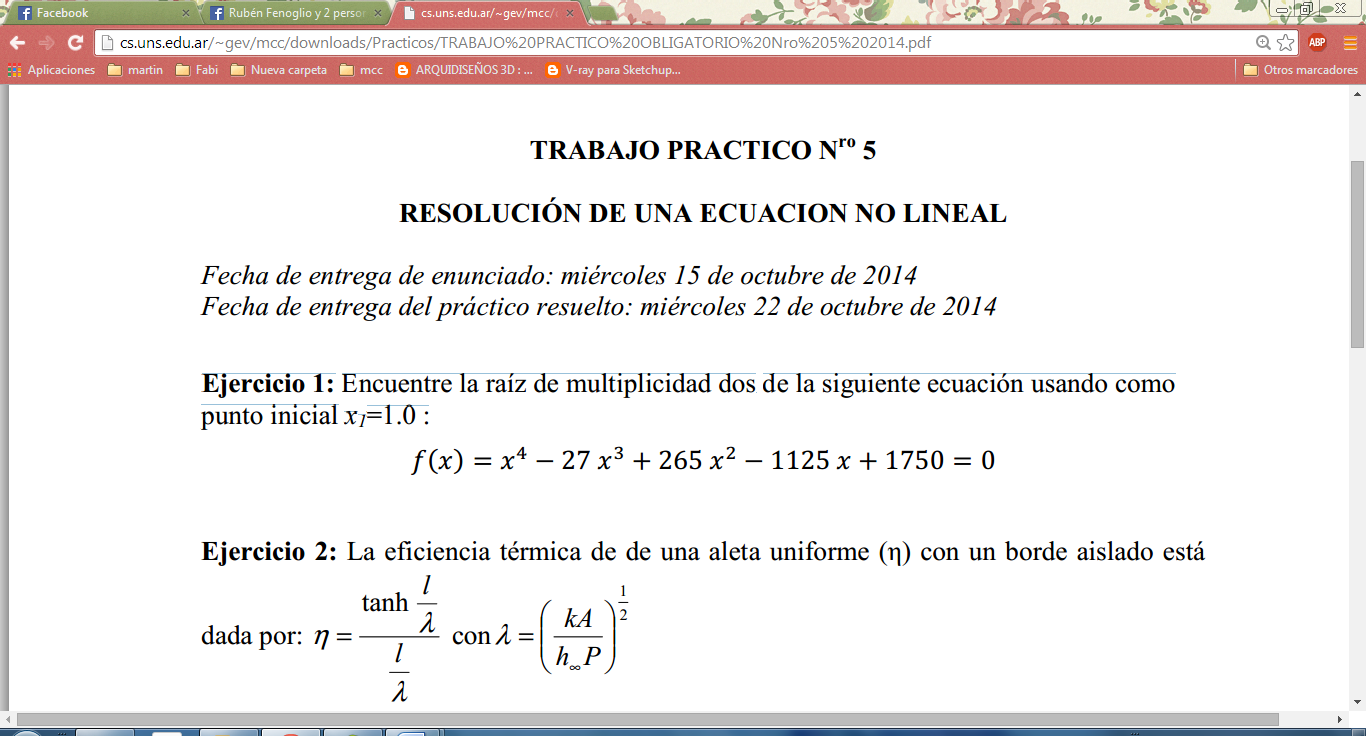
|  |
| --- |
| Métodos de Computación Científica - 2014 - |
| **Trabajo Práctico N° 5: “Resolución de una Ecuación no Lineal”**  **GARAT, Fabiana Yamel - LU 89108 -** |
|  |
|  |



**Ejercicio 1 -** Encuentre la raíz de multiplicidad dos de la siguiente ecuación usando como punto inicial



**Solución –** Lo resolví utilizando el método de Newton-Raphson, modificándolo para raíces múltiples de la siguiente manera

Donde *p* denota la multiplicidad de la raíz. Como quiero encontrar una raíz doble, reemplazo *p* por 2. es la función planteada en el enunciado, es la derivada primera de la misma, y sabemos que en su primera iteración () es 1.0.

Usando el siguiente Script en MatLab, y tomando una tolerancia de 0.001, encontramos la raíz buscada

xa**=**1.0**;**

xb **=** 0**;**

error**=**1**;**

tolerancia **=** 0.001**;**

**while** error**>**tolerancia

xb**=** xa **-** 2 **\*** **(** **(**xa**^**4 **-** 27**\***xa**^**3 **+** 265**\***xa**^**2 **-** 1125**\***xa **+** 1750**)**

**/** **(** 4**\***xa**^**3 **-** 81**\***xa**^**2 **+** 530**\***xa **-** 1125**)** **);**

error **=** abs**((**xb**-**xa**)/**xb**);**

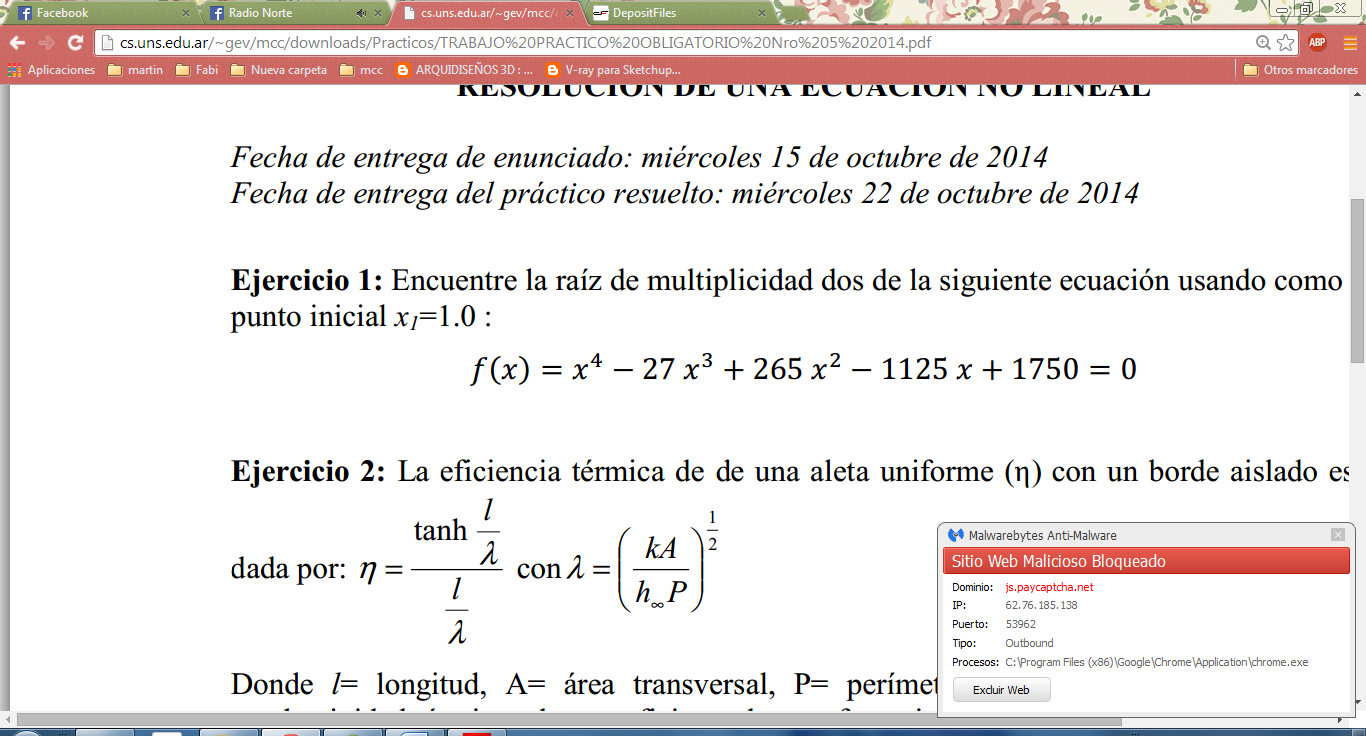
xa**=**xb**;**

**end**

disp**(**xb**);**

El Script da como resultado 5.0000 la cual es una raíz doble de la función.

**Ejercicio 2 –** La eficiencia térmica de una aleta uniforme (η) con un borde aislado está dada por:



Donde

* l= longitud,
* A= área transversal,
* P= perímetro de la sección transversal,
* k= conductividad térmica y
* = coeficiente de transferencia de calor de la aleta.

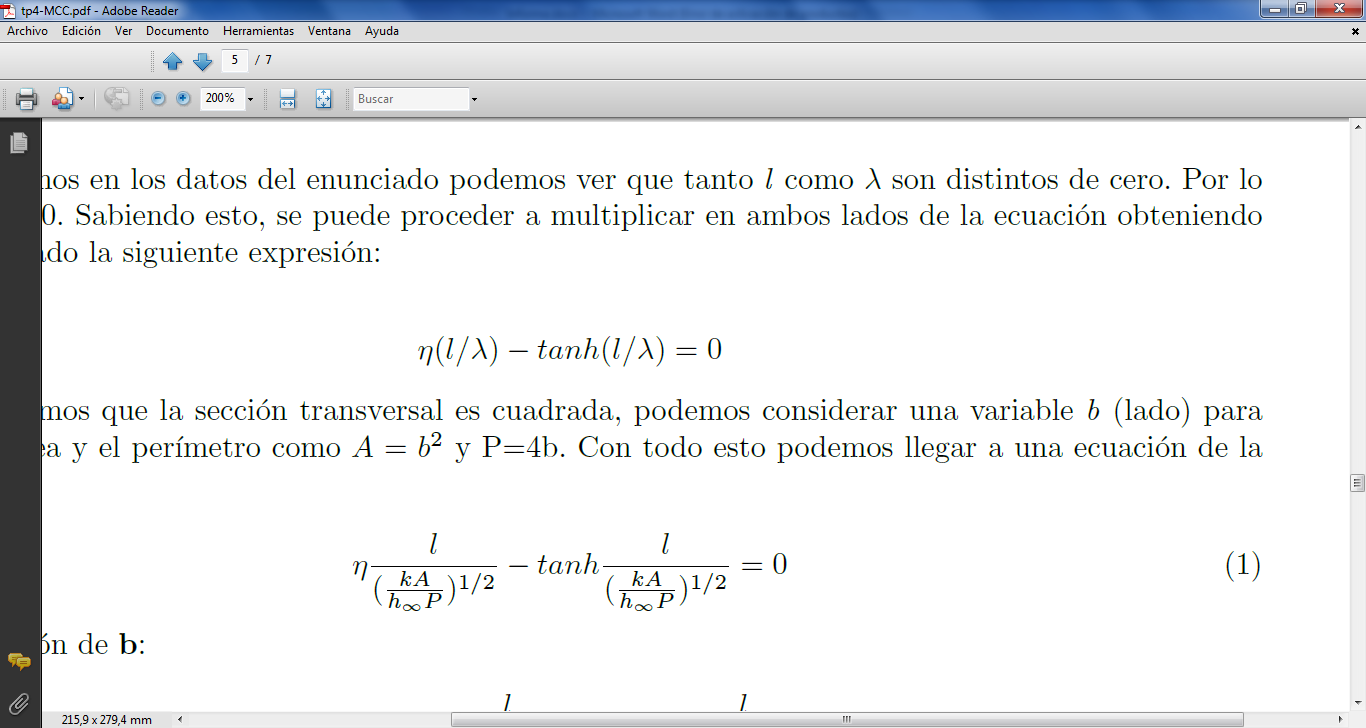
Si la aleta está hecha de aluminio con una sección transversal cuadrada con l=0.1m, k=240 W/(m ºC) y =9 W/(m2 ºC), determinar las dimensiones transversales necesarias de la aleta para lograr una eficiencia de 0.95 usando los siguientes métodos:

a) Método gráfico

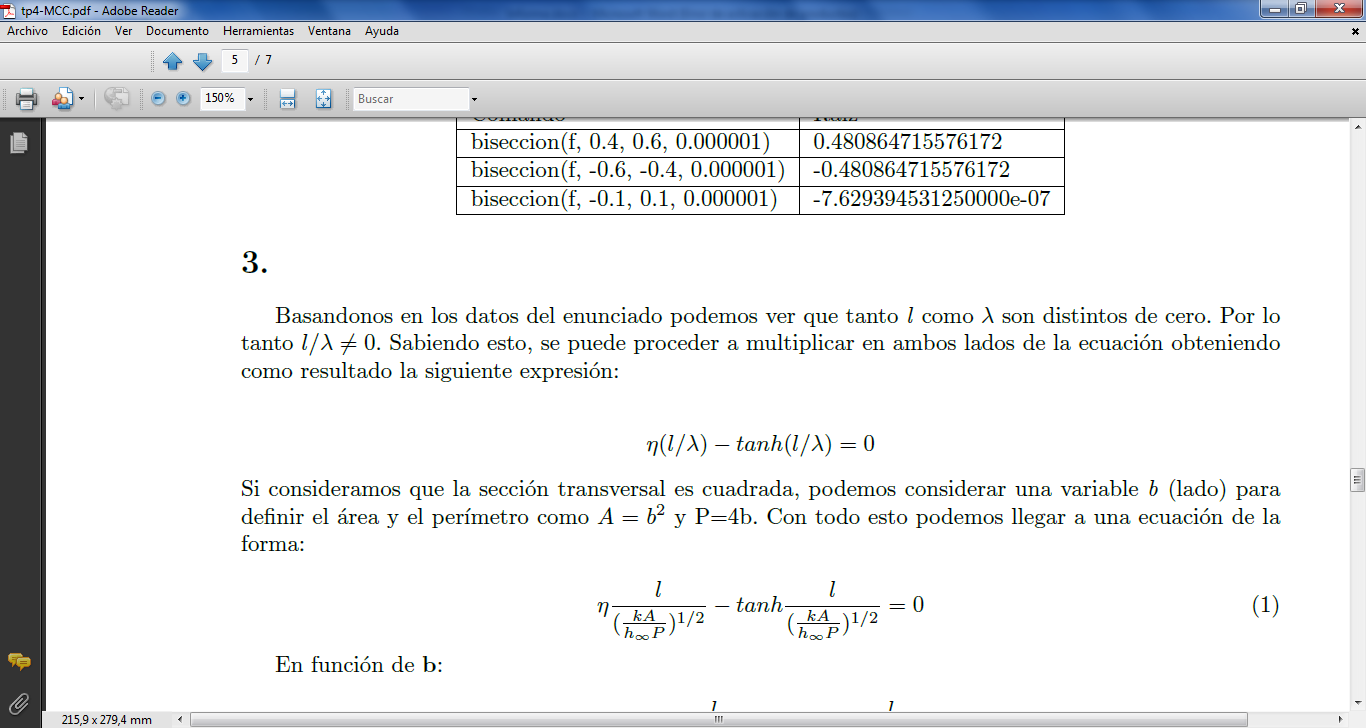
b) Método de bisección

c) Método de Newton-Raphson

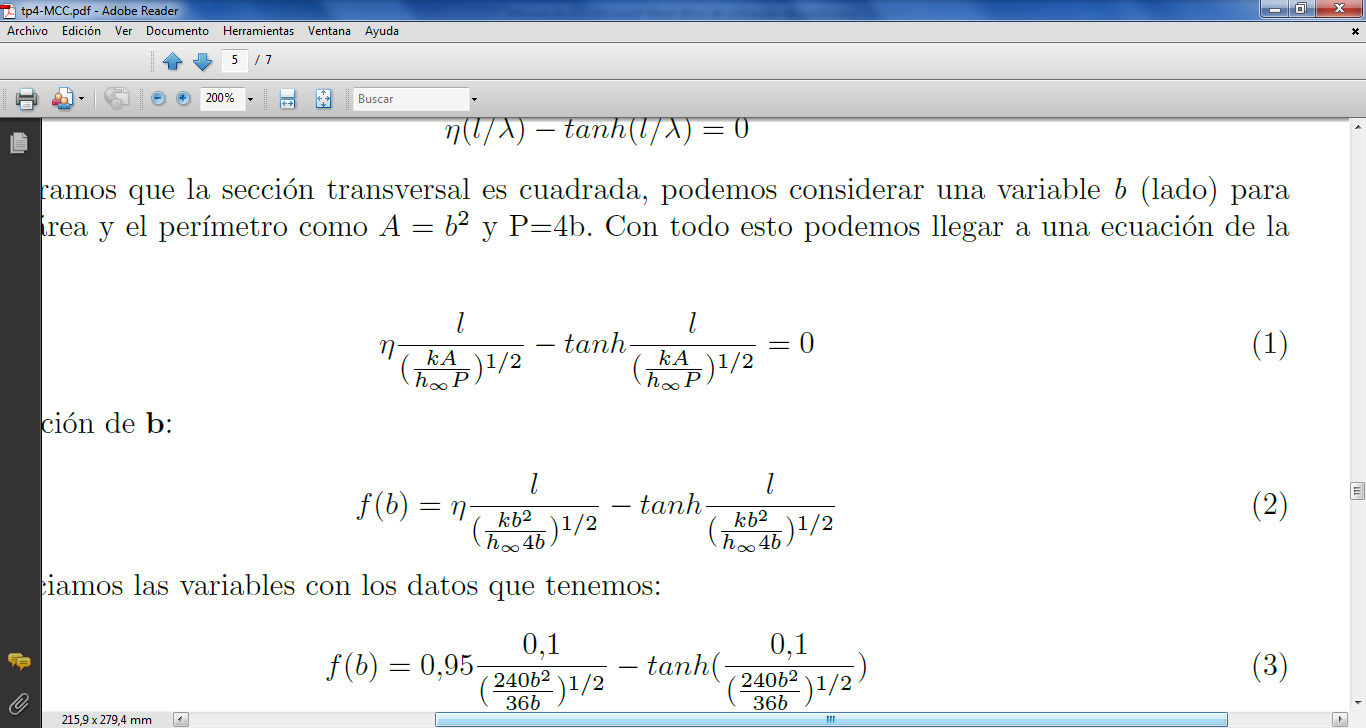
**Solución –** Basándonos en los datos del enunciado podemos ver que tanto como son distintos de cero. Por lo tanto . Sabiendo esto, se puede proceder a multiplicar en ambos lados de la ecuación obteniendo como resultado la siguiente expresión:



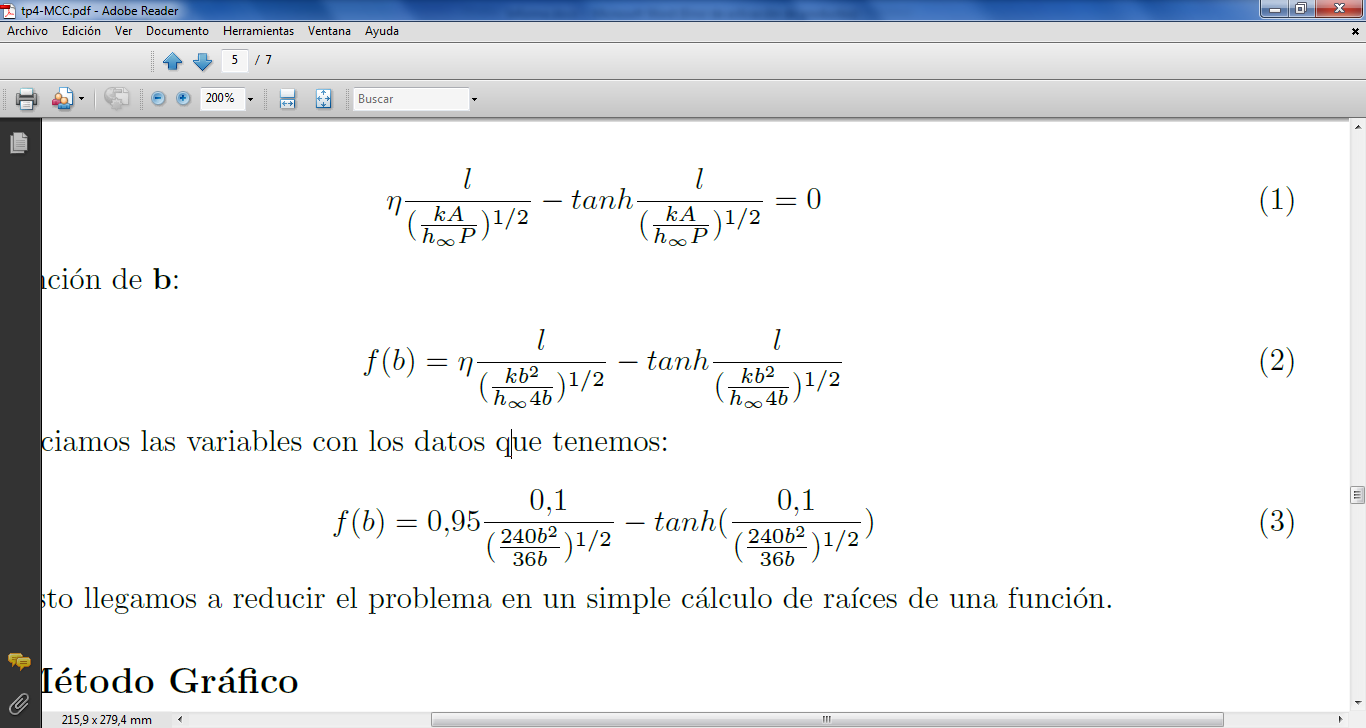
Si consideramos que la sección transversal es cuadrada, podemos considerar una variable (lado) para definir el área y el perímetro como y . Con todo esto podemos llegar a una ecuación de la forma:



En función de



Instanciamos las variables y obtenemos



Reduciendo este problema a un simple cálculo de raíces de una función.

1. Método Grafico

>> lambda = @(d)0.1/(sqrt((240\*d^2)/(36\*d)))

lambda =

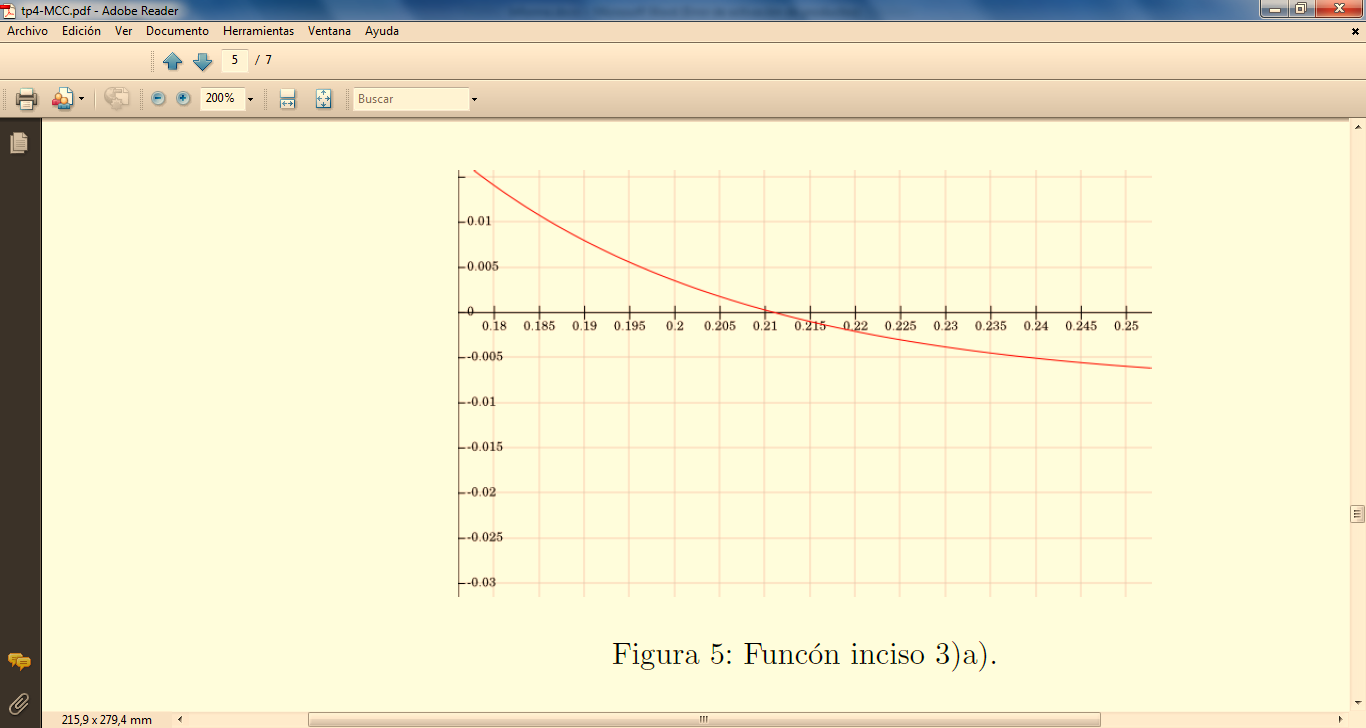
@(d)0.1/(sqrt((240\*d^2)/(36\*d)))

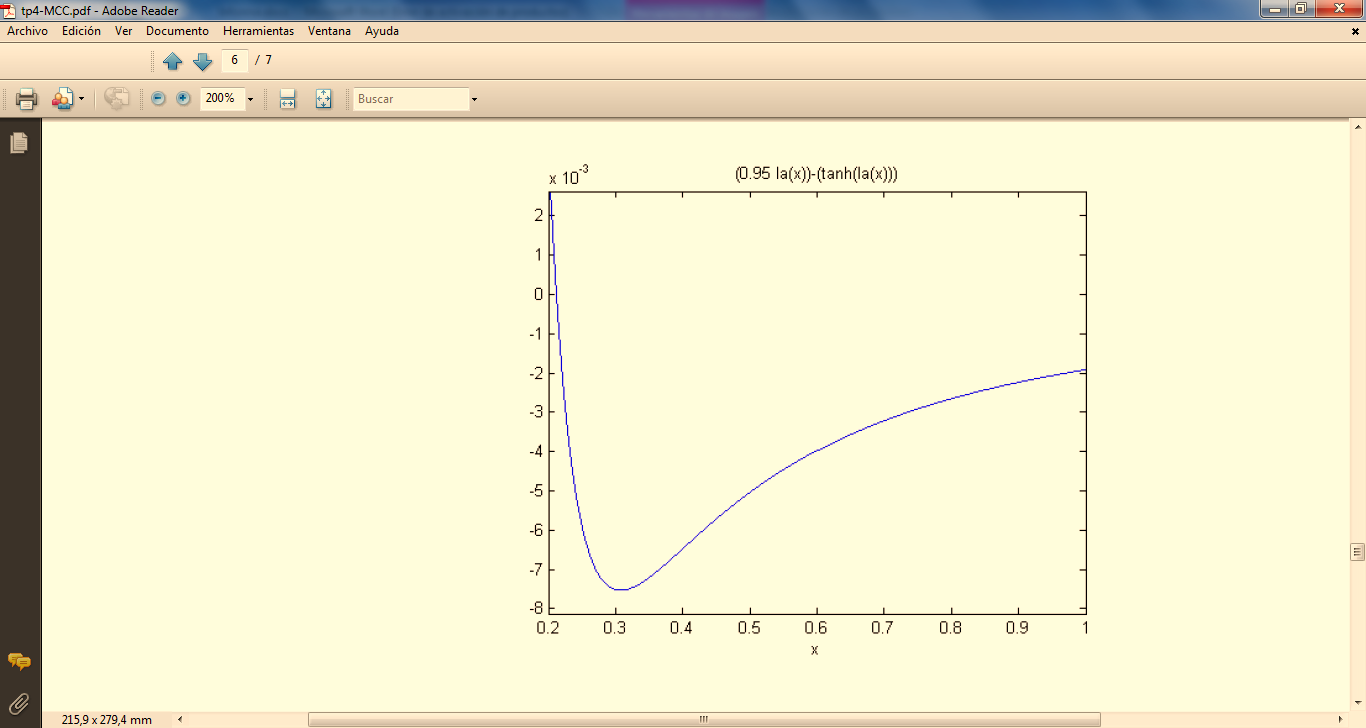
>> f = @(d) 0.95\*lambda(d) - tanh(lambda(d))

f =

@(d)0.95\*lambda(d)-tanh(lambda(d))

>> ezplot(f)





Como se puede ver en la gráfica la función tiene una raíz en las cercanías de x=0.21. Además, a través del gráfico podemos observar que la función está muy mal condicionada.

1. Método de bisección

Implementamos una función auxiliar que permite hallar el cero de la función por medio del método de la bisección:

function [ raiz ] = biseccion(f, a, b, maxIt )

% f: function handler

% [a,b]: extremos del intervalo en donde se busca la raiz.

if ( f(a) == 0 )

raiz = a;

return;

elseif ( f(b) == 0 )

raiz = b;

return;

elseif ( f(a) \* f(b) > 0 )

error( 'f(a) and f(b) no tienen distinto signo' );

end

iteraciones = 0;

tol\_f = eps;

tol\_x = eps;

while (((b - a) >= tol\_x )|| (( abs(f(a)) >= tol\_f && abs(f(b)) >= tol\_f))&&(iteraciones < maxIt))

c = (a + b)/2; %punto medio.

if ( f(c) == 0 ) %se encontró la raíz.

break;

elseif ( f(a)\*f(c) < 0 ) %hay cambio de signo.

b = c;

else

a = c;

end

end

raiz = c;

end

Se invoca dicha función auxiliar con la función “f” a la cual le queremos calcular los ceros (la del problema), un intervalo en el cual puede encontrarse la raíz [0.001,0.3] con un máximo de 100 iteraciones como condición de corte, y obtenemos los siguientes resultados:

>> x = biseccion(f,0.001,0.3,100)

x = 0.009400452696480

1. Método de Newton-Raphson

Se implementa una función auxiliar que permite hallar el cero de la función por medio del método de Newton-Raphson:

function [ raiz ] = NewtonRaphson( f, x0, maxIt )

tol = eps;

i=1;

fx(i) = x0;

syms x;

f1 = subs(f,x,fx(i));

z=diff(f);

d=subs(z,x,fx(i));

ea(1)=100;

while ((abs(ea(i))>=tol)&& (i < maxIt));

fx(i+1)=fx(i)-f1/d;

f1=subs(f,x,fx(i+1));

d=subs(z,x,fx(i+1));

ea(i+1)=abs((fx(i+1)-fx(i))/fx(i+1)\*100);

i=i+1;

end

raiz = fx(i-1);

fprintf('i fx(i) Error aprox (i) \n');

for j=1:i;

fprintf('%2d \t %11.7f \t %7.3f \n',j-1,fx(j),ea(j));

end

end

Al estar mal condicionado el problema, es decir, en un entorno muy grande de la raíz la función es casi cero, entonces es necesario elegir un valor inicial muy cercano a la raíz. Del ejemplo anterior sabemos que la raíz está cerca de 0.009. Veamos que ocurre para un valor no muy lejano a la raíz, supongamos 0.02:

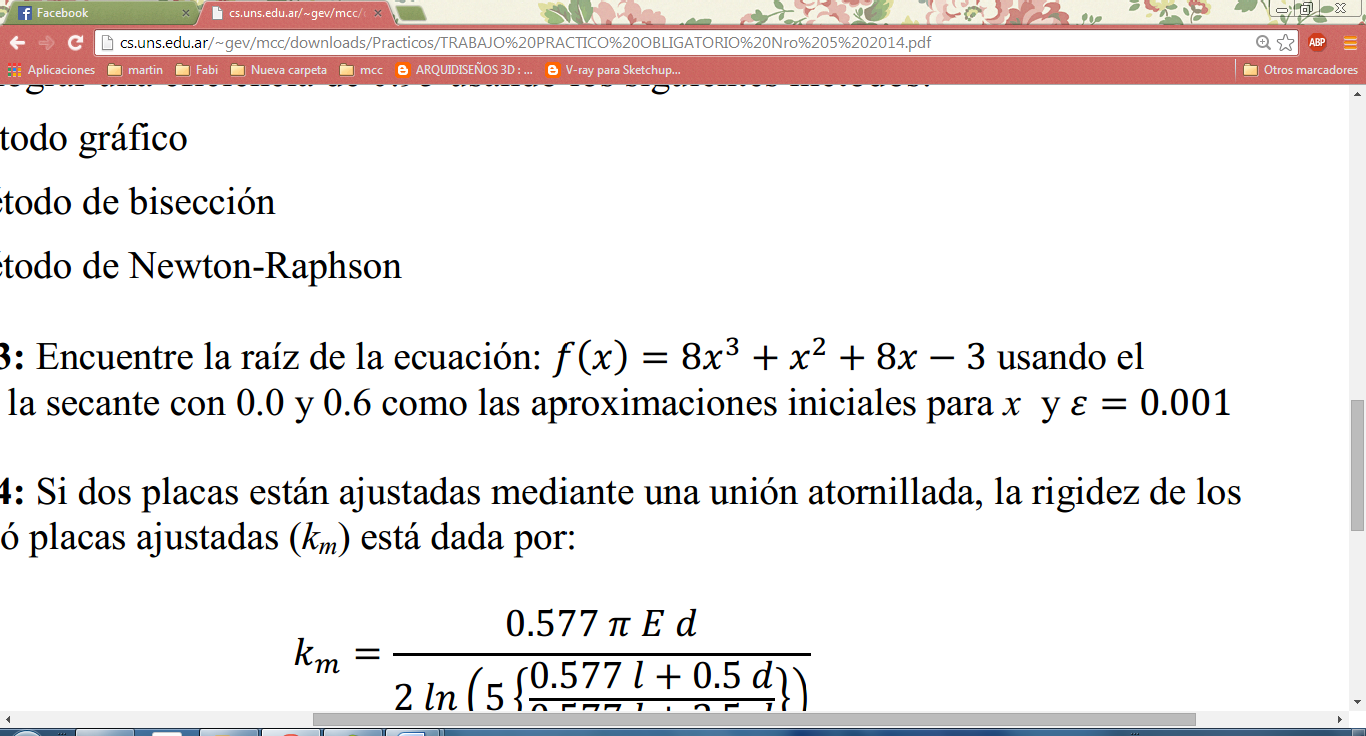
|  |
| --- |
| >> f = '0.95\*(0.1/(sqrt((240\*x^2)/(36\*x))))-tanh(0.1/(sqrt((240\*x^2)/(36\*x))))'  f = 0.95\*(0.1/(sqrt((240\*x^2)/(36\*x))))-tanh(0.1/(sqrt((240\*x^2)/(36\*x))))  >> x = NewtonRaphson(f,0.02,100)  i fx(i) Error aprox (i)  0 0.0200000 100.000  1 -0.0280820 171.220  2 -0.0644282 56.414  3 -0.1656153 61.098  4 -0.4627654 64.212  5 -1.3506389 65.737  6 -4.0127518 66.341  7 -11.9985404 66.556  8 -35.9557171 66.630  9 -107.8271832 66.654  10 -323.4415604 66.663  11 -970.2846846 66.665  12 -2910.8140551 66.666  13 -8732.4021656 66.667  14 -26197.1664969 66.667  15 -78591.4594907 66.667  16 -235774.3384720 66.667  17 -707322.9754161 66.667  18 -2121968.8862482 66.667  19 -6365906.6187445 66.667  20 -19097719.8162334 66.667  21 -57293159.4087003 66.667  22 -171879478.1861013 66.667  23 -515638434.5183047 66.667  24 -1546915303.5149117 66.667  25 -4640745910.5047455 66.667  26 -13922237731.4743000 66.667  27 -41766713194.3829570 66.667  28 -125300139583.1088300 66.667  29 -375900418749.2878400 66.667  30 -1127701256247.8228000 66.667  31 -3383103768743.4165000 66.667  32 -10149311306230.1820000 66.667  33 -30447933918690.5860000 66.667  34 -91343801756072.1720000 66.667  35 -274031405268217.3100000 66.667  36 -822094215804647.7500000 66.667  37 -2466282647413937.0000000 66.667  38 -7398847942241811.0000000 66.667  39 -22196543826725432.0000000 66.667  40 -66589631480176080.0000000 66.667  41 -199768894440527650.0000000 66.667  42 -599306683321579780.0000000 66.667  43 -1797920049964739800.0000000 66.667  44 -5393760149894198300.0000000 66.667  45 -16181280449682604000.0000000 66.667  46 -48543841349047697000.0000000 66.667  47 -145631524047142240000.0000000 66.667  48 -436894572141426110000.0000000 66.667  49 -1310683716424278200000.0000000 66.667  50 -3932051149272818600000.0000000 66.667  51 -11796153447818432000000.0000000 66.667  52 -35388460343455258000000.0000000 66.667  53 -106165381030365250000000.0000000 66.667  54 -318496143091095060000000.0000000 66.667  55 -955488429273284100000000.0000000 66.667  56 -2866465287819839400000000.0000000 66.667  57 -8599395863459507000000000.0000000 66.667  58 -25798187590378491000000000.0000000 66.667  59 -77394562771135026000000000.0000000 66.667  60 -232183688313404670000000000.0000000 66.667  61 -696551064940213040000000000.0000000 66.667  62 -2089653194820632600000000000.0000000 66.667  63 -6268959584461852900000000000.0000000 66.667  64 -18806878753385459000000000000.0000000 66.667  65 -56420636260156314000000000000.0000000 66.667  66 -169261908780468380000000000000.0000000 66.667  67 -507785726341402920000000000000.0000000 66.667  68 -1523357179024205900000000000000.0000000 66.667  69 -4570071537072611100000000000000.0000000 66.667  70 -13710214611217777000000000000000.0000000 66.667  71 -41130643833653236000000000000000.0000000 66.667  72 -123391931500959560000000000000000.0000000 66.667  73 -370175794502877850000000000000000.0000000 66.667  74 -1110527383508630700000000000000000.0000000 66.667  75 -3331582150525884200000000000000000.0000000 66.667  76 -9994746451577623800000000000000000.0000000 66.667  77 -29984239354732831000000000000000000.0000000 66.667  78 -89952718064198415000000000000000000.0000000 66.667  79 -269858154192594400000000000000000000.0000000 66.667  80 -809574462577779680000000000000000000.0000000 66.667  81 -2428723387733334900000000000000000000.0000000 66.667  82 -7286170163199972300000000000000000000.0000000 66.667  83 -21858510489599930000000000000000000000.0000000 66.667  84 -65575531468799756000000000000000000000.0000000 66.667  85 -196726594406398510000000000000000000000.0000000 66.667  86 -590179783219194780000000000000000000000.0000000 66.667  87 -1770539349657577900000000000000000000000.0000000 66.667  88 -5311618048972714800000000000000000000000.0000000 66.667  89 -15934854146918115000000000000000000000000.0000000 66.667  90 -47804562440754356000000000000000000000000.0000000 66.667  91 -143413687322262380000000000000000000000000.0000000 66.667  92 -430241061966786990000000000000000000000000.0000000 66.667  93 -1290723185900359900000000000000000000000000.0000000 66.667  94 -3872169557701059500000000000000000000000000.0000000 66.667  95 -11616508673103127000000000000000000000000000.0000000 66.667  96 -34849526019309316000000000000000000000000000.0000000 66.667  97 -104548578057927710000000000000000000000000000.0000000 66.667  98 -313645734173783260000000000000000000000000000.0000000 66.667  99 -940937202521348350000000000000000000000000000.0000000 66.667  x =  -6.1624e-087 +3.3546e-071i |
| Podemos observar que luego de 100 iteraciones se logra aproximar el valor de la raíz con un error del 66.667% lo cual es un error muy grande. |

Veamos que sucede para un valor inicial más cercano a la raíz, digamos 0.002:

|  |
| --- |
| >> x = NewtonRaphson(f,0.002,100)  i fx(i) Error aprox (i)  0 0.0020000 100.000  1 0.0032978 39.353  2 0.0050453 34.636  3 0.0070218 28.149  4 0.0086274 18.610  5 0.0093124 7.356  6 0.0093993 0.925  7 0.0094005 0.013  8 0.0094005 0.000  9 0.0094005 0.000  x =  0.0094005 |
| Se puede ver que el método halla el valor correcto de la raíz en 10 iteraciones. |

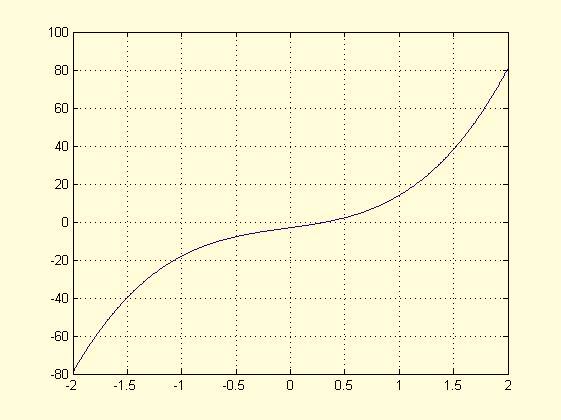
Ejemplos que demuestran que el problema está mal condicionado.

**Ejercicio 3 -**  Encuentre la raíz de la ecuaciónón



Usando el método de la secante con 0.0 y 0.6 como las aproximaciones iniciales para x y

**Solución -**  Partiendo de la gráfica de



A partir de esto y utilizando el método de la secante, tomando a 0.0 y 0.6 como las aproximaciones iniciales para x y un , puedo encontrar una aproximación suficientemente buena para la raíz.

En Matlab lo logre a través de:

f **=** inline**(**'8\*x^3 + x^2 + 8\*x -3'**);**

x0**=**0.0**;**

x1**=**0.6**;**

xra**=**0**;**

xr**=**0**;**

tol**=**0.001**;**

i**=**1**;**

error\_aprox**=**1**;**

error**=**0**;**

fx1**=**f**(**x1**);**

fx0**=**f**(**x0**);**

xr**=**x1**-((**x1**-**x0**)/(**fx1**-**fx0**))\***fx1**;**

fprintf**(**'It. X0 X1 Xr Error aprox \n'**);**

fprintf**(**'%2d \t %11.7f \t %11.7f \t %11.7f \t %11.7f \n'**,**i**,**x0**,**x1**,**xr**,**error**);**

**while** error\_aprox **>=** tol**,**

xra**=**xr**;**

x1**=**xr**;**

fx1**=**f**(**x1**);**

fx0**=**f**(**x0**);**

xr**=**x1**-((**x1**-**x0**)/(**fx1**-**fx0**))\***fx1**;**

error **=** abs**((**xr **-** xra**)** **/** xr**);**

error\_aprox **=** error**;**

i**=**i**+**1**;**

fprintf**(**'%2d \t %11.7f \t %11.7f \t %11.7f \t %11.7f \n'**,**i**,**x0**,**x1**,**xr**,**error\_aprox**);**

**end**

Lo que da como resultado

It. X0 X1 Xr Error aprox

1 0.0000000 0.6000000 0.2613240 0.0000000

2 0.0000000 0.2613240 0.3406131 0.2327834

3 0.0000000 0.3406131 0.3236682 0.0523529

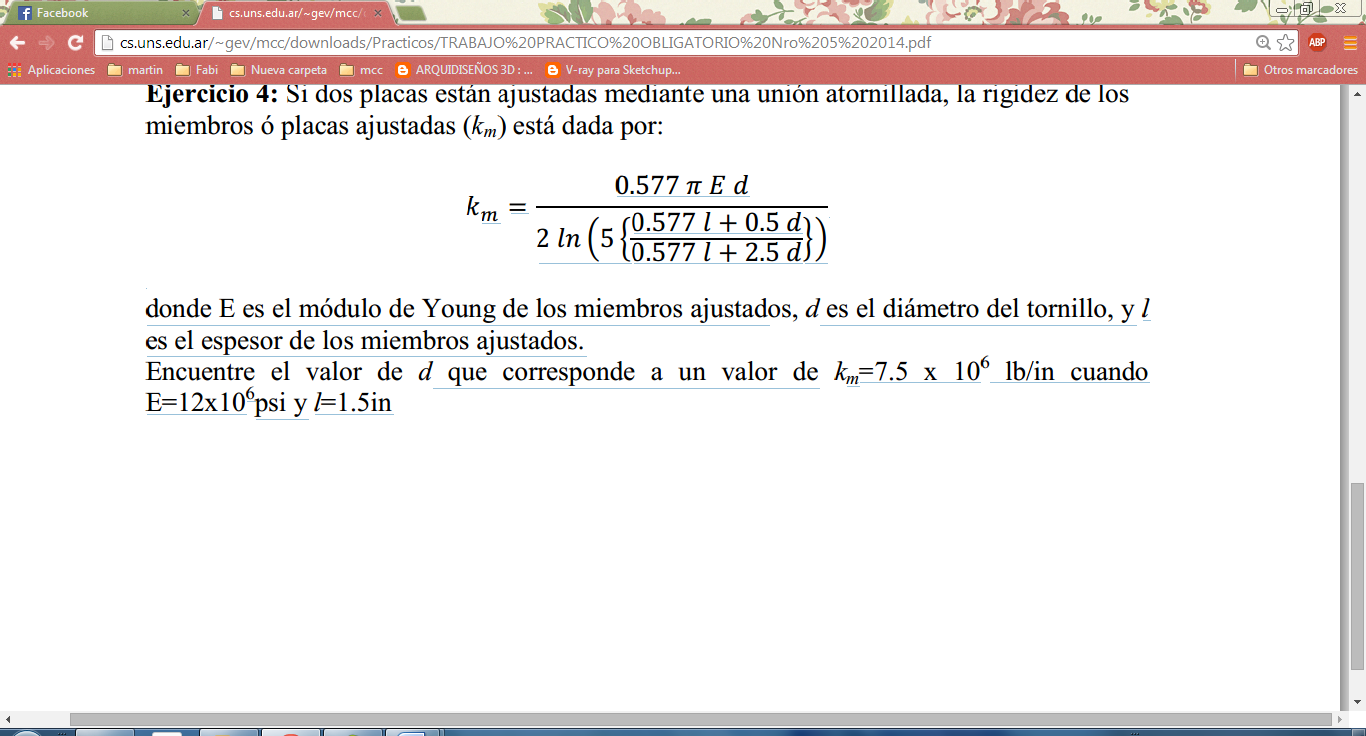
4 0.0000000 0.3236682 0.3274481 0.0115436

5 0.0000000 0.3274481 0.3266114 0.0025617

6 0.0000000 0.3266114 0.3267970 0.0005677

Luego, lo cual es una buena aproximación a la raíz.

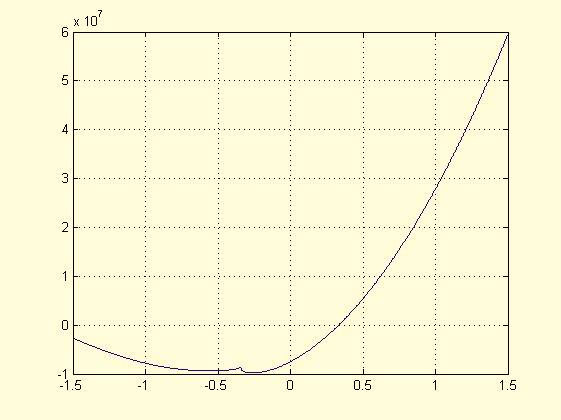
**Ejercicio 4 -** Si dos placas están ajustadas mediante una unión atornillada, la rigidez de los miembros ó placas ajustadas () está dada por:



Donde E es el módulo de Young de los miembros ajustados, d es el diámetro del tornillo, y l es el espesor de los miembros ajustados.

Encuentre el valor de d que corresponde a un valor de lb/in cuando psi y l=1.5in

**Solución –** Decidí proceder de manera similar al ejercicio anterior. Procedí a obtener el grafico de la función para ver en que intervalos la misma se hace cero.



Mirando el gráfico podemos ver que hay una raíz entre 0 y 0.5.

Luego reemplazamos las variables , *E*, y *l* por los valores dados en el enunciado obtenemos una ecuación con una única incógnita *d*. Luego igualando dicha ecuación a cero, el problema se reduce a encontrar los ceros de la ecuación.

Decido luego utilizar el método de la secante descripto con anterioridad para obtener la solución. Tomando a 0.0 y 0.5 como las aproximaciones iniciales para d, y una tolerancia de , dando como resultado:

It. X0 X1 Xr Error aprox

1 0.0000000 0.5000000 0.2903313 0.0000000

2 0.0000000 0.2903313 0.3460432 0.1609971

3 0.0000000 0.3460432 0.3290582 0.0516170

4 0.0000000 0.3290582 0.3340389 0.0149105

5 0.0000000 0.3340389 0.3325613 0.0044433

6 0.0000000 0.3325613 0.3329981 0.0013120

7 0.0000000 0.3329981 0.3328688 0.0003884

Se encuentra , lo cual concuerda con lo obtenido gráficamente.