BANGALORE UNIVERSITY

LABORATORY

MANUAL

FOR

MATHEMATICS PAPER-V

PRACTICALS

(WITH FOSS TOOLS)

5^{th} SEMESTER B.Sc (CBCS)

(EFFECTIVE FROM 2014-15)

DEPARTMENT OF MATHEMATICS

Central College Campus, Bangalore University

Bangalore-560001

&

Bangalore University Mathematics Teachers' forum

CONTENTS

1.Ring-theory

1. Maxima Program to verify given set and operations form a ring.
2. Maxima Program to verify given ring is commutative ring.
3. Maxima Program to verify given ring is ring with unity.
4. Maxima Program to verify given ring is with/Without zero divisors.
5. Maxima Program to verify given ring is a field.

1. Maxima Program to verify $(Z_5, \oplus_5, \otimes_5)$ form a ring.

```
kill(all)$
load(functs)$
makearray(z,5)$
z:[0,1,2,3,4]$
Z:setify(z);
addmod(x,y):=mod(x+y,5)$
multmod(x,y):=mod(x*y,5)$
n:length(z)$
s1:{}$flag1:1$
/*Closure Property wrt addition*/
for i:1 thru n do (
for j:1 thru n do (
s1:union(s1,set(addmod(z[i],z[j])))
))$
disp("s1=",s1)$
if setequalp(Z,s1) then
disp("Closure property is satisfied under addition")
else (
disp("Closure property is not satisfied under addition"),
flag1:0)$
/*Associative property wrt addition*/
for i:1 thru n do (
for j:1 thru n do (
for k:1 thru n do (
if addmod(addmod(z[i],z[j]),z[k])=addmod(z[i],addmod(z[j],z[k])) then
flag:1
else (
flag:0, break)
)))$
if flag=0 then (
disp("Associative Property fails under addition"),
flag1:0)
else
```

```
disp("Associative Property is satisfied under addition")$


/*Existance of Additive Identity*/
for i:1 thru n do (
if addmod(z[1],z[i])=z[i] and addmod(z[i],z[1])=z[i] then
flag:1
else (
flag:0, break)
)$
if flag=0 then (
disp("Additive Identity does not exist"),
flag1:0)
else
disp("Additive Identity is",z[1])$
/*Existance of Additive Inverse*/
s2:{}$
for i:1 thru n do (
for j:1 thru n do (
if addmod(z[i],z[j])=z[1] and addmod(z[j],z[i])=z[1] then (
s2:union(s2,set(z[i])),
print("Additive Inverse of ",z[i],"is",z[j]))
))$
if setequalp(Z,s2) then
disp("Inverse existence property is satisfied")
else (
disp("Inverse existence property is not satisfied"),
flag1:0)$
/*Abelian Property wrt addition*/
for i:1 thru n do (
for j:1 thru n do (
if addmod(z[i],z[j])=addmod(z[j],z[i]) then
flag:1
else (
flag:0, break)
))$
if flag=0 then (
disp("Abelian Property fails under addition"),
flag1:0)
else
disp("Abelian Property is satisfied under addition")$

/*Closure property wrt multiplication*/
for i:1 thru n do (
for j:1 thru n do (
s1:union(s1,set(multmod(z[i],z[j])))
```

```
))$
disp("s1=",s1)$
if setequalp(Z,s1) then
disp("Closure property is satisfied under multiplication")
else (
disp("Closure property is not satisfied under multiplication"),
flag1:0)$
/*Associative property wrt multiplication*/
for i:1 thru n do (
for j:1 thru n do (
for k:1 thru n do (
if multmod(multmod(z[i],z[j]),z[k])=multmod(z[i],multmod(z[j],z[k]))
then
flag:1
else (
flag:0, break)
)))$
if flag=0 then (
disp("Associative Property fails under multiplication"),
flag1:0)
else
disp("Associative Property is satisfied under multiplication")$

if flag1=1 then
disp("Given set with operations is a ring")
else
disp("Given set with operations is not a ring")$
```

2 Maxima Program to verify the ring $(Z_5,\oplus_5,\otimes_5)$ is commutative ring.

```
kill(all)$
load(functs)$
makearray(z,5)$
z:[0,1,2,3,4]$
Z:setify(z);
addmod(x,y):=mod(x+y,5)$
multmod(x,y):=mod(x*y,5)$
n:length(z)$
/*Abelian property wrt multiplication*/
for i:1 thru n do (
for j:1 thru n do (
if multmod(z[i],z[j])=multmod(z[i],z[j]) then
flag:1
else (
flag:0, break)
))$
if flag=0 then (
```

```
disp("Abelian Property fails under multiplication"),
flag1:0)
else
disp("Abelian Property is satisfied under multiplication")$
```

3 Maxima Program to verify the ring (Z_5,$\oplus_5$,$\otimes_5$) is ring with unity.

```
kill(all)$
load(functs)$
makearray(z,5)$
z:[0,1,2,3,4]$
Z:setify(z);
addmod(x,y):=mod(x+y,5)$
multmod(x,y):=mod(x*y,5)$
n:length(z)$
/*Multiplicative identity*/
flag:1$
for i:2 thru n do (
if multmod(z[2],z[i])=z[i] and multmod(z[i],z[2])=z[i] then
flag:1
else (
flag:0, break)
)$
if flag=0 then (
disp("Multiplicative Identity does not exist"),
flag1:0)
else
print("Multiplicative Identity is",1, ". Hence given ring is with
unity")$
```

4 Maxima Program to verify the ring (Z_5,$\oplus_5$,$\otimes_5$) is ring with or without zero divisors.

```
kill(all)$
z:[0,1,2,3,4]$
Z:setify(z);
multmod(x,y):=mod(x*y,5)$
n:length(z)$
flag:1$
for i:2 thru n do (
for j:2 thru n do (
if flag=0 then break else (
if multmod(z[i],z[j])#0 then
flag:1
else
```

```
flag:0)
))$
if flag=0 then
disp("Zero divisors exist")
else
disp("Zero divisors does not exist")$
```

5 Maxima Program to verify the ring $(Z_5, \oplus_5, \otimes_5)$ is a field.

```
kill(all)$load(functs)$
makearray(z,5)$z:[0,1,2,3,4]$
Z:setdifference(setify(z),{0});
addmod(x,y):=mod(x+y,5)$
multmod(x,y):=mod(x*y,5)$
n:length(z)$
flag1:1$
/*Existance of Multiplicative Inverse*/
s2:{}$
for i:2 thru n do (
for j:2 thru n do (
if multmod(z[i],z[j])=z[2] and multmod(z[j],z[i])=z[2] then (
s2:union(s2,set(z[i])),
print("Multiplicative Inverse of ",z[i],"is",z[j]))
))$
if setequalp(Z,s2) then
disp("Multiplicative Inverse exist for non zero elements", "Hence
given ring is a field")
else (
disp("Multiplicative Inverse existence property is not satisfied",
"Hence given ring is not a field"),
flag1:0)$

printf("\ninverse law holds in (z5,*)")
//to verify ring (z5,+5,*5) is a commutative ring under *
for i=2:n
    for j=2:n
        k1=modulo(z(i)*z(j),5);
        k2=modulo(z(j)*z(i),5);
        if k1~=k2 then
printf("\n commutative law fails under multiplication");
printf("z7 is not a commutaive ring");
            abort;
        end
    end
end
printf("\n(Z5,*) is a commutative ring")
printf("\n and  hence (z5,+,*) is a  field")
```

Scilab Code for Ring Theory

1.scilab Program to verify $(Z_7,\oplus_7,\otimes_7)$ form a ring.

```
//to check (z7,+7,*7) is an abelian group under addition
clc;
k=[];
z=[0 1 2 3 4 5 6]
n=length(z);
for i=1:n
    for j=1:n
      k=modulo(z(i)+z(j),7);
      if (find((z==k))==[]) then
printf("\n Z is not closed under addition");
printf("\n Z is not a ring  under multiplication");
       abort;
      end
    end
end
for i=1:n-2
     k1=modulo(z(i)+modulo(z(i+1)+z(i+2),7),7);
     k2=modulo(modulo(z(i)+z(i+1),7)+z(i+2),7);
     if k1~=k2  then
printf("\n Z7 is not Associtive under multiplication");
printf("\n Z7 is not a group under multiplication");
        abort;
    end
 end
  e=0;
 for i=1:n
     if modulo(z(i)+e,7)~=z(i) | modulo(e+z(i),7)~=z(i) then
printf("\nIdentity law doesnot hold zood");
printf("\n Z7 is not a group under multiplication");
        abort;
     end
 end
flag=0
 for i=1:n
    for j=1:n
       if modulo(z(i)+z(j),7)==e &modulo(z(j)+z(i),7)==e then
           flag=1;
           break;
        end
    end
   if flag==0 then
printf("\n inverselaw holds good in Z");
printf("\n Z7 is not a ring ");
```

```
abort;
 end
end
for i=1:n
    for j=1:n
        k1=modulo(z(i)+z(j),7);
        k2=modulo(z(j)+z(i),7);
        if k1~=k2 then
             printf("\n commutative law fails");
             printf("Z isnot an abelian group under addition");
            abort;
        end
    end
end
//to check multiplication is associative
for i=1:n-2
     k1=modulo(z(i)*modulo(z(i+1)*z(i+2),7),7);
     k2=modulo(modulo(z(i)*z(i+1),7)*z(i+2),7);
     if k1~=k2  then
      printf("\n Z7 is not Associative under multiplication");
      printf("\n Z7 is not a semi group under multiplication");
       abort;
    end
 end
//multiplication distributive under addition
//a.(b+c)=a.b+a.c and (b+c).a=b.a+c.a
for i=1:n-2
     k1=modulo(z(i)*modulo(z(i+1)+z(i+2),7),7);
     k2=modulo(z(i)*z(i+1),7); k3=modulo(z(i)*z(i+2),7);
     k=modulo(k2+k3,7);
     if k1~=k  then
      printf("\n Z7 is not distributive under multiplication");
      printf("\n Z7 is not a ring");
        abort;
    end
 end
printf("\n Z7 is ring ");
```

2.scilab Program to verify $(Z_6,\oplus_6,\otimes_6)$ form a ring.

3. scilab Program to verify ring $(Z_7,\oplus_7,\otimes_7)$ is a commutative ring.

```
//to verify ring (z7,+7,*7) is a commutative ring
clc;
clear all;
z=[0 1 2 3 4 5 6];
n=length(z);
for i=1:n
    for j=1:n
        k1=modulo(z(i)*z(j),7);
        k2=modulo(z(j)*z(i),7);
        if k1~=k2 then
             printf("\n commutative law fails under multiplication");
             printf("z7 is not a commutative ring");
abort;
        end
    end
end
printf("z7 is a commutative ring")
```

4. scilab Program to verify ring $(Z_5,\oplus_5,\otimes_5)$ is a commutative ring

5. scilab Program to verify ring $(Z_5,\oplus_5,\otimes_5)$ is a ring with unity

```
//to verify ring (z5,+5,*5) is a ring with unity
clc;
z=[0 1 2 3 4]
n=length(z);
e=1;
for i=1:n
    if modulo(z(i)*e,5)~=z(i) | modulo(e*z(i),5)~=z(i) then
printf("\nIdentity law doesnot hold good");
      printf("\n Z is not a ring with unity");
       abort;
    end
end
printf("Z5 is a ring with unity")
```

6.scilab Program to verify ring $(Z_6,\oplus_6,\otimes_6)$ is a ring with unity.

7 scilab Program to verify ring $(Z_6,\oplus_6,\otimes_6)$ is a ring with zero divisor or not.

```
//to verify ring (z6,+6,*6) is a ring with  zero divisor
clc;
z=[0 1 2 3 4 5 ]
n=length(z);
e=1;
 for i=2:n
     for j=2:n
         if modulo(z(i)*z(j),6)==0 then
      printf("z6 has a zero divisors");
            printf("\n%d and %d are zero divisors in  Z6",z(i),z(j))
            abort;
         end
     end
 end

printf("Z6 is a ring with no zero divisor")
```

8 scilab Program to verify ring $(Z_5,\oplus_5,\otimes_5)$ is a ring with zero divisor or not

9. scilab Program to verify ring $(Z_5,\oplus_5,\otimes_5)$ is a Field

```
 // to verify ring (z5,+5,*5)is a field
//1.to verify ring (z5,+5,*5) is a ring with unity
clc;
z=[0 1 2 3 4]
n=length(z);
e=1;
 for i=1:n
     if modulo(z(i)*e,5)~=z(i) | modulo(e*z(i),5)~=z(i) then
      printf("\nIdentity law doesnot hold good");
      printf("\n Z5 is not a ring with unity");
 abort;
     end
 end
printf("\n(Z5,*) is a ring with unity")
//to verify every non zero element has multiplicative inverse
 flag=0;
 for i=2:n
```

```
    for j=2:n


      if modulo(z(i)*z(j),5)==e &modulo(z(j)*z(i),5)==e then
           flag=1;
           break;
        end
    end
       if flag==0 then
             printf("\n inverselaw doesnot hold good %d in Z5",i);
             printf("\n Z5 is not a field");
      abort;
       end
end
printf("\ninverse law holds in (z5,*)")
//3.to verify ring (z5,+5,*5) is a commutative ring under *
for i=2:n
    for j=2:n
        k1=modulo(z(i)*z(j),5);
        k2=modulo(z(j)*z(i),5);
        if k1~=k2 then
          printf("\n commutative law fails undermultiplication");
      printf("z7 is not a commutative ring");
abort;
        end
    end
end
printf("\n(Z5,*) is a commutative ring")
printf("\n and  hence (z5,+,*) is a  field")
```

10.Scilab Program to verify ring $(Z_7,\oplus_7,\otimes_7)$ is a Field

*****

2.Numerical Analysis

(Scilab code)

a.Interpolation

1. Find f(1.4) from the following table:

x	1	2		3	4	5
f(x)	10	26		58	112	194
Answer: f(1.4)=14.864						

```
//scilab code to estimate the value of f(1.4)
//using Newton's forward difference table
clc;
x=1:5
printf("\nx values: ");
disp(x);
printf("\n y values:  ");
y= [10 26 58 112 194]
disp(y);
X=1.4;  //x at which value of f to be estimated
n=length(x);
h=x(2)-x(1);
p=(X-x(1))/h;
sum1=y(1);
term=1;
printf("\nDifference Table ")
for i=1:n-1
printf("\n")
   for j=1:n-i
        y(j)=y(j+1)-y(j);
printf("\t%d",y(j))
 end
    term=term*(p-i+1)/i;
    sum1=sum1+term*y(1);
end
printf("\n The value of f(1.4): %f ",sum1)
```

2. Evaluate $y=e^{2x}$ for x=0.05 from the following table.

X	0	0.10	0.20	0.30	0.40
Y=e2x	1.00	1.22	1.49	1.82	2.26
Answer: f(0.05)=1.1052					

3. Estimate f(7.5) from the following table:

x	1	2	3	4	5	6	7	8
f(x)	1	8	27	64	125	216	343	512
Answer: f(7.5)=421.87								

```
//scilab code to estimate the value of f(7.5)
//using Newton's backward difference table
clc;
x=1:8
y= [1 8 27 64 125 216 343 512]
X=7.5;
printf("\n x values:  ")
disp(x);
printf("\n y values:  ")
disp(y);
n=length(x);
h=x(2)-x(1);
p=(X-x(n))/h;
sum1=y(n);
term=1;
printf("\nDifference Table")
for i=1:n-1
printf("\n")
   for j=1:n-i
       y(j)=y(j+1)-y(j);
printf("\t%d",y(j))
end
    term=term*(p+i-1)/i;
    sum1=sum1+term*y(n-i);
end
printf("\n value of f(7.5) :  %f",sum1)
```

4. Estimate f(1.28) from the following table

x	1.15	1.20	1.25	1.30
f(x)	1.0723	1.0954	1.1180	1.1401
Answer: f(1.28)=1.1312				

5. Using Lagrange's interpolation formula find f(10) from the following data:

x	5	6	9	11
f(x)	12	13	14	16
Answer: f(10)= 14.6667				

```
//Scilab code to estimate f(10) for the given data
// using lagrange's interpolation formula
clc;
x=[5 6 9 11];
y=[12 13 14 16]
X=10;
printf("\n x values:  ")
disp(x);
printf("\n y values:  ")
disp(y);
sum1=0; n=length(x);
for i=1:n
    term=1;
    for j=1:n
        if i~=j  then
            term=term*(X-x(j))/(x(i)-x(j));
        end
    end
    sum1=sum1+term*y(i);
end
printf(" \n    Estimated value of f(10): %f",sum1)
```

6. Using Lagrange's interpolation formula find f(6) from the following data

x	3	7	9	10
f(x)	168	120	72	63
Answer: f(10)= 14.6667				

b. ***INTEGRATION***

1.Evaluate $\int_0^6 \frac{1}{1+x^2}\,dx$ *using trapezoidal rule. Take n=6 [Ans:1.4108]*

```
//scilab code to evaluate integration using trapezoidal rule
clc;
funcprot();
function y=f(x)
    y=1/(1+x^2)
endfunction
x0=input("Lower limit of the interval: ");
xn=input("upper limit of the interval: ");
n= input("no. of subintervals: ");
h=(xn-x0)/n;
sum1=f(x0)+f(xn);
for i=1:n-1
    sum1=sum1+2*f(x0+i*h);
end
printf("\n Estimated value of given integration: %f", h/2*sum1);
```

2. Evaluate $\int_1^5 log_{10}\, x\, dx$ *taking 8 subinterval by Trapezoidal rule.*
[Ans: 1.7505]

3.Evaluate $\int_0^6 \frac{1}{1+x^2}\,dx$ *using Simpson's 1/3rd rule. Take n=6*

[Ans:1.3662]

```
//scilab code to evaluate integration using simpson's 1/3rd rule
clc;
funcprot();
function y=f(x)
    y=1/(1+x^2)
endfunction
x0=input("Lower limit of the interval: ");
xn=input("upper limit of the interval: ");
n= input("no. of subintervals(even number): ");
h=(xn-x0)/n;
sum1=f(x0)+f(xn);
for i=1:n-1
    if modulo(i,2)==0 then
        sum1=sum1+2*f(x0+i*h);
    else
        sum1=sum1+4*f(x0+i*h);
    end
end
printf("\n Estimated value of given integration: %f", h/3*sum1);
```

4.Evaluate $\int_0^{\pi/2} \sqrt{cosx}\,dx$ *using Simpson's 1/3rd rule. Take n=6 by simpson's 1/3rd rule by dividing* $\left[0,\ \frac{\%pi}{2}\right]$ *into 6 equal part.*

[Ans:1.1873, note: π*=22/7]*

5.Evaluate $\int_0^6 \frac{1}{1+x^2}\,dx$ *using Simpson's 3/8th rule. Take n=6*

[Ans:1.3571]

```
//scilab code to evaluate integration using simpson's 3/8th rule
clc;
funcprot();
function y=f(x)
    y=1/(1+x^2)
endfunction
x0=input("Lower limit of the interval: ");
xn=input("upper limit of the interval: ");
n= input("no. of subintervals (Multiples of 3): ");
h=(xn-x0)/n;
sum1=f(x0)+f(xn);
for i=1:n-1
    if modulo(i,3)==0 then
        sum1=sum1+2*f(x0+i*h);
    else
        sum1=sum1+3*f(x0+i*h);
    end
end
printf("\n Estimated value of given integration: %f", 3*h/8*sum1);
```

6.Evaluate $\int_1^3 \frac{1}{(1+x)^2}\,dx$ *using Simpson's 3/8th rule. Take n=3*

[Ans:0.8047]

3. Vector Differential Calculus

(Maxima code)

1. Find the gradient of x^2yz

Method1:

```
kill(all)$
load("vect")$
f:(x^2*y*z);
express(grad(f))$
grdf:ev(%,nouns)$

print("gradient of f:  ",grdf)$
```

Output:

(%o2) $x^2\ y\ z$

(%i3)

(%i4)

gradient of f: $[2\ x\ y\ z, x^2\ z, x^2\ y]$

Method2:

```
kill(all)$
load("vect")$
f:(x^2*y*z);
express(grad(f))$
g:ev(%,nouns)$
grdf:0$
A:[i,j,k];
for i:1 thru 3 do (
    grdf:grdf + (g[i])*A[i])$
print("gradient of f:  ",grdf)$
```

output:

(%o2) $x^2\ y\ z$

(%o3) $[2\ x\ y\ z, x^2\ z, x^2\ y]$

(%o5) $[i, j, k]$

gradient of f: $2\ i\ x\ y\ z + j\ x^2\ z + k\ x^2\ y$

2. Find the gradient of $x^2+y^2+z^2$ (Ans: 2xi+2yj+2zk)

3. Find the divergence of the $\vec{F} = x^2yi + yz^2j + x^2zk$

```
kill(all)$
load("vect")$
F:[x^2*y, y*z^2, z*x^2];
express(div(F))$
divf:ev(%,nouns)$
print("Divergence of f : ", divf)$
```

output:

(%o2) $[x^2 y, y z^2, x^2 z]$

(%o3) $\frac{d}{d y}(y z^2)+\frac{d}{d z}(x^2 z)+\frac{d}{d x}(x^2 y)$

Divergence of f : $z^2+2 x y+x^2$

4. Find the divergence of $\vec{F} = x^2coszi + ylogx \quad j - yzk$
(Ans: div F = 2*x*cos(z)-y+log(x))

5. Find the curl of $\vec{F} = xy^2i + 2x^2yzj - 3yz^2k$

```
kill(all)$
load("vect")$
f:[x*y^2, 2*x^2*y*z, -3*y*z^2];
express(curl(f))$
curlf:ev(%,nouns)$
print(" Curl of f: ", curlf)$
```

output:

(%o2) $[x y^2, 2 x^2 y z, -3 y z^2]$

Curl of f: $[-3 z^2-2 x^2 y, 0, 4 x y z-2 x y]$

6. Find the curl of $\vec{F} = x^2yi + yz^2j + x^2zk$ (Ans: [2yz,2xz,x^2])

7. Find$\nabla^2\emptyset$ for $\emptyset = x^2 - y^2 + 4z$

```
kill(all)$
load("vect")$
F:x^2-y^2+4*z;
express(laplacian(F))$
lapf:ev(%,nouns)$
print("Laplacian of F: ", lapf)$
```

<u>output:</u>

```
(%o2) 4 z-y^2+x^2
(%i3)
(%i4)
(%i5)
Laplacian of F:  0
```

8. Find Laplacian of $\emptyset = x^2y^2z^2$ (Ans: $2(y^2z^2 + z^2x^2 + x^2y^2)$)

9. Find the differential vector operation in Cartesian coordinate system.

```
/*differential vector operations in Cartesian coordinates*/
kill(all)$
load(vect)$
load(vect_transform)$
declare(F,scalar)$
declare([F,G], nonscalar)$
f:x^2+y^2+z^2;
F:[x+y*z,y+z*x,z+x*y];
scalefactors(cartesian3d)$

express(grad(f))$       ev(%,nouns);
express(div(F))$        ev(%,nouns);
express(laplacian(f))$  ev(%,nouns);
express(curl(F))$       ev(%,nouns);
express(div(f*F))$      ev(%,nouns);
express(curl(grad(f)))$ ev(%,nouns);
```

10. Find the differential vector operation in cylindrical coordinate system

```
/*differential vector operations in cylindrical coordinates*/
kill(all)$
load(vect)$
load(vect_transform)$
declare(f,scalar)$
declare([F,G],nonscalar)$
f:r*cos(theta);
F:[r,r*cos(theta),sin(theta)];
scalefactors(polarcylindrical)$

express(grad(f))$       ev(%,nouns);
express(div(F))$        ev(%,nouns);
express(curl(F))$       ev(%,nouns);
express(laplacian(f))$  ev(%,nouns);
express(curl(grad(f)))$ ev(%,nouns);
```

11. Find the differential vector operation in Spherical coordinate system

```
/*Changing coordinate system to spherical-polar coordinates*/
 kill(all)$
 load(vect)$
 load(vect_transform)$
 declare(f,scalar)$
 declare(F, nonscalar)$
 scalefactors(spherical)$

 express(grad(f));
 express(div(F));
 express(laplacian(f));
 express(div(f*F));
```

*****