- A detailed study may be carried out on the implementation of controller for the simultaneous reduction of SO_2 and NO_x with H_2O_2, and formation of nitric and sulphuric acids as by-products.

APPENDIX 1

HARDWARE SPECIFICATIONS

Table A1.1 Peristaltic pump specifications

Sl. No.	Description	Details
1	Model	RH-P 100VS 100
2	No. of Channels	1
3	Flow rate	0.1 ml to 60 ml / min
4	On timer	0 to 99 hours 59 minutes (In steps of 1 min

		increment)
5	Interval time	0 to 99.9 seconds (In steps of 0.1 sec increment)
6	Tubing	1 mm to 3 mm ID with 1 to 1.5 mm wall thickness
7	Outer arm	NORYL
8	Arm cover	Transparent poly carbonate
9	Rotor	Glass filled nylon
10	Rollers	Carbon filled nylon
11	Motor	DC stepper motor (Cont. duty)
12	Pressure	Upto 2 kg / sq. cm.
13	Accuracy	± 1%
14	Supply	230 v, 50 Hz, Single phase AC
15	Dimension	110 x 225 x 280 mm (H x W x D)
16	Weight	7.5 Kgs (App.)

Table A1.2 SO_2 gas sensor specifications

Sl. No.	Description	Details
1.	Make	VASTHI
2.	Model	Expo-98 FTD
3.	Gas detection	SO_2 in atm
4.	Sensor	Electrochemical
5.	Principle	Diffusion
6.	Input	24V AC
7.	Output	4-20mA
8.	Display	Digital LCD Display

9.	Range	0- 5000 ppm
10.	Sensor Holder	SS Housing
11.	Accuracy	±2%

Table A1.3 General Specifications of VDPID-03 card

Sl. No.	Description	Details
1.	ADuC841 microcontroller	• High Speed 420 KSPS ADCs (Successive approximation converter) • 2 independent 24-bit ADCs • 12-bit voltage output DAC • Upto 10 ADC input channels on all parts • 62-kbytes on-chip Flash/EE program memory
2.	ADC channel	• Input range: (0-5 V) • Input channels: 8 single ended • Current input channels (4-20 mA): 2 • Voltage input channels (0-5 V): 6 • Accuracy: ± 0.8% • Conversion time: 7 conversion per sec • Resolution: 12 bits
3.	DAC channel	• Output range: (0-5 V) • Output channels: 2 • Accuracy: ± 0.02% • Resolution: 12 bits
4.	I/V Converter	• Input and Output channel : 1 • Output range (0-5 V)
5.	V/I Converter	• Input and Output channel : 2 • Output range (4-20 mA)
6.	Interface with Supervisory Computer System	Through RS232
7.	Real time control algorithm	MATLAB/SIMULINK platform

DESIGN DETAILS OF ACCESSORIES

- **A Shell and tube Heat exchanger**

Type	:	Shell and tube
Length	:	300mm
Dia	:	60mm
Number of tube	:	6
Tube Dia	:	6mm
Connection	:	½" Flanged

- **A VFD based pump**

Input	:	(4-20)mA or (0-10)V DC
Flow rate	:	(0-200)lph
Power	:	0.5HP
Body material	:	MS.
Suction material	:	PVDC
Connection	:	½" BSP

- **Process tank (Recirculation tank)**

Material	:	Acrylic
Height	:	300 mm
Diameter	:	300 mm
Thickness	:	5 mm

- **Solution tank (H_2O_2 and Water)**

Material	:	Acrylic
Height	:	300 mm
Diameter	:	300 mm
Thickness	:	5 mm

- **Pressure regulator**

Make	:	ABB
Serial No.	:	8406123
Maximum input	:	18 kg / cm^2
Output	:	(0.2 -8) kg / cm^2
Special Features	:	Regulator/Filter
End connection	:	1/4" BSP(F) Thread

- **Submersible pump**

Make	:	Tullu
Model	:	Handy II
Supply	:	230 V AC
Outlet nozzle	:	13 mm
Max. head	:	1.75 m
Max. discharge	:	800 lph

APPENDIX 2

MATLAB CODING FOR OBTAINING CDM-PID CONTROLLER PARAMETERS

```
% Plant transfer function

num=[0 1.5616];                 % Numerator of plant transfer function
den=[13.58 1];                  % Denominator of plant transfer function
delay=23.48;                    % Equivalent transfer function using
                                   Pade approximation technique

numdel=[-delay 2];              % Pade approximation of the continuous-
time delay exp(-T*s)               [Numerator]
dendel=[delay 2];               % Pade approximation of the continuous-
time delay exp(-T*s)               [Dinaminator]
np=conv(num,numdel)             % polynomial multiplication
                                   [Numerator]
dp=conv(den,dendel)             % polynomial multiplication
                                   [Denominator]

% Selected stability indices and Equivalent time constant
%ts=102;                        % Enter the value of settling time
g1=2.5;                         % Gama1 value
g2=2;                           % Gama2 value
tau=40.8;                       % Equivalent time constant
```

```
% Coefficients of CDM controller polynomials
p3=np(1,3)*(((tau)^3)/(g1^2*g2));          % Coefficient of target
                                              characteristics polynomial
p2=np(1,3)*((tau^2)/g1);                   % Coefficient of target
                                              characteristics polynomial
p1=np(1,3)*tau;                            % Coefficient of target
                                              characteristics polynomial
p0=np(1,3);                                % Coefficient of target
                                              characteristics polynomial
ptarget=[p0 p1 p2 p3];a=[0 0 np(1,3) 0;0 np(1,3) np(1,2) dp(1,3);np(1,3)
np(1,2) 0 dp(1,2);np(1,2) 0 0 dp(1,1)]
b=[p0 p1 p2 p3]'
c=a\b
 % Coefficient of controller polynomials
k2=c(1,1)
k1=c(2,1)
k0=c(3,1)
l1=c(4,1)
k2=(((318.85*g1^2*g2)– tau^3)/(g1^2*g2*36.6664));
k1=(((3.1212*k2*g1)– tau^2)/(36.6664*g1));
k0=(((3.1212*k1)– tau)/36.6664);
% CDM controller parameters
Kc=k1/l1
Ti=k1/k0
Ki=Kc/Ti
Td=k2/k1
Kd=Kc*Td
kf1= Ti*Td;
kf2= Ti;
```

APPENDIX 3

MATLAB CODING FOR OBTAINING PARAMETERS FOR FOCDM-PI$^{\lambda}$D$^{\mu}$ CONTROLLER PARAMETERS

```
function[fu]=fopid(par)
for t=1:20
    nump=[– 36.6664 3.1212];        % Numerator controller polynomial
    denp=[318.85 50.64 2];          % Denominator controller
                                      polynomial
    so2=tf(nump,denp);              % controller transfer function
    %  Initialsization of variables
    lemda =par(t,1);                % lamda from PSO coding
    meu = par(t,2);                 % meu from PSO coding
    Kp = 0.3357;
    Ki = 0.0186;
    Kd = 0.098;
    wc=0.000992;
    wmax=99.2;
    y=1;

% Sigularity function approximation method

  a = 10*(y/(10*(1– lemda)));
    b = 10*(y/(10*lemda));
    Po = wc*(y/(20*lemda));
```

```
Zo = a*Po;
N1 = round([log(wmax/Po)/log(a*b)]+1);
N1=real(N1);
Pi=0;
Zi=0;
for i=0:N1
Pi1=Po*(a*b)^i;
 Pi=Pi+(1/Pi1);
end
for j=0:(N1-1)
   Zi1=Zo*(a*b)^j;
   Zi=Zi+(1/Zi1);
End
% integer order approximation of controller equation
num=[(Ki*Zi) (Ki*N1)];
den=[Pi N1];
Ci=tf(num,den)
a1 = 10*(y/(10*(1– meu)));
b1 = 10*(y/(10*meu));
Z1 = wc*(y/(20*meu));
P1 = a*Z1;
ND = round([log(wmax/Z1)/log(a*b)]+1);
ND=real(ND);
Pd=0;
Zd=0;
for l=0:ND
   Pd1=P1*(a*b)^l;
   Pd=Pd+(1/Pd1);
   Zd1=Z1*(a*b)^l;
```

```
    Zd=Zd+(1/Zd1);
end
% integer order approximation of controller equation
nu=[(Kd*Zd) (Kd*ND)];
de=[Pd ND];
Cd=tf(nu,de);
Cs=Kp*(1+(Ki*Ci)+(Kd*Cd));
% feedback from the controller by applying random lamda and meu
sys_cl=feedback(Cs*so2,1);
t=0:1:500;
bbout=step(500*sys_cl,t);
sp=500;
err=sp-bbout;
ISE=sum(err.*err);
Mp=((max(bbout)− sp)/sp)*− 1;
beta=1;
F=(1− exp(beta))*(ISE);
fu=F;
end
```

APPENDIX 4

MATLAB CODING FOR OBTAINING λ AND μ USING PSO ALGORITHM

```
clc;
clear all;
 n=250;                                  %particles size
dim=2;                                   %number of particles
p(1:n,1)=0; q(1:n,1)=0.2;                %initializing the particles
p(1:n,2)=0.8; q(1:n,2)=1;
s=r/2;
pos=p+r.*rand(n,dim);                    %initialization of position
vel=s.*rand(n,dim);                      %initialization of velocity

obfun= fopidn(pos(:,:));                 % call the objective function
[fgbest,igbest]=min(obfun);
gbest=pos(igbest,:);                     %find the glodal best value
pbest=pos;                               %find the local best value
fpbest=obfun;

itmax=2000;                              %number of iterations
c1=2; c2=2;                              % learning factor
wmax=0.9; wmin=0.4;                      % inertia weights
w=linspace(wmax,wmin,itmax)
 for it=1:itmax;
```

```
% update the velocity and position

vel(1:n,1:dim)=w(it)*vel(1:n,1:dim)+c1*rand*(pbest(1:n,1:dim)– pos(1:n,1:di
m))+c2*rand*(repmat(gbest,n,1)– pos(1:n,1:dim));
pos(1:n,1:dim)=pos(1:n,1:dim)+vel(1:n,1:dim);

% call the objective function

obfun = fopidn(pos(:,:));
[minf,iminf]=min(obfun);
if minf<= fgbest
fgbest=minf; gbest=pos(iminf);
end
q(it,:)=gbest(1,:);
inewpb=find(obfun<=fpbest);
pbest(inewpb,:)=pos(inewpb,:); fpbest(inewpb)=obfun(inewpb);
end
```