

Cálculo Numérico - solução de equações

Suponha que precisamos usar o computador para resolver equações. Já paramos para entender o que é resolver uma equação? Conhece-se a forma geral de uma equação $y = f(x)$. Resolver esta equação vem a ser descobrir quais os valores de x que fazem $f(x)$ ser igual a zero. Se olharmos para o gráfico, veremos que busca-se o valor da abscissa x que determina o cruzamento da curva de $f(x)$ no eixo dos x .

Por exemplo, suponha-se precisar descobrir a raiz 11 do número 1739? Ou seja, qual o número que multiplicado por ele mesmo 11 vezes dá como resultado 1739?

Dizendo isso na forma matemática, pode-se escrever $x^{11} = 1739$, ou melhor ainda $x^{11} - 1739 = 0$. Com isso, chegamos ao ponto.

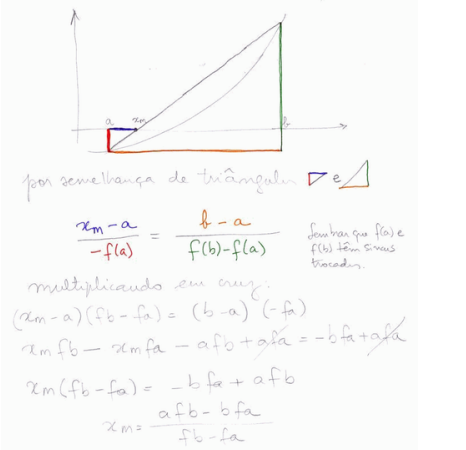
Método da Bissecção Se olharmos o gráfico de uma função, ver-se-á que no entorno da raiz, de um lado a função é positiva, e do outro lado é negativa ou vice versa. Este método começa pela seleção de 2 pontos x_a e x_b sendo que a função $y = f(x)$ têm sinais opostos nesses dois pontos. Graças a essa constatação, pode-se afirmar que a raiz procurada está entre x_a e x_b . Dizendo de outra maneira. O valor de x procurado, está entre a e b . O Método da bissecção vai dividindo o intervalo a, b pela metade, questionando a seguir em qual das metades o x está. Descoberto isso, despreza-se a metade que não contém x . Levando este processo ao limite, descobre-se qual o valor de x . Em resumo, eis as etapas do método:

1. Arbitrar intervalo em $[a, b]$ que compreenda a raiz.
2. Arbitrar um erro permitido ϵ
3. Calcular $x_m = (a + b)/2$
4. Se $f(a).f(x_m) < 0$ o novo intervalo será $[a, x_m]$
5. Se $f(a).f(x_m) > 0$ o novo intervalo será $[x_m, b]$
6. Enquanto o erro $b - a > \epsilon$ voltar ao passo 3.

Seja um exemplo, calcular $f = e^x - \sin(x) - 2 = 0$ nos limites $a = 1$ e $b = 1.25$ com $\epsilon = 0.01$. No primeira etapa tem-se

```
Limite Superior: 1.25
Limite Inferior: 1
Tolerancia: 0.01
-----a-----b-----xm-----fxm-----fa-----fb-----erro
1.1250 1.0000 1.1250 0.1779 0.5414 -0.1232 0.1250
1.0625 1.0000 1.0625 0.0200 0.1779 -0.1232 0.0625
1.0625 1.0312 1.0312 -0.0534 0.0200 -0.1232 0.0312
1.0625 1.0469 1.0469 -0.0171 0.0200 -0.0534 0.0156
1.0547 1.0469 1.0547 0.0013 0.0200 -0.0171 0.0078
Raiz: 1.05078125
Erro: -0.0078125
Eis o programa Python que resolve este caso
import numpy as np
def bissec():
    a=float(input("Limite Superior: "))
    b=float(input("Limite Inferior: "))
    tol=float(input("Tolerancia: "))
    print("-----a-----b-----xm-----fxm-----fa-----fb-----erro")
    while np.abs(b-a)>tol:
        xm=(a+b)/2
        fxm=np.exp(xm)-np.sin(xm)-2
        fa=np.exp(a)-np.sin(a)-2
        fb=np.exp(b)-np.sin(b)-2
        if fa*fxm>0:
            a=xm
        else:
            b=xm
        print('{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}'.format(a,b,xm,fxm,fa,fb,np.abs(b-a)))
    xm=(a+b)/2
    print("Raiz: ",xm)
    print("Erro: ",b-a)
bissec()
```

Cordas ou Falsa Posição



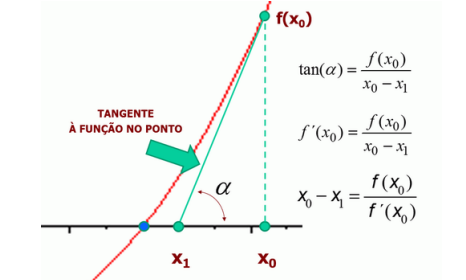
O método começa com uma análise do gráfico da função nas proximidades da raiz. Por semelhança de triângulos, pode-se obter a formulação proposta. Eis os passos do algoritmo:

1. Arbitrar um intervalo $[a, b]$ que compreende a raiz
2. Arbitrar ϵ
3. Calcular $x_m = \frac{a \cdot f(b) - b \cdot f(a)}{f(b) - f(a)}$
4. Se $f(a).f(x_m) < 0$ o novo intervalo será $[a, x_m]$
5. Se $f(a).f(x_m) > 0$ o novo intervalo será $[x_m, b]$
6. Enquanto o erro $f(x_m) > \epsilon$ voltar ao passo 3

Veja a seguir a solução da mesma equação, ($e^x - \sin(x) - 2 = 0$) com os mesmos intervalos do método anterior.

```
Limite inferior: 1
Limite superior: 1.25
Tolerância: 0.001
-----a-----b-----xm-----fxm-----fa-----fb-----erro
1.0463 1.2500 1.0463 -0.0184 -0.1232 0.5414
1.0530 1.2500 1.0530 -0.0026 -0.0184 0.5414
1.0540 1.2500 1.0540 -0.0004 -0.0026 0.5414
Raiz: 1.053972856764054
Erro: -0.0003663936835978099
Eis o programa Python que resolve este caso
from numpy import *
def cordas():
    a=float(input("Limite inferior: "))
    b=float(input("Limite superior: "))
    tol=float(input("Tolerância: "))
    fxm=1
    print("-----a-----b-----xm-----fxm-----fa-----fb-----erro")
    while abs(fxm)>tol:
        fa=exp(a)-sin(a)-2
        fb=exp(b)-sin(b)-2
        fxm=((a*fb)-(b*fa))/(fb-fa)
        fxm=exp(xm)-sin(xm)-2
        if fa*fxm>0:
            a=fxm
        else:
            b=fxm
        print('{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}'.format(a,b,xm,fxm,fa,fb))
    print("Raiz: ",xm)
    print("Erro: ",fxm)
cordas()
```

Método de Newton Este método começa com a interpretação geométrica da derivada. Lembrando, Newton é um monstro da ciência. Entre outras coisas, ele criou o cálculo diferencial e integral, a teoria da luz, a gravitação, as leis do movimento, além de, no fim da vida ter – como funcionário público – dirigido a casa da moeda inglesa, pendurando na forca alguns falsários delinquentes. Voltando ao cálculo diferencial, uma das interpretações da derivada de uma função em um ponto é a tangente da curva da função naquele ponto. Veja-se o que se disse neste desenho



- Eis as etapas do método:
1. Obter a derivada da função a resolver

2. Arbitrar um ponto inicial x_0 para a função
3. Arbitrar o erro ϵ
4. Calcular $f(x_0)$ e $df(x_0)$
5. Calcular o “novo” x , $x = x_0 - f(x)/df(x)$
6. Se $|f(x)| > \epsilon$ retornar ao passo 4

Por exemplo, seja calcular as raízes da mesma equação acima que é $e^x - \sin(x) - 2 = 0$. A derivada é $\frac{d}{dx} = e^x - \cos(x)$. Obteve-se a seguinte tabela

```
-----x-----fx-----dfx
1.0000000 -0.12318915634885141 2.1779795
1.0565612 0.00579321190120519 2.3845934
1.0541318 0.00001105001756940 2.3754999
1.0541271 0.00000000004045120 2.3754825
Raiz: 1.0541271241082415
Erro: 4.0451197946822504e-11
Eis o programa Python que resolve este caso
from numpy import *
def newton1():
    x=1
    tol=0.00001
    fx=exp(x)-sin(x)-2
    dfx=exp(x)-cos(x)
    print("-----x-----fx-----dfx")
    print("{:12.7f}{:20.17f}{:12.7f}".format(x,fx,dfx))
    while abs(fx)>tol:
        x=x-fx/dfx
        fx=exp(x)-sin(x)-2
        dfx=exp(x)-cos(x)
        print("{:12.7f}{:20.17f}{:12.7f}".format(x,fx,dfx))
    print("Raiz: ",x)
    print("Erro: ",fx)
newton1()
```

Para você fazer

1. Resolva pelos 3 métodos, para poder comparar, a seguinte equação $x = \sqrt[6]{5013}$ no intervalo entre 4 e 5 e responda a seguir os 3 valores encontrados

bissecção	
cordas	
Newton	

2. Idem $y = 8 * \sin(x) + 4 * \cos(x) - 6.277242 = 0$ no intervalo entre 1 e 2 e responda a seguir os 3 valores encontrados

bissecção	
cordas	
Newton	

3. Idem $y = ((7 * x)/10) * e^{((3 * x)/10)} - 4.248546 = 0$ no intervalo entre 2 e 3 e responda a seguir os 3 valores encontrados

bissecção	
cordas	
Newton	

4. Idem $y = 6 * x^5 + 4 * x^2 - 40.109440 = 0$ no intervalo entre 1 e 2 e responda a seguir os 3 valores encontrados

bissecção	
cordas	
Newton	



Cálculo Numérico - solução de equações

Suponha que precisamos usar o computador para resolver equações. Já paramos para entender o que é resolver uma equação? Conhece-se a forma geral de uma equação $y = f(x)$. Resolver esta equação vem a ser descobrir quais os valores de x que fazem $f(x)$ ser igual a zero. Se olharmos para o gráfico, veremos que busca-se o valor da abscissa x que determina o cruzamento da curva de $f(x)$ no eixo dos x .

Por exemplo, suponha-se precisar descobrir a raiz 11 do número 1739? Ou seja, qual o número que multiplicado por ele mesmo 11 vezes dá como resultado 1739?

Dizendo isso na forma matemática, pode-se escrever $x^{11} = 1739$, ou melhor ainda $x^{11} - 1739 = 0$. Com isso, chegamos ao ponto.

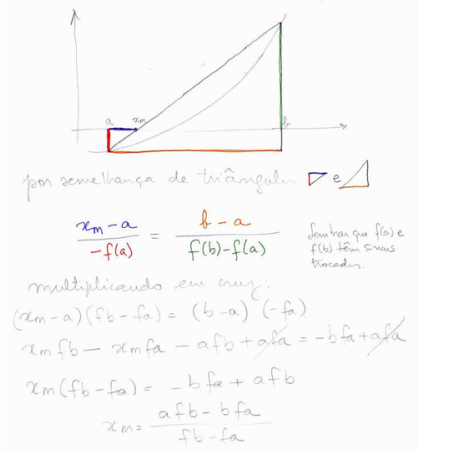
Método da Bissecção Se olharmos o gráfico de uma função, ver-se-á que no entorno da raiz, de um lado a função é positiva, e do outro lado é negativa ou vice versa. Este método começa pela seleção de 2 pontos x_a e x_b sendo que a função $y = f(x)$ têm sinais opostos nesses dois pontos. Graças a essa constatação, pode-se afirmar que a raiz procurada está entre x_a e x_b . Dizendo de outra maneira. O valor de x procurado, está entre a e b . O Método da bissecção vai dividindo o intervalo a, b pela metade, questionando a seguir em qual das metades o x está. Descoberto isso, despreza-se a metade que não contém x . Levando este processo ao limite, descobre-se qual o valor de x . Em resumo, eis as etapas do método:

1. Arbitrar intervalo em $[a, b]$ que compreenda a raiz.
2. Arbitrar um erro permitido ϵ
3. Calcular $x_m = (a + b)/2$
4. Se $f(a).f(x_m) < 0$ o novo intervalo será $[a, x_m]$
5. Se $f(a).f(x_m) > 0$ o novo intervalo será $[x_m, b]$
6. Enquanto o erro $b - a > \epsilon$ voltar ao passo 3.

Seja um exemplo, calcular $f = e^x - \sin(x) - 2 = 0$ nos limites $a = 1$ e $b = 1.25$ com $\epsilon = 0.01$. No primeira etapa tem-se

```
Limite Superior: 1.25
Limite Inferior: 1
Tolerancia: 0.01
-----a-----b-----xm-----fxm-----fa-----fb-----erro
1.1250 1.0000 1.1250 0.1779 0.5414 -0.1232 0.1250
1.0625 1.0000 1.0625 0.0200 0.1779 -0.1232 0.0625
1.0625 1.0312 1.0312 -0.0534 0.0200 -0.1232 0.0312
1.0625 1.0469 1.0469 -0.0171 0.0200 -0.0534 0.0156
1.0547 1.0469 1.0547 0.0013 0.0200 -0.0171 0.0078
Raiz: 1.05078125
Erro: -0.0078125
Eis o programa Python que resolve este caso
import numpy as np
def bissec():
    a=float(input("Limite Superior: "))
    b=float(input("Limite Inferior: "))
    tol=float(input("Tolerancia: "))
    print("-----a-----b-----xm-----fxm-----fa-----fb-----erro")
    while np.abs(b-a)>tol:
        xm=(a+b)/2
        fxm=np.exp(xm)-np.sin(xm)-2
        fa=np.exp(a)-np.sin(a)-2
        fb=np.exp(b)-np.sin(b)-2
        if fa*fxm>0:
            a=xm
        else:
            b=xm
        print('{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}'.format(a,b,xm,fxm,fa,fb,np.abs(b-a)))
    xm=(a+b)/2
    print("Raiz: ",xm)
    print("Erro: ",b-a)
bissec()
```

Cordas ou Falsa Posição



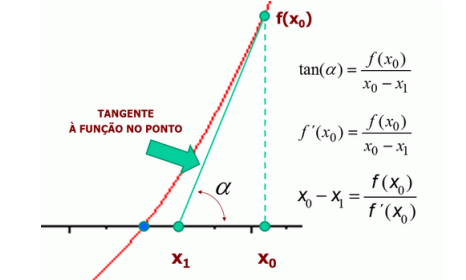
O método começa com uma análise do gráfico da função nas proximidades da raiz. Por semelhança de triângulos, pode-se obter a formulação proposta. Eis os passos do algoritmo:

1. Arbitrar um intervalo $[a, b]$ que compreende a raiz
2. Arbitrar ϵ
3. Calcular $x_m = \frac{a.f(b)-b.f(a)}{f(b)-f(a)}$
4. Se $f(a).f(x_m) < 0$ o novo intervalo será $[a, x_m]$
5. Se $f(a).f(x_m) > 0$ o novo intervalo será $[x_m, b]$
6. Enquanto o erro $f(x_m) > \epsilon$ voltar ao passo 3

Veja a seguir a solução da mesma equação, ($e^x - \sin(x) - 2 = 0$) com os mesmos intervalos do método anterior.

```
Limite inferior: 1
Limite superior: 1.25
Tolerância: 0.001
-----a-----b-----xm-----fxm-----fa-----fb-----erro
1.0463 1.2500 1.0463 -0.0184 -0.1232 0.5414
1.0530 1.2500 1.0530 -0.0026 -0.0184 0.5414
1.0540 1.2500 1.0540 -0.0004 -0.0026 0.5414
Raiz: 1.053972856764054
Erro: -0.0003663936835978099
Eis o programa Python que resolve este caso
from numpy import *
def cordas():
    a=float(input("Limite inferior: "))
    b=float(input("Limite superior: "))
    tol=float(input("Tolerância: "))
    fxm=1
    print("-----a-----b-----xm-----fxm-----fa-----fb-----erro")
    while abs(fxm)>tol:
        fa=exp(a)-sin(a)-2
        fb=exp(b)-sin(b)-2
        fxm=((a*fb)-(b*fa))/(fb-fa)
        fxm=exp(xm)-sin(xm)-2
        if fa*fxm>0:
            a=fxm
        else:
            b=fxm
        print('{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}'.format(a,b,xm,fxm,fa,fb))
    print("Raiz: ",xm)
    print("Erro: ",fxm)
cordas()
```

Método de Newton Este método começa com a interpretação geométrica da derivada. Lembrando, Newton é um monstro da ciência. Entre outras coisas, ele criou o cálculo diferencial e integral, a teoria da luz, a gravitação, as leis do movimento, além de, no fim da vida ter – como funcionário público – dirigido a casa da moeda inglesa, pendurando na forca alguns falsários delinquentes. Voltando ao cálculo diferencial, uma das interpretações da derivada de uma função em um ponto é a tangente da curva da função naquele ponto. Veja-se o que se disse neste desenho



- Eis as etapas do método:
1. Obter a derivada da função a resolver

2. Arbitrar um ponto inicial x_0 para a função
3. Arbitrar o erro ϵ
4. Calcular $f(x_0)$ e $df(x_0)$
5. Calcular o “novo” x , $x = x_0 - f(x)/df(x)$
6. Se $|f(x)| > \epsilon$ retornar ao passo 4

Por exemplo, seja calcular as raízes da mesma equação acima que é $e^x - \sin(x) - 2 = 0$. A derivada é $\frac{d}{dx} = e^x - \cos(x)$. Obteve-se a seguinte tabela

```
-----x-----fx-----dfx
1.0000000 -0.12318915634885141 2.1779795
1.0565612 0.00579321190120519 2.3845934
1.0541318 0.00001105001756940 2.3754999
1.0541271 0.00000000004045120 2.3754825
Raiz: 1.0541271241082415
Erro: 4.0451197946822504e-11
Eis o programa Python que resolve este caso
from numpy import *
def newton1():
    x=1
    tol=0.00001
    fx=exp(x)-sin(x)-2
    dfx=exp(x)-cos(x)
    print("-----x-----fx-----dfx")
    print("{:12.7f}{:20.17f}{:12.7f}".format(x,fx,dfx))
    while abs(fx)>tol:
        x=x-fx/dfx
        fx=exp(x)-sin(x)-2
        dfx=exp(x)-cos(x)
        print("{:12.7f}{:20.17f}{:12.7f}".format(x,fx,dfx))
    print("Raiz: ",x)
    print("Erro: ",fx)
newton1()
```

Para você fazer

1. Resolva pelos 3 métodos, para poder comparar, a seguinte equação $x = \sqrt[12]{5067}$ no intervalo entre 2 e 3 e responda a seguir os 3 valores encontrados

bissecção	
cordas	
Newton	

2. Idem $y = 3*\sin(x) + 8*\cos(x) + 8.373075 = 0$ no intervalo entre 3 e 4 e responda a seguir os 3 valores encontrados

bissecção	
cordas	
Newton	

3. Idem $y = ((6 * x)/10) * e^{((5*x)/10)} - 82.576379 = 0$ no intervalo entre 6 e 7 e responda a seguir os 3 valores encontrados

bissecção	
cordas	
Newton	

4. Idem $y = 6 * x^4 + 3 * x^2 - 47.001600 = 0$ no intervalo entre 1 e 2 e responda a seguir os 3 valores encontrados

bissecção	
cordas	
Newton	



Cálculo Numérico - solução de equações

Suponha que precisamos usar o computador para resolver equações. Já paramos para entender o que é resolver uma equação? Conhece-se a forma geral de uma equação $y = f(x)$. Resolver esta equação vem a ser descobrir quais os valores de x que fazem $f(x)$ ser igual a zero. Se olharmos para o gráfico, veremos que busca-se o valor da abscissa x que determina o cruzamento da curva de $f(x)$ no eixo dos x .

Por exemplo, suponha-se precisar descobrir a raiz 11 do número 1739? Ou seja, qual o número que multiplicado por ele mesmo 11 vezes dá como resultado 1739?

Dizendo isso na forma matemática, pode-se escrever $x^{11} = 1739$, ou melhor ainda $x^{11} - 1739 = 0$. Com isso, chegamos ao ponto.

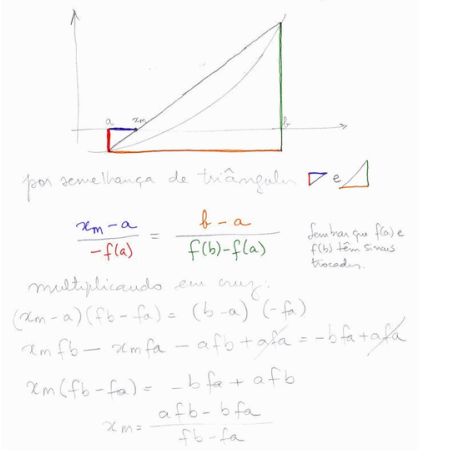
Método da Bissecção Se olharmos o gráfico de uma função, ver-se-á que no entorno da raiz, de um lado a função é positiva, e do outro lado é negativa ou vice versa. Este método começa pela seleção de 2 pontos x_a e x_b sendo que a função $y = f(x)$ têm sinais opostos nesses dois pontos. Graças a essa constatação, pode-se afirmar que a raiz procurada está entre x_a e x_b . Dizendo de outra maneira. O valor de x procurado, está entre a e b . O Método da bissecção vai dividindo o intervalo a, b pela metade, questionando a seguir em qual das metades o x está. Descoberto isso, despreza-se a metade que não contém x . Levando este processo ao limite, descobre-se qual o valor de x . Em resumo, eis as etapas do método:

1. Arbitrar intervalo em $[a, b]$ que compreenda a raiz.
2. Arbitrar um erro permitido ϵ
3. Calcular $x_m = (a + b)/2$
4. Se $f(a).f(x_m) < 0$ o novo intervalo será $[a, x_m]$
5. Se $f(a).f(x_m) > 0$ o novo intervalo será $[x_m, b]$
6. Enquanto o erro $b - a > \epsilon$ voltar ao passo 3.

Seja um exemplo, calcular $f = e^x - \sin(x) - 2 = 0$ nos limites $a = 1$ e $b = 1.25$ com $\epsilon = 0.01$. No primeira etapa tem-se

```
Limite Superior: 1.25
Limite Inferior: 1
Tolerancia: 0.01
-----a-----b-----xm-----fxm-----fa-----fb-----erro
1.1250 1.0000 1.1250 0.1779 0.5414 -0.1232 0.1250
1.0625 1.0000 1.0625 0.0200 0.1779 -0.1232 0.0625
1.0625 1.0312 1.0312 -0.0534 0.0200 -0.1232 0.0312
1.0625 1.0469 1.0469 -0.0171 0.0200 -0.0534 0.0156
1.0547 1.0469 1.0547 0.0013 0.0200 -0.0171 0.0078
Raiz: 1.05078125
Erro: -0.0078125
Eis o programa Python que resolve este caso
import numpy as np
def bissec():
    a=float(input("Limite Superior: "))
    b=float(input("Limite Inferior: "))
    tol=float(input("Tolerancia: "))
    print("-----a-----b-----xm-----fxm-----fa-----fb-----erro")
    while np.abs(b-a)>tol:
        xm=(a+b)/2
        fxm=np.exp(xm)-np.sin(xm)-2
        fa=np.exp(a)-np.sin(a)-2
        fb=np.exp(b)-np.sin(b)-2
        if fa*fxm>0:
            a=xm
        else:
            b=xm
        print('{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}'
              .format(a,b,xm,fxm,fa,fb,np.abs(b-a)))
    xm=(a+b)/2
    print("Raiz: ",xm)
    print("Erro: ",b-a)
bissec()
```

Cordas ou Falsa Posição



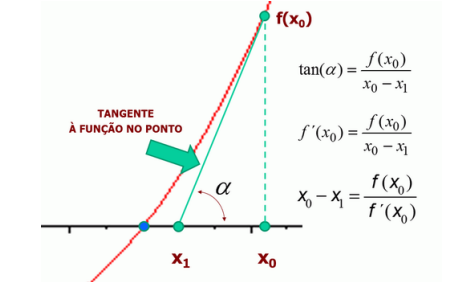
O método começa com uma análise do gráfico da função nas proximidades da raiz. Por semelhança de triângulos, pode-se obter a formulação proposta. Eis os passos do algoritmo:

1. Arbitrar um intervalo $[a, b]$ que compreende a raiz
2. Arbitrar ϵ
3. Calcular $x_m = \frac{a \cdot f(b) - b \cdot f(a)}{f(b) - f(a)}$
4. Se $f(a).f(x_m) < 0$ o novo intervalo será $[a, x_m]$
5. Se $f(a).f(x_m) > 0$ o novo intervalo será $[x_m, b]$
6. Enquanto o erro $f(x_m) > \epsilon$ voltar ao passo 3

Veja a seguir a solução da mesma equação, ($e^x - \sin(x) - 2 = 0$) com os mesmos intervalos do método anterior.

```
Limite inferior: 1
Limite superior: 1.25
Tolerância: 0.001
-----a-----b-----xm-----fxm-----fa-----fb-----erro
1.0463 1.2500 1.0463 -0.0184 -0.1232 0.5414
1.0530 1.2500 1.0530 -0.0026 -0.0184 0.5414
1.0540 1.2500 1.0540 -0.0004 -0.0026 0.5414
Raiz: 1.053972856764054
Erro: -0.0003663936835978099
Eis o programa Python que resolve este caso
from numpy import *
def cordas():
    a=float(input("Limite inferior: "))
    b=float(input("Limite superior: "))
    tol=float(input("Tolerância: "))
    fxm=1
    print("-----a-----b-----xm-----fxm-----fa-----fb-----erro")
    while abs(fxm)>tol:
        fa=exp(a)-sin(a)-2
        fb=exp(b)-sin(b)-2
        xm=((a*fb)-(b*fa))/(fb-fa)
        fxm=exp(xm)-sin(xm)-2
        if fa*fxm>0:
            a=xm
        else:
            b=xm
        print('{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}'
              .format(a,b,xm,fxm,fa,fb))
    print("Raiz: ",xm)
    print("Erro: ",fxm)
cordas()
```

Método de Newton Este método começa com a interpretação geométrica da derivada. Lembrando, Newton é um monstro da ciência. Entre outras coisas, ele criou o cálculo diferencial e integral, a teoria da luz, a gravitação, as leis do movimento, além de, no fim da vida ter – como funcionário público – dirigido a casa da moeda inglesa, pendurando na forca alguns falsários delinquentes. Voltando ao cálculo diferencial, uma das interpretações da derivada de uma função em um ponto é a tangente da curva da função naquele ponto. Veja-se o que se disse neste desenho



Eis as etapas do método:

1. Obter a derivada da função a resolver

2. Arbitrar um ponto inicial x_0 para a função
3. Arbitrar o erro ϵ
4. Calcular $f(x_0)$ e $df(x_0)$
5. Calcular o “novo” x , $x = x_0 - f(x)/df(x)$
6. Se $|f(x)| > \epsilon$ retornar ao passo 4

Por exemplo, seja calcular as raízes da mesma equação acima que é $e^x - \sin(x) - 2 = 0$. A derivada é $\frac{d}{dx} = e^x - \cos(x)$. Obteve-se a seguinte tabela

```
-----x-----fx-----dfx
1.0000000 -0.12318915634885141 2.1779795
1.0565612 0.00579321190120519 2.3845934
1.0541318 0.00001105001756940 2.3754999
1.0541271 0.00000000004045120 2.3754825
Raiz: 1.0541271241082415
Erro: 4.0451197946822504e-11
Eis o programa Python que resolve este caso
from numpy import *
def newton1():
    x=1
    tol=0.00001
    fx=exp(x)-sin(x)-2
    dfx=exp(x)-cos(x)
    print("-----x-----fx-----dfx")
    print("{:12.7f}{:20.17f}{:12.7f}"
          .format(x,fx,dfx))
    while abs(fx)>tol:
        x=x-fx/dfx
        fx=exp(x)-sin(x)-2
        dfx=exp(x)-cos(x)
        print("{:12.7f}{:20.17f}{:12.7f}"
              .format(x,fx,dfx))
    print("Raiz: ",x)
    print("Erro: ",fx)
newton1()
```

Para você fazer

1. Resolva pelos 3 métodos, para poder comparar, a seguinte equação $x = \sqrt[18]{5433}$ no intervalo entre 1 e 2 e responda a seguir os 3 valores encontrados

bissecção	
cordas	
Newton	

2. Idem $y = 4 \sin(x) + 5 \cos(x) - 1.352763 = 0$ no intervalo entre 5 e 6 e responda a seguir os 3 valores encontrados

bissecção	
cordas	
Newton	

3. Idem $y = ((6 * x)/10) * e^{((3*x)/10)} - 10.415049 = 0$ no intervalo entre 4 e 5 e responda a seguir os 3 valores encontrados

bissecção	
cordas	
Newton	

4. Idem $y = 5 * x^4 + 3 * x^2 - 31608.750500 = 0$ no intervalo entre 8 e 9 e responda a seguir os 3 valores encontrados

bissecção	
cordas	
Newton	



Cálculo Numérico - solução de equações

Suponha que precisamos usar o computador para resolver equações. Já paramos para entender o que é resolver uma equação? Conhece-se a forma geral de uma equação $y = f(x)$. Resolver esta equação vem a ser descobrir quais os valores de x que fazem $f(x)$ ser igual a zero. Se olharmos para o gráfico, veremos que busca-se o valor da abscissa x que determina o cruzamento da curva de $f(x)$ no eixo dos x .

Por exemplo, suponha-se precisar descobrir a raiz 11 do número 1739? Ou seja, qual o número que multiplicado por ele mesmo 11 vezes dá como resultado 1739?

Dizendo isso na forma matemática, pode-se escrever $x^{11} = 1739$, ou melhor ainda $x^{11} - 1739 = 0$. Com isso, chegamos ao ponto.

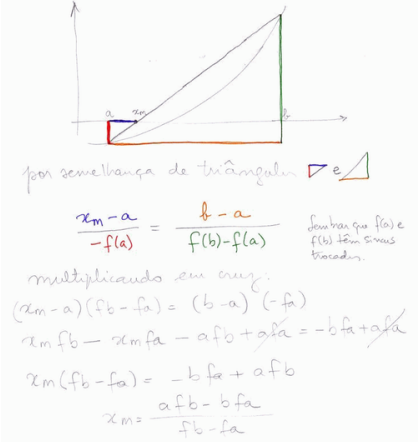
Método da Bissecção Se olharmos o gráfico de uma função, ver-se-á que no entorno da raiz, de um lado a função é positiva, e do outro lado é negativa ou vice versa. Este método começa pela seleção de 2 pontos x_a e x_b sendo que a função $y = f(x)$ têm sinais opostos nesses dois pontos. Graças a essa constatação, pode-se afirmar que a raiz procurada está entre x_a e x_b . Dizendo de outra maneira. O valor de x procurado, está entre a e b . O Método da bissecção vai dividindo o intervalo a, b pela metade, questionando a seguir em qual das metades o x está. Descoberto isso, despreza-se a metade que não contém x . Levando este processo ao limite, descobre-se qual o valor de x . Em resumo, eis as etapas do método:

1. Arbitrar intervalo em $[a, b]$ que compreenda a raiz.
2. Arbitrar um erro permitido ϵ
3. Calcular $x_m = (a + b)/2$
4. Se $f(a).f(x_m) < 0$ o novo intervalo será $[a, x_m]$
5. Se $f(a).f(x_m) > 0$ o novo intervalo será $[x_m, b]$
6. Enquanto o erro $b - a > \epsilon$ voltar ao passo 3.

Seja um exemplo, calcular $f = e^x - \sin(x) - 2 = 0$ nos limites $a = 1$ e $b = 1.25$ com $\epsilon = 0.01$. No primeira etapa tem-se

```
Limite Superior: 1.25
Limite Inferior: 1
Tolerancia: 0.01
-----a-----b-----xm-----fxm-----fa-----fb-----erro
1.1250 1.0000 1.1250 0.1779 0.5414 -0.1232 0.1250
1.0625 1.0000 1.0625 0.0200 0.1779 -0.1232 0.0625
1.0625 1.0312 1.0312 -0.0534 0.0200 -0.1232 0.0312
1.0625 1.0469 1.0469 -0.0171 0.0200 -0.0534 0.0156
1.0547 1.0469 1.0547 0.0013 0.0200 -0.0171 0.0078
Raiz: 1.05078125
Erro: -0.0078125
Eis o programa Python que resolve este caso
import numpy as np
def bissec():
    a=float(input("Limite Superior: "))
    b=float(input("Limite Inferior: "))
    tol=float(input("Tolerancia: "))
    print("-----a-----b-----xm-----fxm-----fa-----fb-----erro")
    while np.abs(b-a)>tol:
        xm=(a+b)/2
        fxm=np.exp(xm)-np.sin(xm)-2
        fa=np.exp(a)-np.sin(a)-2
        fb=np.exp(b)-np.sin(b)-2
        if fa*fxm>0:
            a=xm
        else:
            b=xm
        print('{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}'.format(a,b,xm,fxm,fa,fb,np.abs(b-a)))
    xm=(a+b)/2
    print("Raiz: ",xm)
    print("Erro: ",b-a)
bissec()
```

Cordas ou Falsa Posição



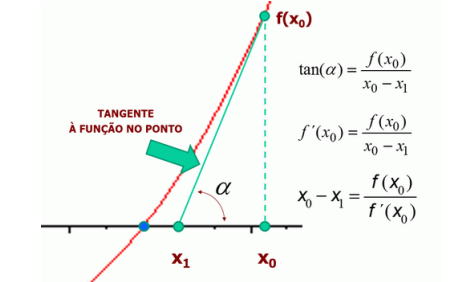
O método começa com uma análise do gráfico da função nas proximidades da raiz. Por semelhança de triângulos, pode-se obter a formulação proposta. Eis os passos do algoritmo:

1. Arbitrar um intervalo $[a, b]$ que compreende a raiz
2. Arbitrar ϵ
3. Calcular $x_m = \frac{a \cdot f(b) - b \cdot f(a)}{f(b) - f(a)}$
4. Se $f(a).f(x_m) < 0$ o novo intervalo será $[a, x_m]$
5. Se $f(a).f(x_m) > 0$ o novo intervalo será $[x_m, b]$
6. Enquanto o erro $f(x_m) > \epsilon$ voltar ao passo 3

Veja a seguir a solução da mesma equação, ($e^x - \sin(x) - 2 = 0$) com os mesmos intervalos do método anterior.

```
Limite inferior: 1
Limite superior: 1.25
Tolerância: 0.001
-----a-----b-----xm-----fxm-----fa-----fb-----erro
1.0463 1.2500 1.0463 -0.0184 -0.1232 0.5414
1.0530 1.2500 1.0530 -0.0026 -0.0184 0.5414
1.0540 1.2500 1.0540 -0.0004 -0.0026 0.5414
Raiz: 1.053972856764054
Erro: -0.0003663936835978099
Eis o programa Python que resolve este caso
from numpy import *
def cordas():
    a=float(input("Limite inferior: "))
    b=float(input("Limite superior: "))
    tol=float(input("Tolerância: "))
    fxm=1
    print("-----a-----b-----xm-----fxm-----fa-----fb-----erro")
    while abs(fxm)>tol:
        fa=exp(a)-sin(a)-2
        fb=exp(b)-sin(b)-2
        xm=((a*fb)-(b*fa))/(fb-fa)
        fxm=exp(xm)-sin(xm)-2
        if fa*fxm>0:
            a=xm
        else:
            b=xm
        print('{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}'.format(a,b,xm,fxm,fa,fb))
    print("Raiz: ",xm)
    print("Erro: ",fxm)
cordas()
```

Método de Newton Este método começa com a interpretação geométrica da derivada. Lembrando, Newton é um monstro da ciência. Entre outras coisas, ele criou o cálculo diferencial e integral, a teoria da luz, a gravitação, as leis do movimento, além de, no fim da vida ter – como funcionário público – dirigido a casa da moeda inglesa, pendurando na força alguns falsários delinquentes. Voltando ao cálculo diferencial, uma das interpretações da derivada de uma função em um ponto é a tangente da curva da função naquele ponto. Veja-se o que se disse neste desenho



- Eis as etapas do método:
1. Obter a derivada da função a resolver

2. Arbitrar um ponto inicial x_0 para a função
3. Arbitrar o erro ϵ
4. Calcular $f(x_0)$ e $df(x_0)$
5. Calcular o “novo” x , $x = x_0 - f(x)/df(x)$
6. Se $|f(x)| > \epsilon$ retornar ao passo 4

Por exemplo, seja calcular as raízes da mesma equação acima que é $e^x - \sin(x) - 2 = 0$. A derivada é $\frac{d}{dx} = e^x - \cos(x)$. Obteve-se a seguinte tabela

```
-----x-----fx-----dfx
1.0000000 -0.12318915634885141 2.1779795
1.0565612 0.00579321190120519 2.3845934
1.0541318 0.00001105001756940 2.3754999
1.0541271 0.00000000004045120 2.3754825
Raiz: 1.0541271241082415
Erro: 4.0451197946822504e-11
Eis o programa Python que resolve este caso
from numpy import *
def newton():
    x=1
    tol=0.00001
    fx=exp(x)-sin(x)-2
    dfx=exp(x)-cos(x)
    print("-----x-----fx-----dfx")
    print("{:12.7f}{:20.17f}{:12.7f}".format(x,fx,dfx))
    while abs(fx)>tol:
        x=x-fx/dfx
        fx=exp(x)-sin(x)-2
        dfx=exp(x)-cos(x)
        print("{:12.7f}{:20.17f}{:12.7f}".format(x,fx,dfx))
    print("Raiz: ",x)
    print("Erro: ",fx)
newton()
```

Para você fazer

1. Resolva pelos 3 métodos, para poder comparar, a seguinte equação $x = \sqrt[11]{9695}$ no intervalo entre 1 e 2 e responda a seguir os 3 valores encontrados

bissecção	
cordas	
Newton	

2. Idem $y = 3 \cdot \sin(x) + 7 \cdot \cos(x) - 2.674809 = 0$ no intervalo entre 7 e 8 e responda a seguir os 3 valores encontrados

bissecção	
cordas	
Newton	

3. Idem $y = ((6 \cdot x)/10) \cdot e^{((5 \cdot x)/10)} - 336.099108 = 0$ no intervalo entre 8 e 9 e responda a seguir os 3 valores encontrados

bissecção	
cordas	
Newton	

4. Idem $y = 5 \cdot x^4 + 3 \cdot x^2 - 6163.110500 = 0$ no intervalo entre 5 e 6 e responda a seguir os 3 valores encontrados

bissecção	
cordas	
Newton	



Cálculo Numérico - solução de equações

Suponha que precisamos usar o computador para resolver equações. Já paramos para entender o que é resolver uma equação? Conhece-se a forma geral de uma equação $y = f(x)$. Resolver esta equação vem a ser descobrir quais os valores de x que fazem $f(x)$ ser igual a zero. Se olharmos para o gráfico, veremos que busca-se o valor da abscissa x que determina o cruzamento da curva de $f(x)$ no eixo dos x .

Por exemplo, suponha-se precisar descobrir a raiz 11 do número 1739? Ou seja, qual o número que multiplicado por ele mesmo 11 vezes dá como resultado 1739?

Dizendo isso na forma matemática, pode-se escrever $x^{11} = 1739$, ou melhor ainda $x^{11} - 1739 = 0$. Com isso, chegamos ao ponto.

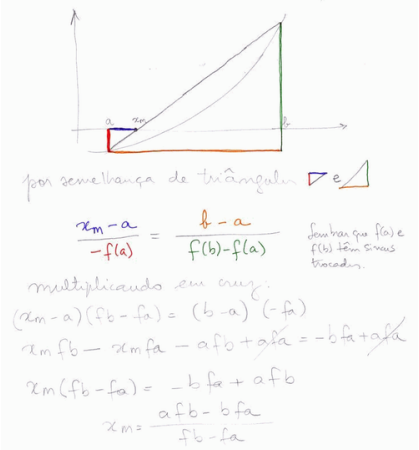
Método da Bissecção Se olharmos o gráfico de uma função, ver-se-á que no entorno da raiz, de um lado a função é positiva, e do outro lado é negativa ou vice versa. Este método começa pela seleção de 2 pontos x_a e x_b sendo que a função $y = f(x)$ têm sinais opostos nesses dois pontos. Graças a essa constatação, pode-se afirmar que a raiz procurada está entre x_a e x_b . Dizendo de outra maneira. O valor de x procurado, está entre a e b . O Método da bissecção vai dividindo o intervalo a, b pela metade, questionando a seguir em qual das metades o x está. Descoberto isso, despreza-se a metade que não contém x . Levando este processo ao limite, descobre-se qual o valor de x . Em resumo, eis as etapas do método:

1. Arbitrar intervalo em $[a, b]$ que compreenda a raiz.
2. Arbitrar um erro permitido ϵ
3. Calcular $x_m = (a + b)/2$
4. Se $f(a).f(x_m) < 0$ o novo intervalo será $[a, x_m]$
5. Se $f(a).f(x_m) > 0$ o novo intervalo será $[x_m, b]$
6. Enquanto o erro $b - a > \epsilon$ voltar ao passo 3.

Seja um exemplo, calcular $f = e^x - \sin(x) - 2 = 0$ nos limites $a = 1$ e $b = 1.25$ com $\epsilon = 0.01$. No primeira etapa tem-se

```
Limite Superior: 1.25
Limite Inferior: 1
Tolerancia: 0.01
-----a-----b-----xm-----fxm-----fa-----fb-----erro
1.1250 1.0000 1.1250 0.1779 0.5414 -0.1232 0.1250
1.0625 1.0000 1.0625 0.0200 0.1779 -0.1232 0.0625
1.0625 1.0312 1.0312 -0.0534 0.0200 -0.1232 0.0312
1.0625 1.0469 1.0469 -0.0171 0.0200 -0.0534 0.0156
1.0547 1.0469 1.0547 0.0013 0.0200 -0.0171 0.0078
Raiz: 1.05078125
Erro: -0.0078125
Eis o programa Python que resolve este caso
import numpy as np
def bissec():
    a=float(input("Limite Superior: "))
    b=float(input("Limite Inferior: "))
    tol=float(input("Tolerancia: "))
    print("-----a-----b-----xm-----fxm-----fa-----fb-----erro")
    while np.abs(b-a)>tol:
        xm=(a+b)/2
        fxm=np.exp(xm)-np.sin(xm)-2
        fa=np.exp(a)-np.sin(a)-2
        fb=np.exp(b)-np.sin(b)-2
        if fa*fxm>0:
            a=xm
        else:
            b=xm
        print('{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}'
              .format(a,b,xm,fxm,fa,fb,np.abs(b-a)))
    xm=(a+b)/2
    print("Raiz: ",xm)
    print("Erro: ",b-a)
bissec()
```

Cordas ou Falsa Posição



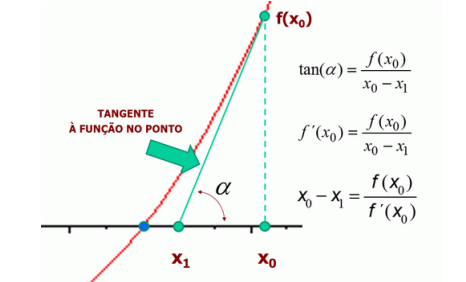
O método começa com uma análise do gráfico da função nas proximidades da raiz. Por semelhança de triângulos, pode-se obter a formulação proposta. Eis os passos do algoritmo:

1. Arbitrar um intervalo $[a, b]$ que compreende a raiz
2. Arbitrar ϵ
3. Calcular $x_m = \frac{a \cdot f(b) - b \cdot f(a)}{f(b) - f(a)}$
4. Se $f(a).f(x_m) < 0$ o novo intervalo será $[a, x_m]$
5. Se $f(a).f(x_m) > 0$ o novo intervalo será $[x_m, b]$
6. Enquanto o erro $f(x_m) > \epsilon$ voltar ao passo 3

Veja a seguir a solução da mesma equação, ($e^x - \sin(x) - 2 = 0$) com os mesmos intervalos do método anterior.

```
Limite inferior: 1
Limite superior: 1.25
Tolerância: 0.001
-----a-----b-----xm-----fxm-----fa-----fb-----erro
1.0463 1.2500 1.0463 -0.0184 -0.1232 0.5414
1.0530 1.2500 1.0530 -0.0026 -0.0184 0.5414
1.0540 1.2500 1.0540 -0.0004 -0.0026 0.5414
Raiz: 1.053972856764054
Erro: -0.0003663936835978099
Eis o programa Python que resolve este caso
from numpy import *
def cordas():
    a=float(input("Limite inferior: "))
    b=float(input("Limite superior: "))
    tol=float(input("Tolerância: "))
    fxm=1
    print("-----a-----b-----xm-----fxm-----fa-----fb-----erro")
    while abs(fxm)>tol:
        fa=exp(a)-sin(a)-2
        fb=exp(b)-sin(b)-2
        fxm=((a*fb)-(b*fa))/(fb-fa)
        fxm=exp(xm)-sin(xm)-2
        if fa*fxm>0:
            a=fxm
        else:
            b=fxm
        print('{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}'
              .format(a,b,xm,fxm,fa,fb))
    print("Raiz: ",xm)
    print("Erro: ",fxm)
cordas()
```

Método de Newton Este método começa com a interpretação geométrica da derivada. Lembrando, Newton é um monstro da ciência. Entre outras coisas, ele criou o cálculo diferencial e integral, a teoria da luz, a gravitação, as leis do movimento, além de, no fim da vida ter – como funcionário público – dirigido a casa da moeda inglesa, pendurando na forca alguns falsários delinquentes. Voltando ao cálculo diferencial, uma das interpretações da derivada de uma função em um ponto é a tangente da curva da função naquele ponto. Veja-se o que se disse neste desenho



Eis as etapas do método:

1. Obter a derivada da função a resolver

2. Arbitrar um ponto inicial x_0 para a função
3. Arbitrar o erro ϵ
4. Calcular $f(x_0)$ e $df(x_0)$
5. Calcular o “novo” x , $x = x_0 - f(x)/df(x)$
6. Se $|f(x)| > \epsilon$ retornar ao passo 4

Por exemplo, seja calcular as raízes da mesma equação acima que é $e^x - \sin(x) - 2 = 0$. A derivada é $\frac{d}{dx} = e^x - \cos(x)$. Obteve-se a seguinte tabela

```
-----x-----fx-----dfx
1.0000000 -0.12318915634885141 2.1779795
1.0565612 0.00579321190120519 2.3845934
1.0541318 0.00001105001756940 2.3754999
1.0541271 0.0000000004045120 2.3754825
Raiz: 1.0541271241082415
Erro: 4.0451197946822504e-11
Eis o programa Python que resolve este caso
from numpy import *
def newton():
    x=1
    tol=0.00001
    fx=exp(x)-sin(x)-2
    dfx=exp(x)-cos(x)
    print("-----x-----fx-----dfx")
    print("{:12.7f}{:20.17f}{:12.7f}".format(x,fx,dfx))
    while abs(fx)>tol:
        x=x-fx/dfx
        fx=exp(x)-sin(x)-2
        dfx=exp(x)-cos(x)
        print("{:12.7f}{:20.17f}{:12.7f}".format(x,fx,dfx))
    print("Raiz: ",x)
    print("Erro: ",fx)
newton()
```

Para você fazer

1. Resolva pelos 3 métodos, para poder comparar, a seguinte equação $x = \sqrt[7]{8080}$ no intervalo entre 3 e 4 e responda a seguir os 3 valores encontrados

bissecção	
cordas	
Newton	

2. Idem $y = 5 \sin(x) + 3 \cos(x) + 5.616628 = 0$ no intervalo entre 3 e 4 e responda a seguir os 3 valores encontrados

bissecção	
cordas	
Newton	

3. Idem $y = ((6 * x)/10) * e^{((7 * x)/10)} - .222062 = 0$ no intervalo entre 0 e 1 e responda a seguir os 3 valores encontrados

bissecção	
cordas	
Newton	

4. Idem $y = 7 * x^6 + 4 * x^2 - 5831180.394503 = 0$ no intervalo entre 9 e 10 e responda a seguir os 3 valores encontrados

bissecção	
cordas	
Newton	



Cálculo Numérico - solução de equações

Suponha que precisamos usar o computador para resolver equações. Já paramos para entender o que é resolver uma equação? Conhece-se a forma geral de uma equação $y = f(x)$. Resolver esta equação vem a ser descobrir quais os valores de x que fazem $f(x)$ ser igual a zero. Se olharmos para o gráfico, veremos que busca-se o valor da abscissa x que determina o cruzamento da curva de $f(x)$ no eixo dos x .

Por exemplo, suponha-se precisar descobrir a raiz 11 do número 1739? Ou seja, qual o número que multiplicado por ele mesmo 11 vezes dá como resultado 1739?

Dizendo isso na forma matemática, pode-se escrever $x^{11} = 1739$, ou melhor ainda $x^{11} - 1739 = 0$. Com isso, chegamos ao ponto.

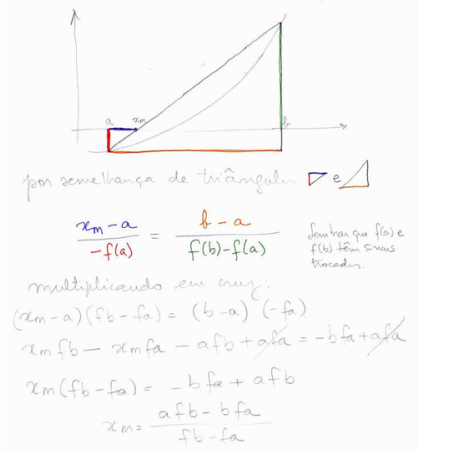
Método da Bissecção Se olharmos o gráfico de uma função, ver-se-á que no entorno da raiz, de um lado a função é positiva, e do outro lado é negativa ou vice versa. Este método começa pela seleção de 2 pontos x_a e x_b sendo que a função $y = f(x)$ têm sinais opostos nesses dois pontos. Graças a essa constatação, pode-se afirmar que a raiz procurada está entre x_a e x_b . Dizendo de outra maneira. O valor de x procurado, está entre a e b . O Método da bissecção vai dividindo o intervalo a, b pela metade, questionando a seguir em qual das metades o x está. Descoberto isso, despreza-se a metade que não contém x . Levando este processo ao limite, descobre-se qual o valor de x . Em resumo, eis as etapas do método:

1. Arbitrar intervalo em $[a, b]$ que compreenda a raiz.
2. Arbitrar um erro permitido ϵ
3. Calcular $x_m = (a + b)/2$
4. Se $f(a) \cdot f(x_m) < 0$ o novo intervalo será $[a, x_m]$
5. Se $f(a) \cdot f(x_m) > 0$ o novo intervalo será $[x_m, b]$
6. Enquanto o erro $b - a > \epsilon$ voltar ao passo 3.

Seja um exemplo, calcular $f = e^x - \sin(x) - 2 = 0$ nos limites $a = 1$ e $b = 1.25$ com $\epsilon = 0.01$. No primeira etapa tem-se

```
Limite Superior: 1.25
Limite Inferior: 1
Tolerancia: 0.01
-----a-----b-----xm-----fxm-----fa-----fb-----erro
1.1250 1.0000 1.1250 0.1779 0.5414 -0.1232 0.1250
1.0625 1.0000 1.0625 0.0200 0.1779 -0.1232 0.0625
1.0625 1.0312 1.0312 -0.0534 0.0200 -0.1232 0.0312
1.0625 1.0469 1.0469 -0.0171 0.0200 -0.0534 0.0156
1.0547 1.0469 1.0547 0.0013 0.0200 -0.0171 0.0078
Raiz: 1.05078125
Erro: -0.0078125
Eis o programa Python que resolve este caso
import numpy as np
def bissec():
    a=float(input("Limite Superior: "))
    b=float(input("Limite Inferior: "))
    tol=float(input("Tolerancia: "))
    print("-----a-----b-----xm-----fxm-----fa-----fb-----erro")
    while np.abs(b-a)>tol:
        xm=(a+b)/2
        fxm=np.exp(xm)-np.sin(xm)-2
        fa=np.exp(a)-np.sin(a)-2
        fb=np.exp(b)-np.sin(b)-2
        if fa*fxm>0:
            a=xm
        else:
            b=xm
        print('{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}'
              .format(a,b,xm,fxm,fa,fb,np.abs(b-a)))
    xm=(a+b)/2
    print("Raiz: ",xm)
    print("Erro: ",b-a)
bissec()
```

Cordas ou Falsa Posição



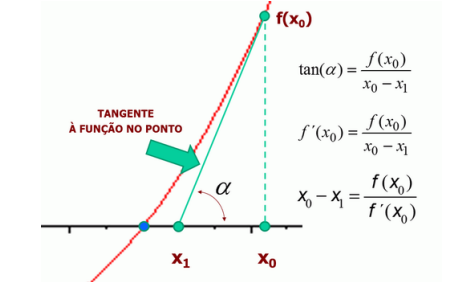
O método começa com uma análise do gráfico da função nas proximidades da raiz. Por semelhança de triângulos, pode-se obter a formulação proposta. Eis os passos do algoritmo:

1. Arbitrar um intervalo $[a, b]$ que compreende a raiz
2. Arbitrar ϵ
3. Calcular $x_m = \frac{a \cdot f(b) - b \cdot f(a)}{f(b) - f(a)}$
4. Se $f(a) \cdot f(x_m) < 0$ o novo intervalo será $[a, x_m]$
5. Se $f(a) \cdot f(x_m) > 0$ o novo intervalo será $[x_m, b]$
6. Enquanto o erro $f(x_m) > \epsilon$ voltar ao passo 3

Veja a seguir a solução da mesma equação, ($e^x - \sin(x) - 2 = 0$) com os mesmos intervalos do método anterior.

```
Limite inferior: 1
Limite superior: 1.25
Tolerância: 0.001
-----a-----b-----xm-----fxm-----fa-----fb-----erro
1.0463 1.2500 1.0463 -0.0184 -0.1232 0.5414
1.0530 1.2500 1.0530 -0.0026 -0.0184 0.5414
1.0540 1.2500 1.0540 -0.0004 -0.0026 0.5414
Raiz: 1.053972856764054
Erro: -0.0003663936835978099
Eis o programa Python que resolve este caso
from numpy import *
def cordas():
    a=float(input("Limite inferior: "))
    b=float(input("Limite superior: "))
    tol=float(input("Tolerância: "))
    fxm=1
    print("-----a-----b-----xm-----fxm-----fa-----fb-----erro")
    while abs(fxm)>tol:
        fa=exp(a)-sin(a)-2
        fb=exp(b)-sin(b)-2
        xm=((a*fb)-(b*fa))/(fb-fa)
        fxm=exp(xm)-sin(xm)-2
        if fa*fxm>0:
            a=xm
        else:
            b=xm
        print('{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}'
              .format(a,b,xm,fxm,fa,fb))
    print("Raiz: ",xm)
    print("Erro: ",fxm)
cordas()
```

Método de Newton Este método começa com a interpretação geométrica da derivada. Lembrando, Newton é um monstro da ciência. Entre outras coisas, ele criou o cálculo diferencial e integral, a teoria da luz, a gravitação, as leis do movimento, além de, no fim da vida ter – como funcionário público – dirigido a casa da moeda inglesa, pendurando na forca alguns falsários delinquentes. Voltando ao cálculo diferencial, uma das interpretações da derivada de uma função em um ponto é a tangente da curva da função naquele ponto. Veja-se o que se disse neste desenho



Eis as etapas do método:

1. Obter a derivada da função a resolver

2. Arbitrar um ponto inicial x_0 para a função
3. Arbitrar o erro ϵ
4. Calcular $f(x_0)$ e $df(x_0)$
5. Calcular o “novo” x , $x = x_0 - f(x)/df(x)$
6. Se $|f(x)| > \epsilon$ retornar ao passo 4

Por exemplo, seja calcular as raízes da mesma equação acima que é $e^x - \sin(x) - 2 = 0$. A derivada é $\frac{d}{dx} = e^x - \cos(x)$. Obteve-se a seguinte tabela

```
-----x-----fx-----dfx
1.0000000 -0.12318915634885141 2.1779795
1.0565612 0.00579321190120519 2.3845934
1.0541318 0.00001105001756940 2.3754999
1.0541271 0.0000000004045120 2.3754825
Raiz: 1.0541271241082415
Erro: 4.0451197946822504e-11
Eis o programa Python que resolve este caso
from numpy import *
def newton1():
    x=1
    tol=0.00001
    fx=exp(x)-sin(x)-2
    dfx=exp(x)-cos(x)
    print("-----x-----fx-----dfx")
    print("{:12.7f}{:20.17f}{:12.7f}".format(x,fx,dfx))
    while abs(fx)>tol:
        x=x-fx/dfx
        fx=exp(x)-sin(x)-2
        dfx=exp(x)-cos(x)
        print("{:12.7f}{:20.17f}{:12.7f}".format(x,fx,dfx))
    print("Raiz: ",x)
    print("Erro: ",fx)
newton1()
```

Para você fazer

1. Resolva pelos 3 métodos, para poder comparar, a seguinte equação $x = \sqrt[24]{5676}$ no intervalo entre 1 e 2 e responda a seguir os 3 valores encontrados

bissecção	
cordas	
Newton	

2. Idem $y = 6 * \sin(x) + 4 * \cos(x) - 3.487632 = 0$ no intervalo entre 6 e 7 e responda a seguir os 3 valores encontrados

bissecção	
cordas	
Newton	

3. Idem $y = ((6 * x)/10) * e^{((5 * x)/10)} - .846889 = 0$ no intervalo entre 0 e 1 e responda a seguir os 3 valores encontrados

bissecção	
cordas	
Newton	

4. Idem $y = 6 * x^5 + 4 * x^3 - 9138.534580 = 0$ no intervalo entre 4 e 5 e responda a seguir os 3 valores encontrados

bissecção	
cordas	
Newton	



Cálculo Numérico - solução de equações

Suponha que precisamos usar o computador para resolver equações. Já paramos para entender o que é resolver uma equação? Conhece-se a forma geral de uma equação $y = f(x)$. Resolver esta equação vem a ser descobrir quais os valores de x que fazem $f(x)$ ser igual a zero. Se olharmos para o gráfico, veremos que busca-se o valor da abscissa x que determina o cruzamento da curva de $f(x)$ no eixo dos x .

Por exemplo, suponha-se precisar descobrir a raiz 11 do número 1739? Ou seja, qual o número que multiplicado por ele mesmo 11 vezes dá como resultado 1739?

Dizendo isso na forma matemática, pode-se escrever $x^{11} = 1739$, ou melhor ainda $x^{11} - 1739 = 0$. Com isso, chegamos ao ponto.

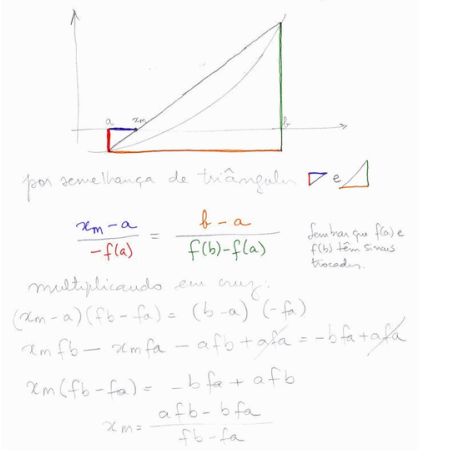
Método da Bissecção Se olharmos o gráfico de uma função, ver-se-á que no entorno da raiz, de um lado a função é positiva, e do outro lado é negativa ou vice versa. Este método começa pela seleção de 2 pontos x_a e x_b sendo que a função $y = f(x)$ têm sinais opostos nesses dois pontos. Graças a essa constatação, pode-se afirmar que a raiz procurada está entre x_a e x_b . Dizendo de outra maneira. O valor de x procurado, está entre a e b . O Método da bissecção vai dividindo o intervalo a, b pela metade, questionando a seguir em qual das metades o x está. Descoberto isso, despreza-se a metade que não contém x . Levando este processo ao limite, descobre-se qual o valor de x . Em resumo, eis as etapas do método:

1. Arbitrar intervalo em $[a, b]$ que compreenda a raiz.
2. Arbitrar um erro permitido ϵ
3. Calcular $x_m = (a + b)/2$
4. Se $f(a).f(x_m) < 0$ o novo intervalo será $[a, x_m]$
5. Se $f(a).f(x_m) > 0$ o novo intervalo será $[x_m, b]$
6. Enquanto o erro $b - a > \epsilon$ voltar ao passo 3.

Seja um exemplo, calcular $f = e^x - \sin(x) - 2 = 0$ nos limites $a = 1$ e $b = 1.25$ com $\epsilon = 0.01$. No primeira etapa tem-se

```
Limite Superior: 1.25
Limite Inferior: 1
Tolerancia: 0.01
-----a-----b-----xm-----fxm-----fa-----fb-----erro
1.1250 1.0000 1.1250 0.1779 0.5414 -0.1232 0.1250
1.0625 1.0000 1.0625 0.0200 0.1779 -0.1232 0.0625
1.0625 1.0312 1.0312 -0.0534 0.0200 -0.1232 0.0312
1.0625 1.0469 1.0469 -0.0171 0.0200 -0.0534 0.0156
1.0547 1.0469 1.0547 0.0013 0.0200 -0.0171 0.0078
Raiz: 1.05078125
Erro: -0.0078125
Eis o programa Python que resolve este caso
import numpy as np
def bissec():
    a=float(input("Limite Superior: "))
    b=float(input("Limite Inferior: "))
    tol=float(input("Tolerancia: "))
    print("-----a-----b-----xm-----fxm-----fa-----fb-----erro")
    while np.abs(b-a)>tol:
        xm=(a+b)/2
        fxm=np.exp(xm)-np.sin(xm)-2
        fa=np.exp(a)-np.sin(a)-2
        fb=np.exp(b)-np.sin(b)-2
        if fa*fxm>0:
            a=xm
        else:
            b=xm
        print('{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}'
              .format(a,b,xm,fxm,fa,fb,np.abs(b-a)))
    xm=(a+b)/2
    print("Raiz: ",xm)
    print("Erro: ",b-a)
bissec()
```

Cordas ou Falsa Posição



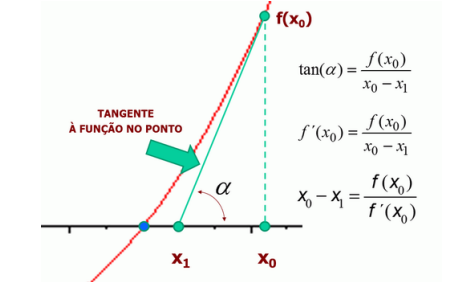
O método começa com uma análise do gráfico da função nas proximidades da raiz. Por semelhança de triângulos, pode-se obter a formulação proposta. Eis os passos do algoritmo:

1. Arbitrar um intervalo $[a, b]$ que compreende a raiz
2. Arbitrar ϵ
3. Calcular $x_m = \frac{a \cdot f(b) - b \cdot f(a)}{f(b) - f(a)}$
4. Se $f(a).f(x_m) < 0$ o novo intervalo será $[a, x_m]$
5. Se $f(a).f(x_m) > 0$ o novo intervalo será $[x_m, b]$
6. Enquanto o erro $f(x_m) > \epsilon$ voltar ao passo 3

Veja a seguir a solução da mesma equação, ($e^x - \sin(x) - 2 = 0$) com os mesmos intervalos do método anterior.

```
Limite inferior: 1
Limite superior: 1.25
Tolerância: 0.001
-----a-----b-----xm-----fxm-----fa-----fb-----erro
1.0463 1.2500 1.0463 -0.0184 -0.1232 0.5414
1.0530 1.2500 1.0530 -0.0026 -0.0184 0.5414
1.0540 1.2500 1.0540 -0.0004 -0.0026 0.5414
Raiz: 1.053972856764054
Erro: -0.0003663936835978099
Eis o programa Python que resolve este caso
from numpy import *
def cordas():
    a=float(input("Limite inferior: "))
    b=float(input("Limite superior: "))
    tol=float(input("Tolerância: "))
    fxm=1
    print("-----a-----b-----xm-----fxm-----fa-----fb-----erro")
    while abs(fxm)>tol:
        fa=exp(a)-sin(a)-2
        fb=exp(b)-sin(b)-2
        xm=((a*fb)-(b*fa))/(fb-fa)
        fxm=exp(xm)-sin(xm)-2
        if fa*fxm>0:
            a=xm
        else:
            b=xm
        print('{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}'
              .format(a,b,xm,fxm,fa,fb))
    print("Raiz: ",xm)
    print("Erro: ",fxm)
cordas()
```

Método de Newton Este método começa com a interpretação geométrica da derivada. Lembrando, Newton é um monstro da ciência. Entre outras coisas, ele criou o cálculo diferencial e integral, a teoria da luz, a gravitação, as leis do movimento, além de, no fim da vida ter – como funcionário público – dirigido a casa da moeda inglesa, pendurando na forca alguns falsários delinquentes. Voltando ao cálculo diferencial, uma das interpretações da derivada de uma função em um ponto é a tangente da curva da função naquele ponto. Veja-se o que se disse neste desenho



- Eis as etapas do método:
1. Obter a derivada da função a resolver

2. Arbitrar um ponto inicial x_0 para a função
3. Arbitrar o erro ϵ
4. Calcular $f(x_0)$ e $df(x_0)$
5. Calcular o “novo” x , $x = x_0 - f(x)/df(x)$
6. Se $|f(x)| > \epsilon$ retornar ao passo 4

Por exemplo, seja calcular as raízes da mesma equação acima que é $e^x - \sin(x) - 2 = 0$. A derivada é $\frac{d}{dx} = e^x - \cos(x)$. Obteve-se a seguinte tabela

```
-----x-----fx-----dfx
1.0000000 -0.12318915634885141 2.1779795
1.0565612 0.00579321190120519 2.3845934
1.0541318 0.00001105001756940 2.3754999
1.0541271 0.0000000004045120 2.3754825
Raiz: 1.0541271241082415
Erro: 4.0451197946822504e-11
Eis o programa Python que resolve este caso
from numpy import *
def newton():
    x=1
    tol=0.00001
    fx=exp(x)-sin(x)-2
    dfx=exp(x)-cos(x)
    print("-----x-----fx-----dfx")
    print("{:12.7f}{:20.17f}{:12.7f}"
          .format(x,fx,dfx))
    while abs(fx)>tol:
        x=x-fx/dfx
        fx=exp(x)-sin(x)-2
        dfx=exp(x)-cos(x)
        print("{:12.7f}{:20.17f}{:12.7f}"
              .format(x,fx,dfx))
    print("Raiz: ",x)
    print("Erro: ",fx)
newton()
```

Para você fazer

1. Resolva pelos 3 métodos, para poder comparar, a seguinte equação $x = \sqrt[22]{9445}$ no intervalo entre 1 e 2 e responda a seguir os 3 valores encontrados

bissecção	
cordas	
Newton	

2. Idem $y = 4 \cdot \sin(x) + 5 \cdot \cos(x) - 3.141885 = 0$ no intervalo entre 5 e 6 e responda a seguir os 3 valores encontrados

bissecção	
cordas	
Newton	

3. Idem $y = ((8 \cdot x)/10) \cdot e^{((3 \cdot x)/10)} - 24.037691 = 0$ no intervalo entre 5 e 6 e responda a seguir os 3 valores encontrados

bissecção	
cordas	
Newton	

4. Idem $y = 5 \cdot x^4 + 3 \cdot x^2 - 75.990500 = 0$ no intervalo entre 1 e 2 e responda a seguir os 3 valores encontrados

bissecção	
cordas	
Newton	



Cálculo Numérico - solução de equações

Suponha que precisamos usar o computador para resolver equações. Já paramos para entender o que é resolver uma equação? Conhece-se a forma geral de uma equação $y = f(x)$. Resolver esta equação vem a ser descobrir quais os valores de x que fazem $f(x)$ ser igual a zero. Se olharmos para o gráfico, veremos que busca-se o valor da abscissa x que determina o cruzamento da curva de $f(x)$ no eixo dos x .

Por exemplo, suponha-se precisar descobrir a raiz 11 do número 1739? Ou seja, qual o número que multiplicado por ele mesmo 11 vezes dá como resultado 1739?

Dizendo isso na forma matemática, pode-se escrever $x^{11} = 1739$, ou melhor ainda $x^{11} - 1739 = 0$. Com isso, chegamos ao ponto.

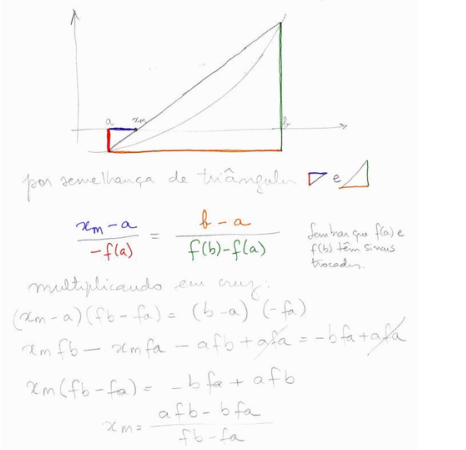
Método da Bissecção Se olharmos o gráfico de uma função, ver-se-á que no entorno da raiz, de um lado a função é positiva, e do outro lado é negativa ou vice versa. Este método começa pela seleção de 2 pontos x_a e x_b sendo que a função $y = f(x)$ têm sinais opostos nesses dois pontos. Graças a essa constatação, pode-se afirmar que a raiz procurada está entre x_a e x_b . Dizendo de outra maneira. O valor de x procurado, está entre a e b . O Método da bissecção vai dividindo o intervalo a, b pela metade, questionando a seguir em qual das metades o x está. Descoberto isso, despreza-se a metade que não contém x . Levando este processo ao limite, descobre-se qual o valor de x . Em resumo, eis as etapas do método:

1. Arbitrar intervalo em $[a, b]$ que compreenda a raiz.
2. Arbitrar um erro permitido ϵ
3. Calcular $x_m = (a + b)/2$
4. Se $f(a).f(x_m) < 0$ o novo intervalo será $[a, x_m]$
5. Se $f(a).f(x_m) > 0$ o novo intervalo será $[x_m, b]$
6. Enquanto o erro $b - a > \epsilon$ voltar ao passo 3.

Seja um exemplo, calcular $f = e^x - \sin(x) - 2 = 0$ nos limites $a = 1$ e $b = 1.25$ com $\epsilon = 0.01$. No primeira etapa tem-se

```
Limite Superior: 1.25
Limite Inferior: 1
Tolerancia: 0.01
-----a-----b-----xm-----fxm-----fa-----fb-----erro
1.1250 1.0000 1.1250 0.1779 0.5414 -0.1232 0.1250
1.0625 1.0000 1.0625 0.0200 0.1779 -0.1232 0.0625
1.0625 1.0312 1.0312 -0.0534 0.0200 -0.1232 0.0312
1.0625 1.0469 1.0469 -0.0171 0.0200 -0.0534 0.0156
1.0547 1.0469 1.0547 0.0013 0.0200 -0.0171 0.0078
Raiz: 1.05078125
Erro: -0.0078125
Eis o programa Python que resolve este caso
import numpy as np
def bissec():
    a=float(input("Limite Superior: "))
    b=float(input("Limite Inferior: "))
    tol=float(input("Tolerancia: "))
    print("-----a-----b-----xm-----fxm-----fa-----fb-----erro")
    while np.abs(b-a)>tol:
        xm=(a+b)/2
        fxm=np.exp(xm)-np.sin(xm)-2
        fa=np.exp(a)-np.sin(a)-2
        fb=np.exp(b)-np.sin(b)-2
        if fa*fxm>0:
            a=xm
        else:
            b=xm
        print('{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}'
              .format(a,b,xm,fxm,fa,fb,np.abs(b-a)))
    xm=(a+b)/2
    print("Raiz: ",xm)
    print("Erro: ",b-a)
bissec()
```

Cordas ou Falsa Posição



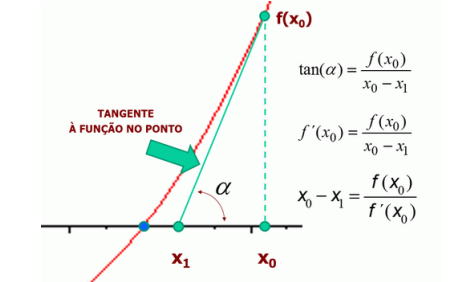
O método começa com uma análise do gráfico da função nas proximidades da raiz. Por semelhança de triângulos, pode-se obter a formulação proposta. Eis os passos do algoritmo:

1. Arbitrar um intervalo $[a, b]$ que compreende a raiz
2. Arbitrar ϵ
3. Calcular $x_m = \frac{a \cdot f(b) - b \cdot f(a)}{f(b) - f(a)}$
4. Se $f(a).f(x_m) < 0$ o novo intervalo será $[a, x_m]$
5. Se $f(a).f(x_m) > 0$ o novo intervalo será $[x_m, b]$
6. Enquanto o erro $f(x_m) > \epsilon$ voltar ao passo 3

Veja a seguir a solução da mesma equação, ($e^x - \sin(x) - 2 = 0$) com os mesmos intervalos do método anterior.

```
Limite inferior: 1
Limite superior: 1.25
Tolerância: 0.001
-----a-----b-----xm-----fxm-----fa-----fb-----erro
1.0463 1.2500 1.0463 -0.0184 -0.1232 0.5414
1.0530 1.2500 1.0530 -0.0026 -0.0184 0.5414
1.0540 1.2500 1.0540 -0.0004 -0.0026 0.5414
Raiz: 1.053972856764054
Erro: -0.0003663936835978099
Eis o programa Python que resolve este caso
from numpy import *
def cordas():
    a=float(input("Limite inferior: "))
    b=float(input("Limite superior: "))
    tol=float(input("Tolerância: "))
    fxm=1
    print("-----a-----b-----xm-----fxm-----fa-----fb-----erro")
    while abs(fxm)>tol:
        fa=exp(a)-sin(a)-2
        fb=exp(b)-sin(b)-2
        xm=((a*fb)-(b*fa))/(fb-fa)
        fxm=exp(xm)-sin(xm)-2
        if fa*fxm>0:
            a=xm
        else:
            b=xm
        print('{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}'
              .format(a,b,xm,fxm,fa,fb))
    print("Raiz: ",xm)
    print("Erro: ",fxm)
cordas()
```

Método de Newton Este método começa com a interpretação geométrica da derivada. Lembrando, Newton é um monstro da ciência. Entre outras coisas, ele criou o cálculo diferencial e integral, a teoria da luz, a gravitação, as leis do movimento, além de, no fim da vida ter – como funcionário público – dirigido a casa da moeda inglesa, pendurando na forca alguns falsários delinquentes. Voltando ao cálculo diferencial, uma das interpretações da derivada de uma função em um ponto é a tangente da curva da função naquele ponto. Veja-se o que se disse neste desenho



Eis as etapas do método:

1. Obter a derivada da função a resolver

2. Arbitrar um ponto inicial x_0 para a função
3. Arbitrar o erro ϵ
4. Calcular $f(x_0)$ e $df(x_0)$
5. Calcular o “novo” x , $x = x_0 - f(x)/df(x)$
6. Se $|f(x)| > \epsilon$ retornar ao passo 4

Por exemplo, seja calcular as raízes da mesma equação acima que é $e^x - \sin(x) - 2 = 0$. A derivada é $\frac{d}{dx} = e^x - \cos(x)$. Obteve-se a seguinte tabela

```
-----x-----fx-----dfx
1.0000000 -0.12318915634885141 2.1779795
1.0565612 0.00579321190120519 2.3845934
1.0541318 0.00001105001756940 2.3754999
1.0541271 0.00000000004045120 2.3754825
Raiz: 1.0541271241082415
Erro: 4.0451197946822504e-11
Eis o programa Python que resolve este caso
from numpy import *
def newton1():
    x=1
    tol=0.00001
    fx=exp(x)-sin(x)-2
    dfx=exp(x)-cos(x)
    print("-----x-----fx-----dfx")
    print("{:12.7f}{:20.17f}{:12.7f}".format(x,fx,dfx))
    while abs(fx)>tol:
        x=x-fx/dfx
        fx=exp(x)-sin(x)-2
        dfx=exp(x)-cos(x)
        print("{:12.7f}{:20.17f}{:12.7f}".format(x,fx,dfx))
    print("Raiz: ",x)
    print("Erro: ",fx)
newton1()
```

Para você fazer

1. Resolva pelos 3 métodos, para poder comparar, a seguinte equação $x = \sqrt[2]{9280}$ no intervalo entre 1 e 2 e responda a seguir os 3 valores encontrados

bissecção	
cordas	
Newton	

2. Idem $y = 8 \cdot \sin(x) + 5 \cdot \cos(x) + 9.149481 = 0$ no intervalo entre 4 e 5 e responda a seguir os 3 valores encontrados

bissecção	
cordas	
Newton	

3. Idem $y = ((6 \cdot x)/10) \cdot e^{((7 \cdot x)/10)} - 16.290408 = 0$ no intervalo entre 3 e 4 e responda a seguir os 3 valores encontrados

bissecção	
cordas	
Newton	

4. Idem $y = 7 \cdot x^6 + 5 \cdot x^4 - 60176.671875 = 0$ no intervalo entre 4 e 5 e responda a seguir os 3 valores encontrados

bissecção	
cordas	
Newton	



Cálculo Numérico - solução de equações

Suponha que precisamos usar o computador para resolver equações. Já paramos para entender o que é resolver uma equação? Conhece-se a forma geral de uma equação $y = f(x)$. Resolver esta equação vem a ser descobrir quais os valores de x que fazem $f(x)$ ser igual a zero. Se olharmos para o gráfico, veremos que busca-se o valor da abscissa x que determina o cruzamento da curva de $f(x)$ no eixo dos x .

Por exemplo, suponha-se precisar descobrir a raiz 11 do número 1739? Ou seja, qual o número que multiplicado por ele mesmo 11 vezes dá como resultado 1739?

Dizendo isso na forma matemática, pode-se escrever $x^{11} = 1739$, ou melhor ainda $x^{11} - 1739 = 0$. Com isso, chegamos ao ponto.

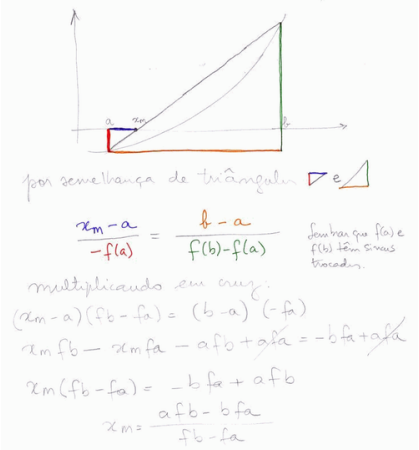
Método da Bissecção Se olharmos o gráfico de uma função, ver-se-á que no entorno da raiz, de um lado a função é positiva, e do outro lado é negativa ou vice versa. Este método começa pela seleção de 2 pontos x_a e x_b sendo que a função $y = f(x)$ têm sinais opostos nesses dois pontos. Graças a essa constatação, pode-se afirmar que a raiz procurada está entre x_a e x_b . Dizendo de outra maneira. O valor de x procurado, está entre a e b . O Método da bissecção vai dividindo o intervalo a, b pela metade, questionando a seguir em qual das metades o x está. Descoberto isso, despreza-se a metade que não contém x . Levando este processo ao limite, descobre-se qual o valor de x . Em resumo, eis as etapas do método:

1. Arbitrar intervalo em $[a, b]$ que compreenda a raiz.
2. Arbitrar um erro permitido ϵ
3. Calcular $x_m = (a + b)/2$
4. Se $f(a).f(x_m) < 0$ o novo intervalo será $[a, x_m]$
5. Se $f(a).f(x_m) > 0$ o novo intervalo será $[x_m, b]$
6. Enquanto o erro $b - a > \epsilon$ voltar ao passo 3.

Seja um exemplo, calcular $f = e^x - \sin(x) - 2 = 0$ nos limites $a = 1$ e $b = 1.25$ com $\epsilon = 0.01$. No primeira etapa tem-se

```
Limite Superior: 1.25
Limite Inferior: 1
Tolerancia: 0.01
-----a-----b-----xm-----fxm-----fa-----fb-----erro
1.1250 1.0000 1.1250 0.1779 0.5414 -0.1232 0.1250
1.0625 1.0000 1.0625 0.0200 0.1779 -0.1232 0.0625
1.0625 1.0312 1.0312 -0.0534 0.0200 -0.1232 0.0312
1.0625 1.0469 1.0469 -0.0171 0.0200 -0.0534 0.0156
1.0547 1.0469 1.0547 0.0013 0.0200 -0.0171 0.0078
Raiz: 1.05078125
Erro: -0.0078125
Eis o programa Python que resolve este caso
import numpy as np
def bissec():
    a=float(input("Limite Superior: "))
    b=float(input("Limite Inferior: "))
    tol=float(input("Tolerancia: "))
    print("-----a-----b-----xm-----fxm-----fa-----fb-----erro")
    while np.abs(b-a)>tol:
        xm=(a+b)/2
        fxm=np.exp(xm)-np.sin(xm)-2
        fa=np.exp(a)-np.sin(a)-2
        fb=np.exp(b)-np.sin(b)-2
        if fa*fxm>0:
            a=xm
        else:
            b=xm
        print('{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}'
              .format(a,b,xm,fxm,fa,fb,np.abs(b-a)))
    xm=(a+b)/2
    print("Raiz: ",xm)
    print("Erro: ",b-a)
bissec()
```

Cordas ou Falsa Posição



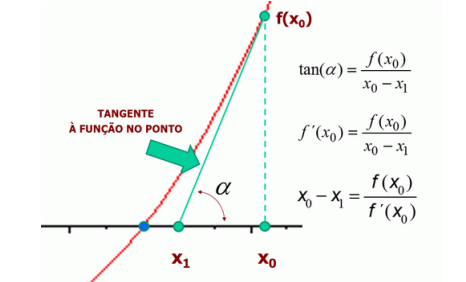
O método começa com uma análise do gráfico da função nas proximidades da raiz. Por semelhança de triângulos, pode-se obter a formulação proposta. Eis os passos do algoritmo:

1. Arbitrar um intervalo $[a, b]$ que compreende a raiz
2. Arbitrar ϵ
3. Calcular $x_m = \frac{a \cdot f(b) - b \cdot f(a)}{f(b) - f(a)}$
4. Se $f(a).f(x_m) < 0$ o novo intervalo será $[a, x_m]$
5. Se $f(a).f(x_m) > 0$ o novo intervalo será $[x_m, b]$
6. Enquanto o erro $f(x_m) > \epsilon$ voltar ao passo 3

Veja a seguir a solução da mesma equação, ($e^x - \sin(x) - 2 = 0$) com os mesmos intervalos do método anterior.

```
Limite inferior: 1
Limite superior: 1.25
Tolerância: 0.001
-----a-----b-----xm-----fxm-----fa-----fb-----erro
1.0463 1.2500 1.0463 -0.0184 -0.1232 0.5414
1.0530 1.2500 1.0530 -0.0026 -0.0184 0.5414
1.0540 1.2500 1.0540 -0.0004 -0.0026 0.5414
Raiz: 1.053972856764054
Erro: -0.0003663936835978099
Eis o programa Python que resolve este caso
from numpy import *
def cordas():
    a=float(input("Limite inferior: "))
    b=float(input("Limite superior: "))
    tol=float(input("Tolerância: "))
    fxm=1
    print("-----a-----b-----xm-----fxm-----fa-----fb-----erro")
    while abs(fxm)>tol:
        fa=exp(a)-sin(a)-2
        fb=exp(b)-sin(b)-2
        fxm=((a*fb)-(b*fa))/(fb-fa)
        fxm=exp(xm)-sin(xm)-2
        if fa*fxm>0:
            a=fxm
        else:
            b=fxm
        print('{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}'
              .format(a,b,xm,fxm,fa,fb))
    print("Raiz: ",xm)
    print("Erro: ",fxm)
cordas()
```

Método de Newton Este método começa com a interpretação geométrica da derivada. Lembrando, Newton é um monstro da ciência. Entre outras coisas, ele criou o cálculo diferencial e integral, a teoria da luz, a gravitação, as leis do movimento, além de, no fim da vida ter – como funcionário público – dirigido a casa da moeda inglesa, pendurando na forca alguns falsários delinquentes. Voltando ao cálculo diferencial, uma das interpretações da derivada de uma função em um ponto é a tangente da curva da função naquele ponto. Veja-se o que se disse neste desenho



Eis as etapas do método:

1. Obter a derivada da função a resolver

2. Arbitrar um ponto inicial x_0 para a função
3. Arbitrar o erro ϵ
4. Calcular $f(x_0)$ e $df(x_0)$
5. Calcular o “novo” x , $x = x_0 - f(x)/df(x)$
6. Se $|f(x)| > \epsilon$ retornar ao passo 4

Por exemplo, seja calcular as raízes da mesma equação acima que é $e^x - \sin(x) - 2 = 0$. A derivada é $\frac{d}{dx} = e^x - \cos(x)$. Obteve-se a seguinte tabela

```
-----x-----fx-----dfx
1.0000000 -0.12318915634885141 2.1779795
1.0565612 0.00579321190120519 2.3845934
1.0541318 0.00001105001756940 2.3754999
1.0541271 0.00000000004045120 2.3754825
Raiz: 1.0541271241082415
Erro: 4.0451197946822504e-11
Eis o programa Python que resolve este caso
from numpy import *
def newton():
    x=1
    tol=0.00001
    fx=exp(x)-sin(x)-2
    dfx=exp(x)-cos(x)
    print("-----x-----fx-----dfx")
    print("{:12.7f}{:20.17f}{:12.7f}".format(x,fx,dfx))
    while abs(fx)>tol:
        x=x-fx/dfx
        fx=exp(x)-sin(x)-2
        dfx=exp(x)-cos(x)
        print("{:12.7f}{:20.17f}{:12.7f}".format(x,fx,dfx))
    print("Raiz: ",x)
    print("Erro: ",fx)
newton()
```

Para você fazer

1. Resolva pelos 3 métodos, para poder comparar, a seguinte equação $x = \sqrt[20]{9326}$ no intervalo entre 1 e 2 e responda a seguir os 3 valores encontrados

bissecção	
cordas	
Newton	

2. Idem $y = 6 * \sin(x) + 4 * \cos(x) - 7.161629 = 0$ no intervalo entre 1 e 2 e responda a seguir os 3 valores encontrados

bissecção	
cordas	
Newton	

3. Idem $y = ((6 * x)/10) * e^{((4 * x)/10)} - 15.344835 = 0$ no intervalo entre 4 e 5 e responda a seguir os 3 valores encontrados

bissecção	
cordas	
Newton	

4. Idem $y = 7 * x^6 + 4 * x^2 - 50871.636992 = 0$ no intervalo entre 4 e 5 e responda a seguir os 3 valores encontrados

bissecção	
cordas	
Newton	



Cálculo Numérico - solução de equações

Suponha que precisamos usar o computador para resolver equações. Já paramos para entender o que é resolver uma equação? Conhece-se a forma geral de uma equação $y = f(x)$. Resolver esta equação vem a ser descobrir quais os valores de x que fazem $f(x)$ ser igual a zero. Se olharmos para o gráfico, veremos que busca-se o valor da abscissa x que determina o cruzamento da curva de $f(x)$ no eixo dos x .

Por exemplo, suponha-se precisar descobrir a raiz 11 do número 1739? Ou seja, qual o número que multiplicado por ele mesmo 11 vezes dá como resultado 1739?

Dizendo isso na forma matemática, pode-se escrever $x^{11} = 1739$, ou melhor ainda $x^{11} - 1739 = 0$. Com isso, chegamos ao ponto.

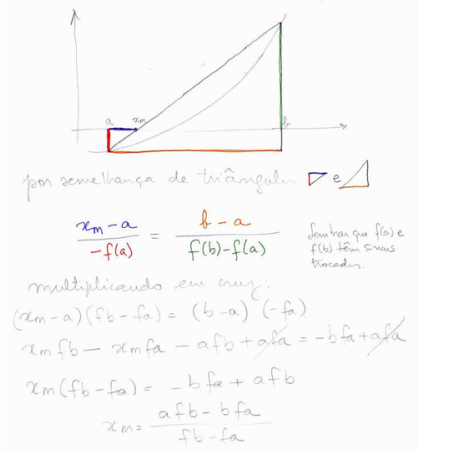
Método da Bissecção Se olharmos o gráfico de uma função, ver-se-á que no entorno da raiz, de um lado a função é positiva, e do outro lado é negativa ou vice versa. Este método começa pela seleção de 2 pontos x_a e x_b sendo que a função $y = f(x)$ têm sinais opostos nesses dois pontos. Graças a essa constatação, pode-se afirmar que a raiz procurada está entre x_a e x_b . Dizendo de outra maneira. O valor de x procurado, está entre a e b . O Método da bissecção vai dividindo o intervalo a, b pela metade, questionando a seguir em qual das metades o x está. Descoberto isso, despreza-se a metade que não contém x . Levando este processo ao limite, descobre-se qual o valor de x . Em resumo, eis as etapas do método:

1. Arbitrar intervalo em $[a, b]$ que compreenda a raiz.
2. Arbitrar um erro permitido ϵ
3. Calcular $x_m = (a + b)/2$
4. Se $f(a).f(x_m) < 0$ o novo intervalo será $[a, x_m]$
5. Se $f(a).f(x_m) > 0$ o novo intervalo será $[x_m, b]$
6. Enquanto o erro $b - a > \epsilon$ voltar ao passo 3.

Seja um exemplo, calcular $f = e^x - \sin(x) - 2 = 0$ nos limites $a = 1$ e $b = 1.25$ com $\epsilon = 0.01$. No primeira etapa tem-se

```
Limite Superior: 1.25
Limite Inferior: 1
Tolerancia: 0.01
-----a-----b-----xm-----fxm-----fa-----fb-----erro
1.1250 1.0000 1.1250 0.1779 0.5414 -0.1232 0.1250
1.0625 1.0000 1.0625 0.0200 0.1779 -0.1232 0.0625
1.0625 1.0312 1.0312 -0.0534 0.0200 -0.1232 0.0312
1.0625 1.0469 1.0469 -0.0171 0.0200 -0.0534 0.0156
1.0547 1.0469 1.0547 0.0013 0.0200 -0.0171 0.0078
Raiz: 1.05078125
Erro: -0.0078125
Eis o programa Python que resolve este caso
import numpy as np
def bissec():
    a=float(input("Limite Superior: "))
    b=float(input("Limite Inferior: "))
    tol=float(input("Tolerancia: "))
    print("-----a-----b-----xm-----fxm-----fa-----fb-----erro")
    while np.abs(b-a)>tol:
        xm=(a+b)/2
        fxm=np.exp(xm)-np.sin(xm)-2
        fa=np.exp(a)-np.sin(a)-2
        fb=np.exp(b)-np.sin(b)-2
        if fa*fxm>0:
            a=xm
        else:
            b=xm
        print('{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}'.format(a,b,xm,fxm,fa,fb,np.abs(b-a)))
    xm=(a+b)/2
    print("Raiz: ",xm)
    print("Erro: ",b-a)
bissec()
```

Cordas ou Falsa Posição



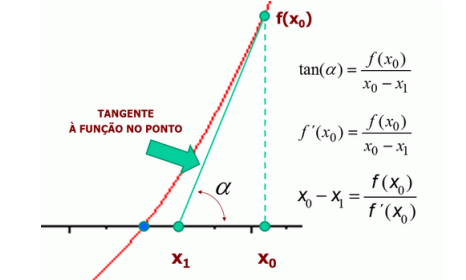
O método começa com uma análise do gráfico da função nas proximidades da raiz. Por semelhança de triângulos, pode-se obter a formulação proposta. Eis os passos do algoritmo:

1. Arbitrar um intervalo $[a, b]$ que compreende a raiz
2. Arbitrar ϵ
3. Calcular $x_m = \frac{a \cdot f(b) - b \cdot f(a)}{f(b) - f(a)}$
4. Se $f(a).f(x_m) < 0$ o novo intervalo será $[a, x_m]$
5. Se $f(a).f(x_m) > 0$ o novo intervalo será $[x_m, b]$
6. Enquanto o erro $f(x_m) > \epsilon$ voltar ao passo 3

Veja a seguir a solução da mesma equação, ($e^x - \sin(x) - 2 = 0$) com os mesmos intervalos do método anterior.

```
Limite inferior: 1
Limite superior: 1.25
Tolerância: 0.001
-----a-----b-----xm-----fxm-----fa-----fb-----erro
1.0463 1.2500 1.0463 -0.0184 -0.1232 0.5414
1.0530 1.2500 1.0530 -0.0026 -0.0184 0.5414
1.0540 1.2500 1.0540 -0.0004 -0.0026 0.5414
Raiz: 1.053972856764054
Erro: -0.0003663936835978099
Eis o programa Python que resolve este caso
from numpy import *
def cordas():
    a=float(input("Limite inferior: "))
    b=float(input("Limite superior: "))
    tol=float(input("Tolerância: "))
    fxm=1
    print("-----a-----b-----xm-----fxm-----fa-----fb-----erro")
    while abs(fxm)>tol:
        fa=exp(a)-sin(a)-2
        fb=exp(b)-sin(b)-2
        fxm=((a*fb)-(b*fa))/(fb-fa)
        fxm=exp(xm)-sin(xm)-2
        if fa*fxm>0:
            a=fxm
        else:
            b=fxm
        print('{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}'.format(a,b,xm,fxm,fa,fb))
    print("Raiz: ",xm)
    print("Erro: ",fxm)
cordas()
```

Método de Newton Este método começa com a interpretação geométrica da derivada. Lembrando, Newton é um monstro da ciência. Entre outras coisas, ele criou o cálculo diferencial e integral, a teoria da luz, a gravitação, as leis do movimento, além de, no fim da vida ter – como funcionário público – dirigido a casa da moeda inglesa, pendurando na forca alguns falsários delinquentes. Voltando ao cálculo diferencial, uma das interpretações da derivada de uma função em um ponto é a tangente da curva da função naquele ponto. Veja-se o que se disse neste desenho



- Eis as etapas do método:
1. Obter a derivada da função a resolver

2. Arbitrar um ponto inicial x_0 para a função
3. Arbitrar o erro ϵ
4. Calcular $f(x_0)$ e $df(x_0)$
5. Calcular o “novo” x , $x = x_0 - f(x)/df(x)$
6. Se $|f(x)| > \epsilon$ retornar ao passo 4

Por exemplo, seja calcular as raízes da mesma equação acima que é $e^x - \sin(x) - 2 = 0$. A derivada é $\frac{d}{dx} = e^x - \cos(x)$. Obteve-se a seguinte tabela

```
-----x-----fx-----dfx
1.0000000 -0.12318915634885141 2.1779795
1.0565612 0.00579321190120519 2.3845934
1.0541318 0.00001105001756940 2.3754999
1.0541271 0.0000000004045120 2.3754825
Raiz: 1.0541271241082415
Erro: 4.0451197946822504e-11
Eis o programa Python que resolve este caso
from numpy import *
def newton1():
    x=1
    tol=0.00001
    fx=exp(x)-sin(x)-2
    dfx=exp(x)-cos(x)
    print("-----x-----fx-----dfx")
    print("{:12.7f}{:20.17f}{:12.7f}".format(x,fx,dfx))
    while abs(fx)>tol:
        x=x-fx/dfx
        fx=exp(x)-sin(x)-2
        dfx=exp(x)-cos(x)
        print("{:12.7f}{:20.17f}{:12.7f}".format(x,fx,dfx))
    print("Raiz: ",x)
    print("Erro: ",fx)
newton1()
```

Para você fazer

1. Resolva pelos 3 métodos, para poder comparar, a seguinte equação $x = \sqrt[6]{7487}$ no intervalo entre 4 e 5 e responda a seguir os 3 valores encontrados

bissecção	
cordas	
Newton	

2. Idem $y = 8 * \sin(x) + 6 * \cos(x) - 6.297854 = 0$ no intervalo entre 8 e 9 e responda a seguir os 3 valores encontrados

bissecção	
cordas	
Newton	

3. Idem $y = ((5 * x)/10) * e^{((8 * x)/10)} - 10.405810 = 0$ no intervalo entre 2 e 3 e responda a seguir os 3 valores encontrados

bissecção	
cordas	
Newton	

4. Idem $y = 6 * x^5 + 4 * x^3 - 1120.430080 = 0$ no intervalo entre 2 e 3 e responda a seguir os 3 valores encontrados

bissecção	
cordas	
Newton	

Cálculo Numérico - solução de equações

Suponha que precisamos usar o computador para resolver equações. Já paramos para entender o que é resolver uma equação? Conhece-se a forma geral de uma equação $y = f(x)$. Resolver esta equação vem a ser descobrir quais os valores de x que fazem $f(x)$ ser igual a zero. Se olharmos para o gráfico, veremos que busca-se o valor da abscissa x que determina o cruzamento da curva de $f(x)$ no eixo dos x .

Por exemplo, suponha-se precisar descobrir a raiz 11 do número 1739? Ou seja, qual o número que multiplicado por ele mesmo 11 vezes dá como resultado 1739?

Dizendo isso na forma matemática, pode-se escrever $x^{11} = 1739$, ou melhor ainda $x^{11} - 1739 = 0$. Com isso, chegamos ao ponto.

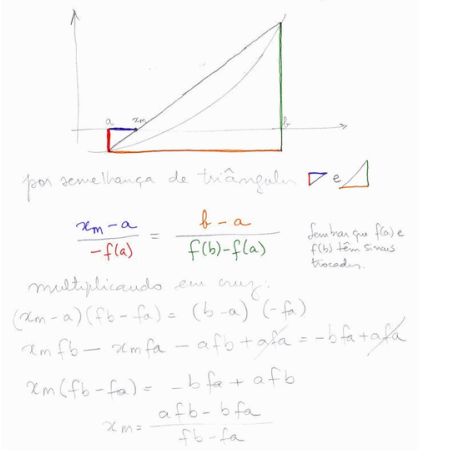
Método da Bissecção Se olharmos o gráfico de uma função, ver-se-á que no entorno da raiz, de um lado a função é positiva, e do outro lado é negativa ou vice versa. Este método começa pela seleção de 2 pontos x_a e x_b sendo que a função $y = f(x)$ têm sinais opostos nesses dois pontos. Graças a essa constatação, pode-se afirmar que a raiz procurada está entre x_a e x_b . Dizendo de outra maneira. O valor de x procurado, está entre a e b . O Método da bissecção vai dividindo o intervalo a, b pela metade, questionando a seguir em qual das metades o x está. Descoberto isso, despreza-se a metade que não contém x . Levando este processo ao limite, descobre-se qual o valor de x . Em resumo, eis as etapas do método:

1. Arbitrar intervalo em $[a, b]$ que compreenda a raiz.
2. Arbitrar um erro permitido ϵ
3. Calcular $x_m = (a + b)/2$
4. Se $f(a).f(x_m) < 0$ o novo intervalo será $[a, x_m]$
5. Se $f(a).f(x_m) > 0$ o novo intervalo será $[x_m, b]$
6. Enquanto o erro $b - a > \epsilon$ voltar ao passo 3.

Seja um exemplo, calcular $f = e^x - \sin(x) - 2 = 0$ nos limites $a = 1$ e $b = 1.25$ com $\epsilon = 0.01$. No primeira etapa tem-se

```
Limite Superior: 1.25
Limite Inferior: 1
Tolerancia: 0.01
-----a-----b-----xm-----fxm-----fa-----fb-----erro
1.1250 1.0000 1.1250 0.1779 0.5414 -0.1232 0.1250
1.0625 1.0000 1.0625 0.0200 0.1779 -0.1232 0.0625
1.0625 1.0312 1.0312 -0.0534 0.0200 -0.1232 0.0312
1.0625 1.0469 1.0469 -0.0171 0.0200 -0.0534 0.0156
1.0547 1.0469 1.0547 0.0013 0.0200 -0.0171 0.0078
Raiz: 1.05078125
Erro: -0.0078125
Eis o programa Python que resolve este caso
import numpy as np
def bissec():
    a=float(input("Limite Superior: "))
    b=float(input("Limite Inferior: "))
    tol=float(input("Tolerancia: "))
    print("-----a-----b-----xm-----fxm-----fa-----fb-----erro")
    while np.abs(b-a)>tol:
        xm=(a+b)/2
        fxm=np.exp(xm)-np.sin(xm)-2
        fa=np.exp(a)-np.sin(a)-2
        fb=np.exp(b)-np.sin(b)-2
        if fa*fxm>0:
            a=xm
        else:
            b=xm
        print('{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}'.format(a,b,xm,fxm,fa,fb,np.abs(b-a)))
    xm=(a+b)/2
    print("Raiz: ",xm)
    print("Erro: ",b-a)
bissec()
```

Cordas ou Falsa Posição



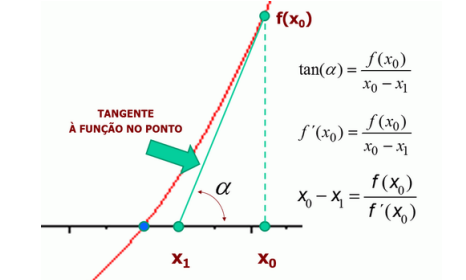
O método começa com uma análise do gráfico da função nas proximidades da raiz. Por semelhança de triângulos, pode-se obter a formulação proposta. Eis os passos do algoritmo:

1. Arbitrar um intervalo $[a, b]$ que compreende a raiz
2. Arbitrar ϵ
3. Calcular $x_m = \frac{a.f(b)-b.f(a)}{f(b)-f(a)}$
4. Se $f(a).f(x_m) < 0$ o novo intervalo será $[a, x_m]$
5. Se $f(a).f(x_m) > 0$ o novo intervalo será $[x_m, b]$
6. Enquanto o erro $f(x_m) > \epsilon$ voltar ao passo 3

Veja a seguir a solução da mesma equação, ($e^x - \sin(x) - 2 = 0$) com os mesmos intervalos do método anterior.

```
Limite inferior: 1
Limite superior: 1.25
Tolerância: 0.001
-----a-----b-----xm-----fxm-----fa-----fb-----erro
1.0463 1.2500 1.0463 -0.0184 -0.1232 0.5414
1.0530 1.2500 1.0530 -0.0026 -0.0184 0.5414
1.0540 1.2500 1.0540 -0.0004 -0.0026 0.5414
Raiz: 1.053972856764054
Erro: -0.0003663936835978099
Eis o programa Python que resolve este caso
from numpy import *
def cordas():
    a=float(input("Limite inferior: "))
    b=float(input("Limite superior: "))
    tol=float(input("Tolerância: "))
    fxm=1
    print("-----a-----b-----xm-----fxm-----fa-----fb-----erro")
    while abs(fxm)>tol:
        fa=exp(a)-sin(a)-2
        fb=exp(b)-sin(b)-2
        xm=((a*fb)-(b*fa))/(fb-fa)
        fxm=exp(xm)-sin(xm)-2
        if fa*fxm>0:
            a=xm
        else:
            b=xm
        print('{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}'.format(a,b,xm,fxm,fa,fb))
    print("Raiz: ",xm)
    print("Erro: ",fxm)
cordas()
```

Método de Newton Este método começa com a interpretação geométrica da derivada. Lembrando, Newton é um monstro da ciência. Entre outras coisas, ele criou o cálculo diferencial e integral, a teoria da luz, a gravitação, as leis do movimento, além de, no fim da vida ter – como funcionário público – dirigido a casa da moeda inglesa, pendurando na forca alguns falsários delinquentes. Voltando ao cálculo diferencial, uma das interpretações da derivada de uma função em um ponto é a tangente da curva da função naquele ponto. Veja-se o que se disse neste desenho



Eis as etapas do método:

1. Obter a derivada da função a resolver

2. Arbitrar um ponto inicial x_0 para a função
3. Arbitrar o erro ϵ
4. Calcular $f(x_0)$ e $df(x_0)$
5. Calcular o “novo” x , $x = x_0 - f(x)/df(x)$
6. Se $|f(x)| > \epsilon$ retornar ao passo 4

Por exemplo, seja calcular as raízes da mesma equação acima que é $e^x - \sin(x) - 2 = 0$. A derivada é $\frac{d}{dx} = e^x - \cos(x)$. Obteve-se a seguinte tabela

```
-----x-----fx-----dfx
1.0000000 -0.12318915634885141 2.1779795
1.0565612 0.00579321190120519 2.3845934
1.0541318 0.00001105001756940 2.3754999
1.0541271 0.00000000004045120 2.3754825
Raiz: 1.0541271241082415
Erro: 4.0451197946822504e-11
Eis o programa Python que resolve este caso
from numpy import *
def newton1():
    x=1
    tol=0.00001
    fx=exp(x)-sin(x)-2
    dfx=exp(x)-cos(x)
    print("-----x-----fx-----dfx")
    print("{:12.7f}{:20.17f}{:12.7f}".format(x,fx,dfx))
    while abs(fx)>tol:
        x=x-fx/dfx
        fx=exp(x)-sin(x)-2
        dfx=exp(x)-cos(x)
        print("{:12.7f}{:20.17f}{:12.7f}".format(x,fx,dfx))
    print("Raiz: ",x)
    print("Erro: ",fx)
newton1()
```

Para você fazer

1. Resolva pelos 3 métodos, para poder comparar, a seguinte equação $x = \sqrt[14]{8740}$ no intervalo entre 1 e 2 e responda a seguir os 3 valores encontrados

bissecção	
cordas	
Newton	

2. Idem $y = 3*\sin(x) + 7*\cos(x) + 4.554984 = 0$ no intervalo entre 8 e 9 e responda a seguir os 3 valores encontrados

bissecção	
cordas	
Newton	

3. Idem $y = ((6 * x)/10) * e^{((2*x)/10)} - 29.740133 = 0$ no intervalo entre 8 e 9 e responda a seguir os 3 valores encontrados

bissecção	
cordas	
Newton	

4. Idem $y = 6 * x^5 + 4 * x^2 - 4811.870080 = 0$ no intervalo entre 3 e 4 e responda a seguir os 3 valores encontrados

bissecção	
cordas	
Newton	



Cálculo Numérico - solução de equações

Suponha que precisamos usar o computador para resolver equações. Já paramos para entender o que é resolver uma equação? Conhece-se a forma geral de uma equação $y = f(x)$. Resolver esta equação vem a ser descobrir quais os valores de x que fazem $f(x)$ ser igual a zero. Se olharmos para o gráfico, veremos que busca-se o valor da abscissa x que determina o cruzamento da curva de $f(x)$ no eixo dos x .

Por exemplo, suponha-se precisar descobrir a raiz 11 do número 1739? Ou seja, qual o número que multiplicado por ele mesmo 11 vezes dá como resultado 1739?

Dizendo isso na forma matemática, pode-se escrever $x^{11} = 1739$, ou melhor ainda $x^{11} - 1739 = 0$. Com isso, chegamos ao ponto.

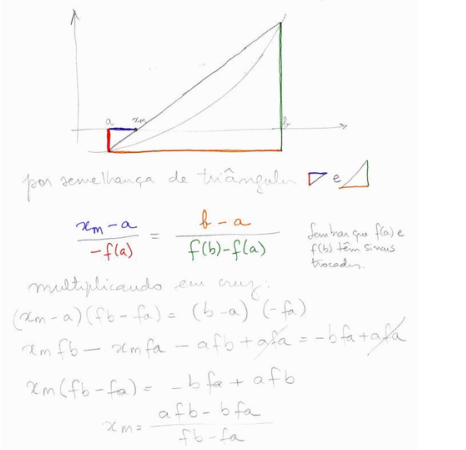
Método da Bissecção Se olharmos o gráfico de uma função, ver-se-á que no entorno da raiz, de um lado a função é positiva, e do outro lado é negativa ou vice versa. Este método começa pela seleção de 2 pontos x_a e x_b sendo que a função $y = f(x)$ têm sinais opostos nesses dois pontos. Graças a essa constatação, pode-se afirmar que a raiz procurada está entre x_a e x_b . Dizendo de outra maneira. O valor de x procurado, está entre a e b . O Método da bissecção vai dividindo o intervalo a, b pela metade, questionando a seguir em qual das metades o x está. Descoberto isso, despreza-se a metade que não contém x . Levando este processo ao limite, descobre-se qual o valor de x . Em resumo, eis as etapas do método:

1. Arbitrar intervalo em $[a, b]$ que compreenda a raiz.
2. Arbitrar um erro permitido ϵ
3. Calcular $x_m = (a + b)/2$
4. Se $f(a).f(x_m) < 0$ o novo intervalo será $[a, x_m]$
5. Se $f(a).f(x_m) > 0$ o novo intervalo será $[x_m, b]$
6. Enquanto o erro $b - a > \epsilon$ voltar ao passo 3.

Seja um exemplo, calcular $f = e^x - \sin(x) - 2 = 0$ nos limites $a = 1$ e $b = 1.25$ com $\epsilon = 0.01$. No primeira etapa tem-se

```
Limite Superior: 1.25
Limite Inferior: 1
Tolerancia: 0.01
-----a-----b-----xm-----fxm-----fa-----fb-----erro
1.1250 1.0000 1.1250 0.1779 0.5414 -0.1232 0.1250
1.0625 1.0000 1.0625 0.0200 0.1779 -0.1232 0.0625
1.0625 1.0312 1.0312 -0.0534 0.0200 -0.1232 0.0312
1.0625 1.0469 1.0469 -0.0171 0.0200 -0.0534 0.0156
1.0547 1.0469 1.0547 0.0013 0.0200 -0.0171 0.0078
Raiz: 1.05078125
Erro: -0.0078125
Eis o programa Python que resolve este caso
import numpy as np
def bissec():
    a=float(input("Limite Superior: "))
    b=float(input("Limite Inferior: "))
    tol=float(input("Tolerancia: "))
    print("-----a-----b-----xm-----fxm-----fa-----fb-----erro")
    while np.abs(b-a)>tol:
        xm=(a+b)/2
        fxm=np.exp(xm)-np.sin(xm)-2
        fa=np.exp(a)-np.sin(a)-2
        fb=np.exp(b)-np.sin(b)-2
        if fa*fxm>0:
            a=xm
        else:
            b=xm
        print('{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}'
              .format(a,b,xm,fxm,fa,fb,np.abs(b-a)))
    xm=(a+b)/2
    print("Raiz: ",xm)
    print("Erro: ",b-a)
bissec()
```

Cordas ou Falsa Posição



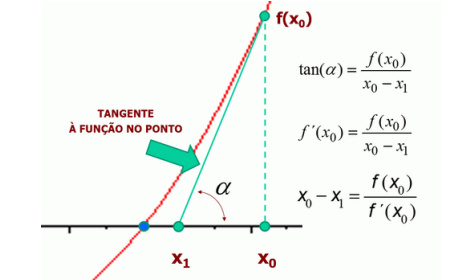
O método começa com uma análise do gráfico da função nas proximidades da raiz. Por semelhança de triângulos, pode-se obter a formulação proposta. Eis os passos do algoritmo:

1. Arbitrar um intervalo $[a, b]$ que compreende a raiz
2. Arbitrar ϵ
3. Calcular $x_m = \frac{a \cdot f(b) - b \cdot f(a)}{f(b) - f(a)}$
4. Se $f(a).f(x_m) < 0$ o novo intervalo será $[a, x_m]$
5. Se $f(a).f(x_m) > 0$ o novo intervalo será $[x_m, b]$
6. Enquanto o erro $f(x_m) > \epsilon$ voltar ao passo 3

Veja a seguir a solução da mesma equação, ($e^x - \sin(x) - 2 = 0$) com os mesmos intervalos do método anterior.

```
Limite inferior: 1
Limite superior: 1.25
Tolerância: 0.001
-----a-----b-----xm-----fxm-----fa-----fb-----erro
1.0463 1.2500 1.0463 -0.0184 -0.1232 0.5414
1.0530 1.2500 1.0530 -0.0026 -0.0184 0.5414
1.0540 1.2500 1.0540 -0.0004 -0.0026 0.5414
Raiz: 1.0539728556764054
Erro: -0.0003663936835978099
Eis o programa Python que resolve este caso
from numpy import *
def cordas():
    a=float(input("Limite inferior: "))
    b=float(input("Limite superior: "))
    tol=float(input("Tolerância: "))
    fxm=1
    print("-----a-----b-----xm-----fxm-----fa-----fb-----erro")
    while abs(fxm)>tol:
        fa=exp(a)-sin(a)-2
        fb=exp(b)-sin(b)-2
        xm=((a*fb)-(b*fa))/(fb-fa)
        fxm=exp(xm)-sin(xm)-2
        if fa*fxm>0:
            a=xm
        else:
            b=xm
        print('{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}'
              .format(a,b,xm,fxm,fa,fb))
    print("Raiz: ",xm)
    print("Erro: ",fxm)
cordas()
```

Método de Newton Este método começa com a interpretação geométrica da derivada. Lembrando, Newton é um monstro da ciência. Entre outras coisas, ele criou o cálculo diferencial e integral, a teoria da luz, a gravitação, as leis do movimento, além de, no fim da vida ter – como funcionário público – dirigido a casa da moeda inglesa, pendurando na forca alguns falsários delinquentes. Voltando ao cálculo diferencial, uma das interpretações da derivada de uma função em um ponto é a tangente da curva da função naquele ponto. Veja-se o que se disse neste desenho



- Eis as etapas do método:
1. Obter a derivada da função a resolver

2. Arbitrar um ponto inicial x_0 para a função
3. Arbitrar o erro ϵ
4. Calcular $f(x_0)$ e $df(x_0)$
5. Calcular o “novo” x , $x = x_0 - f(x)/df(x)$
6. Se $|f(x)| > \epsilon$ retornar ao passo 4

Por exemplo, seja calcular as raízes da mesma equação acima que é $e^x - \sin(x) - 2 = 0$. A derivada é $\frac{d}{dx} = e^x - \cos(x)$. Obteve-se a seguinte tabela

```
-----x-----fx-----dfx
1.0000000 -0.12318915634885141 2.1779795
1.0565612 0.00579321190120519 2.3845934
1.0541318 0.00001105001756940 2.3754999
1.0541271 0.0000000004045120 2.3754825
Raiz: 1.0541271241082415
Erro: 4.0451197946822504e-11
Eis o programa Python que resolve este caso
from numpy import *
def newton1():
    x=1
    tol=0.00001
    fx=exp(x)-sin(x)-2
    dfx=exp(x)-cos(x)
    print("-----x-----fx-----dfx")
    print("{:12.7f}{:20.17f}{:12.7f}"
          .format(x,fx,dfx))
    while abs(fx)>tol:
        x=x-fx/dfx
        fx=exp(x)-sin(x)-2
        dfx=exp(x)-cos(x)
        print("{:12.7f}{:20.17f}{:12.7f}"
              .format(x,fx,dfx))
    print("Raiz: ",x)
    print("Erro: ",fx)
newton1()
```

Para você fazer

1. Resolva pelos 3 métodos, para poder comparar, a seguinte equação $x = \sqrt[24]{6363}$ no intervalo entre 1 e 2 e responda a seguir os 3 valores encontrados

bissecção	
cordas	
Newton	

2. Idem $y = 3 \cdot \sin(x) + 2 \cdot \cos(x) + 2.448197 = 0$ no intervalo entre 3 e 4 e responda a seguir os 3 valores encontrados

bissecção	
cordas	
Newton	

3. Idem $y = ((7 \cdot x)/10) \cdot e^{((8 \cdot x)/10)} - 7703.086200 = 0$ no intervalo entre 8 e 9 e responda a seguir os 3 valores encontrados

bissecção	
cordas	
Newton	

4. Idem $y = 7 \cdot x^5 + 4 \cdot x^2 - 2389.770240 = 0$ no intervalo entre 3 e 4 e responda a seguir os 3 valores encontrados

bissecção	
cordas	
Newton	



Cálculo Numérico - solução de equações

Suponha que precisamos usar o computador para resolver equações. Já paramos para entender o que é resolver uma equação? Conhece-se a forma geral de uma equação $y = f(x)$. Resolver esta equação vem a ser descobrir quais os valores de x que fazem $f(x)$ ser igual a zero. Se olharmos para o gráfico, veremos que busca-se o valor da abscissa x que determina o cruzamento da curva de $f(x)$ no eixo dos x .

Por exemplo, suponha-se precisar descobrir a raiz 11 do número 1739? Ou seja, qual o número que multiplicado por ele mesmo 11 vezes dá como resultado 1739?

Dizendo isso na forma matemática, pode-se escrever $x^{11} = 1739$, ou melhor ainda $x^{11} - 1739 = 0$. Com isso, chegamos ao ponto.

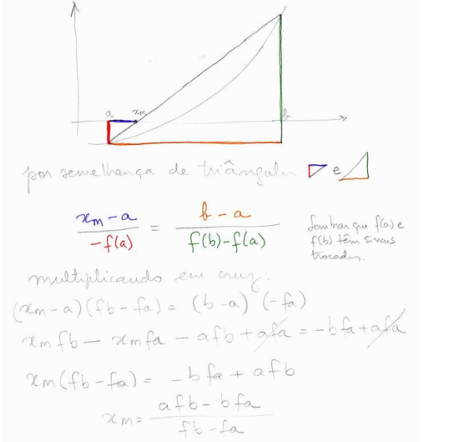
Método da Bissecção Se olharmos o gráfico de uma função, ver-se-á que no entorno da raiz, de um lado a função é positiva, e do outro lado é negativa ou vice versa. Este método começa pela seleção de 2 pontos x_a e x_b sendo que a função $y = f(x)$ têm sinais opostos nesses dois pontos. Graças a essa constatação, pode-se afirmar que a raiz procurada está entre x_a e x_b . Dizendo de outra maneira. O valor de x procurado, está entre a e b . O Método da bissecção vai dividindo o intervalo a, b pela metade, questionando a seguir em qual das metades o x está. Descoberto isso, despreza-se a metade que não contém x . Levando este processo ao limite, descobre-se qual o valor de x . Em resumo, eis as etapas do método:

1. Arbitrar intervalo em $[a, b]$ que compreenda a raiz.
2. Arbitrar um erro permitido ϵ
3. Calcular $x_m = (a + b)/2$
4. Se $f(a).f(x_m) < 0$ o novo intervalo será $[a, x_m]$
5. Se $f(a).f(x_m) > 0$ o novo intervalo será $[x_m, b]$
6. Enquanto o erro $b - a > \epsilon$ voltar ao passo 3.

Seja um exemplo, calcular $f = e^x - \sin(x) - 2 = 0$ nos limites $a = 1$ e $b = 1.25$ com $\epsilon = 0.01$. No primeira etapa tem-se

```
Limite Superior: 1.25
Limite Inferior: 1
Tolerancia: 0.01
-----a-----b-----xm-----fxm-----fa-----fb-----erro
1.1250 1.0000 1.1250 0.1779 0.5414 -0.1232 0.1250
1.0625 1.0000 1.0625 0.0200 0.1779 -0.1232 0.0625
1.0625 1.0312 1.0312 -0.0534 0.0200 -0.1232 0.0312
1.0625 1.0469 1.0469 -0.0171 0.0200 -0.0534 0.0156
1.0547 1.0469 1.0547 0.0013 0.0200 -0.0171 0.0078
Raiz: 1.05078125
Erro: -0.0078125
Eis o programa Python que resolve este caso
import numpy as np
def bissec():
    a=float(input("Limite Superior: "))
    b=float(input("Limite Inferior: "))
    tol=float(input("Tolerancia: "))
    print("-----a-----b-----xm-----fxm-----fa-----fb-----erro")
    while np.abs(b-a)>tol:
        xm=(a+b)/2
        fxm=np.exp(xm)-np.sin(xm)-2
        fa=np.exp(a)-np.sin(a)-2
        fb=np.exp(b)-np.sin(b)-2
        if fa*fxm>0:
            a=xm
        else:
            b=xm
        print('{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}'.format(a,b,xm,fxm,fa,fb,np.abs(b-a)))
    xm=(a+b)/2
    print("Raiz: ",xm)
    print("Erro: ",b-a)
bissec()
```

Cordas ou Falsa Posição



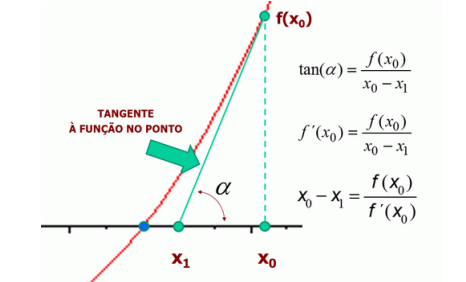
O método começa com uma análise do gráfico da função nas proximidades da raiz. Por semelhança de triângulos, pode-se obter a formulação proposta. Eis os passos do algoritmo:

1. Arbitrar um intervalo $[a, b]$ que compreende a raiz
2. Arbitrar ϵ
3. Calcular $x_m = \frac{a \cdot f(b) - b \cdot f(a)}{f(b) - f(a)}$
4. Se $f(a).f(x_m) < 0$ o novo intervalo será $[a, x_m]$
5. Se $f(a).f(x_m) > 0$ o novo intervalo será $[x_m, b]$
6. Enquanto o erro $f(x_m) > \epsilon$ voltar ao passo 3

Veja a seguir a solução da mesma equação, ($e^x - \sin(x) - 2 = 0$) com os mesmos intervalos do método anterior.

```
Limite inferior: 1
Limite superior: 1.25
Tolerância: 0.001
-----a-----b-----xm-----fxm-----fa-----fb-----erro
1.0463 1.2500 1.0463 -0.0184 -0.1232 0.5414
1.0530 1.2500 1.0530 -0.0026 -0.0184 0.5414
1.0540 1.2500 1.0540 -0.0004 -0.0026 0.5414
Raiz: 1.053972856764054
Erro: -0.0003663936835978099
Eis o programa Python que resolve este caso
from numpy import *
def cordas():
    a=float(input("Limite inferior: "))
    b=float(input("Limite superior: "))
    tol=float(input("Tolerância: "))
    fxm=1
    print("-----a-----b-----xm-----fxm-----fa-----fb-----erro")
    while abs(fxm)>tol:
        fa=exp(a)-sin(a)-2
        fb=exp(b)-sin(b)-2
        xm=((a*fb)-(b*fa))/(fb-fa)
        fxm=exp(xm)-sin(xm)-2
        if fa*fxm>0:
            a=xm
        else:
            b=xm
        print('{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}'.format(a,b,xm,fxm,fa,fb))
    print("Raiz: ",xm)
    print("Erro: ",fxm)
cordas()
```

Método de Newton Este método começa com a interpretação geométrica da derivada. Lembrando, Newton é um monstro da ciência. Entre outras coisas, ele criou o cálculo diferencial e integral, a teoria da luz, a gravitação, as leis do movimento, além de, no fim da vida ter – como funcionário público – dirigido a casa da moeda inglesa, pendurando na forca alguns falsários delinquentes. Voltando ao cálculo diferencial, uma das interpretações da derivada de uma função em um ponto é a tangente da curva da função naquele ponto. Veja-se o que se disse neste desenho



- Eis as etapas do método:
1. Obter a derivada da função a resolver

2. Arbitrar um ponto inicial x_0 para a função
3. Arbitrar o erro ϵ
4. Calcular $f(x_0)$ e $df(x_0)$
5. Calcular o “novo” x , $x = x_0 - f(x)/df(x)$
6. Se $|f(x)| > \epsilon$ retornar ao passo 4

Por exemplo, seja calcular as raízes da mesma equação acima que é $e^x - \sin(x) - 2 = 0$. A derivada é $\frac{d}{dx} = e^x - \cos(x)$. Obteve-se a seguinte tabela

```
-----x-----fx-----dfx
1.0000000 -0.12318915634885141 2.1779795
1.0565612 0.00579321190120519 2.3845934
1.0541318 0.00001105001756940 2.3754999
1.0541271 0.00000000004045120 2.3754825
Raiz: 1.0541271241082415
Erro: 4.0451197946822504e-11
Eis o programa Python que resolve este caso
from numpy import *
def newton1():
    x=1
    tol=0.00001
    fx=exp(x)-sin(x)-2
    dfx=exp(x)-cos(x)
    print("-----x-----fx-----dfx")
    print("{:12.7f}{:20.17f}{:12.7f}".format(x,fx,dfx))
    while abs(fx)>tol:
        x=x-fx/dfx
        fx=exp(x)-sin(x)-2
        dfx=exp(x)-cos(x)
        print("{:12.7f}{:20.17f}{:12.7f}".format(x,fx,dfx))
    print("Raiz: ",x)
    print("Erro: ",fx)
newton1()
```

Para você fazer

1. Resolva pelos 3 métodos, para poder comparar, a seguinte equação $x = \sqrt[26]{8263}$ no intervalo entre 1 e 2 e responda a seguir os 3 valores encontrados

bissecção	
cordas	
Newton	

2. Idem $y = 4 \cdot \sin(x) + 3 \cdot \cos(x) + 3.651371 = 0$ no intervalo entre 9 e 10 e responda a seguir os 3 valores encontrados

bissecção	
cordas	
Newton	

3. Idem $y = ((6 \cdot x)/10) \cdot e^{((8 \cdot x)/10)} - 5018.752017 = 0$ no intervalo entre 8 e 9 e responda a seguir os 3 valores encontrados

bissecção	
cordas	
Newton	

4. Idem $y = 7 \cdot x^6 + 5 \cdot x^2 - 4829530.267392 = 0$ no intervalo entre 9 e 10 e responda a seguir os 3 valores encontrados

bissecção	
cordas	
Newton	



Cálculo Numérico - solução de equações

Suponha que precisamos usar o computador para resolver equações. Já paramos para entender o que é resolver uma equação? Conhece-se a forma geral de uma equação $y = f(x)$. Resolver esta equação vem a ser descobrir quais os valores de x que fazem $f(x)$ ser igual a zero. Se olharmos para o gráfico, veremos que busca-se o valor da abscissa x que determina o cruzamento da curva de $f(x)$ no eixo dos x .

Por exemplo, suponha-se precisar descobrir a raiz 11 do número 1739? Ou seja, qual o número que multiplicado por ele mesmo 11 vezes dá como resultado 1739?

Dizendo isso na forma matemática, pode-se escrever $x^{11} = 1739$, ou melhor ainda $x^{11} - 1739 = 0$. Com isso, chegamos ao ponto.

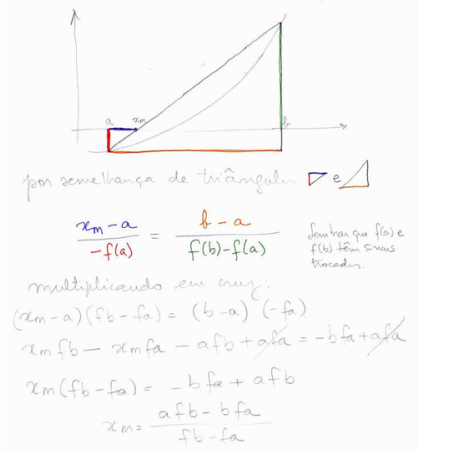
Método da Bissecção Se olharmos o gráfico de uma função, ver-se-á que no entorno da raiz, de um lado a função é positiva, e do outro lado é negativa ou vice versa. Este método começa pela seleção de 2 pontos x_a e x_b sendo que a função $y = f(x)$ têm sinais opostos nesses dois pontos. Graças a essa constatação, pode-se afirmar que a raiz procurada está entre x_a e x_b . Dizendo de outra maneira. O valor de x procurado, está entre a e b . O Método da bissecção vai dividindo o intervalo a, b pela metade, questionando a seguir em qual das metades o x está. Descoberto isso, despreza-se a metade que não contém x . Levando este processo ao limite, descobre-se qual o valor de x . Em resumo, eis as etapas do método:

1. Arbitrar intervalo em $[a, b]$ que compreenda a raiz.
2. Arbitrar um erro permitido ϵ
3. Calcular $x_m = (a + b)/2$
4. Se $f(a).f(x_m) < 0$ o novo intervalo será $[a, x_m]$
5. Se $f(a).f(x_m) > 0$ o novo intervalo será $[x_m, b]$
6. Enquanto o erro $b - a > \epsilon$ voltar ao passo 3.

Seja um exemplo, calcular $f = e^x - \sin(x) - 2 = 0$ nos limites $a = 1$ e $b = 1.25$ com $\epsilon = 0.01$. No primeira etapa tem-se

```
Limite Superior: 1.25
Limite Inferior: 1
Tolerancia: 0.01
-----a-----b-----xm-----fxm-----fa-----fb-----erro
1.1250 1.0000 1.1250 0.1779 0.5414 -0.1232 0.1250
1.0625 1.0000 1.0625 0.0200 0.1779 -0.1232 0.0625
1.0625 1.0312 1.0312 -0.0534 0.0200 -0.1232 0.0312
1.0625 1.0469 1.0469 -0.0171 0.0200 -0.0534 0.0156
1.0547 1.0469 1.0547 0.0013 0.0200 -0.0171 0.0078
Raiz: 1.05078125
Erro: -0.0078125
Eis o programa Python que resolve este caso
import numpy as np
def bissec():
    a=float(input("Limite Superior: "))
    b=float(input("Limite Inferior: "))
    tol=float(input("Tolerancia: "))
    print("-----a-----b-----xm-----fxm-----fa-----fb-----erro")
    while np.abs(b-a)>tol:
        xm=(a+b)/2
        fxm=np.exp(xm)-np.sin(xm)-2
        fa=np.exp(a)-np.sin(a)-2
        fb=np.exp(b)-np.sin(b)-2
        if fa*fxm>0:
            a=xm
        else:
            b=xm
        print('{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}'.format(a,b,xm,fxm,fa,fb,np.abs(b-a)))
    xm=(a+b)/2
    print("Raiz: ",xm)
    print("Erro: ",b-a)
bissec()
```

Cordas ou Falsa Posição



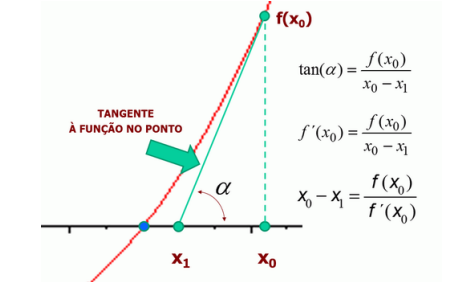
O método começa com uma análise do gráfico da função nas proximidades da raiz. Por semelhança de triângulos, pode-se obter a formulação proposta. Eis os passos do algoritmo:

1. Arbitrar um intervalo $[a, b]$ que compreende a raiz
2. Arbitrar ϵ
3. Calcular $x_m = \frac{a \cdot f(b) - b \cdot f(a)}{f(b) - f(a)}$
4. Se $f(a).f(x_m) < 0$ o novo intervalo será $[a, x_m]$
5. Se $f(a).f(x_m) > 0$ o novo intervalo será $[x_m, b]$
6. Enquanto o erro $f(x_m) > \epsilon$ voltar ao passo 3

Veja a seguir a solução da mesma equação, ($e^x - \sin(x) - 2 = 0$) com os mesmos intervalos do método anterior.

```
Limite inferior: 1
Limite superior: 1.25
Tolerância: 0.001
-----a-----b-----xm-----fxm-----fa-----fb-----erro
1.0463 1.2500 1.0463 -0.0184 -0.1232 0.5414
1.0530 1.2500 1.0530 -0.0026 -0.0184 0.5414
1.0540 1.2500 1.0540 -0.0004 -0.0026 0.5414
Raiz: 1.053972856764054
Erro: -0.0003663936835978099
Eis o programa Python que resolve este caso
from numpy import *
def cordas():
    a=float(input("Limite inferior: "))
    b=float(input("Limite superior: "))
    tol=float(input("Tolerância: "))
    fxm=1
    print("-----a-----b-----xm-----fxm-----fa-----fb-----erro")
    while abs(fxm)>tol:
        fa=exp(a)-sin(a)-2
        fb=exp(b)-sin(b)-2
        fxm=((a*fb)-(b*fa))/(fb-fa)
        fxm=exp(xm)-sin(xm)-2
        if fa*fxm>0:
            a=fxm
        else:
            b=fxm
        print('{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}'.format(a,b,xm,fxm,fa,fb))
    print("Raiz: ",xm)
    print("Erro: ",fxm)
cordas()
```

Método de Newton Este método começa com a interpretação geométrica da derivada. Lembrando, Newton é um monstro da ciência. Entre outras coisas, ele criou o cálculo diferencial e integral, a teoria da luz, a gravitação, as leis do movimento, além de, no fim da vida ter – como funcionário público – dirigido a casa da moeda inglesa, pendurando na forca alguns falsários delinquentes. Voltando ao cálculo diferencial, uma das interpretações da derivada de uma função em um ponto é a tangente da curva da função naquele ponto. Veja-se o que se disse neste desenho



- Eis as etapas do método:
1. Obter a derivada da função a resolver

2. Arbitrar um ponto inicial x_0 para a função
3. Arbitrar o erro ϵ
4. Calcular $f(x_0)$ e $df(x_0)$
5. Calcular o “novo” x , $x = x_0 - f(x)/df(x)$
6. Se $|f(x)| > \epsilon$ retornar ao passo 4

Por exemplo, seja calcular as raízes da mesma equação acima que é $e^x - \sin(x) - 2 = 0$. A derivada é $\frac{d}{dx} = e^x - \cos(x)$. Obteve-se a seguinte tabela

```
-----x-----fx-----dfx
1.0000000 -0.12318915634885141 2.1779795
1.0565612 0.00579321190120519 2.3845934
1.0541318 0.00001105001756940 2.3754999
1.0541271 0.00000000004045120 2.3754825
Raiz: 1.0541271241082415
Erro: 4.0451197946822504e-11
Eis o programa Python que resolve este caso
from numpy import *
def newton():
    x=1
    tol=0.00001
    fx=exp(x)-sin(x)-2
    dfx=exp(x)-cos(x)
    print("-----x-----fx-----dfx")
    print("{:12.7f}{:20.17f}{:12.7f}".format(x,fx,dfx))
    while abs(fx)>tol:
        x=x-fx/dfx
        fx=exp(x)-sin(x)-2
        dfx=exp(x)-cos(x)
        print("{:12.7f}{:20.17f}{:12.7f}".format(x,fx,dfx))
    print("Raiz: ",x)
    print("Erro: ",fx)
newton()
```

Para você fazer

1. Resolva pelos 3 métodos, para poder comparar, a seguinte equação $x = \sqrt[2]{5291}$ no intervalo entre 1 e 2 e responda a seguir os 3 valores encontrados

bissecção	
cordas	
Newton	

2. Idem $y = 4 \sin(x) + 6 \cos(x) - 6.066407 = 0$ no intervalo entre 6 e 7 e responda a seguir os 3 valores encontrados

bissecção	
cordas	
Newton	

3. Idem $y = ((3 * x)/10) * e^{((4 * x)/10)} - 5.567821 = 0$ no intervalo entre 3 e 4 e responda a seguir os 3 valores encontrados

bissecção	
cordas	
Newton	

4. Idem $y = 7 * x^6 + 5 * x^4 - 367565.541027 = 0$ no intervalo entre 6 e 7 e responda a seguir os 3 valores encontrados

bissecção	
cordas	
Newton	



Cálculo Numérico - solução de equações

Suponha que precisamos usar o computador para resolver equações. Já paramos para entender o que é resolver uma equação? Conhece-se a forma geral de uma equação $y = f(x)$. Resolver esta equação vem a ser descobrir quais os valores de x que fazem $f(x)$ ser igual a zero. Se olharmos para o gráfico, veremos que busca-se o valor da abscissa x que determina o cruzamento da curva de $f(x)$ no eixo dos x .

Por exemplo, suponha-se precisar descobrir a raiz 11 do número 1739? Ou seja, qual o número que multiplicado por ele mesmo 11 vezes dá como resultado 1739?

Dizendo isso na forma matemática, pode-se escrever $x^{11} = 1739$, ou melhor ainda $x^{11} - 1739 = 0$. Com isso, chegamos ao ponto.

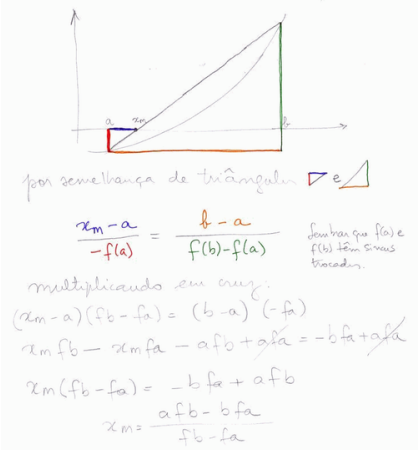
Método da Bissecção Se olharmos o gráfico de uma função, ver-se-á que no entorno da raiz, de um lado a função é positiva, e do outro lado é negativa ou vice versa. Este método começa pela seleção de 2 pontos x_a e x_b sendo que a função $y = f(x)$ têm sinais opostos nesses dois pontos. Graças a essa constatação, pode-se afirmar que a raiz procurada está entre x_a e x_b . Dizendo de outra maneira. O valor de x procurado, está entre a e b . O Método da bissecção vai dividindo o intervalo a, b pela metade, questionando a seguir em qual das metades o x está. Descoberto isso, despreza-se a metade que não contém x . Levando este processo ao limite, descobre-se qual o valor de x . Em resumo, eis as etapas do método:

1. Arbitrar intervalo em $[a, b]$ que compreenda a raiz.
2. Arbitrar um erro permitido ϵ
3. Calcular $x_m = (a + b)/2$
4. Se $f(a).f(x_m) < 0$ o novo intervalo será $[a, x_m]$
5. Se $f(a).f(x_m) > 0$ o novo intervalo será $[x_m, b]$
6. Enquanto o erro $b - a > \epsilon$ voltar ao passo 3.

Seja um exemplo, calcular $f = e^x - \sin(x) - 2 = 0$ nos limites $a = 1$ e $b = 1.25$ com $\epsilon = 0.01$. No primeira etapa tem-se

```
Limite Superior: 1.25
Limite Inferior: 1
Tolerancia: 0.01
-----a-----b-----xm-----fxm-----fa-----fb-----erro
1.1250 1.0000 1.1250 0.1779 0.5414 -0.1232 0.1250
1.0625 1.0000 1.0625 0.0200 0.1779 -0.1232 0.0625
1.0625 1.0312 1.0312 -0.0534 0.0200 -0.1232 0.0312
1.0625 1.0469 1.0469 -0.0171 0.0200 -0.0534 0.0156
1.0547 1.0469 1.0547 0.0013 0.0200 -0.0171 0.0078
Raiz: 1.05078125
Erro: -0.0078125
Eis o programa Python que resolve este caso
import numpy as np
def bissec():
    a=float(input("Limite Superior: "))
    b=float(input("Limite Inferior: "))
    tol=float(input("Tolerancia: "))
    print("-----a-----b-----xm-----fxm-----fa-----fb-----erro")
    while np.abs(b-a)>tol:
        xm=(a+b)/2
        fxm=np.exp(xm)-np.sin(xm)-2
        fa=np.exp(a)-np.sin(a)-2
        fb=np.exp(b)-np.sin(b)-2
        if fa*fxm>0:
            a=xm
        else:
            b=xm
        print('{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}'.format(a,b,xm,fxm,fa,fb,np.abs(b-a)))
    xm=(a+b)/2
    print("Raiz: ",xm)
    print("Erro: ",b-a)
bissec()
```

Cordas ou Falsa Posição



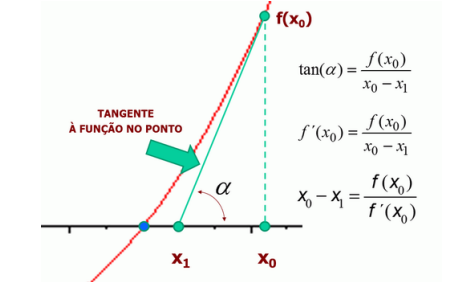
O método começa com uma análise do gráfico da função nas proximidades da raiz. Por semelhança de triângulos, pode-se obter a formulação proposta. Eis os passos do algoritmo:

1. Arbitrar um intervalo $[a, b]$ que compreende a raiz
2. Arbitrar ϵ
3. Calcular $x_m = \frac{a \cdot f(b) - b \cdot f(a)}{f(b) - f(a)}$
4. Se $f(a).f(x_m) < 0$ o novo intervalo será $[a, x_m]$
5. Se $f(a).f(x_m) > 0$ o novo intervalo será $[x_m, b]$
6. Enquanto o erro $f(x_m) > \epsilon$ voltar ao passo 3

Veja a seguir a solução da mesma equação, ($e^x - \sin(x) - 2 = 0$) com os mesmos intervalos do método anterior.

```
Limite inferior: 1
Limite superior: 1.25
Tolerância: 0.001
-----a-----b-----xm-----fxm-----fa-----fb-----erro
1.0463 1.2500 1.0463 -0.0184 -0.1232 0.5414
1.0530 1.2500 1.0530 -0.0026 -0.0184 0.5414
1.0540 1.2500 1.0540 -0.0004 -0.0026 0.5414
Raiz: 1.053972856764054
Erro: -0.0003663936835978099
Eis o programa Python que resolve este caso
from numpy import *
def cordas():
    a=float(input("Limite inferior: "))
    b=float(input("Limite superior: "))
    tol=float(input("Tolerância: "))
    fxm=1
    print("-----a-----b-----xm-----fxm-----fa-----fb-----erro")
    while abs(fxm)>tol:
        fa=exp(a)-sin(a)-2
        fb=exp(b)-sin(b)-2
        xm=((a*fb)-(b*fa))/(fb-fa)
        fxm=exp(xm)-sin(xm)-2
        if fa*fxm>0:
            a=xm
        else:
            b=xm
        print('{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}'.format(a,b,xm,fxm,fa,fb))
    print("Raiz: ",xm)
    print("Erro: ",fxm)
cordas()
```

Método de Newton Este método começa com a interpretação geométrica da derivada. Lembrando, Newton é um monstro da ciência. Entre outras coisas, ele criou o cálculo diferencial e integral, a teoria da luz, a gravitação, as leis do movimento, além de, no fim da vida ter – como funcionário público – dirigido a casa da moeda inglesa, pendurando na forca alguns falsários delinquentes. Voltando ao cálculo diferencial, uma das interpretações da derivada de uma função em um ponto é a tangente da curva da função naquele ponto. Veja-se o que se disse neste desenho



- Eis as etapas do método:
1. Obter a derivada da função a resolver

2. Arbitrar um ponto inicial x_0 para a função
3. Arbitrar o erro ϵ
4. Calcular $f(x_0)$ e $df(x_0)$
5. Calcular o “novo” x , $x = x_0 - f(x)/df(x)$
6. Se $|f(x)| > \epsilon$ retornar ao passo 4

Por exemplo, seja calcular as raízes da mesma equação acima que é $e^x - \sin(x) - 2 = 0$. A derivada é $\frac{d}{dx} = e^x - \cos(x)$. Obteve-se a seguinte tabela

```
-----x-----fx-----dfx
1.0000000 -0.12318915634885141 2.1779795
1.0565612 0.00579321190120519 2.3845934
1.0541318 0.00001105001756940 2.3754999
1.0541271 0.00000000004045120 2.3754825
Raiz: 1.0541271241082415
Erro: 4.0451197946822504e-11
Eis o programa Python que resolve este caso
from numpy import *
def newton1():
    x=1
    tol=0.00001
    fx=exp(x)-sin(x)-2
    dfx=exp(x)-cos(x)
    print("-----x-----fx-----dfx")
    print("{:12.7f}{:20.17f}{:12.7f}".format(x,fx,dfx))
    while abs(fx)>tol:
        x=x-fx/dfx
        fx=exp(x)-sin(x)-2
        dfx=exp(x)-cos(x)
        print("{:12.7f}{:20.17f}{:12.7f}".format(x,fx,dfx))
    print("Raiz: ",x)
    print("Erro: ",fx)
newton1()
```

Para você fazer

1. Resolva pelos 3 métodos, para poder comparar, a seguinte equação $x = \sqrt[24]{6386}$ no intervalo entre 1 e 2 e responda a seguir os 3 valores encontrados

bissecção	
cordas	
Newton	

2. Idem $y = 4 \sin(x) + 3 \cos(x) - 1.860530 = 0$ no intervalo entre 8 e 9 e responda a seguir os 3 valores encontrados

bissecção	
cordas	
Newton	

3. Idem $y = ((5 * x)/10) * e^{((6 * x)/10)} - 2.089357 = 0$ no intervalo entre 1 e 2 e responda a seguir os 3 valores encontrados

bissecção	
cordas	
Newton	

4. Idem $y = 7 * x^6 + 5 * x^2 - 817.859328 = 0$ no intervalo entre 2 e 3 e responda a seguir os 3 valores encontrados

bissecção	
cordas	
Newton	

Cálculo Numérico - solução de equações

Suponha que precisamos usar o computador para resolver equações. Já paramos para entender o que é resolver uma equação? Conhece-se a forma geral de uma equação $y = f(x)$. Resolver esta equação vem a ser descobrir quais os valores de x que fazem $f(x)$ ser igual a zero. Se olharmos para o gráfico, veremos que busca-se o valor da abscissa x que determina o cruzamento da curva de $f(x)$ no eixo dos x .

Por exemplo, suponha-se precisar descobrir a raiz 11 do número 1739? Ou seja, qual o número que multiplicado por ele mesmo 11 vezes dá como resultado 1739?

Dizendo isso na forma matemática, pode-se escrever $x^{11} = 1739$, ou melhor ainda $x^{11} - 1739 = 0$. Com isso, chegamos ao ponto.

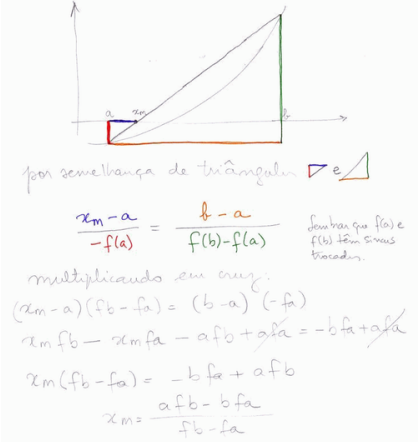
Método da Bissecção Se olharmos o gráfico de uma função, ver-se-á que no entorno da raiz, de um lado a função é positiva, e do outro lado é negativa ou vice versa. Este método começa pela seleção de 2 pontos x_a e x_b sendo que a função $y = f(x)$ têm sinais opostos nesses dois pontos. Graças a essa constatação, pode-se afirmar que a raiz procurada está entre x_a e x_b . Dizendo de outra maneira. O valor de x procurado, está entre a e b . O Método da bissecção vai dividindo o intervalo a, b pela metade, questionando a seguir em qual das metades o x está. Descoberto isso, despreza-se a metade que não contém x . Levando este processo ao limite, descobre-se qual o valor de x . Em resumo, eis as etapas do método:

1. Arbitrar intervalo em $[a, b]$ que compreenda a raiz.
2. Arbitrar um erro permitido ϵ
3. Calcular $x_m = (a + b)/2$
4. Se $f(a).f(x_m) < 0$ o novo intervalo será $[a, x_m]$
5. Se $f(a).f(x_m) > 0$ o novo intervalo será $[x_m, b]$
6. Enquanto o erro $b - a > \epsilon$ voltar ao passo 3.

Seja um exemplo, calcular $f = e^x - \sin(x) - 2 = 0$ nos limites $a = 1$ e $b = 1.25$ com $\epsilon = 0.01$. No primeira etapa tem-se

```
Limite Superior: 1.25
Limite Inferior: 1
Tolerancia: 0.01
-----a-----b-----xm-----fxm-----fa-----fb-----erro
1.1250 1.0000 1.1250 0.1779 0.5414 -0.1232 0.1250
1.0625 1.0000 1.0625 0.0200 0.1779 -0.1232 0.0625
1.0625 1.0312 1.0312 -0.0534 0.0200 -0.1232 0.0312
1.0625 1.0469 1.0469 -0.0171 0.0200 -0.0534 0.0156
1.0547 1.0469 1.0547 0.0013 0.0200 -0.0171 0.0078
Raiz: 1.05078125
Erro: -0.0078125
Eis o programa Python que resolve este caso
import numpy as np
def bissec():
    a=float(input("Limite Superior: "))
    b=float(input("Limite Inferior: "))
    tol=float(input("Tolerancia: "))
    print("-----a-----b-----xm-----fxm-----fa-----fb-----erro")
    while np.abs(b-a)>tol:
        xm=(a+b)/2
        fxm=np.exp(xm)-np.sin(xm)-2
        fa=np.exp(a)-np.sin(a)-2
        fb=np.exp(b)-np.sin(b)-2
        if fa*fxm>0:
            a=xm
        else:
            b=xm
        print('{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}'.format(a,b,xm,fxm,fa,fb,np.abs(b-a)))
    xm=(a+b)/2
    print("Raiz: ",xm)
    print("Erro: ",b-a)
bissec()
```

Cordas ou Falsa Posição



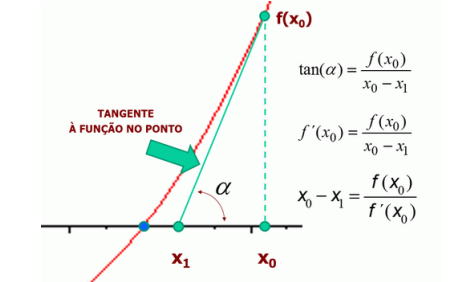
O método começa com uma análise do gráfico da função nas proximidades da raiz. Por semelhança de triângulos, pode-se obter a formulação proposta. Eis os passos do algoritmo:

1. Arbitrar um intervalo $[a, b]$ que compreende a raiz
2. Arbitrar ϵ
3. Calcular $x_m = \frac{a \cdot f(b) - b \cdot f(a)}{f(b) - f(a)}$
4. Se $f(a).f(x_m) < 0$ o novo intervalo será $[a, x_m]$
5. Se $f(a).f(x_m) > 0$ o novo intervalo será $[x_m, b]$
6. Enquanto o erro $f(x_m) > \epsilon$ voltar ao passo 3

Veja a seguir a solução da mesma equação, ($e^x - \sin(x) - 2 = 0$) com os mesmos intervalos do método anterior.

```
Limite inferior: 1
Limite superior: 1.25
Tolerância: 0.001
-----a-----b-----xm-----fxm-----fa-----fb-----erro
1.0463 1.2500 1.0463 -0.0184 -0.1232 0.5414
1.0530 1.2500 1.0530 -0.0026 -0.0184 0.5414
1.0540 1.2500 1.0540 -0.0004 -0.0026 0.5414
Raiz: 1.053972856764054
Erro: -0.0003663936835978099
Eis o programa Python que resolve este caso
from numpy import *
def cordas():
    a=float(input("Limite inferior: "))
    b=float(input("Limite superior: "))
    tol=float(input("Tolerância: "))
    fxm=1
    print("-----a-----b-----xm-----fxm-----fa-----fb-----erro")
    while abs(fxm)>tol:
        fa=exp(a)-sin(a)-2
        fb=exp(b)-sin(b)-2
        xm=((a*fb)-(b*fa))/(fb-fa)
        fxm=exp(xm)-sin(xm)-2
        if fa*fxm>0:
            a=xm
        else:
            b=xm
        print('{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}'.format(a,b,xm,fxm,fa,fb))
    print("Raiz: ",xm)
    print("Erro: ",fxm)
cordas()
```

Método de Newton Este método começa com a interpretação geométrica da derivada. Lembrando, Newton é um monstro da ciência. Entre outras coisas, ele criou o cálculo diferencial e integral, a teoria da luz, a gravitação, as leis do movimento, além de, no fim da vida ter – como funcionário público – dirigido a casa da moeda inglesa, pendurando na força alguns falsários delinquentes. Voltando ao cálculo diferencial, uma das interpretações da derivada de uma função em um ponto é a tangente da curva da função naquele ponto. Veja-se o que se disse neste desenho



- Eis as etapas do método:
1. Obter a derivada da função a resolver

2. Arbitrar um ponto inicial x_0 para a função
3. Arbitrar o erro ϵ
4. Calcular $f(x_0)$ e $df(x_0)$
5. Calcular o “novo” x , $x = x_0 - f(x)/df(x)$
6. Se $|f(x)| > \epsilon$ retornar ao passo 4

Por exemplo, seja calcular as raízes da mesma equação acima que é $e^x - \sin(x) - 2 = 0$. A derivada é $\frac{d}{dx} = e^x - \cos(x)$. Obteve-se a seguinte tabela

```
-----x-----fx-----dfx
1.0000000 -0.12318915634885141 2.1779795
1.0565612 0.00579321190120519 2.3845934
1.0541318 0.00001105001756940 2.3754999
1.0541271 0.00000000004045120 2.3754825
Raiz: 1.0541271241082415
Erro: 4.0451197946822504e-11
Eis o programa Python que resolve este caso
from numpy import *
def newton():
    x=1
    tol=0.00001
    fx=exp(x)-sin(x)-2
    dfx=exp(x)-cos(x)
    print("-----x-----fx-----dfx")
    print("{:12.7f}{:20.17f}{:12.7f}".format(x,fx,dfx))
    while abs(fx)>tol:
        x=x-fx/dfx
        fx=exp(x)-sin(x)-2
        dfx=exp(x)-cos(x)
        print("{:12.7f}{:20.17f}{:12.7f}".format(x,fx,dfx))
    print("Raiz: ",x)
    print("Erro: ",fx)
newton()
```

Para você fazer

1. Resolva pelos 3 métodos, para poder comparar, a seguinte equação $x = \sqrt[18]{8822}$ no intervalo entre 1 e 2 e responda a seguir os 3 valores encontrados

bissecção	
cordas	
Newton	

2. Idem $y = 8*\sin(x) + 7*\cos(x) - 10.038689 = 0$ no intervalo entre 6 e 7 e responda a seguir os 3 valores encontrados

bissecção	
cordas	
Newton	

3. Idem $y = ((6 * x)/10) * e^{((5*x)/10)} - 1.905300 = 0$ no intervalo entre 1 e 2 e responda a seguir os 3 valores encontrados

bissecção	
cordas	
Newton	

4. Idem $y = 7 * x^4 + 3 * x^2 - 292.187500 = 0$ no intervalo entre 2 e 3 e responda a seguir os 3 valores encontrados

bissecção	
cordas	
Newton	

Cálculo Numérico - solução de equações

Suponha que precisamos usar o computador para resolver equações. Já paramos para entender o que é resolver uma equação? Conhece-se a forma geral de uma equação $y = f(x)$. Resolver esta equação vem a ser descobrir quais os valores de x que fazem $f(x)$ ser igual a zero. Se olharmos para o gráfico, veremos que busca-se o valor da abscissa x que determina o cruzamento da curva de $f(x)$ no eixo dos x .

Por exemplo, suponha-se precisar descobrir a raiz 11 do número 1739? Ou seja, qual o número que multiplicado por ele mesmo 11 vezes dá como resultado 1739?

Dizendo isso na forma matemática, pode-se escrever $x^{11} = 1739$, ou melhor ainda $x^{11} - 1739 = 0$. Com isso, chegamos ao ponto.

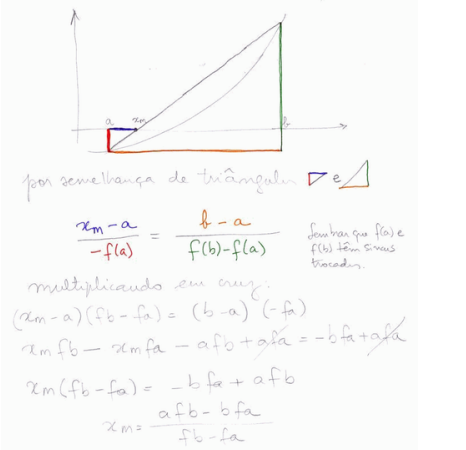
Método da Bissecção Se olharmos o gráfico de uma função, ver-se-á que no entorno da raiz, de um lado a função é positiva, e do outro lado é negativa ou vice versa. Este método começa pela seleção de 2 pontos x_a e x_b sendo que a função $y = f(x)$ têm sinais opostos nesses dois pontos. Graças a essa constatação, pode-se afirmar que a raiz procurada está entre x_a e x_b . Dizendo de outra maneira. O valor de x procurado, está entre a e b . O Método da bissecção vai dividindo o intervalo a, b pela metade, questionando a seguir em qual das metades o x está. Descoberto isso, despreza-se a metade que não contém x . Levando este processo ao limite, descobre-se qual o valor de x . Em resumo, eis as etapas do método:

1. Arbitrar intervalo em $[a, b]$ que compreenda a raiz.
2. Arbitrar um erro permitido ϵ
3. Calcular $x_m = (a + b)/2$
4. Se $f(a).f(x_m) < 0$ o novo intervalo será $[a, x_m]$
5. Se $f(a).f(x_m) > 0$ o novo intervalo será $[x_m, b]$
6. Enquanto o erro $b - a > \epsilon$ voltar ao passo 3.

Seja um exemplo, calcular $f = e^x - \sin(x) - 2 = 0$ nos limites $a = 1$ e $b = 1.25$ com $\epsilon = 0.01$. No primeira etapa tem-se

```
Limite Superior: 1.25
Limite Inferior: 1
Tolerancia: 0.01
-----a-----b-----xm-----fxm-----fa-----fb-----erro
1.1250 1.0000 1.1250 0.1779 0.5414 -0.1232 0.1250
1.0625 1.0000 1.0625 0.0200 0.1779 -0.1232 0.0625
1.0625 1.0312 1.0312 -0.0534 0.0200 -0.1232 0.0312
1.0625 1.0469 1.0469 -0.0171 0.0200 -0.0534 0.0156
1.0547 1.0469 1.0547 0.0013 0.0200 -0.0171 0.0078
Raiz: 1.05078125
Erro: -0.0078125
Eis o programa Python que resolve este caso
import numpy as np
def bissec():
    a=float(input("Limite Superior: "))
    b=float(input("Limite Inferior: "))
    tol=float(input("Tolerancia: "))
    print("-----a-----b-----xm-----fxm-----fa-----fb-----erro")
    while np.abs(b-a)>tol:
        xm=(a+b)/2
        fxm=np.exp(xm)-np.sin(xm)-2
        fa=np.exp(a)-np.sin(a)-2
        fb=np.exp(b)-np.sin(b)-2
        if fa*fxm>0:
            a=xm
        else:
            b=xm
        print('{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}'
              .format(a,b,xm,fxm,fa,fb,np.abs(b-a)))
    xm=(a+b)/2
    print("Raiz: ",xm)
    print("Erro: ",b-a)
bissec()
```

Cordas ou Falsa Posição



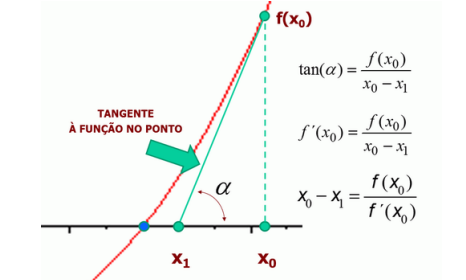
O método começa com uma análise do gráfico da função nas proximidades da raiz. Por semelhança de triângulos, pode-se obter a formulação proposta. Eis os passos do algoritmo:

1. Arbitrar um intervalo $[a, b]$ que compreende a raiz
2. Arbitrar ϵ
3. Calcular $x_m = \frac{a \cdot f(b) - b \cdot f(a)}{f(b) - f(a)}$
4. Se $f(a).f(x_m) < 0$ o novo intervalo será $[a, x_m]$
5. Se $f(a).f(x_m) > 0$ o novo intervalo será $[x_m, b]$
6. Enquanto o erro $f(x_m) > \epsilon$ voltar ao passo 3

Veja a seguir a solução da mesma equação, ($e^x - \sin(x) - 2 = 0$) com os mesmos intervalos do método anterior.

```
Limite inferior: 1
Limite superior: 1.25
Tolerância: 0.001
-----a-----b-----xm-----fxm-----fa-----fb-----erro
1.0463 1.2500 1.0463 -0.0184 -0.1232 0.5414
1.0530 1.2500 1.0530 -0.0026 -0.0184 0.5414
1.0540 1.2500 1.0540 -0.0004 -0.0026 0.5414
Raiz: 1.053972856764054
Erro: -0.0003663936835978099
Eis o programa Python que resolve este caso
from numpy import *
def cordas():
    a=float(input("Limite inferior: "))
    b=float(input("Limite superior: "))
    tol=float(input("Tolerância: "))
    fxm=1
    print("-----a-----b-----xm-----fxm-----fa-----fb-----erro")
    while abs(fxm)>tol:
        fa=exp(a)-sin(a)-2
        fb=exp(b)-sin(b)-2
        fxm=((a*fb)-(b*fa))/(fb-fa)
        fxm=exp(xm)-sin(xm)-2
        if fa*fxm>0:
            a=fxm
        else:
            b=fxm
        print('{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}'
              .format(a,b,xm,fxm,fa,fb))
    print("Raiz: ",xm)
    print("Erro: ",fxm)
cordas()
```

Método de Newton Este método começa com a interpretação geométrica da derivada. Lembrando, Newton é um monstro da ciência. Entre outras coisas, ele criou o cálculo diferencial e integral, a teoria da luz, a gravitação, as leis do movimento, além de, no fim da vida ter – como funcionário público – dirigido a casa da moeda inglesa, pendurando na forca alguns falsários delinquentes. Voltando ao cálculo diferencial, uma das interpretações da derivada de uma função em um ponto é a tangente da curva da função naquele ponto. Veja-se o que se disse neste desenho



- Eis as etapas do método:
1. Obter a derivada da função a resolver

2. Arbitrar um ponto inicial x_0 para a função
3. Arbitrar o erro ϵ
4. Calcular $f(x_0)$ e $df(x_0)$
5. Calcular o “novo” x , $x = x_0 - f(x)/dfx$
6. Se $|fx| > \epsilon$ retornar ao passo 4

Por exemplo, seja calcular as raízes da mesma equação acima que é $e^x - \sin(x) - 2 = 0$. A derivada é $\frac{d}{dx} = e^x - \cos(x)$. Obteve-se a seguinte tabela

```
-----x-----fx-----dfx
1.0000000 -0.12318915634885141 2.1779795
1.0565612 0.00579321190120519 2.3845934
1.0541318 0.00001105001756940 2.3754999
1.0541271 0.00000000004045120 2.3754825
Raiz: 1.0541271241082415
Erro: 4.0451197946822504e-11
Eis o programa Python que resolve este caso
from numpy import *
def newton1():
    x=1
    tol=0.00001
    fx=exp(x)-sin(x)-2
    dfx=exp(x)-cos(x)
    print("-----x-----fx-----dfx")
    print("{:12.7f}{:20.17f}{:12.7f}".format(x,fx,dfx))
    while abs(fx)>tol:
        x=x-fx/dfx
        fx=exp(x)-sin(x)-2
        dfx=exp(x)-cos(x)
        print("{:12.7f}{:20.17f}{:12.7f}".format(x,fx,dfx))
    print("Raiz: ",x)
    print("Erro: ",fx)
newton1()
```

Para você fazer

1. Resolva pelos 3 métodos, para poder comparar, a seguinte equação $x = \sqrt[2]{6331}$ no intervalo entre 1 e 2 e responda a seguir os 3 valores encontrados

bissecção	
cordas	
Newton	

2. Idem $y = 8 * \sin(x) + 7 * \cos(x) - 9.143959 = 0$ no intervalo entre 6 e 7 e responda a seguir os 3 valores encontrados

bissecção	
cordas	
Newton	

3. Idem $y = ((7 * x)/10) * e^{((8 * x)/10)} - 10.137427 = 0$ no intervalo entre 2 e 3 e responda a seguir os 3 valores encontrados

bissecção	
cordas	
Newton	

4. Idem $y = 5 * x^4 + 3 * x^2 - 183.168000 = 0$ no intervalo entre 2 e 3 e responda a seguir os 3 valores encontrados

bissecção	
cordas	
Newton	



Cálculo Numérico - solução de equações

Suponha que precisamos usar o computador para resolver equações. Já paramos para entender o que é resolver uma equação? Conhece-se a forma geral de uma equação $y = f(x)$. Resolver esta equação vem a ser descobrir quais os valores de x que fazem $f(x)$ ser igual a zero. Se olharmos para o gráfico, veremos que busca-se o valor da abscissa x que determina o cruzamento da curva de $f(x)$ no eixo dos x .

Por exemplo, suponha-se precisar descobrir a raiz 11 do número 1739? Ou seja, qual o número que multiplicado por ele mesmo 11 vezes dá como resultado 1739?

Dizendo isso na forma matemática, pode-se escrever $x^{11} = 1739$, ou melhor ainda $x^{11} - 1739 = 0$. Com isso, chegamos ao ponto.

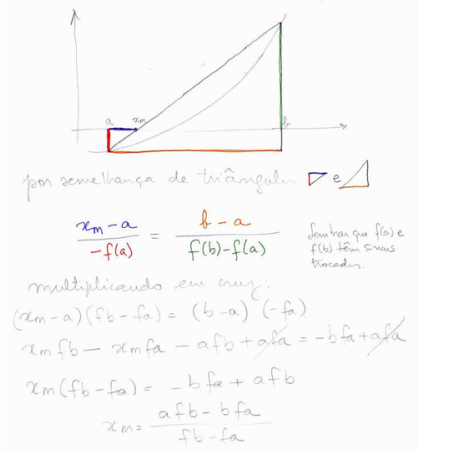
Método da Bissecção Se olharmos o gráfico de uma função, ver-se-á que no entorno da raiz, de um lado a função é positiva, e do outro lado é negativa ou vice versa. Este método começa pela seleção de 2 pontos x_a e x_b sendo que a função $y = f(x)$ têm sinais opostos nesses dois pontos. Graças a essa constatação, pode-se afirmar que a raiz procurada está entre x_a e x_b . Dizendo de outra maneira. O valor de x procurado, está entre a e b . O Método da bissecção vai dividindo o intervalo a, b pela metade, questionando a seguir em qual das metades o x está. Descoberto isso, despreza-se a metade que não contém x . Levando este processo ao limite, descobre-se qual o valor de x . Em resumo, eis as etapas do método:

1. Arbitrar intervalo em $[a, b]$ que compreenda a raiz.
2. Arbitrar um erro permitido ϵ
3. Calcular $x_m = (a + b)/2$
4. Se $f(a).f(x_m) < 0$ o novo intervalo será $[a, x_m]$
5. Se $f(a).f(x_m) > 0$ o novo intervalo será $[x_m, b]$
6. Enquanto o erro $b - a > \epsilon$ voltar ao passo 3.

Seja um exemplo, calcular $f = e^x - \sin(x) - 2 = 0$ nos limites $a = 1$ e $b = 1.25$ com $\epsilon = 0.01$. No primeira etapa tem-se

```
Limite Superior: 1.25
Limite Inferior: 1
Tolerancia: 0.01
-----a-----b-----xm-----fxm-----fa-----fb-----erro
1.1250 1.0000 1.1250 0.1779 0.5414 -0.1232 0.1250
1.0625 1.0000 1.0625 0.0200 0.1779 -0.1232 0.0625
1.0625 1.0312 1.0312 -0.0534 0.0200 -0.1232 0.0312
1.0625 1.0469 1.0469 -0.0171 0.0200 -0.0534 0.0156
1.0547 1.0469 1.0547 0.0013 0.0200 -0.0171 0.0078
Raiz: 1.05078125
Erro: -0.0078125
Eis o programa Python que resolve este caso
import numpy as np
def bissec():
    a=float(input("Limite Superior: "))
    b=float(input("Limite Inferior: "))
    tol=float(input("Tolerancia: "))
    print("-----a-----b-----xm-----fxm-----fa-----fb-----erro")
    while np.abs(b-a)>tol:
        xm=(a+b)/2
        fxm=np.exp(xm)-np.sin(xm)-2
        fa=np.exp(a)-np.sin(a)-2
        fb=np.exp(b)-np.sin(b)-2
        if fa*fxm>0:
            a=xm
        else:
            b=xm
        print('{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}'.format(a,b,xm,fxm,fa,fb,np.abs(b-a)))
    xm=(a+b)/2
    print("Raiz: ",xm)
    print("Erro: ",b-a)
bissec()
```

Cordas ou Falsa Posição



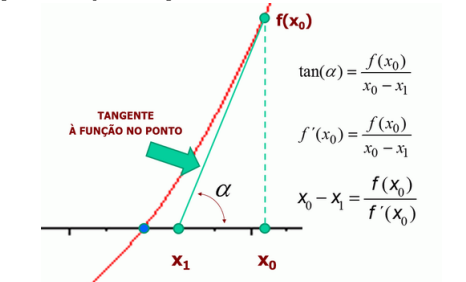
O método começa com uma análise do gráfico da função nas proximidades da raiz. Por semelhança de triângulos, pode-se obter a formulação proposta. Eis os passos do algoritmo:

1. Arbitrar um intervalo $[a, b]$ que compreende a raiz
2. Arbitrar ϵ
3. Calcular $x_m = \frac{a \cdot f(b) - b \cdot f(a)}{f(b) - f(a)}$
4. Se $f(a).f(x_m) < 0$ o novo intervalo será $[a, x_m]$
5. Se $f(a).f(x_m) > 0$ o novo intervalo será $[x_m, b]$
6. Enquanto o erro $f(x_m) > \epsilon$ voltar ao passo 3

Veja a seguir a solução da mesma equação, ($e^x - \sin(x) - 2 = 0$) com os mesmos intervalos do método anterior.

```
Limite inferior: 1
Limite superior: 1.25
Tolerância: 0.001
-----a-----b-----xm-----fxm-----fa-----fb-----erro
1.0463 1.2500 1.0463 -0.0184 -0.1232 0.5414
1.0530 1.2500 1.0530 -0.0026 -0.0184 0.5414
1.0540 1.2500 1.0540 -0.0004 -0.0026 0.5414
Raiz: 1.053972856764054
Erro: -0.0003663936835978099
Eis o programa Python que resolve este caso
from numpy import *
def cordas():
    a=float(input("Limite inferior: "))
    b=float(input("Limite superior: "))
    tol=float(input("Tolerância: "))
    fxm=1
    print("-----a-----b-----xm-----fxm-----fa-----fb-----erro")
    while abs(fxm)>tol:
        fa=exp(a)-sin(a)-2
        fb=exp(b)-sin(b)-2
        fxm=((a*fb)-(b*fa))/(fb-fa)
        fxm=exp(xm)-sin(xm)-2
        if fa*fxm>0:
            a=fxm
        else:
            b=fxm
        print('{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}'.format(a,b,xm,fxm,fa,fb))
    print("Raiz: ",xm)
    print("Erro: ",fxm)
cordas()
```

Método de Newton Este método começa com a interpretação geométrica da derivada. Lembrando, Newton é um monstro da ciência. Entre outras coisas, ele criou o cálculo diferencial e integral, a teoria da luz, a gravitação, as leis do movimento, além de, no fim da vida ter – como funcionário público – dirigido a casa da moeda inglesa, pendurando na forca alguns falsários delinquentes. Voltando ao cálculo diferencial, uma das interpretações da derivada de uma função em um ponto é a tangente da curva da função naquele ponto. Veja-se o que se disse neste desenho



- Eis as etapas do método:
1. Obter a derivada da função a resolver

2. Arbitrar um ponto inicial x_0 para a função
3. Arbitrar o erro ϵ
4. Calcular $f(x_0)$ e $df(x_0)$
5. Calcular o “novo” x , $x = x_0 - f(x)/df(x)$
6. Se $|f(x)| > \epsilon$ retornar ao passo 4

Por exemplo, seja calcular as raízes da mesma equação acima que é $e^x - \sin(x) - 2 = 0$. A derivada é $\frac{d}{dx} = e^x - \cos(x)$. Obteve-se a seguinte tabela

```
-----x-----fx-----dfx
1.0000000 -0.12318915634885141 2.1779795
1.0565612 0.00579321190120519 2.3845934
1.0541318 0.00001105001756940 2.3754999
1.0541271 0.00000000004045120 2.3754825
Raiz: 1.0541271241082415
Erro: 4.0451197946822504e-11
Eis o programa Python que resolve este caso
from numpy import *
def newton():
    x=1
    tol=0.00001
    fx=exp(x)-sin(x)-2
    dfx=exp(x)-cos(x)
    print("-----x-----fx-----dfx")
    print("{:12.7f}{:20.17f}{:12.7f}".format(x,fx,dfx))
    while abs(fx)>tol:
        x=x-fx/dfx
        fx=exp(x)-sin(x)-2
        dfx=exp(x)-cos(x)
        print("{:12.7f}{:20.17f}{:12.7f}".format(x,fx,dfx))
    print("Raiz: ",x)
    print("Erro: ",fx)
newton()
```

Para você fazer

1. Resolva pelos 3 métodos, para poder comparar, a seguinte equação $x = \sqrt[5]{8930}$ no intervalo entre 4 e 5 e responda a seguir os 3 valores encontrados

bissecção	
cordas	
Newton	

2. Idem $y = 8 \cdot \sin(x) + 5 \cdot \cos(x) + 6.927059 = 0$ no intervalo entre 4 e 5 e responda a seguir os 3 valores encontrados

bissecção	
cordas	
Newton	

3. Idem $y = ((6 \cdot x)/10) \cdot e^{((5 \cdot x)/10)} - 6.812736 = 0$ no intervalo entre 2 e 3 e responda a seguir os 3 valores encontrados

bissecção	
cordas	
Newton	

4. Idem $y = 7 \cdot x^5 + 4 \cdot x^3 - 313050.218750 = 0$ no intervalo entre 8 e 9 e responda a seguir os 3 valores encontrados

bissecção	
cordas	
Newton	



Cálculo Numérico - solução de equações

Suponha que precisamos usar o computador para resolver equações. Já paramos para entender o que é resolver uma equação? Conhece-se a forma geral de uma equação $y = f(x)$. Resolver esta equação vem a ser descobrir quais os valores de x que fazem $f(x)$ ser igual a zero. Se olharmos para o gráfico, veremos que busca-se o valor da abscissa x que determina o cruzamento da curva de $f(x)$ no eixo dos x .

Por exemplo, suponha-se precisar descobrir a raiz 11 do número 1739? Ou seja, qual o número que multiplicado por ele mesmo 11 vezes dá como resultado 1739?

Dizendo isso na forma matemática, pode-se escrever $x^{11} = 1739$, ou melhor ainda $x^{11} - 1739 = 0$. Com isso, chegamos ao ponto.

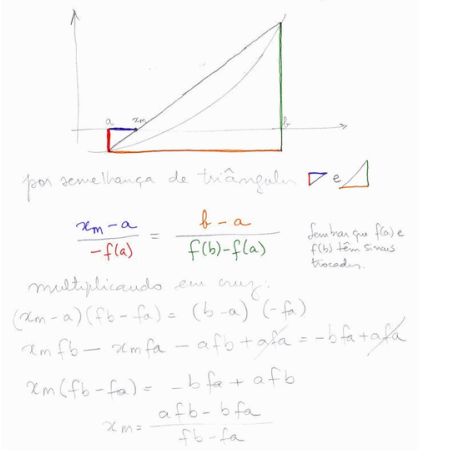
Método da Bissecção Se olharmos o gráfico de uma função, ver-se-á que no entorno da raiz, de um lado a função é positiva, e do outro lado é negativa ou vice versa. Este método começa pela seleção de 2 pontos x_a e x_b sendo que a função $y = f(x)$ têm sinais opostos nesses dois pontos. Graças a essa constatação, pode-se afirmar que a raiz procurada está entre x_a e x_b . Dizendo de outra maneira. O valor de x procurado, está entre a e b . O Método da bissecção vai dividindo o intervalo a, b pela metade, questionando a seguir em qual das metades o x está. Descoberto isso, despreza-se a metade que não contém x . Levando este processo ao limite, descobre-se qual o valor de x . Em resumo, eis as etapas do método:

1. Arbitrar intervalo em $[a, b]$ que compreenda a raiz.
2. Arbitrar um erro permitido ϵ
3. Calcular $x_m = (a + b)/2$
4. Se $f(a).f(x_m) < 0$ o novo intervalo será $[a, x_m]$
5. Se $f(a).f(x_m) > 0$ o novo intervalo será $[x_m, b]$
6. Enquanto o erro $b - a > \epsilon$ voltar ao passo 3.

Seja um exemplo, calcular $f = e^x - \sin(x) - 2 = 0$ nos limites $a = 1$ e $b = 1.25$ com $\epsilon = 0.01$. No primeira etapa tem-se

```
Limite Superior: 1.25
Limite Inferior: 1
Tolerancia: 0.01
-----a-----b-----xm-----fxm-----fa-----fb-----erro
1.1250 1.0000 1.1250 0.1779 0.5414 -0.1232 0.1250
1.0625 1.0000 1.0625 0.0200 0.1779 -0.1232 0.0625
1.0625 1.0312 1.0312 -0.0534 0.0200 -0.1232 0.0312
1.0625 1.0469 1.0469 -0.0171 0.0200 -0.0534 0.0156
1.0547 1.0469 1.0547 0.0013 0.0200 -0.0171 0.0078
Raiz: 1.05078125
Erro: -0.0078125
Eis o programa Python que resolve este caso
import numpy as np
def bissec():
    a=float(input("Limite Superior: "))
    b=float(input("Limite Inferior: "))
    tol=float(input("Tolerancia: "))
    print("-----a-----b-----xm-----fxm-----fa-----fb-----erro")
    while np.abs(b-a)>tol:
        xm=(a+b)/2
        fxm=np.exp(xm)-np.sin(xm)-2
        fa=np.exp(a)-np.sin(a)-2
        fb=np.exp(b)-np.sin(b)-2
        if fa*fxm>0:
            a=xm
        else:
            b=xm
        print('{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}'.format(a,b,xm,fxm,fa,fb,np.abs(b-a)))
    xm=(a+b)/2
    print("Raiz: ",xm)
    print("Erro: ",b-a)
bissec()
```

Cordas ou Falsa Posição



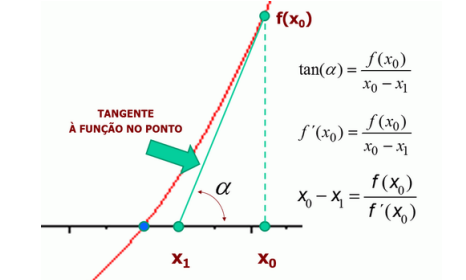
O método começa com uma análise do gráfico da função nas proximidades da raiz. Por semelhança de triângulos, pode-se obter a formulação proposta. Eis os passos do algoritmo:

1. Arbitrar um intervalo $[a, b]$ que compreende a raiz
2. Arbitrar ϵ
3. Calcular $x_m = \frac{a \cdot f(b) - b \cdot f(a)}{f(b) - f(a)}$
4. Se $f(a).f(x_m) < 0$ o novo intervalo será $[a, x_m]$
5. Se $f(a).f(x_m) > 0$ o novo intervalo será $[x_m, b]$
6. Enquanto o erro $f(x_m) > \epsilon$ voltar ao passo 3

Veja a seguir a solução da mesma equação, ($e^x - \sin(x) - 2 = 0$) com os mesmos intervalos do método anterior.

```
Limite inferior: 1
Limite superior: 1.25
Tolerância: 0.001
-----a-----b-----xm-----fxm-----fa-----fb-----erro
1.0463 1.2500 1.0463 -0.0184 -0.1232 0.5414
1.0530 1.2500 1.0530 -0.0026 -0.0184 0.5414
1.0540 1.2500 1.0540 -0.0004 -0.0026 0.5414
Raiz: 1.053972856764054
Erro: -0.0003663936835978099
Eis o programa Python que resolve este caso
from numpy import *
def cordas():
    a=float(input("Limite inferior: "))
    b=float(input("Limite superior: "))
    tol=float(input("Tolerância: "))
    fxm=1
    print("-----a-----b-----xm-----fxm-----fa-----fb-----erro")
    while abs(fxm)>tol:
        fa=exp(a)-sin(a)-2
        fb=exp(b)-sin(b)-2
        xm=((a*fb)-(b*fa))/(fb-fa)
        fxm=exp(xm)-sin(xm)-2
        if fa*fxm>0:
            a=xm
        else:
            b=xm
        print('{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}'.format(a,b,xm,fxm,fa,fb))
    print("Raiz: ",xm)
    print("Erro: ",fxm)
cordas()
```

Método de Newton Este método começa com a interpretação geométrica da derivada. Lembrando, Newton é um monstro da ciência. Entre outras coisas, ele criou o cálculo diferencial e integral, a teoria da luz, a gravitação, as leis do movimento, além de, no fim da vida ter – como funcionário público – dirigido a casa da moeda inglesa, pendurando na forca alguns falsários delinquentes. Voltando ao cálculo diferencial, uma das interpretações da derivada de uma função em um ponto é a tangente da curva da função naquele ponto. Veja-se o que se disse neste desenho



- Eis as etapas do método:
1. Obter a derivada da função a resolver

2. Arbitrar um ponto inicial x_0 para a função
3. Arbitrar o erro ϵ
4. Calcular $f(x_0)$ e $df(x_0)$
5. Calcular o “novo” x , $x = x_0 - f(x)/df(x)$
6. Se $|f(x)| > \epsilon$ retornar ao passo 4

Por exemplo, seja calcular as raízes da mesma equação acima que é $e^x - \sin(x) - 2 = 0$. A derivada é $\frac{d}{dx} = e^x - \cos(x)$. Obteve-se a seguinte tabela

```
-----x-----fx-----dfx
1.0000000 -0.12318915634885141 2.1779795
1.0565612 0.00579321190120519 2.3845934
1.0541318 0.00001105001756940 2.3754999
1.0541271 0.00000000004045120 2.3754825
Raiz: 1.0541271241082415
Erro: 4.0451197946822504e-11
Eis o programa Python que resolve este caso
from numpy import *
def newton1():
    x=1
    tol=0.00001
    fx=exp(x)-sin(x)-2
    dfx=exp(x)-cos(x)
    print("-----x-----fx-----dfx")
    print("{:12.7f}{:20.17f}{:12.7f}".format(x,fx,dfx))
    while abs(fx)>tol:
        x=x-fx/dfx
        fx=exp(x)-sin(x)-2
        dfx=exp(x)-cos(x)
        print("{:12.7f}{:20.17f}{:12.7f}".format(x,fx,dfx))
    print("Raiz: ",x)
    print("Erro: ",fx)
newton1()
```

Para você fazer

1. Resolva pelos 3 métodos, para poder comparar, a seguinte equação $x = \sqrt[11]{7363}$ no intervalo entre 2 e 3 e responda a seguir os 3 valores encontrados

bissecção	
cordas	
Newton	

2. Idem $y = 5 \sin(x) + 2 \cos(x) - 3.306355 = 0$ no intervalo entre 2 e 3 e responda a seguir os 3 valores encontrados

bissecção	
cordas	
Newton	

3. Idem $y = ((5 * x)/10) * e^{((4 * x)/10)} - 6.176645 = 0$ no intervalo entre 3 e 4 e responda a seguir os 3 valores encontrados

bissecção	
cordas	
Newton	

4. Idem $y = 6 * x^5 + 3 * x^2 - 299279.622420 = 0$ no intervalo entre 8 e 9 e responda a seguir os 3 valores encontrados

bissecção	
cordas	
Newton	



Cálculo Numérico - solução de equações

Suponha que precisamos usar o computador para resolver equações. Já paramos para entender o que é resolver uma equação? Conhece-se a forma geral de uma equação $y = f(x)$. Resolver esta equação vem a ser descobrir quais os valores de x que fazem $f(x)$ ser igual a zero. Se olharmos para o gráfico, veremos que busca-se o valor da abscissa x que determina o cruzamento da curva de $f(x)$ no eixo dos x .

Por exemplo, suponha-se precisar descobrir a raiz 11 do número 1739? Ou seja, qual o número que multiplicado por ele mesmo 11 vezes dá como resultado 1739?

Dizendo isso na forma matemática, pode-se escrever $x^{11} = 1739$, ou melhor ainda $x^{11} - 1739 = 0$. Com isso, chegamos ao ponto.

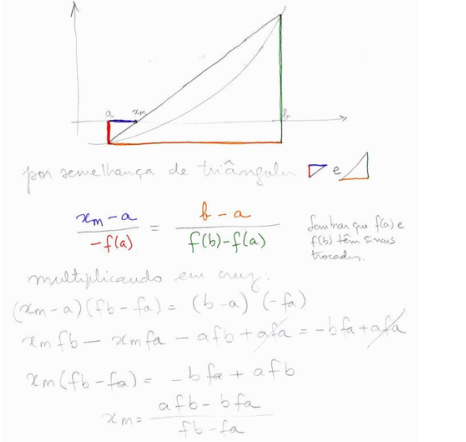
Método da Bissecção Se olharmos o gráfico de uma função, ver-se-á que no entorno da raiz, de um lado a função é positiva, e do outro lado é negativa ou vice versa. Este método começa pela seleção de 2 pontos x_a e x_b sendo que a função $y = f(x)$ têm sinais opostos nesses dois pontos. Graças a essa constatação, pode-se afirmar que a raiz procurada está entre x_a e x_b . Dizendo de outra maneira. O valor de x procurado, está entre a e b . O Método da bissecção vai dividindo o intervalo a, b pela metade, questionando a seguir em qual das metades o x está. Descoberto isso, despreza-se a metade que não contém x . Levando este processo ao limite, descobre-se qual o valor de x . Em resumo, eis as etapas do método:

1. Arbitrar intervalo em $[a, b]$ que compreenda a raiz.
2. Arbitrar um erro permitido ϵ
3. Calcular $x_m = (a + b)/2$
4. Se $f(a).f(x_m) < 0$ o novo intervalo será $[a, x_m]$
5. Se $f(a).f(x_m) > 0$ o novo intervalo será $[x_m, b]$
6. Enquanto o erro $b - a > \epsilon$ voltar ao passo 3.

Seja um exemplo, calcular $f = e^x - \sin(x) - 2 = 0$ nos limites $a = 1$ e $b = 1.25$ com $\epsilon = 0.01$. No primeira etapa tem-se

```
Limite Superior: 1.25
Limite Inferior: 1
Tolerancia: 0.01
-----a-----b-----xm-----fxm-----fa-----fb-----erro
1.1250 1.0000 1.1250 0.1779 0.5414 -0.1232 0.1250
1.0625 1.0000 1.0625 0.0200 0.1779 -0.1232 0.0625
1.0625 1.0312 1.0312 -0.0534 0.0200 -0.1232 0.0312
1.0625 1.0469 1.0469 -0.0171 0.0200 -0.0534 0.0156
1.0547 1.0469 1.0547 0.0013 0.0200 -0.0171 0.0078
Raiz: 1.05078125
Erro: -0.0078125
Eis o programa Python que resolve este caso
import numpy as np
def bissec():
    a=float(input("Limite Superior: "))
    b=float(input("Limite Inferior: "))
    tol=float(input("Tolerancia: "))
    print("-----a-----b-----xm-----fxm-----fa-----fb-----erro")
    while np.abs(b-a)>tol:
        xm=(a+b)/2
        fxm=np.exp(xm)-np.sin(xm)-2
        fa=np.exp(a)-np.sin(a)-2
        fb=np.exp(b)-np.sin(b)-2
        if fa*fxm>0:
            a=xm
        else:
            b=xm
        print('{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}'
              .format(a,b,xm,fxm,fa,fb,np.abs(b-a)))
    xm=(a+b)/2
    print("Raiz: ",xm)
    print("Erro: ",b-a)
bissec()
```

Cordas ou Falsa Posição



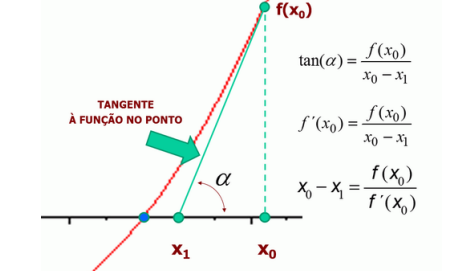
O método começa com uma análise do gráfico da função nas proximidades da raiz. Por semelhança de triângulos, pode-se obter a formulação proposta. Eis os passos do algoritmo:

1. Arbitrar um intervalo $[a, b]$ que compreende a raiz
2. Arbitrar ϵ
3. Calcular $x_m = \frac{a.f(b)-b.f(a)}{f(b)-f(a)}$
4. Se $f(a).f(x_m) < 0$ o novo intervalo será $[a, x_m]$
5. Se $f(a).f(x_m) > 0$ o novo intervalo será $[x_m, b]$
6. Enquanto o erro $f(x_m) > \epsilon$ voltar ao passo 3

Veja a seguir a solução da mesma equação, ($e^x - \sin(x) - 2 = 0$) com os mesmos intervalos do método anterior.

```
Limite inferior: 1
Limite superior: 1.25
Tolerancia: 0.001
-----a-----b-----xm-----fxm-----fa-----fb-----erro
1.0463 1.2500 1.0463 -0.0184 -0.1232 0.5414
1.0530 1.2500 1.0530 -0.0026 -0.0184 0.5414
1.0540 1.2500 1.0540 -0.0004 -0.0026 0.5414
Raiz: 1.053972856764054
Erro: -0.0003663936835978099
Eis o programa Python que resolve este caso
from numpy import *
def cordas():
    a=float(input("Limite inferior: "))
    b=float(input("Limite superior: "))
    tol=float(input("Tolerancia: "))
    fxm=1
    print("-----a-----b-----xm-----fxm-----fa-----fb-----erro")
    while abs(fxm)>tol:
        fa=exp(a)-sin(a)-2
        fb=exp(b)-sin(b)-2
        fxm=((a*fb)-(b*fa))/(fb-fa)
        fxm=exp(xm)-sin(xm)-2
        if fa*fxm>0:
            a=fxm
        else:
            b=fxm
        print('{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}'
              .format(a,b,xm,fxm,fa,fb))
    print("Raiz: ",xm)
    print("Erro: ",fxm)
cordas()
```

Método de Newton Este método começa com a interpretação geométrica da derivada. Lembrando, Newton é um monstro da ciência. Entre outras coisas, ele criou o cálculo diferencial e integral, a teoria da luz, a gravitação, as leis do movimento, além de, no fim da vida ter – como funcionário público – dirigido a casa da moeda inglesa, pendurando na forca alguns falsários delinquentes. Voltando ao cálculo diferencial, uma das interpretações da derivada de uma função em um ponto é a tangente da curva da função naquele ponto. Veja-se o que se disse neste desenho



Eis as etapas do método:

1. Obter a derivada da função a resolver

2. Arbitrar um ponto inicial x_0 para a função
3. Arbitrar o erro ϵ
4. Calcular $f(x_0)$ e $df(x_0)$
5. Calcular o “novo” x , $x = x_0 - f(x)/df(x)$
6. Se $|f(x)| > \epsilon$ retornar ao passo 4

Por exemplo, seja calcular as raízes da mesma equação acima que é $e^x - \sin(x) - 2 = 0$. A derivada é $\frac{d}{dx} = e^x - \cos(x)$. Obteve-se a seguinte tabela

```
-----x-----fx-----dfx
1.0000000 -0.12318915634885141 2.1779795
1.0565612 0.00579321190120519 2.3845934
1.0541318 0.00001105001756940 2.3754999
1.0541271 0.00000000004045120 2.3754825
Raiz: 1.0541271241082415
Erro: 4.0451197946822504e-11
Eis o programa Python que resolve este caso
from numpy import *
def newton1():
    x=1
    tol=0.00001
    fx=exp(x)-sin(x)-2
    dfx=exp(x)-cos(x)
    print("-----x-----fx-----dfx")
    print("{:12.7f}{:20.17f}{:12.7f}".format(x,fx,dfx))
    while abs(fx)>tol:
        x=x-fx/dfx
        fx=exp(x)-sin(x)-2
        dfx=exp(x)-cos(x)
        print("{:12.7f}{:20.17f}{:12.7f}".format(x,fx,dfx))
    print("Raiz: ",x)
    print("Erro: ",fx)
newton1()
```

Para você fazer

1. Resolva pelos 3 métodos, para poder comparar, a seguinte equação $x = \sqrt[18]{7879}$ no intervalo entre 1 e 2 e responda a seguir os 3 valores encontrados

bissecção	
cordas	
Newton	

2. Idem $y = 3*\sin(x) + 8*\cos(x) + 7.049917 = 0$ no intervalo entre 2 e 3 e responda a seguir os 3 valores encontrados

bissecção	
cordas	
Newton	

3. Idem $y = ((7 * x)/10) * e^{((4*x)/10)} - .427491 = 0$ no intervalo entre 0 e 1 e responda a seguir os 3 valores encontrados

bissecção	
cordas	
Newton	

4. Idem $y = 7 * x^5 + 4 * x^2 - 101960.309760 = 0$ no intervalo entre 6 e 7 e responda a seguir os 3 valores encontrados

bissecção	
cordas	
Newton	



Cálculo Numérico - solução de equações

Suponha que precisamos usar o computador para resolver equações. Já paramos para entender o que é resolver uma equação? Conhece-se a forma geral de uma equação $y = f(x)$. Resolver esta equação vem a ser descobrir quais os valores de x que fazem $f(x)$ ser igual a zero. Se olharmos para o gráfico, veremos que busca-se o valor da abscissa x que determina o cruzamento da curva de $f(x)$ no eixo dos x .

Por exemplo, suponha-se precisar descobrir a raiz 11 do número 1739? Ou seja, qual o número que multiplicado por ele mesmo 11 vezes dá como resultado 1739?

Dizendo isso na forma matemática, pode-se escrever $x^{11} = 1739$, ou melhor ainda $x^{11} - 1739 = 0$. Com isso, chegamos ao ponto.

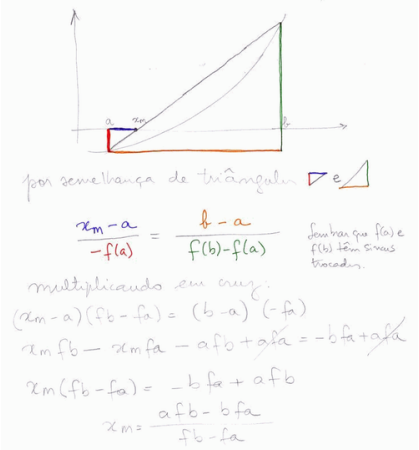
Método da Bissecção Se olharmos o gráfico de uma função, ver-se-á que no entorno da raiz, de um lado a função é positiva, e do outro lado é negativa ou vice versa. Este método começa pela seleção de 2 pontos x_a e x_b sendo que a função $y = f(x)$ têm sinais opostos nesses dois pontos. Graças a essa constatação, pode-se afirmar que a raiz procurada está entre x_a e x_b . Dizendo de outra maneira. O valor de x procurado, está entre a e b . O Método da bissecção vai dividindo o intervalo a, b pela metade, questionando a seguir em qual das metades o x está. Descoberto isso, despreza-se a metade que não contém x . Levando este processo ao limite, descobre-se qual o valor de x . Em resumo, eis as etapas do método:

1. Arbitrar intervalo em $[a, b]$ que compreenda a raiz.
2. Arbitrar um erro permitido ϵ
3. Calcular $x_m = (a + b)/2$
4. Se $f(a).f(x_m) < 0$ o novo intervalo será $[a, x_m]$
5. Se $f(a).f(x_m) > 0$ o novo intervalo será $[x_m, b]$
6. Enquanto o erro $b - a > \epsilon$ voltar ao passo 3.

Seja um exemplo, calcular $f = e^x - \sin(x) - 2 = 0$ nos limites $a = 1$ e $b = 1.25$ com $\epsilon = 0.01$. No primeira etapa tem-se

```
Limite Superior: 1.25
Limite Inferior: 1
Tolerancia: 0.01
-----a-----b-----xm-----fxm-----fa-----fb-----erro
1.1250 1.0000 1.1250 0.1779 0.5414 -0.1232 0.1250
1.0625 1.0000 1.0625 0.0200 0.1779 -0.1232 0.0625
1.0625 1.0312 1.0312 -0.0534 0.0200 -0.1232 0.0312
1.0625 1.0469 1.0469 -0.0171 0.0200 -0.0534 0.0156
1.0547 1.0469 1.0547 0.0013 0.0200 -0.0171 0.0078
Raiz: 1.05078125
Erro: -0.0078125
Eis o programa Python que resolve este caso
import numpy as np
def bissec():
    a=float(input("Limite Superior: "))
    b=float(input("Limite Inferior: "))
    tol=float(input("Tolerancia: "))
    print("-----a-----b-----xm-----fxm-----fa-----fb-----erro")
    while np.abs(b-a)>tol:
        xm=(a+b)/2
        fxm=np.exp(xm)-np.sin(xm)-2
        fa=np.exp(a)-np.sin(a)-2
        fb=np.exp(b)-np.sin(b)-2
        if fa*fxm>0:
            a=xm
        else:
            b=xm
        print('{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}'.format(a,b,xm,fxm,fa,fb,np.abs(b-a)))
    xm=(a+b)/2
    print("Raiz: ",xm)
    print("Erro: ",b-a)
bissec()
```

Cordas ou Falsa Posição



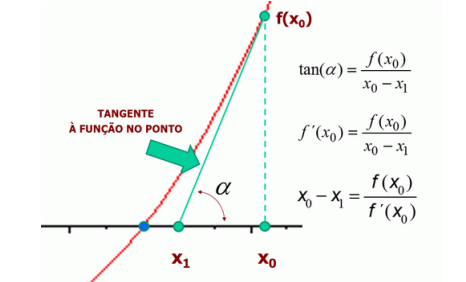
O método começa com uma análise do gráfico da função nas proximidades da raiz. Por semelhança de triângulos, pode-se obter a formulação proposta. Eis os passos do algoritmo:

1. Arbitrar um intervalo $[a, b]$ que compreende a raiz
2. Arbitrar ϵ
3. Calcular $x_m = \frac{a \cdot f(b) - b \cdot f(a)}{f(b) - f(a)}$
4. Se $f(a).f(x_m) < 0$ o novo intervalo será $[a, x_m]$
5. Se $f(a).f(x_m) > 0$ o novo intervalo será $[x_m, b]$
6. Enquanto o erro $f(x_m) > \epsilon$ voltar ao passo 3

Veja a seguir a solução da mesma equação, ($e^x - \sin(x) - 2 = 0$) com os mesmos intervalos do método anterior.

```
Limite inferior: 1
Limite superior: 1.25
Tolerância: 0.001
-----a-----b-----xm-----fxm-----fa-----fb-----erro
1.0463 1.2500 1.0463 -0.0184 -0.1232 0.5414
1.0530 1.2500 1.0530 -0.0026 -0.0184 0.5414
1.0540 1.2500 1.0540 -0.0004 -0.0026 0.5414
Raiz: 1.053972856764054
Erro: -0.0003663936835978099
Eis o programa Python que resolve este caso
from numpy import *
def cordas():
    a=float(input("Limite inferior: "))
    b=float(input("Limite superior: "))
    tol=float(input("Tolerância: "))
    fxm=1
    print("-----a-----b-----xm-----fxm-----fa-----fb-----erro")
    while abs(fxm)>tol:
        fa=exp(a)-sin(a)-2
        fb=exp(b)-sin(b)-2
        xm=((a*fb)-(b*fa))/(fb-fa)
        fxm=exp(xm)-sin(xm)-2
        if fa*fxm>0:
            a=xm
        else:
            b=xm
        print('{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}'.format(a,b,xm,fxm,fa,fb))
    print("Raiz: ",xm)
    print("Erro: ",fxm)
cordas()
```

Método de Newton Este método começa com a interpretação geométrica da derivada. Lembrando, Newton é um monstro da ciência. Entre outras coisas, ele criou o cálculo diferencial e integral, a teoria da luz, a gravitação, as leis do movimento, além de, no fim da vida ter – como funcionário público – dirigido a casa da moeda inglesa, pendurando na força alguns falsários delinquentes. Voltando ao cálculo diferencial, uma das interpretações da derivada de uma função em um ponto é a tangente da curva da função naquele ponto. Veja-se o que se disse neste desenho



Eis as etapas do método:

1. Obter a derivada da função a resolver

2. Arbitrar um ponto inicial x_0 para a função
3. Arbitrar o erro ϵ
4. Calcular $f(x_0)$ e $df(x_0)$
5. Calcular o “novo” x , $x = x_0 - f(x)/df(x)$
6. Se $|f(x)| > \epsilon$ retornar ao passo 4

Por exemplo, seja calcular as raízes da mesma equação acima que é $e^x - \sin(x) - 2 = 0$. A derivada é $\frac{d}{dx} = e^x - \cos(x)$. Obteve-se a seguinte tabela

```
-----x-----fx-----dfx
1.0000000 -0.12318915634885141 2.1779795
1.0565612 0.00579321190120519 2.3845934
1.0541318 0.00001105001756940 2.3754999
1.0541271 0.00000000004045120 2.3754825
Raiz: 1.0541271241082415
Erro: 4.0451197946822504e-11
Eis o programa Python que resolve este caso
from numpy import *
def newton():
    x=1
    tol=0.00001
    fx=exp(x)-sin(x)-2
    dfx=exp(x)-cos(x)
    print("-----x-----fx-----dfx")
    print("{:12.7f}{:20.17f}{:12.7f}".format(x,fx,dfx))
    while abs(fx)>tol:
        x=x-fx/dfx
        fx=exp(x)-sin(x)-2
        dfx=exp(x)-cos(x)
        print("{:12.7f}{:20.17f}{:12.7f}".format(x,fx,dfx))
    print("Raiz: ",x)
    print("Erro: ",fx)
newton()
```

Para você fazer

1. Resolva pelos 3 métodos, para poder comparar, a seguinte equação $x = \sqrt[18]{8426}$ no intervalo entre 1 e 2 e responda a seguir os 3 valores encontrados

bissecção	
cordas	
Newton	

2. Idem $y = 6 \cdot \sin(x) + 5 \cdot \cos(x) - 5.665220 = 0$ no intervalo entre 6 e 7 e responda a seguir os 3 valores encontrados

bissecção	
cordas	
Newton	

3. Idem $y = ((7 \cdot x)/10) \cdot e^{((5 \cdot x)/10)} - 5.577796 = 0$ no intervalo entre 2 e 3 e responda a seguir os 3 valores encontrados

bissecção	
cordas	
Newton	

4. Idem $y = 6 \cdot x^4 + 3 \cdot x^2 - 1.29600 = 0$ no intervalo entre 0 e 1 e responda a seguir os 3 valores encontrados

bissecção	
cordas	
Newton	



Cálculo Numérico - solução de equações

Suponha que precisamos usar o computador para resolver equações. Já paramos para entender o que é resolver uma equação? Conhece-se a forma geral de uma equação $y = f(x)$. Resolver esta equação vem a ser descobrir quais os valores de x que fazem $f(x)$ ser igual a zero. Se olharmos para o gráfico, veremos que busca-se o valor da abscissa x que determina o cruzamento da curva de $f(x)$ no eixo dos x .

Por exemplo, suponha-se precisar descobrir a raiz 11 do número 1739? Ou seja, qual o número que multiplicado por ele mesmo 11 vezes dá como resultado 1739?

Dizendo isso na forma matemática, pode-se escrever $x^{11} = 1739$, ou melhor ainda $x^{11} - 1739 = 0$. Com isso, chegamos ao ponto.

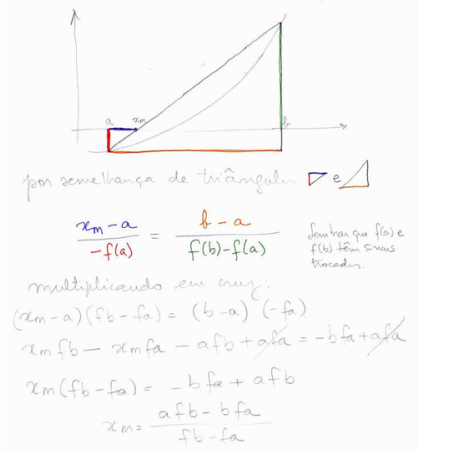
Método da Bissecção Se olharmos o gráfico de uma função, ver-se-á que no entorno da raiz, de um lado a função é positiva, e do outro lado é negativa ou vice versa. Este método começa pela seleção de 2 pontos x_a e x_b sendo que a função $y = f(x)$ têm sinais opostos nesses dois pontos. Graças a essa constatação, pode-se afirmar que a raiz procurada está entre x_a e x_b . Dizendo de outra maneira. O valor de x procurado, está entre a e b . O Método da bissecção vai dividindo o intervalo a, b pela metade, questionando a seguir em qual das metades o x está. Descoberto isso, despreza-se a metade que não contém x . Levando este processo ao limite, descobre-se qual o valor de x . Em resumo, eis as etapas do método:

1. Arbitrar intervalo em $[a, b]$ que compreenda a raiz.
2. Arbitrar um erro permitido ϵ
3. Calcular $x_m = (a + b)/2$
4. Se $f(a).f(x_m) < 0$ o novo intervalo será $[a, x_m]$
5. Se $f(a).f(x_m) > 0$ o novo intervalo será $[x_m, b]$
6. Enquanto o erro $b - a > \epsilon$ voltar ao passo 3.

Seja um exemplo, calcular $f = e^x - \sin(x) - 2 = 0$ nos limites $a = 1$ e $b = 1.25$ com $\epsilon = 0.01$. No primeira etapa tem-se

```
Limite Superior: 1.25
Limite Inferior: 1
Tolerancia: 0.01
-----a-----b-----xm-----fxm-----fa-----fb-----erro
1.1250 1.0000 1.1250 0.1779 0.5414 -0.1232 0.1250
1.0625 1.0000 1.0625 0.0200 0.1779 -0.1232 0.0625
1.0625 1.0312 1.0312 -0.0534 0.0200 -0.1232 0.0312
1.0625 1.0469 1.0469 -0.0171 0.0200 -0.0534 0.0156
1.0547 1.0469 1.0547 0.0013 0.0200 -0.0171 0.0078
Raiz: 1.05078125
Erro: -0.0078125
Eis o programa Python que resolve este caso
import numpy as np
def bissec():
    a=float(input("Limite Superior: "))
    b=float(input("Limite Inferior: "))
    tol=float(input("Tolerancia: "))
    print("-----a-----b-----xm-----fxm-----fa-----fb-----erro")
    while np.abs(b-a)>tol:
        xm=(a+b)/2
        fxm=np.exp(xm)-np.sin(xm)-2
        fa=np.exp(a)-np.sin(a)-2
        fb=np.exp(b)-np.sin(b)-2
        if fa*fxm>0:
            a=xm
        else:
            b=xm
        print('{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}'
              .format(a,b,xm,fxm,fa,fb,np.abs(b-a)))
    xm=(a+b)/2
    print("Raiz: ",xm)
    print("Erro: ",b-a)
bissec()
```

Cordas ou Falsa Posição



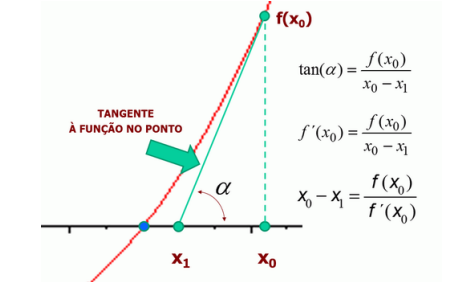
O método começa com uma análise do gráfico da função nas proximidades da raiz. Por semelhança de triângulos, pode-se obter a formulação proposta. Eis os passos do algoritmo:

1. Arbitrar um intervalo $[a, b]$ que compreende a raiz
2. Arbitrar ϵ
3. Calcular $x_m = \frac{a.f(b)-b.f(a)}{f(b)-f(a)}$
4. Se $f(a).f(x_m) < 0$ o novo intervalo será $[a, x_m]$
5. Se $f(a).f(x_m) > 0$ o novo intervalo será $[x_m, b]$
6. Enquanto o erro $f(x_m) > \epsilon$ voltar ao passo 3

Veja a seguir a solução da mesma equação, ($e^x - \sin(x) - 2 = 0$) com os mesmos intervalos do método anterior.

```
Limite inferior: 1
Limite superior: 1.25
Tolerância: 0.001
-----a-----b-----xm-----fxm-----fa-----fb-----erro
1.0463 1.2500 1.0463 -0.0184 -0.1232 0.5414
1.0530 1.2500 1.0530 -0.0026 -0.0184 0.5414
1.0540 1.2500 1.0540 -0.0004 -0.0026 0.5414
Raiz: 1.053972856764054
Erro: -0.0003663936835978099
Eis o programa Python que resolve este caso
from numpy import *
def cordas():
    a=float(input("Limite inferior: "))
    b=float(input("Limite superior: "))
    tol=float(input("Tolerância: "))
    fxm=1
    print("-----a-----b-----xm-----fxm-----fa-----fb-----erro")
    while abs(fxm)>tol:
        fa=exp(a)-sin(a)-2
        fb=exp(b)-sin(b)-2
        xm=((a*fb)-(b*fa))/(fb-fa)
        fxm=exp(xm)-sin(xm)-2
        if fa*fxm>0:
            a=xm
        else:
            b=xm
        print('{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}'
              .format(a,b,xm,fxm,fa,fb))
    print("Raiz: ",xm)
    print("Erro: ",fxm)
cordas()
```

Método de Newton Este método começa com a interpretação geométrica da derivada. Lembrando, Newton é um monstro da ciência. Entre outras coisas, ele criou o cálculo diferencial e integral, a teoria da luz, a gravitação, as leis do movimento, além de, no fim da vida ter – como funcionário público – dirigido a casa da moeda inglesa, pendurando na forca alguns falsários delinquentes. Voltando ao cálculo diferencial, uma das interpretações da derivada de uma função em um ponto é a tangente da curva da função naquele ponto. Veja-se o que se disse neste desenho



Eis as etapas do método:

1. Obter a derivada da função a resolver

2. Arbitrar um ponto inicial x_0 para a função
3. Arbitrar o erro ϵ
4. Calcular $f(x_0)$ e $df(x_0)$
5. Calcular o “novo” x , $x = x_0 - f(x)/df(x)$
6. Se $|f(x)| > \epsilon$ retornar ao passo 4

Por exemplo, seja calcular as raízes da mesma equação acima que é $e^x - \sin(x) - 2 = 0$. A derivada é $\frac{d}{dx} = e^x - \cos(x)$. Obteve-se a seguinte tabela

```
-----x-----fx-----dfx
1.0000000 -0.12318915634885141 2.1779795
1.0565612 0.00579321190120519 2.3845934
1.0541318 0.00001105001756940 2.3754999
1.0541271 0.00000000004045120 2.3754825
Raiz: 1.0541271241082415
Erro: 4.0451197946822504e-11
Eis o programa Python que resolve este caso
from numpy import *
def newton1():
    x=1
    tol=0.00001
    fx=exp(x)-sin(x)-2
    dfx=exp(x)-cos(x)
    print("-----x-----fx-----dfx")
    print("{:12.7f}{:20.17f}{:12.7f}".format(x,fx,dfx))
    while abs(fx)>tol:
        x=x-fx/dfx
        fx=exp(x)-sin(x)-2
        dfx=exp(x)-cos(x)
        print("{:12.7f}{:20.17f}{:12.7f}".format(x,fx,dfx))
    print("Raiz: ",x)
    print("Erro: ",fx)
newton1()
```

Para você fazer

1. Resolva pelos 3 métodos, para poder comparar, a seguinte equação $x = \sqrt[2]{6821}$ no intervalo entre 1 e 2 e responda a seguir os 3 valores encontrados

bissecção	
cordas	
Newton	

2. Idem $y = 8*\sin(x) + 4*\cos(x) - 3.160297 = 0$ no intervalo entre 8 e 9 e responda a seguir os 3 valores encontrados

bissecção	
cordas	
Newton	

3. Idem $y = ((8 * x)/10) * e^{((5*x)/10)} - 34.155849 = 0$ no intervalo entre 4 e 5 e responda a seguir os 3 valores encontrados

bissecção	
cordas	
Newton	

4. Idem $y = 7 * x^5 + 4 * x^2 - 17928.437760 = 0$ no intervalo entre 4 e 5 e responda a seguir os 3 valores encontrados

bissecção	
cordas	
Newton	



Cálculo Numérico - solução de equações

Suponha que precisamos usar o computador para resolver equações. Já paramos para entender o que é resolver uma equação? Conhece-se a forma geral de uma equação $y = f(x)$. Resolver esta equação vem a ser descobrir quais os valores de x que fazem $f(x)$ ser igual a zero. Se olharmos para o gráfico, veremos que busca-se o valor da abscissa x que determina o cruzamento da curva de $f(x)$ no eixo dos x .

Por exemplo, suponha-se precisar descobrir a raiz 11 do número 1739? Ou seja, qual o número que multiplicado por ele mesmo 11 vezes dá como resultado 1739?

Dizendo isso na forma matemática, pode-se escrever $x^{11} = 1739$, ou melhor ainda $x^{11} - 1739 = 0$. Com isso, chegamos ao ponto.

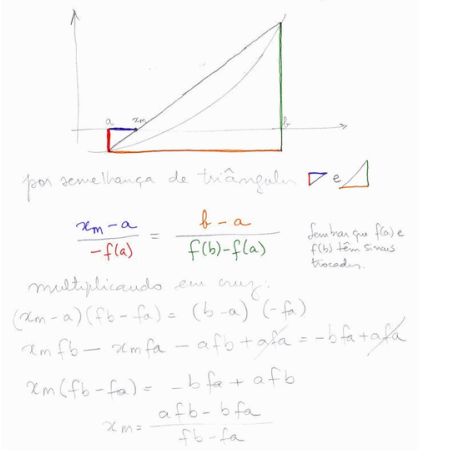
Método da Bissecção Se olharmos o gráfico de uma função, ver-se-á que no entorno da raiz, de um lado a função é positiva, e do outro lado é negativa ou vice versa. Este método começa pela seleção de 2 pontos x_a e x_b sendo que a função $y = f(x)$ têm sinais opostos nesses dois pontos. Graças a essa constatação, pode-se afirmar que a raiz procurada está entre x_a e x_b . Dizendo de outra maneira. O valor de x procurado, está entre a e b . O Método da bissecção vai dividindo o intervalo a, b pela metade, questionando a seguir em qual das metades o x está. Descoberto isso, despreza-se a metade que não contém x . Levando este processo ao limite, descobre-se qual o valor de x . Em resumo, eis as etapas do método:

1. Arbitrar intervalo em $[a, b]$ que compreenda a raiz.
2. Arbitrar um erro permitido ϵ
3. Calcular $x_m = (a + b)/2$
4. Se $f(a).f(x_m) < 0$ o novo intervalo será $[a, x_m]$
5. Se $f(a).f(x_m) > 0$ o novo intervalo será $[x_m, b]$
6. Enquanto o erro $b - a > \epsilon$ voltar ao passo 3.

Seja um exemplo, calcular $f = e^x - \sin(x) - 2 = 0$ nos limites $a = 1$ e $b = 1.25$ com $\epsilon = 0.01$. No primeira etapa tem-se

```
Limite Superior: 1.25
Limite Inferior: 1
Tolerancia: 0.01
-----a-----b-----xm-----fxm-----fa-----fb-----erro
1.1250 1.0000 1.1250 0.1779 0.5414 -0.1232 0.1250
1.0625 1.0000 1.0625 0.0200 0.1779 -0.1232 0.0625
1.0625 1.0312 1.0312 -0.0534 0.0200 -0.1232 0.0312
1.0625 1.0469 1.0469 -0.0171 0.0200 -0.0534 0.0156
1.0547 1.0469 1.0547 0.0013 0.0200 -0.0171 0.0078
Raiz: 1.05078125
Erro: -0.0078125
Eis o programa Python que resolve este caso
import numpy as np
def bissec():
    a=float(input("Limite Superior: "))
    b=float(input("Limite Inferior: "))
    tol=float(input("Tolerancia: "))
    print("-----a-----b-----xm-----fxm-----fa-----fb-----erro")
    while np.abs(b-a)>tol:
        xm=(a+b)/2
        fxm=np.exp(xm)-np.sin(xm)-2
        fa=np.exp(a)-np.sin(a)-2
        fb=np.exp(b)-np.sin(b)-2
        if fa*fxm>0:
            a=xm
        else:
            b=xm
        print('{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}'
              .format(a,b,xm,fxm,fa,fb,np.abs(b-a)))
    xm=(a+b)/2
    print("Raiz: ",xm)
    print("Erro: ",b-a)
bissec()
```

Cordas ou Falsa Posição



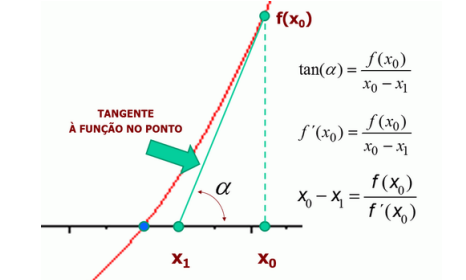
O método começa com uma análise do gráfico da função nas proximidades da raiz. Por semelhança de triângulos, pode-se obter a formulação proposta. Eis os passos do algoritmo:

1. Arbitrar um intervalo $[a, b]$ que compreende a raiz
2. Arbitrar ϵ
3. Calcular $x_m = \frac{a \cdot f(b) - b \cdot f(a)}{f(b) - f(a)}$
4. Se $f(a).f(x_m) < 0$ o novo intervalo será $[a, x_m]$
5. Se $f(a).f(x_m) > 0$ o novo intervalo será $[x_m, b]$
6. Enquanto o erro $f(x_m) > \epsilon$ voltar ao passo 3

Veja a seguir a solução da mesma equação, ($e^x - \sin(x) - 2 = 0$) com os mesmos intervalos do método anterior.

```
Limite inferior: 1
Limite superior: 1.25
Tolerância: 0.001
-----a-----b-----xm-----fxm-----fa-----fb-----erro
1.0463 1.2500 1.0463 -0.0184 -0.1232 0.5414
1.0530 1.2500 1.0530 -0.0026 -0.0184 0.5414
1.0540 1.2500 1.0540 -0.0004 -0.0026 0.5414
Raiz: 1.053972856764054
Erro: -0.0003663936835978099
Eis o programa Python que resolve este caso
from numpy import *
def cordas():
    a=float(input("Limite inferior: "))
    b=float(input("Limite superior: "))
    tol=float(input("Tolerância: "))
    fxm=1
    print("-----a-----b-----xm-----fxm-----fa-----fb-----erro")
    while abs(fxm)>tol:
        fa=exp(a)-sin(a)-2
        fb=exp(b)-sin(b)-2
        fxm=((a*fb)-(b*fa))/(fb-fa)
        fxm=exp(xm)-sin(xm)-2
        if fa*fxm>0:
            a=fxm
        else:
            b=fxm
        print('{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}'
              .format(a,b,xm,fxm,fa,fb))
    print("Raiz: ",xm)
    print("Erro: ",fxm)
cordas()
```

Método de Newton Este método começa com a interpretação geométrica da derivada. Lembrando, Newton é um monstro da ciência. Entre outras coisas, ele criou o cálculo diferencial e integral, a teoria da luz, a gravitação, as leis do movimento, além de, no fim da vida ter – como funcionário público – dirigido a casa da moeda inglesa, pendurando na força alguns falsários delinquentes. Voltando ao cálculo diferencial, uma das interpretações da derivada de uma função em um ponto é a tangente da curva da função naquele ponto. Veja-se o que se disse neste desenho



- Eis as etapas do método:
1. Obter a derivada da função a resolver

2. Arbitrar um ponto inicial x_0 para a função
3. Arbitrar o erro ϵ
4. Calcular $f(x_0)$ e $df(x_0)$
5. Calcular o “novo” x , $x = x_0 - f(x)/df(x)$
6. Se $|f(x)| > \epsilon$ retornar ao passo 4

Por exemplo, seja calcular as raízes da mesma equação acima que é $e^x - \sin(x) - 2 = 0$. A derivada é $\frac{\partial}{\partial x} = e^x - \cos(x)$. Obteve-se a seguinte tabela

```
-----x-----fx-----dfx
1.0000000 -0.12318915634885141 2.1779795
1.0565612 0.00579321190120519 2.3845934
1.0541318 0.00001105001756940 2.3754999
1.0541271 0.00000000004045120 2.3754825
Raiz: 1.0541271241082415
Erro: 4.0451197946822504e-11
Eis o programa Python que resolve este caso
from numpy import *
def newton1():
    x=1
    tol=0.00001
    fx=exp(x)-sin(x)-2
    dfx=exp(x)-cos(x)
    print("-----x-----fx-----dfx")
    print("{:12.7f}{:20.17f}{:12.7f}".format(x,fx,dfx))
    while abs(fx)>tol:
        x=x-fx/dfx
        fx=exp(x)-sin(x)-2
        dfx=exp(x)-cos(x)
        print("{:12.7f}{:20.17f}{:12.7f}".format(x,fx,dfx))
    print("Raiz: ",x)
    print("Erro: ",fx)
newton1()
```

Para você fazer

1. Resolva pelos 3 métodos, para poder comparar, a seguinte equação $x = \sqrt[24]{6106}$ no intervalo entre 1 e 2 e responda a seguir os 3 valores encontrados

bissecção	
cordas	
Newton	

2. Idem $y = 8 \cdot \sin(x) + 4 \cdot \cos(x) - 5.627310 = 0$ no intervalo entre 6 e 7 e responda a seguir os 3 valores encontrados

bissecção	
cordas	
Newton	

3. Idem $y = ((8 \cdot x)/10) \cdot e^{((6 \cdot x)/10)} - 321.751356 = 0$ no intervalo entre 6 e 7 e responda a seguir os 3 valores encontrados

bissecção	
cordas	
Newton	

4. Idem $y = 7 \cdot x^4 + 3 \cdot x^2 - 11146.142700 = 0$ no intervalo entre 6 e 7 e responda a seguir os 3 valores encontrados

bissecção	
cordas	
Newton	



Cálculo Numérico - solução de equações

Suponha que precisamos usar o computador para resolver equações. Já paramos para entender o que é resolver uma equação? Conhece-se a forma geral de uma equação $y = f(x)$. Resolver esta equação vem a ser descobrir quais os valores de x que fazem $f(x)$ ser igual a zero. Se olharmos para o gráfico, veremos que busca-se o valor da abscissa x que determina o cruzamento da curva de $f(x)$ no eixo dos x .

Por exemplo, suponha-se precisar descobrir a raiz 11 do número 1739? Ou seja, qual o número que multiplicado por ele mesmo 11 vezes dá como resultado 1739?

Dizendo isso na forma matemática, pode-se escrever $x^{11} = 1739$, ou melhor ainda $x^{11} - 1739 = 0$. Com isso, chegamos ao ponto.

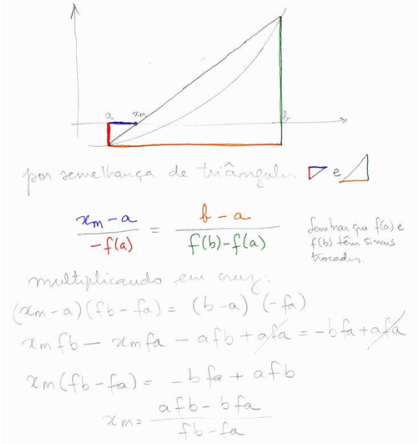
Método da Bissecção Se olharmos o gráfico de uma função, ver-se-á que no entorno da raiz, de um lado a função é positiva, e do outro lado é negativa ou vice versa. Este método começa pela seleção de 2 pontos x_a e x_b sendo que a função $y = f(x)$ têm sinais opostos nesses dois pontos. Graças a essa constatação, pode-se afirmar que a raiz procurada está entre x_a e x_b . Dizendo de outra maneira. O valor de x procurado, está entre a e b . O Método da bissecção vai dividindo o intervalo a, b pela metade, questionando a seguir em qual das metades o x está. Descoberto isso, despreza-se a metade que não contém x . Levando este processo ao limite, descobre-se qual o valor de x . Em resumo, eis as etapas do método:

1. Arbitrar intervalo em $[a, b]$ que compreenda a raiz.
2. Arbitrar um erro permitido ϵ
3. Calcular $x_m = (a + b)/2$
4. Se $f(a).f(x_m) < 0$ o novo intervalo será $[a, x_m]$
5. Se $f(a).f(x_m) > 0$ o novo intervalo será $[x_m, b]$
6. Enquanto o erro $b - a > \epsilon$ voltar ao passo 3.

Seja um exemplo, calcular $f = e^x - \sin(x) - 2 = 0$ nos limites $a = 1$ e $b = 1.25$ com $\epsilon = 0.01$. No primeira etapa tem-se

```
Limite Superior: 1.25
Limite Inferior: 1
Tolerancia: 0.01
-----a-----b-----xm-----fxm-----fa-----fb-----erro
1.1250 1.0000 1.1250 0.1779 0.5414 -0.1232 0.1250
1.0625 1.0000 1.0625 0.0200 0.1779 -0.1232 0.0625
1.0625 1.0312 1.0312 -0.0534 0.0200 -0.1232 0.0312
1.0625 1.0469 1.0469 -0.0171 0.0200 -0.0534 0.0156
1.0547 1.0469 1.0547 0.0013 0.0200 -0.0171 0.0078
Raiz: 1.05078125
Erro: -0.0078125
Eis o programa Python que resolve este caso
import numpy as np
def bissec():
    a=float(input("Limite Superior: "))
    b=float(input("Limite Inferior: "))
    tol=float(input("Tolerancia: "))
    print("-----a-----b-----xm-----fxm-----fa-----fb-----erro")
    while np.abs(b-a)>tol:
        xm=(a+b)/2
        fxm=np.exp(xm)-np.sin(xm)-2
        fa=np.exp(a)-np.sin(a)-2
        fb=np.exp(b)-np.sin(b)-2
        if fa*fxm>0:
            a=xm
        else:
            b=xm
        print('{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}'.format(a,b,xm,fxm,fa,fb,np.abs(b-a)))
    xm=(a+b)/2
    print("Raiz: ",xm)
    print("Erro: ",b-a)
bissec()
```

Cordas ou Falsa Posição



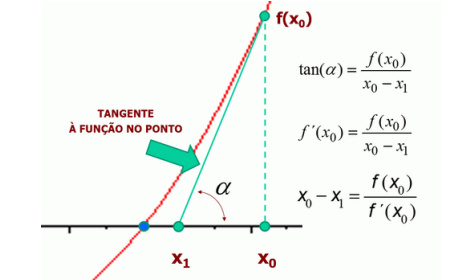
O método começa com uma análise do gráfico da função nas proximidades da raiz. Por semelhança de triângulos, pode-se obter a formulação proposta. Eis os passos do algoritmo:

1. Arbitrar um intervalo $[a, b]$ que compreende a raiz
2. Arbitrar ϵ
3. Calcular $x_m = \frac{a \cdot f(b) - b \cdot f(a)}{f(b) - f(a)}$
4. Se $f(a).f(x_m) < 0$ o novo intervalo será $[a, x_m]$
5. Se $f(a).f(x_m) > 0$ o novo intervalo será $[x_m, b]$
6. Enquanto o erro $f(x_m) > \epsilon$ voltar ao passo 3

Veja a seguir a solução da mesma equação, ($e^x - \sin(x) - 2 = 0$) com os mesmos intervalos do método anterior.

```
Limite inferior: 1
Limite superior: 1.25
Tolerância: 0.001
-----a-----b-----xm-----fxm-----fa-----fb-----erro
1.0463 1.2500 1.0463 -0.0184 -0.1232 0.5414
1.0530 1.2500 1.0530 -0.0026 -0.0184 0.5414
1.0540 1.2500 1.0540 -0.0004 -0.0026 0.5414
Raiz: 1.053972856764054
Erro: -0.0003663936835978099
Eis o programa Python que resolve este caso
from numpy import *
def cordas():
    a=float(input("Limite inferior: "))
    b=float(input("Limite superior: "))
    tol=float(input("Tolerância: "))
    fxm=1
    print("-----a-----b-----xm-----fxm-----fa-----fb-----erro")
    while abs(fxm)>tol:
        fa=exp(a)-sin(a)-2
        fb=exp(b)-sin(b)-2
        fxm=((a*fb)-(b*fa))/(fb-fa)
        fxm=exp(xm)-sin(xm)-2
        if fa*fxm>0:
            a=fxm
        else:
            b=fxm
        print('{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}'.format(a,b,xm,fxm,fa,fb))
    print("Raiz: ",xm)
    print("Erro: ",fxm)
cordas()
```

Método de Newton Este método começa com a interpretação geométrica da derivada. Lembrando, Newton é um monstro da ciência. Entre outras coisas, ele criou o cálculo diferencial e integral, a teoria da luz, a gravitação, as leis do movimento, além de, no fim da vida ter – como funcionário público – dirigido a casa da moeda inglesa, pendurando na forca alguns falsários delinquentes. Voltando ao cálculo diferencial, uma das interpretações da derivada de uma função em um ponto é a tangente da curva da função naquele ponto. Veja-se o que se disse neste desenho



Eis as etapas do método:

1. Obter a derivada da função a resolver

2. Arbitrar um ponto inicial x_0 para a função
3. Arbitrar o erro ϵ
4. Calcular $f(x_0)$ e $df(x_0)$
5. Calcular o “novo” x , $x = x_0 - f(x)/df(x)$
6. Se $|f(x)| > \epsilon$ retornar ao passo 4

Por exemplo, seja calcular as raízes da mesma equação acima que é $e^x - \sin(x) - 2 = 0$. A derivada é $\frac{d}{dx} = e^x - \cos(x)$. Obteve-se a seguinte tabela

```
-----x-----fx-----dfx
1.0000000 -0.12318915634885141 2.1779795
1.0565612 0.00579321190120519 2.3845934
1.0541318 0.00001105001756940 2.3754999
1.0541271 0.00000000004045120 2.3754825
Raiz: 1.0541271241082415
Erro: 4.0451197946822504e-11
Eis o programa Python que resolve este caso
from numpy import *
def newton():
    x=1
    tol=0.00001
    fx=exp(x)-sin(x)-2
    dfx=exp(x)-cos(x)
    print("-----x-----fx-----dfx")
    print("{:12.7f}{:20.17f}{:12.7f}".format(x,fx,dfx))
    while abs(fx)>tol:
        x=x-fx/dfx
        fx=exp(x)-sin(x)-2
        dfx=exp(x)-cos(x)
        print("{:12.7f}{:20.17f}{:12.7f}".format(x,fx,dfx))
    print("Raiz: ",x)
    print("Erro: ",fx)
newton()
```

Para você fazer

1. Resolva pelos 3 métodos, para poder comparar, a seguinte equação $x = \sqrt[2]{6085}$ no intervalo entre 1 e 2 e responda a seguir os 3 valores encontrados

bissecção	
cordas	
Newton	

2. Idem $y = 3 \cdot \sin(x) + 2 \cdot \cos(x) - 3.507271 = 0$ no intervalo entre 7 e 8 e responda a seguir os 3 valores encontrados

bissecção	
cordas	
Newton	

3. Idem $y = ((8 \cdot x)/10) \cdot e^{((5 \cdot x)/10)} - 421.201762 = 0$ no intervalo entre 8 e 9 e responda a seguir os 3 valores encontrados

bissecção	
cordas	
Newton	

4. Idem $y = 6 \cdot x^5 + 4 \cdot x^3 - 268679.687500 = 0$ no intervalo entre 8 e 9 e responda a seguir os 3 valores encontrados

bissecção	
cordas	
Newton	



Cálculo Numérico - solução de equações

Suponha que precisamos usar o computador para resolver equações. Já paramos para entender o que é resolver uma equação? Conhece-se a forma geral de uma equação $y = f(x)$. Resolver esta equação vem a ser descobrir quais os valores de x que fazem $f(x)$ ser igual a zero. Se olharmos para o gráfico, veremos que busca-se o valor da abscissa x que determina o cruzamento da curva de $f(x)$ no eixo dos x .

Por exemplo, suponha-se precisar descobrir a raiz 11 do número 1739? Ou seja, qual o número que multiplicado por ele mesmo 11 vezes dá como resultado 1739?

Dizendo isso na forma matemática, pode-se escrever $x^{11} = 1739$, ou melhor ainda $x^{11} - 1739 = 0$. Com isso, chegamos ao ponto.

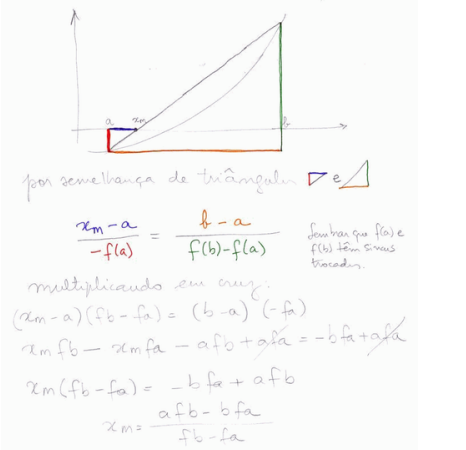
Método da Bissecção Se olharmos o gráfico de uma função, ver-se-á que no entorno da raiz, de um lado a função é positiva, e do outro lado é negativa ou vice versa. Este método começa pela seleção de 2 pontos x_a e x_b sendo que a função $y = f(x)$ têm sinais opostos nesses dois pontos. Graças a essa constatação, pode-se afirmar que a raiz procurada está entre x_a e x_b . Dizendo de outra maneira. O valor de x procurado, está entre a e b . O Método da bissecção vai dividindo o intervalo a, b pela metade, questionando a seguir em qual das metades o x está. Descoberto isso, despreza-se a metade que não contém x . Levando este processo ao limite, descobre-se qual o valor de x . Em resumo, eis as etapas do método:

1. Arbitrar intervalo em $[a, b]$ que compreenda a raiz.
2. Arbitrar um erro permitido ϵ
3. Calcular $x_m = (a + b)/2$
4. Se $f(a).f(x_m) < 0$ o novo intervalo será $[a, x_m]$
5. Se $f(a).f(x_m) > 0$ o novo intervalo será $[x_m, b]$
6. Enquanto o erro $b - a > \epsilon$ voltar ao passo 3.

Seja um exemplo, calcular $f = e^x - \sin(x) - 2 = 0$ nos limites $a = 1$ e $b = 1.25$ com $\epsilon = 0.01$. No primeira etapa tem-se

```
Limite Superior: 1.25
Limite Inferior: 1
Tolerancia: 0.01
-----a-----b-----xm-----fxm-----fa-----fb-----erro
1.1250 1.0000 1.1250 0.1779 0.5414 -0.1232 0.1250
1.0625 1.0000 1.0625 0.0200 0.1779 -0.1232 0.0625
1.0625 1.0312 1.0312 -0.0534 0.0200 -0.1232 0.0312
1.0625 1.0469 1.0469 -0.0171 0.0200 -0.0534 0.0156
1.0547 1.0469 1.0547 0.0013 0.0200 -0.0171 0.0078
Raiz: 1.05078125
Erro: -0.0078125
Eis o programa Python que resolve este caso
import numpy as np
def bissec():
    a=float(input("Limite Superior: "))
    b=float(input("Limite Inferior: "))
    tol=float(input("Tolerancia: "))
    print("-----a-----b-----xm-----fxm-----fa-----fb-----erro")
    while np.abs(b-a)>tol:
        xm=(a+b)/2
        fxm=np.exp(xm)-np.sin(xm)-2
        fa=np.exp(a)-np.sin(a)-2
        fb=np.exp(b)-np.sin(b)-2
        if fa*fxm>0:
            a=xm
        else:
            b=xm
        print('{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}'
              .format(a,b,xm,fxm,fa,fb,np.abs(b-a)))
    xm=(a+b)/2
    print("Raiz: ",xm)
    print("Erro: ",b-a)
bissec()
```

Cordas ou Falsa Posição



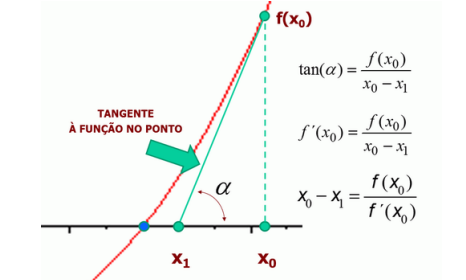
O método começa com uma análise do gráfico da função nas proximidades da raiz. Por semelhança de triângulos, pode-se obter a formulação proposta. Eis os passos do algoritmo:

1. Arbitrar um intervalo $[a, b]$ que compreende a raiz
2. Arbitrar ϵ
3. Calcular $x_m = \frac{a \cdot f(b) - b \cdot f(a)}{f(b) - f(a)}$
4. Se $f(a).f(x_m) < 0$ o novo intervalo será $[a, x_m]$
5. Se $f(a).f(x_m) > 0$ o novo intervalo será $[x_m, b]$
6. Enquanto o erro $f(x_m) > \epsilon$ voltar ao passo 3

Veja a seguir a solução da mesma equação, ($e^x - \sin(x) - 2 = 0$) com os mesmos intervalos do método anterior.

```
Limite inferior: 1
Limite superior: 1.25
Tolerância: 0.001
-----a-----b-----xm-----fxm-----fa-----fb-----erro
1.0463 1.2500 1.0463 -0.0184 -0.1232 0.5414
1.0530 1.2500 1.0530 -0.0026 -0.0184 0.5414
1.0540 1.2500 1.0540 -0.0004 -0.0026 0.5414
Raiz: 1.053972856764054
Erro: -0.0003663936835978099
Eis o programa Python que resolve este caso
from numpy import *
def cordas():
    a=float(input("Limite inferior: "))
    b=float(input("Limite superior: "))
    tol=float(input("Tolerância: "))
    fxm=1
    print("-----a-----b-----xm-----fxm-----fa-----fb-----erro")
    while abs(fxm)>tol:
        fa=exp(a)-sin(a)-2
        fb=exp(b)-sin(b)-2
        xm=((a*fb)-(b*fa))/(fb-fa)
        fxm=exp(xm)-sin(xm)-2
        if fa*fxm>0:
            a=xm
        else:
            b=xm
        print('{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}'
              .format(a,b,xm,fxm,fa,fb))
    print("Raiz: ",xm)
    print("Erro: ",fxm)
cordas()
```

Método de Newton Este método começa com a interpretação geométrica da derivada. Lembrando, Newton é um monstro da ciência. Entre outras coisas, ele criou o cálculo diferencial e integral, a teoria da luz, a gravitação, as leis do movimento, além de, no fim da vida ter – como funcionário público – dirigido a casa da moeda inglesa, pendurando na forca alguns falsários delinquentes. Voltando ao cálculo diferencial, uma das interpretações da derivada de uma função em um ponto é a tangente da curva da função naquele ponto. Veja-se o que se disse neste desenho



Eis as etapas do método:

1. Obter a derivada da função a resolver

2. Arbitrar um ponto inicial x_0 para a função
3. Arbitrar o erro ϵ
4. Calcular $f(x_0)$ e $df(x_0)$
5. Calcular o “novo” x , $x = x_0 - f(x)/df(x)$
6. Se $|f(x)| > \epsilon$ retornar ao passo 4

Por exemplo, seja calcular as raízes da mesma equação acima que é $e^x - \sin(x) - 2 = 0$. A derivada é $\frac{d}{dx} = e^x - \cos(x)$. Obteve-se a seguinte tabela

```
-----x-----fx-----dfx
1.0000000 -0.12318915634885141 2.1779795
1.0565612 0.00579321190120519 2.3845934
1.0541318 0.00001105001756940 2.3754999
1.0541271 0.00000000004045120 2.3754825
Raiz: 1.0541271241082415
Erro: 4.0451197946822504e-11
Eis o programa Python que resolve este caso
from numpy import *
def newton1():
    x=1
    tol=0.00001
    fx=exp(x)-sin(x)-2
    dfx=exp(x)-cos(x)
    print("-----x-----fx-----dfx")
    print("{:12.7f}{:20.17f}{:12.7f}".format(x,fx,dfx))
    while abs(fx)>tol:
        x=x-fx/dfx
        fx=exp(x)-sin(x)-2
        dfx=exp(x)-cos(x)
        print("{:12.7f}{:20.17f}{:12.7f}".format(x,fx,dfx))
    print("Raiz: ",x)
    print("Erro: ",fx)
newton1()
```

Para você fazer

1. Resolva pelos 3 métodos, para poder comparar, a seguinte equação $x = \sqrt[18]{5115}$ no intervalo entre 1 e 2 e responda a seguir os 3 valores encontrados

bissecção	
cordas	
Newton	

2. Idem $y = 2 * \sin(x) + 6 * \cos(x) - 4.383913 = 0$ no intervalo entre 5 e 6 e responda a seguir os 3 valores encontrados

bissecção	
cordas	
Newton	

3. Idem $y = ((7 * x)/10) * e^{((3 * x)/10)} - 13.475693 = 0$ no intervalo entre 4 e 5 e responda a seguir os 3 valores encontrados

bissecção	
cordas	
Newton	

4. Idem $y = 7 * x^6 + 5 * x^4 - 18897.165363 = 0$ no intervalo entre 3 e 4 e responda a seguir os 3 valores encontrados

bissecção	
cordas	
Newton	



Cálculo Numérico - solução de equações

Suponha que precisamos usar o computador para resolver equações. Já paramos para entender o que é resolver uma equação? Conhece-se a forma geral de uma equação $y = f(x)$. Resolver esta equação vem a ser descobrir quais os valores de x que fazem $f(x)$ ser igual a zero. Se olharmos para o gráfico, veremos que busca-se o valor da abscissa x que determina o cruzamento da curva de $f(x)$ no eixo dos x .

Por exemplo, suponha-se precisar descobrir a raiz 11 do número 1739? Ou seja, qual o número que multiplicado por ele mesmo 11 vezes dá como resultado 1739?

Dizendo isso na forma matemática, pode-se escrever $x^{11} = 1739$, ou melhor ainda $x^{11} - 1739 = 0$. Com isso, chegamos ao ponto.

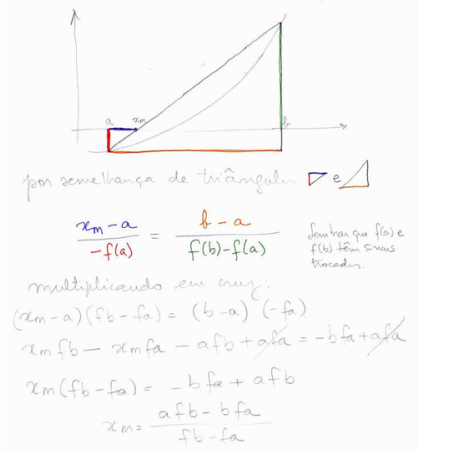
Método da Bissecção Se olharmos o gráfico de uma função, ver-se-á que no entorno da raiz, de um lado a função é positiva, e do outro lado é negativa ou vice versa. Este método começa pela seleção de 2 pontos x_a e x_b sendo que a função $y = f(x)$ têm sinais opostos nesses dois pontos. Graças a essa constatação, pode-se afirmar que a raiz procurada está entre x_a e x_b . Dizendo de outra maneira. O valor de x procurado, está entre a e b . O Método da bissecção vai dividindo o intervalo a, b pela metade, questionando a seguir em qual das metades o x está. Descoberto isso, despreza-se a metade que não contém x . Levando este processo ao limite, descobre-se qual o valor de x . Em resumo, eis as etapas do método:

1. Arbitrar intervalo em $[a, b]$ que compreenda a raiz.
2. Arbitrar um erro permitido ϵ
3. Calcular $x_m = (a + b)/2$
4. Se $f(a).f(x_m) < 0$ o novo intervalo será $[a, x_m]$
5. Se $f(a).f(x_m) > 0$ o novo intervalo será $[x_m, b]$
6. Enquanto o erro $b - a > \epsilon$ voltar ao passo 3.

Seja um exemplo, calcular $f = e^x - \sin(x) - 2 = 0$ nos limites $a = 1$ e $b = 1.25$ com $\epsilon = 0.01$. No primeira etapa tem-se

```
Limite Superior: 1.25
Limite Inferior: 1
Tolerancia: 0.01
-----a-----b-----xm-----fxm-----fa-----fb-----erro
1.1250 1.0000 1.1250 0.1779 0.5414 -0.1232 0.1250
1.0625 1.0000 1.0625 0.0200 0.1779 -0.1232 0.0625
1.0625 1.0312 1.0312 -0.0534 0.0200 -0.1232 0.0312
1.0625 1.0469 1.0469 -0.0171 0.0200 -0.0534 0.0156
1.0547 1.0469 1.0547 0.0013 0.0200 -0.0171 0.0078
Raiz: 1.05078125
Erro: -0.0078125
Eis o programa Python que resolve este caso
import numpy as np
def bissec():
    a=float(input("Limite Superior: "))
    b=float(input("Limite Inferior: "))
    tol=float(input("Tolerancia: "))
    print("-----a-----b-----xm-----fxm-----fa-----fb-----erro")
    while np.abs(b-a)>tol:
        xm=(a+b)/2
        fxm=np.exp(xm)-np.sin(xm)-2
        fa=np.exp(a)-np.sin(a)-2
        fb=np.exp(b)-np.sin(b)-2
        if fa*fxm>0:
            a=xm
        else:
            b=xm
        print('{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}'.format(a,b,xm,fxm,fa,fb,np.abs(b-a)))
    xm=(a+b)/2
    print("Raiz: ",xm)
    print("Erro: ",b-a)
bissec()
```

Cordas ou Falsa Posição



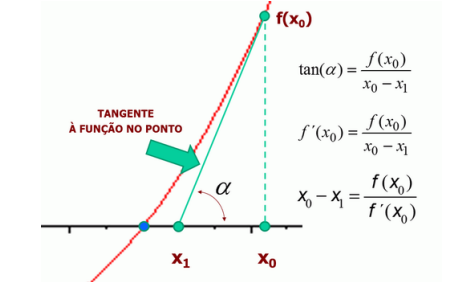
O método começa com uma análise do gráfico da função nas proximidades da raiz. Por semelhança de triângulos, pode-se obter a formulação proposta. Eis os passos do algoritmo:

1. Arbitrar um intervalo $[a, b]$ que compreende a raiz
2. Arbitrar ϵ
3. Calcular $x_m = \frac{a.f(b)-b.f(a)}{f(b)-f(a)}$
4. Se $f(a).f(x_m) < 0$ o novo intervalo será $[a, x_m]$
5. Se $f(a).f(x_m) > 0$ o novo intervalo será $[x_m, b]$
6. Enquanto o erro $f(x_m) > \epsilon$ voltar ao passo 3

Veja a seguir a solução da mesma equação, ($e^x - \sin(x) - 2 = 0$) com os mesmos intervalos do método anterior.

```
Limite inferior: 1
Limite superior: 1.25
Tolerância: 0.001
-----a-----b-----xm-----fxm-----fa-----fb-----erro
1.0463 1.2500 1.0463 -0.0184 -0.1232 0.5414
1.0530 1.2500 1.0530 -0.0026 -0.0184 0.5414
1.0540 1.2500 1.0540 -0.0004 -0.0026 0.5414
Raiz: 1.053972856764054
Erro: -0.0003663936835978099
Eis o programa Python que resolve este caso
from numpy import *
def cordas():
    a=float(input("Limite inferior: "))
    b=float(input("Limite superior: "))
    tol=float(input("Tolerância: "))
    fxm=1
    print("-----a-----b-----xm-----fxm-----fa-----fb-----erro")
    while abs(fxm)>tol:
        fa=exp(a)-sin(a)-2
        fb=exp(b)-sin(b)-2
        xm=((a*fb)-(b*fa))/(fb-fa)
        fxm=exp(xm)-sin(xm)-2
        if fa*fxm>0:
            a=xm
        else:
            b=xm
        print('{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}'.format(a,b,xm,fxm,fa,fb))
    print("Raiz: ",xm)
    print("Erro: ",fxm)
cordas()
```

Método de Newton Este método começa com a interpretação geométrica da derivada. Lembrando, Newton é um monstro da ciência. Entre outras coisas, ele criou o cálculo diferencial e integral, a teoria da luz, a gravitação, as leis do movimento, além de, no fim da vida ter – como funcionário público – dirigido a casa da moeda inglesa, pendurando na forca alguns falsários delinquentes. Voltando ao cálculo diferencial, uma das interpretações da derivada de uma função em um ponto é a tangente da curva da função naquele ponto. Veja-se o que se disse neste desenho



- Eis as etapas do método:
1. Obter a derivada da função a resolver

2. Arbitrar um ponto inicial x_0 para a função
3. Arbitrar o erro ϵ
4. Calcular $f(x_0)$ e $df(x_0)$
5. Calcular o “novo” x , $x = x_0 - f(x)/df(x)$
6. Se $|f(x)| > \epsilon$ retornar ao passo 4

Por exemplo, seja calcular as raízes da mesma equação acima que é $e^x - \sin(x) - 2 = 0$. A derivada é $\frac{d}{dx} = e^x - \cos(x)$. Obteve-se a seguinte tabela

```
-----x-----fx-----dfx
1.0000000 -0.12318915634885141 2.1779795
1.0565612 0.00579321190120519 2.3845934
1.0541318 0.00001105001756940 2.3754999
1.0541271 0.0000000004045120 2.3754825
Raiz: 1.0541271241082415
Erro: 4.0451197946822504e-11
Eis o programa Python que resolve este caso
from numpy import *
def newton():
    x=1
    tol=0.00001
    fx=exp(x)-sin(x)-2
    dfx=exp(x)-cos(x)
    print("-----x-----fx-----dfx")
    print("{:12.7f}{:20.17f}{:12.7f}".format(x,fx,dfx))
    while abs(fx)>tol:
        x=x-fx/dfx
        fx=exp(x)-sin(x)-2
        dfx=exp(x)-cos(x)
        print("{:12.7f}{:20.17f}{:12.7f}".format(x,fx,dfx))
    print("Raiz: ",x)
    print("Erro: ",fx)
newton()
```

Para você fazer

1. Resolva pelos 3 métodos, para poder comparar, a seguinte equação $x = \sqrt[18]{6167}$ no intervalo entre 1 e 2 e responda a seguir os 3 valores encontrados

bissecção	
cordas	
Newton	

2. Idem $y = 4*\sin(x) + 5*\cos(x) - 6.216428 = 0$ no intervalo entre 7 e 8 e responda a seguir os 3 valores encontrados

bissecção	
cordas	
Newton	

3. Idem $y = ((8 * x)/10) * e^{((5*x)/10)} - 84.328035 = 0$ no intervalo entre 5 e 6 e responda a seguir os 3 valores encontrados

bissecção	
cordas	
Newton	

4. Idem $y = 7 * x^6 + 4 * x^2 - 896903.627447 = 0$ no intervalo entre 7 e 8 e responda a seguir os 3 valores encontrados

bissecção	
cordas	
Newton	



Cálculo Numérico - solução de equações

Suponha que precisamos usar o computador para resolver equações. Já paramos para entender o que é resolver uma equação? Conhece-se a forma geral de uma equação $y = f(x)$. Resolver esta equação vem a ser descobrir quais os valores de x que fazem $f(x)$ ser igual a zero. Se olharmos para o gráfico, veremos que busca-se o valor da abscissa x que determina o cruzamento da curva de $f(x)$ no eixo dos x .

Por exemplo, suponha-se precisar descobrir a raiz 11 do número 1739? Ou seja, qual o número que multiplicado por ele mesmo 11 vezes dá como resultado 1739?

Dizendo isso na forma matemática, pode-se escrever $x^{11} = 1739$, ou melhor ainda $x^{11} - 1739 = 0$. Com isso, chegamos ao ponto.

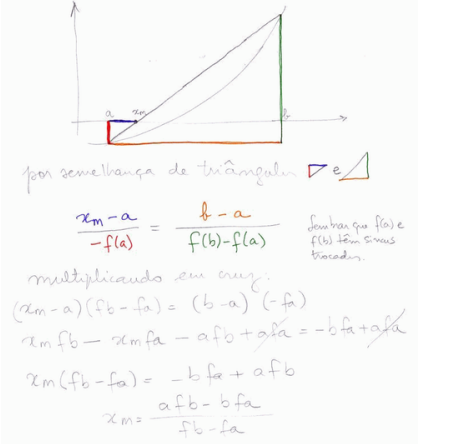
Método da Bissecção Se olharmos o gráfico de uma função, ver-se-á que no entorno da raiz, de um lado a função é positiva, e do outro lado é negativa ou vice versa. Este método começa pela seleção de 2 pontos x_a e x_b sendo que a função $y = f(x)$ têm sinais opostos nesses dois pontos. Graças a essa constatação, pode-se afirmar que a raiz procurada está entre x_a e x_b . Dizendo de outra maneira. O valor de x procurado, está entre a e b . O Método da bissecção vai dividindo o intervalo a, b pela metade, questionando a seguir em qual das metades o x está. Descoberto isso, despreza-se a metade que não contém x . Levando este processo ao limite, descobre-se qual o valor de x . Em resumo, eis as etapas do método:

1. Arbitrar intervalo em $[a, b]$ que compreenda a raiz.
2. Arbitrar um erro permitido ϵ
3. Calcular $x_m = (a + b)/2$
4. Se $f(a).f(x_m) < 0$ o novo intervalo será $[a, x_m]$
5. Se $f(a).f(x_m) > 0$ o novo intervalo será $[x_m, b]$
6. Enquanto o erro $b - a > \epsilon$ voltar ao passo 3.

Seja um exemplo, calcular $f = e^x - \sin(x) - 2 = 0$ nos limites $a = 1$ e $b = 1.25$ com $\epsilon = 0.01$. No primeira etapa tem-se

```
Limite Superior: 1.25
Limite Inferior: 1
Tolerancia: 0.01
-----a-----b-----xm-----fxm-----fa-----fb-----erro
1.1250 1.0000 1.1250 0.1779 0.5414 -0.1232 0.1250
1.0625 1.0000 1.0625 0.0200 0.1779 -0.1232 0.0625
1.0625 1.0312 1.0312 -0.0534 0.0200 -0.1232 0.0312
1.0625 1.0469 1.0469 -0.0171 0.0200 -0.0534 0.0156
1.0547 1.0469 1.0547 0.0013 0.0200 -0.0171 0.0078
Raiz: 1.05078125
Erro: -0.0078125
Eis o programa Python que resolve este caso
import numpy as np
def bissec():
    a=float(input("Limite Superior: "))
    b=float(input("Limite Inferior: "))
    tol=float(input("Tolerancia: "))
    print("-----a-----b-----xm-----fxm-----fa-----fb-----erro")
    while np.abs(b-a)>tol:
        xm=(a+b)/2
        fxm=np.exp(xm)-np.sin(xm)-2
        fa=np.exp(a)-np.sin(a)-2
        fb=np.exp(b)-np.sin(b)-2
        if fa*fxm>0:
            a=xm
        else:
            b=xm
        print('{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}'
              .format(a,b,xm,fxm,fa,fb,np.abs(b-a)))
    xm=(a+b)/2
    print("Raiz: ",xm)
    print("Erro: ",b-a)
bissec()
```

Cordas ou Falsa Posição



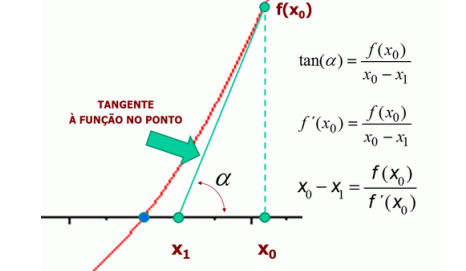
O método começa com uma análise do gráfico da função nas proximidades da raiz. Por semelhança de triângulos, pode-se obter a formulação proposta. Eis os passos do algoritmo:

1. Arbitrar um intervalo $[a, b]$ que compreende a raiz
2. Arbitrar ϵ
3. Calcular $x_m = \frac{a \cdot f(b) - b \cdot f(a)}{f(b) - f(a)}$
4. Se $f(a).f(x_m) < 0$ o novo intervalo será $[a, x_m]$
5. Se $f(a).f(x_m) > 0$ o novo intervalo será $[x_m, b]$
6. Enquanto o erro $f(x_m) > \epsilon$ voltar ao passo 3

Veja a seguir a solução da mesma equação, ($e^x - \sin(x) - 2 = 0$) com os mesmos intervalos do método anterior.

```
Limite inferior: 1
Limite superior: 1.25
Tolerância: 0.001
-----a-----b-----xm-----fxm-----fa-----fb-----erro
1.0463 1.2500 1.0463 -0.0184 -0.1232 0.5414
1.0530 1.2500 1.0530 -0.0026 -0.0184 0.5414
1.0540 1.2500 1.0540 -0.0004 -0.0026 0.5414
Raiz: 1.053972856764054
Erro: -0.0003663936835978099
Eis o programa Python que resolve este caso
from numpy import *
def cordas():
    a=float(input("Limite inferior: "))
    b=float(input("Limite superior: "))
    tol=float(input("Tolerância: "))
    fxm=1
    print("-----a-----b-----xm-----fxm-----fa-----fb-----erro")
    while abs(fxm)>tol:
        fa=exp(a)-sin(a)-2
        fb=exp(b)-sin(b)-2
        fxm=((a*fb)-(b*fa))/(fb-fa)
        fxm=exp(xm)-sin(xm)-2
        if fa*fxm>0:
            a=fxm
        else:
            b=fxm
        print('{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}'
              .format(a,b,xm,fxm,fa,fb))
    print("Raiz: ",xm)
    print("Erro: ",fxm)
cordas()
```

Método de Newton Este método começa com a interpretação geométrica da derivada. Lembrando, Newton é um monstro da ciência. Entre outras coisas, ele criou o cálculo diferencial e integral, a teoria da luz, a gravitação, as leis do movimento, além de, no fim da vida ter – como funcionário público – dirigido a casa da moeda inglesa, pendurando na forca alguns falsários delinquentes. Voltando ao cálculo diferencial, uma das interpretações da derivada de uma função em um ponto é a tangente da curva da função naquele ponto. Veja-se o que se disse neste desenho



- Eis as etapas do método:
1. Obter a derivada da função a resolver

2. Arbitrar um ponto inicial x_0 para a função
3. Arbitrar o erro ϵ
4. Calcular $f(x_0)$ e $df(x_0)$
5. Calcular o “novo” x , $x = x_0 - f(x)/df(x)$
6. Se $|f(x)| > \epsilon$ retornar ao passo 4

Por exemplo, seja calcular as raízes da mesma equação acima que é $e^x - \sin(x) - 2 = 0$. A derivada é $\frac{d}{dx} = e^x - \cos(x)$. Obteve-se a seguinte tabela

```
-----x-----fx-----dfx
1.0000000 -0.12318915634885141 2.1779795
1.0565612 0.00579321190120519 2.3845934
1.0541318 0.00001105001756940 2.3754999
1.0541271 0.0000000004045120 2.3754825
Raiz: 1.0541271241082415
Erro: 4.0451197946822504e-11
Eis o programa Python que resolve este caso
from numpy import *
def newton1():
    x=1
    tol=0.00001
    fx=exp(x)-sin(x)-2
    dfx=exp(x)-cos(x)
    print("-----x-----fx-----dfx")
    print("{:12.7f}{:20.17f}{:12.7f}"
          .format(x,fx,dfx))
    while abs(fx)>tol:
        x=x-fx/dfx
        fx=exp(x)-sin(x)-2
        dfx=exp(x)-cos(x)
        print("{:12.7f}{:20.17f}{:12.7f}"
              .format(x,fx,dfx))
    print("Raiz: ",x)
    print("Erro: ",fx)
newton1()
```

Para você fazer

1. Resolva pelos 3 métodos, para poder comparar, a seguinte equação $x = \sqrt[24]{7370}$ no intervalo entre 1 e 2 e responda a seguir os 3 valores encontrados

bissecção	
cordas	
Newton	

2. Idem $y = 6 * \sin(x) + 7 * \cos(x) + 8.659896 = 0$ no intervalo entre 3 e 4 e responda a seguir os 3 valores encontrados

bissecção	
cordas	
Newton	

3. Idem $y = ((6 * x)/10) * e^{((4 * x)/10)} - 218.848095 = 0$ no intervalo entre 9 e 10 e responda a seguir os 3 valores encontrados

bissecção	
cordas	
Newton	

4. Idem $y = 6 * x^4 + 3 * x^2 - 8419.134600 = 0$ no intervalo entre 6 e 7 e responda a seguir os 3 valores encontrados

bissecção	
cordas	
Newton	



Cálculo Numérico - solução de equações

Suponha que precisamos usar o computador para resolver equações. Já paramos para entender o que é resolver uma equação? Conhece-se a forma geral de uma equação $y = f(x)$. Resolver esta equação vem a ser descobrir quais os valores de x que fazem $f(x)$ ser igual a zero. Se olharmos para o gráfico, veremos que busca-se o valor da abscissa x que determina o cruzamento da curva de $f(x)$ no eixo dos x .

Por exemplo, suponha-se precisar descobrir a raiz 11 do número 1739? Ou seja, qual o número que multiplicado por ele mesmo 11 vezes dá como resultado 1739?

Dizendo isso na forma matemática, pode-se escrever $x^{11} = 1739$, ou melhor ainda $x^{11} - 1739 = 0$. Com isso, chegamos ao ponto.

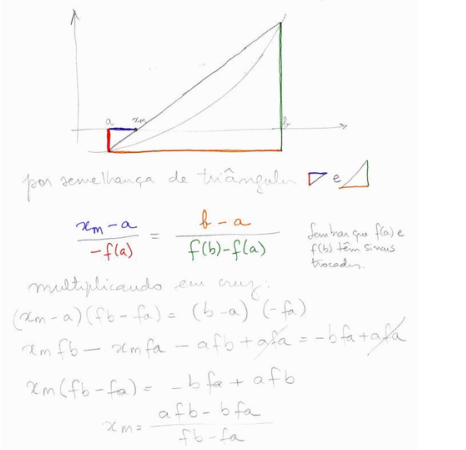
Método da Bissecção Se olharmos o gráfico de uma função, ver-se-á que no entorno da raiz, de um lado a função é positiva, e do outro lado é negativa ou vice versa. Este método começa pela seleção de 2 pontos x_a e x_b sendo que a função $y = f(x)$ têm sinais opostos nesses dois pontos. Graças a essa constatação, pode-se afirmar que a raiz procurada está entre x_a e x_b . Dizendo de outra maneira. O valor de x procurado, está entre a e b . O Método da bissecção vai dividindo o intervalo a, b pela metade, questionando a seguir em qual das metades o x está. Descoberto isso, despreza-se a metade que não contém x . Levando este processo ao limite, descobre-se qual o valor de x . Em resumo, eis as etapas do método:

1. Arbitrar intervalo em $[a, b]$ que compreenda a raiz.
2. Arbitrar um erro permitido ϵ
3. Calcular $x_m = (a + b)/2$
4. Se $f(a).f(x_m) < 0$ o novo intervalo será $[a, x_m]$
5. Se $f(a).f(x_m) > 0$ o novo intervalo será $[x_m, b]$
6. Enquanto o erro $b - a > \epsilon$ voltar ao passo 3.

Seja um exemplo, calcular $f = e^x - \sin(x) - 2 = 0$ nos limites $a = 1$ e $b = 1.25$ com $\epsilon = 0.01$. No primeira etapa tem-se

```
Limite Superior: 1.25
Limite Inferior: 1
Tolerancia: 0.01
-----a-----b-----xm-----fxm-----fa-----fb-----erro
1.1250 1.0000 1.1250 0.1779 0.5414 -0.1232 0.1250
1.0625 1.0000 1.0625 0.0200 0.1779 -0.1232 0.0625
1.0625 1.0312 1.0312 -0.0534 0.0200 -0.1232 0.0312
1.0625 1.0469 1.0469 -0.0171 0.0200 -0.0534 0.0156
1.0547 1.0469 1.0547 0.0013 0.0200 -0.0171 0.0078
Raiz: 1.05078125
Erro: -0.0078125
Eis o programa Python que resolve este caso
import numpy as np
def bissec():
    a=float(input("Limite Superior: "))
    b=float(input("Limite Inferior: "))
    tol=float(input("Tolerancia: "))
    print("-----a-----b-----xm-----fxm-----fa-----fb-----erro")
    while np.abs(b-a)>tol:
        xm=(a+b)/2
        fxm=np.exp(xm)-np.sin(xm)-2
        fa=np.exp(a)-np.sin(a)-2
        fb=np.exp(b)-np.sin(b)-2
        if fa*fxm>0:
            a=xm
        else:
            b=xm
        print('{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}'.format(a,b,xm,fxm,fa,fb,np.abs(b-a)))
    xm=(a+b)/2
    print("Raiz: ",xm)
    print("Erro: ",b-a)
bissec()
```

Cordas ou Falsa Posição



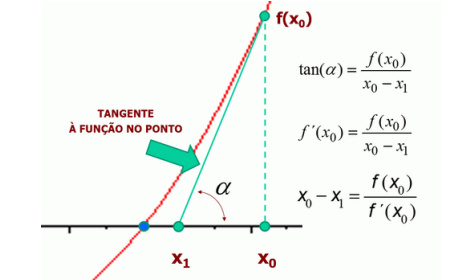
O método começa com uma análise do gráfico da função nas proximidades da raiz. Por semelhança de triângulos, pode-se obter a formulação proposta. Eis os passos do algoritmo:

1. Arbitrar um intervalo $[a, b]$ que compreende a raiz
2. Arbitrar ϵ
3. Calcular $x_m = \frac{a \cdot f(b) - b \cdot f(a)}{f(b) - f(a)}$
4. Se $f(a).f(x_m) < 0$ o novo intervalo será $[a, x_m]$
5. Se $f(a).f(x_m) > 0$ o novo intervalo será $[x_m, b]$
6. Enquanto o erro $f(x_m) > \epsilon$ voltar ao passo 3

Veja a seguir a solução da mesma equação, ($e^x - \sin(x) - 2 = 0$) com os mesmos intervalos do método anterior.

```
Limite inferior: 1
Limite superior: 1.25
Tolerancia: 0.001
-----a-----b-----xm-----fxm-----fa-----fb-----erro
1.0463 1.2500 1.0463 -0.0184 -0.1232 0.5414
1.0530 1.2500 1.0530 -0.0026 -0.0184 0.5414
1.0540 1.2500 1.0540 -0.0004 -0.0026 0.5414
Raiz: 1.053972856764054
Erro: -0.0003663936835978099
Eis o programa Python que resolve este caso
from numpy import *
def cordas():
    a=float(input("Limite inferior: "))
    b=float(input("Limite superior: "))
    tol=float(input("Tolerancia: "))
    fxm=1
    print("-----a-----b-----xm-----fxm-----fa-----fb-----erro")
    while abs(fxm)>tol:
        fa=exp(a)-sin(a)-2
        fb=exp(b)-sin(b)-2
        fxm=((a*fb)-(b*fa))/(fb-fa)
        fxm=exp(xm)-sin(xm)-2
        if fa*fxm>0:
            a=fxm
        else:
            b=fxm
        print('{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}'.format(a,b,xm,fxm,fa,fb))
    print("Raiz: ",xm)
    print("Erro: ",fxm)
cordas()
```

Método de Newton Este método começa com a interpretação geométrica da derivada. Lembrando, Newton é um monstro da ciência. Entre outras coisas, ele criou o cálculo diferencial e integral, a teoria da luz, a gravitação, as leis do movimento, além de, no fim da vida ter – como funcionário público – dirigido a casa da moeda inglesa, pendurando na forca alguns falsários delinquentes. Voltando ao cálculo diferencial, uma das interpretações da derivada de uma função em um ponto é a tangente da curva da função naquele ponto. Veja-se o que se disse neste desenho



Eis as etapas do método:

1. Obter a derivada da função a resolver

2. Arbitrar um ponto inicial x_0 para a função
3. Arbitrar o erro ϵ
4. Calcular $f(x_0)$ e $df(x_0)$
5. Calcular o “novo” x , $x = x_0 - f(x)/df(x)$
6. Se $|f(x)| > \epsilon$ retornar ao passo 4

Por exemplo, seja calcular as raízes da mesma equação acima que é $e^x - \sin(x) - 2 = 0$. A derivada é $\frac{d}{dx} = e^x - \cos(x)$. Obteve-se a seguinte tabela

```
-----x-----fx-----dfx
1.0000000 -0.12318915634885141 2.1779795
1.0565612 0.00579321190120519 2.3845934
1.0541318 0.00001105001756940 2.3754999
1.0541271 0.00000000004045120 2.3754825
Raiz: 1.0541271241082415
Erro: 4.0451197946822504e-11
Eis o programa Python que resolve este caso
from numpy import *
def newton():
    x=1
    tol=0.00001
    fx=exp(x)-sin(x)-2
    dfx=exp(x)-cos(x)
    print("-----x-----fx-----dfx")
    print("{:12.7f}{:20.17f}{:12.7f}".format(x,fx,dfx))
    while abs(fx)>tol:
        x=x-fx/dfx
        fx=exp(x)-sin(x)-2
        dfx=exp(x)-cos(x)
        print("{:12.7f}{:20.17f}{:12.7f}".format(x,fx,dfx))
    print("Raiz: ",x)
    print("Erro: ",fx)
newton()
```

Para você fazer

1. Resolva pelos 3 métodos, para poder comparar, a seguinte equação $x = \sqrt[2]{7062}$ no intervalo entre 1 e 2 e responda a seguir os 3 valores encontrados

bissecção	
cordas	
Newton	

2. Idem $y = 8 \sin(x) + 7 \cos(x) - 5.151760 = 0$ no intervalo entre 8 e 9 e responda a seguir os 3 valores encontrados

bissecção	
cordas	
Newton	

3. Idem $y = ((6 * x)/10) * e^{((3*x)/10)} - 1.278448 = 0$ no intervalo entre 1 e 2 e responda a seguir os 3 valores encontrados

bissecção	
cordas	
Newton	

4. Idem $y = 7 * x^6 + 3 * x^2 - 295368.165487 = 0$ no intervalo entre 5 e 6 e responda a seguir os 3 valores encontrados

bissecção	
cordas	
Newton	



Cálculo Numérico - solução de equações

Suponha que precisamos usar o computador para resolver equações. Já paramos para entender o que é resolver uma equação? Conhece-se a forma geral de uma equação $y = f(x)$. Resolver esta equação vem a ser descobrir quais os valores de x que fazem $f(x)$ ser igual a zero. Se olharmos para o gráfico, veremos que busca-se o valor da abscissa x que determina o cruzamento da curva de $f(x)$ no eixo dos x .

Por exemplo, suponha-se precisar descobrir a raiz 11 do número 1739? Ou seja, qual o número que multiplicado por ele mesmo 11 vezes dá como resultado 1739?

Dizendo isso na forma matemática, pode-se escrever $x^{11} = 1739$, ou melhor ainda $x^{11} - 1739 = 0$. Com isso, chegamos ao ponto.

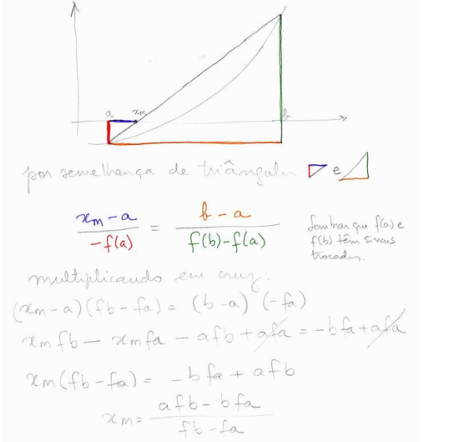
Método da Bissecção Se olharmos o gráfico de uma função, ver-se-á que no entorno da raiz, de um lado a função é positiva, e do outro lado é negativa ou vice versa. Este método começa pela seleção de 2 pontos x_a e x_b sendo que a função $y = f(x)$ têm sinais opostos nesses dois pontos. Graças a essa constatação, pode-se afirmar que a raiz procurada está entre x_a e x_b . Dizendo de outra maneira. O valor de x procurado, está entre a e b . O Método da bissecção vai dividindo o intervalo a, b pela metade, questionando a seguir em qual das metades o x está. Descoberto isso, despreza-se a metade que não contém x . Levando este processo ao limite, descobre-se qual o valor de x . Em resumo, eis as etapas do método:

1. Arbitrar intervalo em $[a, b]$ que compreenda a raiz.
2. Arbitrar um erro permitido ϵ
3. Calcular $x_m = (a + b)/2$
4. Se $f(a) \cdot f(x_m) < 0$ o novo intervalo será $[a, x_m]$
5. Se $f(a) \cdot f(x_m) > 0$ o novo intervalo será $[x_m, b]$
6. Enquanto o erro $b - a > \epsilon$ voltar ao passo 3.

Seja um exemplo, calcular $f = e^x - \sin(x) - 2 = 0$ nos limites $a = 1$ e $b = 1.25$ com $\epsilon = 0.01$. No primeira etapa tem-se

```
Limite Superior: 1.25
Limite Inferior: 1
Tolerancia: 0.01
-----a-----b-----xm-----fxm-----fa-----fb-----erro
1.1250 1.0000 1.1250 0.1779 0.5414 -0.1232 0.1250
1.0625 1.0000 1.0625 0.0200 0.1779 -0.1232 0.0625
1.0625 1.0312 1.0312 -0.0534 0.0200 -0.1232 0.0312
1.0625 1.0469 1.0469 -0.0171 0.0200 -0.0534 0.0156
1.0547 1.0469 1.0547 0.0013 0.0200 -0.0171 0.0078
Raiz: 1.05078125
Erro: -0.0078125
Eis o programa Python que resolve este caso
import numpy as np
def bissec():
    a=float(input("Limite Superior: "))
    b=float(input("Limite Inferior: "))
    tol=float(input("Tolerancia: "))
    print("-----a-----b-----xm-----fxm-----fa-----fb-----erro")
    while np.abs(b-a)>tol:
        xm=(a+b)/2
        fxm=np.exp(xm)-np.sin(xm)-2
        fa=np.exp(a)-np.sin(a)-2
        fb=np.exp(b)-np.sin(b)-2
        if fa*fxm>0:
            a=xm
        else:
            b=xm
        print('{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}'
              .format(a,b,xm,fxm,fa,fb,np.abs(b-a)))
    xm=(a+b)/2
    print("Raiz: ",xm)
    print("Erro: ",b-a)
bissec()
```

Cordas ou Falsa Posição



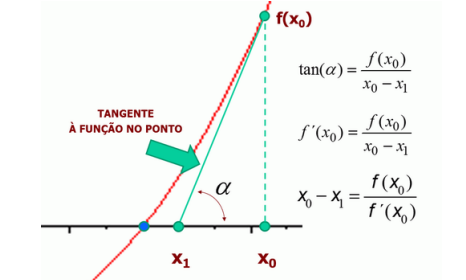
O método começa com uma análise do gráfico da função nas proximidades da raiz. Por semelhança de triângulos, pode-se obter a formulação proposta. Eis os passos do algoritmo:

1. Arbitrar um intervalo $[a, b]$ que compreende a raiz
2. Arbitrar ϵ
3. Calcular $x_m = \frac{a \cdot f(b) - b \cdot f(a)}{f(b) - f(a)}$
4. Se $f(a) \cdot f(x_m) < 0$ o novo intervalo será $[a, x_m]$
5. Se $f(a) \cdot f(x_m) > 0$ o novo intervalo será $[x_m, b]$
6. Enquanto o erro $f(x_m) > \epsilon$ voltar ao passo 3

Veja a seguir a solução da mesma equação, ($e^x - \sin(x) - 2 = 0$) com os mesmos intervalos do método anterior.

```
Limite inferior: 1
Limite superior: 1.25
Tolerância: 0.001
-----a-----b-----xm-----fxm-----fa-----fb-----erro
1.0463 1.2500 1.0463 -0.0184 -0.1232 0.5414
1.0530 1.2500 1.0530 -0.0026 -0.0184 0.5414
1.0540 1.2500 1.0540 -0.0004 -0.0026 0.5414
Raiz: 1.053972856764054
Erro: -0.0003663936835978099
Eis o programa Python que resolve este caso
from numpy import *
def cordas():
    a=float(input("Limite inferior: "))
    b=float(input("Limite superior: "))
    tol=float(input("Tolerância: "))
    fxm=1
    print("-----a-----b-----xm-----fxm-----fa-----fb-----erro")
    while abs(fxm)>tol:
        fa=exp(a)-sin(a)-2
        fb=exp(b)-sin(b)-2
        xm=((a*fb)-(b*fa))/(fb-fa)
        fxm=exp(xm)-sin(xm)-2
        if fa*fxm>0:
            a=xm
        else:
            b=xm
        print('{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}{:8.4f}'
              .format(a,b,xm,fxm,fa,fb))
    print("Raiz: ",xm)
    print("Erro: ",fxm)
cordas()
```

Método de Newton Este método começa com a interpretação geométrica da derivada. Lembrando, Newton é um monstro da ciência. Entre outras coisas, ele criou o cálculo diferencial e integral, a teoria da luz, a gravitação, as leis do movimento, além de, no fim da vida ter – como funcionário público – dirigido a casa da moeda inglesa, pendurando na forca alguns falsários delinquentes. Voltando ao cálculo diferencial, uma das interpretações da derivada de uma função em um ponto é a tangente da curva da função naquele ponto. Veja-se o que se disse neste desenho



Eis as etapas do método:

1. Obter a derivada da função a resolver

2. Arbitrar um ponto inicial x_0 para a função
3. Arbitrar o erro ϵ
4. Calcular $f(x_0)$ e $df(x_0)$
5. Calcular o “novo” x , $x = x_0 - f(x)/df(x)$
6. Se $|f(x)| > \epsilon$ retornar ao passo 4

Por exemplo, seja calcular as raízes da mesma equação acima que é $e^x - \sin(x) - 2 = 0$. A derivada é $\frac{d}{dx} = e^x - \cos(x)$. Obteve-se a seguinte tabela

```
-----x-----fx-----dfx
1.0000000 -0.12318915634885141 2.1779795
1.0565612 0.00579321190120519 2.3845934
1.0541318 0.00001105001756940 2.3754999
1.0541271 0.00000000004045120 2.3754825
Raiz: 1.0541271241082415
Erro: 4.0451197946822504e-11
Eis o programa Python que resolve este caso
from numpy import *
def newton():
    x=1
    tol=0.00001
    fx=exp(x)-sin(x)-2
    dfx=exp(x)-cos(x)
    print("-----x-----fx-----dfx")
    print("{:12.7f}{:20.17f}{:12.7f}".format(x,fx,dfx))
    while abs(fx)>tol:
        x=x-fx/dfx
        fx=exp(x)-sin(x)-2
        dfx=exp(x)-cos(x)
        print("{:12.7f}{:20.17f}{:12.7f}".format(x,fx,dfx))
    print("Raiz: ",x)
    print("Erro: ",fx)
newton()
```

Para você fazer

1. Resolva pelos 3 métodos, para poder comparar, a seguinte equação $x = \sqrt[18]{8808}$ no intervalo entre 1 e 2 e responda a seguir os 3 valores encontrados

bissecção	
cordas	
Newton	

2. Idem $y = 3 \cdot \sin(x) + 6 \cdot \cos(x) + 4.142293 = 0$ no intervalo entre 2 e 3 e responda a seguir os 3 valores encontrados

bissecção	
cordas	
Newton	

3. Idem $y = ((4 \cdot x)/10) \cdot e^{((3 \cdot x)/10)} - 16.680809 = 0$ no intervalo entre 6 e 7 e responda a seguir os 3 valores encontrados

bissecção	
cordas	
Newton	

4. Idem $y = 7 \cdot x^6 + 4 \cdot x^3 - 1578295.412928 = 0$ no intervalo entre 7 e 8 e responda a seguir os 3 valores encontrados

bissecção	
cordas	
Newton	

