Semester-V

16CHU502B APPLICATIONS OF COMPUTERS IN CHEMISTRY

4H 4C

Instruction Hours/week: L:04 T:0 P:0 Marks: Internal: 40 External: 60 Total:100

Scope

The course deals with the basics of computers in chemistry, numerical methods, Differential and integral calculus, simuntaneous equations and molecular modelling.

Objectives

This course enables the student to

1. Understand the basics of computers in chemistry
2. Understand the numerical methods to find the roots of equation
3. Understand the differential and integral calculus
4. Understand to handle experimental data using simuntaneous equations
5. Understand the molecular modelling.

Methodology

Blackboard teaching, Powerpoint presentation and group discussion.

UNIT I

Basics:

Constants, variables, bits, bytes, binary and ASCII formats, arithmetic expressions, hierarchy of operations, inbuilt functions. Elements of the BASIC language. BASIC keywords and commands. Logical and relative operators. Strings and graphics. Compiled versus interpreted languages. Debugging. Simple programs using these concepts. Matrix addition and multiplication. Statistical analysis.

UNIT II

Numerical methods:

Roots of equations: Numerical methods for roots of equations: Quadratic formula, iterative method, Newton-Raphson method, Binary bisection and Regula-Falsi.

UNIT III

Differential calculus: Numerical differentiation.

Integral calculus: Numerical integration (Trapezoidal and Simpson's rule), probability distributions and mean values.

UNIT IV

Simultaneous equations: Matrix manipulation: addition, multiplication. Gauss-Siedal method.

Interpolation, extrapolation and curve fitting: Handling of experimental data.

UNIT V

Conceptual background of molecular modelling: Potential energy surfaces. Elementary ideas of molecular mechanics and practical MO methods.

Suggested Readings
Text Books:

1. Harris, D. C. (2007). *Quantitative Chemical Analysis*. 6th Ed. (Chapters 3-5). Freeman.
2. Levie, R. De. (2001). *How to use Excel in analytical chemistry and in general scientific data analysis*, Cambridge Univ. Press 487 pages.

Reference Books

1. Noggle, J. H. (1985). *Physical chemistry on a Microcomputer.* Little Brown & Co.
2. Venit, S.M. (1996). *Programming in BASIC: Problem solving with structure and style.* Delhi: Jaico Publishing House.

KARPAGAM ACADEMY OF HIGHER EDUCATION

(Deemed to be University)
(Established Under Section 3 of UGC Act, 1956)
Coimbatore – 641 021

Lecture Plan
Department of Chemistry

Name of the Staff : Dr. S. Ravi and Dr. K. Sundaram
Title of the Paper : Applications of Computers in Chemistry
Subject code : 16CHU502B Class: III B.Sc. Chemistry
Year and Semester : 2018–2019 & V-Semester Total hours : 48 Hours

Unit-I Total hrs: 12

S. No	Lecture hours	Titles	Reference
1	1	Constants, variables, bits	T1: 11,48-49 234-235
2	1	Bytes, binary and ASCII formats	T1:11
3	1	Arithmetic expressions, hierarchy of operations	T1: 46 T2:140
4	1	Elements of the BASIC language	T2:220
5	1	BASIC keywords and commands	T1:234 T2:113
6	1	Logical and relative operators	T1:235-237
7	1	Strings and graphics	T2:6
8	1	Compiled versus interpreted languages	T1:31
9	1	Debugging	T1: 33, 247
10	1	Simple programs using these concepts	T1: 33
11	1	Matrix addition and multiplication	T2: 190
12	1	Recapitulation and discussion of important questions	

Unit-2 Total hrs: 7

S. No	Lecture hours	Titles	Reference
1	1	Roots of equations	T3:123
2	1	Numerical methods for roots of equations	T3:121
3	1	Quadratic formula	T2:245-246
4	1	Iterative method	T2:268
5	1	Newton-Raphson method	T2:268
6	1	Binary bisection and Regula-Falsi	T1:11, 171
7	1	Recapitulation and discussion of important questions	

Unit-3 Total hrs: 10

S. No	Lecture hours	Titles	Reference
1	1	Differential calculus	T1:181
2	2	Numerical differentiation.	T1:181-183
3	1	Integral calculus	T4:1-4
4	1	Numerical integration	T2:328
5	2	Trapezoidal and Simpson's rule	T1:177-178
6	1	Probability distributions	
7	1	Mean values	
8	1	Recapitulation and discussion of important questions	

Text books:

Unit-4 Total hrs: 10

S. No	Lecture hours	Titles	Reference
1	1	Simultaneous equations	
2	1	Matrix manipulation	T2:190
3	1	Addition	T1:102
4	1	Multiplication	T2:190
5	1	Gauss-Siedal method	T2:359
6	1	Interpolation	T1:191
7	1	Extrapolation	T1:191
8	1	Curve fitting	T1:183
9	1	Handling of experimental data	
10	1	Recapitulation and discussion of important questions	

Unit-5 Total hrs: 9

S. No	Lecture hours	Titles	Reference
1	1	Conceptual background of molecular modelling	W2
2	2	Potential energy surfaces	W1
3	1	Elementary ideas of molecular mechanics	W2
4	2	Practical MO methods	W3
5	1	Recapitulation and discussion of important questions	
6	1	Discussion of previous ESE question papers	
7	1	Discussion of previous ESE question papers	

Text books:

T1. Ramesh Kumari, 2014, Computers and their applications to chemistry, Narosa publishing house, NewDelhi.

T2: P. Riyazuddin, 2012, Computers in chemistry, I.K. International Publishing House Pvt. Ltd., NewDelhi.

T3: E. Balagurusamy, 2012, Computer Oriented Statistical and Numerical methods, Macmillan India Limited, New Delhi.

T4: D.C. Agarwal, 2011, Advanced Integral Calculus, Krishna Prakasan Media (P) Ltd, Meerut, India.

Website:

W1. http://pollux.chem.umn.edu/8021/Lectures/Preamble_1.pdf

W2: http://vergil.chemistry.gatech.edu/courses/chem6485/pdf/molmech-lecture.pdf

W3: https://www.ch.ic.ac.uk/local/organic/mod/Chem99.pdf

UNIT I

Syllabus

Basics:

Constants, variables, bits, bytes, binary and ASCII formats, arithmetic expressions, hierarchyof operations, inbuilt functions.Elements of the BASIC language.BASIC keywords andcommands.Logical and relative operators.Strings and graphics.Compiled versus interpretedlanguages.Debugging.Simple programs using these concepts.Matrix addition andmultiplication.Statistical analysis.

Variables

A C variable can be declared as follows

int x = 10;

This defines a variable x of type int (32-bits) and assigns 10 as the initial value. It is possible to find both the value and a pointer to (address of) the variable x. The value and the address of the variable can be found using:

printf("The value is %d and the address is %x \n", x, &x);

The format statements %d (decimal) and %x(hexadecimal) are used to format the output. A variable declared as int x is called an automatic variables. Automatic variables are not initialized, and given a place in the runtime stack, and it is the programmer's responsibility to initialize the variables. When a variable goes out of scope, the memory is given back. In addition to automatic storage class, C variables can be declared to be static, extern or register variables.

Storage Class Specifiers

Auto, Static, Extern and Register Variables

Any variable defined inside a function or file are considered to be an auto variable unless specified otherwise. That is, the scope of the variable is within the function or file it is declared.

For example

#includeint n;

file scope below this line

```
#include <stdio.h>
int n;      ← file scope below this line

int main() {
  int n;    ←main function scope only

   int foo( ) {
     int n;   ←foo function scope only
   }
```

Let us look at the scope of the variables declared in this file. The first **int n** that is declared just below the #include statement can be seen by any function below it. In otherwords, the location of the variable determines where the variable is meaningful. Anyfunction declared above first **int n** will not be able to see the variable n. The variables ndefined inside functions main and foo are only meaningful within the functions.

STATIC VARIABLES

Static variables can be declared externally, that is, outside of any function, or internally, that is inside a function. An external static variable declared outside of functions isvisible to users of the file, but not to the functions in other files. An internal staticvariable declared inside a function retains its value during all function call to thefunction. For example, consider the file program. That contains the following declarations.

```
#include <stdio.h>
static int n;     ← external static variable – not visible to any other file

int main() {
  foo();
  foo();
}

int foo( ) {
  static int n;   ←internal static variable – retains its value during multiple calls to foo
}
```

Functions can also be declared static, making them visible only to the source file whereit is declared. Normally functions are visible to any part of the program. But by makingthem static, we can limit the scope of the function only to the source file where it isdeclared.

EXTERN VARIABLES

Extern variables can be used to share value of a variable among many functions. Typically, all variables declared inside a function are local variables and they are createdwhen function is invoked and released when function is exited. However, if we are toshare the same variable among many functions, then we can define the variable to be"extern" as follows.

```
int n;  ← allocates storage for the external variable

int foo ( ) {
   extern int n;   ← refers to the n defined above
}

int foo2( ) {
  extern int n;  ← refers to the same n as above
}
```

It should be noted that the use of "extern" inside the function can be avoided if all functions using n are defined below the external definition of n. The use of extern is more evident when "extern" variables can be shared across multiplefiles. For example, if program.c defines an external variable n, then program2.c andprogram3.c can refer to the value of the external variable by using the qualifier "extern".Extern variables therefore are useful for sharing variables among several files.

REGISTER VARIABLES

Some variables that may be accessed by the program frequently can be specified to be a"register" variable.

registerint x;

This will request the compiler to consider allocating a register location, a fast access memory location, for the variable x. However, compiler may completely ignore this request. The use of a register variable depends on hardware restrictions such as numberof registers available etc. If a register cannot be allocated for a particular variable, thenthe directive is ignored. However, the address of a register variable cannot be accessedregardless of whether it is placed in a register or not.

Bits, Bytes and Data Types

A bit is the smallest unit of storage represented by 0 or 1. A byte is typically 8 bits. C character data type requires one byte of storage. A file is a sequence of bytes. A size of the file is the number of bytes within the file. Although all files are a sequence of bytes,m files can be regarded as text files or binary files. Text files are human readable (it consists of a sequence of ASCII characters) and binary files may not be human readable (eg: image file such as bitmap file).

If you have a UNIX shell, you can type

>ls -l filename // to find out the size of the file(and many other things).

Eg: -rw-r--r-- 1 guna staff 11977 Feb 24 2004 joel.txt

Standard units of memory

1000 bytes = 1 Kilobytes(KB)

1000 KB = 1 megabyte (MB)

1000MB = 1 Gigabyte(GB)

1000 GB = 1 Terabyte(TB)

1000 TB = 1 Petabyte(PB)

Each data byte can be represented using an ASCII (or extended ASCII) value. An ASCII table is given below. Standard ASCII table assigns each character to a numerical value. For example ‘A’ = 65 and ‘a’ = 97. Printable standard ASCII values are between 32 and 126. The 8th bit in the byte may be used for parity checking in communication or other device specific functions.

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Each ASCII value can be represented using 7 bits. 7 bits can represent numbers from0 = 0000 0000 to 127 = 0111 1111 (total of 128 numbers or 2^7)

The Basic programming language is a simple and easy to understand programming language. To use it correctly, it is sufficient to know just a few basic elements that every program consists of. These are:

Identifiers

Comments

Operators

Expressions

Instructions

Constants

Variables

Symbols

Directives

Labels

Procedures and functions

Modules

Here is an example of how you should not write a program. No comments are included, labels' names are meaningless, code sections are not grouped… This program is going to work properly, but its purpose and way of execution will be only known to the programmer who has written it (at least for a day or two).

Identifiers are arbitrary names assigned to the basic language objects such as constants, variables, functions, procedures etc. Somebody just came to an idea to use the word identifier instead of name. As simple as that. Here are a few rules to be observed when using identifiers:

Identifiers may contain all the letters of alphabet (both upper and lower case), digits (0-9) and the underscore character (_).

The first character of an identifier must not be a digit.

No identifier may contain special characters such as ! [{ # $ % & etc.

Basic is not case-sensitive, which means that FIRST, first and First will be considered identical.

COMMENTS

Comments are parts of the program used to provide more information about the program and make it clear to the user. In Basic, any text following a single quotation mark (') is considered a comment. Comments are not compiled into executable code. The compiler is capable of recognizing special characters used to mark where comments start and completely ignores the following text during compilation. Even though comments cannot affect the program execution, they are as important as any other part of the program because almost every program needs to be improved, modified, upgraded or simplified at some point. Without comments, it is almost impossible to understand even the simplest programs.

LABELS

Labels provide the easiest way of controlling the program flow. They are used to mark particular lines in the program where jump instruction and appropriate subroutine are to be executed. All labels must be terminated by ':' so that the compiler can easily recognize them.

CONSTANTS

A constant is a number or a character the value of which cannot be changed during the program execution. Unlike variables, constants are stored in ROM memory of the microcontroller in order to save as much memory space of RAM as possible. The compiler

recognizes constants by their names and prefix *const*. Every constant is declared under unique name which must be a valid identifier. Constants are available in decimal, hexadecimal and binary formats. The compiler distinguishes between them according to their prefixes. If a constant has no prefix, it is considered as default.

Constants are declared in the declaration part of the program or routine. The syntax of constants is:

Constants are declared in the declaration part of the program or routine. The syntax of constants is:

```
constconstant_name [as type] = value
```

Constant names are usually written in capitals. The type of a constant is automatically recognized by its size. In the following example, the constant *MINIMUM* is considered a signed integer and will be stored within two bytes of *Flash* memory (16 bits):

```
const MINIMUM = -1000 ' Declare constant MINIMUM
```

Type of constant is optionally specified. In the absence of type, the compiler assumes the 'smallest' type that can accommodate the constant value.

```
const MAX as longint = 10000
const MIN = 1000 ' Compiler will assume word type
const SWITCH = "n" ' Compiler will assume char type
```

In the following example, a constant named *T_MAX* is declared so as to have a fractional value 32.60. Now, the program can compare the measured temperature to that constant with a meaningful name instead to number 32.60.

```
const T_MAX = 32.60 ' Declare temperature T_MAX
const T_MAX = 3.260E1 ' Another way of declaring constant T_MAX
```

A string constant consists of a sequence of characters. They are enclosed within double quotation marks. A blank space may also be included in the string constant as a character. String constants are used to represent non-numeric quantities such as names, addresses, messages etc.

```
const Message_1 = "Press the START button" ' Message 1 for LCD
```

```
const Message_2 = "Press the RIGHT button" ' Message 2 for LCD
const Message_3 = "Press the LEFT button" ' Message 3 for LCD
```

In this example, sending the *Message_1* constant to an LCD will cause the message *'press the START button'* to be displayed.

VARIABLES

A variable is a named object able to contain a data which can be modified during program execution. Every variable is declared under a unique name which must be a valid identifier. For example, to add two numbers (*number1* + *number2*) in the program, it is necessary to have a variable to represent what we in everyday life call the sum. In this case *number1*, *number* and *sum* are variables. The syntax of one single variable declaration is as follows:

```
dimvariable_name as type
```

Variables in Basic are typed, which means that it is necessary to specify the type of data a variable is to receive. Variables are stored in RAM and the memory space occupied (in bytes) depends on their type. In addition to single declarations, variables of the same type can be declared as a list. Here, *identifier_list* is a comma-delimited list of valid identifiers, whereas *type* can be any data type.

```
dim i, j, k as byte 'Define variables i, j, k
dim counter, temp as word 'Define variables counter and temp
```

SYMBOLS

Symbols in Basic allow you to create simple macros without parameters. It means that any code line may be replaced with one single identifier. Symbols, when used properly, can increase code legibility and reusability. Symbols are declared at the beginning of the module, right after the module name and optional *include* directive. The scope of a symbol is always limited to the module in which it has been declared.

```
symbolsymbol_name = code
```

Here, *symbol_name* must be a valid identifier to be used throughout the *code*. The code specifier can be any code line (literals, assignments, function calls, etc).

```
symbol MAXALLOWED = 216 ' Symbol MAXALLOWED for numeric value
```

```
symbol OUT = PORTA ' Symbol OUT for SFR
symbol MYDELAY = Delay_ms(762) ' Symbol MYDELAY for procedure call
dimcnt as byte ' Variable cnt
main:
ifcnt> MAXALLOWED then ' Program checks whether cnt> 216
cnt = 0 ' If yes,
 OUT.1 = 0 ' the following three commands
 MYDELAY ' are to be executed
end if
... ' If not, program continues here
```

No RAM memory space is used for storing symbols being used in the program. The compiler will simply replace all symbols with appropriate code lines assigned to them when declared.

CLS

An abbreviation that stands for the words **CL**ear **S**creen. In the above program, when you used CLS on line 60, all of the words that were printed to the screen were wiped away.

PRINT

Writes to the screen. There are commands for printing to other things, like a printer, but that's to be discussed later. Each new PRINT command will start printing on a new line. To insert a blank line, don't specify a string to print. The syntax for "PRINT" is: PRINT "[whatever you want to be printed here]"

END

It stops the program at that line; that is, anything that's added after that won't show. That's why the PRINT command on line 90 didn't print anything. The END command can be included in control structures to end the program if a condition is met. This will be discussed with control structures.

Typical BASIC keywords[edit]

Data manipulation

- LET—assigns a value (which may be the result of an expression) to a variable.

- DATA—holds a list of values which are assigned sequentially using the READ command.

Program flow control

- IF ... THEN ... {ELSE}—used to perform comparisons or make decisions.
- FOR ... TO ... {STEP} ... NEXT—repeat a section of code a given number of times. A variable that acts as a counter is available within the loop.
- WHILE ... WEND and REPEAT ... UNTIL—repeat a section of code while the specified condition is true. The condition may be evaluated before each iteration of the loop, or after.
- DO ... LOOP {WHILE} or {UNTIL}—repeat a section of code Forever or While/Until the specified condition is true. The condition may be evaluated before each iteration of the loop, or after.
- GOTO—jumps to a numbered or labelled line in the program.
- GOSUB—jumps to a numbered or labelled line, executes the code it finds there until it reaches a RETURN Command, on which it jumps back to the operator following the GOSUB – either after a colon, or on the next line. This is used to implement subroutines.
- ON ... GOTO/GOSUB—chooses where to jump based on the specified conditions. See Switch statement for other forms.
- DEF FN—a pair of keywords introduced in the early 1960s to define functions. The original BASIC functions were modeled on FORTRAN single-line functions. BASIC functions were one expression with variable arguments, rather than subroutines, with a syntax on the model of DEF FND(x) = x*x at the beginning of a program. Function names were originally restricted to FN+one letter.

Input and output

- LIST—displays all inputted code.
- PRINT—displays a message on the screen or other output device.

- INPUT—asks the user to enter the value of a variable. The statement may include a prompt message.
- TAB or AT: sets the position where the next character will be shown on the screen or printed on paper.

List of functions

- ABS—Absolute value
- ATN—Arctangent value (result in radians)
- COS—Cosine value (argument in radians)
- EXP—Exponential value
- INT—Integer value
- LOG—Natural Logarithmic value
- RND—Random value
- SIN—Sine value (argument in radians)
- SQR—Square root value
- TAN—Tangent value (argument in radians)

RelationalOperators:

Relational operators are used for comparison of two values. Let's see them one by one:

- **'=='** operator checks whether the two given operands are equal or not. If so, it returns true. Otherwise it returns false. For example, **5==5** will return true.
- **'!='** operator checks whether the two given operands are equal or not. If not, it returns true. Otherwise it returns false. It is the exact boolean complement of the **'=='** operator. For example, **5!=5** will return false.
- **'>'** operator checks whether the first operand is greater than the second operand. If so, it returns true. Otherwise it returns false. For example, **6>5** will return true.
- **'<'** operator checks whether the first operand is lesser than the second operand. If so, it returns true. Otherwise it returns false. For example, **6<5** will return false.

- **'>='** operator checks whether the first operand is greater than or equal to the second operand. If so, it returns true. Otherwise it returns false. For example, **5>=5** will return true.
- **'<='** operator checks whether the first operand is lesser than or equal to the second operand. If so, it returns true. Otherwise it returns false. For example, **5<=5** will also return true.
-

LogicalOperators:

They are used to combine two or more conditions/constraints or to complement the evaluation of the original condition in consideration. They are described below:

- **Logical AND:** The **'&&'** operator returns true when both the conditions in consideration are satisfied. Otherwise it returns false. For example, **a && b** returns true when both a and b are true (i.e. non-zero).
- **Logical OR:** The **'||'** operator returns true when one (or both) of the conditions in consideration is satisfied. Otherwise it returns false. For example, **a || b** returns true if one of a or b is true (i.e. non-zero). Of course, it returns true when both a and b are true.
- **Logical NOT:** The **'!'** operator returns true the condition in consideration is not satisfied. Otherwise it returns false. For example, **!a** returns true if a is false, i.e. when a=0

The String Type

• One of the most important data types in any programming language is the string type.

• The domain of the string type is all sequences of characters. In JavaScript, you create a string simply by including that sequence of characters inside quotation marks, as in "Jerry".

• The set of operations that can be applied to strings is large, but you don't need to know the entire set. In fact, for the first five chapters in the text, the only string operation you need to know is concatenation, as described on the next slide. You will learn about other operations in Chapter 6.

• All values—including numbers, strings, graphical objects, and values of many other types—can be assigned to variables, passed as arguments to functions, and returned as results.

Concatenation

• One of the most useful operations available for strings is concatenation, which consists of combining two strings end to end with no intervening characters.

• Concatenation is built into JavaScript using the + operator. For example, the expression "ABC" + "DEF" returns the string "ABCDEF".

• If you use + with numeric operands, it signifies addition. If at least one of its operands is a string, JavaScript interprets + as concatenation. It automatically converts the other operand to a string and concatenates the two strings, so that "Catch" + -22 "Catch-22"

The Graphics Model

• SJS uses the same graphics model that we have used for the last decade, which is based on the metaphor of a collage.

• A collage is similar to a child's felt board that serves as a backdrop for colored shapes that stick to the felt surface. As an example, the following diagram illustrates the process of adding a blue rectangle and a red oval to a felt board:

• Note that newer objects can obscure those added earlier. This layering arrangement is called the stacking order.

Interpretation and compilation are properties of the implementation of a language

It's not accurate to say that a *language* is interpreted or compiled because interpretation and compilation are both properties of the *implementation* of that particular language, and ***not*** a property of the *language* itself. So, in theory, any language can be compiled or interpreted – it just depends on what the particular implementation that you are using does.

What exactly is compilation?

In a compiled implementation of a language, a compiler will translate the program directly into code that is *specific to the target machine*, which is also known as machine code – basically code that is specific to a given processor and operating system. Then the computer will run the machine code on its own.

What exactly is interpretation?

In an interpreted implementation of a language, the source code is not directly run by the target machine. What happens instead is that another program reads and then executes the original source code. This other program is also known as the interpreter. The interpreter is

usually written specifically for the native machine. As an example, you could consider the multiplier operation – the "*". If the interpreter sees this in your code, then at run time it would call its own definition of the multiplier function – maybe something called "multiply(x,y)". And then that "multiply(x,y)" would execute the machine code's equivalent of the multiply instruction.

In interpretation, the original source code is also typically converted into some intermediate code which is then processed by an interpreter that converts the intermediate code into machine specific code.

Are interpreters and compilers mutually exclusive?

No, they are not mutually exclusive – because there are some interpreters which also do some translation work, just like compilers normally do. So, the work that a compiler typically does can also be done by an interpreter.

Can a compiled implementation of a language do the same things as an interpreted implementation of a language?

Yes, you can accomplish exactly the same things with both interpreted and compiled languages. But, in general terms there are advantages and disadvantages of both compiled and interpreted languages.

So, what are the advantages and disadvantages of compilation and interpretation?

Because comparing compilation and interpretation is so dependent on the specific implementation of the interpreter and compiler, we can only compare compilation and interpretation in very general terms – there will be exceptions to what we say below, but in general these things are true:

- Compiled implementations of languages have the advantage of being faster because they translate directly to the native code of the specific machine.
- Interpreted implementations of languages tend to be more portable as well.
- Interpreted implementations of languages are generally easier to create because writing compilers is difficult.
- Debugging is the routine process of locating and removing computer program bugs, errors or abnormalities, which is methodically handled by software programmers via debugging tools. Debugging checks, detects and corrects errors or bugs to allow proper program operation according to set specifications.

- Debugging is also known as debug.

Techopedia explains *Debugging*

Developing software programs undergo heavy testing, updating, troubleshooting and maintenance. Normally, software contains errors and bugs, which are routinely removed. In the debugging process, complete software programs are regularly compiled and executed to identify and rectify issues. Large software programs, which contain millions of source code lines, are divided into small components. For efficiency, each component is debugged separately at first, followed by the program as a whole.

Matrix addition in C

Matrix addition in C: C program to add two matrices, i.e., compute the sum of two matrices and then print it. Firstly a user will be asked to enter the order of matrix (number of rows and columns) and then two matrices. For example, if a user input order as 2, 2, i.e., two rows and two columns and matrices as

First matrix:

1 2

3 4

Second matrix:

4 5

-1 5

then the output of the program (Summation of the two matrices) is:

5 7

2 9

Matrices are frequently used in programming to represent graph data structure, in solving equations and in many other ways.

Addition of two matrix in C

C program for matrix addition:

```
#include <stdio.h>
 int main()
{
int m, n, c, d, first[10][10], second[10][10], sum[10][10];
 printf("Enter the number of rows and columns of matrix\n");
```

```
scanf("%d%d",&m,&n);
printf("Enter the elements of first matrix\n");
 for(c =0; c < m;c++)
for(d =0; d < n; d++)
scanf("%d",&first[c][d]);
 printf("Enter the elements of second matrix\n");
 for(c =0; c < m;c++)
for(d =0; d < n; d++)
scanf("%d",&second[c][d]);
 printf("Sum of entered matrices:-\n");
 for(c =0; c < m;c++){
for(d =0; d < n; d++){
sum[c][d]= first[c][d]+ second[c][d];
printf("%d\t", sum[c][d]);
}
printf("\n");
}
 return0;
}
```

Matrix multiplication in C language: C program to multiply two matrices (two-dimensional array) which will be entered by a user. The user will enter the order of a matrix and then its elements and similarly input the second matrix. If the entered orders of two matrices are such that they can't be multiplied by each other, then an error message is displayed on the screen. You may have studied the method to multiply matrices in Mathematics.

Matrix multiplication in C language

```
#include <stdio.h>


int main()
{
int m, n, p, q, c, d, k, sum =0;
```

```
int first[10][10], second[10][10], multiply[10][10];

printf("Enter number of rows and columns of first matrix\n");
scanf("%d%d",&m,&n);
printf("Enter elements of first matrix\n");

for(c =0; c < m;c++)
for(d =0; d < n; d++)
scanf("%d",&first[c][d]);

printf("Enter number of rows and columns of second matrix\n");
scanf("%d%d",&p,&q);

if(n != p)
printf("The matrices can't be multiplied with each other.\n");
else
{
printf("Enter elements of second matrix\n");

for(c =0; c < p;c++)
for(d =0; d < q; d++)
scanf("%d",&second[c][d]);

for(c =0; c < m;c++){
for(d =0; d < q; d++){
for(k =0; k < p; k++){
sum= sum + first[c][k]*second[k][d];
}
 multiply[c][d]= sum;
sum=0;
}
```

```
}
 printf("Product of the matrices:\n");
 for(c =0; c < m;c++){
for(d =0; d < q; d++)
printf("%d\t", multiply[c][d]);
 printf("\n");
}
}
 return0;
}
```

atistical analysis

Measures of Location: Averages

The **average** gives you information about the size of the effect of whatever you are testing, in other words, whether it is large or small. There are three measures of average: mean, median and mode.

See our page on **Averages** for more about calculating each one, and for a quick calculator.

When most people say average, they are talking about the **mean**. It has the advantage that it uses all the data values obtained and can be used for further statistical analysis. However, it can be skewed by 'outliers', values which are atypically large or small.

As a result, researchers sometimes use the **median** instead. This is the mid-point of all the data. The median is not skewed by extreme values, but it is harder to use for further statistical analysis.

The **mode** is the most common value in a data set. It cannot be used for further statistical analysis.

The values of mean, median and mode are **not** the same, which is why it is really important to be clear which 'average' you are talking about.

Measures of Spread: Range, Variance and Standard Deviation

Researchers often want to look at the **spread** of the data, that is, how widely the data are spread across the whole possible measurement scale.

There are three measures which are often used for this:

The **range** is the difference between the largest and smallest values. Researchers often quote the **interquartile range**, which is the range of the middle half of the data, from 25%, the lower quartile, up to 75%, the upper quartile, of the values (the median is the 50% value). To find the quartiles, use the same procedure as for the median, but take the quarter- and three-quarter-point instead of the mid-point.

The **standard deviation** measures the average spread around the mean, and therefore gives a sense of the 'typical' distance from the mean.

The **variance** is the square of the standard deviation. They are calculated by:

1. calculating the difference of each value from the mean;
2. squaring each one (to eliminate any difference between those above and below the mean);
3. summing the squared differences;
4. dividing by the number of items minus one.

This gives the **variance**.

To calculate the **standard deviation**, take the square root of the variance.

Skew

The **skew** measures how symmetrical the data set is, or whether it has more high values, or more low values. A sample with more low values is described as negatively skewed and a sample with more high values as positively skewed.

Generally speaking, the more skewed the sample, the less the mean, median and mode will coincide.

More Advanced Analysis

Once you have calculated some basic values of **location**, such as mean or median, **spread**, such as range and variance, and established the level of **skew**, you can move to more advanced statistical analysis, and start to look for patterns in the data.

POSSIBLE QUESTIONS

PART A (20 multiple choice questions)

Online Examinations

PART B (2 marks questions)

1. What is BASIC?
2. Give the different types of constant in BASIC?
3. Mention the rules in constructing of integer variables?
4. State square matrix?
5. What is meant by binary?

PART C (6 marks questions)

1. Explain three types of BASIC constants.
2. Write note on BASIC variables.
3. What is meant by matrix? Give the many types of matrices depending upon their characteristics.
4. What is meant by BASIC language? Give the many types of special character.
5. Give a brief account on graphics.
6. Write note on matrix addition.
7. Explain (i) Integer constant (ii) Real constant
8. Describe the following

(i) Integer variable name (ii) Real variable name

9. Discuss briefly about operators.
10. Give a detailed an account on matrix multiplication.

KARPAGAM ACADEMY OF HIGHER EDUCATION

Class: III B.Sc Chemistry Course Name: Applications of Computers in Chemistry

Course Code: 16CHU502B Unit: I Batch-2016-2019

1. CPU stands for
a. central processing unit b. center processing unit
c. central programming unit d. center programming unit

2. A variable name can have?
a. Any special symbol b. Blank space c. Comma (,) **d. Underscore**

3. In C language, one of the following is not a valid data type
a. Long b. Float c. Double d. Char

4. The format string %lf is used for?
a. Float b. Double c. Unsigned int **d. Long double**

5. Which data type is not a primary data type?
a. Int **b. Array** c. Float d. Char

6. Which of the format string is not valid?
a. %ld b. %lf c. %lu **d. %lc**

7. Which is the valid string data?
a. 'A' b. A **c. "A"** d. ‘A”

8. How much memory is required to store a value of type double?
a. 4 bytes b. 6 bytes **c. 8 bytes** d. 10 bytes

9. The modifier which is used to declare a variable as constant
a. Short b. Signed c. Unsigned **d. Const**

10. Which of the following statement is not true about preprocessor directives?
a. These are lines read and processed by the preprocessor
b. They do not produce any code by themselves
c. These must be written on their own line
d. They end with a semicolon

11. Which is false?
a. Constant variables need not be defined as they are declared and can be defined later
b. Global constant variables are initialised to zero
c. const keyword is used to define constant values
d. You cannot reassign a value to a constant variable

12. An integer constant must have atleast

a. 1 digit b. 2 digit c. 3 digit d. 5 digit

13. The range of integer constant is
a. -43757 to +43756 **b. -32768 to + 32767** c. -23589 to +23588 d. -14567 to +14566

14. Which of the following one is not true in integer constant?
a. 145 b. -400 c. +40 **d. 425+**

15. If no sign proceeds an integer constant it is assumed to be
a. negative b. positive and negative **c. positive** d. neutral

16. Which of the following one is true in integer constant?
a. 995, 72 b. 6.7.8 c. 234V6 **d. +40**

17. The GW-BASIC was developed by
a. John Kemeny b. Kenneth Kurtz **c. Bill Gates** d. Satya Nadella

18. The presence of pulse is referesented as
a. 1 b. 3 c. 2 d. 0

19. Binary comes from the word of
a. Latin b. English c. Spanish d. Greek

20. Computer Engineer call a single 0 or 1 a
a. binary b. bytes **c. bit** d. constants

21. Graphics are very important part of
a. BASICA b. turbo BASIC c. GW-BASIC **d. Q-BASIC**

22. The capacity of a computer's memory is measured in byte "K" usually stand for
a. 500 **b. 1000** c. 2000 d. 100

23. The maximum length of a string constant can be
a. 154 character b. 354 character c. 454 character **d. 254 character**

24. Range of real constants expressed in exponential form is
a. 10^{-38} to 10^{+38} b. 10^{-48} to 10^{+48} c. 10^{-28} to 10^{+28} d. 10^{-18} to 10^{+18}

25. In real constant the mantissa part and exponential part should be separated by a letter
a. B b. C c. A **d. E**

26. A matrix whose rows and columns are equal is known as
a. row matrix b. constant matrix **c. square matrix** d. zero matrix

27. When 1100010 is divided by 0101, what will be the decimal remainder?

a. 2 **b.3** c.4 d.6

28. Which of the following one is not allowed in arithmetic expression?
a. integer b. real **c. string** d. characters

29. In integer variable name, the last character must be a
a. @ sign b. # sign c. $ sign **d. % sign**

30. The integer variable name any combination of
a. 1 to 40 digits b. 1 to 10 digits c. 1 to 20 digits d. 1 to 30 digits

31. The smallest individual unit in a program is known as
a. bit **b. token** c. bytes d. binary

32. In computer, nibble has
a. 4 bits b. 8 bits c. 2 bits d. 16 bits

33. In computer, a byte consist
a. 8 bits b. 16 bits c. 64 bits d. 16 bits

34. A computer may consist two nibble has
a. 8 bit word b. 16 bit word c. 32 bit word d. 64 bit word

35. A computer may consist of a single nibble has
a. 32 bit word b. 8 bit word c.16 bit word **d. 4 bit word**

36. BASIC is an acronym was developed by
a. John Kemeny & Bill Gates b. Kenneth Kurtz & Bill Gates
c. John Kemeny & Kenneth Kurtz d. Bill Gates & Satyan Nadella

37. BASIC is an acronym that stands for
a. Beginners All Purpose Symbolic Instruction Code
b. Beginners And Purpose Symbolic Instruction Code
c. Beginners All Purpose Symbolic Intimation Code
d. Beginners And Purpose Symbolic Instruction Code

38. TROFF is a
a. WORM b. BUG c. ERROR **d. DEBUGGING**

39. A square matrix is also called as
a. scalar matrix b. zero matrix **c. symmetric matrix** d. constant matrix

40. A square matrix having value 1 to the diagonals of the matrix and zero everywhere else is called
a. unit matrix b. constant matrix c. zero matrix d. scalar matrix

41. A matrix whose numbers of rows and columns are not equal is called
a. Diagonal b. column c. row **d. rectangular**

42. Graphics is a very important part of
a. Q-BASIC language b. BASICA language
c. turbo BASIC language d. GW-BASIC language

43. Which of the following one is not true in graphics?
a. color graphic adapter b. enhanced graphic adapter
c. video graphic arrays **d. audio graphic adapter**

44. Graphics is one of the part the following language
a. GW-BASIC b. turbo BASIC **c. Q-BASIC** d. BASICA

45. C:\DOS>MODE CO80 will take
a. 50 columns X 25 rows graphics mode b. 80 columns X 25 rows graphics mode
c. 50 columns X 50 rows graphics mode d. 100 columns X 25 rows graphics mode

46. C:\DOS>MODE CO40 will take
a. 50 columns X 25 rows graphics mode **b. 40 columns X 25 rows graphics mode**
c. 40 columns X 50 rows graphics mode d. 80 columns X 25 rows graphics mode

47. The normal computer coordinate system is call
a. screen coordinate system b. science coordinate system
c. screen control system d. science control system

48. In computer graphics, the default screen mode is
a. 1 b. 2 c. 3 **d.0**

49. In computer graphics, the numbers of rows are 200 whereas columns can vary e
a. either 189 or 639 b. either 179 or 639 **c. either 199 or 639** d. either 169 or 639

50. The following screen modes are mostly used in computer graphics.
a. 0, 1 and 2 b. 0, 1, 2 and 3 c. 1, 2 and 3 d. 0 and 1

51. In computer graphics, the text mode is represented as
a. 1 **b. 0** c.2 d. -1

52. Medium-resolution graphic mode is represented as
a. 1 b. 0 c.2 d. -1

53. High-resolution graphic mode is represented as
a. 1 b. 0 c.-1 **d. 2**

54. If all the elements of a matrix are 1 then the matrix is called
a. diagonal matrix b. scalar matrix **c. constant matrix** d. identity matrix

55. The other name of diagonal matrix is
a. symmetric matrix b. zero matrix c. square matrix **d. scalar matrix**

56. The other name of square matrix is
a. skew matrix b. symmetric matrix c. zero matrix d. unit matrix

57. In integer variable name, the last character must be a
a. @ sign b. # sign c. $ sign **d. % sign**

58. An integer constant must have atleast
a. 1 digit b. 2 digit c. 3 digit d. 5 digit

59. The following one is not true
a. analog computers b. digital computer c. hybrid computer **d. android computer**

60. String constant has
a. 154 character b. 354 character c. 454 character **d. 254 character**

UNIT II

Syllabus

Numerical methods:

Roots of equations: Numerical methods for roots of equations: Quadratic formula, iterative method, Newton-Raphson method, Binary bisection and Regula-Falsi.

The roots of a function $f(x)$ are defined as the values for which the value of the function becomes equal to zero. So, finding the roots of $f(x)$ means solving the equation

$$f(x) = 0.$$

Example 1: If $f(x) = ax^2+bx+c$ is a quadratic polynomial, the roots are given by the well-known formula

$$x_1, x_2 = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Example 2: For a polynomial of degree 3 or higher, it is sometimes (but not very often!) possible to find the roots by factorising the polynomial

$f(x) = x^3 - 6x^2 + 11x - 6 = (x-1)(x-2)(x-3)$ so the roots are 1, 2 and 3
$f(x) = x^4 - 16 = (x^2-4)(x^2+4)$ so the roots are 2 and -2

For a large number of problems it is, however, not possible to find exact values for the roots of the function so we have to find approximations instead.

The function f(x) of the equation will usually have at least one continuous derivative, and often we will have some estimate of the root that is being sought. By using this information, most numerical methods for compute a sequence of increasingly accurate estimates of the root. These methods are called iteration methods.

Bisection Method

Bisection method is a method provides practical method to find roots of equation. This method also helps to prove the intermediate theorem. Among all the numeral methods bisection method is the simplest one to solve the transcendental equations.

This method helps to find the zero of a function by repeatedly halving the selected interval. The bisection method is a straightforward technique for finding numerical solutions to equations in one unknown.

It works by narrowing the gap between pos and neg until it closes in on the correct answer. It narrow the gap by taking average of pos and neg. The average may be positive or negative. It is slow compared with other numerical techniques. In this section we will be dealing with bisection method with some solved problems.

Definition

In mathematics, bisection method used to find the roots of an equation. It separate intervals and select a sub-interval in which root of the equation lies. It is simple and also relatively slow method. It is based on the intermediate theorem for continuous functions. This is also called as root finding method or binary search method, dichotomy method or the interval halving method. Consider a continuous function g which is defined on closed interval [c, d] is given with g(c) and g(d) of different sign. Then by intermediate theorem, there exists a point m belongs to (c, d) for whichg(m)=0.

When we find more than one root in the selected interval, for simplicity we assume that the root in the selected interval is unique. In this case m is a unique root of function. **Let us see how this**

method is different from intermediate value theorem:

Bisection method is the one of the applications of the intermediate theorem. Intermediate value theorem states that: if a function defined and continuous on a closed interval, say [m, n], then there exist a number between [m, n], say t. The function has at least one solution t in the open interval (m, n).

Whereas bisection method find an approximation to a zero of a continuous function. In case of IVT function could jump over some values in the interior of the interval, which possibility arises if the function is discontinuous at both endpoints.

For a continuous function g(x)

Step 1: Find two points, say m and n st m < n and g(m) * g(n) < 0

Step 2: Find the midpoint of m and n, say t.

Step 3: t is root of function if g(t) = 0, else follow the next step.

Step 4: Divide the interval [m, n]. If g(t) * g(n) < 0, let m = t, else if g(t) * g(m) < 0 then let n = t.

Step 5: Repeat above two steps until g(t) = 0.

Example 1: Find the root of the polynomial, g(x) = x33 - 5 + 3x using bisection method. Where m = 1 and n = 2?

Solution: First find the value of g(x) at m = 1 and n = 2

g(1) = 133 - 5 + 3*1 = -1 < 0

g(2) = 233 - 5 + 3*2 = 9 > 0

Since function is continuous, its root lies in the interval [1, 2].

Let t be the average of the interval i.e t = 1+221+22 = 1.5

The value of the function at t is

g(1.5) = (1.5)33 - 5 + 3*(1.5) = 2.875

As g(t) is negative so n = 2 is replaced with t = 1.5 for the next iteration. Make sure that g(m) and g(n) have opposite signs.

Below table contains nine iterations of the function.

Iteration	**m**	**n**	**t**	**g(m)**	**g(n)**	**g(t)**
1	1	2	1.5	-1	9	2.875
2	1	1.5	1.25	-1	2.875	0.703125
3	1	1.25	1.125	-1	0.703125	-0.201171875
4	1.125	1.25	1.1875	-0.201171875	0.703125	0.237060546875
5	1.125	1.1875	1.15625	-0.201171875	0.237060546875	0.014556884765625
6	1.125	1.15625	1.140625	-0.201171875	0.014556884765625	-0.094142913818359 4
7	1.140625	1.15625	1.1484375	-0.0941429138183594	0.014556884765625	-0.040003299713134 8
8	1.148437	1.156	1.1523437	-	0.014556884765	-

	5	25	5	0.0400032997131348	625	0.0127759575843811
9	1.15234375	1.15625	1.154296875	-0.0127759575843811	0.0145568847656625	0.000877253711223602

Therefore we chose m = 1.15234375 to be our approximated solution.

Example 2

Consider finding the root of f(x) = x^2 - 3. Let ε_{step} = 0.01, ε_{abs} = 0.01 and start with the interval [1, 2].

Table 1. Bisection method applied to f(x) = x^2 - 3.

a	***b***	**f(*a*)**	**f(*b*)**	***c* = (a + b)/2**	**f(*c*)**	**Update**	**new b − a**
1.0	2.0	-2.0	1.0	1.5	-0.75	a = c	0.5
1.5	2.0	-0.75	1.0	1.75	0.062	b = c	0.25
1.5	1.75	-0.75	0.0625	1.625	-0.359	a = c	0.125
1.625	1.75	-0.3594	0.0625	1.6875	-0.1523	a = c	0.0625
1.6875	1.75	-0.1523	0.0625	1.7188	-0.0457	a = c	0.0313
1.7188	1.75	-0.0457	0.0625	1.7344	0.0081	b = c	0.0156
1.71988/td>	1.7344	-0.0457	0.0081	1.7266	-0.0189	a = c	0.0078

Example 3: Find a root of x^2 - 3 = 0 for x ∈∈ [1, 2].

Solution:

Let g(x) = x^2 - 3

First find the value of g(x) at m = 1 and n = 2

g(1) = 122 - 3 = -2 < 0

g(2) = 222 - 3 = 1 > 0

Since function is continuous, its root lies in the interval [1, 2].

Let t be the average of the interval i.e t = 1+2/2 = 1.5

The value of the function at t is

g(1.5) = (1.5)22 - 3 = -0.75 < 0

g(t) is negative so n is replaced with t = 1.5 for the next iteration.

Below table contains nine iterations of the function.

Iteration	m	n	c	g(m)	g(n)	g(t)
1	1	2	1.5	-2	1	-0.75
2	1.5	2	1.75	-0.75	1	0.062
3	1.5	1.75	1.625	-0.75	0.0625	-0.359
4	1.625	1.75	1.6875	-0.3594	0.0625	-0.1523
5	1.6875	1.75	1.7188	-01523	0.0625	-0.0457
6	1.7188	1.75	1.7344	-0.0457	0.0625	0.0081
7	1.7188	1.7344	1.7266	-0.0457	0.0081	-0.0189

At seventh iteration, we got the final interval, [1.7266, 1.7344]. Its width is less than 0.01 and the value of g(x) also less than 0.01.

Therefore we chose n = 1.7344 to be our approximated solution.

Advantages and disadvantages of the bisection method

1 The method is guaranteed to converge

2 The error bound decreases by half with each iteration

3 The bisection method converges very slowly

4 The bisection method cannot detect multiple roots

0.1 Newton Raphson Method

The Newton Raphson method is for solving equations of the form $f(x) = 0$. We make an initial guess for the root we are trying to find, and we call this initial guess x_0. The sequence $x_0, x_1, x_2, x_3, \ldots$ generated in the manner described below should converge to the exact root.
To implement it analytically we need a formula for each approximation in terms of the previous one, i.e. we need x_{n+1} in terms of x_n.
The equation of the tangent line to the graph $y = f(x)$ at the point $(x_0, f(x_0))$ is

$$y - f(x_0) = f'(x_0)(x - x_0)$$

The tangent line intersects the x-axis when $y = 0$ and $x = x_1$, so

$$-f(x_0) = f'(x_0)(x_1 - x_0)$$

Solving this for x_1 gives

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$$

and, more generally,

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

You should **memorize** the above formula. Its application to solving equations of the form $f(x) = 0$, as we now demonstrate, is called the **Newton Raphson method**. It is guaranteed to converge if the initial guess x_0 is close enough, but it is hard to make a clear statement about what we mean by 'close enough' because this is highly problem specific. A sketch of the graph of $f(x)$ can help us decide on an appropriate initial guess x_0 for a particular problem.

0.2 Example

Let us solve $x^3 - x - 1 = 0$ for x.
In this case $f(x) = x^3 - x - 1$, so $f'(x) = 3x^2 - 1$. So the recursion formula (1) becomes

$$x_{n+1} = x_n - \frac{(x_n^3 - x_n - 1)}{3x_n^2 - 1}$$

Need to decide on an appropriate initial guess x_0 for this problem. A rough graph can help. Note that $f(1) = -1 < 0$ and $f(2) = 5 > 0$. Therefore, a root of $f(x) = 0$ must exist between 1 and 2. Let us take $x_0 = 1$ as our initial guess. Then

$$x_1 = x_0 - \frac{(x_0^3 - x_0 - 1)}{3x_0^2 - 1}$$

and with $x_0 = 1$ we get $x_1 = 1.5$.
Now

$$x_2 = x_1 - \frac{(x_1^3 - x_1 - 1)}{3x_1^2 - 1}$$

and with $x_1 = 1.5$ we get $x_2 = 1.34783$. For the next stage,

$$x_3 = x_2 - \frac{(x_2^3 - x_2 - 1)}{3x_2^2 - 1}$$

and with the value just found for x_2, we find $x_3 = 1.32520$.
Carrying on, we find that $x_4 = 1.32472$, $x_5 = 1.32472$, etc. We can stop when the digits stop changing to the required degree of accuracy. We conclude that the root is 1.32472 to 5 decimal places.

Example: 3

The Newton-Raphson method works most of the time if your initial guess is good enough. Occasionally it fails but sometimes you can make it work by changing the initial guess. Let's try to solve $x = \tan x$ for x. In other words, we solve $f(x) = 0$ where $f(x) = x - \tan x$. The recursion formula (1) becomes

$$x_{n+1} = x_n - \frac{(x_n - \tan x_n)}{1 - \sec^2 x_n}$$

Let's try an initial guess of $x_0 = 4$. With this initial guess we find that $x_1 = 6.12016$, $x_2 = 238.40428$, $x_3 = 1957.26490$, etc. Clearly these numbers are not converging. We need a new initial guess. Let's try $x_0 = 4.6$. Then we find $x_1 = 4.54573$, $x_2 = 4.50615$, $x_3 = 4.49417$, $x_4 = 4.49341$, $x_5 = 4.49341$, etc. A couple of further iterations will confirm that the digits are no longer changing to 5 decimal places. As a result, we conclude that a root of $x = \tan x$ is $x = 4.49341$ to 5 decimal places.

Background

Recall that the equation of a straight line is given by the equation

$$y = mx + n$$

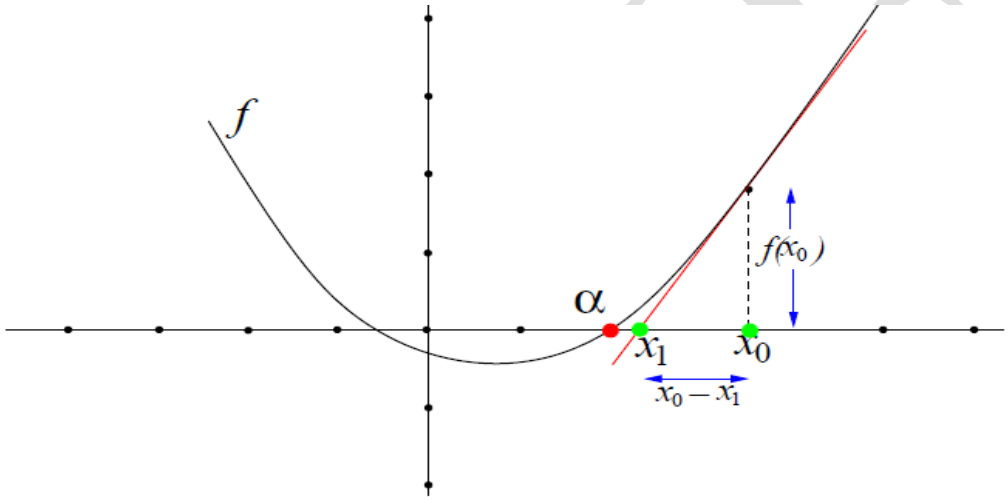
where *m* is called the *slope* of the line. (This means that all points (*x*,*y*) on the line satisfy the equation above.)

If we know the slope *m* and one point (*x*0,*y*0) on the line, equation (1) becomes

$$y - y_0 = m(x - x_0)$$

Idea behind Newton's method

Assume we need to find a root of the equation $f(x) = 0$. Consider the graph of the function $f(x)$ and an initial estimate of the root, $x0$. To improve this estimate, take the tangent to the graph of $f(x)$ through the point $(x0, f(x0)$ and let $x1$ be the point where this line crosses the horizontal axis.



Exercise 1: Find a root of the equation

$$x^6 - x - 1 = 0$$

(Note that the true root is $\alpha = 1.134724138$.)

We already know that this equation has a root between 1 and 2 so we take $x0 = 1.5$ as our first approximation.

For each iteration we calculate the value $xn-xn-1$ which, as we shall see later, is a good approximation to the absolute error □$-xn-1$.

The iterations for Newton's algorithm are shown in the table below

n	x_n	$f(x_n)$	$x_n - x_{n-1}$	$\alpha - x_{n-1}$
0	1.5	8.890625		
1	1.30049	2.54	-2.0E-1	-3.65E-1
2	1.18148	5.38E-1	-1.19E-1	-1.66E-1
3	1.13945	4.92E-2	-4.20E-2	-4.68E-2
4	1.13477	5.50E-4	-4.68E-3	-4.73E-3
5	1.13472	7.11E-8	-5.35E-5	-5.35E-5
6	1.13472	1.55E-15	-6.91E-9	-6.91E-9

Advantages and disadvantages of Newton's method:

The error decreases rapidly with each iteration

Newton's method is very fast. (Compare with bisection method!)

Unfortunately, for bad choices of $x0$ (the initial guess) the method can fail to converge!

Therefore the choice of $x0$ is VERY IMPORTANT!

Each iteration of Newton's method requires two function evaluations, while the bisection method requires only one.

Note: A good strategy for avoiding failure to converge would be to use the bisection method for a few steps (to give an initial estimate and make sure the sequence of guesses is going in the right direction) followed by Newton's method, which should converge very fast at this point.

Notes on root finding

• Roots of equations can be either real or complex.
• Recall is a real number; is a complex number, where .
• A large variety of root finding algorithms exist, we will look at only a few.
• Each algorithm has advantages/disadvantages, possible restrictions, etc.

Method	Must Specify Interval Containing Root	$f^{(1)}(x)$ Continuous	Features
Bisection	yes	no	Robust
Newton-Raphson (Newton)	no	yes	Fast and applies to complex roots

The secant method

Idea behind the secant method

Assume we need to find a root of the equation $f(x) = 0$, called □. Consider the graph of the function $f(x)$ and two initial estimates of the root, $x0$ and $x1$. The two points $(x0, f(x0))$ and $(x1, f(x1))$ on the graph of

$f(x)$ determine a straight line, called a secant line which can be viewed as an approximation to the graph. The point $x2$ where this secant line crosses the x axis is then an approximation for the root .

This is the same idea as in Newton's method, where the tangent line has been approximated by a secant line.

The equation of the secant line will be given by

$$y = f(x_1) + (x - x_1)\frac{f(x_1) - f(x_0)}{x_1 - x_0}$$

so

$$x_2 = x_1 - f(x_1)\frac{x_1 - x_0}{f(x_1) - f(x_0)}$$

Then take x_1 and x_2 as the next two estimates and continue the procedure. The general iteration will be given by

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

and so on.

Advantages and disadvantages:

1The error decreases slowly at first but then rapidly after a few iterations.

2 The secant method is slower than Newton's method but faster than the bisection method.

3 Each iteration of Newton's method requires two function evaluations, while the secant method requires

only one

4 The secant method does not require differentiation.

Newton's Method

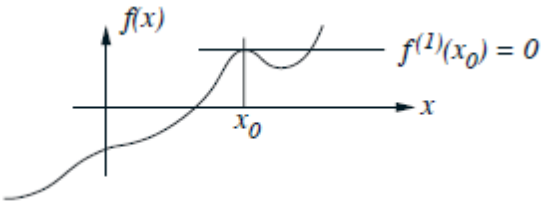
• Convergence rate for Newton's method is ***very high***!!

• Error estimates are very good (however will be case dependent on the form of the function

• Newton's method can find complex roots.

Problems with Newton's Method

• If the local min/max is selected as an initial guess



• The slope at x_0 does not intersect with *x*-axis!

• The formula for x_1 will lead to an infinite value.

The convergce process in the bisection method is very slow. It depends only on the choice of end points of the interval [a,b]. The function f(x) does not have any role in finding the point c (which is just the mid-point of a and b). It is used only to decide the next smaller interval [a,c] or [c,b]. A better approximation to c can be obtained by taking the straight line L joining the points (a,f(a)) and (b,f(b)) intersecting the x-axis. To obtain the value of c we can equate the two expressions of the slope m of the line L.

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

$$\Rightarrow (c-b) * (f(b)-f(a)) = -(b-a) * f(b)$$

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

Now the next smaller interval which brackets the root can be obtained by checking

$$f(a) * f(b) < 0 \text{ then } b = c$$

> 0 then a = c

= 0 then c is the root.

Selecting **c** by the above expression is called Regula-Falsi method or False position method.

Algorithm - False Position Scheme

Given a function f (x) continuos on an interval [a,b] such that f (a) * f (b) < 0

Do

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

if f (a) * f (c) < 0 then b = c

else a = c

while (none of the convergence criterion C1, C2 or C3 is satisfied)

The false position method is again bound to converge because it brackets the root in the whole of its convergence process.

The false position method is again bound to converge because it brackets the root in the whole of its convergence process.

Example

Find a root of 3x + sin(x) - exp(x) = 0.

The graph of this equation is given in the figure.

From this it's clear that there is a root between 0 and 0.5 and also another root between 1.5 and 2.0. Now let us consider the function f (x) in the interval [0, 0.5] where f (0) * f (0.5) is less than zero and use the regula-falsi scheme to obtain the zero of f (x) = 0.

Iteration No.	a	b	c	f(a) * f(c)
1	0	0.5	0.376	1.38 (+ve)

2	0.376	0.5	0.36	-0.102 (-ve)
3	0.376	0.36	**0.36**	-0.085 (-ve)

So one of the roots of 3x + sin(x) - exp(x) = 0 is approximately **0.36**.

Note : Although the length of the interval is getting smaller in each iteration, it is possible that it may not go to zero. If the graph y = f(x) is concave near the root 's', one of the endpoints becomes fixed and the other end marches towards the root.

Background. The Regula Falsi method is one of the bracketing methods for finding roots of equations.

Implementation. Given a function f(x) and an interval which might contain a root, perform a predetermined number of iterations using the Regula Falsi method.

Limitations. Investigate the result of applying the Regula Falsi method over an interval where there is a discontinuity. Apply the Regula Falsi method for a function using an interval where there are distinct roots. Apply the Regula Falsi method over a "large" interval.

Algorithm:

Given a function $f(x)$ continuous on an interval $[a,b]$ satisfying the criteria $f(a)f(b) < 0$, carry out the following steps to find the root ξ of $f(x)$ in $[a,b]$:

(1) Set $a_0 = a, \quad b_0 = b$

(2) For n = 0,1,2.... until convergence criteria is satisfied , do:

(a) Compute $= \dfrac{f(2).1 - f(1).2}{f(2) - f(1)} = 1.47826087$

(b) If $(f(a_n)f(w) \leq 0)$, then set $a_{n+1} = a_n \quad ; \quad b_{n+1} = w$

otherwise set $a_{n+1} = w \quad ; \quad b_{n+1} = b_n$

Note:

Use any one of the convergence criteria discussed earlier under bisection method. For the sake of carrying out a comparative study we will stick both to the same convergence criteria as before

i.e. $|f(\xi_n)| = |f(w)| < \epsilon = 10^{-6}$ (say) and to the example problems.

Example:

Solve $2x^3 - 2.5x - 5 = 0$ for the root in the interval [1,2] by Regula-Falsi method:

Solution: Since $f(1)\,f(2) = -33 < 0$, we go ahead in finding the root of given function f(x) in [1,2].

Set $a_0 = 1, \quad b_0 = 2$.

$$= \frac{f(2).1 - f(1).2}{f(2) - f(1)} = 1.47826087$$

$$= \frac{f(2).1 - f(1).2}{f(2) - f(1)} = 1.47826087$$

$$f(w) = f(1.47826087) = -2.23489761$$

$$\because f(a_0)f(w) = (-5.5) \times (-2.23489761) > 0$$

$$a_1 = w = 1.47826087 \quad ; \quad b_1 = b_0 = 2$$

set

$$\because \quad |f(w)| > \varepsilon = 10^{-6}$$

,

proceed with iteration.

Iteration details are provide below in a tabular form:

Regula Falsi Method

Iteration no.	a_n	b_n	ω_n	$f(\omega_n)$
0	1.0000000000	2.0000000000	1.4782608747	-2.2348976135
1	1.4782608747	2.0000000000	1.6198574305	-0.5488323569
2	1.6198574305	2.0000000000	1.6517157555	-0.1169833690
3	1.6517157555	2.0000000000	1.6583764553	-0.0241659321
4	1.6583764553	2.0000000000	1.6597468853	-0.0049594725
5	1.6597468853	2.0000000000	1.6600278616	-0.0010169938
6	1.6600278616	2.0000000000	1.6600854397	-0.0002089010
7	1.6600854397	2.0000000000	1.6600972414	-0.0000432589
8	1.6600972414	2.0000000000	1.6600997448	-0.0000081223

Note : One may note that Regula Falsi method has converged faster than the Bisection method.

POSSIBLE QUESTIONS

PART A (20 multiple choice questions)

Online Examinations

PART B (2 marks questions)

1. State roots of the equation.
2. Write a quadratic equation.
3. State Newton-Raphson method.
4. Give a formula of Regula-Falsi method.
5. What is iterative method?

PART C (6 marks questions)

1. Give a brief account on Newton-Raphson method.
2. Write a note on Regula-Falsi method
3. Explain bisection method.
4. Describe the two variables of Newton-Raphson method.
5. Explain the Iterative method of non-linear equation.
6. (i) Find by Newton-Raphson Method the real root of $3x - \cos x - 1 = 0$

   (ii) Find a real root of $x^x + x - 4 = 0$, by Newton-Raphson method, correct to six decimal places.
7. What is Method of False Position? Explain.
8. (i) Find a root of the equation $3x - \cos x - 1 = 0$ by Regula-Falsi method, correct to four significant figures.

(ii) Compute a root of $x \ln(x) = 1$ by Regula-Falsi Method, correct to three decimal places.

KARPAGAM ACADEMY OF HIGHER EDUCATION

Class: III B.Sc Chemistry Course Name: Applications of Computers in Chemistry

Course Code: 16CHU502B Unit: II Batch-2016-2019

S. No	Questions	Option A	Option B	Option C	Option D	Answer
1.	-------- Method is based on the repeated application of the intermediate value theorem.	Gauss Seidal	Bisection	Regula Falsi	Newton Raphson	Bisection
2.	The formula for Newton Raphson method is ----------.	$x_{n+1} = f(x_n) / f'(xn)$	$x_{n+1} = xn + f(x_n) / f'(xn)$	$x_{n+1} = xn - f(x_n) / f'(xn)$	$x_{n+1} = xn - f'(x_n) / f(xn)$	$x_{n+1} = xn - f(x_n) / f'(xn)$
3.	The order of convergence of Newton Raphson method is ----------	4	2	1	0	2
4.	Graeffe's root squaring method is useful to find ----------	complex roots	single roots	unequal roots	polynomial roots	polynomial roots
5.	The approximate value of the root of f(x) given by the bisection method is ----	$x_0 = a + b$	$x_0 = f(a) + f(b)$	$x_0 = (a + b)/ 2$	$x_0 = (f(a) + f(b))/2$	$x_0 = (a + b)/ 2$
6.	In Newton Raphson method, the error at any stage is proportional to the -------of the error in the previous stage.	cube	square	square root	equal	square
7.	The convergence of bisection method is -------.	linear	quadratic	slow	fast	slow
8.	The order of convergence of Regula falsi method may be assumed to ----------.	1	1.618	0	0.5	1.618
9.	---------- Method is also called method of tangents.	Gauss Seidal	Secant	Bisection	Newton Raphson	Newton Raphson
10.	If f (x) contains some functions like exponential, trigonometric, logarithmic etc., then f (x) is called ---------- equation.	Algebraic	transcendental	numerical	polynomial	transcendental
11.	A polynomial in x of degree n is called an algebraic equation of degree n if -----	f (x) = 0	f (x) = 1	f (x) <1	f (x) >1	f (x) = 0

12.	The method of false position is also known as ---------- method.	Gauss Seidal	Secant	Bisection	Regula falsi	Regula falsi
13.	The Newton Rapson method fails if ----------.	f ' (x) = 0	f (x) = 0	f (x) =1	f(x)¹0	f ' (x) = 0
14.	The bisection method is simple but ----------.	slowly divergent	fast convergent	slowly convergent	divergent	slowly convergent
15.	_____Method is also called as Bolzano method or interval having method.	Bisection	false position	Newton raphson	Horner's	Bisection
16.	The another name of Bisection method is _____	Bozano	Regula falsi	Newtons	Giraffes	Bozano
17.	The convergence of Bisection is Very_____	slow	fast	moderate	normal	slow
18.	In Regula-Falsi method, to reduce the number of iterations we start with _____ interval	Small	large	equal	none	Small
19.	The rate of convergence in Newton-Raphson method is of order _____	1	2	3	4	2
20.	Newton's method is useful when the graph of the function crosses the x-axis is nearly _____.	vertical	horizontal	close to zero	none	vertical
21.	If the initial approximation to the root is not given we can find any two values of x say a and bsuch that f (a) and f(b) are of _____signs.	opposite	same	positive	negative	opposite
22.	The Newton – Raphson method is also known as method of _____	secant	tangent	iteration	interpolation	tangent
23.	If the derivative of f(x) = 0, then _____ method should be used.	Newton – Raphson	Regula-Falsi	iteration	interpolation	Regula-Falsi
24.	The rate of convergence of Newton – Raphson method is _____	quadratic	cubic	4	5	quadratic
25.	If f (a) and f (b) are of opposite signs the actual root lies between _____	(a, b)	(0, a)	(0, b)	(0, 0)	(a, b)

26.	The convergence of root in Regula-Falsi method is slower than _____	Gauss – Elimination	Gauss – Jordan	Newton – Raphson	Power method	Newton – Raphson
27.	Regula-Falsi method is known as method of _____	secant	tangent	chords	elimination	chords
28.	_____method converges faster than Regula-Falsi method.	Newton – Raphson	Power method	elimination	interpolation	Newton – Raphson
29.	If f(x) is continuous in the interval (a, b) and if f (a) and f (b) are of opposite signs the equation f(x) = 0 has at least one _____ lying between a and b.	equation	function	root	polynomial	root
30.	x^2 + 3x – 3 = 0 is a polynomial of order	2	3	1	0	2
31.	Errors which are already present in the statement of the problem are called____ errors.	Inherent	Rounding	Truncation	Absolute	Inherent
32.	Rounding errors arise during_____	Solving	Algorithm	Truncation	Computation	Computation
33.	The other name for truncation error is _____ error.	Absolute	Rounding	Inherent	Algorithm	Algorithm
34.	Rounding errors arise from the process of _____ the numbers.	Truncating	Rounding off	Approximating	Solving	Rounding off
35.	Absolute error is denoted by_____	Ea	Er	Ep	Ex	Ea
36.	Truncation errors are caused by using _____ results.	Exact	True	Approximate	Real	Approximate
37.	Truncation errors are caused on replacing an infinite process by _____ one.	Approximate	True	Finite	Exact	Finite
38.	If a word length is 4 digits, then rounding off of 15.758 is	15.75	15.76	15.758	16	15.76
39.	The actual root of the equation lies between a and b when f (a) and f (b) are of ____ signs.	Opposite	same	negative	positive	Opposite
40.	To find the unknown values of y for some x which lies at the ------	beginning	end	center	outside	end

	--- of the table, we use Newton's Backward formula.					
41.	To find the unknown values of y for some x which lies at the ---------- of the table, we use Newton's Forward formula.	beginning	end	center	outside	beginning
42.	To find the unknown value of x for some y, which lies at the unequal intervals we use ---------- formula.	Newton's forward	Newton's backward	Lagrange	inverse interpolation	Lagrange
43.	If the values of the variable y are given, then the method of finding the unknown variable x is called ----------	Newton's forward	Newton's backward	interpolation	inverse interpolation	inverse interpolation
44.	In Newton's backward difference formula, the value of n is calculated by -------.	$n = (x–x_n) / h$	$n = (x_n–x) / h$	$n = (x–x_0) / h$	$n = (x_0–x) / h$	$n = (x–x_n) / h$
45.	In Newton's forward difference formula, the value x can be written as ----------.	$x_0–nh$	$x_n–nh$	$x_n + nh$	$x_0 + nh$	$x_0 + nh$
46.	In Newton's backward difference formula, the value x can be written as ---------	$x_0–nh$	$x_n–nh$	$x_n + nh$	$x_0 + nh$	$x_n + nh$
47.	---------- Interpolation formula can be used for equal and unequal intervals.	Newton's forward	Newton's backward	Lagrange	none	Lagrange
48.	The fourth differences of a polynomial of degree four are ---------.	zero	one	two	three	zero
49.	If the values $x_0 = 0$, $y_0 = 0$ and $h = 1$ are given for Newton's forward method, then the value of x is ----------.	0	1	n	X	n
50.	The second difference D^2y_0 is equal to	$y_2 + 2y_1 –y_0$	$y_2 - 2y_1 –y_0$	$y_2 - 2y_1 + y_0$	$y_2 + 2y_1 + y_0$	$y_2 - 2y_1 + y_0$
51.	In Newton's forward interpolation formula, the first two terms will give the -----	extrapolation	linear interpolation	parabolic interpolation	interpolation	linear interpolation
52.	In Newton's forward interpolation formula, the three	extrapolation	linear interpolation	parabolic interpolation	interpolation	parabolic interpolation

	terms will give the ------					
53.	The difference $D^3f(x)$ is called -------differences f(x).	first	fourth	second	third	third
54.	n th difference of a polynomial of n th degree are constant and all higher order difference are	constant	variable	zero	negative	zero
55.	In divided difference the value of any difference is ----- of the order of their argument	Independent	dependent	Inverse	direct	Independent
56.	Central difference equivalent to shift operator is	$E^{\frac{1}{2}} + E^{-\frac{1}{2}}$	$E^{\frac{1}{2}} - E^{-\frac{1}{2}}$	$E^{\frac{1}{2}} . E^{-\frac{1}{2}}$	E	$E^{\frac{1}{2}} . E^{-\frac{1}{2}}$
57.	The differences Dy are called ----differences f(x).	first	fourth	second	third	first
58.	The value (delta +1)is _____	E	h	h2	h4	E
59.	Relation between ,Δ ∇ and E	extrapolation	linear interpolation	parabolic interpolation	interpolation	linear interpolation
60.	Rounding errors arise during_____	Solving	Algorithm	Truncation	Computation	Computation

UNIT III

Syllabus

Differential calculus: Numerical differentiation.

Integral calculus: Numerical integration (Trapezoidal and Simpson's rule), probabilitydistributions and mean values.

Differentiation and integration are basic mathematical operations with a wide range of applications in many areas of science. It is therefore important to have good methods to compute and manipulate derivatives and integrals. You probably learnt the basic rules of differentiation and integration in school—symbolic methods suitable for pencil-and-paper calculations. These are important, and most derivatives can be computed this way. Integration however, is different, and most integrals cannot be determined with symbolic methods like the ones you learnt in school.

Another complication is the fact that in practical applications a function is only known at a few points. For example, we may measure the position of a car every minute via a GPS (Global Positioning System) unit, and we want to compute its speed. If the position is known as a continuous function of time, we can find the speed by differentiating this function. But when the position is only known at isolated times, this is not possible. The same applies to integrals.

The solution, both when it comes to integrals that cannot be determined by the usual methods, and functions that are only known at isolated points, is to use approximate methods of differentiation and integration. In our context, these are going to be numerical methods. We are going to present a number of methods for doing numerical integration and differentiation, but more importantly, we are going to present a general strategy for deriving such methods. In this way you will not only have a number of methods available to you, but you will also be able to develop new methods, tailored to special situations that you may encounter.

We use the same general strategy for deriving both numerical integration and numerical differentiation methods. The basic idea is to evaluate a function at a few points, find the polynomial that interpolates the function at these points, and use the derivative or integral of the polynomial as an approximation to the function. This technique also allows us to keep track of the so-called *truncation error*, the mathematical error committed by integrating or differentiating the polynomial instead of the function itself. However, when it comes to round off error, we have to treat differentiation and integration differently: Numerical integration is very *insensitive* to round-off errors, while numerical differentiation behaves in the opposite way; it is very *sensitive* to round-off errors.

Numerical Differentiation

This deals with numerical approximations of derivatives. The first question that comes up to mind is: why do we need to approximate derivatives at all? After all, we do know how to analytically differentiate every function. Nevertheless, there are several reasons as of why we still need to approximate derivatives:

• Even if there exists an underlying function that we need to differentiate, we might know its values only at a sampled data set without knowing the function itself.

• There are some cases where it may not be obvious that an underlying function exists and all that we have is a discrete data set. We may still be interested in studying changes in the data, which are related, of course, to derivatives.

• There are times in which exact formulas are available but they are very complicated to the point that an exact computation of the derivative requires a lot of function evaluations. It might be significantly simpler to approximate the derivative instead of computing its exact value.

• When approximating solutions to ordinary (or partial) differential equations, we typically represent the solution as a discrete approximation that is defined on a grid. Since we then have to

evaluate derivatives at the grid points, we need to be able to come up with methods for approximating the derivatives at these points, and again, this will typically be done using only values that are defined on a lattice. The underlying function itself (which in this cased is the solution of the equation) is unknown.

A simple method for numerical differentiation

We start by studying numerical differentiation. We first introduce the simplest method, derive its error, and its sensitivity to round-off errors.

Let us first make it clear what numerical differentiation is.

Let f be a given function that is only known at a number of isolated points. The problem of numerical differentiation is to compute an approximation to the derivative $f\,0$ of f by suitable combinations of the known values of f.

A typical example is that f is given by a computer program (more specifically a function, procedure or method, depending on your choice of programming language), and you can call the program with a floating-point argument x and receive back a floating-point approximation of f (x). The challenge is to compute an approximation to $f\,0(a)$ for some real number a when the only aid we have at our disposal is the program to compute values of f.

Since we are going to compute derivatives, we must be clear about they are defined. Recall that $f\,'(a)$ is defined by

$$f'(a) = \lim_{h \to 0} \frac{f(a+h) - f(a)}{h}. \qquad \text{----------} \quad (1)$$

In the following we will assume that this limit exists; i.e., that *f* is differentiable. From (1) we immediately have a natural approximation to *f'*(*a*); we simply pick a positive *h* and use the approximation

$$f'(a) \approx \frac{f(a+h)-f(a)}{h}.$$

---------- (2)

Note that this corresponds to approximating *f* by the straight line *p*1 that interpolates *f* at *a* and *a*-*h*, and then using *p'*1(*a*) as an approximation to *f* '(*a*).

The derivative of *f* at *a* can be approximated by

$$f'(a) \approx \frac{f(a+h)-f(a)}{h}.$$

---------- (3)

In a practical situation, the number *a* would be given, and we would have to locate the two nearest values *a*1 and *a*2 to the left and right of *a* such that *f* (*a*1) and *f* (*a*2) can be found. Then we would use the approximation

$$f'(a) \approx \frac{f(a_2)-f(a_1)}{a_2-a_1}.$$

Example 1

Let us test the approximation (11.2) for the function $f(x)$ = Sin (x) at a = 0.5 (using 64-bit floating-point numbers). In this case we have $f'(x) = \cos x$ so $f'0(a) = 0.87758256$. This makes it is easy to check the accuracy. We try with a few values of h and find

h	$\big(f(a+h)-f(a)\big)/h$	$E_1(f;a,h)$
10^{-1}	0.8521693479	2.5×10^{-2}
10^{-2}	0.8751708279	2.4×10^{-3}
10^{-3}	0.8773427029	2.4×10^{-4}
10^{-4}	0.8775585892	2.4×10^{-5}
10^{-5}	0.8775801647	2.4×10^{-6}
10^{-6}	0.8775823222	2.4×10^{-7}

where $E_1(f;a,h) = f(a) - \big(f(a+h)-f(a)\big)/h.$

In other words, the approximation seems to improve with decreasing h, as expected. More precisely, when h is reduced by a factor of 10, the error is reduced by the same factor.

The truncation error

Whenever we use approximations, it is important to try and keep track of the error, if at all possible. To analyse the error in numerical differentiation, Taylor polynomials with remainders are useful. To analyse the error in the approximation above, we do a Taylor expansion of $f(a+h)$. We have

$$f(a+h) = f(a) + hf'(a) + \frac{h^2}{2}f''(\xi_h),$$

where εh lies in the interval $(a,a+h)$. If we rearrange this formula, we obtain

$$f'(a)-\frac{f(a+h)-f(a)}{h}=-\frac{h}{2}f''(\xi_h). \qquad \text{----------(3)}$$

This is often referred to as the *truncation error* of the approximation, and is a reasonable error formula, but it would be nice to get rid of εh. We first take absolute values in (3).

$$\left|f'(a)-\frac{f(a+h)-f(a)}{h}\right|=\frac{h}{2}\left|f''(\xi_h)\right|.$$

Recall from the Extreme value theorem that if a function is continuous, then its maximum always exists on any closed and bounded interval. In our setting here, it is natural to let the closed and bounded interval be $[a,a+h]$. This leads to the following.

Suppose that f has continuous derivatives up to order two near a. If the derivative $f'(a)$ is approximated by

$$\frac{f(a+h)-f(a)}{h},$$

then the truncation error *is bounded by*

$$E(f;a,h)=\left|f'(a)-\frac{f(a+h)-f(a)}{h}\right|\le\frac{h}{2}\max_{x\in[a,a+h]}\left|f''(x)\right|.$$

So, the truncation error is approximately given by

$$\left|f'(a)-\frac{f(a+h)-f(a)}{h}\right|\approx\frac{h}{2}\left|f''(a)\right|.$$

The round-off error

So far, we have just considered the mathematical error committed when $f'(a)$ is approximated by $\lceil f(a+h)-f(a)\rceil/h$. But what about the round-off error? In fact, when we compute this approximation we have to perform the one critical operation $f(a+h)¡ - f(a)$—subtraction of two almost equal numbers. Let us continue the previous example and see what happens if we use smaller values of h.

Recall that we estimated the derivative of $f(x) = \sin x$ at $a = 0.5$ and that the correct value with ten digits is $f'(0.5) = 0.8775825619$. If we check values of h from 10^{-7} and smaller we find

h	$\lceil f(a+h)-f(a)\rceil/h$	$E(f;a,h)$
10^{-7}	0. 8775825372	2.5×10^{-8}
10^{-8}	0.8775825622	-2.9×10^{-10}
10^{-9}	0.8775825622	-2.9×10^{-10}
10^{-11}	0.8775813409	1.2×10^{-6}
10^{-14}	0.8770761895	5.1×10^{-4}
10^{-15}	0.8881784197	-1.1×10^{-2}
10^{-16}	1.110223025	-2.3×10^{-1}
10^{-17}	0.000000000	8.8×10^{-1}

This shows very clearly that something quite dramatic happens, and when we come to $h = 10^{-7}$, the derivative is computed as zero.

The maximum relative error when a real number is represented by a floating-point number is denoted by ε^*

Summary of the general strategy

Before we continue, let us sum up the derivaton and analysis of the numerical differentiation method in section (1), sincewewill use this over and over again. The first step was to derive the numerical method. In section 11.1 this was very simple since the method came straight out of the definition of the derivative. Just before observation (2) we indicated that the method can also be

de rived by approximating *f* by a polynomial *p* and using *p'*(*a*) as an approximation to *f* '(*a*). This is the general approach that we will use below.

Once the numerical method is known, we estimate the mathematical error in the approximation, *the truncation error*. This we do by performing Taylor expansions with remainders. For numerical differentiation methods which provide estimates of a derivative at a point *a*, we replace all function values at points other than *a* by Taylor polynomials with remainders. There may be a challenge to choose the degree of the Taylor polynomial.

Algorithm

To derive and analyse a numerical differentiation method, the following steps are necessary:

1. Derive the method using polynomial interpolation.
2. Estimate the truncation error using Taylor series with remainders.
3. Estimate the total error (truncation error + round-off error) by assuming all function evaluations are replaced by the nearest floating-point numbers.
4. Estimate the optimal value of *h*.

A four-pointmethod for differentiation

In a way, the two methods for numerical differentiation that we have considered so far are the same. If we use a step length of 2*h* in the first method, the approximation becomes

$$f'(a) \approx \frac{f(a+2h) - f(a)}{2h}.$$

The analysis of the symmetric method shows that the approximation is considerably better if we associate the approximation with the midpoint between *a* and *a* +*h*,

$$f'(a+h) \approx \frac{f(a+2h) - f(a)}{2h}.$$

At the point $a + h$ the approximation is proportional to $h2$ rather than h, and this makes a big difference as to how quickly the error goes to zero, as is evident.In this section we derive another method for which the truncation error is proportional to h^4. The computations below may seem overwhelming, and have in fact been done with the help of a computer to save time and reduce the risk of miscalculations. The method is included here just to illustrate that the principle for deriving both the method and the error terms is just the same as for the simple symmetric method in the previous section. To save space we have only included one highlight, of the approximation method and the total error.

Numerical approximation of the second derivative

We consider one more method for numerical approximation of derivatives, this time of the second derivative. The approach is the same: We approximate f by a polynomial and approximate the second derivative of f by the second derivative of the polynomial. As in the other cases, the error analysis is based on expansion in Taylor series.

A simple approximation of the first derivative is

$$f'(x) \approx \frac{f(x+h) - f(x)}{h}, \quad \text{----------(5.1)}$$

where we assume that h > 0. What do we mean when we say that the expression on the right-hand-side of the above equation is an approximation of the derivative? For linear functions the equation is actually an exact expression for the derivative. For almost all other functions, it is not the exact derivative.

Let's compute the approximation error. We write a Taylor expansion of f(x + h) about x, i.e

$$f(x+h) = f(x) + hf'(x) + \frac{h^2}{2}f''(\xi), \qquad \xi \in (x, x+h).$$

For such an expansion to be valid, we assume that f(x) has two continuous derivatives.
The Taylor expansion means that we can now replace the approximation equation with an exact formula of the form

$$f'(x) = \frac{f(x+h) - f(x)}{h} - \frac{h}{2}f''(\xi), \qquad \xi \in (x, x+h).$$

---------- **(5.3)**

Since this approximation of the derivative at x is based on the values of the function at x and x + h, the approximation (5.1) is called a **forward differencing or one-sided differencing.** The approximation of the derivative at x that is based on the values of the function at x − h and x, i.e.,

$$f'(x) \approx \frac{f(x) - f(x-h)}{h},$$

is called a **backward differencing** (which is obviously also a one-sided differencing formula).

The second term on the right-hand-side of (5.3) is the **error term**. Since the approximation (5.1) can be thought of as being obtained by truncating this term from the exact formula (5.3), this error is called the **truncation error.** The small parameter h denotes the distance between the two points x and x+h. As this distance tends to zero, i.e., $h \to 0$, the two points approach each other and we expect the approximation equation to improve. This is indeed the case if the truncation error goes to zero, which in turn is the case if $f''(\xi)$ is well defined in the interval (x, x+h). The

"speed" in which the error goes to zero as $h \to 0$ is called the **rate of convergence**. When the truncation error is of the order of O(h), we say that the method is a **first order method**. We refer to a methods as a **pth-order method** if the truncation error is of the order of $O(h^p)$. It is possible to write more accurate formulas than for the first derivative. For example, a more accurate approximation for the first derivative that is based on the values of the function at the points f(x−h) and f(x+h) is the centered differencing formula

$$f'(x) \approx \frac{f(x+h) - f(x-h)}{2h}.$$

General background on integration

Recall that if $f(x)$ is a function, then the integral of f from $x = a$ to $x = b$ is written

$$\int_a^b f(x)\,dx.$$

This integral gives the area under the graph of f, with the area under the positive part counting as positive area, and the area under the negative part of f counting as negative area.

Before we continue, we need to define a term which we will use repeatedly in our description of integration.

Let a and b be two real numbers with $a < b$. A partition *of $[a, b]$ is a finite sequence $\{x_i\}_{i=0}^n$ of increasing numbers in $[a, b]$ with $x_0 = a$ and $x_n = b$,*

$$a = x_0 < x_1 < x_2 \cdots < x_{n-1} < x_n = b.$$

The partition is said to be uniform *if there is a fixed number h, called the* step length, *such that $x_i - x_{i-1} = h = (b-a)/n$ for $i = 1, \ldots, n$.*

The traditional definition of the integral is based on a numerical approximation to the area. We pick a partition {*xi* } of [*a*,*b*], and in each subinterval [*xi*¡1,*xi*] we determine the maximum and minimum of *f* (for convenience we assume that these values exist),

$$m_i = \min_{x\in[x_{i-1},x_i]} f(x), \quad M_i = \max_{x\in[x_{i-1},x_i]} f(x),$$

for $i = 1, 2, \ldots, n$. We use these values to compute the two sums

$$\underline{I} = \sum_{i=1}^{n} m_i(x_i - x_{i-1}), \quad \overline{I} = \sum_{i=1}^{n} M_i(x_i - x_{i-1}).$$

To define the integral, we consider larger partitions and consider the limits of *I* and *I* as the instance between neighboring *xi* s goes to zero. If those limits are the same, we say that *f* is integrable, and the integral is given by this limit. More precisely,

$$I = \int_a^b f(x)\,dx = \sup \underline{I} = \inf \overline{I},$$

where the sup and inf are taken over all partitions of the interval [*a*,*b*]. This process is illustrated in figure 11.6 where we see how the piecewise constant approximations become better when the rectangles become narrower. The above definition can be used as a numerical method for computing approximations to the integral. We choose to work with either maxima or minima, select a partition of [*a*,*b*] as in figure 11.6, and add together the areas of the rectangles. The problem with this technique is that it can be both difficult and time consuming to determine the maxima or minima, even on a computer.

However, it can be shown that the integral has a very convenient property: If we choose a point *ti* in each interval [*xi*-1, *xi*], then the sum

$$\tilde{I} = \sum_{i=1}^{n} f(t_i)(x_i - x_{i-1})$$

will also converge to the integral when the distance between neighbouring *xi* s goes to zero. If we choose *ti* equal to *xi*-1 or *xi* , we have a simple numerical method for computing the integral. An even better choice is the more symmetric *ti* - (*xi* –*xi*-1)/2 which leads to the approximation

$$I \approx \sum_{i=1}^{n} f\big((x_i + x_{i-1})/2\big)(x_i - x_{i-1}).$$

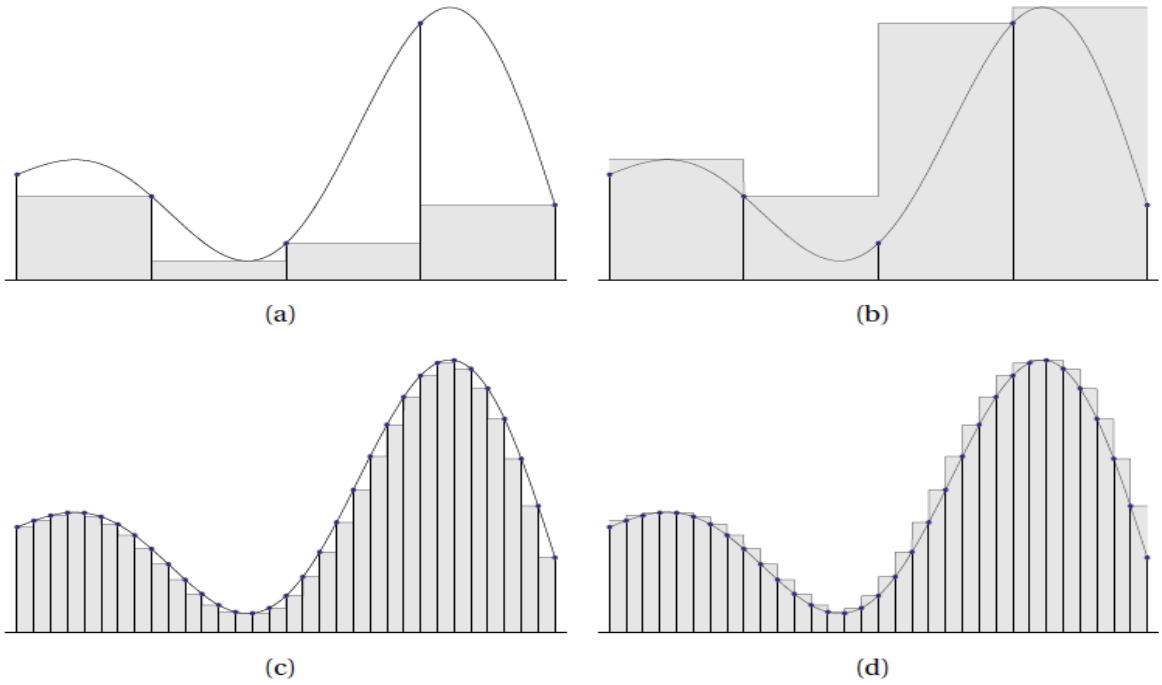


Figure 11.6. The definition of the integral via inscribed and circumsribed step functions.

This is the so-called *midpoint method* which we will study in the next section.

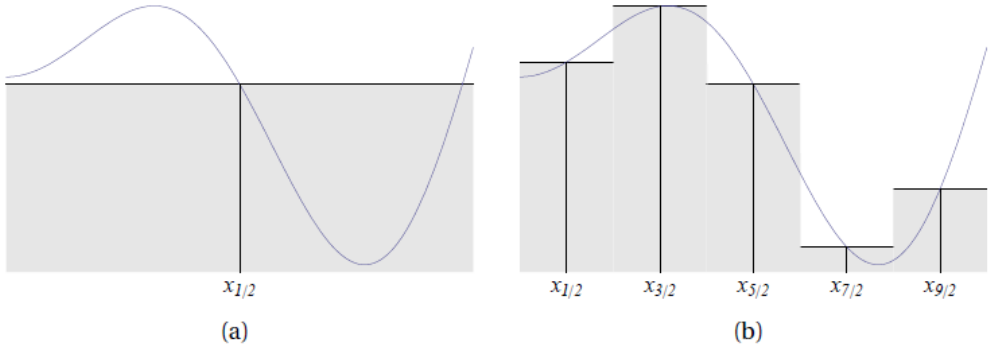


Figure 11.7. The midpoint rule with one subinterval (a) and five subintervals (b).

In general, we can derive numerical integration methods by splitting the interval [*a*,*b*] into small subintervals, approximate *f* by a polynomial on each subinterval, integrate this polynomial rather than *f* , and then add together the contributions from each subinterval. This is the strategy we will follow, and this works as long as *f* can be approximated well by polynomials on each subinterval.

The midpoint method for numerical integration

We have already introduced the midpoint rule (11.28) for numerical integration. In our standard framework for numerical methods based on polynomial approximation, we can consider this as using a constant approximation to the function *f* on each subinterval. Note that in the following we will always assume the partition to be uniform.

Let *f* a function which is integrable on the interval [*a*,*b*] and let {*xi* }*ni* = 0 be a uniform partition of [*a*,*b*]. In the midpoint rule, the integral of *f* is approximated by

$$\int_a^b f(x)\,dx \approx I_{mid}(h) = h\sum_{i=1}^{n} f(x_{i-1/2}), \qquad (11.29)$$

where

$$x_{i-1/2} = (x_i + x_{i-1})/2 = a + (i - 1/2)h.$$

This may seem like a strangely formulated algorithm, but all there is to it is to compute the sumon the right in (11.29).

Example: Let us try the midpoint method on an example. As usual, it is wise to test on an example where we know the answer, so we can easily check the quality of the method. We choose the integral

$$\int_0^1 \cos x \, dx = \sin 1 \approx 0.8414709848$$

where the exact answer is easy to compute by traditional, symbolic methods. To test the method, we split the interval into 2*k* subintervals, for *k* Æ 1, 2, . . . , 10, i.e., we halve the step length each time. The result is

h	$I_{mid}(h)$	Error
0.500000	0.85030065	-8.8×10^{-3}
0.250000	0.84366632	-2.2×10^{-3}
0.125000	0.84201907	-5.5×10^{-4}
0.062500	0.84160796	-1.4×10^{-4}
0.031250	0.84150523	-3.4×10^{-5}
0.015625	0.84147954	-8.6×10^{-6}
0.007813	0.84147312	-2.1×10^{-6}
0.003906	0.84147152	-5.3×10^{-7}
0.001953	0.84147112	-1.3×10^{-7}
0.000977	0.84147102	-3.3×10^{-8}

By error, we here mean,

$$\int_0^1 f(x) \, dx - I_{mid}(h).$$

Note that each time the step length is halved, the error seems to be reduced by a factor of 4.

Local error analysis

As usual, we should try and keep track of the error. We first focus on what happens on one subinterval. In other words we want to study the error.

$$\int_a^b f(x)\,dx - f\big(a_{1/2}\big)(b-a), \quad a_{1/2} = (a+b)/2.$$

Let f be a continuous function whose first two derivatives are continuous on the interval $[a.b]$. The the error in the midpoint method, with only one interval, is bounded by

$$\left|\int_a^b f(x)\,dx - f\big(a_{1/2}\big)(b-a)\right| \le \frac{7M}{24}(b-a)^3,$$

where $M = \max_{x\in[a,b]} |f''(x)|$ and $a_{1/2} = (a+b)/2$.

The importance of this lemma lies in the factor $(b\text{-}a)^3$. This means that if we reduce the size of the interval to half its width, the error in the midpoint method will be reduced by a factor of 8. Perhaps you feel completely lost in the work that led up to lemma 11.27. The wise way to read something like this is to first focus on the general idea that was used: Consider the error (11.30) and replace both f (x) and f $(a1/2)$ by its quadratic Taylor polynomials with remainders. If we do this, a number of terms cancel out and we are left with (11.33). At this point we use some standard techniques that give us the final inequality.

Once you have an overview of the derivation, you should check that the details are correct and make sure you understand each step.

Global error analysis

Above, we analysed the error on one subinterval. Now we want to see what happens when we add together the contributions frommany subintervals; it should not surprise us that this may affect the error.

We consider the general case where we have a partition that divides [*a*,*b*] into *n* subintervals, each of width *h*. On each subinterval we use the simple midpoint rule that we analysed.

Suppose that *f* and its first two derivatives are continuous on the interval [*a*,*b*], and that the integral of *f* on [*a*,*b*] is approximated by the midpoint rule with *n* subintervals of equal width,

$$I = \int_a^b f(x)\,dx \approx I_{mid} = \sum_{i=1}^{n} f(x_{i-1/2})\,h.$$

Then the error is bounded by

$$|I - I_{mid}| \le (b-a)\frac{7h^2}{24} \max_{x\in[a,b]} |f''(x)| \qquad (11.38)$$

where $x_{i-1/2} = a + (i - 1/2)\,h.$

This confirms the error behaviour that we saw in example 11.26: If *h* is reduced by a factor of 2, the error is reduced by a factor of 2^2 - 4.

One notable omission in our discussion of the midpoint method is round-off error, which was a major concern in our study of numerical differentiation. The good news is that round-off error is not usually a problem in numerical integration. The only situation where round-off may cause problems is when the value of the integral is 0. In such a situation we may potentially add many numbers that sum to 0, and this may lead to cancellation effects. However, this is so rare that we will not discuss it here.

You should be aware of the fact that the error estimate (11.38) is not the best possible in that the constant 7/24 can be reduced to 1/24, but then the derivation becomes much more complicated.

The trapezoid rule

The midpoint method is based on a very simple polynomial approximation to the function *f* to be integrated on each subinterval; we simply use a constant approximation by interpolating the function value at the middle point. We are now going to consider a natural alternative; we approximate *f* on each subinterval with the secant that interpolates *f* at both ends of the subinterval.

The situation is shown in figure 11.8a. The approximation to the integral is

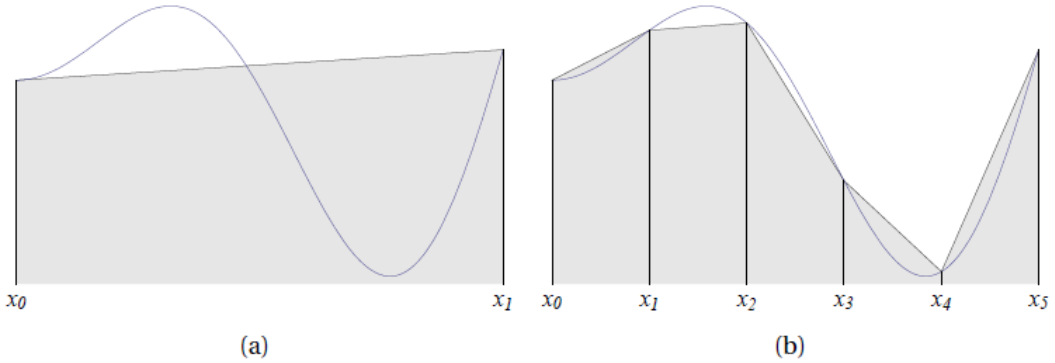


Figure 11.8. The trapezoid rule with one subinterval (a) and five subintervals (b).

the area of the trapezoidal figure under the secant so we have

$$\int_a^b f(x)\,dx \approx \frac{f(a)+f(b)}{2}(b-a). \qquad (11.39)$$

To get good accuracy, we will have to split [*a*,*b*] into subintervalswith a partition and use this approximation on each subinterval, see figure 11.8b. If we have a uniformpartition $\{x_i\}_{i=0}^{n}$ with step length *h*, we get the approximation

$$\int_a^b f(x)\,dx = \sum_{i=1}^{n} \int_{x_{i-1}}^{x_i} f(x)\,dx \approx \sum_{i=1}^{n} \frac{f(x_{i-1}) + f(x_i)}{2} h. \qquad (11.40)$$

We should always aim to make our computational methods as efficient as possible,and in this case an improvement is possible. Note that on the interval [*xi*-1,*xi*] we use the function values *f* (*xi*-1) and *f* (*xi*), and on the next interval we use the values *f* (*xi*) and *f* (*xi*=1). All function values, except the first and last, therefore occur twice in the sum on the right in (11.40). This means that if we implement this formula directly we do a lot of unnecessary work. From the explanation above the following observation follows.

(Trapezoid rule)**.** Suppose we have a function *f* defined on an interval [*a*,*b*] and a partition $\{x_i\}_{i=0}^{n}$ of [*a*.*b*]. If we approximate *f* by its secant on each subinterval and approximate the integral of *f* by the integral of the resulting piecewise linear approximation, we obtain the approximation

$$\int_a^b f(x)\,dx \approx h\left(\frac{f(a)+f(b)}{2} + \sum_{i=1}^{n-1} f(x_i)\right). \qquad (11.41)$$

In the formula (11.41) there are no redundant function evaluations.

Local error analysis

Let *f* be a continuous function whose first two derivatives are continuous on the interval [*a*.*b*]. The the error in the trapezoid rule, with only one line segment on [*a*,*b*], is bounded by

$$\left|\int_a^b f(x)\,dx - \frac{f(a)+f(b)}{2}(b-a)\right| \le \frac{5M}{12}(b-a)^3,$$

where $M = \max_{x\in[a,b]} |f''(x)|$.

This lemma is completely analogous to lemma 11.27 which describes the local error in the midpoint method. We particularly notice that even though the trapezoid rule uses two values of *f* , the error estimate is slightly larger than the estimate for the midpoint method. The most important feature is the exponent on (*b*-*a*), which tells us how quickly the error goes to 0 when the interval width is reduced, and from this point of view the two methods are the same. In other words, we have gained nothing by approximating *f* by linear functions instead of a constant. This does not mean that the trapezoid rule is bad, it rather means that the midpoint rule is unusually good.

Global error

We can find an expression for the global error in the trapezoid rule in exactly the same way as we did for the midpoint rule, so we skip the proof. We sum everything up in a theorem about the trapezoid rule.

Theorem : Suppose that *f* and its first two derivatives are continuous on the interval [*a*,*b*], and that the integral of *f* on [*a*,*b*] is approximated by the trapezoid rule with *n* subintervals of equal width *h*,

$$I = \int_a^b f(x)\,dx \approx I_{trap} = h\left(\frac{f(a)+f(b)}{2} + \sum_{i=1}^{n-1} f(x_i)\right).$$

Then the error is bounded by

$$\left|I - I_{trap}\right| \le (b-a)\frac{5h^2}{12}\max_{x\in[a,b]}\left|f''(x)\right|. \qquad (11.43)$$

As we mentioned when we commented on the midpoint rule, the error estimates that we obtain are not best possible in the sense that it is possible to derive better error estimates (using other techniques) with smaller constants. In the case of the trapezoid rule, the constant can be reduced from 5/12 to 1/12.

However, the fact remains that the trapezoid rule is a disappointing method compared to the midpoint rule.

Simpson's rule

The final method for numerical integration that we consider is *Simpson's rule*. This method is based on approximating *f* by a parabola on each subinterval, which makes the derivation a bit more involved. The error analysis is essentially the same as before, but because the expressions are more complicated, it pays off to plan the analysis better. You may therefore find the material in this section more challenging than the treatment of the other two methods, and should make sure that you have a good understanding of the error analysis for these methods before you start studying

Deriving Simpson's rule

As for the other methods, we derive Simpson's rule in the simplest case where we use one parabola on all of [*a*,*b*]. We find the polynomial *p*2 that interpolates *f* at *a*, *a*1/2 =(*a* + *b*)/2 and *b*, and approximate the integral of *f* by the integral of *p*2. We could find *p*2 via the Newton form, but in this case it is easier to use the Lagrange form. Another simplification is to first construct Simpson's rule in the case where *a* = -1, *a*1/2 = 0, and *b*= 1, and then use this to generalize the method.

The Lagrange form of the polyomial that interpolates *f* at -1, 0, 1, is given by

$$p_2(x) = f(-1)\frac{x(x-1)}{2} - f(0)(x+1)(x-1) + f(1)\frac{(x+1)x}{2},$$

and it is easy to check that the interpolation conditions hold.

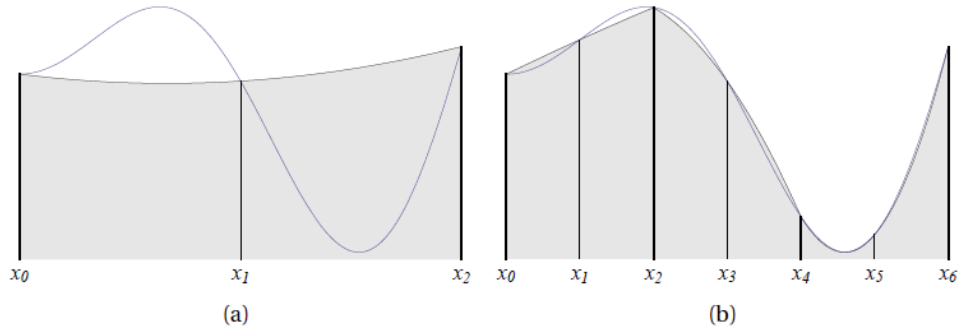


Figure 11.9. Simpson's rule with one subinterval (a) and three subintervals (b).

Observation: Let f be an integrable function on the interval $[a,b]$. If f is interpolated by a quadratic polynomial $p2$ at the points a, $(a+b)/2$ and b, then the integral of f can be approximated by the integral of $p2$,

$$\int_a^b f(x)\,dx \approx \int_a^b p_2(x)\,dx = \frac{b-a}{6}\left(f(a)+4f\left(\frac{a+b}{2}\right)+f(b)\right). \qquad (11.47)$$

We could just as well have derived this formula by doing the interpolation directly on the interval $[a,b]$, but then the algebra becomes quite messy.

Simpson's rule is exact for cubic polynomials

Lemma: If f is continuous and has continuous derivatives up to order 4 on the interval $[a,b]$, the error in Simpson's rule is bounded by

$$|E(f)| \le \frac{49}{2880}(b-a)^5 \max_{x\in[a,b]} \left|f^{(iv)}(x)\right|.$$

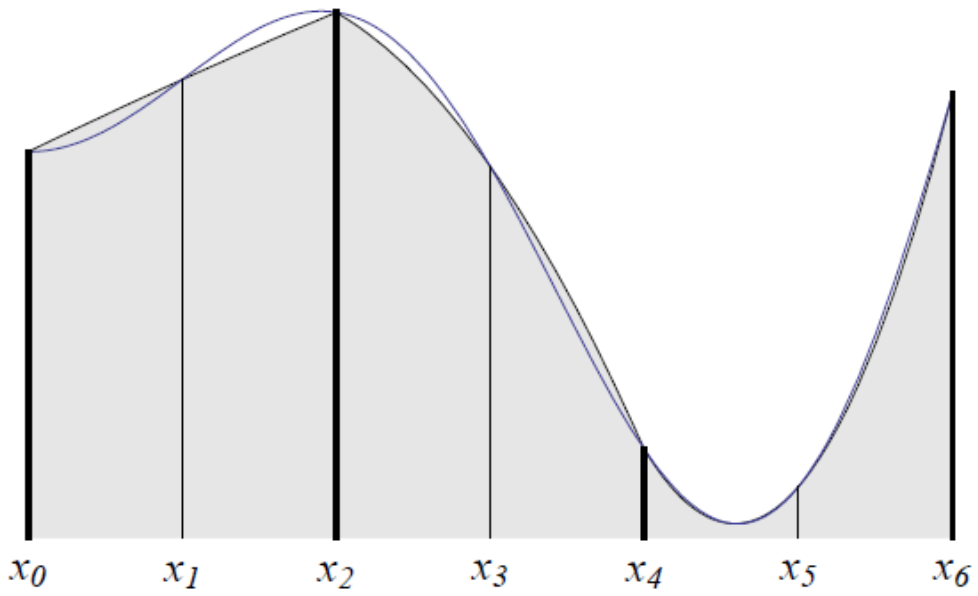


Figure 11.10. Simpson's rule with three subintervals.

We note that the error in Simpson's rule depends on $(b - a)^5$, while the error in the midpoint rule and trapezoid rule depend on $(b - a)^3$. This means that the error in Simpson's rule goes to zero much more quickly than for the other two methods when the width of the interval $[a,b]$ is reduced. More precisely, a reduction of h by a factor of 2 will reduce the error by a factor of 32. As for the other two methods the constant 49/2880 is not best possible; it can be reduced to 1/2880 by using other techniques.

Composite Simpson's rule

Simpson's rule is used just like the other numerical integration techniques we have studied: The interval over which f is to be integrated is split into subintervals and Simpson's rule is applied on neighbouring pairs of intervals, see figure 11.10. In other words, each parabola is defined over *two* subintervals which means that the total number of subintervals must be even and the number of given values of f must be odd.

In this sum we observe that the right endpoint of one subinterval becomes the left endpoint of the following subinterval to the right. Therefore, if this is implemented directly, the function values

at the points with an even subscript will be evaluated twice, except for the extreme endpoints *a* and *b* which only occur once in the sum. We can therefore rewrite the sum in a way that avoids these redundant evaluations.

Observation : Suppose *f* is a function defined on the interval [*a*,*b*], and let $\{x_i\}_{i=0}^{2n}$ be a uniform partition of [*a*,*b*] with step length *h*. The composite Simpson's rule approximates the integral of *f* by

$$\int_a^b f(x)\,dx \approx \frac{h}{3}\left(f(a)+f(b)+2\sum_{i=1}^{n-1} f(x_{2i})+4\sum_{i=1}^{n} f(x_{2i-1})\right).$$

With the midpoint rule, we computed a sequence of approximations to the integral by successively halving the width of the subintervals. The same is often done with Simpson's rule, but then care should be taken to avoid unnecessary function evaluations since all the function values computed at one step will also be used at the next step.

The error in the composite Simpson's rule

The approach we used to deduce the global error for themidpoint rule, can also be used for Simpson's rule, see theorem 11.28. The following theorem sums this up.

Theorem: Suppose that *f* and its first four derivatives are continuous on the interval [*a*,*b*], and that the integral of *f* on [*a*,*b*] is approximated by Simpson's rule with 2*n* subintervals of equal width *h*. Then the error is bounded by

$$\left|E(f)\right| \le (b-a)\frac{49h^4}{2880}\max_{x\in[a,b]}\left|f^{(iv)}(x)\right|.$$

Procedure 11.38. *The following is a general procedure for deriving numerical methods for differentiation and integration:*

1. *Interpolate the function f by a polynomial p at suitable points.*
2. *Approximate the derivative or integral of f by the derivative or integral of p. This makes it possible to express the approximation in terms of function values of f.*
3. *Derive an estimate for the error by expanding the function values (other than the one at a) in Taylor series with remainders.*

4D. *For numerical differentiation, derive an estimate of the round-off error by assuming that the relative errors in the function values are bounded by ϵ^*. By minimising the total error, an optimal step length h can be determined.*

4I. *For numerical integration, the global error can easily be derived from the local error using the technique leading up to theorem 11.28.*

Exercises

11.1 **a)** Write a program that implements the numerical differentiation method

$$f'(a) \approx \frac{f(a+h) - f(a-h)}{2h},$$

and test the method on the function $f(x) = e^x$ at $a = 1$.

b) Determine the optimal value of h given in section 11.3.4 which minimises the total error. Use $\epsilon^* = 7 \times 10^{-17}$.

c) Use your program to determine the optimal value h of experimentally.

d) Use the optimal value of h that you found in (c) to determine a better value for ϵ^* in this specific example.

11.2 Repeat exercise 1, but compute the second derivative using the approximation

$$f''(a) \approx \frac{f(a+h) - 2f(a) + f(a-h)}{h^2}.$$

In (b) you should use the value of h given in observation 11.23.

11.3 **a)** Suppose that we want to derive a method for approximating the derivative of f at a which has the form

$$f'(a) \approx c_1 f(a-h) + c_2 f(a+h), \quad c_1, c_2 \in \mathbb{R}.$$

We want the method to be exact when $f(x) = 1$ and $f(x) = x$. Use these conditions to determine c_1 and c_2.

b) Show that the method in (a) is exact for all polynomials of degree 1, and compare it to the methods we have discussed in this chapter.

c) Use the procedure in (a) and (b) to derive a method for approximating the second derivative of f,

$$f''(a) \approx c_1 f(a-h) + c_2 f(a) + c_3 f(a+h), \quad c_1, c_2, c_3 \in \mathbb{R},$$

by requiring that the method should be exact when $f(x) = 1$, x and x^2.

d) Show that the method in (c) is exact for all quadratic polynomials.

11.4 **a)** Write a program that implements the midpoint method as in algorithm 11.29 and test it on the integral

$$\int_0^1 e^x \, dx = e - 1.$$

b) Determine a value of h that guarantees that the absolute error is smaller than 10^{-10}. Run your program and check what the actual error is for this value of h. (You may have to adjust algorithm 11.29 slightly and print the absolute error.)

11.5 Repeat exercise 4, but use Simpson's rule instead of the midpoint method.

11.6 It may sometimes be difficult to judge how many terms to include in the Taylor series used in the analysis of numerical methods. In this exercise we are going to see how this can be done. We use the numerical approximation

$$f'(a) \approx \frac{f(a+h) - f(a-h)}{2h}$$

in section 11.3 for our experiments.

a) Do the same derivation as section 11.3.2, but include only two terms in the Taylor series (plus remainder). What happens?

b) Do the same derivation as section 11.3.2, but include four terms in the Taylor series (plus remainder). What happens now?

11.7 When h is halved in the trapezoid method, some of the function values used with step length $h/2$ are the same as those used for step length h. Derive a formula for the trapezoid

POSSIBLE QUESTIONS

PART A (20 multiple choice questions)

Online Examinations

PART B (2 marks questions)

1. Define numerical differentiation.
2. Write two types of situations of numerical integration.
3. What is meant by Trapezoidal rule?
4. What are probability distributions?
5. State numerical integration.

PART C (6 marks questions)

1. Describe the Trapezoidal rule for performing numerical integration.
2. Explain Simpson's rule for performing numerical integration.
3. Give a brief account on numerical differentiation.
4. Describe numerical differentiation by Newtons forwarded law and Stirling's law.
5. Derive the following (i) Stirling's law (ii) Bessel's law
6. Write note on probability distributions.
7. Derive a general Quadrature formula for equidistant ordinates.
8. Describe numerical differentiation by Newton's forwarded and back warded law

KARPAGAM ACADEMY OF HIGHER EDUCATION

Class: III B.Sc Chemistry Course Name: Applications of Computers in Chemistry

Course Code: 16CHU502B Unit: III Batch-2016-2019

S. No	Questions	Option A	Option B	Option C	Option D	Answer
1.	If the given integral is approximated by the sum of 'n' trapezoids, then the rule is called as ----------.	Newton's method	Trapezoidal rule	simpson's rule	power	Trapezoidal rule
2.	The order of error in Trapezoidal rule is ----------.	h	h^3	h^2	h^4	h^2
3.	The general quadratic formula for equidistant ordinates is _____	raphson	Newton-cote's	interpolation	divide difference	Newton-cote's
4.	h/2[(sum of the first and last ordinates)+2(sum of the remaining ordinates)] is_____	simphson's 3/8	simphson's 1/3	trapezoidal	taylor series	trapezoidal
5.	Use trapezoidal rule for y(x) _____	linear	second degree	third degree	degree n	linear
6.	Simpson's rule is exact for a ---------- even though it was derived for a Quadratic.	cubic	less than cubic	linear	quadratic	linear
7.	What is the order of the error in Simpson's formula?	Four	three	two	one	Four
8.	Simpson's 1/3 is findind y(x) upto _____	linear	second degree	degree n	third degree	second degree
9.	In simpson's 1/3, the number of intervels must be _____	any integer	odd	even	prime	even
10.	In simpson's 1/3, the number of ordinates must be _____	any integer	odd	prime	even	odd
11.	Simpson's one-third rule on numerical integration is called a --------- formula.	closed	open	semi closed	semi opened	closed
12.	In simphson's 3/8 rule, we	degree n	linear	second degree	third degree	third degree

	calculate the polynomial of degree _____					
13.	The number of interval is multiple of three the use_____	simpson's 1/3	trapezoidal	simpson's 3/8	taylor series	simpson's 3/8
14.	The number of interval is multiple of six _____	simpson's 1/3	simphson's 3/8	weddle	trapezoidal	weddle
15.	The error in Simpson's 1/3 is ----------.	h	h^3	h^2	h^4	h^4
16.	Modulus of E is _____	<M(b-a)h4/180	0	>M(b-a)h4/180	M(b-a)h4/180	<M(b-a)h4/180
17.	The order of error is h^2 for_____	lagrange's	trapezoidal	weddle	simpson's 1/3	trapezoidal
18.	h^4 is the error of _____	simphson's 3/8	simphson's 1/3	trapezoidal	taylor series	simphson's 1/3
19.	The value of integral e^x is evaluated from 0 to 0.4 by the following formula. Which method will give the least error ?	Trapezoidal rule with h = 0.2	Trapezoidal rule with h = 0.1	Simpson's 1/3 rule with h = 0.1.	weddle	Simpson's 1/3 rule with h = 0.1.
20.	Using Simpson's rule the area in square meters included between the chain line, irregular boundary and the first and the last offset will be _____	7.33.28 sq-m	744.18 sq-m	880.48 sq-m.	820.38 sq-m	820.38 sq-m
21.	By putting n = 1 in Newton cote's formula we get ---------- rule.	Simpson's 1/3 rule	Simpson's 3/8 rule	Trapezoidal rule	Simpson's rule	Trapezoidal rule
22.	I = (3h / 8) { $(y_0 + y_n) + 3 (y_1 + y_2 + y_4 + y_5 + \ldots..) + 2(y_3 + y_6 + y_9 + \ldots..)$ } is known as ----------.	Simpson's 1/3 rule	Simpson's 3/8 rule	Trapezoidal rule	Simpson's rule	Simpson's 3/8 rule
23.	I = (h / 3) { $(y_0 + y_n) + 2 (y_2 + y_4 + y_6 + y_8 + \ldots..) + 4(y_1 + y_3 + y_5 + \ldots..)$ } is known as ----------	Simpson's 1/3 rule	Simpson's 3/8 rule	Trapezoidal rule	Simpson's rule	Simpson's 1/3 rule
24.	The differentiation of log x is _____	1/x	e(x)	sinx	cosx	1/x
25.	$\int f(x)\,dx$ of (a, b) is _____	F(a)	F(b)	F(a+b)	F(b)-F(a)	F(b)-F(a)
26.	h/3[(sum of first and last ordinates)+2(sum of even ordinates)+4(sum of odd ordinates)] is the formula for	trapezoidal	simphson's 1/3	simphson's 3/8	taylor series	simphson's 1/3

	_____					
27.	In simpson 1/3 rule, the integral value is h/3[y0+4(y1)+y2] _____	for n=1	for n=2	for n=3	for n=4	for n=2
28.	Differentiation of sinx is _____	cosx	tanx	sinx	logx	cosx
29.	Integration of cosx _____	cosx	tanx	sinx	logx	sinx
30.	If y(x) is linear then use _____	simphson's 3/8	simphson's 1/3	trapezoidal	taylor series	trapezoidal
31.	The differentiation of secx is _____	secx tanx	cotx	cosecx	tanx	secx tanx
32.	The notation h is _____	differece of ordinates	sum of ordinates	number of ordinates	product of ordinates	differece of ordinates
33.	While evaluating the definite integral by Trapezoidal rule, the accuracy can be increased by taking----------	Large number of sub-intervals	even number of sub-intervals	multipleof6	has multiple of 3	Large number of sub-intervals
34.	Numerical integration when applied to a function of a single variable, it is known as--------	maxima	minima	quadrature	quadrant	quadrature
35.	The numerical backward differentiation of y w.r.t. x once is ----------.	$f'(x) = (1/h)*(Dy_0 + (2r-1)/2 * D^2y_0 + (3r^2-6r+2)/6 * D^3y_0 + \ldots\ldots\ldots)$	$y = y_n + n \tilde{N}y_n + \{n(n+1)/2!\} \tilde{N}^2y_n + \{n(n+1)(n+2)/3!\} \tilde{N}^3y_n + \ldots\ldots\ldots)$	$f'(x) = (1/h)*(Dy_n + (2r+1)/2 * D^2y_n + (3r^2+6r+2)/6 * D^3y_n + \ldots\ldots\ldots)$	$f'(x) = (1/h)*(\tilde{N}y_n + (2r+1)/2 * \tilde{N}^2y_n + (3r^2+6r+2)/6 * \tilde{N}^3y_n + \ldots\ldots\ldots)$	$f'(x) = (1/h)*(\tilde{N}y_n + (2r+1)/2 * \tilde{N}^2y_n + (3r^2+6r+2)/6 * \tilde{N}^3y_n + \ldots\ldots\ldots)$
36.	The second derivative of the Newton's forward differentiation is ----------.	$y'' = (1/h^2)*\{D^2y_0 - D^3y_0 + (11/12) D^4y_0 \ldots\ldots\ldots\}$	$y'' = (1/h^2)*\{D^2y_0 + D^3y_0 + (11/12) D^4y_0 \ldots\ldots\ldots\}$	$y'' = (1/h)*\{D^2y_0 + D^3y_0 + (11/12) D^4y_0 \ldots\ldots\ldots\}$	$y'' = (1/h)*\{D^2y_0 - D^3y_0 + (11/12) D^4y_0 \ldots\ldots\ldots\}$	$y'' = (1/h^2)*\{D^2y_0 - D^3y_0 + (11/12) D^4y_0 \ldots\ldots\ldots\}$
37.	The second derivative of the Newton's backward differentiation is ----------.	$y'' = (1/h^2)*\{D^2y_0 + D^3y_0 + (11/12) D^4y_0 \ldots\ldots\ldots\}$	$y'' = (1/h^2)*\{D^2y_0 - D^3y_0 + (11/12) D^4y_0 \ldots\ldots\ldots\}$	$y'' = (1/h)*\{D^2y_0 + D^3y_0 + (11/12) D^4y_0 \ldots\ldots\ldots\}$	$y'' = (1/h)*\{D^2y_0 - D^3y_0 + (11/12) D^4y_0 \ldots\ldots\ldots..\}$	$y'' = (1/h^2)*\{D^2y_0 + D^3y_0 + (11/12) D^4y_0 \ldots\ldots\ldots\}$
38.	The order of error in Trapezoidal rule is ----------.	h	h^3	h^2	h^4	h^2
39.	The order of error in Simpson's rule is ----------.	h	h^3	h^2	h^4	h^4
40.	Numerical evaluation of a definite integral is called --------.	Integration	Differentiation	Interpolation	Triangularization	Integration
41.	Simpson's ⅜ rule can be applied only if the number of sub	Equal	even	multiple of three	unequal	multiple of three

	interval is in ------.					
42.	By putting n = 2 in Newton cote's formula we get ---------- rule.	Simpson's 1/3	Simpson's ⅜	Trapezoidal	Romberg	Simpson's 1/3
43.	The Newton Cote's formula is also known as ---------- formula.	Simpson's 1/3	Simpson's 3/8	Trapezoidal	quadrature	quadrature
44.	By putting n = 3 in Newton cote's formula we get ---------- rule.	Simpson's 1/3	Simpson's ⅜	Trapezoidal	Romberg	Simpson's ⅜
45.	By putting n = 1 in Newton cote's formula we get ---------- rule.	Simpson's 1/3	Simpson's ⅜	Trapezoidal	newton's	Trapezoidal
46.	The systematic improvement of Richardon's method is called---------- method	Simpson's 1/3	Simpson's ⅜	Trapezoidal	Romberg	Romberg
47.	Simpson's 1/3 rule can be applied only when the number of interval is ------.	Equal	even	multiple of three	unequal	even
48.	Simpson's rule is exact for a ---------- even though it was derived for a Quadratic.	cubic	less than cubic	linear	quadratic	linear
49.	The accuracy of the result using the Trapezoidal rule can be improved by --------	Increasing the interval h	Decreasing the interval h	Increasing the number of iterations	Altering the given function	Increasing the number of iterations
50.	A particular case of Runge Kutta method of second order is ----------.	Milne's method	Picard's method	Modified Euler method	Runge's method	Modified Euler method
51.	Runge Kutta of first order is nothing but the ----------.	modified Euler method	Euler method	Taylor series	none of these	Euler method
52.	In Runge Kutta second and fourth order methods, the values of k1 and k2 are ----	same	differ	always positive	always negative	same
53.	The formula of Dy in fourth order Runge Kutta method is given by ----------.	$1/6 * (k_1 + 2k_2 + 3k_3 + 4k_4)$	$1/6 * (k_1 + k_2 + k_3 + k_4)$	$1/6 * (k_1 + 2k_2 + 2k_3 + k_4)$	$1/6 * (k_1 + 2k_2 + 2k_3 + k_4)$	$1/6 * (k_1 + 2k_2 + 2k_3 + k_4)$
54.	In second order Runge Kutta method the value of k2 is calculated by ----------.	$h f(x + h/2 , y + k_1/2)$	$h f(x - h/2 , y - k_1/2)$	$h f(x , y)$	$h f(0,0)$	$h f(x + h/2 , y + k_1/2)$

55.	_____values are calculated in Runge Kutta fourth order method.	k_1, k_2, k_3, k_4 and Dy	k_1, k_2 and Dy	k_1, k_2, k_3 and Dy	none of these	k_1, k_2, k_3, k_4 and Dy
56.	The use of Runge kutta method gives ---------- to the solutions of the differential equation than Taylor's series method.	Slow convergence	quick convergence	oscillation	divergence	quick convergence
57.	In Runge – kutta method the value x is taken as ----------.	$x = x_0 + h$	$x_0 = x + h$	$h = x_0 + x$	$h = x_0 - x$	$x = x_0 + h$
58.	In Runge – kutta method the value y is taken as ----------.	$y = y_0 + h$	$y_0 = x_0 + h$	$y = y_0 – Dy$	$y = y_0 + Dy$	$y = y_0 + Dy$
59.	In fourth order Runge Kutta method the value of k3 is calculated by ----------.	$h f(x - h/2 , y - k_2/2)$	$h f(x + h/2 , y + k_2/2)$	$h f(x , y)$	$h f(x – h/2 , y – k_1/2)$	$h f(x + h/2 , y + k_2/2)$
60.	In fourth order Runge Kutta method the value of k4 is calculated by ----------.	$h f(x + h/2 , y + k_1/2)$	$h f(x + h/2 , y + k_2/2)$	$h f(x + h , y + k_3)$	$h f(x – h , y – k_3)$	$h f(x + h , y + k_3)$

UNIT-IV

Syllabus

Simultaneous equations: Matrix manipulation: addition, multiplication. Gauss-Siedal method. *Interpolation, extrapolation and curve fitting:* Handling of experimental data.

Matrix operations:

Mathcad is designed to be a tool for quick and easy manipulation of matrix forms of data. We've seen the matrix before in Lecture 1 as a 2-D array. That is, many pieces of information are stored under a single name. Different pieces of information are then retrieved by pointing to different parts of the matrix by row and column. Here we will learn some basic matrix operations: Adding and Subtracting, Transpose, Multiplication.

Adding matrices

Add two matrices together is just the addition of each of their respective elements. If A and B are both matrices of the same dimensions (size), then

```
C := A + B
```

produces C, where the i^{th} row and j^{th} column are just the addition of the elements (numbers) in the i^{th} and j^{th} column of A and B

Given: $A = \begin{bmatrix} 1 & 3 & 5 \\ 7 & 9 & 11 \end{bmatrix}$, and $B = \begin{bmatrix} 2 & 4 & 6 \\ 8 & 10 & 12 \end{bmatrix}$

so that the addition is : $C := A + B = \begin{bmatrix} 3 & 7 & 11 \\ 15 & 19 & 23 \end{bmatrix}$

The Mathcad commands to perform these matrix assignments and the addition are:

```
A := Ctrl-M (choose 2 x 3) 1 3 5 7 9 11
B := Ctrl-M (choose 2 x 3) 2 4 6 8 10 12
C := A + B        C =
```

Rule: A, B, and C must all have the same dimensions

Transpose

Transposing a matrix means swapping rows and columns of a matrix. No matrix dimension restrictions

Some examples:

1-D $A = \begin{bmatrix} 5 & 2 & 9 \end{bmatrix}$, $A^T = \begin{bmatrix} 5 \\ 2 \\ 9 \end{bmatrix}$ 1x3 becomes ==> 3x1

2-D $B = \begin{bmatrix} 8.1 & -4.5 & -7.6 \\ 3.2 & 3.1 & 3.9 \end{bmatrix}$, $B^T = \begin{bmatrix} 8.1 & 3.2 \\ -4.5 & 3.1 \\ -7.6 & 3.9 \end{bmatrix}$ 2x3 becomes ==> 3x2

In general

$$B(i, j) = B^T(j, i)$$

In Mathcad, **The transpose is can be keystroked by Ctrl - 1 (the number one)**

$B = \begin{bmatrix} 5 & 3 & 6 & 2 \\ 9 & 8 & 4 & 7 \end{bmatrix}$ B Ctrl-1 = $ans = \begin{bmatrix} 5 & 9 \\ 3 & 8 \\ 6 & 4 \\ 2 & 7 \end{bmatrix}$

Multiplication

Multiplication of matrices is not as simple as addition or subtraction. It is not an element by element multiplication as you might suspect it would be. Rather, matrix multiplication is the result of the dot products of rows in one matrix with columns of another. Consider:

C := A * B

matrix multiplication gives the ith row and kth column spot in C as the scalar results of the dot product of the ith row in A with the kth column in B. In equation form this looks like:

$$C_{i,k} = \sum_{j=1}^{\#\text{ of columns in A}} A_{i,j} * B_{j,k}$$

Let's break this down in a step-by-step example:

Step 1: Dot Product (a 1-row matrix times a 1-column matrix)
The Dot product is the **scalar** result of multiplying one row by one column

$$\underset{1x3}{\begin{bmatrix}2 & 5 & 3\end{bmatrix}} * \underset{3x1}{\begin{bmatrix}6\\8\\7\end{bmatrix}} = 2*6 + 5*8 + 3*7 = 73_{1x1}$$ DOT PRODUCT OF ROW AND COLUMN

Rule:

1) # of elements in the row and column must be the same

2) must be a row times a column, not a column times a row

Step 2: general matrix multiplication is taking a series of dot products

each row in pre-matrix by each column in post-matrix

$$\underset{2x3}{\begin{bmatrix}1 & 4 & 2\\9 & 3 & 7\end{bmatrix}} * \underset{3x2}{\begin{bmatrix}5 & 6\\8 & 12\\10 & 11\end{bmatrix}} = \begin{bmatrix}1*5+4*8+2*10 & 1*6+4*12+2*11\\9*5+3*8+7*10 & 9*6+3*12+7*11\end{bmatrix} = \underset{2x2}{\begin{bmatrix}57 & 76\\139 & 167\end{bmatrix}}$$

C(i,k) is the result of the dot product of row i in A with column k in B

Matrix Multiplication Rules:

1) The # of columns in the pre-matrix must equal # of rows in post-matrix inner matrix dimensions must agree

2) The result of the multiplication will have the outer dimensions # rows in pre-matrix by # columns in post-matrix

For this example, apply rules

C := A * B

A is nra x nca (# rows in a by # columns in a)

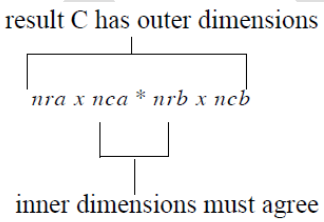
B is nrb x ncb

Rule 1 says:

nca = nrb or else we can't multiply (can't take dot products with different number of terms in row and column)

Rule 2 says:

C will be of size nra x ncb



How to perform matrix multiplication in Mathcad???
Easy

```
A := [4 5; 2 1]
B := [9 1; 6 12]
C := A*B
```

Note: Now that we know how to enter a matrix into Mathcad, I'll use a shortcut notation for these notes as applied above, where:

A := [4 5; 2 1]

means a 2 x 2 array where the ; indicates the start of the next row. The ; is not actually used in Mathcad for this purpose, its just a shorthand notation that I'll use for these notes.

D := [12 34 56; 45 89 9] is a 2 x 3 matrix using this notation

Note: If inner matrix dimensions don't match, Mathcad can't perform the operation since it violates the rules of matrix multiplication, and you'll get an error that says:

"the number of rows and or columns in these arrays do not match"

Example: Let's try to multiply a 2x3 by another 2x3 (rules say we can't do this)

```
A := [3 4 1 ; 0 4 9];
B := [2 9 5 ; 9 4 5];
C := A * B
```

Mathcad will tell you:
"the number of rows and or columns in these arrays do not match" and won't provide an answer
Since the # of columns in A was not equal to # of rows in B, we can't multiply A * B

IMPORTANT: Another example: Say we create a 1-D vector x with the following:

x := [2 4 9 5 2];

Now say we want to square each number in x. It would seem natural to do this:

y := x^2

But Mathcad tells us:
"This Matrix must be square. It should have the same number of rows as columns"
Note that y : = x^2 is the same as saying y := x*x
Mathcad by default will always interpret any multiplication as a standard dot product type matrix multiplication, thus we can't take a dot product of two row vectors, since rules of matrix multiplication are violated in this case. The exception to this default assumption in Mathcad is if

the vector is a column instead of a row. In that case, Mathcad will assume you want to square each element in the vector rather that apply standard matrix multiplication.

If we just want to square the numbers in x, we can do this:

y = x Ctrl-1 shift ^2

This first transposes the row vector into a column vector, then squares the elements in the vector

Try this out

$$a := (2 \quad 5 \quad 4) \qquad \left(a^T\right)^2 = \begin{pmatrix} 4 \\ 25 \\ 16 \end{pmatrix}$$

Practice matrix operations on the following examples.

List the size of the resulting matrix first. then perform the operations by hands. Use Mathcad to confirm each of your answers.

$$\begin{bmatrix}9 & 12 & 5\end{bmatrix} * \begin{bmatrix}2\\8\\3\end{bmatrix} = \begin{bmatrix}\quad\end{bmatrix}$$

$$\begin{bmatrix}9 & 7\\3 & 6\\7 & 1\end{bmatrix}\begin{bmatrix}2 & 4 & 6\\8 & 10 & 12\end{bmatrix} = \begin{bmatrix}\quad\end{bmatrix}$$

$$\begin{bmatrix}4\\9\\7\end{bmatrix} * \begin{bmatrix}2 & 1 & 8\end{bmatrix} = \begin{bmatrix}\quad\end{bmatrix}$$

$$\begin{bmatrix}4 & 3\\2 & 1\end{bmatrix} * \begin{bmatrix}8 & 7\\9 & 6\end{bmatrix} = \begin{bmatrix}\quad\end{bmatrix}$$

$$\begin{bmatrix}4 & 9 & 8\end{bmatrix}^T * \begin{bmatrix}2 & 7 & 1\end{bmatrix} = \begin{bmatrix}\quad\end{bmatrix}$$

$$\begin{bmatrix}1 & 4 & 9\\8 & 6 & 4\end{bmatrix} * \begin{bmatrix}3 & 0\\9 & 3\\4 & 7\end{bmatrix} = \begin{bmatrix}\quad\end{bmatrix}$$

$$\left(\begin{bmatrix}4 & 7\\8 & 9\end{bmatrix} + \begin{bmatrix}5 & 3\\0 & 1\end{bmatrix}\right)^T * \begin{bmatrix}10 & 2 & 7\\6 & 3 & 5\end{bmatrix} = \begin{bmatrix}\quad\end{bmatrix}$$

Solving simultaneous linear equations

Now we'll use matrices to represent sets of algebraic equations. The solution to these sets of equations can be solved using matrix methods. The simultaneous solution of multiple equations finds its way in to many common engineering problems. In fact, modern structural engineering analysis techniques are ALL ABOUT solving systems of equations simultaneously. You'll see the following material in CES 4141 (structures II) for sure.

• Matrices - an organized way of presenting a set of coupled equations.

• We seek a single unique solution that satisfies all the equations at the same time. Consider the three *coupled linear* equations below:

$$3X_1 + 5X_2 + 2X_3 = 8$$
$$2X_1 + 3X_2 - 1X_3 = 1$$
$$1X_1 - 2X_2 - 3X_3 = -1$$

- *Coupled* because each equation has one or more terms in common with the others, X_1, X_2, X_3, so that a change in one of these variables will affect more than one equation.
- *Linear* because each equation contains only first order terms of X_1, X_2, X_3. There are no terms like X_1^2, or $\sqrt{X_2}$, or $\log(X_3)$, or $1/(X_1X_2)$, etc.
- Using the rules of matrix multiplication, we can represent the above equations in matrix form:

$$\begin{bmatrix} 3 & 5 & 2 \\ 2 & 3 & -1 \\ 1 & -2 & -3 \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \\ X_3 \end{bmatrix} = \begin{bmatrix} 8 \\ 1 \\ -1 \end{bmatrix}$$

coefficient matrix A unknowns vector X solution vector B

Try multiplying the matrices A and X together, make sure you can get the original equations above. There are several ways to solve for unknown vector. Each method involves some manipulations to the coefficient matrix using algebraic rules, creating a new and equivalent problem in a more easily solvable form. These manipulations involve the addition of multiples of one row to another. Adding one row to another result in an equivalent equation, since both sides are equal.

For example, starting with the two equations:

$$X_1 + 5X_2 = 3 \quad (1)$$

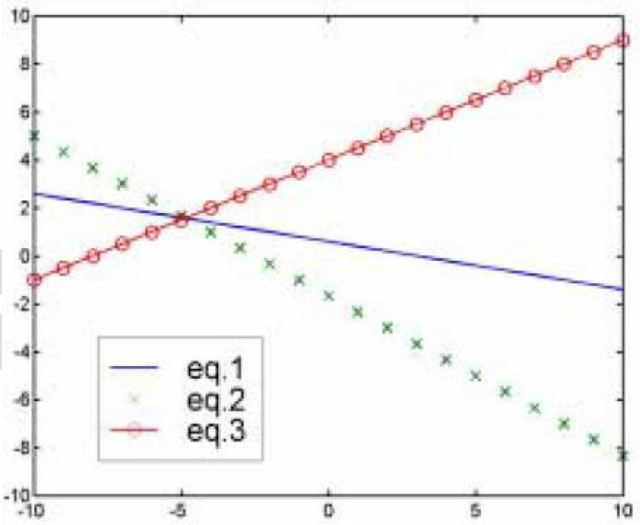
$$-2X_1 - 3X_2 = 5 \quad (2)$$

their addition gives:

$$-1X_1 + 2X_2 = 8 \quad (3)$$

This addition does not add any new information, but it does present a new form of the old information.

If we plot these 3 equations, the solution is the place where they intersect. That is, we are seeking the one pair of X1 and X2 values which lines along both or the original lines (eq1, eq2). For only two equations and two unknowns, finding the solution is trivial substitution. However, many problems require the solution of many equations and as many unknowns. If we had 5 equations and 5 unknowns, things get more difficult to handle with just substitution. We'll now look at several ways to systematically solve a set of simultaneous equations of any size. First we put the equations in matrix form, then manipulate the matrix to get a solution.



Gaussian Elimination (method #1):

Consider the three coupled linear equations given earlier. The original form looks like this:

$$\begin{bmatrix} 3 & 5 & 2 \\ 2 & 3 & -1 \\ 1 & -2 & -3 \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \\ X_3 \end{bmatrix} = \begin{bmatrix} 8 \\ 1 \\ -1 \end{bmatrix}. \quad (4)$$

But what if we could recast the same problem to look like this?

$$\begin{bmatrix} 3 & 5 & 2 \\ 0 & -\frac{11}{3} & -\frac{11}{3} \\ 0 & 0 & -2 \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \\ X_3 \end{bmatrix} = \begin{bmatrix} 8 \\ -\frac{11}{3} \\ -4 \end{bmatrix} \quad (5)$$

This makes life easier, since there is less coupling between equations. In fact, X_3 can be solved immediately using the bottom equation $\Longrightarrow 0X_1 + 0X_2 - 2X_3 = -4$ to give $X_3 = 2$. Now the result can be used to write the middle equation as $0X_1 - \frac{11}{3}X_2 - \frac{11}{3}(2) = -\frac{11}{3}$ to get $\Longrightarrow X_2 = -1$. Finally, the known values X_1, X_2 are used to solve for X_1 in the first equation to get $\Longrightarrow X_1 = 3$.

This easy solution process is called ***back-substitution***, and is made possible by the lower triangle of zeros in the coefficient matrix, marked with a dashed triangular box.

Great, it would be nice if the problem looked like Eq. (5), but it actually looks like Eq. (4), so what now?

We can use a series of additions of the 3 equations in Eq. (4) to get it to look like Eq. (5). In a series of steps just like the above addition of Eqs (1), (2) to get Eq. (3) we'll reduce the coefficients $A(2, 1), A(3, 1), A(3, 2)$ to zero, in that order.

KEY: Whatever we do to the l.h.s. of an equation, we do to the r.h.s. so we don't change the problem.

Let's rewrite the matrix in Eq. (4) to get one matrix with both A and B in it:

This is called augmenting the matrix.

$$\text{l.h.s.} \quad \left[\begin{array}{ccc|c} 3 & 5 & 2 & 8 \\ 2 & 3 & -1 & 1 \\ 1 & -2 & -3 & -1 \end{array}\right] \leftarrow \text{pivot row} \qquad \text{r.h.s.} \qquad (6)$$

$$\text{A} \qquad \text{B}$$

Step 1) - reduce A(2,1) to zero

New Row 2 = (Row 1)(-2/3) + (Row 2)

Row 1 is called pivot row for this step. Some multiple of it is added to another equation, but the pivot row remains unchanged.

New Row 2 = (Row 1)(-2/3) + (Row 2)
Row 1 is called pivot row for this step. Some multiple of it is added to another equation, but the pivot row remains unchanged

$$\text{add} \quad \frac{\left[\begin{array}{cccc} 3\left(-\frac{2}{3}\right) & 5\left(-\frac{2}{3}\right) & 2\left(-\frac{2}{3}\right) & 8\left(-\frac{2}{3}\right) \\ 2 & 3 & -1 & 1 \end{array}\right]}{\left[\begin{array}{cccc} 0 & -\frac{1}{3} & -\frac{7}{3} & -\frac{13}{3} \end{array}\right]} \qquad (7)$$

$$\left[\begin{array}{ccc|c} 3 & 5 & 2 & 8 \\ 0 & -\frac{1}{3} & -\frac{7}{3} & -\frac{13}{3} \\ 1 & -2 & -3 & -1 \end{array}\right] \leftarrow \text{New Row 2} \qquad (8)$$

Step 2) - reduce A(3,1) to zero

New Row 3 = (Row1)(-1/3) + (Row 3)
Row 1 is the pivot row again
Expanding this instruction like we did in Eq.(7), the result is

$$\left[\begin{array}{cccc} 3 & 5 & 2 & 8 \\ 0 & -\frac{1}{3} & -\frac{7}{3} & -\frac{13}{3} \\ 0 & -\frac{11}{3} & -\frac{11}{3} & -\frac{11}{3} \end{array}\right] \quad \text{New Row 3} \qquad (9)$$

Now we need to reduce A(3,2) to zero. If we added some multiple of Row 1, then A(3,1) would become non-zero. Instead, we'll need to add some multiple of Row 2 to Row 3 to get a new Row 3.

Before we go on, let's consider error reduction...

error reduction - swap Rows 2 and 3

• If there were some numerical error in the computer storage of any coefficient, say the error from rounding off the -1/3 currently in spot A(2,2), then when we multiply Row 2 by some factor and add it to Row 3, we also multiply the error by that factor.

• If we can always multiply by some small number (less than 1), we can reduce the propagation of that round-off error.

• We can enforce this by making sure the lead coefficient (the pivot coefficient) in the pivot row has the largest absolute value among itself and all the coefficients under it (the coefficients to be reduced to zero).

• Since it does not matter what order I put the equations in, we will rearrange rows when we find the current pivot coefficient has a smaller absolute value than those beneath it. In the current example we have:

$$\left[\begin{array}{ccc|c} 3 & 5 & 2 & 8 \\ 0 & -\frac{1}{3} & -\frac{7}{3} & -\frac{13}{3} \\ 0 & -\frac{11}{3} & -\frac{11}{3} & -\frac{11}{3} \end{array}\right]$$

Row 2 will be the pivot row to eliminate A(3,2), which makes A(2,2) the pivot coefficient. Since $\left|-\frac{11}{3}\right| > \left|-\frac{1}{3}\right|$, we'll swap Rows 2 and 3, and the new pivot coefficient will be largest:

$$\left[\begin{array}{ccc|c} 3 & 5 & 2 & 8 \\ 0 & -\frac{11}{3} & -\frac{11}{3} & -\frac{11}{3} \\ 0 & -\frac{1}{3} & -\frac{7}{3} & -\frac{13}{3} \end{array}\right] \quad (10)$$

Now we can continue with minimal round-off error propagation

Step 3) - reduce A(3,2) to zero

New Row 3 = (Row 2)(-1/11) + (Row 3)
Row 2 is the pivot row now

$$\left[\begin{array}{ccc|c} 3 & 5 & 2 & 8 \\ 0 & -\frac{11}{3} & -\frac{11}{3} & -\frac{11}{3} \\ 0 & 0 & -2 & -4 \end{array}\right] \quad (11)$$

New Row 3

Now let's expand this to its full form with A, X, B in separate matrices

$$\begin{bmatrix} 3 & 5 & 2 \\ 0 & -\frac{11}{3} & -\frac{11}{3} \\ 0 & 0 & -2 \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \\ X_3 \end{bmatrix} = \begin{bmatrix} 8 \\ -\frac{11}{3} \\ -4 \end{bmatrix} \quad (12)$$

Summary of Gaussian Elimination:

Starting with a problem defined as in Eq. (4), we created some equivalent problem Eq. (5),(12) with the desired lower triangle of zeros. (12) and (4) are equivalent since all we did was a series of steps where we added the same thing to both sides of a row.

• Now a little back-substitution (the paragraph following Eq. (5)) gives us

$$\begin{bmatrix} X_1 \\ X_2 \\ X_3 \end{bmatrix} = \begin{bmatrix} 3 \\ -1 \\ 2 \end{bmatrix} \tag{13}$$

Solution Method 2

The Inverse matrix (matrix version of division):

Let's take another look at the matrix problem

$$AX = B \tag{14}$$

where we have A, B, and we want to solve for the unknown(s) X. If this were a scalar problem, the answer is just to divide both sides by A

$$\frac{A}{A}X = \frac{B}{A} \qquad \text{or} \qquad X = B/A\,. \tag{15}$$

There is no dividing with matrices, but we can try to find some other matrix (M) to pre-multiply both sides with such that we're left with

$$M(AX) = M(B) = X \tag{16}$$

which says that

$$MAX = X. \tag{17}$$

What we seek what is called the Identity matrix (denoted I). A square matrix with ones on the diagonal, and zeros everywhere else. Multiplying any matrix by the Identity matrix (I) is equivalent to multiplying by one in the scalar world.

$$IX = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \\ X_3 \end{bmatrix} = \begin{bmatrix} X_1 \\ X_2 \\ X_3 \end{bmatrix} = X \tag{18}$$

TRY IT!! Perform the multiplication in Eq. (18) by hand to see that it's true.

So Eqs. (16), and (17) say that what we want is some 'magic' matrix M such that

$$MAX = IX = X \quad \text{or} \quad MA = I \ . \tag{19}$$

Back in the scalar world again, we can always get 1 by dividing anything by itself, or multiplying anything by its inverse

$$1 = \frac{5}{5} = 5*\frac{1}{5} = \frac{Y}{Y} = \frac{1}{Y}Y.$$

So let's call the matrix we seek, M, the inverse of A, since we want the resulting multiplication to be the identity matrix. We'll change the symbol from M to A^{-1} to indicate the inverse of A.

$$MA = A^{-1}A = I.$$

What we seek is A^{-1} so we can get the solution for X from

$$AX = B \tag{20}$$

equivalent to

$$A^{-1}AX = A^{-1}B \tag{21}$$

equivalent to

$$IX = A^{-1}B \tag{22}$$

equivalent to

$$X = A^{-1}B \tag{23}$$

Gauss-Jordan Elimination - finding the inverse

Gauss-Jordan Elimination I: For the first version of this method, let's continue where we left off in the Gaussian elimination example:

$$\begin{bmatrix} 3 & 5 & 2 \\ 0 & -\frac{11}{3} & -\frac{11}{3} \\ 0 & 0 & -2 \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \\ X_3 \end{bmatrix} = \begin{bmatrix} 8 \\ -\frac{11}{3} \\ -4 \end{bmatrix} \tag{24}$$

where Gaussian elimination is used to get the lower triangle of zeros.

• The idea of Gauss Jordan elimination is to continue the algebra to get an upper triangle of zeros, until the three equations are completely uncoupled.

• Just as we worked from top to bottom to get zeros for A(2,1), A(3,1), and A(3,2) in that order, we can start working from the bottom up to make A(2,3), A(1,3), and A(1,2) zero, in that order.

• Just start with the bottom as the pivot row to zero A(2,3), A(1,3), then switch the pivot row to the second row to make A(1,2) zero.

The three steps to get rid of A(2,3), A(1,3), A(1,2) would be

Step 1) New Row 2 = (Row 3)(-11/6) + (Row 2)

Step 2) New Row 1 = (Row 3)(1) + (Row 1)

Step 3) New Row 1 = (Row 2)(15/11) + (Row 1)

And the result is

$$\begin{bmatrix} 3 & 0 & 0 \\ 0 & -\frac{11}{3} & 0 \\ 0 & 0 & -2 \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \\ X_3 \end{bmatrix} = \begin{bmatrix} 9 \\ \frac{11}{3} \\ -4 \end{bmatrix} \qquad (25)$$

We see now that the three equations have been completely uncoupled.

Now if we multiply each row by whatever it takes to make the A coefficient in that row equal to 1, we get

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \\ X_3 \end{bmatrix} = \begin{bmatrix} 3 \\ -1 \\ 2 \end{bmatrix} \qquad (26)$$

This last bit of making each coefficient equal to one is called **normalizing**

The solution $X_1 = 3$, $X_2 = -1$, $X_3 = 2$ is now shown directly on the right hand side.

Notice that A has been transformed into the identity matrix!

What we have done is built the inverse matrix A^{-1} into the right hand side of the equation. So what we see in Eq. (26) is exactly the result of

$$A^{-1}AX = A^{-1}B$$

Summary of Gauss Jordan Elimination I -

1) Augment coefficient matrix with the solution vector

2) Create lower triangle of zeros

3) Create upper triangle of zeros

4) Normalize the coefficient matrix

5) Values of the unknowns are now in the right hand column

Gauss-Jordan Elimination part II - Finding A^{-1} directly

- In the above method, we used Gauss-Jordan elimination, and wound up with the result of $A^{-1}B$ on the right hand side.
- We never actually explicitly 'saw' A^{-1}, it was just built in.

We can also used Gauss-Jordan elimination to *explicitly compute the inverse* of A, then multiply this A^{-1} by B to get the desired unknowns in X as in $X = A^{-1}B$.

As you may have noticed, finding A^{-1} has *nothing* to do with what's in B. Only A is needed to find A^{-1}. Let's look way back at Eq. (6), repeated below

$$\left[\begin{array}{ccc|c} 3 & 5 & 2 & 8 \\ 2 & 3 & -1 & 1 \\ 1 & -2 & -3 & -1 \end{array}\right] \qquad \text{A } \textit{augmented} \text{ with B.} \qquad (27)$$

A B

We *augmented* the matrix A with B, since we wanted to operate on B.

Now Let's throw out B and replace it with the identity matrix

If we now use the Gauss-Jordan elimination method to reduce the three left hand columns to the identity matrix, the result on the right hand side is the inverse of A. We do the exact same steps we did before to get to Eq. (26). Only this time, instead of tacking on B for the ride, we tack on what starts as for the ride. When its all over, the right three columns are A^{-1}.

Example: we'll just solve for the inverse of a 2x2
Start with

$$\left[\begin{array}{cc|cc} 1 & 2 & 1 & 0 \\ 3 & 4 & 0 & 1 \end{array}\right] \quad (29)$$

The steps are:

Step 1) New Row 2 = (Row 1)(-3) + Row 2 ..... $\left[\begin{array}{cc|cc} 1 & 2 & 1 & 0 \\ 0 & -2 & -3 & 1 \end{array}\right]$

finished lower triangle of zeros, now get upper triangle of zeros

Step 2) New Row 1 = (Row 2)(1) + Row 1 ..... $\left[\begin{array}{cc|cc} 1 & 0 & -2 & 0 \\ 0 & -2 & -3 & 1 \end{array}\right]$

Step 3) Normalize each row: (Row 2)(-1/2)

The result is: $\left[\begin{array}{cc|cc} 1 & 0 & -2 & 1 \\ 0 & 1 & 1.5 & -0.5 \end{array}\right]$. (30)

Which means $A^{-1} = \begin{bmatrix} -2 & 1 \\ 1.5 & -0.5 \end{bmatrix}$. (31)

How can I check to make sure the inverse of $\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$ is really $\begin{bmatrix} -2 & 1 \\ 1.5 & -0.5 \end{bmatrix}$?????

We know that $A^{-1}A = I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$,

so let's try it

$\begin{bmatrix} -2 & 1 \\ 1.5 & -0.5 \end{bmatrix} * \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$. This checks out, so we've found the inverse

Solving equations using Mathcad

Let's go back to the original problem in this lecture

$$3X_1 + 5X_2 + 2X_3 = 8$$
$$2X_1 + 3X_2 - 1X_3 = 1$$
$$1X_1 - 2X_2 - 3X_3 = -1$$

Re-written as

$$\begin{bmatrix} 3 & 5 & 2 \\ 2 & 3 & -1 \\ 1 & -2 & -3 \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \\ X_3 \end{bmatrix} = \begin{bmatrix} 8 \\ 1 \\ -1 \end{bmatrix}$$

coefficient matrix A — unknowns vector X — solution vector B

Let's use the built in Mathcad commands to solve this problem

$$A := \begin{pmatrix} 3 & 5 & 2 \\ 2 & 3 & -1 \\ 1 & -2 & -3 \end{pmatrix} \qquad B := \begin{pmatrix} 8 \\ 1 \\ -1 \end{pmatrix}$$

$$C := A^{-1} \cdot B \qquad C = \begin{pmatrix} 3 \\ -1 \\ 2 \end{pmatrix}$$

Yep, 13 pages of lecture notes captured in a single Mathcad line

(of course the inverse function contains the Mathcad code to augment the matrix, create the upper and lower triangles, and normalize)

We can also use a built in function called 'lsolve' to calculate the solution t A*X = B

$$C := \text{lsolve}(A, B) \qquad C = \begin{pmatrix} 3 \\ -1 \\ 2 \end{pmatrix}$$

Curve fitting - motivation

For root finding, we used a given function to identify where it crossed zero

where does $f(x) = 0$??

Q: Where does this given function $f(x)$ come from in the first place?

- Analytical models of phenomena (e.g. equations from physics)
- Create an equation from observed data

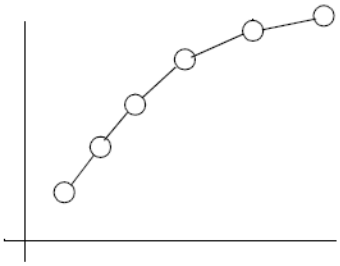
1) Interpolation (connect the data-dots)

If data is reliable, we can plot it and connect the dots

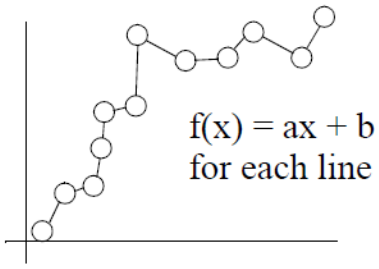
This is piece-wise, linear interpolation

This has limited use as a general function $f(x)$

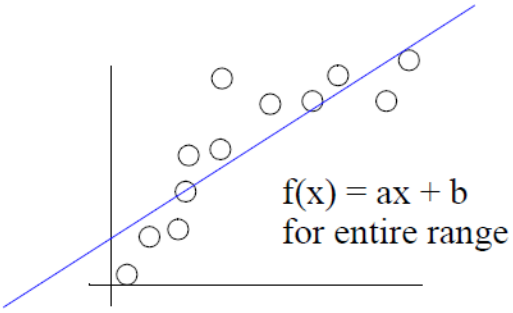
Since its really a group of small $f(x)$ s, connecting one point to the next it doesn't work very well for data that has built in random error (scatter)



2) Curve fitting - capturing the trend in the data by assigning a single function across the entire range. The example below uses a straight line function



Interpolation

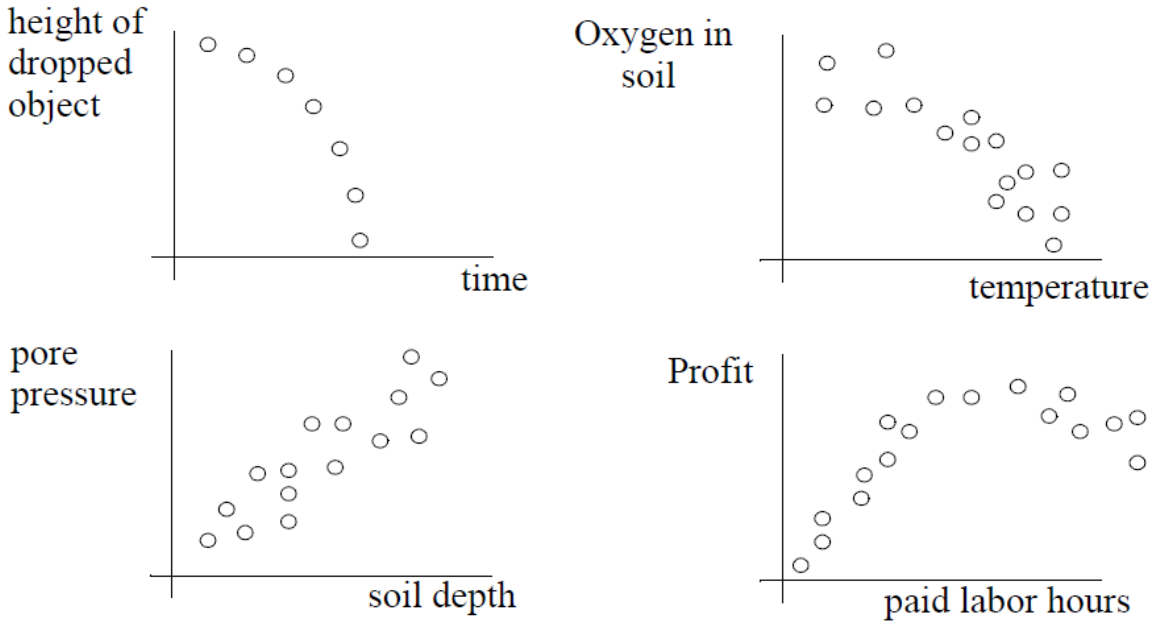


Curve Fitting

A straight line is described generically by $f(x) = ax + b$

The goal is to identify the coefficients 'a' and 'b' such that f(x) 'fits' the data well

other examples of data sets that we can fit a function to.



Is a straight line suitable for each of these cases ?
No. But we're not stuck with just straight line fits. We'll start with straight lines, then expand the concept.

POSSIBLE QUESTIONS

PART A (20 multiple choice questions)

Online Examinations

PART B (2 marks questions)

1. What is meant by matrix?
2. State interpolation.
3. What is meant by curve fitting?
4. What is Gauss-Seidal method?
5. What is extrapolation?

PART C (6 marks questions)

1. Give an account on Gauss-Seidal method.
2. Illustrate matrix addition.
3. Describe matrix multiplication.
4. Write note on interpolation.
5. Give a detailed account on extrapolation.
6. Give a brief account on curve fitting.
7. Write note on matrix addition and multiplication.
8. Explain Gauss-Seidal iteration matrix.

KARPAGAM ACADEMY OF HIGHER EDUCATION

Class: III B.Sc Chemistry Course Name: Applications of Computers in Chemistry

Course Code: 16CHU502B Unit: IV Batch-2016-2019

S. No	Questions	Option A	Option B	Option C	Option D	Answer
1.	Iterative method is a ---------- method	Direct method	InDirect method	both 1st & 2nd	either st &2nd	InDirect method
2.	The condition for convergence of Gauss Seidal method is that the ------should be diagonally dominant	Constant matrix	unknown matrix	Coefficient matrix	Unit matrix	Coefficient matrix
3.	In ---------- method, the coefficient matrix is transformed into diagonal matrix	Gauss elimination	Gauss jordan	Gauss jacobi	Gauss seidal	Gauss jordan
4.	---------- Method takes less time to solve a system of equations comparatively than ' iterative method'	Direct method	Indirect method	Regula falsi	Bisection	Direct method
5.	The iterative process continues till ---------- is secured.	convergency	divergency	oscillation	none	convergency
6.	In Gauss elimination method, the solution is getting by means of ---------- from which the unknowns are found by back substitution.	Elementary operations	Elementary column operations	Elementary diagonal operations	Elementary row operations	Elementary row operations
7.	The ---------- is reduced to an upper triangular matrix or a diagonal matrix in direct methods.	Coefficient matrix	Constant matrix	unknown matrix	Augment matrix	Augment matrix
8.	The augment matrix is the combination of ----------.	Coefficient matrix and constant matrix	Unknown matrix and constant matrix	Coefficient matrix and Unknown matrix	Coefficient matrix, constant matrix and Unknown matrix	Coefficient matrix and constant matrix
9.	The given system of equations can be taken as in the form of ----------	A = B	BX= A	AX= B	AB = X	AX= B
10.	Which is the condition to apply Gauss Seidal method to solve a	1st row is dominant	1st column is dominant	diagonally dominant	last row dominant	diagonally dominant

	system of equations?					
11.	Crout's method and triangularisation method are ----- - method.	Direct	Indirect	Iterative	Interpolation	Direct
12.	The solution of simultaneous linear algebraic equations are found by using-----	Direct method	Indirect method	both 1st & 2nd	Bisection	InDirect method
13.	The matrix is ____ if the numerical value of the leading diagonal element in each row is greater than or equal to the sum of the numerical value of other element in that row.	orthogonal	symmetric	diagonally dominant	singular	diagonally dominant
14.	If the Eigen values of A are -6, 2, 4 then _____ is dominant.	2	-6	4	-2	2
15.	The Gauss – Jordan method is the modification of _____ method.	Gauss –Elimination	Gauss – Jacobi	Gauss – Seidal	interpolation	Gauss –Elimination
16.	x^2 + 5x + 4 = 0 is a _____ equation.	algebraic	transcendental	wave	heat	algebraic
17.	a + b logx + c sinx + d = 0 is a _____ equation.	algebraic	transcendental	wave	heat	transcendental
18.	In Gauss – Jordan method, the augmented matrix is reduced into _____matrix	upper triangular	lower triangular	diagonal	scalar	diagonal
19.	The 1st equation in Gauss – Jordan method, is called _____ equation.	pivotal	dominant	reduced	normal	pivotal
20.	The element a11 in Gauss – Jordan method is called _____ element.	Eigen value	Eigen vector	pivot	root	pivot
21.	The system of simultaneous linear equation in n unknowns AX = B if A is diagonally dominant then the system is said to be _____ system	dominant	diagonal	scalar	singular	diagonal
22.	The convergence of Gauss – Seidal method is roughly _____ that of Gauss – Jacobi method	twice	thrice	once	4 times	twice
23.	Jacobi's method is used only	symmetric	skew-symmetric	singular	non-singular	symmetric

	when the matrix is ____					
24.	Gauss Seidal method always ---------- for a special type of systems.	Converges	diverges	oscillates	equal	Converges
25.	Condition for convergence of Gauss Seidal method is ----------	Coefficient matrix is diagonally dominant	pivot element is Zero	Coefficient matrix is not diagonally dominant	pivot element is non Zero	Coefficient matrix is diagonally dominant
26.	Modified form of Gauss Jacobi method is ---------- method.	Gauss Jordan	Gauss Siedal	Gauss Jacobbi	Gauss Elimination	Gauss Siedal
27.	In Gauss elimination method by means of elementary row operations, from which the unknowns are found by ---------- method	Forward substitution	Backward substitution	random	Gauss Elimination	Backward substitution
28.	In iterative methods, the solution to a system of linear equations will exist if the absolute value of the largest coefficient is ---------- the sum of the absolute values of all remaining coefficients in each equation.	less than	greater than or equal to	equal to	not equal	greater than or equal to
29.	In ---------- iterative method, the current values of the unknowns at each stage of iteration are used in proceeding to the next stage of iteration.	Gauss Siedal	Gauss Jacobi	Gauss Jordan	Gauss Elimination	Gauss Siedal
30.	The direct method fails if any one of the pivot elements become ----.	Zero	one	two	negative	Zero
31.	In Gauss elimination method the given matrix is transformed into ----------.	Unit matrix	diagonal matrix	Upper triangular matrix	lower triangular matrix	Upper triangular matrix
32.	If the coefficient matrix is not diagonally dominant, then by ---------- that diagonally dominant coefficient matrix is formed.	Interchanging rows	Interchanging Columns	adding zeros	Interchanging row and Columns	Interchanging row and Columns
33.	Gauss Jordan method is a ------	Direct method	InDirect method	iterative method	convergent	Direct method
34.	Gauss Jacobi method is a -------	Direct method	InDirect method	iterative method	convergent	InDirect method
35.	The modification of Gauss –	Gauss Jordan	Gauss Siedal	Gauss Jacobbi	gauss elemination	Gauss Siedal

	Jordan method is called ----------					
36.	Gauss Seidal method always converges for ---------- of systems	Only the special type	all types	quadratic types	first type	Only the special type
37.	In solving the system of linear equations, the system can be written as ---	BX = B	AX = A	AX = B	AB = X	AX = B
38.	In solving the system of linear equations, the augment matrix is	(A, A)	(B, B)	(A, X)	(A, B)	(A, B)
39.	In the direct methods of solving a system of linear equations, at first the given system is written as ---------- form.	An augment matrix	a triangular matrix	constant matrix	Coefficient matrix	An augment matrix
40.	All the row operations in the direct methods can be carried out on the basis of --	all elements	pivot element	negative element	Positive element	pivot element
41.	The direct method fails if --------	1st row elements 0	1st column elements 0	Either 1st or 2nd	2 nd row is dominant	Either 1st or 2nd
42.	The elimination of the unknowns is done not only in the equations below, but also in the equations above the leading diagonal is called ----------	Gauss elimination	Gauss jordan	Gauss jacobi	Gauss siedal	Gauss jordan
43.	In Gauss Jordan method, we get the solution ----------	without using back substitution method	By using back substitution method	by using forward substitution method	Without using forward substitution method	By using back substitution method
44.	If the coefficient matrix is diagonally dominant, then ---------- method converges quickly.	Gauss elimination	Gauss jordan	Direct	Gauss siedal	Gauss siedal
45.	Which is the condition to apply Jocobi's method to solve a system of equations	1st row is dominant	1st column is dominant	diagonally dominant	2 nd row is dominant	diagonally dominant
46.	Iterative method is a ---------- method	Direct method	InDirect method	Interpolation	extrapolation	InDirect method
47.	As soon as a new value for a variable is found by iteration it is used immediately in the equations is called ----------.	Iteration method	Direct method	Interpolation	extrapolation	Iteration method
48.	---------- is also a self-	Iteration method	Direct method	Interpolation	extrapolation	Iteration method

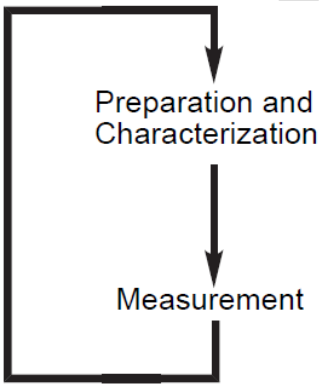
	correction method.					
49.	The condition for convergence of Gauss Seidal method is that the ------ should be diagonally dominant	Constant matrix	unknown matrix	Coefficient matrix	extrapolation	Coefficient matrix
50.	In ---------- method, the coefficient matrix is transformed into diagonal matrix	Gauss elimination	Gauss jordan	Gauss jacobi	Gauss seidal	Gauss jordan
51.	We get the approximate solution from the ----------.	Direct method	InDirect method	fast method	Bisection	InDirect method
52.	The iterative process continues till ---------- is secured.	convergency	divergency	oscillation	point	convergency
53.	In Gauss elimination method, the solution is getting by means of ---------- from which the unknowns are found by back substitution.	Elementary operations	Elementary column operations	Elementary diagonal operations	Elementary row operations	Elementary row operations
54.	The method of iteration is applicable only if all equation must contain one coefficient of different unknowns as ---------- than other coefficients.	smaller	larger	equal	non zero	larger
55.	The ---------- is reduced to an upper triangular matrix or a diagonal matrix in direct methods.	Coefficient matrix	Constant matrix	unknown matrix	Augment matrix	Augment matrix
56.	The augment matrix is the combination of ----------.	Coefficient matrix and constant matrix	Unknown matrix and constant matrix	Coefficient matrix and Unknown matrix	Coefficient matrix, constant matrix and Unknown matrix	Coefficient matrix and constant matrix
57.	The given system of equations can be taken as in the form of ----------	A = B	BX= A	AX= B	AB = X	AX= B
58.	The sufficient condition of iterative methods will be satisfied if the large coefficients are along the ---------- of the coefficient matrix.	Rows	Coloumns	Leading Diagonal	elements	Leading Diagonal
59.	Which is the condition to apply Gauss Seidal method to solve a	1st row is dominant	1st column is dominant	diagonally dominant	Leading Diagonal	diagonally dominant

	system of equations?					
60.	In the absence of any better estimates, the ----------of the function are taken as x = 0, y = 0, z = 0.	Initial approximations	roots	points	final value	Initial approximations
61.	The solution of simultaneous linear algebraic equations are found by using-	Direct method	InDirect method	fast method	Bisection	InDirect method

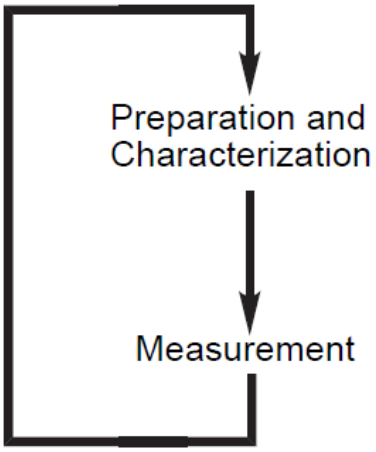
UNIT V
Syllabus

Conceptual background of molecular modelling: Potential energy surfaces. Elementary ideas of molecular mechanics and practical MO methods.

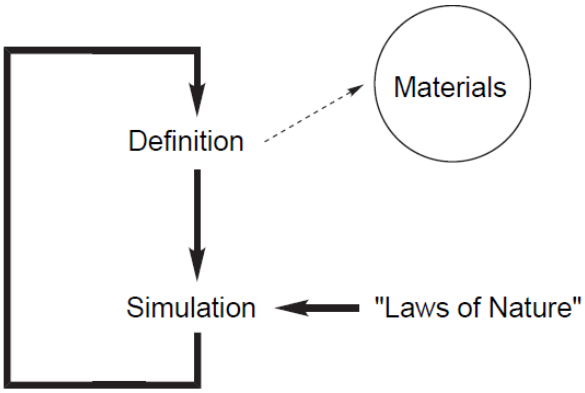
Chemists make useful materials (pharmaceuticals, polymers, etc.). Typically, they do this by seeking to establish connections with molecular structure.



By Molecular Modeling



The goal is exactly the same as is the procedure. The difficult step (materials preparation) has been removed.



Why i s molecular modeling important to the teaching of chemistry?

Models are what we teach. Students need to learn to "think like a molecule". To do this they need to "see" what a molecule sees and "feel" what a molecule feels. Models give us the best and most direct view of the molecular world.

Modeling is the best tool for learning about chemical theory. VSEPR, Lewis structures, Hückel MO are all crude attempts to convert good theories into chemical predictions. Modern computational methods give a much more accurate assessment of theoretical predictions.

Models are easy to use, inexpensive, safe. Modeling is a student-friendly educational too. It is not just for "experts".

Should molecular modeling replace e x p e r i m e n t a l chemistry?

Of course not!

The goals of chemistry are not changed by molecular modeling. On a practical level we want to learn how to make things (synthesis) and how to figure out what things are made of (analysis). On an intellectual level we want to understand the "rules" that describe chemical behavior.

Molecular modeling is, like NMR, a tool for achieving these goals. Since two of the goals - synthesis and analysis - are experimental, they cannot (and should not) be done away with.

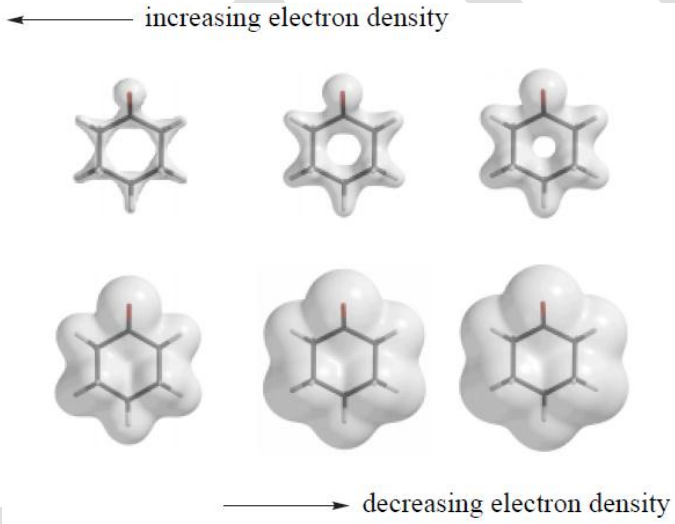
However, modeling does change the way we do syntheses and analysis. And, it speaks directly to the intellectual goals of chemistry.

A modern chemical education still requires practical training in experimentation, but it requires training in modeling too.

Molecular models

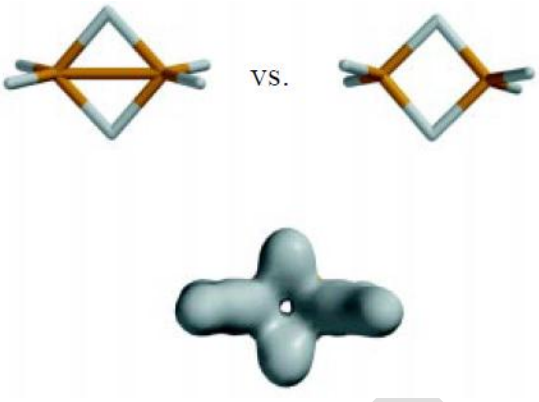
Electron Densities

Electron densities show the locations of electrons. Large values of the density will first reveal atomic positions (the X-ray diffraction experiment) and then chemical bonds, while smaller values will indicate overall molecular size.



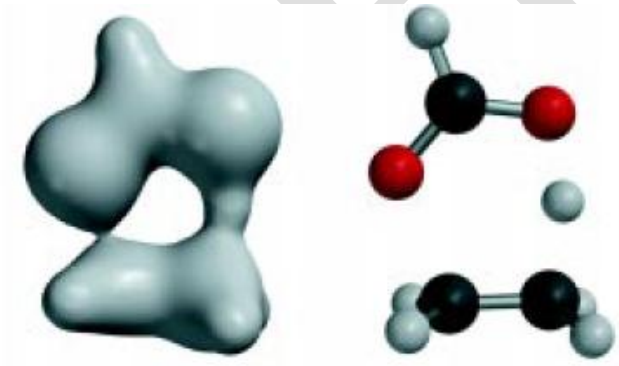
Electron Densities

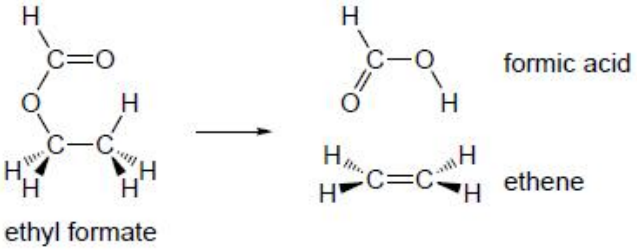
Unlike conventional structure models, electron densities assume no prior knowledge about bonding and, therefore, can be used to elucidate bonding. For example, the electron density for diborane shows no evidence for boron-boron bonding.



Electron Densities

Electron densities allow description of the bonding in transition states where there can be no (direct) information from experiment. For example, the electron density for the transition state for pyrolysis of ethyl formate, leading to formic acid and ethene, shows that the CO bond has nearly fully cleaved and that the hydrogen is midway between carbon and oxygen.

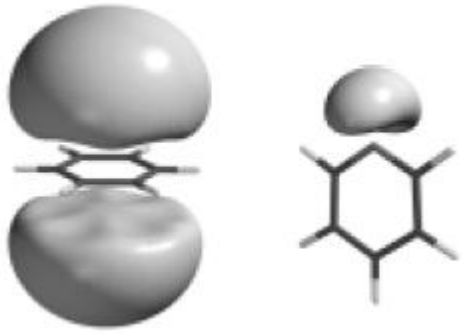




Electrostatic Potentials

The electrostatic potential is the energy of interaction of a point positive charge (an electrophile) with the nuclei and electrons of a molecule. Negative electrostatic potentials indicate areas

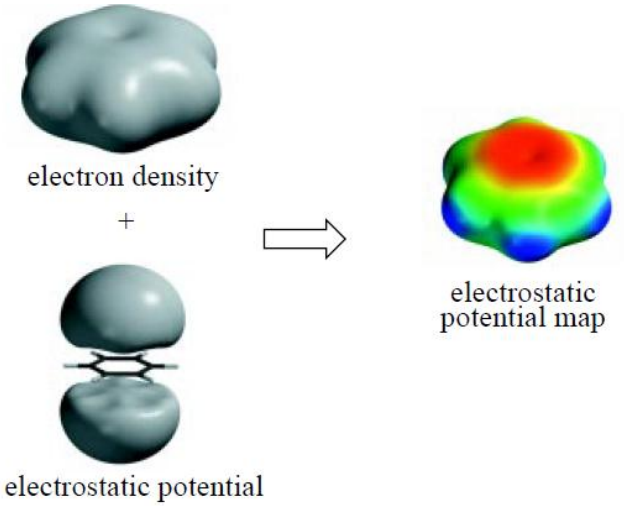
that are prone to electrophilic attack. For example, a negative electrostatic potential of benzene (left) shows that electrophilic attack should occur onto the system, above and below the plane of the ring, while the corresponding electrostatic potential for pyridine (right) shows that an electrophile should attack the nitrogen in the plane, and not the system of the ring.



The "electrophilic chemistry" of these two seemingly similar molecules is very different.

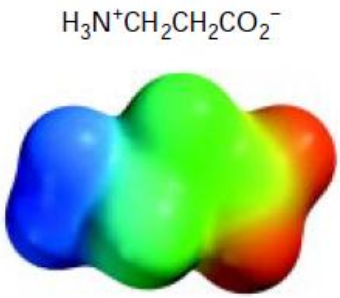
Electrostatic Potential Maps

A sufficiently small value of the electron density provides overall molecular size and shape (as given by a conventional space-filling or CPK model). The electrostatic potential can then be mapped onto the electron density by using color to represent the value of the potential. The resulting model simultaneously displays molecular size and shape and electrostatic potential value. For example, the electrostatic potential of benzene can be mapped onto the electron density. Colors toward "red" indicate negative values of the electrostatic potential, while colors toward "blue" indicate positive values of the potential.



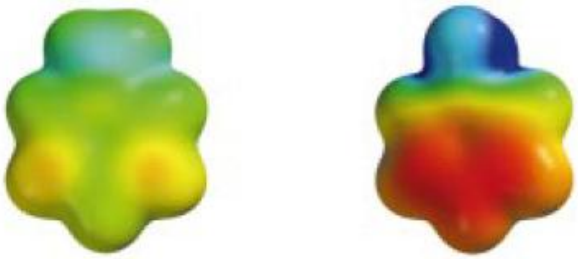
Electrostatic Potential Maps

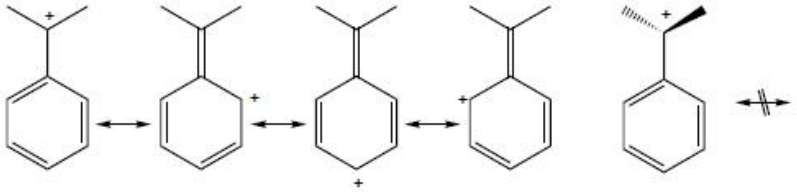
An electrostatic potential map conveys information about the distribution of charge in a molecule. For example, the electrostatic potential map of the zwitterionic form of -alanine shows the negative carboxylate (red) and the positive ammonium (blue) termini separated by the neutral (green) carbon chain. This is consistent with the usual resonance structure.



Electrostatic Potential Maps

An electrostatic potential map also gives information about delocalization of charge. For example, the electrostatic potential map for perpendicular benzyl cation (right) shows that the postive charge (blue) is localized on the benzylic carbon, while the charge in the planar cation (left) is fully delocalized. This agrees with conventional resonance arguments.

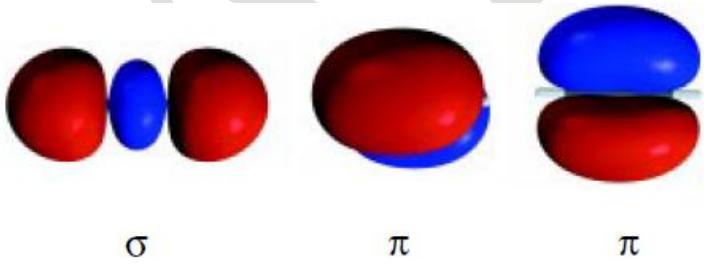




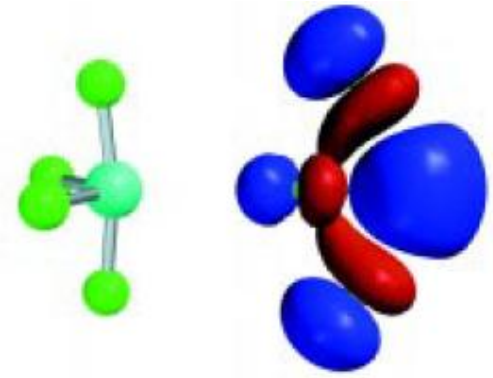
Molecular Orbitals

Molecular orbitals, solutions of the approximate quantum mechanical equations of electron motion, are made up of sums and differences of atomic solutions (atomic orbitals), just like molecules are made up of combinations of atoms.

Molecular orbitals for very simple molecules may often be interpreted in terms of familiar chemical bonds, for example, in acetylene.

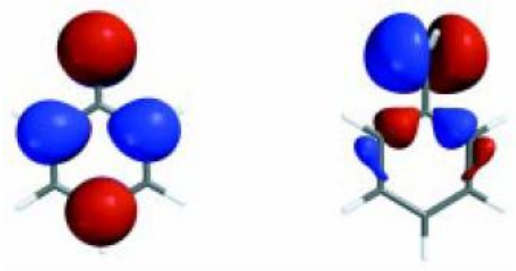


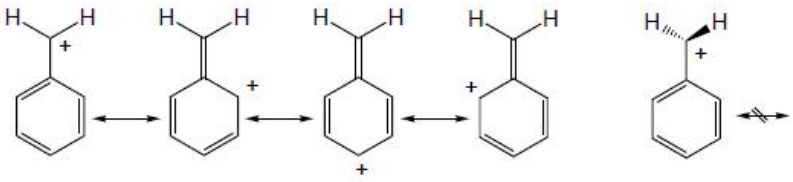
or of nonbonded lone pairs, for example, in sulfur tetrafluoride.



Molecular Orbitals

Unoccupied molecular orbitals may also provide useful information. For example, the lowest- unoccupied molecular orbital (LUMO) in planar benzyl cation (left) indicates that the positive charge is delocalized away from the benzylic carbon onto the *ortho* and *para* ring positions. The LUMO in perpendicular benzyl cation (right) resides almost entirely on the benzylic carbon. This agrees with conventional resonance arguments.





Molecular Orbitals

Why should molecular orbital descriptions be used instead of conventional Lewis structures?

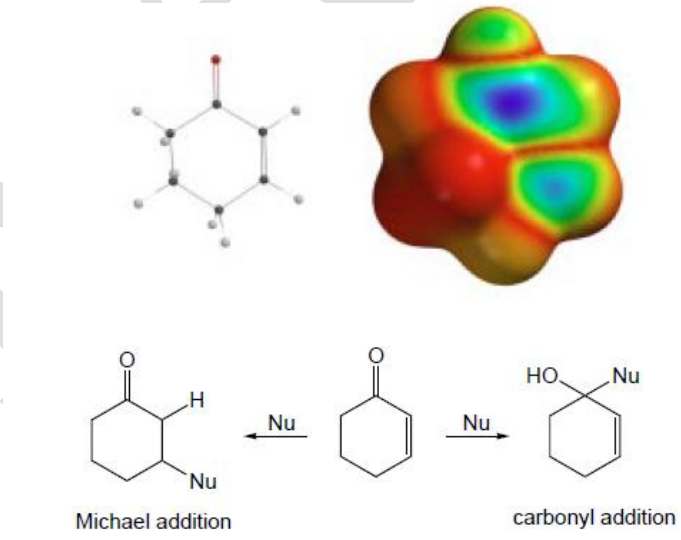
1. Molecular orbitals descriptions are often more compact than Lewis structures.

2. Molecular orbital descriptions offer quantitative information about molecular charge distributions. Lewis descriptions are strictly qualitative.

3. Molecular orbital descriptions are more generally applicable than Lewis descriptions.

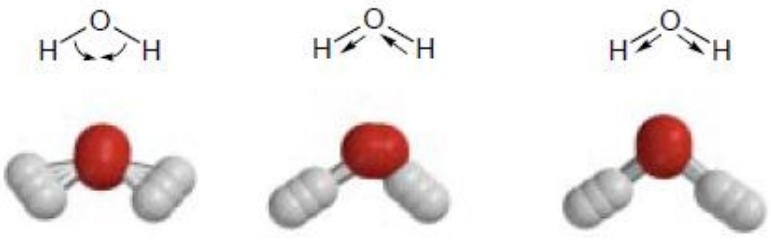
Molecular Orbital Maps

Molecular orbitals may also be mapped onto electron density surfaces. For example, a map of the lowest- unoccupied molecular orbital (LUMO) of cyclohexenone, where the "blue spots" indicate maximum values of the LUMO, reveals likely sites for nucleophilic attack, and anticipates both the "carbonyl chemistry" and "Michael chemistry" known for enones.



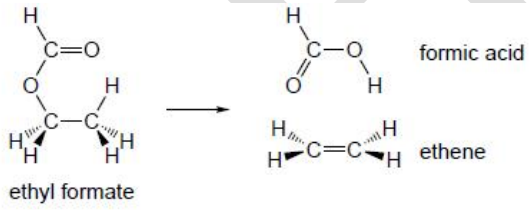
Models t h a t Move

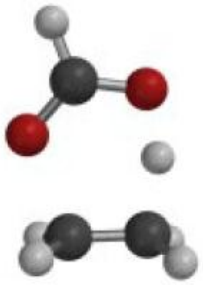
Models need not be limited to static pictures. "Movies" can be used to depict vibrations in stable molecules, for example, in water.



Models that Move

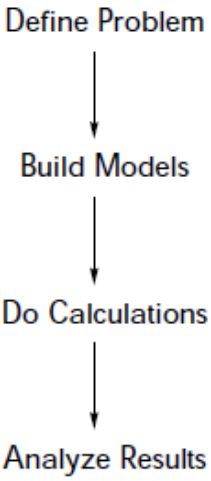
Motion along the reaction coordinate provides details about mechanism. For example, motion along the reaction coordinate for the pyrolysis of ethyl formate shows the simultaneous transfer of the hydrogen atom to the carbonyl oxygen along with carbon-oxygen bond cleavage.





Molecular Modeling in the Curriculum

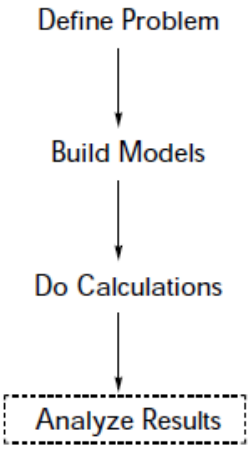
"Doing chemistry" with molecular modeling is a multi-step progress . . . not so different from doing experimental chemistry.



Given a "full" curriculum, the question that needs to be answered is how much of this process to turn over to students.

The Workbook Approach

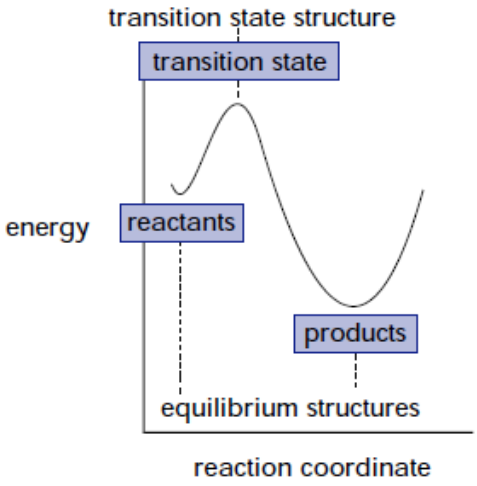
This leaves only the analysis of modeling results (and learning the chemistry that follows from these results) to the student.



The advantage of this approach is that it requires the fewest resources (hardware and software support, student training), while guaranteeing high quality models and maximum student-model contact.

Potential Energy Surfaces

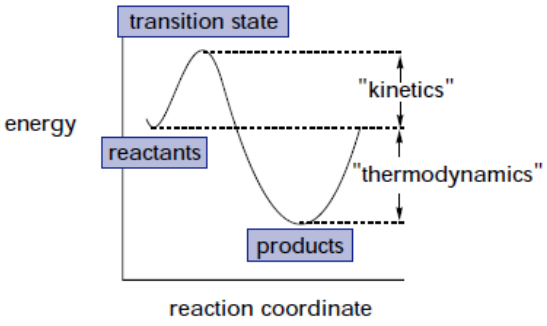
A potential energy surface is a plot of energy vs. reaction coordinate. It connects reactants to products via a transition state.



Energy minima correspond to equilibrium structures.

The energy maximum corresponds to a transition state structure.

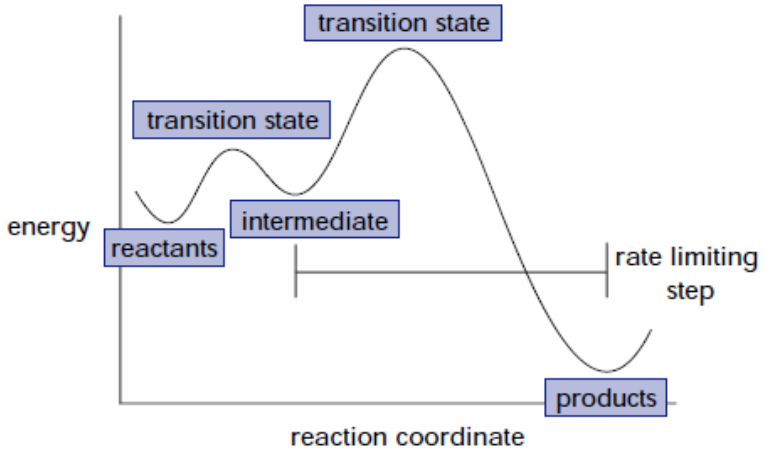
Potential Energy Surfaces



The relative energies of equilibrium structures give the relative stabilities of the reactant and product (the **thermodynamics** of reaction).

The energy of the transition state relative to the equilibrium structures provides information about the relative difficulties going on between them (the **kinetics** of reaction).

A complete reaction pathway may comprise several steps and involve several different transition states and high-energy **reactive intermediates.**



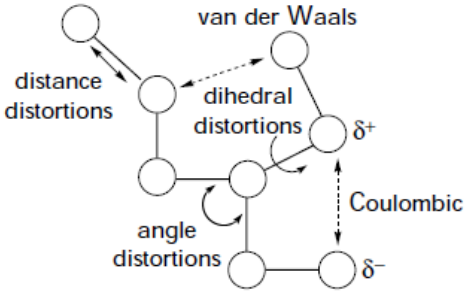
Such a diagram describes the **mechanism** of a reaction, the **rate-limiting step** for which proceeds via the highest-energy transition state.

Molecular modeling is primarily a tool for calculating the energy of a given molecular structure. Thus, the first step in designing a molecular modeling investigation is to define the problem as one involving a structure-energy relationship.

There are two conceptually different ways of thinking about energy.

Molecular Mechanics A "Chemist's" Model

Molecular mechanics describes the energy of a molecule in terms of a simple function which accounts for distortion from "ideal" bond distances and angles, as well as and for nonbonded van der Waals and Coulombic interactions.



$$E = \underbrace{\sum_{\text{bonds}} \text{bond stretching} + \sum_{\text{angles}} \text{angle bending} + \sum_{\text{dihedral angles}} \text{dihedral twisting}}_{\text{energy of distortion from idealized values}} + \underbrace{\sum_{\substack{\text{pairs of} \\ \text{atoms}}} \text{van der Waals} + \sum_{\substack{\text{pairs of} \\ \text{atoms}}} \text{Coulombic}}_{\text{energy of nonbonded interactions}}$$

Quantum Mechanics A "Physicist's" Model

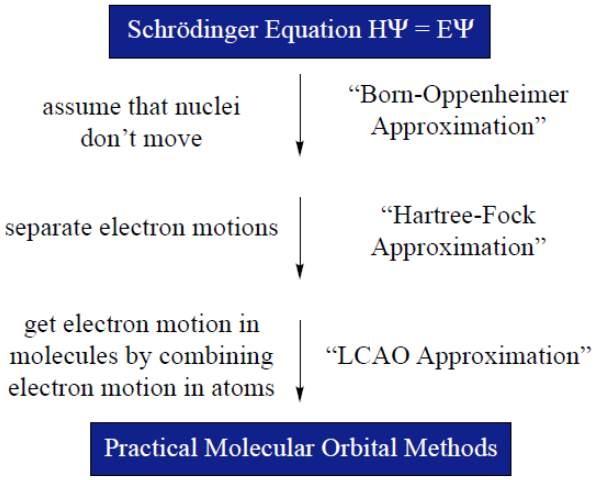
Quantum mechanics describes the energy of a molecule in terms of interactions among nuclei and electrons as given by the Schrödinger equation. The solutions ("wavefunctions") for the hydrogen atom are the familiar s, p, d . . . atomic orbitals.

$$\underbrace{\left(\frac{-h^2}{8\,^2m}\nabla^2 + \overbrace{V(x,y,z)}^{\text{potential energy}}\right)}_{\text{kinetic energy}} \underbrace{\psi(x,y,z)}_{\substack{\text{wavefunction describing} \\ \text{the system}}} = \overbrace{E}^{\text{total energy}}\,\psi(x,y,z)$$

The square of the wave function gives the probability of finding an electron. This is the electron density, as obtained from an X-ray diffraction experiment.

Quantum Mechanics

While the Schrödinger equation is easy to write down for many-electron atoms and molecules, it is impossible to solve. Approximations are needed.

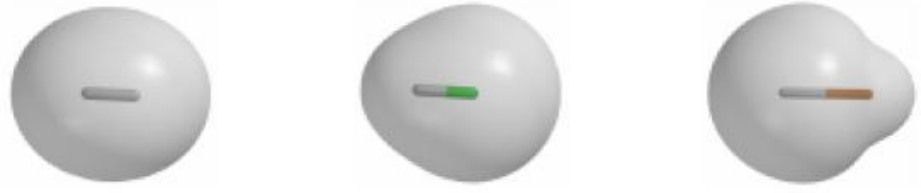


Molecular Modeling in Lecture

Molecular models can be introduced into almost any chemistry lecture. They not only liven the discussion with "pretty pictures", but more importantly encourage students to see and think for themselves. They free teachers from the "limits of the chalkboard" and allow examination and discussion of "real" molecules.

Visualizing Chemical Bonds

Are the different kinds of chemical bonds to which chemists refer (nonpolar covalent, polar covalent and ionic) fundamentally different? Look at electron densities.

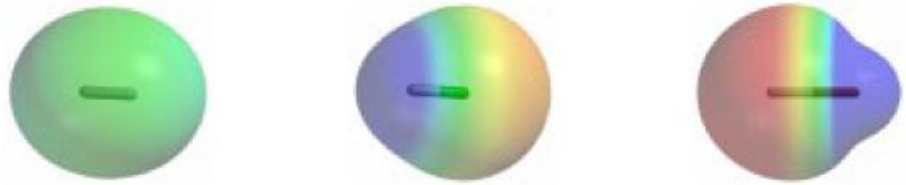


H-H H-F H-Li

The size of the electron density surface indicates the size of the electron cloud. The H electron cloud is largest in HLi and smallest in HF. This is persuasive evidence that the atoms in these molecules do not "share" electrons equally.

Visualizing Chemical Bonds

A clearer picture comes from electrostatic potential maps, where the color red demarks regions with excess negative charge and the color blue demarks regions with excess positive charge.

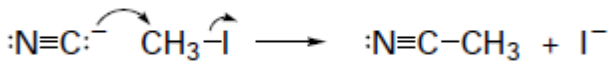


H-H H-F H-Li

Hydrogen fluoride and lithium hydride look very similar, except that the hydrogen in the former is positively charged while the hydrogen in the latter is negatively charged.

The S_N2 Reaction

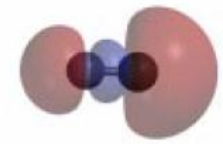
Molecular modeling provides insight into the familiar S_N2 reaction.



An animation of the reaction clearly shows the inversion at carbon, but there are other important questions.

Why does cyanide attack from carbon and not nitrogen? Doesn't this contradict the fact that nitrogen is more electronegative than carbon?

Look at the highest-occupied molecular orbital of cyanide. This is where the "most available" electrons reside.

N≡C 

It is more heavily concentrated on carbon, meaning that cyanide is a carbon nucleophile.

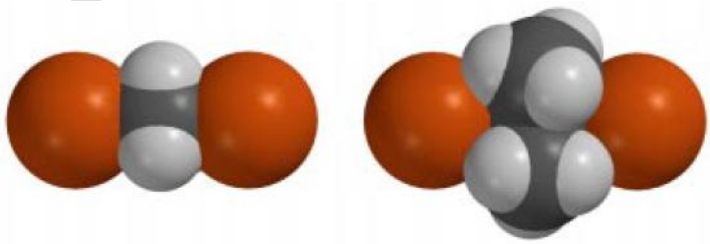
Why does iodide leave following attack by cyanide? Look at the lowest-unoccupied molecular orbital of methyl iodide. This is where the electrons will go.

It is antibonding between carbon and iodine meaning that the CI bond will cleave during attack.

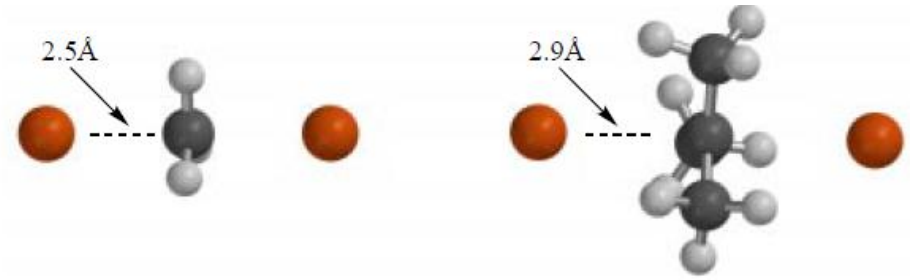
The S_N2 Reaction

We tell students that bromide reacts faster with methyl bromide than with *tert*-butyl bromide because of steric effects, that is, increased crowding of the transition state.

This is not true. Space-filling models of the two transition states show both to be uncrowded.

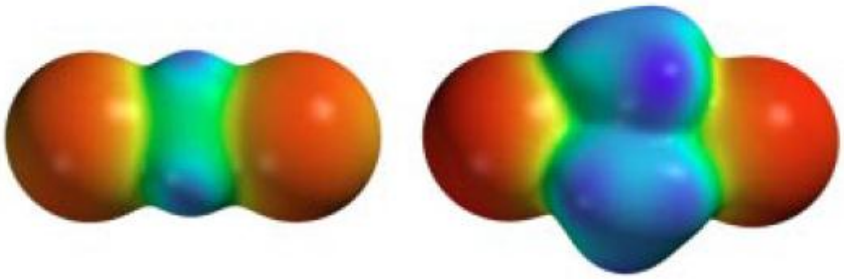


What is true is that the carbon-bromine distance in the transition state in the *tert*-butyl system is larger than that in the methyl system.



The S_N2 Reaction

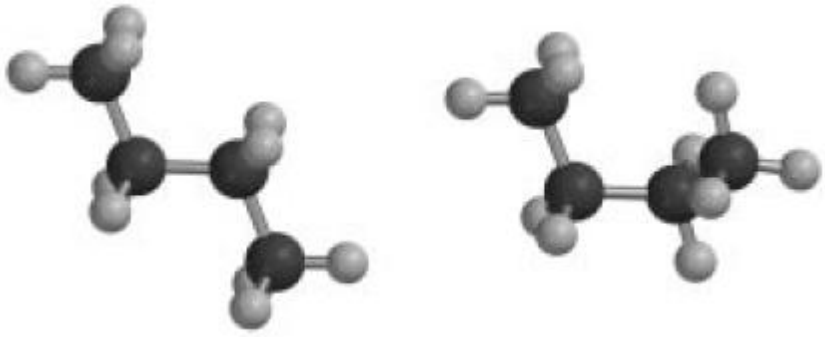
This leads to an increase in charge separation as clearly shown by electrostatic potential maps for the two transition states.



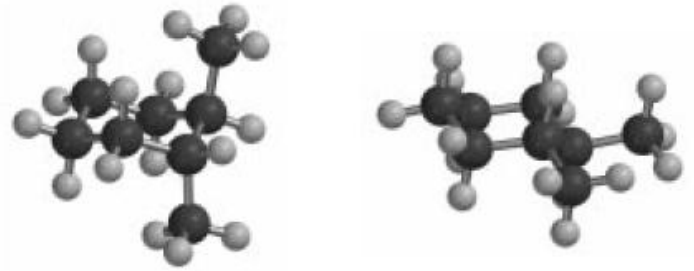
This is the cause of the decrease in reaction rate.

Flexible Molecules

Interconversion of *anti* and *gauche* forms of *n*-butane is readily visualized "on a blackboard" using conventional structure models.

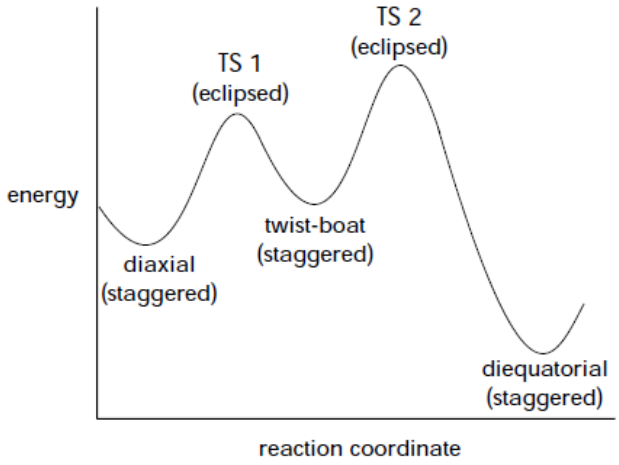


Trans-1,2-dimethylcyclohexane undergoes the same type of conformational change as *n*-butane. The difficulty for the lecturer is that this change is not easily portrayed "on a blackboard". This obstacle is readily overcome with molecular models.



diaxial diequatorial

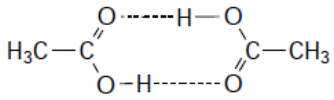
Flexible Molecules



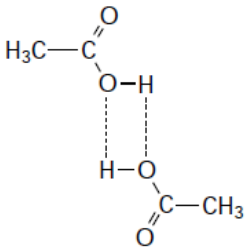
An animation shows that the conformational change in *trans*-dimethylcyclohexane is not so different from that in *n*-butane. There are two distinct steps in the ring flip mechanism, but each involves rotation about carbon-carbon bonds. The three minima all correspond to structures in which bonds are staggered. The two transition states involve eclipsing interactions.

Intermolecular Interactions

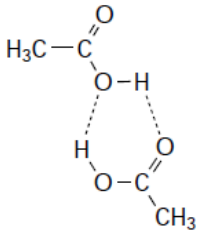
Acetic acid is known to form a stable hydrogen- bonded dimer. What is it's structure?



A



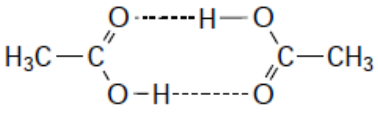
B



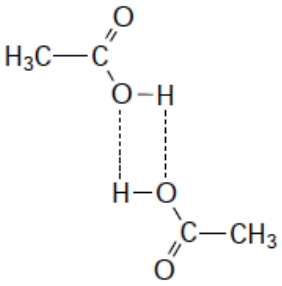
C

Instead of giving students the right structure, give them the tools to find it for themselves.

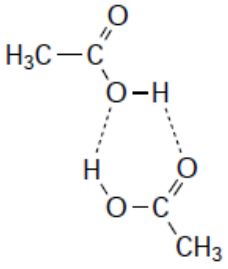
Intermolecular Interactions



A



B



C

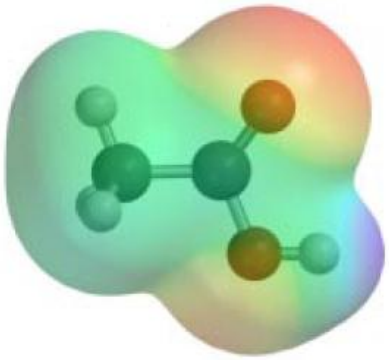
Energy Tool

Energies follow the order A<C<B (A is best). The alert student will be surprised. This is because six- membered rings (as in C) are common while eight- membered rings (as in A) are not.

Intermolecular Interactions

Electrostatic Potential Tool

Look at an electrostatic potential map for acetic acid



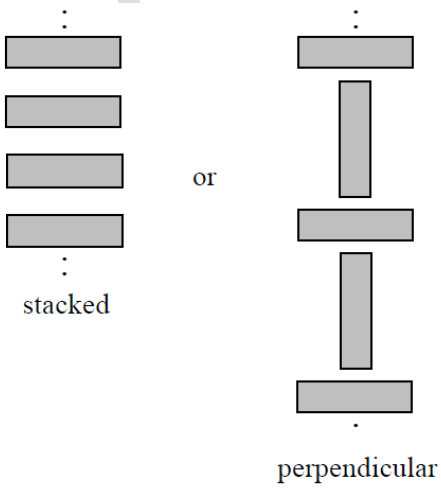
Which atom is positively charged and most likely to act as a hydrogen-bond donor?

Which atom is most negatively charged and most likely to act as a hydrogen-bond acceptor?

The model gives the answers and allows assignment of the proper dimer structure.

Apply these same "tools" to a related question where you don't know the answer (or where you "know" the wrong answer).

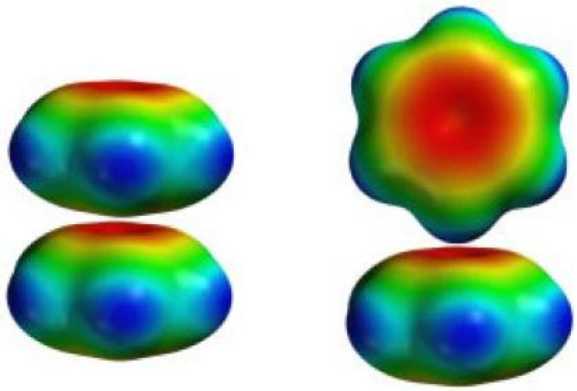
What is the crystal structure of benzene?



Intermolecular Interactions

The **energy tool** shows that the "stacked" benzene dimer dissociates into the two benzenes, while the perpendicular dimer sticks together. It does not show why.

The **electrostatic potential tool** clearly shows that stacking the rings results in unfavorable electrostatic interactions, while a perpendicular arrangement of benzene rings results in favorable electrostatic interactions between and systems.



POSSIBLE QUESTIONS

PART A (20 multiple choice questions)

Online Examinations

PART B (2 marks questions)

1. What is meant by electrostatics potential?
2. Define molecular modeling.
3. Expand HOMO and LUMO.
4. Why should molecular orbital descriptions be used instead of conventional Lewis structures?
5. Draw the structures of carbonyl addition and Michael addition in α,β-unsaturated ketone.

PART C (6 marks questions)

1. Write notes on electron densities.
2. Explain molecular orbital maps with suitable examples.
3. Give a brief account on potential energy surfaces.
4. How do you explain molecular modeling using S_N2 reaction?
5. Describe intermolecular interactions with suitable examples.
6. Explain elementary ideas of molecular mechanics.
7. Describe the roll of potential energy surfaces in molecular modeling.
8. Give an account on molecular orbital theory.

KARPAGAM ACADEMY OF HIGHER EDUCATION

Class: III B.Sc Chemistry Course Name: Applications of Computers in Chemistry

Course Code: 16CHU502B Unit: V Batch-2016-2019

1. Which of the following terms refers to the molecular modelling computational method that uses equations obeying the laws of classical physics?
a. quantum mechanics b. molecular calculations
c. molecular mechanics d. quantum theory

2. Which of the following terms refers to the molecular modelling computational method that uses quantum physics?
a. quantum mechanics b. molecular calculations
c. molecular mechanics d. quantum theory

3. Which of the following statement is true?
a. energy minimization is carried out using quantum mechanics
b. energy minimization is used to find a stable conformation for a molecule
c. energy minimization is carried out by varying only bond angles and bond lengths
d. energy minimization is stopped when a structure is formed with a much greater stability than the previous one in the processes

4. Which of the following needs to be known before two drugs can be overlaid to compare their structure?
a. the pharmacophore of the each drug
b. the active conformation of each drug
c. **pharmacophore and active conformation**
d. neither pharmacophore nor active conformation

5. Which of the following statement is true?
a. the most stable conformation of a drug is also the active conformation
b. the active conformation is the most reactive conformation of a structure
c. the active conformation is the conformation adopted by a drug when its binds to their target binding sites
d. the active conformation is can by determined by conformational analysis

6. Which of the following statement is not true of cyclic structures?
a. they are normally more rigid than acyclic structures
b. they are locked in to the active conformation
c. they are useful in determining the active conformation of a series of related compounds
d. they are normally more difficult to synthesis than acyclic compounds

7. What is meant by docking?
a. the process by which two different structures are compared by molecular modelling
b. the process by which a lead compound is simplified by removing excesses functional groups

c. the process by which drugs are fitted into their target binding sites using molecular modelling
d. the process by which a pharmacophore is identified

8. What is meant by de nova drug design?
a. the synthesis of the compound from simple starting materials
b. the design of the synthesis required to generate novel range of structures
c. the design of novel drug based on molecular modelling studies of binding site
d. the modification of a drug based on molecular modeling studies into how its binds to its target binding site

9. Which of the following statement is true in de nova drug design?
a. the design of the rigid molecules is superior to flexible ones
b. the molecules should be designed to fit as snugly as possible into the target binding site
c. molecules that have to adopt an unstable conformation in order to bind should be rejected
d. desolvation energies can be ignored since they are likely to be the same for different molecules having the same pharmacophore

10. Which of the following operations or calculations would generally be carried out using molecular mechanics?
a. Molecular orbital energies **b. Energy minimization**
c. Electrostatic potentials d. Transition-state geometries

11. Energy a system possesses because of its velocity relative to the surrounding is
a. kinetic energy b. potential Energy c. work d. electrical energy

12. A system is at rest, the kinetic energy of the system is
a. Infinite b. Greater than zero c. Less than zero d. d) Zero

13. What is the unit of Specific kinetic energy?
a. J **b. J/kg** c. Pa d. N

14. Specific Potential energy is
a. mgh **b. gh** c. h d. hg

15. PES has the same dimensionality as the number of geometric degrees of freedom of the molecule
a. 3N–4 b. 3N–5 **c. 3N–6** d. 3N–2

16. Equilibrium constant between two minima approximately determined as
a. $K = e^{-E\Delta/T}$ **a. $K = e^{-E\Delta/RT}$** a. $K = e^{E/RT}$ a. $K = e^{E\Delta/RT}$

17. Molecular mechanics predict the energy of
a. atoms b. anions c. cations **d.molecules**

18. Most C–H bond lengths are
a. 1.06 to 1.10° b. 1.05 to 1.10° c. 1.06 to 1.20° d. 1.20 to 1.10°

19. Most C–H bond lengths are 1.06 to 1.10 A° in just about any molecule, with stretching frequencies between
a. 2800-3000 cm^{-1} **b. 2900-3300 cm^{-1}** c. 3000-3300 cm^{-1} d. 2900-3100 cm^{-1}

20. Who is proposed extending spectroscopic force field ideas to doing molecular mechanics?
a. D. H. Andrews b. F. H. Westheimer c. J. B. Hendrickson d. K. B. Wiberg

21. Who is performed molecular mechanics calculation done by hand to determine the transition state of a tetrasubstituted biphenyl?
a. D. H. Andrews **b. F. H. Westheimer** c. J. B. Hendrickson d. K. B. Wiberg

22. The following one scientist who is performs conformational analysis of larger than 6 membered rings
a. D. H. Andrews b. F. H. Westheimer **c. J. B. Hendrickson** d. K. B. Wiberg

23. Who is publishes first general molecular mechanics type program with ability to find energy minimum
a. D. H. Andrews b. F. H. Westheimer c. J. B. Hendrickson **d. K. B. Wiberg**

24. The stretching potential for a bond between atoms A and B is given by the
a. Taylor series b. Fourier series c. Force field d. Transition series

25. J. B. Hendrickson performs conformational analysis of larger than
a. 5 membered rings b. 3 membered rings **c. 6 membered rings** d. 4 membered rings

26. F. H. Westheimer performed a molecular mechanics calculation done by hand to determine the transition state of a
a. tetrasubstituted benzene b. disubstituted biphenyl
c. tetrasubstituted phenyl **d. tetrasubstituted biphenyl**

27. Molecular mechanics energy equation together with the data required to describe the behavior of different kinds of atoms and bonds, is called a
a. Taylor series b. Fourier series **c. Force field** d. Transition series

28. A simple molecular mechanics energy equation is given by
a. Stretching Energy + Bending Energy + Torsion Energy + Non-Bonded Interaction Energy
b. Stretching Energy + Bending Energy + Transition energy + Non-Bonded Interaction Energy
c. Stretching Energy + Bending Energy + Torsion Energy + Bonded Interaction Energy
d. Stretching Energy + Bending Energy + Torsion Energy + kinetic Energy

29. The stretching energy equation is based on
a. Hooke's law b.Huckles law c. Beer- Lambert's law d. Beer law

30. The van der Waals energy arises from the interactions between electron clouds around
a. 4 non bonded atoms b. 3 non bonded atoms
c. 2 non bonded atoms d. 5 nonbonded atoms

31. Forces will affect the motions of the atoms tells by
a. Hooke's law **b. Newton's laws** c. Beer- **Lambert's** law d. Beer law

32. Which below mentioned bond cannot break or form during (standard) molecular dynamics simulations
a. ionic bond b. pi bond c. sigma bond **d. covalent bond**

33. Molecular mechanics energy equation together with the data required to describe the behavior of different kinds of atoms and bonds, is called a
a. Taylor series b. Fourier series **c. Force field** d. Transition series

34. Which system/mode/ applies computational procedures to solve equations
a. dynamic model b. static model c. analytical model **d. numerical model**

35. Which system/mode/ applies deductive reasoning of mathematical theory to solve a model
a. dynamic model b. static model **c. analytical model** d. numerical model

36. Which model follows the change over time that results from the system activities?
a. dynamic model b. static model c. analytical model d. numerical model

37. Motion along the reaction coordinates for the pyrolysis of ethyl formate to gives
a. formic acid + ethane b. formic acid + ethane
c. acetic acid + methane d. acetice acid + ethane

38. The energy maximum corresponds to a
a. Transition energy b. lower energy c. kinetic energy d. potential energy

39. Does cyanide attack from carbon and not nitrogen due to
a. nitrogen more electronegativity's b nitrogen low electronegativity's
c. nitrogen neutral d. carbon neutral

40. What is meant by a binding site?
a. The area of a macromolecular target that is occupied by a drug when it binds.
b. The portion of the drug to which a drug target binds.
c. The functional groups used by a drug in binding to a drug target.
d. The bonds involved in binding a drug to its target.

41. Which of the following binding interactions is likely to be the most important initial interaction when a drug enters a binding site?
a. van der Waals interactions b. hydrogen bond
c. ionic d. induced dipole-dipole interactions

42. Which of the following underlined atoms is likely to be the strongest hydrogen bond acceptor?
a. amide nitrogen (R<u>N</u>HCOR') b. aniline nitrogen (Ar<u>N</u>H_2)
c. amine nitrogen (R<u>N</u>H_2) **d. carboxylate oxygen (RC<u>O</u>$_2^-$)**

43. Which of the following underlined protons is likely to be the strongest hydrogen bond donor?
a. alcohol (RO<u>H</u>) b. amine (RN<u>H_2</u>) c. phenol (ArO<u>H</u>) **d. ammonium ion (RN<u>H_3</u>$^+$)**

44. Which of the following functional groups is most likely to participate in a dipole-dipole interaction?
a. Aromatic ring **b. Ketone** c. Alcohol d. Alkene

45. Which of the following statements is true?
a. Drugs and drug targets generally have similar molecular weights.
b. Drugs are generally smaller than drug targets.
c. Drugs are generally larger than drug targets.
d. There is no general rule regarding the relative size of drugs and their targets.

46. In which molecule is the bond the most polar?
a.HI b. HCl c. HBr **d. HF**

47. Which of the intermolecular bonding interactions below are possible for an alcohol?
a. Hydrogen bonding only b. van der Waals interactions only
c. Ionic bonding only c. Both hydrogen bonding and ionic bonding

48. Which of the intermolecular bonding interactions below are possible for a primary amine?
a. Hydrogen bonding only b. van der Waals interactions only
c. Ionic bonding only **c. Both hydrogen bonding and ionic bonding**

49. Which of the intermolecular bonding interactions below are possible for an alkene?
a. Hydrogen bonding only **b. van der Waals interactions only**
c. Ionic bonding only c. Both hydrogen bonding and ionic bonding

50. Which of the intermolecular bonding interactions below are possible for a secondary amide?
a. Hydrogen bonding only b. van der Waals interactions only
c. Ionic bonding only c. Both hydrogen bonding and ionic bonding

51. Which of the following major aims in drug design is not related to the pharmacodynamics of a drug?
a. The reduction of side effects b. The maximisation of activity
c. The reduction of toxicity **d. The maximisation of oral bioavailability**

52. What is meant by docking?
a. the process by which two different structures are compared by molecular modelling
b. the process by which a lead compound is simplified by removing excesses functional groups

c. the process by which drugs are fitted into their target binding sites using molecular modelling
d. the process by which a pharmacophore is identified

53. Specific Potential energy is
a. mgh **b. gh** c. h d. hg

54. The energy maximum corresponds to a
a. Transition energy b. lower energy c. kinetic energy d. potential energy

55. Which system/mode/ applies computational procedures to solve equations
a. dynamic model b. static model c. analytical model **d. numerical model**

57. Molecular mechanics predict the energy of
a. atoms b. anions c. cations **d.molecules**

58. The stretching potential for a bond between atoms A and B is given by the
a. Taylor series b. Fourier series c. Force field d. Transition series

59. A system is at rest, the kinetic energy of the system is
a. Infinite b. Greater than zero c. Less than zero d. d) Zero

60. J. B. Hendrickson performs conformational analysis of larger than
a. 5 membered rings b. 3 membered rings **c. 6 membered rings** d. 4 membered rings

[16CHU502B] Reg. No----------

Karpagam Academy of Higher Education

(Deemed to be University)

(Established Under Section 3 of UGC Act, 1956)

Coimbatore-21

(For the candidates admitted from 2016 onwards)

III B.Sc. Chemistry

Applications of Computer in Chemistry

Internal Test-I

Date: Maximum: 50 marks

Time: 2hrs

PART A (20 x 1 = 20 Marks)
Answer all the questions

1. An integer constant must have atleast
a. 1 digit b. 2 digit c. 3 digit d. 5 digit

2. The range of integer constant is
a. -43757 to +43756 b. -32768 to + 32767 c. -23589 to +23588 d. -14567 to +14566

3. Which of the following one is not true in integer constant?
a. 145 b. -400 c. +40 d. 425+

4. If no sign proceeds an integer constant it is assumed to be
a. negative b. positive and negative c. positive d. neutral

5. Which of the following one is true in integer constant?
a. 995, 72 b. 6.7.8 c. 234V6 d. +40

6. The GW-BASIC was developed by
a. John Kemeny b. Kenneth Kurtz c. Bill Gates d. SatyaNadella

7. The presence of pulse is referesented as
a. 1 b. 3 c. 2 d. 0

8. Binary comes from the word of
a. Latin b. English c. Spanish d. Greek

9. Computer Engineer call a single 0 or 1 a
a. binary b. bytes c. bit d. constants

10. Graphics are very important part of
a. BASICA b. turbo BASIC c. GW-BASIC d. Q-BASIC

11. The capacity of a computer's memory is measured in byte "K" usually stand for
a. 500 b. 1000 c. 2000 d. 100

12. The maximum length of a string constant can be
a. 154 character b. 254 character c. 454 character d. 254 character

13. Range of real constants expressed in exponential form is
a. 10^{-38} to 10^{+38} b. 10^{-48} to 10^{+48} c. 10^{-28} to 10^{+28} d. 10^{-18} to 10^{+18}

14. In real constant the mantissa part and exponential part should be separated by a letter
a. B b. C c. A d. E

15. A matrix whose rows and columns are equal is known as
a. row matrix b. constant matrix c. square matrix d. zero matrix

16. A root of a function f, from the real numbers to real numbers is a number x such that
a. $f(x) = 0$ b. $f(x)$ = infinity c. $f(x)$ = imaginary number d. $f(x) = 1$

17. The method consisting of defining an auxiliary function, which is applied to the last computed approximation of a root for getting a new approximation.
a. Newton method b. Iterative method c. bisection method d. Raphson method

18. Three values define a quadratic function, which approximates the graph of the function by a parabola. This is
a. Newton method b. Iterative method c. bisection method d. Muller's method

19. For solving an equation Regula falsi method is a
a. Interpolation method b. approximation method
c. Iterative method d. graphical method

20. Replacing the derivative in Newton's method with a finite difference, we get
a. bisection method b. Secant method c. Muller's method d. approximation method

PART B (3 x 2 = 6 marks)
Answer all the questions

21. What do you mean by variables?

22. What are valid string constants?

23. What is the disadvantage of bisection method in solving quadratic equations?

PART C (3 x 8 = 24 marks)

Answer all the questions

24. a. Explain the different types of basic constants.

Or

b.Write notes on BASIC language.

25. a. Explain the following

(i) Integer variable name and (ii) Real variable name

Or

b.Illustrate about matrix addition and multiplication.

26.a. Explain the Bisection method for solving a quadratic equations

Or

b. How a quadratic equation is solved by Newton-Raphson method.

[16CHU502B] **Reg. No----------**

Karpagam Academy of Higher Education

(Deemed to be University)

(Established Under Section 3 of UGC Act, 1956)

Coimbatore-21

(For the candidates admitted from 2016 onwards)

II B.Sc. Chemistry

Applications of Computer in Chemistry

Internal Test-I

Date: Maximum: 50 marks

Time: 2hrs

PART A (20 x 1 = 20 Marks)
Answer all the questions

a. 1 digit

2.**b. -32768 to + 32767**

3.**d. 425+**

4.**c. positive**

5.**d. +40**

6. **c. Bill Gates**

7. **a. 1**

8. **a. Latin**

9. **c. bit**

10. **d. Q-BASIC**

11. **b. 1000**

12. **d. 254 character**

13. **a. 10^{-38} to 10^{+38}**

14.**d. E**

15. **c. square matrix**

16. a. f(x) = 0

17. d. Raphson method

18.c. bisection method

19. a. interpolation method

20. c. Mullers method

PART B (3 x 2 = 6 marks)
Answer all the questions

21. What do you mean by variables?

A variable is something that may change in value

22. What are valid string constants?

A character string, a string constant consists of a sequence of characters enclosed in double quotes. A string constant may consist of any combination of digits, letters, escaped sequences and spaces. Note that a character constant ʼAʼ and the corresponding single character string constant "A" are not equivalent.

23. What is the disadvantage of bisection method in solving quadratic equations?

The bisection method converges very slowly.

The bisection method cannot detect multiple roots.

PART C (3 x 8 = 24 marks)
Answer all the questions

24. a. Explain the different types of basic constants.

The value of a constant cannot be changed during execution of the program, neither by the programmer nor by the computer. The character 'A' is a constant having numerical value equal to 65 in decimal number system.

Integer Constants

An integer constant is a sequence of digits from 0 to 9 without decimal points or fractional part or any other symbols. There are 3 types of integers namely decimal integer, octal integers and hexadecimal integer.

Decimal Integers consists of a set of digits 0 to 9 preceded by an optional + or - sign. Spaces, commas and non digit characters are not permitted between digits. Example for valid decimal integer constants are int y=123; //here 123 is a decimal integer constant

Octal Integers constant consists of any combination of digits from 0 through 7 with a O at the beginning. Some examples of octal integers are
int X=O123; // here 0123 is a octal integer constant .

Hexadecimal integer constant is preceded by OX or Ox, they may contain alphabets from A to F or a to f. The alphabets A to F refers to 10 to 15 in decimal digits. Example of valid hexadecimal integers are

int x=Ox12 // here Ox12 is a Hexa-Decimal integer constant

Real Constants

Real Constants consists of a fractional part in their representation. Integer constants are inadequate to represent quantities that vary continuously. These quantities are represented by numbers containing fractional parts like 26.082. Example of real constants are

float x = 6.3; //here 6.3 is a double constant.
float y = 6.3f; //here 6.3f is a float constant.
float z = 6.3 e + 2; //here 6.3 e + 2 is a exponential constant.
float s = 6.3L ; //here 6.3L is a long double constant

Real Numbers can also be represented by exponential notation. The general form for exponential notation is mantissa exponent. The mantissa is either a real number expressed in decimal notation or an integer. The exponent is an integer number with an optional plus or minus sign.

Or

b. Write notes on BASIC language.

To facilitate the definition of abstract data types, the programming language Simula (1967) introduced a concept called a class. (The word class is not important in itself, except that it has stuck and so is now a standard term in the field.) A class is a “type definition” consisting of two parts: 1. How the data is stored in this type. 2. What functions are available to manipulate this data. For instance, we might represent a stack of integers by the following class definition: class stack { int values[100]; /* values are stored in an array */ int tos = 0; /* top of stack, initialize to

0 */ push (int i, ...){ /* push i onto stack */ values[tos] = i; tos = tos+1; /* Should check that tos < 100!! */ } int pop (...){ /* pop top of stack and return it */ tos = tos - 1; /* Should check that tos > 0!! */ return values[tos];

bool is_empty (...){ /* is the stack empty? */ return (tos == 0); /* yes iff tos is 0 */ } } At some places in this hypothetical definition, the function parameters have been left unspecified and denoted ". . . ". We will fill in these blanks shortly. We can use the class name as a new type in the language and define variables of this type elsewhere in the program, as follows:

stack s,t; These definitions only provide the capability for s and t to denote stacks. No storage is allocated at this point. They correspond to a pointer declaration of the type int *p in C, which gives p the ability to point to a memory location holding an integer but does not, in itself, allocate a valid memory location for p to point to. In order to get a "real" stack, we have to ask for it. For instance, we might say: s = new stack; t = new stack; This generates two independent stacks, one named s and the other t. If, instead, we had written s = new stack; t = s; we get a single stack with two names, s and t, either of which can be used to access the stack. The class definition only provides a template for a datatype. The operation new generates an instance of the template. The word object in object-oriented programming is synonymous with the phrase instance of a class. How do we manipulate the objects s and t? Suppose we want to push the value 7 onto the stack s. We do this by writing: s.push(7); Note the difference in style with a conventional language. In C or Haskell, push would take two arguments, the stack itself and the value to be pushed. In the object-oriented framework, each instance of the class has its "own" copy of these functions and is thus an implicit argument to the function.

25. a. Explain the following
(i) Integer variable name and (ii) Real variable name
Variables

A C variable can be declared as follows

int x = 10;

This defines a variable x of type int (32-bits) and assigns 10 as the initial value. It is possible to find both the value and a pointer to (address of) the variable x. The value and the address of the variable can be found using:

printf("The value is %d and the address is %x \n", x, &x);

The format statements %d (decimal) and %x(hexadecimal) are used to format the output. A variable declared as int x is called an automatic variables. Automatic variables are not initialized,

and given a place in the runtime stack, and it is the programmer's responsibility to initialize the variables. When a variable goes out of scope, the memory is given back. In addition to automatic storage class, C variables can be declared to be static, extern or register variables.

Storage Class Specifiers

Auto, Static, Extern and Register Variables

Any variable defined inside a function or file are considered to be an auto variable unless specified otherwise. That is, the scope of the variable is within the function or file it is declared.

For example

#includeint n;

file scope below this line

```
#include <stdio.h>
int n;      ← file scope below this line

int main() {
  int n;    ←main function scope only

   int foo( ) {
     int n;   ←foo function scope only
   }
```

Let us look at the scope of the variables declared in this file. The first **int n** that is declared just below the #include statement can be seen by any function below it. In otherwords, the location of the variable determines where the variable is meaningful. Anyfunction declared above first **int n** will not be able to see the variable n. The variables ndefined inside functions main and foo are only meaningful within the functions.

Static Variables

Static variables can be declared externally, that is, outside of any function, or internally, that is inside a function. An external static variable declared outside of functions isvisible to users of the file, but not to the functions in other files. An internal staticvariable declared inside a function

retains its value during all function call to thefunction. For example, consider the file program. That contains the following declarations.

```
#include <stdio.h>
static int n;     ← external static variable – not visible to any other file

int main() {
  foo();
  foo();
}

int foo( ) {
  static int n;   ←internal static variable – retains its value during multiple calls to foo
}
```

Functions can also be declared static, making them visible only to the source file whereit is declared. Normally functions are visible to any part of the program. But by makingthem static, we can limit the scope of the function only to the source file where it isdeclared.

Extern Variables

Extern variables can be used to share value of a variable among many functions. Typically, all variables declared inside a function are local variables and they are createdwhen function is invoked and released when function is exited. However, if we are toshare the same variable among many functions, then we can define the variable to be“extern” as follows.

```
int n;  ← allocates storage for the external variable

int foo ( ) {
   extern int n;   ← refers to the n defined above
}

int foo2( ) {
  extern int n;  ← refers to the same n as above
}
```

It should be noted that the use of “extern” inside the function can be avoided if all functions using n are defined below the external definition of n. The use of extern is more evident when “extern” variables can be shared across multiplefiles. For example, if program.c defines an external variable n, then program2.c andprogram3.c can refer to the value of the external variable by using the qualifier “extern”.Extern variables therefore are useful for sharing variables among several files.

Register Variables

Some variables that may be accessed by the program frequently can be specified to be a“register” variable.

registerint x;

This will request the compiler to consider allocating a register location, a fast access memory location, for the variable x. However, compiler may completely ignore this request. The use of a register variable depends on hardware restrictions such as numberof registers available etc. If a register cannot be allocated for a particular variable, thenthe directive is ignored. However, the address of a register variable cannot be accessedregardless of whether it is placed in a register or not.

Or

b. Illustrate about matrix addition and multiplication.

Matrix addition in C: C program to add two matrices, i.e., compute the sum of two matrices and then print it. Firstly a user will be asked to enter the order of matrix (number of rows and columns) and then two matrices. For example, if a user input order as 2, 2, i.e., two rows and two columns and matrices as

First matrix:

1 2

3 4

Second matrix:

4 5

-1 5

then the output of the program (Summation of the two matrices) is:

5 7

2 9

Matrices are frequently used in programming to represent graph data structure, in solving equations and in many other ways.

Addition of two matrix in C

C program for matrix addition:

```
#include <stdio.h>
 int main()
{
int m, n, c, d, first[10][10], second[10][10], sum[10][10];
 printf("Enter the number of rows and columns of matrix\n");
scanf("%d%d",&m,&n);
printf("Enter the elements of first matrix\n");
 for(c =0; c < m;c++)
for(d =0; d < n; d++)
scanf("%d",&first[c][d]);
 printf("Enter the elements of second matrix\n");
 for(c =0; c < m;c++)
for(d =0; d < n; d++)
scanf("%d",&second[c][d]);
 printf("Sum of entered matrices:-\n");
 for(c =0; c < m;c++){
for(d =0; d < n; d++){
sum[c][d]= first[c][d]+ second[c][d];
printf("%d\t", sum[c][d]);
}
printf("\n");
}
 return0;
}
```

Matrix multiplication in C language: C program to multiply two matrices (two-dimensional array) which will be entered by a user. The user will enter the order of a matrix and then its elements and similarly input the second matrix. If the entered orders of two matrices are such that they can't be multiplied by each other, then an error message is displayed on the screen. You may have studied the method to multiply matrices in Mathematics.

Matrix multiplication in C language

```
#include <stdio.h>

int main()
{
int m, n, p, q, c, d, k, sum =0;
int first[10][10], second[10][10], multiply[10][10];

printf("Enter number of rows and columns of first matrix\n");
scanf("%d%d",&m,&n);
printf("Enter elements of first matrix\n");

for(c =0; c < m;c++)
for(d =0; d < n; d++)
scanf("%d",&first[c][d]);

printf("Enter number of rows and columns of second matrix\n");
scanf("%d%d",&p,&q);

if(n != p)
printf("The matrices can't be multiplied with each other.\n");
else
{
printf("Enter elements of second matrix\n");
```

```
for(c =0; c < p;c++)
for(d =0; d < q; d++)
scanf("%d",&second[c][d]);

for(c =0; c < m;c++){
for(d =0; d < q; d++){
for(k =0; k < p; k++){
sum= sum + first[c][k]*second[k][d];
}
 multiply[c][d]= sum;
sum=0;
}
}
 printf("Product of the matrices:\n");
 for(c =0; c < m;c++){
for(d =0; d < q; d++)
printf("%d\t", multiply[c][d]);
 printf("\n");
}
}
 return0;
}
```

26. a. Explain the Bisection method for solving a quadratic equation.

Bisection method is a method provides practical method to find roots of equation. This method also helps to prove the intermediate theorem. Among all the numeral methods bisection method is the simplest one to solve the transcendental equations.

This method helps to find the zero of a function by repeatedly halving the selected interval. The bisection method is a straightforward technique for finding numerical solutions to equations in

one unknown.

It works by narrowing the gap between pos and neg until it closes in on the correct answer. It narrow the gap by taking average of pos and neg. The average may be positive or negative. It is slow compared with other numerical techniques. In this section we will be dealing with bisection method with some solved problems.

Definition

In mathematics, bisection method used to find the roots of an equation. It separate intervals and select a sub-interval in which root of the equation lies. It is simple and also relatively slow method. It is based on the intermediate theorem for continuous functions. This is also called as root finding method or binary search method, dichotomy method or the interval halving method. Consider a continuous function g which is defined on closed interval [c, d] is given with g(c) and g(d) of different sign. Then by intermediate theorem, there exists a point m belongs to (c, d) for whichg(m)=0.

When we find more than one root in the selected interval, for simplicity we assume that the root in the selected interval is unique. In this case m is a unique root of function. **Let us see how this method is different from intermediate value theorem:**

Bisection method is the one of the applications of the intermediate theorem. Intermediate value theorem states that: if a function defined and continuous on a closed interval, say [m, n], then there exist a number between [m, n], say t. The function has at least one solution t in the open interval (m, n).

Whereas bisection method find an approximation to a zero of a continuous function. In case of IVT function could jump over some values in the interior of the interval, which possibility arises if the function is discontinuous at both endpoints.

For a continuous function g(x)

Step 1: Find two points, say m and n st $m < n$ and $g(m) * g(n) < 0$

Step 2: Find the midpoint of m and n, say t.

Step 3: t is root of function if $g(t) = 0$, else follow the next step.

Step 4: Divide the interval [m, n]. If g(t) * g(n) < 0, let m = t, else if g(t) * g(m) < 0 then let n = t.

Step 5: Repeat above two steps until g(t) = 0.

Example 1: Find the root of the polynomial, g(x) = x33 - 5 + 3x using bisection method. Where m = 1 and n = 2?

Solution: First find the value of g(x) at m = 1 and n = 2

g(1) = 133 - 5 + 3*1 = -1 < 0

g(2) = 233 - 5 + 3*2 = 9 > 0

Since function is continuous, its root lies in the interval [1, 2].

Let t be the average of the interval i.e t = 1+221+22 = 1.5

The value of the function at t is

g(1.5) = (1.5)33 - 5 + 3*(1.5) = 2.875

As g(t) is negative so n = 2 is replaced with t = 1.5 for the next iteration. Make sure that g(m) and g(n) have opposite signs.

Or

b. How a quadratic equation is solved by Newton-Raphson method.

When the derivative of $f(x)$ is of the simple form, the real root (non-repeated) of the equation $f(x) = 0$, can be computed rapidly by a process known as the **Newton Raphson method.** Usually the problem is to find a recurrence relation which enables us to find out a sequence $\{x_n\}$ converging to the desired root α.

Let x_0 be an approximation of the root of $f(x) = 0$, whose real root is α. Thus, $\alpha = x_0 + h$, where h is the correction (small) to be applied to x_0 to give the exact value of the root. Therefore,

$$f(\alpha) = f(x_o + h) = 0$$

By Taylor series expansion we get,

$$f(x_o) + hf'(x_o) + \frac{h^2}{2!}f''(x_o) + \cdots = 0$$

Since h is small, (neglecting higher orders of $h, i.e., h^2, h^3, etc.$) we get,

$$f(x_0) + hf'(x_0) \approx 0 \quad \Rightarrow h = -\frac{f(x_0)}{f'(x_{o)}}$$

Substituting this value of h in $\alpha = x_0 + h$, we get a better approximation to the root a of $f(x) = 0$ as

$$x_1 = x_o - \frac{f(x_o)}{f'(x_o)}$$

Therefore, the successive approximations are

$$x_2 = x_1 - \frac{f(x_1)}{f'(x_1)}$$

$$x_3 = x_2 - \frac{f(x_2)}{f'(x_2)}$$

.....

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

This formula is known as the iteration formula for Newton Raphson Method.

[16CHU502B] Reg. No----------

Karpagam Academy of Higher Education

(Deemed to be University)

(Established Under Section 3 of UGC Act, 1956)

Coimbatore-21

(For the candidates admitted from 2016 onwards)

III B.Sc. Chemistry

Applications of Computer in Chemistry

Internal Test-II

Date: 10/9/18 AN Maximum: 50 marks

Time: 2hrs

PART A (20 x 1 = 20 Marks)
Answer all the questions

1. The convergence of bisection method is -------.
a. linear b. quadratic c. slow d. fast

2. The order of convergence of Regulafalsi method may be assumed to ----------.
a.1 b. 1.618 c. 0 d. 0.5

3. The order of error in Trapezoidal rule is ----------.
a. h b. h3 c. h2 d. h4

4. The general quadratic formula for equidistant ordinates is _____
a. raphson b. newton-cote's c. interpolation d. divide difference

5. What is the order of the error in Simpson's formula?
a. four b. three c. two d. one

6. Use trapezoidal rule for y(x) _____
a. linear . b. second degree c. third degree d. degree n

7. The bisection method is simple but ----------.
a. slowly divergent b. fast convergent c.slowly convergent d. divergent

8. In simpson's 1/3, the number of ordinates must be _____
a. any integer b.odd c. prime d. even

9. The number of interval is multiple of six _____
a. simpson's 1/3 b. simphson's 3/8 c. weddle d. trapezoidal

10. The differentiation of log x is
a. 1/x b. e(x) c. sinx d. cosx

11. $\int f(x)\,dx$ of (a, b) is _____
a. f(a) b. f(b) c. f(a+b) d. f(b)-f(a)

12. The rate of convergence in Newton-Raphson method is of order _____
a. 1 b. 2 c. 3 d. 4

13. Differentiation of sinx is _____
a. cosx b. tanx c. sinx d. logx

14. Integration of cosx _____
a. cosx b. tanx c. sinx d. logx

15. If y(x) is linear then use _____
a. simphson's 3/8 b. simphson's 1/3 c. trapezoidal d. taylor series

16. The differentiation of secx is _____
a. secx tanx b. cotx c. cosecx d. tanx

17. The Newton – Raphson method is also known as method of
a. secant b. tangent c. iteration d. interpolation

18. If the derivative of f(x) = 0, then _____ method should be used.
a. Newton –Raphson b. Regula-Falsi c. Iteration d. Interpolation

19. By putting n = 2 in Newton cote's formula we get ---------- rule.
a. Simpson's 1/3 b. Simpson's ⅜ c. Trapezoidal d. Romberg

20. The Newton Cote's formula is also known as ---------- formula.
a. Simpson's 1/3 b. Simpson's 3/8 c. Trapezoidal d.Quadrature

PART B (3 x 2 = 6 marks)
Answer all the questions

21. Define Bessel's law.

22. State numerical integration.

23. What are probability distributions?

40 - Copies

PART C (3 x 8 = 24 marks)
Answer all the questions

24. a. Explain the Iterative method of non-linear equation.

Or

b. Write a note on Regula-Falsi method

25. a. Describe the Trapezoidal rule for performing numerical integration.

Or

b. Describe numerical differentiation by Newtons forwarded law and Stirling's law

26. a. Explain Simpson's rule for performing numerical integration.

Or

b. Derive a general Quadrature formula for equidistant ordinates.

[16CHU502B] **Reg. No----------**

Karpagam Academy of Higher Education

(Deemed to be University)

(Established Under Section 3 of UGC Act, 1956)

Coimbatore-21

(For the candidates admitted from 2016 onwards)

III B.Sc. Chemistry

Applications of Computer in Chemistry

Internal Test-II

Date: Maximum: 50 marks

Time: 2 hrs

PART A (20 x 1 = 20 Marks)
Answer all the questions

1. **c. slow**

2. **b. 1.618**

3. **c. h2**

4. **b. newton-cote's**

5. **a. four**

6. **a. linear**

7. **c. slowly convergent**

8. **b. odd**

9. **c. weddle**

10. **a. 1/x**

11. **a. small**

12. **b. 2**

13. **a. cosx**

14. **c. sinx**

15. **c. trapezoidal**

16. **a. secx tanx**

17. **b. tangent**

18. **b. Regula-Falsi**

19. **a. Simpson's 1/3**

20. d. **Quadrature**

PART B (3 x 2 = 6 marks)
Answer all the questions

21. Define Bessel's law.

The Bessel function of order ν∈Cν∈C can be defined, when νν is *not* a negative integer.

22. State numerical integration

Numerical integration is the approximate computation of anintegral using numerical techniques. The numericalcomputation of an integral is sometimes called quadrature.

23. What are probability distributions?

A function of a discrete variable whose integral over any interval is the probability that the variate specified by it will lie within that interval.

PART C (3 x 8 = 24 marks)
Answer all the questions

24. a. Explain the Iterative method of non-linear equation.

The roots of a function $f(x)$ are defined as the values for which the value of the function becomes equal to zero. So, finding the roots of $f(x)$ means solving the equation

$$f(x) = 0.$$

Example 1: If $f(x) = ax^2+bx+c$ is a quadratic polynomial, the roots are given by the well-known formula

$$x_1, x_2 = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Example 2: For a polynomial of degree 3 or higher, it is sometimes (but not very often!) possible to find the roots by factorising the polynomial

$f(x) = x^3 - 6x^2 + 11x - 6 = (x-1)(x-2)(x-3)$ so the roots are 1, 2 and 3
$f(x) = x^4 - 16 = (x^2 - 4)(x^2 + 4)$ so the roots are 2 and -2

For a large number of problems it is, however, not possible to find exact values for the roots of the function so we have to find approximations instead.

The function f(x) of the equation will usually have at least one continuous derivative, and often we will have some estimate of the root that is being sought. By using this information, most numerical methods for compute a sequence of increasingly accurate estimates of the root. These methods are called iteration methods.

Or

b. Write a note on Regula-Falsi method

Regula Falsi Method

Iteration no.	a_n	b_n	ω_n	$f(\omega_n)$
0	1.0000000000	2.0000000000	1.4782608747	-2.2348976135
1	1.4782608747	2.0000000000	1.6198574305	-0.5488323569
2	1.6198574305	2.0000000000	1.6517157555	-0.1169833690
3	1.6517157555	2.0000000000	1.6583764553	-0.0241659321
4	1.6583764553	2.0000000000	1.6597468853	-0.0049594725
5	1.6597468853	2.0000000000	1.6600278616	-0.0010169938
6	1.6600278616	2.0000000000	1.6600854397	-0.0002089010

7	1.6600854397	2.0000000000	1.6600972414	-0.0000432589
8	1.6600972414	2.0000000000	1.6600997448	-0.0000081223

25. a. Describe the Trapezoidal rule for performing numerical integration.

The midpoint method is based on a very simple polynomial approximation to the function *f* to be integrated on each subinterval; we simply use a constant approximation by interpolating the function value at the middle point. We are now going to consider a natural alternative; we approximate *f* on each subinterval with the secant that interpolates *f* at both ends of the subinterval.

The situation is shown in figure 11.8a. The approximation to the integral is

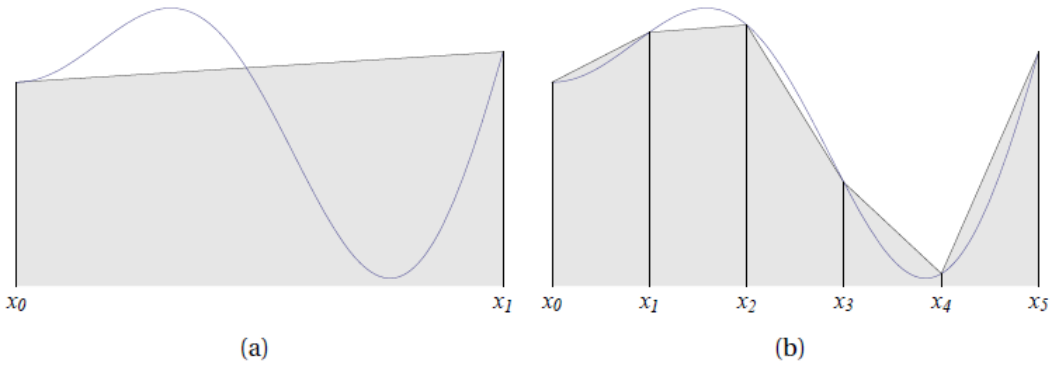


Figure 11.8. The trapezoid rule with one subinterval (a) and five subintervals (b).

the area of the trapezoidal figure under the secant so we have

$$\int_a^b f(x)\,dx \approx \frac{f(a)+f(b)}{2}(b-a). \qquad (11.39)$$

To get good accuracy, we will have to split [*a*,*b*] into subintervalswith a partition and use this approximation on each subinterval, see figure 11.8b. If we have a uniformpartition $\{x_i\}_{i=0}^{n}$ with step length *h*, we get the approximation

$$\int_a^b f(x)\,dx = \sum_{i=1}^{n} \int_{x_{i-1}}^{x_i} f(x)\,dx \approx \sum_{i=1}^{n} \frac{f(x_{i-1}) + f(x_i)}{2} h. \qquad (11.40)$$

We should always aim to make our computational methods as efficient as possible,and in this case an improvement is possible. Note that on the interval [*xi*-1,*xi*] we use the function values *f* (*xi*-1) and *f* (*xi*), and on the next interval we use the values *f* (*xi*) and *f* (*xi*=1). All function values, except the first and last, therefore occur twice in the sum on the right in (11.40). This means that if we implement this formula directly we do a lot of unnecessary work. From the explanation above the following observation follows.

(Trapezoid rule)**.** Suppose we have a function *f* defined on an interval [*a*,*b*] and a partition $\{x_i\}_{i=0}^{n}$ of [*a.b*]. If we approximate *f* by its secant on each subinterval and approximate the integral of *f* by the integral of the resulting piecewise linear approximation, we obtain the approximation

$$\int_a^b f(x)\,dx \approx h\left(\frac{f(a) + f(b)}{2} + \sum_{i=1}^{n-1} f(x_i)\right). \qquad (11.41)$$

In the formula (11.41) there are no redundant function evaluations.

Or

b. Describe numerical differentiation by Newtons forwarded law and Stirling's law

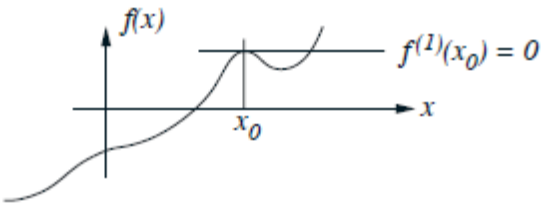
• Convergence rate for Newton's method is ***very high***!!

• Error estimates are very good (however will be case dependent on the form of the function

• Newton's method can find complex roots.

Problems with Newton's Method

• If the local min/max is selected as an initial guess



• The slope at x_0 does not intersect with *x*-axis!

• The formula for x_1 will lead to an infinite value.

The convergce process in the bisection method is very slow. It depends only on the choice of end points of the interval [a,b]. The function f(x) does not have any role in finding the point c (which is just the mid-point of a and b). It is used only to decide the next smaller interval [a,c] or [c,b]. A better approximation to c can be obtained by taking the straight line L joining the points (a,f(a)) and (b,f(b)) intersecting the x-axis. To obtain the value of c we can equate the two expressions of the slope m of the line L.

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

=> (c-b) * (f(b)-f(a)) = -(b-a) * f(b)

c = b -f(b) * (b-a) f(b) - f(a)

Now the next smaller interval which brackets the root can be obtained by checking

f(a) * f(b) < 0 then b = c

> 0 then a = c

= 0 then c is the root.

Selecting **c** by the above expression is called Regula-Falsi method or False position method.

26. a. Explain Simpson's rule for performing numerical integration.

The final method for numerical integration that we consider is *Simpson's rule*. This method is based on approximating *f* by a parabola on each subinterval, which makes the derivation a bit more involved. The error analysis is essentially the same as before, but because the expressions are more complicated, it pays off to plan the analysis better. You may therefore find the material in this section more challenging than the treatment of the other two methods, and should make sure that you have a good understanding of the error analysis for these methods before you start studying

Deriving Simpson's rule

As for the other methods, we derive Simpson's rule in the simplest case where we use one parabola on all of [*a*,*b*]. We find the polynomial *p*2 that interpolates *f* at *a*, *a*1/2 =(*a* + *b*)/2 and *b*, and approximate the integral of *f* by the integral of *p*2. We could find *p*2 via the Newton form, but in this case it is easier to use the Lagrange form. Another simplification is to first construct Simpson's rule in the case where *a* = -1, *a*1/2 = 0, and *b*= 1, and then use this to generalize the method.

The Lagrange form of the polyomial that interpolates *f* at -1, 0, 1, is given by

$$p_2(x) = f(-1)\frac{x(x-1)}{2} - f(0)(x+1)(x-1) + f(1)\frac{(x+1)x}{2},$$

and it is easy to check that the interpolation conditions hold.

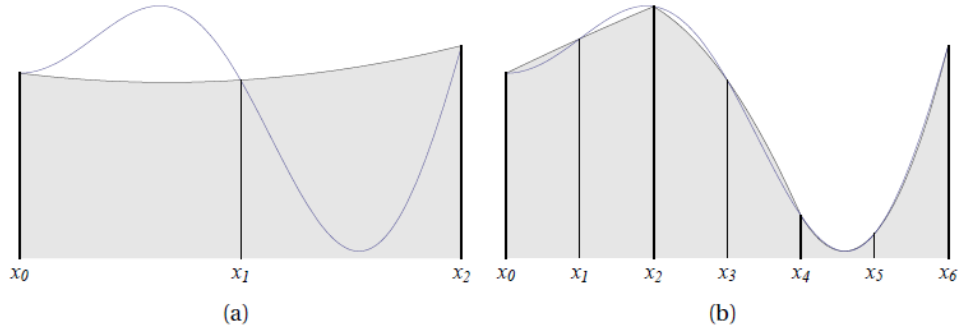


Figure 11.9. Simpson's rule with one subinterval (a) and three subintervals (b).

Observation: Let f be an integrable function on the interval $[a,b]$. If f is interpolated by a quadratic polynomial $p2$ at the points a, $(a+b)/2$ and b, then the integral of f can be approximated by the integral of $p2$,

$$\int_a^b f(x)\,dx \approx \int_a^b p_2(x)\,dx = \frac{b-a}{6}\left(f(a)+4f\left(\frac{a+b}{2}\right)+f(b)\right). \qquad (11.47)$$

We could just as well have derived this formula by doing the interpolation directly on the interval $[a,b]$, but then the algebra becomes quite messy.

Or

b. Derive a general Quadrature formula for equidistant ordinates.

[16CHU502B] Reg. No----------

Karpagam Academy of Higher Education

(Deemed to be University)

(Established Under Section 3 of UGC Act, 1956)

Coimbatore-21

(For the candidates admitted from 2016 onwards)

III B.Sc. Chemistry

Applications of Computer in Chemistry

Internal Test-III

Date: 8.10.2018 (AN) Maximum: 50 marks

Time: 2hrs

PART A (20 x 1 = 20 Marks)

Answer all the questions

1. In ---------- method, the coefficient matrix is transformed into diagonal matrix
a. Gauss elimination b. Gauss jordan c. Gauss jacobi d. Gauss seidal

2. The given system of equations can be taken as in the form of ---------
a. A = B b. BX= A c. AX= B d. AB = X

3. The Gauss – Jordan method is the modification of _____ method.
a. Gauss –Elimination b. Gauss – Jacobi c. Gauss – Seidal d. interpolation

4. The convergence of Gauss – Seidal method is roughly _____ that of Gauss – Jacobi method
a. twice b. thrice c. once d. 4 times

5. Jacobi's method is used only when the matrix is ____
a. symmetric b. skew-symmetric c. singular d. non-singular

6. Modified form of Gauss Jacobi method is ---------- method.
a. Gauss Jordan b. Gauss Siedal c. Gauss Jacobbi d. Gauss Elimination

7. In ---------- iterative method, the current values of the unknowns at each stage of iteration are used in proceeding to the next stage of iteration.
a. Gauss Siedal b. Gauss Jacobi c. Gauss Jordan d. Gauss Elimination

8. Gauss Jacobi method is a -------
a. direct method b. indirect method c. iterative method d. convergent

9. In the direct methods of solving a system of linear equations, at first the given system is written as ---------- form.
a. An augment matrix b. triangular matrix c. constant matrix d. co-efficient matrix

10. The condition for convergence of Gauss Seidal method is that they ------ should be diagonally dominant
a. constant matrix b. unknown matrix c. co-efficient matrix d. extrapolation

11. Which of the following terms refers to the molecular modelling computational method that uses equations obeying the laws of classical physics?
a. quantum mechanics b. molecular calculations
c. molecular mechanics d. quantum theory

12. Which of the following terms refers to the molecular modelling computational method that uses quantum physics?
a. quantum mechanics b. molecular calculations
c. molecular mechanics d. quantum theory

13. A system is at rest, the kinetic energy of the system is
a. infinite b. greater than zero c. less than zero d) zero

14. Which of the following needs to be known before two drugs can be overlaid to compare their structure?
a. the pharmacophore of the each drug b. the active conformation of each drug
c. pharmacophore and active conformation d. neither pharmacophore nor active conformation

15. Which of the following operations or calculations would generally be carried out using molecular mechanics?
a. molecular orbital energies b. energy minimization
c. electrostatic potentials d. transition-state geometries

16. Energy of a system possesses----------because of its velocity relative to the surrounding is
a. kinetic energy b. potential Energy c. work d. electrical energy

17. What is the unit of Specific kinetic energy?
a. J b. J/kg c. Pa d. N

18. Molecular mechanics energy equation together with the data required to describe the behavior of different kinds of atoms and bonds, is called a
a. Taylor series b. Fourier series c. Force field d. Transition series

19. PES has the same dimensionality as the number of geometric degrees of freedom of the molecule
a. 3N–4 b. 3N–5 c. 3N–6 d. 3N–2

20. Equilibrium constant between two minima approximately determined as
a. $K = e^{-EA/T}$ b. $K = e^{-EA/RT}$ c. $K = e^{E/RT}$ d. $K = e^{EA/RT}$

PART B (3 x 2 = 6 marks)
Answer all the questions

21. What is meant by square matrix?

22. Define molecular modeling.

23. What is meant by electrostatic potential?

PART C (3 x 8 = 24 marks)
Answer all the questions

24. a. Give an account on Gauss-Seidal method.

Or

b. Give a detailed account on extrapolation.

25. a. Write notes on electron densities.

Or

b. Explain elementary ideas of molecular mechanics.

26. a. Explain molecular orbital maps with suitable examples.

Or

b. Describe the roll of potential energy surfaces in molecular modeling.

[16CHU502B] **Reg. No----------**

Karpagam Academy of Higher Education

(Deemed to be University)

(Established Under Section 3 of UGC Act, 1956)

Coimbatore-21

(For the candidates admitted from 2016 onwards)

III B.Sc. Chemistry

Applications of Computer in Chemistry

Internal Test-III

Date: Maximum: 50 marks

Time: 2 hrs

Key Answer

PART A (20 x 1 = 20 Marks)

Answer all the questions

1. **b. Gauss jordan**

2. **c. AX= B**

3. **a. Gauss –Elimination**

4. **a. twice**

5. **a. symmetric**

6. **b. Gauss Siedal**

7. **a. Gauss Siedal**

8. **b. InDirect method**

9. **a. An augment matrix**

10. **c. co-efficient matrix**

11.**c. molecular mechanics**

12. **a. quantum mechanics**

13. **a. Infinite**

14. c. **pharmacophore and active conformation**

15.**b. Energy minimization**

16. **a. kinetic energy**

17. **b. J/kg**

18. **c. Force field**

19. **c. 3N–6**

20.**b. $K = e^{-E\Delta/RT}$**

PART B (3 x 2 = 6 marks)

Answer all the questions

21. What is meant by square matrix?

It should have the same number of rows as columns

22. Define molecular modeling.

Molecular modelling is a technique for deriving, representing and manipulating the structures and reactions of molecules, and those properties that are dependent on these three dimensional structures.

23. What is meant by electrostatics potential?

The electrostatic potential is the energy of interaction of a point positive charge (an electrophile) with the nuclei and electrons of a molecule.

PART C (3 x 8 = 24 marks)

Answer all the questions

24. a. Give an account on Gauss-Seidal method.

The Gauss-Seidel method (called Seidel's method by Jeffreys and Jeffreys 1988, p. 305) is a technique for solving the n equations of the linear system of equations $\mathbf{A}\mathbf{x} = \mathbf{b}$ one at a time in sequence, and uses previously computed results as soon as they are available,

$$x_i^{(k)} = \frac{b_i - \sum_{j<i} a_{ij} x_j^{(k)} - \sum_{j>i} a_{ij} x_j^{(k-1)}}{a_{ii}}.$$

There are two important characteristics of the Gauss-Seidel method should be noted. Firstly, the computations appear to be serial. Since each component of the new iterate depends upon all previously computed components, the updates cannot be done simultaneously as in the Jacobi method. Secondly, the new iterate $x^{(k)}$ depends upon the order in which the equations are examined. If this ordering is changed, the *components* of the new iterates (and not just their order) will also change.

In terms of matrices, the definition of the Gauss-Seidel method can be expressed as

$$\mathbf{x}^{(k)} = (\mathbf{D} - \mathbf{L})^{-1} \left(\mathbf{U}\mathbf{x}^{(k-1)} + \mathbf{b}\right),$$

where the matrices $\mathbf{D}$, $-\mathbf{L}$, and $-\mathbf{U}$ represent the diagonal, strictly lower triangular, and strictly upper triangular parts of $\mathbf{A}$, respectively.

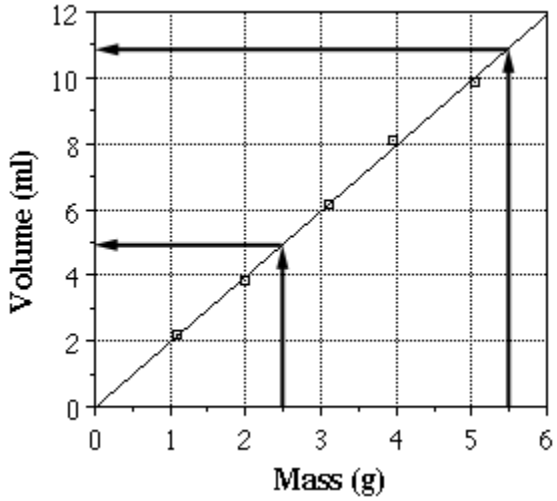
The Gauss-Seidel method is applicable to strictly diagonally dominant, or symmetric positive definite matrices $\mathbf{A}$.

Or

b. Give a detailed account on extrapolation.

Besides being able to show trends between variables, plotting data on a graph allows us to predict values for which we have taken no data. When we predict values that fall within the range of data points taken it is called interpolation. When we predict values for points outside the range of data taken it is called extrapolation. Extrapolation over too far a range can be dangerous unless it is certain that the relationship between the variables continues over the entire range.

Consider these examples based on the volume/mass data from the previous page. We could use our graph to interpolate the volume for a sample with a mass of 2.5 g. This is done by drawing a vertical line from the x-axis at a value of 2.5 g until it crosses our best fit line, and then drawing a horizontal line to the y-axis. The y-value at this point, 4.9 ml, is equal to the volume of 2.5 g sample. The same process is used for extrapolation. A sample with a mass of 5.5 g, will have a volume of 10.8 ml.



These values could also be determined using the equation for the best fit line determined previously.

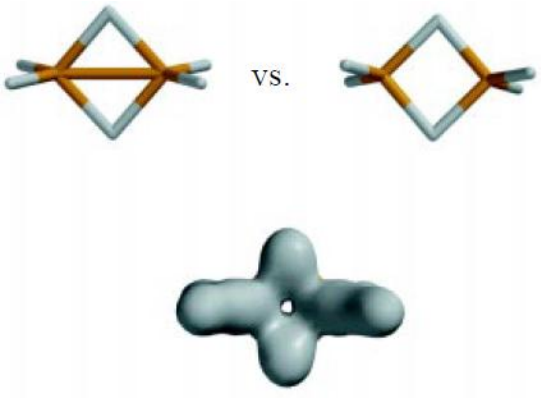
volume = 1.982 (mass) + 0.002

If 2.5 g is substituted for the mass, the calculated volume will be 4.96 ml. If a mass of 5.5 g is used, the volume will be 10.90 ml. Both of these values are close to those read off the graph.

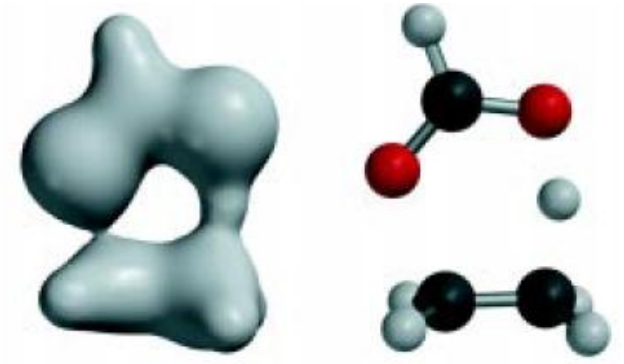
Note that while in both of the above examples we determined the volume from a given mass, the opposite could also be performed. If a sample has a volume of 7.0 ml, we could determine that its mass should be 3.53 g.

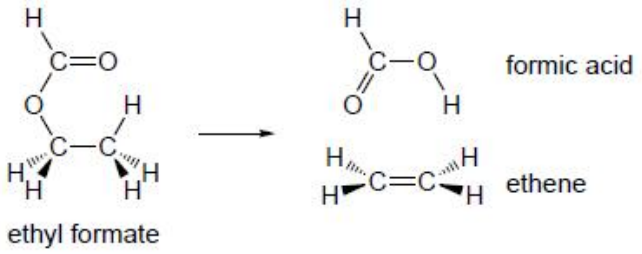
25. a. Write notes on electron densities.

Unlike conventional structure models, electron densities assume no prior knowledge about bonding and, therefore, can be used to elucidate bonding. For example, the electron density for diborane shows no evidence for boron-boron bonding.



Electron densities allow description of the bonding in transition states where there can be no (direct) information from experiment. For example, the electron density for the transition state for pyrolysis of ethyl formate, leading to formic acid and ethene, shows that the CO bond has nearly fully cleaved and that the hydrogen is midway between carbon and oxygen.

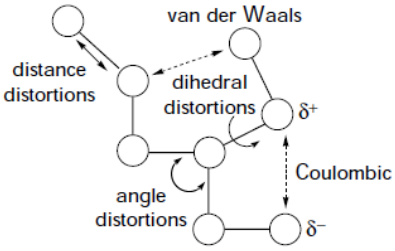




Or

b. Explain elementary ideas of molecular mechanics

Molecular mechanics describes the energy of a molecule in terms of a simple function which accounts for distortion from "ideal" bond distances and angles, as well as and for nonbonded van der Waals and Coulombic interactions.

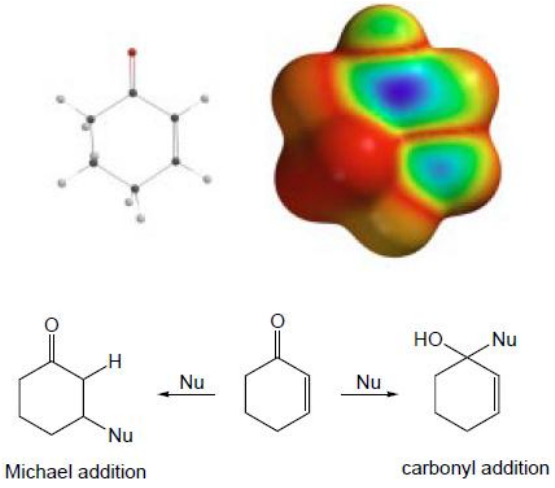


$$E = \underbrace{\sum_{\text{bonds}} \text{bond stretching} + \sum_{\text{angles}} \text{angle bending} + \sum_{\text{dihedral angles}} \text{dihedral twisting}}_{\text{energy of distortion from idealized values}}$$

$$\underbrace{+ \sum_{\substack{\text{pairs of} \\ \text{atoms}}} \text{van der Waals} + \sum_{\substack{\text{pairs of} \\ \text{atoms}}} \text{Coulombic}}_{\text{energy of nonbonded interactions}}$$

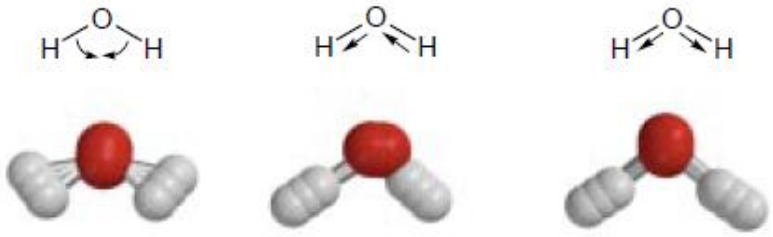
26. a. Explain molecular orbital maps with suitable examples.

Molecular orbitals may also be mapped onto electron density surfaces. For example, a map of the lowest- unoccupied molecular orbital (LUMO) of cyclohexenone, where the "blue spots" indicate maximum values of the LUMO, reveals likely sites for nucleophilic attack, and anticipates both the "carbonyl chemistry" and "Michael chemistry" known for enones.



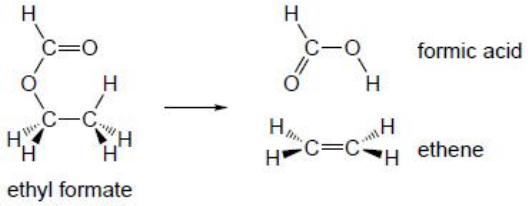
Models t h a t Move

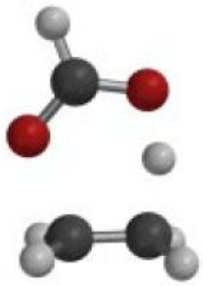
Models need not be limited to static pictures. “Movies” can be used to depict vibrations in stable molecules, for example, in water.



Models that Move

Motion along the reaction coordinate provides details about mechanism. For example, motion along the reaction coordinate for the pyrolysis of ethyl formate shows the simultaneous transfer of the hydrogen atom to the carbonyl oxygen along with carbon-oxygen bond cleavage.

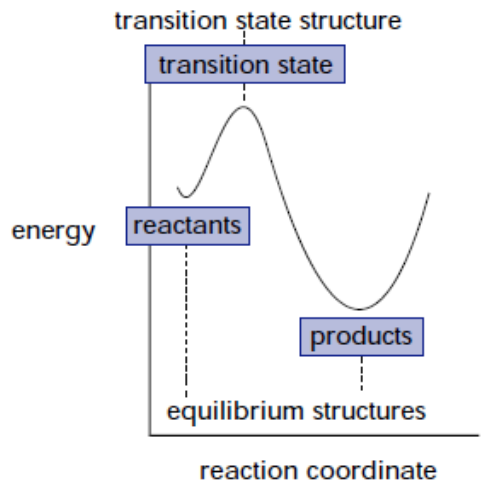




Or

b. Describe the roll of potential energy surfaces in molecular modeling.

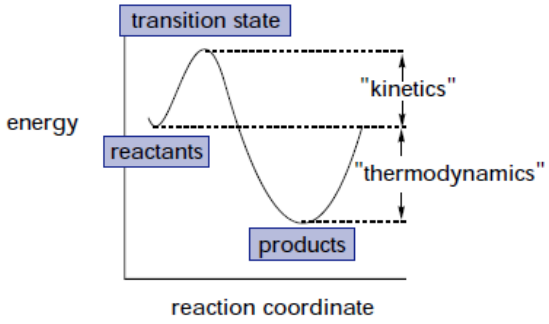
A potential energy surface is a plot of energy vs. reaction coordinate. It connects reactants to products via a transition state.



Energy minima correspond to equilibrium structures.

The energy maximum corresponds to a transition state structure.

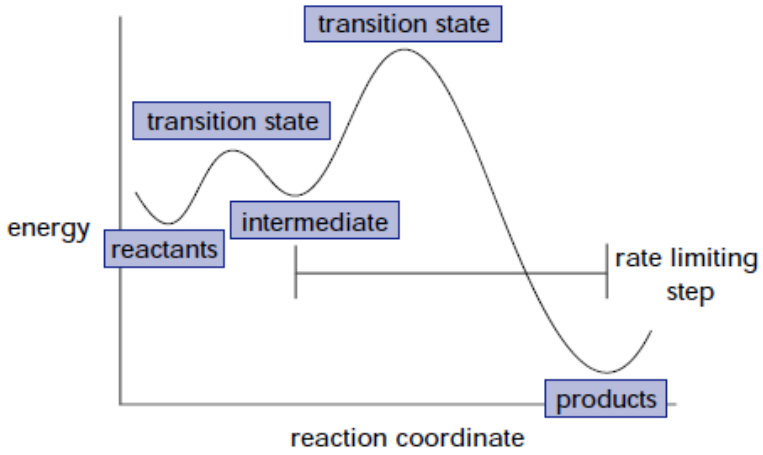
Potential Energy Surfaces



The relative energies of equilibrium structures give the relative stabilities of the reactant and product (the **thermodynamics** of reaction).

The energy of the transition state relative to the equilibrium structures provides information about the relative difficulties going on between them (the **kinetics** of reaction).

A complete reaction pathway may comprise several steps and involve several different transition states and high-energy **reactive intermediates.**



Such a diagram describes the **mechanism** of a reaction, the **rate-limiting step** for which proceeds via the highest-energy transition state.

Molecular modeling is primarily a tool for calculating the energy of a given molecular structure. Thus, the first step in designing a molecular modeling investigation is to define the problem as one involving a structure-energy relationship.

There are two conceptually different ways of thinking about energy.