Exercici_1_BVP_Casas_Mercade

Table of Contents

Section A) 1
Section B) 3

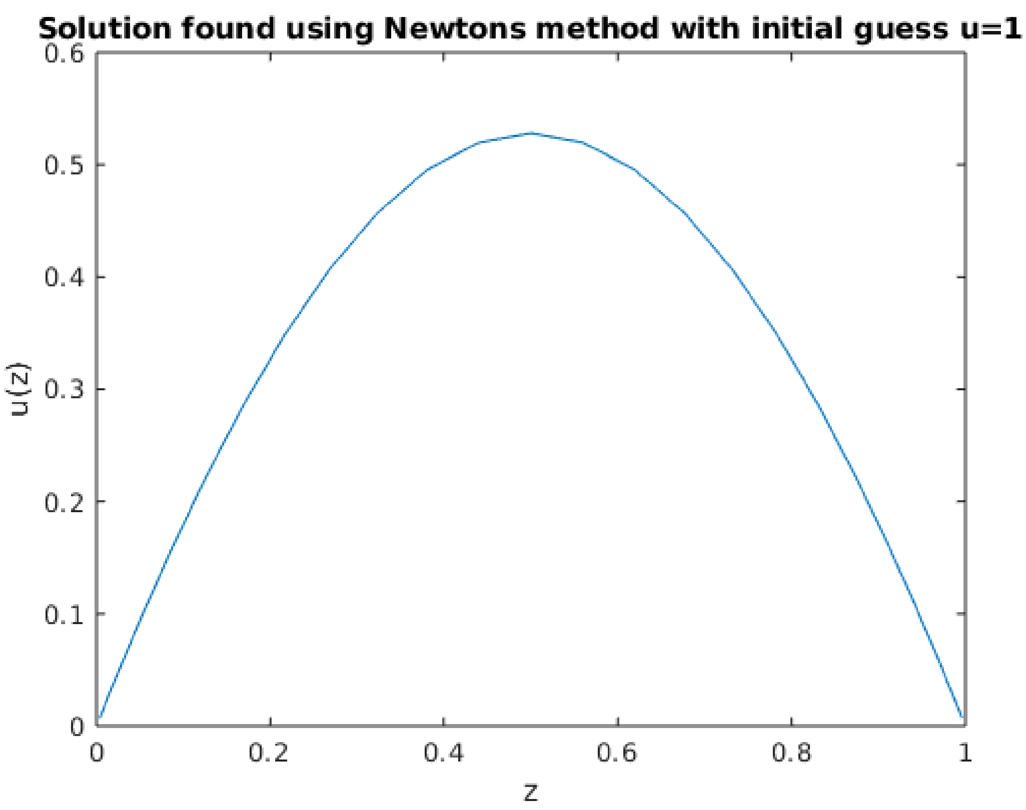
Non linear differential operator

Section A)

```
clc
clear all
close all

n = 26;
f0 = ones(n - 1, 1); %Initial guess
[XK, resd, it] = newtonn(f0, 1e-12, 100, @newtonFunction);

[D, x] = chebdiff(n, 0, 1);
plot(x(2:end - 1), XK(:, end));
title('Solution found using Newtons method with initial guess u=1')
xlabel('z')
ylabel('u(z)')

%Special norm defined for this problem, we will use on the next
 section
normSpecial = @(u)(sqrt(cuadratura_cc(0, 1, n - 2, u.^2)));
disp(normSpecial(XK(:, end)))

% The newton function is the discretization of the non-linar
 differential
% equation, once we have comupted it we apply the Newtons method using
 the intital guess given in the hint.
% Doing it we find one of the two solutions.

%{
function F = newtonFunction(f)

    p = @(x)(x .* 0 +1);
    q = @(x)(0 .* x + 0);
    r = @(x)(0 .* x + 0);
    n = 26;
    C = [1 0 0; 1 0 0];

    a = 0; b = 1;
    [M1, M2, M3, Lhat, x] = crearMatriusODE(n, C, p, q, r, a, b);
    mCoef = [C(2, 2), 0; 0, C(1, 2)];
    N = -exp(f + 1);
    F = (Lhat + M3 * inv(M2) * mCoef * M1) * f + M3 * inv(M2) * [C(2,
 3); C(1, 3)] - N;
```

```
end
%}

%{
function [D,x] = chebdiff(n,a,b)
z =cos([0:n]'*pi/n); d = [.5 ;ones(n-1,1);.5]; % Va de -1 a 1
D = zeros(n+1,n+1);
for ii = 0:n
    for jj = 0:n
        ir = ii + 1 ; jc = jj + 1;
        if ii == jj
            kk = [0:ii-1 ii+1:n]'; num = (-1).^kk.*d(kk+1) ;
            D(ir,jc) =((-1)^(ir)/d(ir))*sum(num./(z(ir)-z(kk+1)));
        else
            D(ir,jc) = d(jc)*(-1)^(ii+jj)/((z(ir)-z(jc))*d(ir));
        end
    end
end
D=2/(b-a) .* D;
x = a+(b-a)/2 * (z+1);% x son nodes Chebyshev amb 'allargats' a el
 domini del problema

%}
```

0.3932

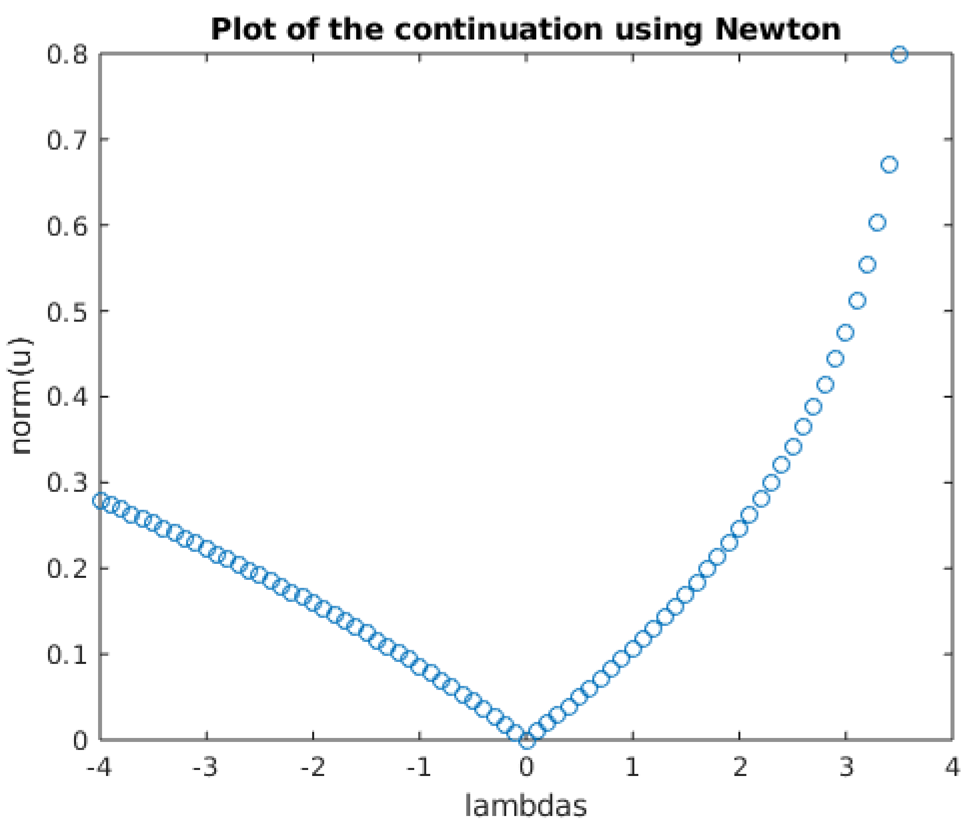


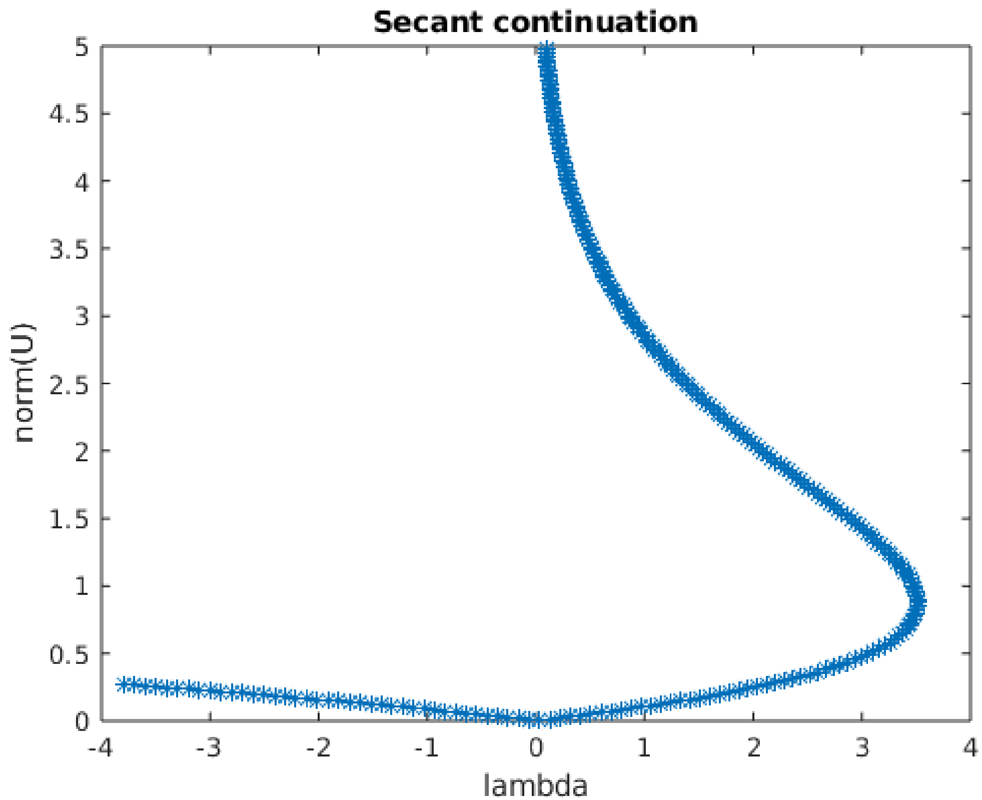
Section B)

```
clc
clear all;

%First we discretize the nonlinear ODE
n = 26; a = 0; b = 1;
p = @(x)(x .* 0 +1);
q = @(x)(0 .* x + 0);
r = @(x)(0 .* x + 0);
C = [1 0 0; 1 0 0];
mCoef = [C(2, 2), 0; 0, C(1, 2)];
[M1, M2, M3, Lhat, x] = crearMatriusODE(n, C, p, q, r, a, b);

% Special norm for this problem
normSpecial = @(u)(sqrt(cuadratura_cc(0, 1, n - 2, u.^2)));

U = [];
Norm = [];
f0 = ones(n - 1, 1);
lambda = [];
% First, we apply newton with the intial guess given and changing the
 lambdas in order to find
% another possibel soltuon, however there is a turning point that
 causes
% the fail of the natural continuation method. The values of the
 lambda will go form -4 to 3.5
% because for further values the newton does not compute well due to
 the turning point.
for i = [-4:0.1:3.5]
    F = @(f)((Lhat + M3 * inv(M2) * mCoef * M1) * f + M3 * inv(M2) *
 [C(2, 3); C(1, 3)] + i .* exp(f));
    [XK, resd, it] = newtonn(f0, 1e-10, 100, F);

    if XK(:, end) < 5
        U = [U XK(:, end)];
        f0 = XK(:, end);
        Norm = [Norm normSpecial(XK(:, end))];
        lambda = [lambda i];
    end

end

figure;
plot(lambda, Norm, 'o')
title('Plot of the continuation using Newton')
xlabel('lambdas')
ylabel('norm(u)')

% So we apply the secant continuation step method to solve this
 problem (as it was indicated on the problem).
% The two intial guess needed for this method will be the solutions
 found before for lambda=-4 and lambda=-4+epsilon
```

```
%To do the continuation step we set this parameters:
s = 1; itmax = 500; tol = 1e-10;
y0 = [U(:, 1); lambda(1)];
y1 = [U(:, 2); lambda(2)];

% We set the function with a dependence of the parameter lambda.
% Lambda will be stored on the last position of the independent vector
funLamb = @(f)((Lhat + M3 * inv(M2) * mCoef * M1) * f(1:end - 1) + M3
 * inv(M2) * [C(2, 3); C(1, 3)] + f(end) .* exp(f(1:end - 1)));
iterator = 0;
y = y0;
Y = [];
yy = [];

while -5 < y(end) && y(end) < 4 && iterator < itmax
    [y, iconv] = continuationStep(funLamb, y0, y1, s, tol, itmax);
    y0 = y1;
    y1 = y;
    Y = [Y, y];
    iterator = iterator +1;
    yy = [yy normSpecial(y(1:end - 1))];
end

figure;
plot(Y(end, :), yy, '*');
axis([-4 4 0 5]);
title('Secant continuation');
xlabel('lambda')
ylabel('norm(U)')
hold off


%{
function [y, iconv] = continuationStep(fun, y0, y1, s, tol, itmax)

%NORMA MODIFICADA PER LA PRACTICA 9!!!

    it = 1;
    tolk = 1;
    v = y1 - y0;
    yp = y1 + v * s; % Si s = 1 conseguim que la separaci# entre
 solucions sigui el maxim de "constant"
    xk = yp;
    XK = [];
    a=0; b=1;
    % A part de les ecuacions que teniem en el nnewton normal, li
    % imposareem que el preoducte escalar entre v i (xk(punt
    % buscat)- xk(predictor)) sigui 0
    n = length(y0)+1;
    normSpecial=@(u)(sqrt(cuadratura_cc(a, b, n-2, u.^2)));

    while it < itmax && tolk > tol
        J = jaco(fun, xk); % Jacobia en la posicio anterior
```

```
        J = [J; v'];
        fk = [fun(xk); v' * (xk - yp)]; % TODO: Copiat de teoria
        [P, L, U] = PLU(J);
        Dx = pluSolve(L, U, P, -fk); %Solucio de la ecuacio J*Dx = -fk

        %DxM = J\-fk;
        xk = xk + Dx;
        XK = [XK, xk];

        tolk = normSpecial(Dx); % Mirem la distancia entre el anterior
 i l'actual
        it = it + 1;
    end

    y = xk;

    %Retornem si convergeix o no per modificar la s si cal:
    if it <= itmax && tolk < tol
        iconv = 0; %OK
    else
        iconv = 1; %No em arribat a enlloc
    end

end
%}
```





Published with MATLAB® R2019b