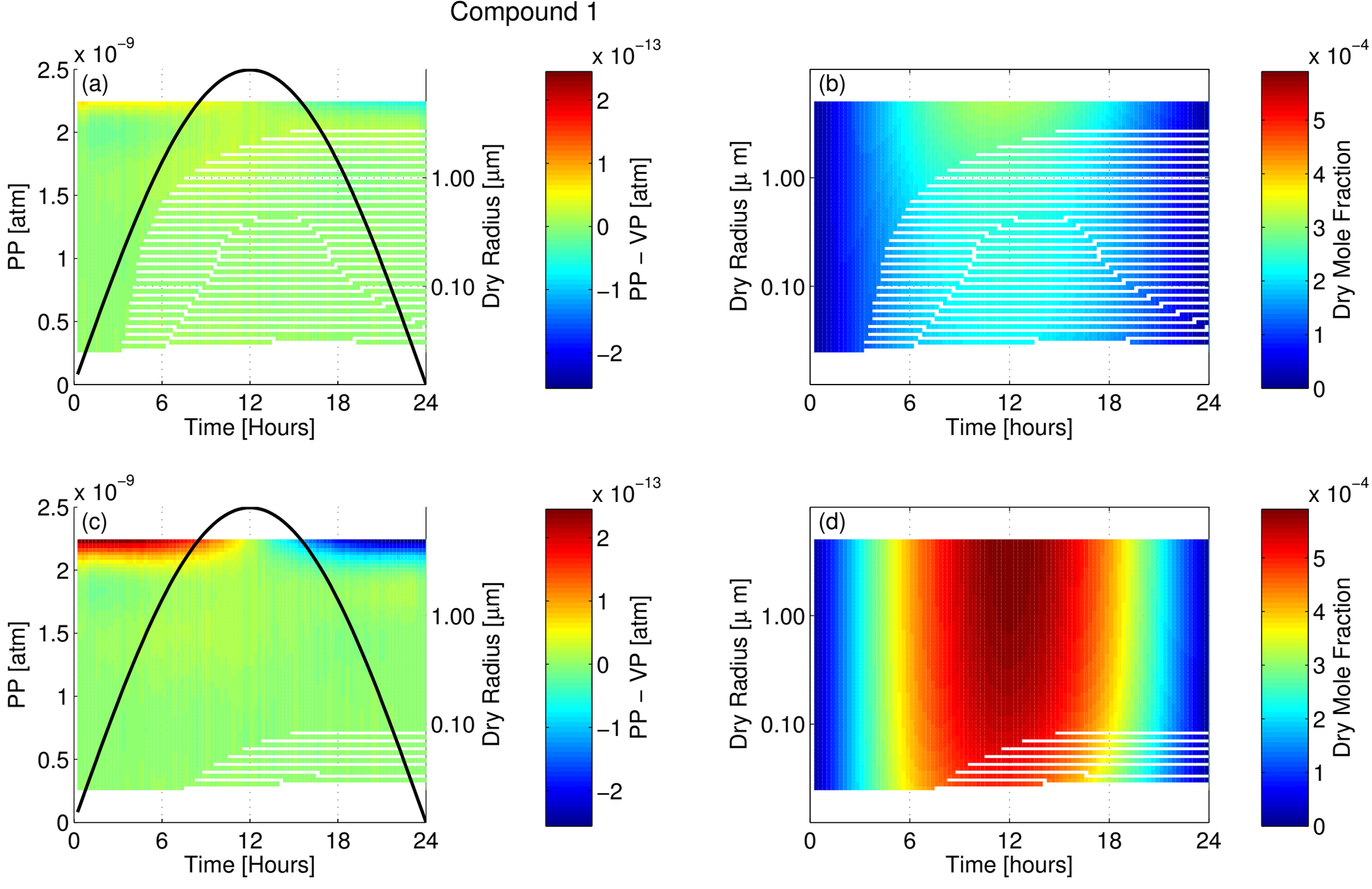
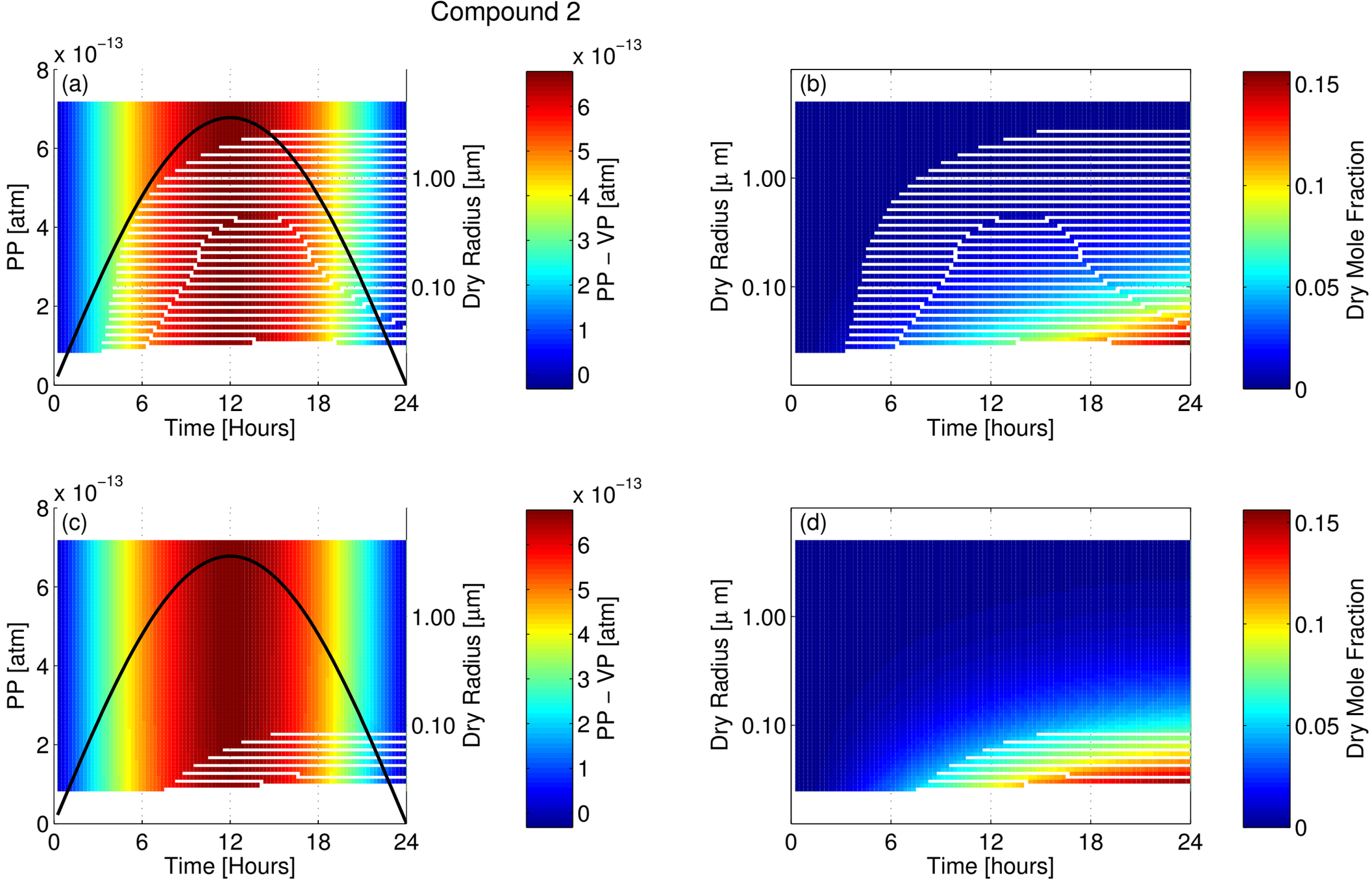
Vapour pressures and dry mole fractions for semi-volatile Compounds 1-13 and dry mole fractions for the involatile Compounds 14 and 15.

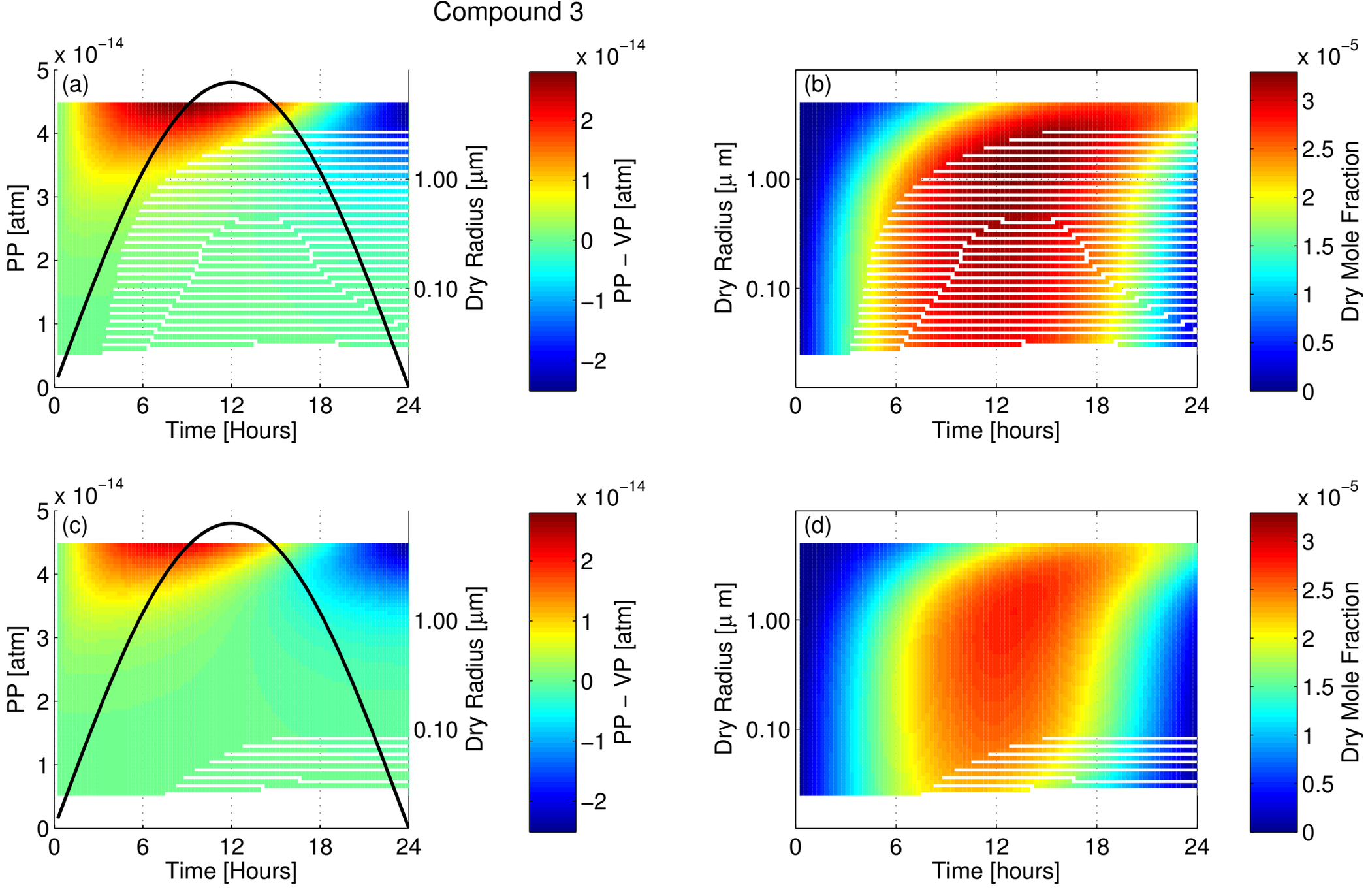
In all plots panels (a) and (b) are for the ideal model run, while panels (c) and (d) are for the non-ideal model run using PDFiTE.

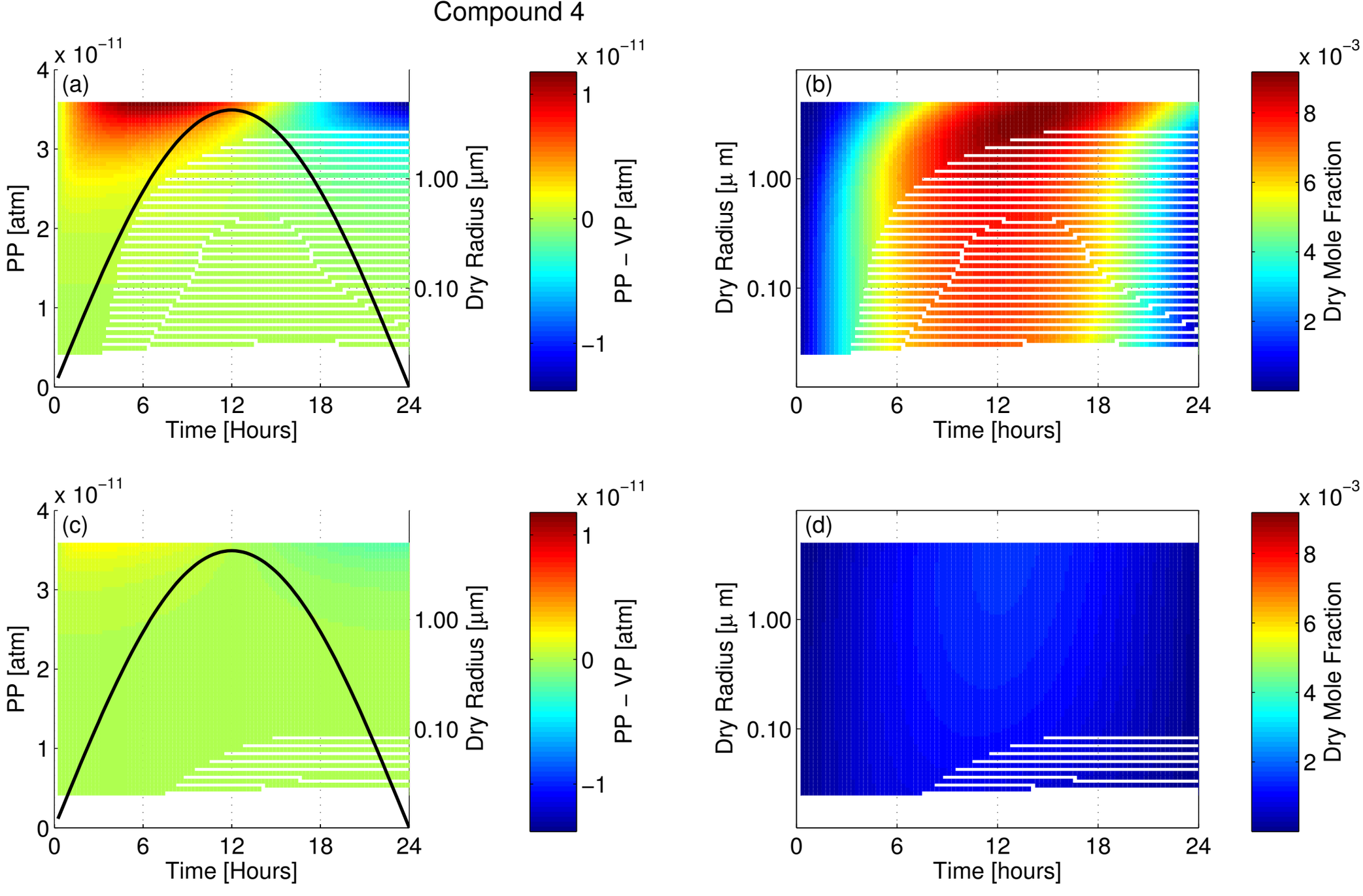
For Compounds 1-13 panels (a) and (c) show the size resolved differences between vapour pressure (VP) over the aerosol particle and the gas-phase partial pressure (PP), as well as the temporal variation in partial pressure. Panels (b) and (d) show the size resolved dry mole fractions of the SOA compounds within the condensed phase.

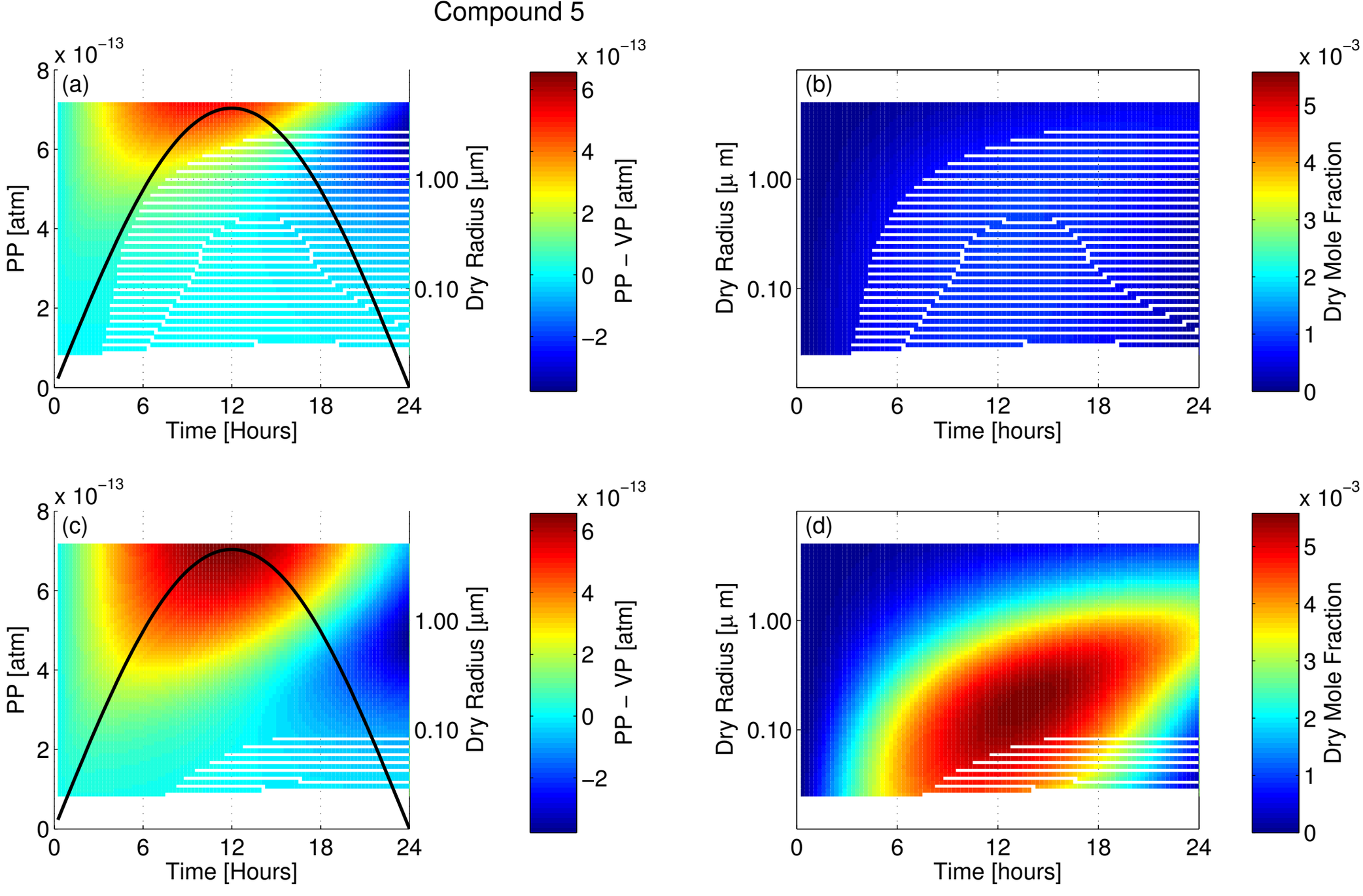
In the Compound 14 and 15 plot panels (a)-(d) all show size resolved dry mole fraction.


Compound 1
(a)
x 10^−9
PP [atm]
Dry Radius [µm]
x 10^−13
PP – VP [atm]
Time [Hours]
(b)
Dry Radius [µ m]
x 10^−4
Dry Mole Fraction
Time [hours]
(c)
(d)

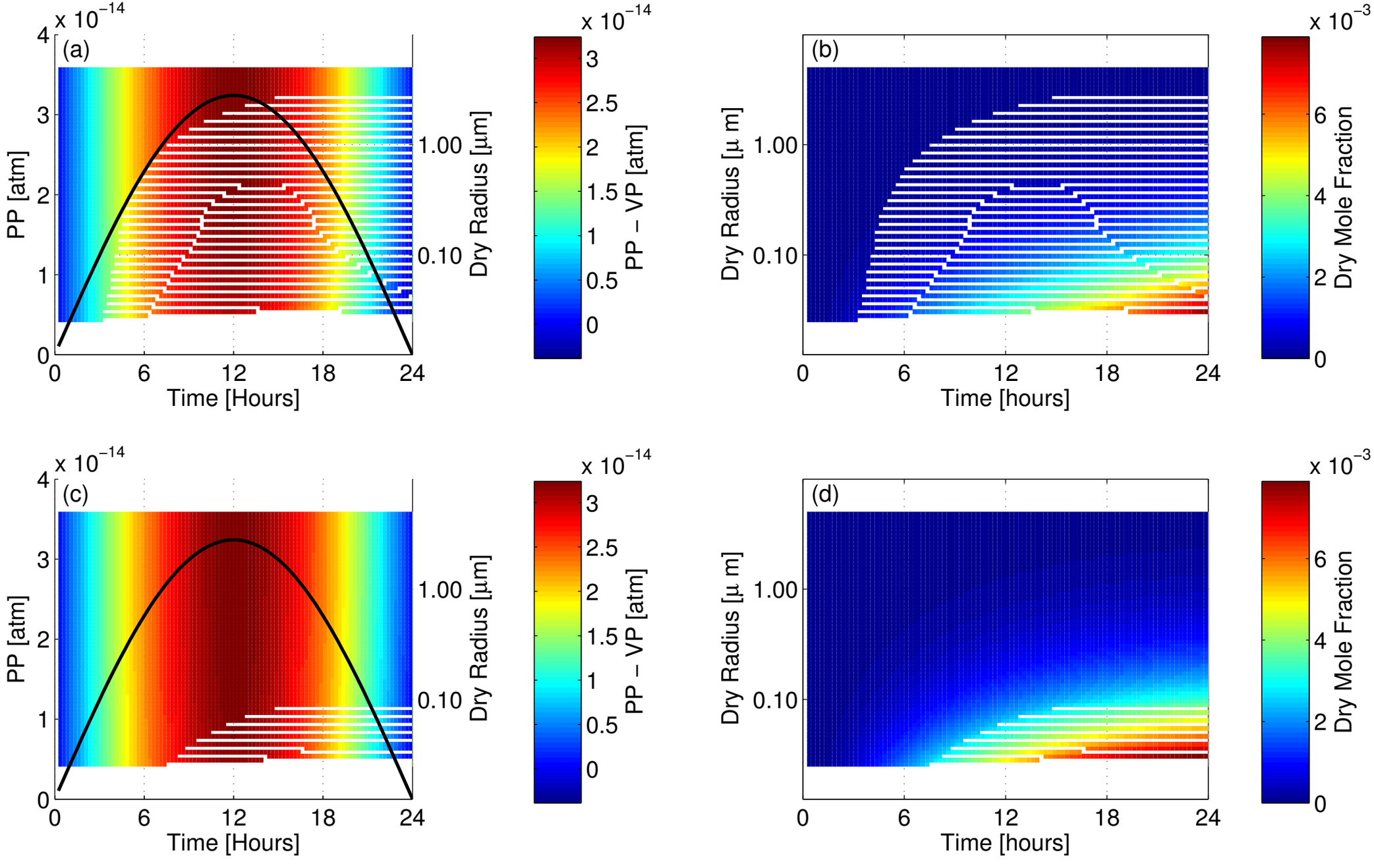

Compound 2
(a)
(b)
(c)
(d)
x 10^−13
PP [atm]
Dry Radius [µm]
PP – VP [atm]
Dry Mole Fraction
Time [Hours]
Time [hours]
0
6
12
18
24
0.10
1.00
0.05
0.1
0.15
1
2
3
4
5
8

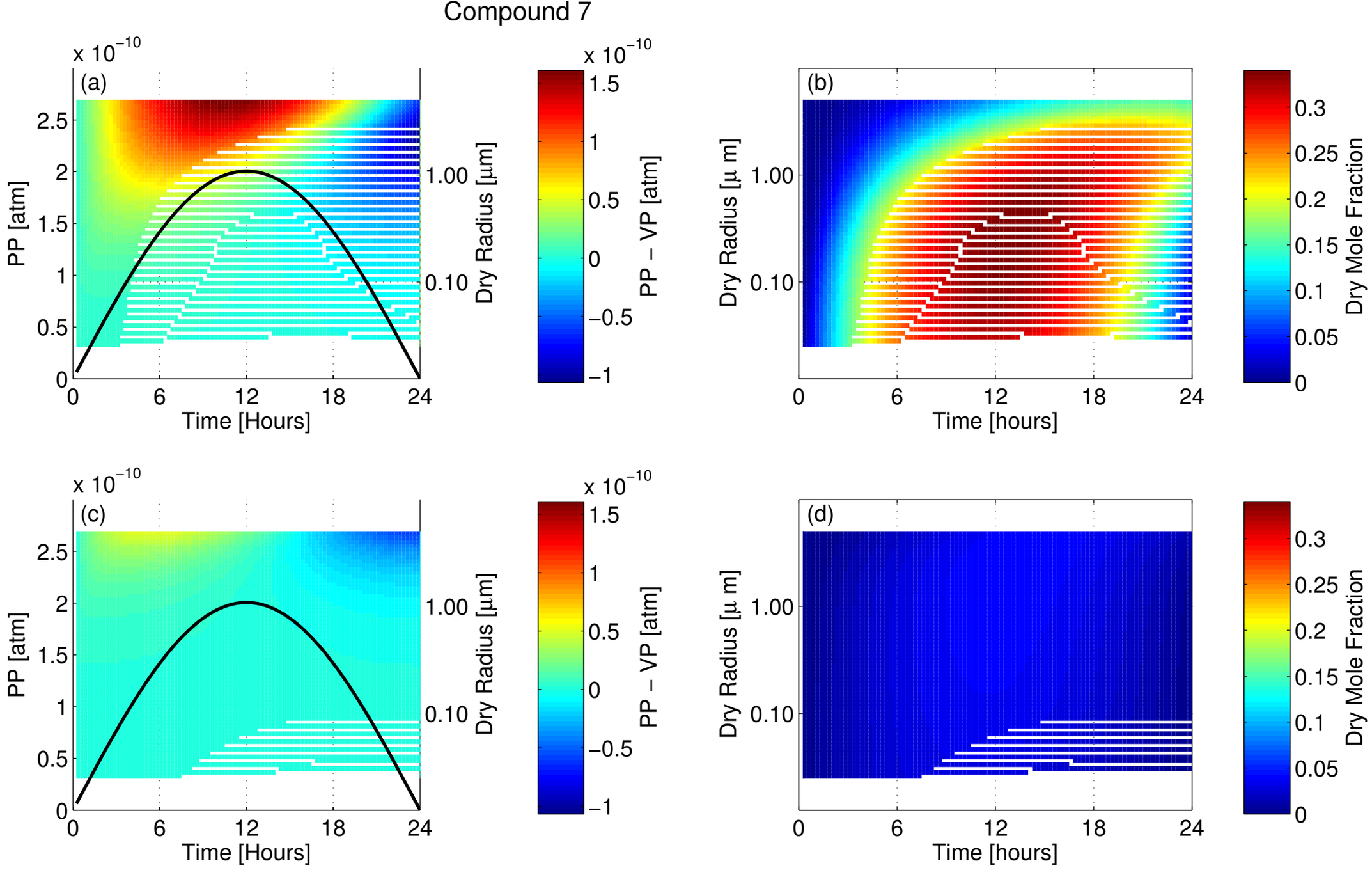

Compound 3
(a)
(b)
(c)
(d)
x 10^−14
x 10^−5
PP [atm]
Dry Radius [µm]
PP – VP [atm]
Dry Mole Fraction
Time [Hours]
Time [hours]
0
6
12
18
24
1.00
0.10
5
4
3
2
1
0
−1
−2
0.5
1.5
2.5

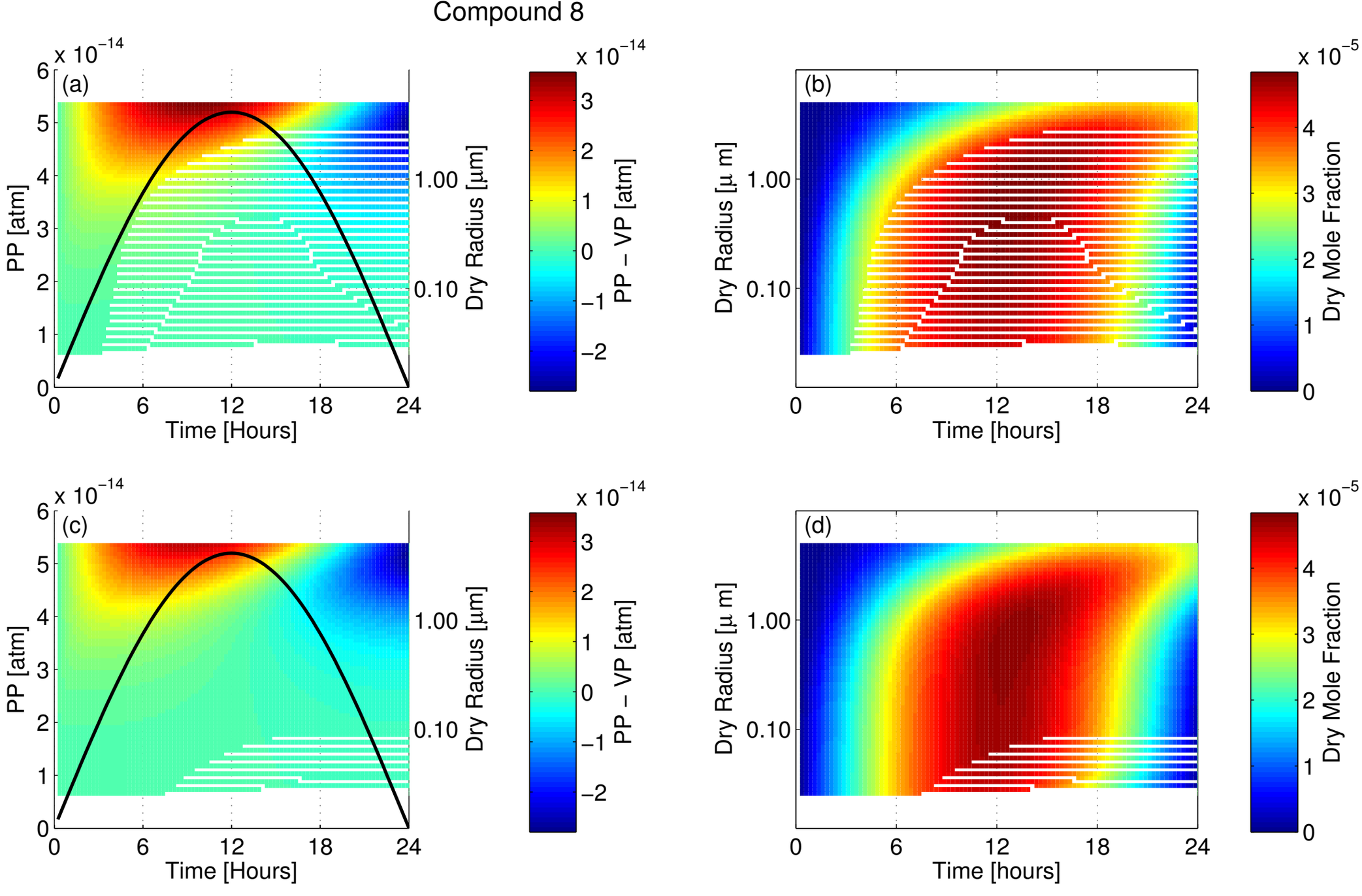

Compound 4
(a)
(b)
(c)
(d)
x 10^−11
PP [atm]
Dry Radius [µm]
PP – VP [atm]
Time [Hours]
Time [hours]
Dry Radius [µ m]
x 10^−3
Dry Mole Fraction
0
6
12
18
24
1.00
0.10
1
0
−1
2
4
8

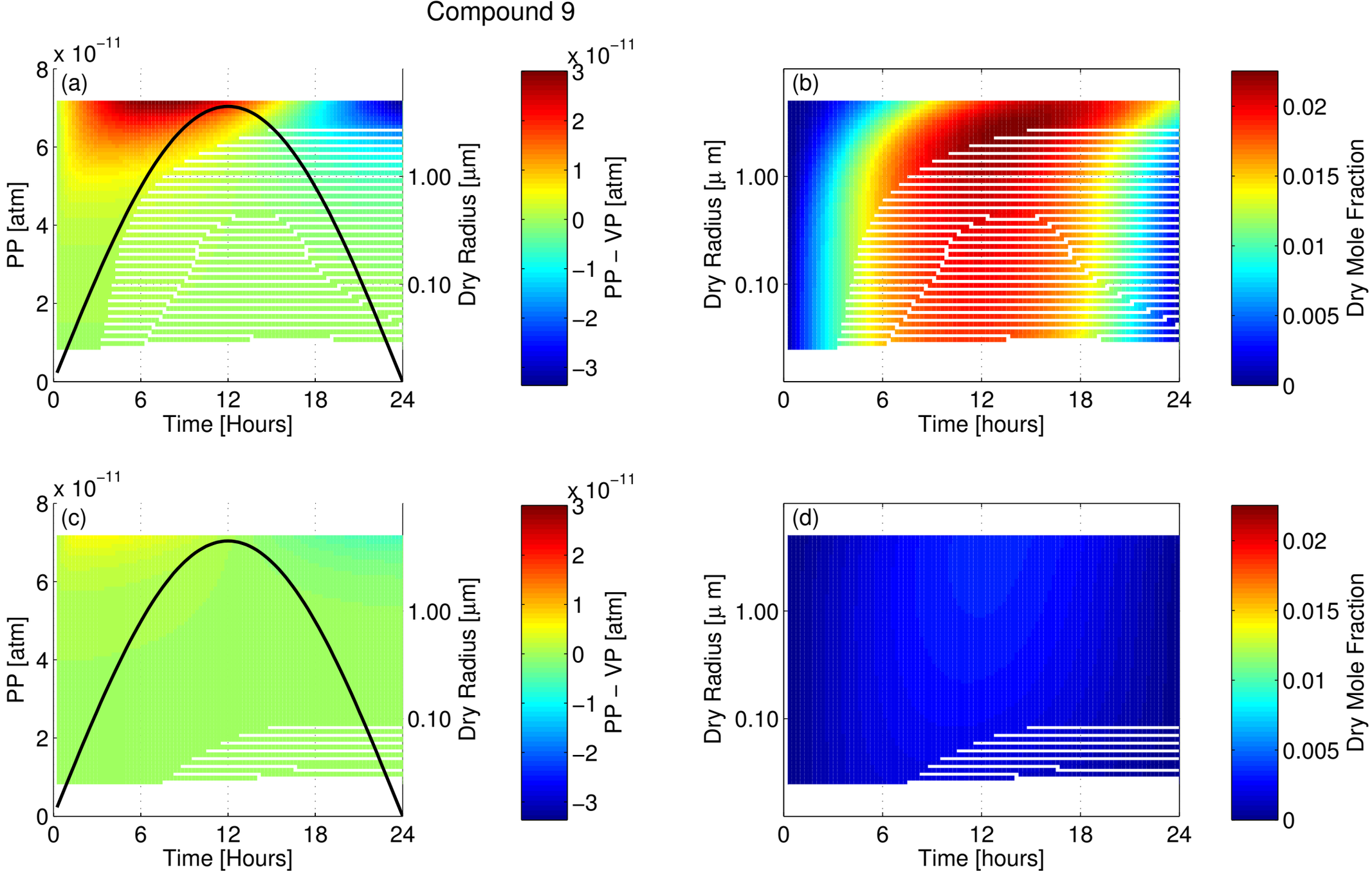

Compound 5
(a)
(b)
(c)
(d)
x 10^−13
PP [atm]
Dry Radius [µm]
PP – VP [atm]
Time [Hours]
Time [hours]
Dry Radius [µ m]
x 10^−3
Dry Mole Fraction

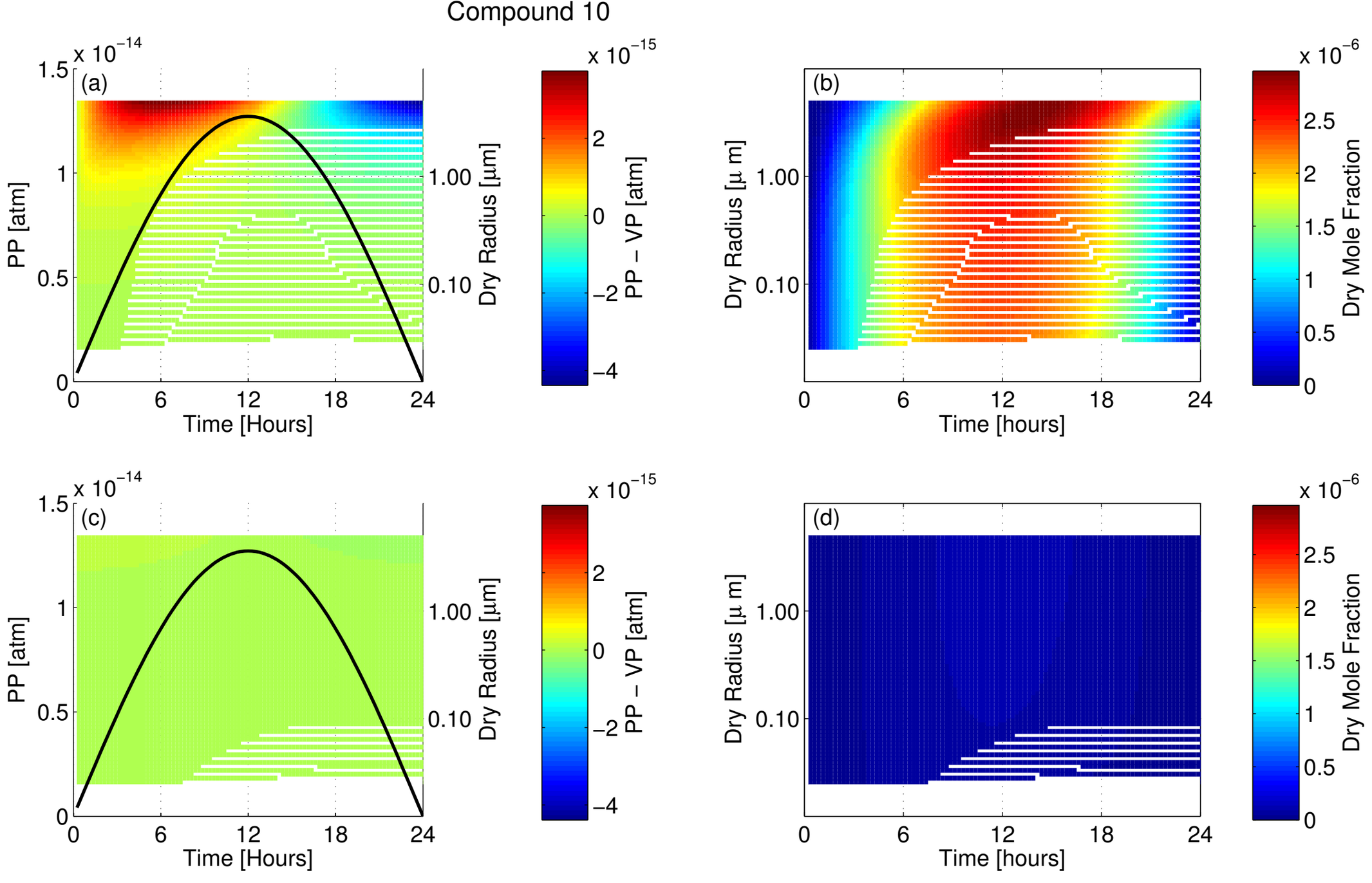
Compound 6

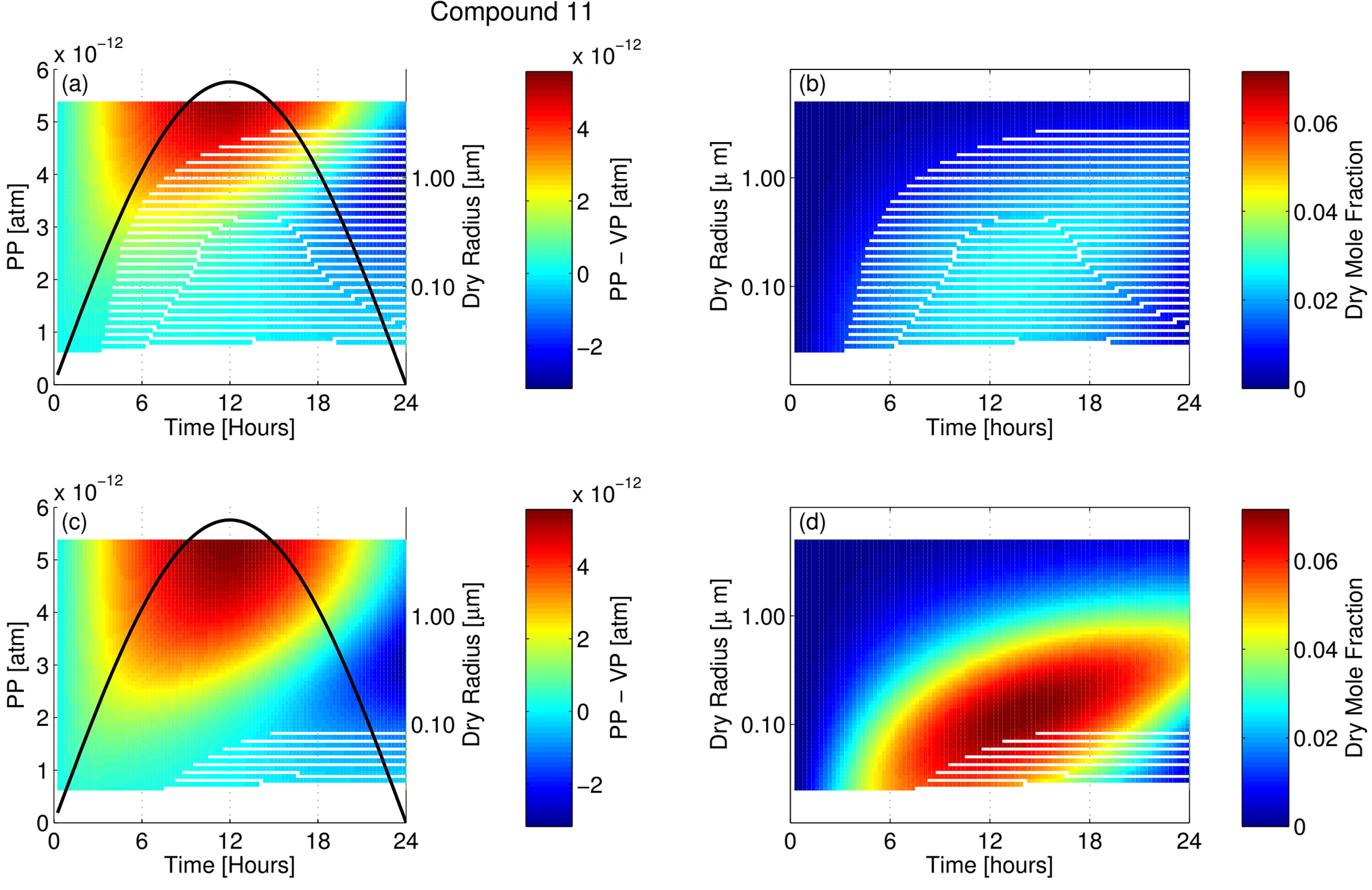

(a)
(b)
(c)
(d)
x 10^−14
PP [atm]
Dry Radius [µm]
PP – VP [atm]
Time [Hours]
Time [hours]
Dry Radius [µ m]
x 10^−3
Dry Mole Fraction

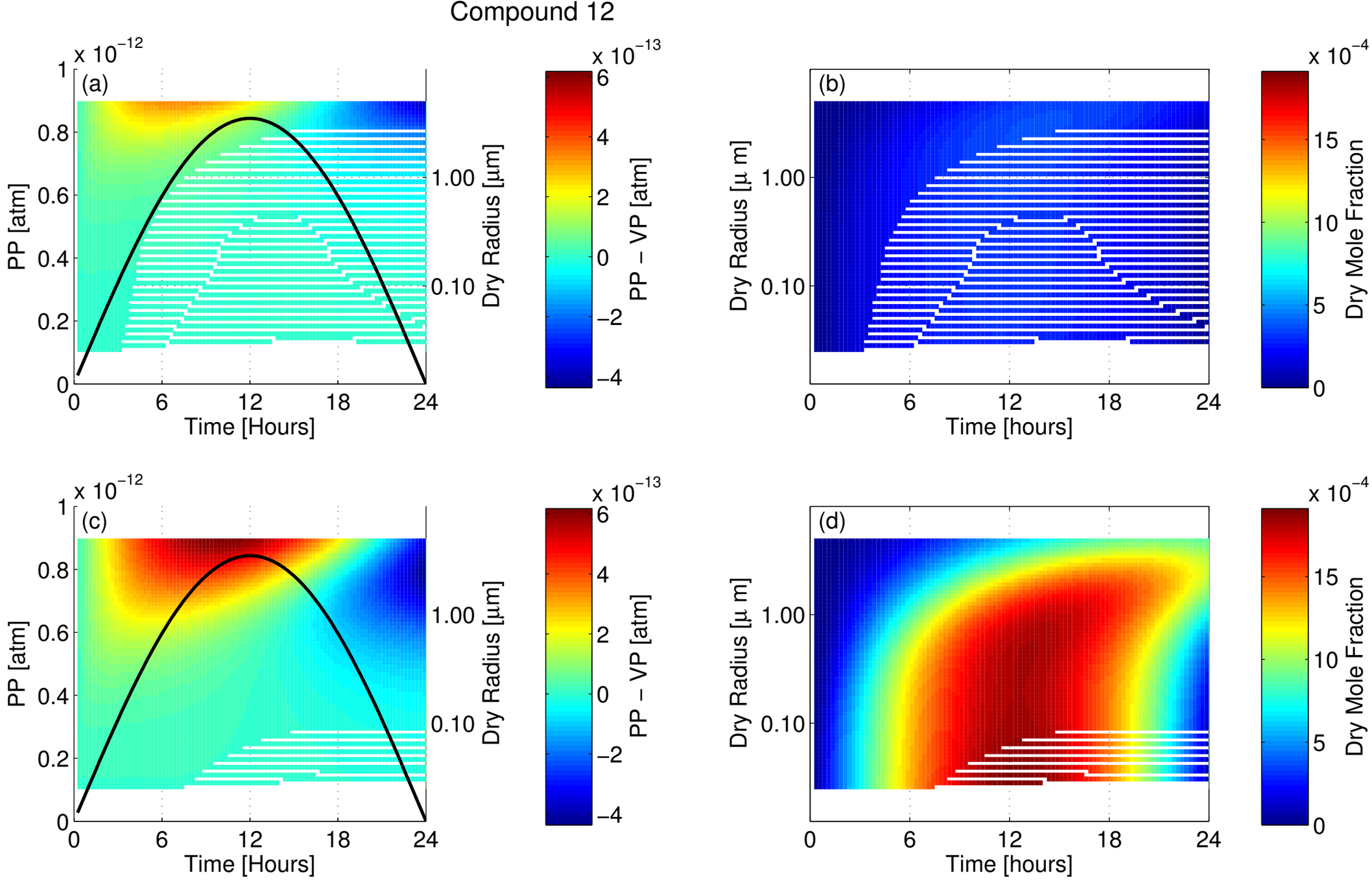

Compound 7
x 10^−10
(a)
PP [atm]
Dry Radius [µm]
Time [Hours]
PP – VP [atm]
(b)
Dry Radius [µ m]
Time [hours]
Dry Mole Fraction
(c)
(d)

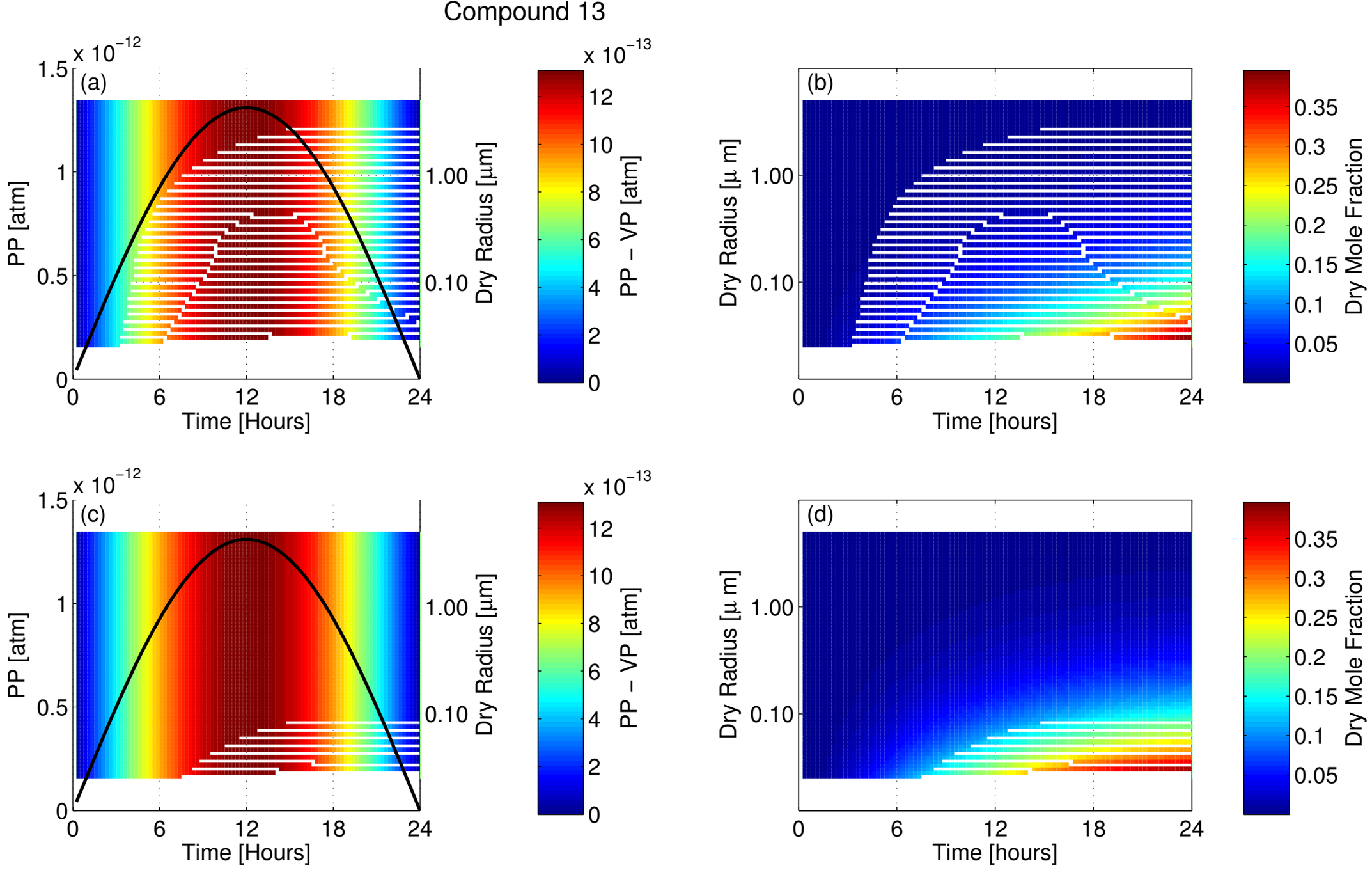

Compound 8
(a)
(b)
(c)
(d)
x 10^−14
PP [atm]
Dry Radius [µm]
PP – VP [atm]
Time [Hours]
x 10^−5
Dry Mole Fraction
Time [hours]

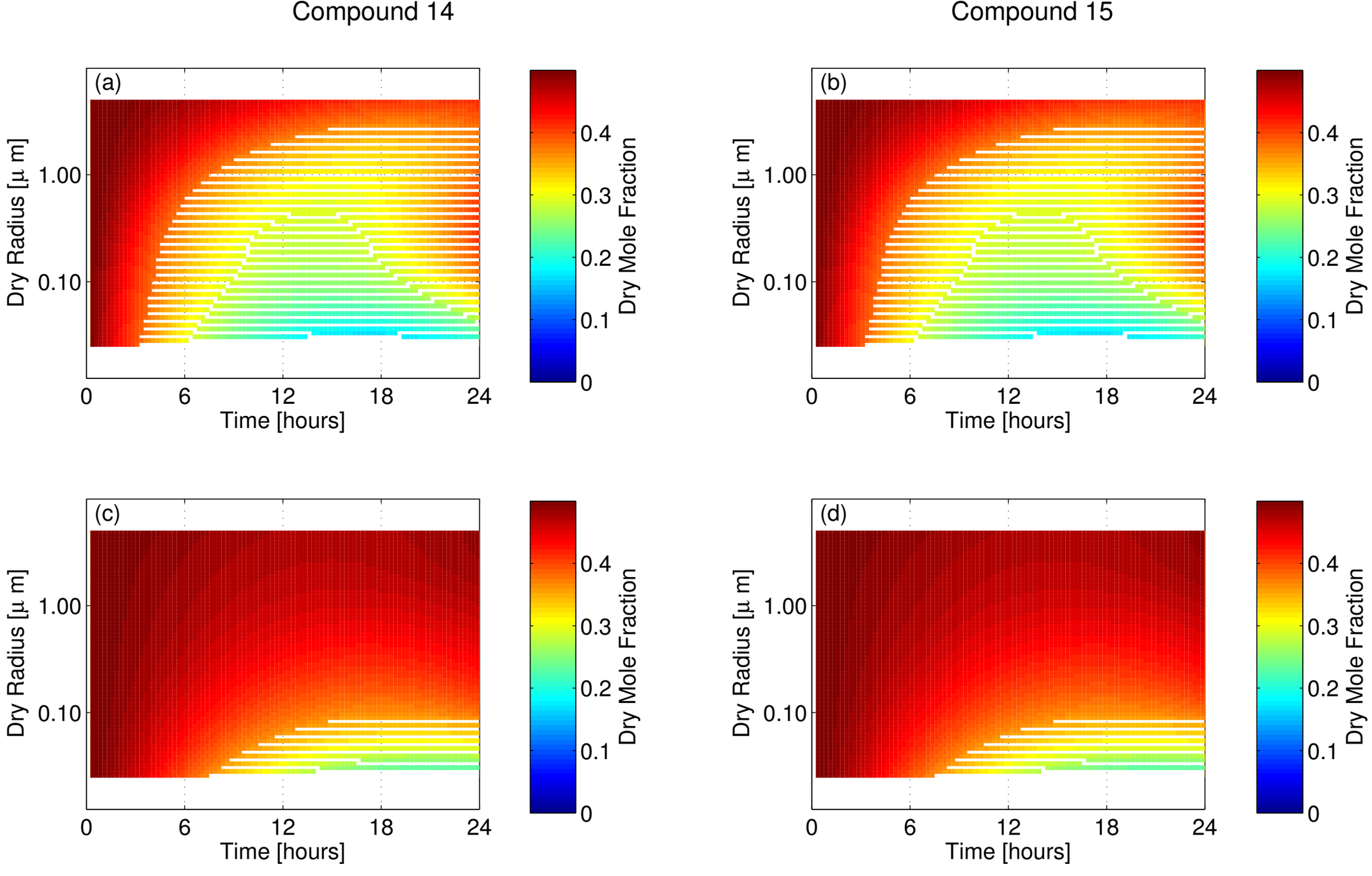

Compound 9
x 10^−11
(a)
PP [atm]
Dry Radius [µm]
PP – VP [atm]
Time [Hours]
(b)
Dry Radius [µ m]
Dry Mole Fraction
Time [hours]
(c)
(d)


Compound 10
(a)
(b)
(c)
(d)
x 10^−14
x 10^−15
x 10^−6
PP [atm]
Dry Radius [µm]
PP – VP [atm]
Dry Mole Fraction
Time [Hours]
Time [hours]
0
6
12
18
24
0.5
1
1.5
0.10
1.00
2
0
−2
−4
2.5


Compound 11
x 10^−12
(a)
PP [atm]
Dry Radius [µm]
PP – VP [atm]
Time [Hours]
(b)
Dry Radius [µ m]
Dry Mole Fraction
Time [hours]
(c)
(d)


Compound 12
(a)
x 10^−12
PP [atm]
Dry Radius [µm]
Time [Hours]
x 10^−13
PP – VP [atm]
(b)
Dry Radius [µ m]
Time [hours]
x 10^−4
Dry Mole Fraction
(c)
(d)


Compound 13
x 10^−12
(a)
1.5
1
0.5
0
PP [atm]
0
6
12
18
24
Time [Hours]
1.00
0.10
Dry Radius [µm]
x 10^−13
12
10
8
6
4
2
0
PP – VP [atm]
(b)
Dry Radius [µ m]
1.00
0.10
Time [hours]
0.35
0.3
0.25
0.2
0.15
0.1
0.05
Dry Mole Fraction
x 10^−12
(c)
x 10^−13
(d)


Compound 14
Compound 15
(a)
(b)
(c)
(d)
Dry Radius [µ m]
1.00
0.10
0
6
12
18
24
Time [hours]
0.4
0.3
0.2
0.1
0
Dry Mole Fraction

```
% This text file demonstrates the Matlab implementation of the organic
% PDFiTE model described in the main body of text.  Whilst not all
% parameters are displayed for brevity, examples are given with full
% references to equations described in the main body of text.

%The input variables are as follows

% 1) molecules | scalar variable : the number of compounds
% 2) ammount | array of size (molecules): the concentration of each
% compound [moles]
% 3) vapour_pure | array of size (molecules): the pure component vapour
%     pressure of each compound [atmospheres]
% 4) T | scalar variable : temperature [K]
% 5) RH | scalar variable : ambient relative humidity
% 6) high_order | scalar variable : a flag to determine if only the binary
%     PDFiTE model is used (value~=1) or the complete ternary model is used
%     (value=1)

%The output variables are as follows

% 1) water_content | scalar variable : the water content associated with
%     the mixture [moles]
% 2) vapour_pressures | array of size (molecules) : the equilibrium vapour
%     pressures of each compound above the mixed solution [atmospheres]
% 3) gamma | array of size (molecules) : the activity coefficients of each
% compound in the mixture.

function [water_content,vapour_pressures,gamma]=Matlab_pdfite_2ndorder(molecules,ammount,vapour_pure,T,RH,high_order)
%%------------------------------------------------------
%%------------- Non-persistent Variable initialisation------------
Mol_frac_w(1:molecules)=[];
log_gamma(1:molecules)=[];
gamma(1:molecules)=[];
```

```
vapour_pressures(1:molecules)=[];
F(1:molecules,1:molecules)=0;
diff_log_gammaw(1:molecules,1:molecules)=0;

%%---------------- Persistent Variable initialisation---------------
% These variables are determined using the automated fitting
% routine described in the main body of text
persistent Activity_coeffs %Polynomial coefficients for binary activity coefficients (the first term
%                           outside the summation in equation 13.
persistent Activity_coeffs_w2nd %Polynomial coefficients to account for ternary interactions (equation 14)
persistent Activity_coeffs_dry2nd % %Polynomial coefficients to account for ternary interactions (equation 15)
persistent Water_coeffs %Polynomial coefficients for use in ZSR mixing rule (equation 16)
persistent water_act_order %Polynomial order for calculating water content associated with each compound (equation 17)
persistent act_order %Polynomial order for calculating binary activity coefficients of each compound (equation 13)
persistent act_orderw2nd %Polynomial order to account for ternary interactions (equation 14)
persistent act_order_dry %Polynomial order to account for ternary interactions (equation 15)

if isempty(Water_coeffs) %initialise all above persistent variables.
    %Examples are now given based on the test case used within the main
    %body of text

    % ****************************************************************
    % Binary interaction matrices
    % Format: Water_coeffs(first compound,second compound)
    %           water_act_order(first compound)
    %           Activity_coeffs(first compound,second compound)
    %           act_order(first compound)

    % Compound number : 1 Polynomial order :3
    % Pearsons R : 0.999997 Sr :0.000001
    Water_coeffs(1,1)=0.698628;   Water_coeffs(1,2)=-1.323814;
    Water_coeffs(1,3)=1.613212;   Water_coeffs(1,4)=0.011863;
    water_act_order(1)=3;
```

```
% Compound number : 1 Polynomial order :2
% Pearsons R : 0.995218 Sr :0.000669
Activity_coeffs(1,1)=-0.680844;   Activity_coeffs(1,2)=-0.030474;
Activity_coeffs(1,3)=0.013327;   act_order(1)=2;

% ****************************************************************
% Ternary interaction matrices
% Format: Activity_coeffs_dry2nd(first compound,second compound,
%                                polynomial coefficient)
%          act_order_dry(first compound,second compound)
%          Activity_coeffs_w2nd(first compound,second compound,
%                                polynomial coefficient)
%          act_orderw2nd(first compound,second compound)

% Compound number: 1 Polynomial order 2nddry:2
% Pearsons R : 1.000000 Sr2nddry w:0.000000
Activity_coeffs_dry2nd(1,2,1)=-0.103695;   Activity_coeffs_dry2nd(1,2,2)=1.103537;
Activity_coeffs_dry2nd(1,2,3)=0.000080;   act_order_dry(1,2)=2;
% Compound number 2nddry: 1 Polynomial order 2nddry:2
% Pearsons R 2nddry: 0.999981 Sr2nddry w:0.000022
Activity_coeffs_dry2nd(1,3,1)=-0.177541;   Activity_coeffs_dry2nd(1,3,2)=1.174729;
Activity_coeffs_dry2nd(1,3,3)=0.001455;   act_order_dry(1,3)=2;
% Compound number w2nd: 1 Polynomial order w2nd:4
% Pearsons R w2nd: 0.996097 Sr2nd w:0.000059
Activity_coeffs_w2nd(1,2,1)=1.876666;   Activity_coeffs_w2nd(1,2,2)=-3.063020;
Activity_coeffs_w2nd(1,2,3)=1.346617;   Activity_coeffs_w2nd(1,2,4)=-0.273526;
Activity_coeffs_w2nd(1,2,5)=0.028828;   act_orderw2nd(1,2)=4;
% Compound number w2nd: 1 Polynomial order w2nd:4
% Pearsons R w2nd: 0.990553 Sr2nd w:0.000857
Activity_coeffs_w2nd(1,3,1)=7.016484;   Activity_coeffs_w2nd(1,3,2)=-10.467822;
Activity_coeffs_w2nd(1,3,3)=3.198646;   Activity_coeffs_w2nd(1,3,4)=0.207864;
Activity_coeffs_w2nd(1,3,5)=-0.434888;   act_orderw2nd(1,3)=4;
```

```
end

% A) Water content calculator, the ZSR mixing rule (equations 16–17)

 Nw_total=0;
 for water_step1=1:molecules
    if (ammount(water_step1) > 0.0d0)
        Mol_frac_w(water_step1)=0;
        for water_step2=1:water_act_order(water_step1)+1
            Mol_frac_w(water_step1)=Mol_frac_w(water_step1)+...
            (RH^(water_act_order(water_step1)-(water_step2-1)))*...
            Water_coeffs(water_step1,water_step2);
        end
        Nw(water_step1)=(-1*Mol_frac_w(water_step1)*...
        ammount(water_step1))/(Mol_frac_w(water_step1)-1);
        %  Total water content calculator
        Nw_total=Nw_total+Nw(water_step1);
        water_content=Nw_total;
    end
 end

% B1) Mole fractions (including water)
Solute_total=sum(ammount);
Mole_frac_w=Nw_total/(Solute_total+Nw_total);
mole_frac(1:molecules)=0;
for mole_frac_step=1:molecules
    if (ammount(mole_frac_step) > 0.0d0)
        mole_frac(mole_frac_step)=(ammount(mole_frac_step)/(Solute_total+Nw_total));
    end
end

% B2) Mole fractions (neglecting water)
mole_frac_dry(1:molecules)=0;
```

```
for mole_frac_dry_step=1:molecules
  if (ammount(mole_frac_dry_step) > 0.0d0)
     mole_frac_dry(mole_frac_dry_step)=(ammount(mole_frac_dry_step)/(Solute_total));
  end
end

% C1) Activity coefficients (binary –  first term outside of the summation in equation 13)

log_gamma(1:molecules)=0;
for act_step1=1:molecules
   if (ammount(act_step1) > 0.0d0)
      log_gamma(act_step1)=0;
      for act_step2=1:act_order(act_step1)+1
         log_gamma(act_step1)=log_gamma(act_step1)+...
         (Mole_frac_w^(act_order(act_step1)-(act_step2-1)))*...
         Activity_coeffs(act_step1,act_step2);
      end
   end
end

if (high_order==1)

% C2) Activity coefficients (ternary –  summation in equation 13)
for act_step1=1:molecules
   for act_step2=1:molecules
      if (act_step1~=act_step2)
         if (ammount(act_step1) > 0.0d0 && ammount(act_step2) > 0.0d0)
         %Now calculate dry mole fraction contributions
         for act_step3=1:act_order_dry(act_step1,act_step2)+1
            F(act_step1,act_step2)=F(act_step1,act_step2)+...
            (mole_frac_dry(act_step2)^(act_order_dry(act_step1,act_step2)-(act_step3-1)))*...
            Activity_coeffs_dry2nd(act_step1,act_step2,act_step3);
         end
```

```
            %Now calculate water mole fraction contributions
            for act_step3=1:act_orderw2nd(act_step1,act_step2)+1
               diff_log_gammaw(act_step1,act_step2)=diff_log_gammaw(act_step1,act_step2)+...
               (Mole_frac_w^(act_orderw2nd(act_step1,act_step2)-(act_step3-1)))*...
               Activity_coeffs_w2nd(act_step1,act_step2,act_step3);
            end
            end
         end
      end
      F_sum(act_step1)=sum(F(act_step1,:).*diff_log_gammaw(act_step1,:));
      log_gamma(act_step1)=log_gamma(act_step1)+F_sum(act_step1);
      gamma(act_step1)=exp(log_gamma(act_step1));
      vapour_pressures(act_step1)=vapour_pure(act_step1)*gamma(act_step1)*mole_frac(act_step1);
   end

   elseif (high_order~=1)
      for act_step1=1:molecules
         if (ammount(act_step1) > 0.0d0)
            gamma(act_step1)=exp(log_gamma(act_step1));
            vapour_pressures(act_step1)=vapour_pure(act_step1)*gamma(act_step1)*mole_frac(act_step1);
         end
      end
   end
   %*********************************************************************************
```