PRACTICA_15_Casas_Mercade

Table of Contents

Section A)

For the temperatures T = {0.99, 0.98, 0.97, . . . , 0.85}, use Newtons method to compute the coordinates vl(T) and vg(T), along with the corresponding pressure. When changing T, use the previous result as initial guess. For T = 0.99, use v(0) = 0.8 and v(0) = 1.2.

```
format long g;
close all;
clear all;

Ts = 0.99:-0.01:0.85; %Vector of temperatures

v0 = [0.8; 1.2]; %Initial guess for T=0.99

v = [v0]; %vector of volumes
V = []; %matrice that will contain the final solution for each
 temperature

xarxa = 0.3:0.005:5; %the volume points where the fucntion will be
 evaluated in order to plot the isotherm lines

colors = jet(length(Ts)); % Rainbow colors for the plot
colors = fliplr(colors); %We fliplr in order to have the red colors
 for highest temperatures and blue for the lowest
ii = 0;
figure(1);

for T = Ts
    %The functionP5 have the two equations that have to be solved
    funcioP5 = @(x)([log((3 * x(2) - 1) / (3 * x(1) - 1)) + 9 / (4 *
 T) * (1 / x(2) - 1 / x(1)) - 1 / (3 * x(2) - 1) + 1 / (3 * x(1) - 1),
 (8 * T) / 3 * (1 / (3 * x(2) - 1) - 1 / (3 * x(1) - 1)) - 1 / x(2)^2
 + 1 / x(1)^2]);
    vanDerWaals = @(vol)((8 * T ./ (3 .* vol - 1)) - (3 ./ (vol.^2)));
    ii = ii + 1;
    plot(xarxa, vanDerWaals(xarxa), 'Color', colors(ii, :)); %here we
 represent an isotherm line
    text(1.1, vanDerWaals(1.1), ['T = ', num2str(T)], 'FontSize',
 6); %It shows which isotherm is showing the plot
    axis([0.5, 3.5, 0, 1.5]);
    hold on;

    [XK, resd, it] = newtonn(v, 10^-8, 100, funcioP5); %we call
 newtonn to solve funcioP5
```

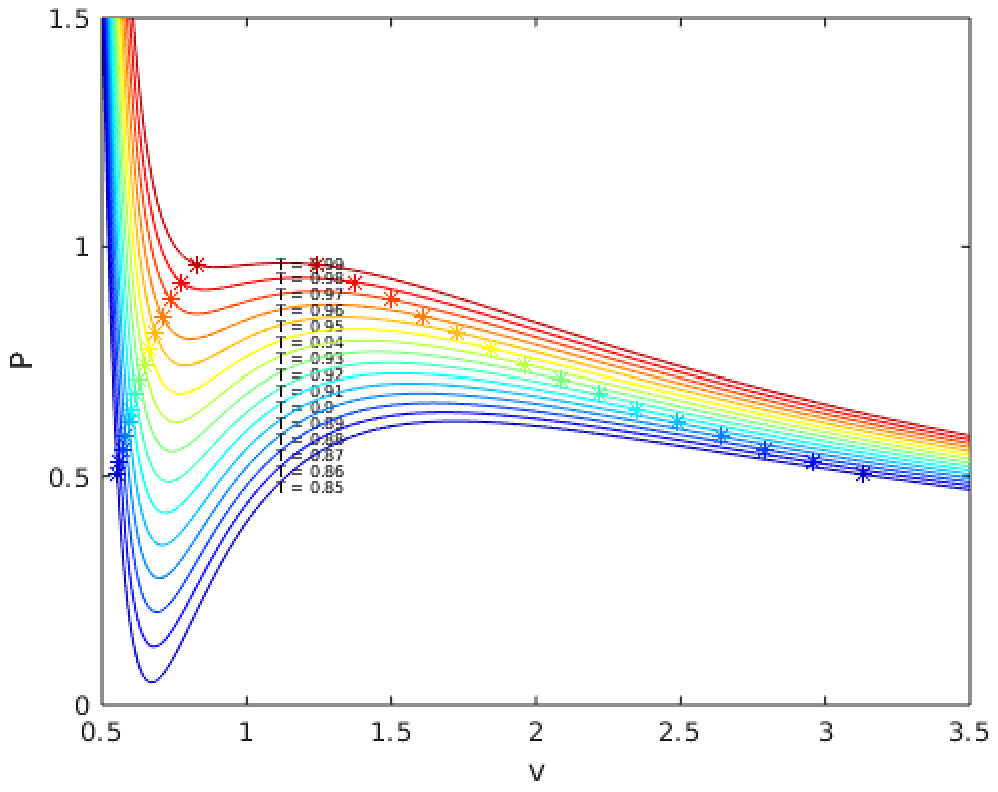
```
   v = XK(:, end); %We take the result to be the new initial guess
 for the next temperature
   V = [V, v]; %we save this result in the matrice

   plot(v(1), vanDerWaals(v(1)), '-*', 'Color', colors(ii, :)); %plot
 of the point (vl, pl)
   hold on

   plot(v(2), vanDerWaals(v(2)), '-*', 'Color', colors(ii, :)); %plot
 of the point (vg,pg)
   hold on

end

set(get(gca, 'XLabel'), 'String', 'v');  %description of each axis
set(get(gca, 'YLabel'), 'String', 'P');
hold off


disp('Values vl for each temperature')
V(1,2:end)'
disp('Values vg for each temperature')
V(2,2:end)'
disp('Pressure for each temperature')
(vanDerWaals(V(1,2:end)))'


%The newtonn code:
%{
function [XK, resd, it] = newtonn(x0, tol, itmax, fun)
   % This code is the newton method for nonlionear systems, is an
 iterative
   % method that allows you to approximate the solution of the system
 with a
   % presision tol

   %INPUTS:
   % x0 = initial guess  --> column or file vector (specify later)
   % tol = tolerance so that ||x_{k+1} - x_{k} | < tol
   % itmax = max number of iterations allowed
   % fun = @ function's name

   % OUTPUT:
   % XK = matrix where the xk form 0 to the last one are saved (the
 last
   % one is the solution) --> saved as columns
   % Resd = resulting residuals of iteration: ||F_k||, we want it to
 be 0,
   % as we are looking for f(x)=0
   % it = number of required iterations to satisfy tolerance

   %addpath('../Practica_1');
   % Atencio, pirmer comprobara a a la carpeta actual si hi son
```

```
    xk = [x0];
    XK = [x0];
    resd = [norm(feval(fun, xk))];
    it = 1;
    tolk = 1;

    while it < itmax && tolk > tol
        J = jaco(fun, xk); % Jacobia en la posicio anterior
        fk = feval(fun, xk);
        Dx = J\(-fk)';
        xk = xk + Dx;
        XK = [XK xk];
        resd = [resd, norm(fk)];
        tolk = norm(XK(:, end) - XK(:, end - 1));
        it = it + 1;


    end
%}

%The jaco code which gives the jacobian matrice:
%{
function DF = jaco(F,x)
% This code give you the Jacobian matrix of the function F evaluated
 in x.
% The Jacobian matrix is (n,m) meanwhile the sixe of F is n and the
 size of
% x is m.

% F son les n funcions escalars


% Input: F(x):R^m ---> R^n
       % x: (m x 1)-vector ; F: (n x 1)-vector
% Output: DF(x) (n x m) Jacobian matrix at x

f1=feval(F,x);    m=length(x);    n=length(f1);

h=sqrt(eps);  H=eye(m)*h;

DF = zeros(n,m);


for j=1:m

    f2=feval(F,x+H(:,j));

    DF(:,j)=(f2-f1)/h;
end

end
%}
```

Values vl for each temperature

```
ans =

         0.775538648215473
         0.737556218066331
         0.708189271482402
         0.684122113656141
         0.663692138223617
         0.645932202941631
         0.630224550945243
         0.616148134248055
         0.603401903178003
         0.591762523156609
         0.581059384405405
         0.571159003360017
         0.561954848373084
         0.553360458439842

Values vg for each temperature

ans =

          1.37610007703365
          1.49602772723826
          1.61180673020116
          1.72707119225589
          1.84383020863449
          1.96342844115086
          2.08689551163478
          2.21510240493274
          2.34884237620222
          2.48887796884135
          2.63597135621164
          2.79090600009133
          2.95450370035441
          3.12763929244118

Pressure for each temperature

ans =

         0.137962863756393
        0.0926518219613319
        0.0650957716170062
          0.05168769327764
        0.0505870008681581
        0.0607330990609407
        0.0814769975941632
          0.11241386332748
         0.153296573060965
         0.203986903491744
         0.264426094113388
         0.334616190755369
         0.414607771677167
```

```
0.504491649787488
```



Section B)

Repeat A) for the Dieterici's equation

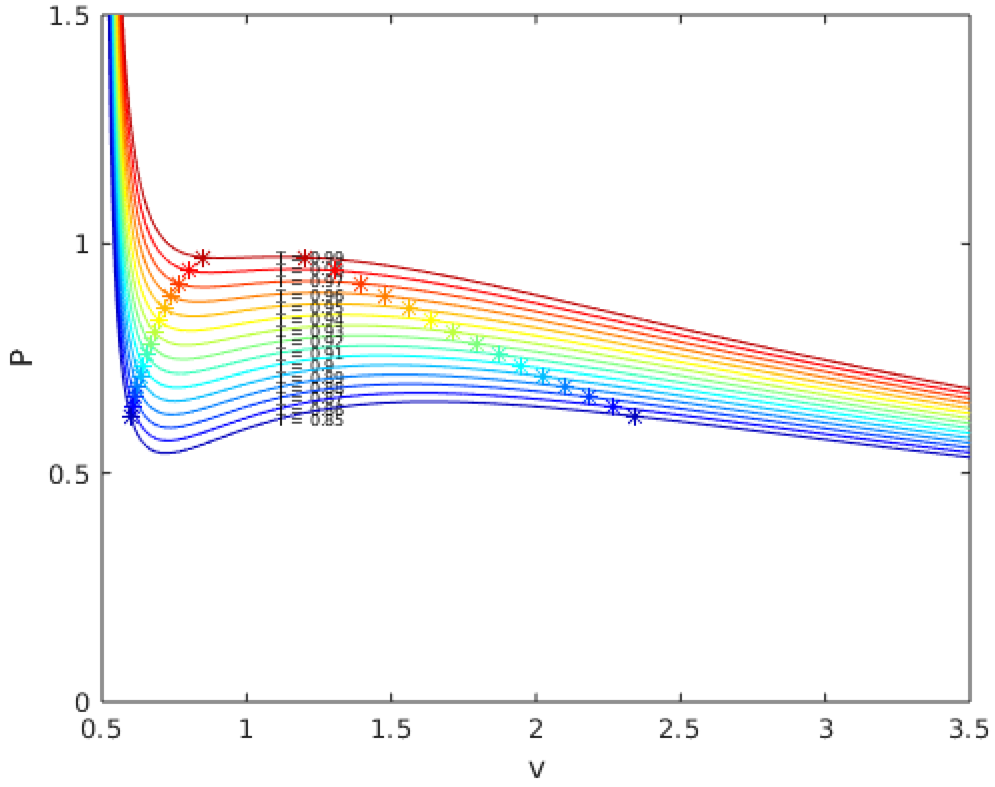
```
close all;
clear all;

Ts = 0.99:-0.01:0.85; %Vector of temperatures

v0 = [0.8; 1.2]; %Initial guess

% v = [v0]; %vector of volumes
V = []; %matrice that will contain the final solution for each
 temperature

xarxa = 0.3:0.005:5; %the volume points where the fucntion will be
 evaluated in order to plot the isotherm lines

colors = jet(length(Ts)); % Rainbow colors for the plot
colors = fliplr(colors); %We fliplr in order to have the red colors
 for highest temperatures and blue for the lowest
ii = 0;
figure(2);
```

```
addpath('../MNC1/PreExamen') %we call the clenshaw curtis code that we
 did last year


for T = Ts

    %Dieterici's equation
    dieterici = @(x)((T ./ (2 .* x - 1)) .* exp(2 - 2 ./ (x.* T)));

    %funcioGrossa is the vector that contains the two functions that
 we must
    %solve to find vl and vg. In the first position we have the
 function that
    %results from the demand of the Maxwell construction, which
 implies that
    %bpth areas I and II must have the sam evalue, that's why we use
 the
    %Clenshaw Curtis Code that we saw last year in MNC1. The function
    %introduced in the crenshaw code is the dietricie displaced
    %(p(v,T)-p(vl,T)) this way we can demand that using a=vl and b=vg
 the
    %resulting area must be 0. In the second position we have just
 rest the
    %dietricie equation (keeping constant de temperature) evaluated in
 vl and
    %in vg which must be also 0.

    N = 60;
    funcioGrossa = @(x) ([cuadratura_cc(x(1), x(2), N, @(v)(((T ./
 (2.* v - 1)).* exp(2 - 2 ./ (v.* T))) - ((T ./ (2.* x(1) - 1)).*
 exp(2 - 2 ./ (x(1).* T))))), dieterici(x(1)) - dieterici(x(2))]);

    ii = ii + 1;

    plot(xarxa, dieterici(xarxa), 'Color', colors(ii, :)); %plot of
 each isotherm line
    text(1.1, dieterici(1.1), ['T = ', num2str(T)], 'FontSize',
 6); %Text of each isotherm line
    axis([0.5, 3.5, 0, 1.5]);
    hold on;

    if T==0.99 || T==0.98
        v = v0;
 %for T=0.98 the initial guess must be the same used in T=0.99
 otherwise it does not work not only for this temperature but also for
 the followings
    else
        v=V(:,end);
    end

    [XK, resd, it] = newtonn(v, 10^-8, 60,funcioGrossa); %here as done
 in A) we solve using newton method
```

```
        v = XK(:, end);

        V = [V, v];


        plot(v(1), dieterici(v(1)), '-*', 'Color', colors(ii, :));
        hold on

        plot(v(2), dieterici(v(2)), '-*', 'Color', colors(ii, :));
        hold on

    end

    set(get(gca, 'XLabel'), 'String', 'v');  %description of each axis
    set(get(gca, 'YLabel'), 'String', 'P');
    hold off

    disp('Values vl for each temperature')
    V(1,2:end)'
    disp('Values vg for each temperature')
    V(2,2:end)'
    disp('Pressure for each temperature')
    (dieterici(V(1,2:end)))'

    %Clenshaw curtis code:
    %{
    function integral = cuadratura_cc(a, b, N, fun)
    %   Fer cuadratura Clenshaw-Curtis:
    %   La funcio entregada fun ha de ser capaÃ§ de tractar vectors si se
     li
    %   donen com arguments.

    j = [0:1:N];
    xcheb = cos(j.*pi./N);
    xk = a + ((b-a)./2).*(xcheb+1);
    fx = fun(xk);

    Wj = [];
    p = (1/((N^2)-1));
    for j=0:N
        if j==0||j==N
            Wj = [Wj p];
        else
            suma = 0;
            % n serà  sempre par per aixo podem dividir per 2.
            for k=0:N/2
                if k == 0 || k == N/2
                    suma = suma + (1/2)*(1/(1-4*(k^2)))*cos((2*pi*k*j)/N);
                else
                    suma = suma + (1/(1-4*(k^2)))*cos((2*pi*k*j)/N);
                end

            end
            Wj= [Wj (4/N)*suma];
        end
    end

    integral = Wj*fx';
    integral = ((b-a)/2)*integral;

    end
    %}
```

```
Values vl for each temperature

ans =

         0.799759611190289
         0.765007714839237
         0.738085516465666
         0.716033340120983
         0.697356284360387
         0.681179142065284
         0.666939618205985
         0.654252427164541
         0.642840743502614
         0.632498277423599
         0.623066781346448
         0.614421962245538
          0.60646426287909
         0.599112607104091

Values vg for each temperature

ans =

           1.3068531807897
          1.39692005091912
          1.48054092296002
          1.56071003660451
          1.63895776119743
          1.71619571477016
          1.79302358891112
          1.86986509830318
           1.9470365815422
            2.024784982541
          2.10331040671389
          2.18278027796525
          2.26333863410876
          2.34511246634669

Pressure for each temperature

ans =

          0.55269320903441
         0.546960412383566
         0.544199958368269
```

```
0.543665419854501
0.544978434751133
0.547926023587769
0.552384824910047
0.558286782556704
0.565601648051041
0.574327373356254
0.584484632845808
0.596113709361821
0.609272848661949
 0.62403760244963
```



Published with MATLAB® R2019b