Estudo e resolução de sistemas lineares e não lineares em compostos orgânicos.

Projeto I

Matheus Araujo Souza RA:184145

Supervisionado por

Giusepp Romanazzi

Assistant Professor

IMECC, instituto de Matemática Estatística e Computação Cientifica

Unicamp, Campinas, Brasil.

UNICAMP

Outubro de 2019

Universidade Estadual de Campinas (Unicamp)

SUMÁRIO

LISTA DE FIGURAS

Introdução

Nesse trabalho vamos analisar o desempenho de um conjunto de algoritmos para a resolução de sistemas lineares e não lineares, as aplicações desses métodos não se limitam apenas para os problemas propostos nesse projeto podemos encontrar inúmeras aplicações para os métodos aqui apresentados na física ou na Engenharia.

A ideia básica do primeiro projeto é a familiarização com métodos de busca de raízes reais de equações através do refinamento, onde podemos nos aproximar o tanto quanto possível de um valor desejado.

Objetivo

Consideramos três compostos orgânicos polares C1, C2, C3 que podem dissolver-se na água em três tempos diferentes, respectivamente t1, t2, t3 horas.
Sabemos também que quando os três compostos são separados, os tempos t1, t2 t3 >0 obedecem a relação

$$\mathrm{t}1^3 - 9t1 = -3$$
$$\log(t2)\, t2 = 3$$
$$2\sin\left(\left(\frac{3pi}{2}\right) t3\right) = 1 - t3$$

Quando os três compostos se juntam em uma cadeia de solubilidade T da cadeia completa resulta ser o máximo de t1, t2, t3 que satisfazem as seguintes relações

$$t1^3 - 9t1 = 4 + 6t2$$
$$\log(t2)\,(t2 - t1) = 3(t3 - t2) - 1$$
$$2\sin\left(\left(\frac{3pi}{2}\right)(t3 - t2)\right) = t1 - t3$$

Nosso objetivo é resolver o primeiro problema encontrando os valores para ti para todo i=1:3 utilizando três métodos diferentes para resolução de sistemas lineares, sempre buscando o menor cuspo computacional possível e a otimização de processos computacionais. Na segunda parte vamos implementar um método de Newton para a resolução do sistema não linear.

Analise de dados/ código I

Metodologia

Implementamos o método da bisseção para resolver a primeira função.

$$t1^3 - 9t1 = -3$$

O objetivo do método da bisseção é reduzir a amplitude do intervalo [a, b] que contem a raiz até contermos uma precisão requerida (b - a) < e. Seja a função f(x) continua no intervalo [a, b] e tal que f(a)f(b)<0.

Por ser um polinômio não precisamos analisar a continuidade em um determinado intervalo todos os polinômios são contínuos para toda a reta dos reais.

Uma maneira de fazer a escolha dos pontos iniciais é olhando para o gráfico da função, podemos ver que os pontos escolhidos para a aplicação desse método [0, 1] respeitam a propriedade estabelecida da imagem de ambos terem sinais contrários. Em alguns casos mais específicos temos que fazer a analise do sinal da derivada da primeira e da segunda na função, com isso podemos verificar se existe apenas um zero no intervalo delimitado.

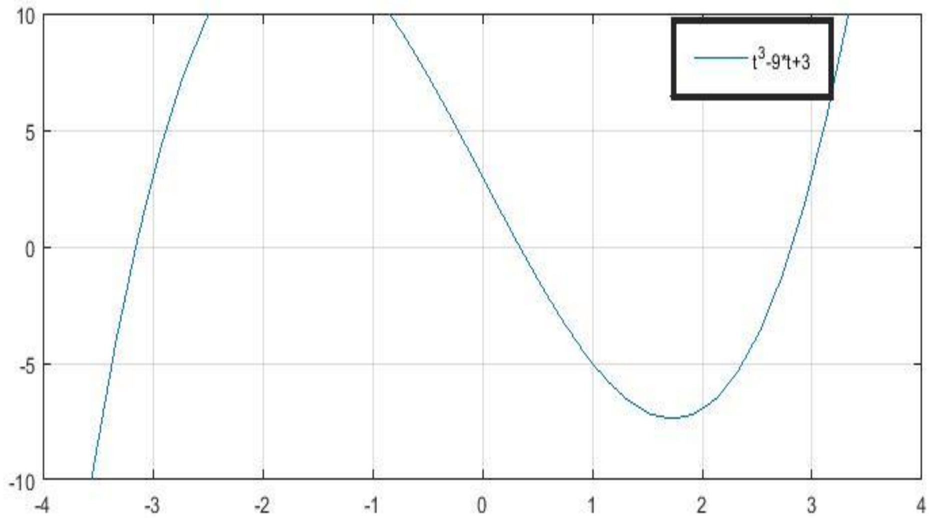

Figura 1.0

Código

Segue abaixo o código implementado para a resolução do problema, nele foram feitos alguns comentários e ideias de melhora que estavam sendo executadas no momento do seu desenvolvimento.

O ambiente de desenvolvimento integrado de código aberto e multiplataforma que foi escolhido para a criação dos códigos foi o matlab2018.

```
                         % Método da Bisseção

%Não precisamos fazer a análise de convergência dessa função, estamos
%trabalhando com polinómios eles são contínuos em toda reta real

a = 0; % Limite inferior do intervalo
b = 1; % Limite superior do intervalo
% f(t)=t^3-9t=-3 primeira função que vamos avaliar

    f = 't^3-9*t+3';
    e1 = 10^-3; % Cota de erro absoluto máxima e para |f(t)|


% Cálculos Iniciais
t = (a+b)/2; % Aproximação inicial
% erro absoluto inicial vamos trabalhar sempre com uma modificação do
erro
% absoluto inicial assim podemos fazer a análise antes e depois da
troca de
% dados

EA = abs(b-a);


k = 0; % Contador de iterações
g = abs(eval(f)); % Módulo de f(t) na raiz aproximada inicial


%vamos também calcular o número de interações
%dado meu a e b inicial e meu e1 podemos fazer uma aproximação para o
%número de interações totais do nosso método intKBi=Log(na base
2)(E0/e1)
E0 = b-a;
intKBi=log2(E0/e1);

%assim conseguimos estimar um número exato de interações

fprintf(1,'%s %2d %s %12.9f %s %12.9f %s %12.9f\n','k =',k,...
' t =',t,' f(t) =',eval(f),' EA|(b-a)| =',EA);

%aqui já estamos prontos para começar nosso loop vamos usar como
critério
%de parada vamos usar
           %|(b-a)| > e1 && |f(t)| > e1 && k < Log2(E0/e1)

               while EA > e1 && g > e1 && k < intKBi
                            t = a;
              fa = eval(f); %calculo o valor do meu fa usando a função
eval
                             t = b;
              fb = eval(f); %calculo o valor do meu fb usando a função
eval
                             t = (a+b)/2;
                             ft = eval(f);
%comando que testa sempre se a imagem da minha função está em
```

```
              %um intervalo certo.

                              if fa*ft < 0
                                 b = t;
                              else
                                 a = t;
                              end
                              k = k+1;
t = (a+b)/2; % Nova aproximação da raiz
EA = abs(b-a); %erro absoluto
g = abs(eval(f)); % Módulo de f(t) na nova aproximação da raiz
fprintf(1,'%s %2d %s %12.9f %s %12.9f %s %12.9f\n','k =',k,...
' t =',t,' f(t) =',eval(f),' EA|(b-a)|=',EA);
end
```

Análise do Código

Quando estamos trabalhando com polinômios existe uma certa dificuldade na atualização dos dados internamente nele, o comando eval acabou possibilitando o contorno desse problema trabalhando com polinômios em forma de strings, ele sempre muda os parâmetros por referência comparando os elementos contidos na string com um inteiro ou double, que tem o nome ou letra de um elemento que também está contido nela.

Análise de Dados

A convergência desse método é muito lenta, satisfeitas as hipóteses de convergência ele vai chegar no resultado que estávamos desejando, e para qualquer tipo de função satisfeitas as regras de convergência e da escolha do intervalo ele vai chegar, mesmo que seja em n interações.

```
Command Window
>> Bissecao
k =  0  t =  0.500000000  f(t) = -1.375000000  EA|(b-a)| =  1.000000000
k =  1  t =  0.250000000  f(t) =  0.765625000  EA|(b-a)|=  0.500000000
k =  2  t =  0.375000000  f(t) = -0.322265625  EA|(b-a)|=  0.250000000
k =  3  t =  0.312500000  f(t) =  0.218017578  EA|(b-a)|=  0.125000000
k =  4  t =  0.343750000  f(t) = -0.053131104  EA|(b-a)|=  0.062500000
k =  5  t =  0.328125000  f(t) =  0.082202911  EA|(b-a)|=  0.031250000
k =  6  t =  0.335937500  f(t) =  0.014474392  EA|(b-a)|=  0.015625000
k =  7  t =  0.339843750  f(t) = -0.019343913  EA|(b-a)|=  0.007812500
k =  8  t =  0.337890625  f(t) = -0.002438627  EA|(b-a)|=  0.003906250
k =  9  t =  0.336914063  f(t) =  0.006016918  EA|(b-a)|=  0.001953125
k = 10  t =  0.337402344  f(t) =  0.001788904  EA|(b-a)|=  0.000976563
fx >>
```

Figura 1.1- exibição dos resultados obtidos na execução do código.

Analise de dados/ código II

Metodologia

Para a segunda função implementamos o método da falsa posição.

$$\log(t2)\, t2 = 3$$

O método da falsa posição pode ser considerado como uma variação do método da falsa posição ou uma variação do método da secante, como vimos o método da bisseção tem uma convergência lenta porque ele pega sempre o ponto médio do nosso intervalo, então agora vamos considerar uma média ponderada entre a e b com os pesos |f(a)| e |f(b)|.

$$x = \frac{a|f(b)| + b|a|}{|f(a)| + |f(b)|}$$

Visto que f(a) e f(b) tem sinais opostos.
Graficamente, esse ponto x é a intersecção entre o eixo ox e a reta r que passa entre (a, f(a)) e (b, f(b)).

Gráfico Y vs T

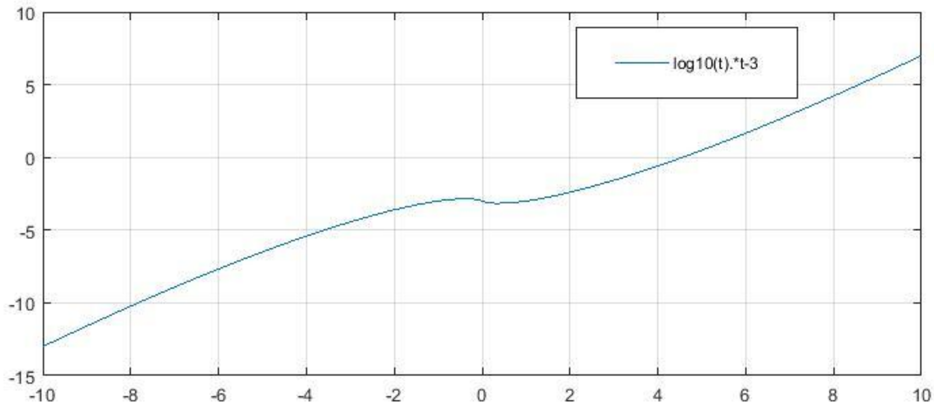

Figura 1.2

Código

Segue abaixo a implementação do código da falsa posição.

```
                    %método da falsa posição

 %o melhor ponto é entre 1 e 3 conseguimos ver isso pela plotagem do
 %gráfico também podemos analisar isso pelas derivadas que existem uma
 %única no intervalo, mas vamos deitar isso para o relatório
% Parâmetros
a = 4; % Limite inferior do intervalo
b = 5; % Limite superior do intervalo
```

```
%analisamos a função graficamente logo esses pontos iniciais respeitão
a
%convergência da função com logaritmo
f = 'log10(t).*t-3';
e1 = 10^-3;
  % Cálculos Iniciais
t = a;
fa = eval(f);
t = b;
fb = eval(f);
%Temos aqui que nossa primeira aproximação da reta do método da falsa
%posição
t = (a*fb-b*fa)/(fb-fa);
%aqui temos a imagem do próximo ponto xk encontrado no nosso método
%que busca sempre a intersecção com o eixo das abscissas
ft = eval(f);
k=0;
fprintf(1,'%s %2d %s %12.9f %s %12.9f %s %12.9f\n','k =',k,...
' t =',t,' f(t) =',ft,'|b-a| =',abs(b-a));
  %aqui vamos usar apenas dois critérios de parada como foi passado em
aula
  %poderíamos usar até mais como diz no livro da Márcia A.gomes
  %temos então |f(t)| > e1 && |(b-a)| >e1
              while abs(ft) > e1 && abs(b-a) >e1

                           k = k+1;
                           if fa*ft < 0
                                    b = t;
                           else
                                    a = t;
                           end
                            t = a;
                            fa = eval(f);
                            t = b;
                            fb = eval(f);
                            t = (a*fb-b*fa)/(fb-fa);
                            ft = eval(f);
fprintf(1,'%s %2d %s %12.9f %s %12.9f %s %12.9f\n',...
    'k =',k,' t =',t,' f(t) =',ft,' |b-a|=',abs(b-a));
end
```

Análise do Código

Podemos ver que na implementação do código não ocorreram muitas mudanças nos critérios de parada entre o método da bisseção e da falsa, analisamos sempre o f(t) e o(b - a) com relação ao erro e. Quanto a convergência escolhemos pontos não conflituosos com as derivadas, mas isso também pode ser verificado na análise gráfica, segue o critério de convergência.

Se f(x) é continua no intervalo [a, b] com f(a)f(b)<0 então o método da posição falsa gera um sequencia convergente.

Análise de Dados

Quando olhamos os resultados finais que obtemos fica claro que o método é mais rápido do que o da falsa posição, mas não podemos esquecer que estamos trabalhando com uma função diferente e iniciamos em pontos diferentes, antes de executar o programa buscamos sempre aproximar os pontos iniciais da raiz, e sempre verificamos se existia apenas uma única raiz no intervalo.

```
Command Window
>> falsaP
k =  0  t =  4.544592820  f(t) = -0.011953044 |b-a| =  1.000000000
k =  1  t =  4.555333682  f(t) = -0.000220775  |b-a|=  0.455407180
>>
```

Figura 1.3

Analise de dados/ código III

Metodologia

Buscando a raiz da terceira função.

$$2 \sin\left(\left(\frac{3pi}{2}\right) t3\right) = 1 - t3$$

Um dos métodos se não o mais famoso dentre aqueles que são utilizados para a resolução de sistemas lineares é o Método de Newton Raphson, que consiste de uma tentativa de acelerar a convergência do método do ponto fixo, olhando para um lado mais geométrico ele pode ser obtido facilmente através da equação da reta.

$$\mathbf{x}_{K+1} = \mathbf{x}_K - \frac{\mathbf{f}(\mathbf{x}_K)}{\mathbf{f}'(\mathbf{x}_K)}$$

Seja f(x), f’’(x), f’’’’(x) continuas em um intervalo I que contem a raiz x=ξ de f(x)=0.
Supor que f’(ξ)≠0
Então existe um intervalo $\dot{I} \subset I$ e contém a raiz ξ tal que x pertence a $\dot{I}$ a sequencia gerada por {xk} gerada pela formula recursiva convergira a raiz.

Escolhemos então o método de Newton Raphson para trabalhar com a terceira função sendo ele um dos métodos de convergência mais rápida, tivemos então que calcular sua derivada e ir fazendo as interações até que os critérios de parada fossem satisfeitos.

Gráfico da Função Y Vs t

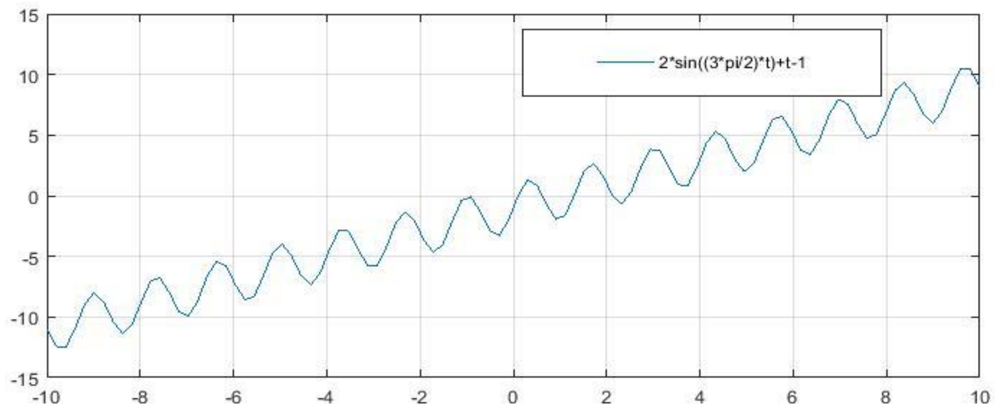

Figura 1.4

Código

```
                    % Método de Newton


t=1.4;
f1=2*sin((3*pi/2)*t)+t-1;
f2=3*cos((3/2)*pi*t)+1;% df(t)/dt
f3 = abs(-1*(9/2)*sin((3/2)*pi*t));% d^2f(t)/dt^2
res = abs((f1*f3)/f2.^2);%|ft*(d^2/dt^2)ft/(d/dtft)^2|
if res < 1
 disp('O ponto inicial satisfaz a condição
|ft*(d^2/dt^2)ft/(d/dtft)^2|<1');
end

%O comando eval acaba possibilitando essas execuções no nosso código
%O trabalho que teriamos em ficar recopiando a função ele acaba
puchando
%ela como um novo parâmetro
f = '2*sin((3*pi/2)*t)+t-1'; % f(t)
flin = '3*pi*cos((3*pi/2)*t)+1'; % df(t)/dt

e1 = 10^-3; % Diferença entre aproximações e Erro máximo para |f(t)|


k = 0; % Contador de iterações
%vamos colocar nosso primeiro ponto na função e assim encontramos a
nossa
ft = eval(f);

fprintf(1,'%s %2d %s %12.9f %s %12.9f\n','k =',k,' t =',t,...
' f(t) =',ft);
                %vamos usar apenas o |f(t)| como critério de parada
como já
                %colocamos ele em um bom ponto não precisou usar a
%diferença dos ta e t
```

```
                    while abs(ft) > e1
                                k = k+1;%contador de interações
                                ta = t; % Aproximação anterior
                                flt = eval(flin); % df(t)/dt no ponto
atual
                                t = t-(ft/flt); % Novo valor de t
                                ft = eval(f); % Novo valor de f(t)
                                dt = abs(t-ta);%módulo da diferença
|t-ta|
fprintf(1,'%s %2d %s %12.9f %s %12.9f %s %12.9f\n',...
    'k =',k,' t =',t,' f(t) =',ft,' d(t) =',dt);
end
```

Análise do código/ dados

Podemos reparar que esse código é mais elaborado que os outros por sua vez, precisamos agora não só atualizar os novos valores de t na função, mas também na sua derivada, escolhendo o ponto inicial e fazendo a verificação que determina se o ponto inicial, converge, chegamos rapidamente na solução que satisfaz os critérios de parada.

```
Command Window
>> NewtonRa
O ponto inicial satisfaz a condição |ft*(d^2/dt^2)ft/(d/dtft)^2|<1
k =  0  t =  1.400000000  f(t) =  1.018033989
k =  1  t =  1.297823622  f(t) = -0.035287848  d(t) =  0.102176378
k =  2  t =  1.301251912  f(t) =  0.000042084  d(t) =  0.003428290
>>
```

Figura 1.5

Analise de dados/ código IV

Resolução da Segunda Parte sistema não linear.

$$t1^3 - 9t1 = 4 + 6t2$$
$$\log(t2)\,(t2 - t1) = 3(t3 - t2) - 1$$
$$2\sin\left(\left(\frac{3pi}{2}\right)(t3 - t2)\right) = t1 - t3$$

Metodologia

Existem várias maneiras para resolver esse sistema, dentre elas vou selecionar duas uma conhecida como Newton inexato e outra como Newton modificado, a maior diferença entre eles é que no Newton modificado o valor do jacobiano se mantem constante, sempre usando o valor do chute inicial X0 para as próximas interações. Para a resolução desse dele vamos trabalhar com o método de Newton para sistemas não lineares, mesmo sendo computacionalmente mais cara.

Código

```
                    %Método de newton inexato

           e= 10^-3; % Erro Máximo para |f(t)|
%vamos escolher esses pontos iniciais porque eles foram os obtidos dos
%métodos passados, assim podemos chegar mais próximo da função.


 t= [0.337402344;2.857244673;1.301251912];
%poderíamos mandar os t1 dessa forma, mas podemos em relação
%ao nosso x t1=xk(1,1),t2=xk(2,1),t3=xk(3,1)

           ft=[t(1).^3-9*t(1)-6*t(2)-4 ;log10(t(2)).*(t(2)-t(1))-
3*(t(3)-t(2))+1;
                2*sin(((3*pi)/2)*(t(3)-t(2)))-t(1)+t(3)];
%assim teremos um controle melhor das nossas variáveis
Jk=[       3*t(1)^2 - 9,                                       -6,
0;
 -log(t(2))/log(10), log(t(2))/log(10) - (t(1) - t(2))/(t(2)*log(10))
+ 3,                             -3;
              -1,              -3*pi*cos((3*pi*(t(3) - t(2)))/2),
3*pi*cos((3*pi*(t(3) - t(2)))/2) + 1];

k=0;
dt=9999;


                         while max(abs(ft)) > e && max(abs(dt))> e

                    %vamos implementar Lu com pivotemento parcial
                    [L,U,p]=lu(Jk);
                    y=L\(p*(-ft));
                    s=U\y;
                    ta=t;
                    t = t + s;
                    ft=[t(1).^3-9*t(1)-6*t(2)-4 ;log10(t(2)).*(t(2)-
t(1))-3*(t(3)-t(2))+1;
                2*sin(((3*pi)/2)*(t(3)-t(2)))-t(1)+t(3)];
            Jk=[       3*t(1)^2 - 9,
-6,                              0;
 -log(t(2))/log(10), log(t(2))/log(10) - (t(1) - t(2))/(t(2)*log(10))
+ 3,                             -3;
              -1,              -3*pi*cos((3*pi*(t(3) - t(2)))/2),
3*pi*cos((3*pi*(t(3) - t(2)))/2) + 1];
                    dt=t-ta;
                    k=k+1;


                    fprintf(1,'%s %2d %s %12.9f %s %12.9f\n ',...
    'k =',k,' d(t) =',max(abs(dt)),' f(t) =',max(abs(ft)));
                         end
```

Análise do código/dados

O valor inicial escolhido para começar as interações é formado pelos valores que encontramos implementando os métodos na primeira e na terceira função, o valor que foi colocado como t(2) foi obtido através de um erro, no matlab2018 a função log(x) representa lnx, se você quiser usar log na base 10 vai precisar colocar log10(x), quando a primeira função foi resolvida ela estava indo para ln logo obtivemos esse valor que aproximou ainda mais os valores, fazendo ele ter uma convergência mais rápida.

A implementação do vetor t para a atualização da nossa função foi necessária o que acabou dificultando na utilização da função eval, podemos reparar que no nosso while tivemos que repetir novamente o F(t) e seu Jacobiano, acarretando um custo computacional maior ainda, mas conseguimos resolver o problema com poucas interações, mesmo sabendo que o número de interações não é determinante para dizer a velocidade do seu código.

```
Command Window
>> NewtonInexato
k =  1  d(t) =  4.471272794  f(t) =  3.832830496
 k =  2  d(t) =  2.212710848  f(t) =  3.952632775
 k =  3  d(t) =  0.808736991  f(t) =  0.418731062
 k =  4  d(t) =  0.093443723  f(t) =  0.020409692
 k =  5  d(t) =  0.002349829  f(t) =  0.000048362
fx >>
```

Figura 1.6

Conclusão

Pode ser verificado que todos os nossos problemas foram resolvidos satisfazendo seus critérios de convergência e de parada, tentamos a todo momento buscar caminhos para otimizar nossos passos e aproximar ainda mais os resultados obtidos, buscando sempre o menor número de interações possíveis. Para a implementação de modelos mais complexos foram feitos testes com funções mais básicas para comprovar a exatidão não só de um resultado, mas da lógica do método empregado.

BIBLIOGRAFIA

Notas de aula do Professor Giusepp Romanazzi.

Cálculo Numérico aspectos teóricos e computacionais, Márcia A.gomes Ruggiero, Vera lúcia da Rocha Lopes.