Laborator VBA pentru Excel 2

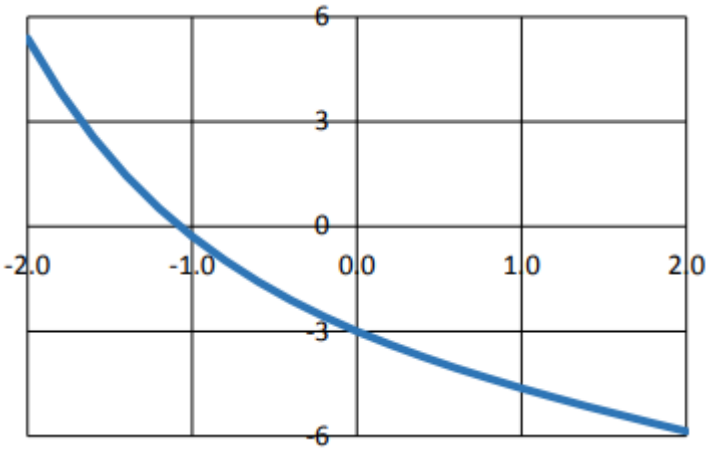
Metode numerice in VBA

Vom folosi VB pentru a găsi rădăcinile ecuației

$$e^{-x} - x - 4 = 0$$

prin două metode:

- metoda bisecției și
- metoda Newton-Raphson.



Ecuația de mai sus are o rădăcină în intervalul -2, 2.

Determinarea unei rădăcini prin metoda bisecției:

1. Lansați Microsoft Excel
2. Lansați Visual Basic Editor
3. In VBE adăugați un modul **Module1**
4. Tastați codul de mai jos in modulul **Module1**

```
Option Explicit
Private Const Toleranta As Double = 0.001

Function Bisectie(a As Double, b As Double) As Variant
    If (Sgn(f(a)) = Sgn(f(b))) Then
        Bisectie = "f(a) si f(b) trebuie sa aiba semn diferit"
    Else
        Bisectie = BisectionMethod(a, b)
    End If
End Function

Private Function f(x)
    f = Exp(-x) - x - 4
End Function

Private Function BisectionMethod(a As Double, b As Double) As Double
    Dim Midpoint As Double
    Dim Approx As Double

    Do
        Midpoint = (a + b) / 2
        Approx = f(Midpoint)
        If Sgn(f(a)) = Sgn(Approx) Then
```

```
            a = Midpoint
        Else
            b = Midpoint
    End If
    Loop Until (((b - a) <= Toleranta) Or (Approx = 0))
    BisectionMethod = Midpoint
End Function
```

Noi funcții și operatori VBA în codul de mai sus

Exp(x) = Returnează un **Double** care este e (baza logaritmilor naturali) ridicat la puterea x.

Sgn(x) = Returnează un **Variant** (**Integer**) ce indică semnul numărului x.

Or = Execută o disjuncție logică asupra a două expresii.

Remarcați că funcțiile **f** și **BisectionMethod** sunt private (vizibile doar în modulul **Module1**), funcția care va fi folosită în Excel fiind **Bisectie**, singura implicită **Public** în modul.

Am declarat și o constantă denumită Toleranta care este **Private** (vizibila doar în modul **Module1**), dar fiind declarată în afara procedurilor ea este vizibilă în tot modulul.

5. Inserați formula simplă Bisectie în Excel și modificați valoarea constantei Toleranta pentru a vedea efectul ei asupra rezultatului
6. Executați codul pas cu pas pentru a vedea modul de calcul al metodei

Determinarea unei rădăcini prin metoda Newton Raphson:

1. In VBE adăugați un nou modul **Module2**
2. Tastați codul de mai jos in modulul **Module2**

```
Option Explicit
Private Const Toleranta As Double = 0.0001
Private Const PI As Double = 3.14159265358979
Private Const MaxIteratii = 1000

Private Function Func(ByVal x As Double, Optional ByRef df As Double) As Double
    Func = Exp(-x) - x - 4
    df = -Exp(-x) - 1
End Function

Public Function NewtonRaphson(x As Double, tip As Integer) As Variant
    Select Case tip
        Case 1
            NewtonRaphson = Newton(x)
        Case 2
```

```
            NewtonRaphson = NewtonNumDeriv(x)
        Case Else
            NewtonRaphson = "Selectati 1 sau 2"
    End Select
End Function

Private Function Newton(x As Double) As Variant
    ' Determina o radacina prin metoda Newton-Raphson
    ' utilizand derivata analitica
    Dim iter As Integer
    Dim f As Double, df As Double, dx As Double

    For iter = 1 To MaxIteratii
        ' functia si derivata
        f = Func(x, df)
        ' corectia radacinii
         dx = IIf((Abs(df) > Toleranta), -f / df, -f)
        ' noua aproximatie
         x = x + dx
        If (Abs(dx) <= Toleranta * Abs(x)) Then
            Newton = x
            Exit Function
        End If
    Next iter

    Newton = "Nr. maxim de iteratii depasit!"
End Function

Private Function NewtonNumDeriv(x As Double) As Variant
    ' Determina prin metoda Newton-Raphson utilizand derivarea numerica
    Dim iter As Integer
    Dim f As Double, df As Double, dx As Double

    For iter = 1 To MaxIteratii
        f = Func(x)
        ' pasul de derivare
        dx = 0.000000001
        ' derivata numerica
        df = (Func(x + dx) - f) / dx
        ' corectia radacinii
        dx = IIf((Abs(df) > Toleranta), -f / df, -f)
        ' noua aproximatie
```

```
        x = x + dx
        If (Abs(dx) <= Toleranta * Abs(x)) Then
            NewtonNumDeriv = x
            Exit Function
        End If
    Next iter

    NewtonNumDeriv = "Nr. maxim de iteratii depasit!"
End Function
```

Noi funcții și instrucțiuni VBA în codul de mai sus

IIf(conditie, parte True, parte False) = este o funcție ce evaluează condiția. Dacă condiția este adevărată, returnează partea **True**, altfel partea **False**

Abs(x) = Returnează valoarea absolută a numărului x.

Select Case conditie = Permite executarea uneia sau a mai multor instrucțiuni funcție de valoarea condiției.

Optional = semnifică faptul că parametrul ce urmează poate să lipsească.

Remarcați că în funcția **Func**, parametrul x este transmis prin valoare (**ByVal**), pe când parametrul opțional df prin referință (**ByRef**). Acest lucru precizează faptul că în urma apelului funcției **Func**, cel de-al doilea parametru **poate fi modificat** în funcție și **va fi returnat cu noua valoare** în procedura apelantă. Dacă cuvintele cheie **ByRef** sau **ByVal** lipsesc, VB consideră implicit modul de transfer al variabilelor **ByRef**.

3. Inserați formula simplă Newton în Excel și modificați valoarea constantei Toleranta pentru a vedea efectul ei asupra rezultatului
4. Executați codul pas cu pas pentru a vedea modul de calcul al metodei

Calculul unei integrale

Vom folosi VB pentru a calcula integrala definită pe intervalul a, b

$$\int_{x=a}^{x=b} \frac{e^{\frac{-x^2}{2}}}{\sqrt{2\pi}} dx$$

folosind metoda dreptunghiului și a trapezului.

Pentru a aplica metodele de integrare:

1. Adăugați un nou modul **Module3**
2. Tastați codul de mai jos in modulul **Module3**

```
Option Explicit
Public Enum TipIntegrala
    Stanga = 1
```

```
    Centru = 2
    Trapez = 3
End Enum
Private Const PI As Double = 3.14159265358979

Private Function f(x As Double) As Double
    f = Exp(-x ^ 2 / 2) / Sqr(2 * PI)
End Function

Function Integrala(a As Double, b As Double, nInt As Integer, tip _
As TipIntegrala)
    Select Case tip
        Case Stanga
            Integrala = DreptunghiStanga(a, b, nInt)
        Case Centru
            Integrala = DreptunghiCentru(a, b, nInt)
        Case Trapez
            Integrala = IntegralaTrapez(a, b, nInt)
        Case Else
            Integrala = "Selectati tip intre 1 si 4"
    End Select
End Function

Private Function DreptunghiStanga(a As Double, b As Double, nInt As Integer) _
As Double
    Dim i As Integer
    Dim x As Double, y As Double, dx As Double

    ' Calculeaza pasul de integrare
    dx = (b - a) / nInt
    DreptunghiStanga = 0
    For i = 1 To nInt
        'Limita stanga a lui x
        x = a + (i - 1) * dx
        'y = f(x) la Limita stanga
        y = f(x)
        DreptunghiStanga = DreptunghiStanga + y * dx
    Next i
End Function

Private Function DreptunghiCentru(a As Double, b As Double, nInt As Integer) _
As Double
```

```
    Dim i As Integer
    Dim x As Double, y As Double, dx As Double
    Dim xstng As Double, xdrpt As Double

    dx = (b - a) / nInt
    DreptunghiCentru = 0
    For i = 1 To nInt
        xstng = a + (i - 1) * dx
        xdrpt = a + i * dx
        x = (xstng + xdrpt) / 2
        y = f(x)
        DreptunghiCentru = DreptunghiCentru + y * dx
    Next i
End Function

Private Function IntegralaTrapez(a As Double, b As Double, nInt As Integer) _
As Double
    Dim i As Integer
    Dim x As Double, y As Double, dx As Double
    Dim xstng As Double, xdrpt As Double
    Dim ystng As Double, ydrpt As Double

    dx = (b - a) / nInt
    IntegralaTrapez = 0

    For i = 1 To nInt
        xstng = a + (i - 1) * dx
        ystng = f(xstng)
        xdrpt = a + i * dx
        ydrpt = f(xdrpt)
        IntegralaTrapez = IntegralaTrapez + (ystng + ydrpt) * dx / 2
    Next i
End Function
```

nInt reprezintă numărul de intervale, **a** și **b** limitele de integrare

3. Inserați formula simplă Integrala în Excel și modificați numărul de intervale pentru a vedea efectul asupra rezultatului.
4. Executați codul pas cu pas pentru a vedea modul de calcul al metodei