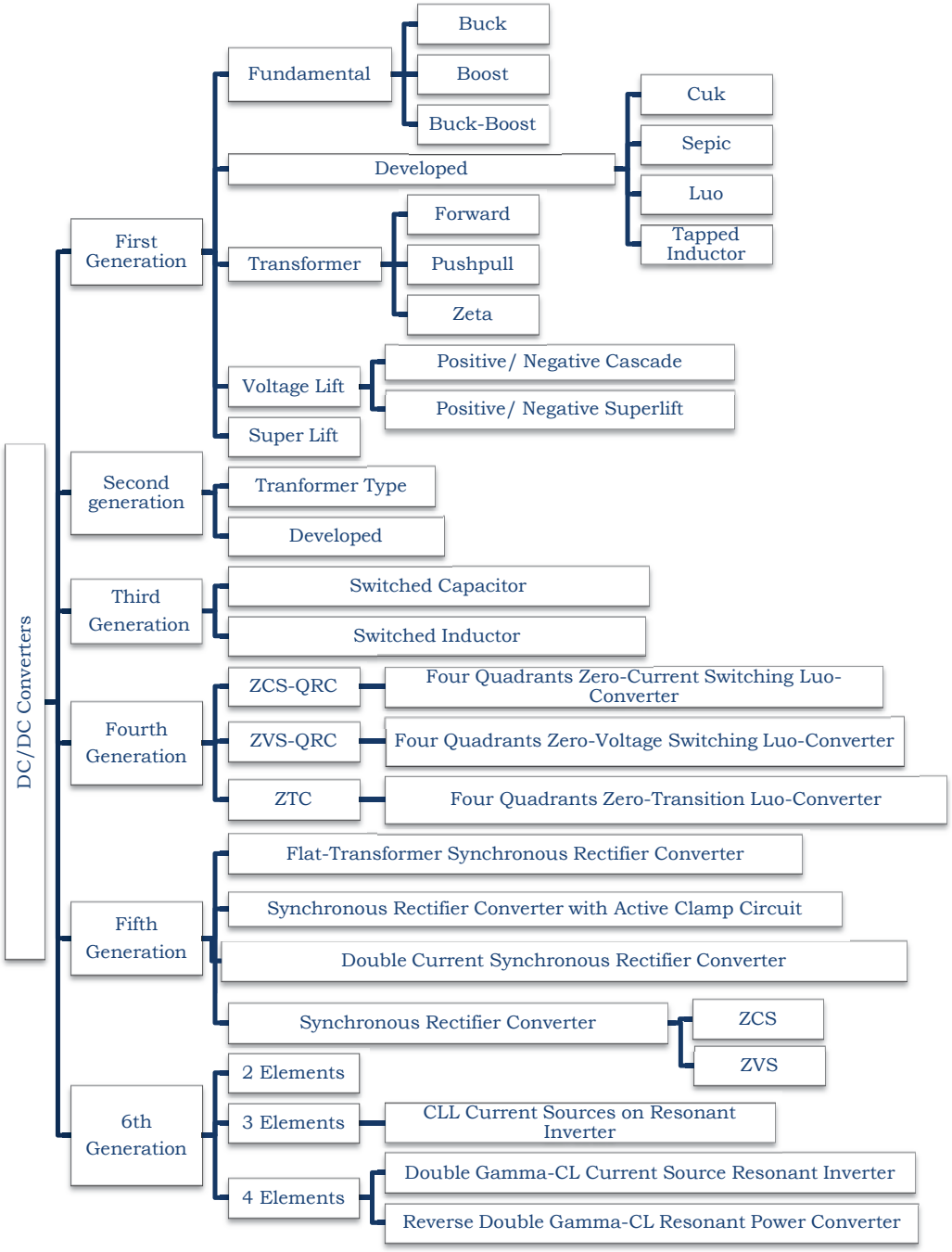
Appendices

Appendix A: DC-DC Converter family[101]



Several researches has highlighted that converters must have high power density and better transfer efficiency. Usually, the repeating frequency is not very high and the converter works in the resonance state, the components of higher order harmonics is very low. Further, the electromagnetic interference (EMI), electromagnetic susceptibility (EMS) and electromagnetic compatibility (EMC) are major factors for its performance. These factors introduce the sixth generation converters also called multiple energy-storage elements resonant (MER) converters are better options for future work.

Appendix B: MATLAB Program for Energy generation prognosis

Sun_Angle function:

```
function[azimuth,altitude]=Sun_Angle(date,month,year,hours,min,placelat,placelon
,gmtdif) d = 367*year
floor((7*(year+(floor((month+9)/12))))/4)+floor((275*month)/9)+date-730530;
w = 282.9404 + 4.70935*10^(-5)*d; %longitude of perihelion
a = 1.00000; % mean distance
e = 0.016709 - 1.151*10^(-9)*d; %eccentricity
M = to360deg(356.0470 + 0.9856002585 * d); %mean anomaly
oblecl = 23.4393 - 3.563E-7 * d; %obliquity of the ecliptic
L = to360deg(w + M); % mean longitude
E = M + (180/pi) * e * sin(M*pi/180) * (1 + e * cos(M*pi/180)); %eccentric
anomaly
%rectangular coordinates in the plane of the ecliptic
x = cos(E*pi/180)-e;
y = sin(E*pi/180)*sqrt(1-e^2);
r = sqrt(x*x + y*y);
v = (180/pi)*atan2(y, x);
lon = to360deg(v + w);
%ecliptic rectangular coordinates
x = r*cos(lon*pi/180);
y = r*sin(lon*pi/180);
z = 0.0;
%equatorial coordinates
xequat = x;
yequat = y*cos(oblecl*pi/180) + z*sin(oblecl*pi/180);
zequat = y*sin(oblecl*pi/180) + z*cos(oblecl*pi/180);
% to RA and Declination
RA = (180/pi)*atan2(yequat,xequat);
Decl = (180/pi)*asin(zequat/r);
%Sidereal time
GMST0 = L/15+12;
UT = hours+min/60-gmtdif;
SIDETIME = GMST0+UT+placelon/15;
%hour angle
HA = to360deg(15*(SIDETIME-RA/15));
x = cos(HA*pi/180)*cos(Decl*pi/180);
y = sin(HA*pi/180)*cos(Decl*pi/180);
z = sin(Decl*pi/180);
xhor = x*sin(placelat*pi/180)-z*cos(placelat*pi/180);
yhor = y;
zhor = x*cos(placelat*pi/180)+z*sin(placelat*pi/180);
azimuth = (180/pi)*atan2(yhor,xhor)+180;
altitude = asin( zhor )*180/pi;
end
```

Vector function:

```
function vec=Vector(azimuth, height)
vec=[cosd(azimuth).*cosd(height),cosd(90-azimuth).*cosd(height),-
sind(height)];
```

Vec_Angle function:

```
function angles=Vec_Angle(Vec1,Vec2)
angles=[];
if size(Vec1)==size(Vec2)
[max,q]=size(Vec1);
for i=1:max
angles=[angles,acosd(Vec1(i,:)*Vec2(i,:)'/(sqrt(Vec1(i,:)*Vec1(i,:)')*sqrt(Vec
2(i,:)*Vec2(i,:)')))];
end
else if (all(size(Vec2)==[1,3]))&&all(size(Vec1)>=[1,3])
[max,q]=size(Vec1);
for i=1:max
angles=[angles;acosd(Vec1(i,:)*Vec2'/(sqrt(Vec1(i,:)*Vec1(i,:)')*sqrt(Vec2*Vec
2')))];
end
else
```

```
angles='error';
end
end
```

Day_Irradiation function:

```
function [dir,dif,Time]=Day_Irradiation(date,month,year)
global gmt_diff pan_az pan_incl place_lat place_long
H=[];
M=[];
Time=[];
angles=[];
for h=0:23
for m=0:1:59
[az,al]=Sun_Angle(date,month,year,h,m,place_lat,place_long,gmt_diff);
%if(al>0)
angles=[angles; az,al];
H=[H,h];
M=[M,m];
Time=[Time, h+m/60];
%end
end
end
vectors=Vector(angles(:,1),angles(:,2));
panel_vec=Vector(pan_az+180,pan_incl-90);
panel_angles=Vec_Angle(vectors,panel_vec);
dir=sind(panel_angles-90).*(sind(panel_angles-90)>0);
dif=sind(angles(:,2)/6).*(sind(angles(:,2)/6)>0);
```

Power function:

```
function [est_power, Time]=Power(date,month,year,clouds,rain)
global max_power prev_date dif dir Time;
(year+(month/120)+(date/31000))
if ((year+(month/120)+(date/31000))>prev_date)
[dir,dif,Time]=Day_Irradiation(22,6,2012);
prev_date=year+month/120+date/31000;
else if year+month/120+date/31000<prev_date
est_power='error'
return
end
end
est_power=max_power.*((1-clouds*0.95).*dir'+((1-rain).*clouds*0.95).*dif');
return
```

Despatcher function:

```
%Despatcher
Date=19
Month=6
Year=2017
hours_electr_price=[ones(1,24)*25];%electricity price in inr
%cons_by_time=[power(W),work_time(m),due_time(h+m/60)]
%cons_bycond=[power(W),target_value,dead zone, max_value, on(1)/off(0)]
cons_by_time=[2000,60,11;2000,60,12.5;2000,60,14;2000,60,16.5;2000,60,19;2000,
60,21.5];
cons_by_cond=[200,20,3,30,0];
global prev_date clouds rain place_lat place_long pan_az pan_incl gmt_diff
max_power;
prev_date=0;
place_lat;
place_long=;
pan_az=;
pan_incl=;
max_power=2200;
if (Date<0||Date>31||Month<0||Month>12||Year<2017)
'error wrong date'
return
end
if(Month>3&&Month<11)
```

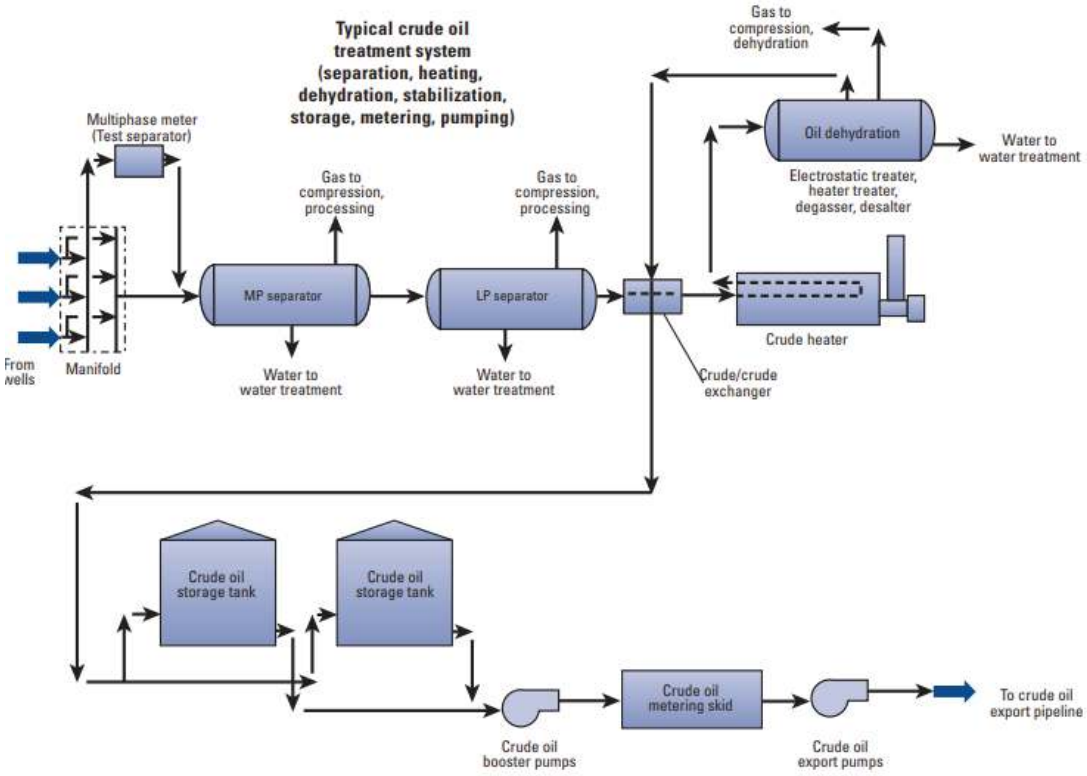
```
gmt_diff=3;
else
gmt_diff=2;
end
power_usage=[zeros(1,24*60)];% power usage controlled by system
for(real_hour=0:23)
[est_p,Time]=Power(Date,Month,Year,clouds,rain);
est_c=Consumption(1);
free_power=est_p-est_c-power_usage;
real_hour% moment of observation
possible_times=[];
for(i=1:size(cons_by_time))
p2=[];
climbdown=1;
while(size(p2)==0)
p1=[(free_power>=climbdown.*cons_by_time(i,1)).*[ones(1,60*cons_by_time(i,3))
zeros(1,60*(24-cons_by_time(i,3)))].*[zeros(1,60*real_hour) ones(1, 60*(24-
real_hour))]];
[qq,first_pos]=max(p1)
first_pos=(ceil(first_pos/20))*20
for(j=first_pos:20:cons_by_time(i,3)*60-cons_by_time(i,2))
if p1(j:j+cons_by_time(i,2)-1)==ones(1,cons_by_time(i,2))
p2=[p2, j];%j - number of time instance
end
end
climbdown=climbdown-0.01;
end
[sz,ln]=size(possible_times);
[sz2,ln2]=size(p2);
if(ln>ln2)
p2=[p2, ones(1,ln-ln2
possible_times=[possible_times,possible_times(:,ln)*ones(1,ln2-
ln)];
end
end
possible_times=[possible_times; p2];
end
var=Variants(possible_times);
var=unique(var,'rows');
[sz,ln]=size(var);
best=var(1,:);
best_fun=-inf;
for (i=1:sz)
power_res=free_power;
for (j=1:ln)
power_res=power_res-[zeros(1,var(i,j))
ones(1,cons_by_time(j,2)).*cons_by_time(j,1) zeros(1,24*60-var(i,j)-
cons_by_time(j,2))];
end
fun=[];
for (j=1:ln)
fun=[fun min(power_res(var(i,j):var(i,j)+cons_by_time(j,2)))];
end
fun=(min(fun));hours_power_res=[];
for(jj=0:23)
hours_power_res=[hours_power_res,
sum(power_res(60*jj+1:60*(jj+1)))/60];
end
price=sum(hours_power_res.*(hours_power_res<0).*hours_electr_price);
fun=fun+price;
if(fun>best_fun)
best_fun=fun;
best=var(i,:);
best_power_res=power_res;
bhpr=hours_power_res;
best_price=price;
end
end
```

```
best
hour_power_res=free_power(1+real_hour*60:(real_hour+1)*60);
for i=1:length(best)
if best(i)>real_hour*60&&best(i)<(real_hour+1)*60
hour_power_res=hour_power_res-[zeros(1,best(i)-real_hour*60)
cons_by_time(i,1).*ones(1,(real_hour+1)*60-best(i))];
end
end
for minut=0:59
real_time=real_hour*60+minut;
% control of divices with programmed time counter=1
while counter<=length(best)
if best(i)==real_time
'start i time programmed device'
power_usage=power_usage+[zeros(1,best(i))
ones(1,cons_by_time(i,2)).*cons_by_time(i,1) zeros(1,24*60-best(i)-
cons_by_time(i,2))];
cons_by_time(i,:)=[];
best(i)=[];
else
counter=counter+1;
end
%control of devises with condition control
[sz,ln]=size(cons_by_cond);
for i=1:sz
if cons_by_cond(i,5)==0
if sensor(i)>cons_by_cond(i,4) 'start i condition programmed device'
cons_by_cond(i,5)=1;
power_usage(real_time)=power_usage(real_time)+cons_by_cond(i,1);
hour_power_res(minut+1)=hour_power_res(minut+1)+cons_by_cond(i,1);
else if
(sensor(i)>cons_by_cond(i,2)+cons_by_cond(i,3))&&sum(hour_power_res)>0
'start i condition programmed device'
cons_by_cond(i,5)=1;
power_usage(real_time)=power_usage(real_time)+cons_by_cond(i,1);
hour_power_res(minut+1)=hour_power_res(minut+1)+cons_by_cond(i,1);
end
end
if cons_by_cond(i,5)==1
if sensor(i)<cons_by_cond(i,2)
'stop i condition programmed device'
cons_by_cond(i,5)=0;
else if
(sensor(i)<cons_by_cond(i,2)+cons_by_cond(i,3))&&sum(hour_power_res)<=0
'stop i condition programmed device'
cons_by_cond(i,5)=0;
else
power_usage(real_time)=power_usage(real_time)+cons_by_cond(i,1);
hour_power_res(minut+1)=hour_power_res(minut+1)+cons_by_cond(i,1);
end
end
```

Variants function:

```
function mtrx=Variants(base)
mtrx=[];
for i=1:length(base(1,:))
[sz,ln]=size(base);
if(sz==1)
mtrx=[mtrx; base(1,i)];
else
next=Variants(base(2:sz,:));
[sz,ln]=size(next);
mtrx=[mtrx; base(1,i)*ones(size(next),1) next];
end
end
return
```

Appendix C: Typical crude oil treatment system



The heat requirement for crude oil heating depends upon the design of horizontal heater treater with the input parameters as oil gravity, oil flow rate (bbl/day), gas gravity, number of wells, inlet oil temperature and others. The settling equation and retention time equations are used for determine the heat requirement and viscosity of oil. The water cut in the crude oil also affects the heat requirement. Assuming that the free water has been separated from the emulsion and the remaining amount of water is less than 10% of the oil.

Preheat energy requirement to heat the wet crude oil entering the heater treater is calculated by the following formula [102]:

$$Q = \frac{350}{24}[(0.5\gamma_o q_o) + (\gamma_w q_w)]\Delta T$$

where

q	Treating rate, bbl/ (day.ft²)
qo	Oil flow rate, bbl/ day
qw	Water flow rate, bbl/ day
API	Oil gravity
γ_o	Oil specific gravity
γ_w	Water specific gravity
T	Treating temperature,˚F
Q	Heat, BTU/hr
ΔT	Temperature difference in ˚F

Appendix D: Anaerobic digesters for on-site carbon capture and biogas

Vent / Flue gas is inexpensive and rich source of CO_2, approximately 400 times more concentrated than atmospheric CO_2 and thus can be exploited for microalgal mediated CO_2 fixation. However, the major constraints are due to the presence of toxic compounds such as NOx, SOx and CO which are inhibitory for microalgal growth and biomass productivity, primarily due to acidification. In order to treat Vent/ flue gas by microalgae in a cost effective manner, there is need to develop effective strategies including design of PBR system, flue gas input system and anaerobic digesters which will not only reduce the cost of its pre-treatment but also increase CO_2 fixation.

The above figures (A) to (F) shown the various reactor configuration for microalgal cultivation at (A) raceway pond (from Sapphire Inc, America); (B) Floting photobioreactor (from OMEGA system- ASA); (C) tubular bioreactor (Wolfsburg, Germany); (D) Plastic Bag bioreactor (from Algenol Inc, America); E) Multi-layer bioreactor (from center for biorefining, University of Minnesota; (F) Flat-panel Bioreactor (from Nanovoltaics technologies, America) [103]