N+1} = u_{N-1}$):

$$\frac{\partial u_N(t)}{\partial t} = \beta \frac{2u_{N-1}(t) - 2u_N(t)}{\Delta x^2} + g_N(t) \tag{14.6}$$

That is, we have a special version of (14.4) at the boundary $i = N$.

A summarize in approximating the partial differential equation problem (14.2) to (14.6) by an ODE's system is:

$$\frac{du_0}{dt} = s'(t) \tag{14.7}$$

$$\frac{du_i}{dt} = \frac{\beta}{\Delta x^2}\left(u_{i+1}(t) - 2u_i(t) + u_{i-1}(t)\right) + g_i(t) \quad , \; i = 1, \dots, N-1 \tag{14.8}$$

$$\frac{du_N}{dt} = \frac{2\beta}{\Delta x^2}\left(u_{N+1}(t) - u_N(t)\right) + g_i(t) \tag{14.9}$$

The initial conditions are

$$u_0(0) = s(0) \tag{14.10}$$

$$u_i(0) = I(x_i) \; , \; i = 1, \dots, N-1 \tag{14.11}$$

Algorithm

1. Discretize solution $u(x, t)$ into discrete values

2. Define the step size h

3. Define the initial and boundary conditions.

4. Compute $g(x_i, t_i)$ using finite differences than compute $u(x_i, t_i)$ based on $g(x_i, t_i)$

5. Plot the solution from i to $i + 1$.

PARTIAL DIFFERENTIAL EQUATION (PDE)

Task

The diffusion equation is written in one spatial dimension as:

$$\frac{\partial u}{\partial t} = D\frac{\partial^2 u}{\partial x^2}$$

Let us assume an initial concentration:

$$u(x, t_0) = \frac{1}{\sigma\sqrt{2\pi}}\exp\left(-\frac{(x - x_{mean})^2}{\sigma^2}\right)$$

Where x_{mean} = 0 and width σ = 0.5. By using finite difference method, plot the solution of the diffusion equation in interval -5 to 5 with step size h = 0.1.

References

1. King M.R and Mody N.A, 2010, “Numerical and Statistical Methods for Bioengineering”, Cambridge University Press, New York.

2. Langtangen Hans Petter and Linge Svein, 2017, “Finite Difference Computing with PDEs - A Modern Software Approach”. Springer International Publishing.