实用计算方法原理及算法实现

（VB. NET 版）

杨超　范士娟　周建民　◎　编著

西南交通大学出版社

·成　都·

内容简介

本书是针对“计算方法”“数值分析”“数值计算方法”“数值分析与算法”等课程编写的，介绍了科学与工程计算中的常用算法，重点给出了算法思想、原理、实现步骤、实现程序和验证算例。主要内容包括非线性方程根的数值解法、线性方程组的直接解法、线性方程组的迭代解法、非线性方程组的数值解法、插值方法、曲线拟合、矩阵特征值、数值积分与数值微分，以及常微分方程数值解法。每个算法都有相应的实现例程，多数算法写成了通用子程序。全书共有 45 个常用算法，所有算法示例程序都是基于可视化语言 Visual Studio 2017 的 VB. NET 编写的并进行了验证。

本书可作为高等院校理工类专业本科生及工科类硕士研究生“计算方法”或“数值分析”课程的辅助教材或教学参考书，也可供科研院所、工矿企业从事科学与工程计算的科技人员使用和参考。

前言 Preface

随着计算机技术的飞速发展及其在各个领域的广泛应用，理论科学、试验科学和计算科学并列成为现代科学活动的三种重要手段。通过计算机可以处理实际问题、分析计算误差，并与相应的理论和可能的试验进行对比和印证，因此，掌握基本的数值计算方法并能编程实现、应用，已成为理工科学生必备的技能之一。本书是在工程教育认证背景下、以成果导向教育(OBE)理念为指引编写而成的，重在培养和提高学习者自我学习和动手解决实际问题的能力。

本书的宗旨是让学习者学习并掌握常用计算方法的原理与算法步骤，并能用计算机编程实现，从而培养和提高学习者选用合适的算法来独立处理数值计算问题的能力。

作者多年来为工科本科生和工科研究生讲授“计算方法”“数值分析”和“计算方法与程序设计”课程，深感学生理解算法原理和编写算法程序的困难，所以，在介绍算法时，本书尽量简化算法的理论推导过程，重点说明算法的思想、原理、实现步骤及注意事项，侧重于学习者对算法原理的理解和掌握与算法的程序实现和应用。

全书分 9 章，共有 45 个常用算法，包括求非线性方程根的 8 个迭代算法，求解线性方程组的 6 个直接算法和 4 个迭代算法，求解非线性方程组的 6 个迭代算法，4 个插值算法，2 个曲线拟合算法，3 个求解矩阵特征值的算法，6 个求解数值积分算法和 1 个数值微分算法，5 个常微分方程数值解法。每个算法都有详细的算法步骤和例程，配有程序界面和详尽的程序代码，所有的算法例程均采用可视化编程语言 Visual Studio 2017 的 VB. NET 作为开发工具编写而成，

为便于学习和理解，所有例程都坚持简洁性、一致性、易用性、重用性和实用性原则：算法的例程界面设计尽可能简洁、美观，类似算法的界面风格尽可能保持一致；参数的输入、求解结果的输出都在程序界面上完成，简单且直观，也便于算法的对比；不同的算法程序里，参数的读取过程、计算结果的输出过程等尽可能采用相同的过程名称；类似的子程序或过程里调用参数的形式尽可能保持一致；多数算法都做成了通用子程序的形式，只需按要求传递参数，就可返回需要的数据，便于代码重用，甚至直接使用；需要输入方程或方程组表达式的地方，单独给出了子程序框架，实际应用时，只需在指定的子程序框架内输入必要的表达式即可，其余程序无须改动；设计程序时，总是尽可能采用内存占用少的方式；涉及多项式或函数的地方，都给出求解结果曲线，便于对比。这些都有利于培养学习者良好的编程习惯。

每章都留有若干上机实验题，供学习者练习。

本书 46 个算例程序都在“Windows 7+Visual Studio 2017-VB. NET”和“Windows 10+Visual Studio 2017-VB. NET”环境下调试通过。

杨超编写了第 3 章、第 4 章、第 5 章和第 9 章，范士娟副教授编写了第 2 章、第 6 章和第 8 章，周建民教授编写了第 1 章和第 7 章。

本书在成稿过程中，华东交通大学刘正平教授和江西农业大学吴彦红教授、薛龙副教授提出了许多宝贵意见，在此表示衷心感谢！

由于作者水平有限，书中难免还存在不足和疏漏，恳请读者和广大同仁多提宝贵意见，以便今后改进。

杨超　范士娟　周建民

2020 年 10 月于南昌

目 录
Contents

第 1 章　非线性方程根的数值解法

科学工程中的很多问题常常归结为求解一个非线性方程，很难或无法得到解析解，往往需要寻求数值解法。本章将给出两类数值解法：① 通过构造不断缩小的有根区间序列，当有根区间缩小到一定程度时可用有根区间内的任一点作为根的近似值；② 构造收敛于方程根 x^*的数列$\{x_k\}$，当满足一定条件时 x_k可作为根 x^*的近似值。

1.1　二分法

1.1.1　算法原理与步骤

设非线性方程 $f(x)=0$ 在区间$[a, b]$上连续，如果 $f(a)f(b)<0$，则 $f(x)$在$[a, b]$内至少有一个零点。将区间$[a, b]$二等分，计算其中点 $c=(a+b)/2$ 的函数值$f(c)$，如果 $f(c)=0$，则 c 为方程的根；如果 $f(a)f(c)<0$，则区间$[a, c]$包含方程的根；否则，方程的根在区间$[c, b]$内。将包含方程根的区间仍记为$[a, b]$，再将其二等分，并计算其中点 $c=(a+b)/2$ 的函数值$f(c)$，重复前面的判断。该过程每进行一次，解区间就缩小为原来的一半，重复此过程，直到区间长度缩小到指定的精度。二分法求解非线性方程根的过程如图 1.1 所示。

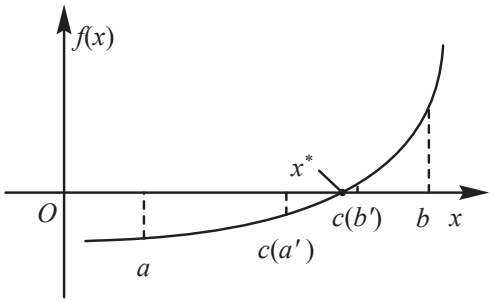

图 1.1　二分法求解过程

二分法是一种逐步搜索算法，其全局收敛，但收敛速度较慢。

用二分法求解非线性方程$f(x)$的算法步骤如下：

步骤 1：输入有根区间端点 a 和 b、解精度 $E1$ 和方程精度 $E2$、最大迭代次数 M。

步骤 2：计算$f(x)$在区间$[a, b]$端点处的函数值$f(a)$和$f(b)$。

步骤 3：若$|f(a)|<E2$或$|f(b)|<E2$，则输出 a 或 b，计算结束；否则，往下进行。

对于 $K=1, 2, \cdots, M$，执行步骤 4 至步骤 6。

步骤 4：计算 $f(x)$在区间中点 $c=(a+b)/2$ 处的值 $f(c)$。

步骤 5：若$|f(c)|<E2$，则输出 c，计算结束；若 $f(c)f(a)<0$，则方程的根在区间$[a, c]$内，令 $b=c$，$f(b)=f(c)$；否则，方程的根在区间$[c, b]$内，令 $a=c$，$f(a)=f(c)$。

步骤 6：若区间长度$|b-a|<E1$，输出$(a+b)/2$，计算结束。

步骤 7：输出“二分法已迭代 M 次，没有得到符合要求的解”，停止计算。

1.1.2 算法实现程序

二分法程序界面如图 1.2 所示。

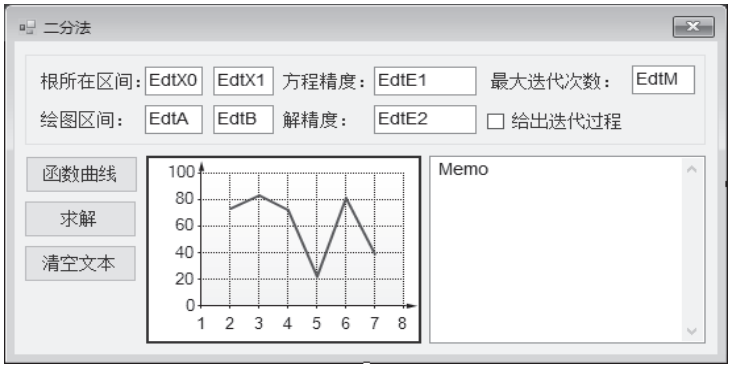

图 1.2　二分法程序界面

程序实现的主要代码如下：

```
Imports System.Math
Public Class frmMain
    Dim E1 As Double, E2 As Double, M As Integer, SS As String, ShowGuoCheng As Boolean
    Private Function Func(X As Double)
        '这里写入待求函数 f(x)的表达式
        Func = 2^(-X)-cos(X)
    End Function
    '交换变量 X 和 Y 的值
    Private Sub SwapXY(ByRef X As Double, ByRef Y As Double)
    Dim T As Double
        T = X: X = Y: Y = T
    End Sub
    '按钮 BtnClear 的 Click 事件。清除 Memo 的内容
    Private Sub BtnClear_Click(sender As Object, e As EventArgs) Handles BtnClear.Click
    Memo.Clear()
    End Sub
```

'按钮 BtnCurve 的 **Click** 事件。绘制方程函数在区间[a, b]内的图形

```
Private Sub BtnCurve_Click(sender As Object, e As EventArgs) Handles BtnCurve.Click
    Dim I As Integer, A As Double, B As Double, X As Double, Y As Double, Dx As Double
    A = Val(EdtA.Text)
    B = Val(EdtB.Text)
    If A > B Then SwapXY(A, B)
    Chart.Series(0).Points.Clear()
    Dx = (B - A) / 200
    For I = 0 To 199
        X = A + I * Dx
        Y = Func(X)
        Chart.Series(0).Points.AddXY(X, Y)
    Next
End Sub
```

'按钮 BtnResult 的 **Click** 事件。本过程调用了子程序 **SwapXY**、输出计算结果子程序 **OutputResult** 和二分法求根子程序 **ErFen**

```
Private Sub BtnResult_Click(sender As Object, e As EventArgs) Handles BtnResult.Click
    Dim X As Double, F As Double, X0 As Double, X1 As Double, K As Integer, Msg As String
    X0 = Val(EdtX0.Text): X1 = Val(EdtX1.Text)
    E1 = Val(EdtE1.Text): E2 = Val(EdtE2.Text)
    M = Val(EdtM.Text)
    ShowGuoCheng = CheckBox.Checked
    If X0> X1 Then SwapXY(X0, X1)
    Msg = ""
    SS = "求解过程："
    ErFen(X, F, K, Msg, X0, X1)
    If (ShowGuoCheng = True) And (SS <> "") Then
        Memo.Text = SS & vbCrLf
    End If
    If Msg = "" Then
        OutputResult(X, F, K)
    Else
        MessageBox.Show(Msg)
    End If
End Sub
```

'二分法求根子程序 **ErFen**。使用控制精度 E1 和 E2、最大迭代次数 M，输入根所在区间端点 X0 和 X1，返回方程的近似解 X、方程的值 F、迭代次数 K、迭代过程数据 SS 和消息 Msg

```
Private Sub ErFen(ByRef X As Double, ByRef F As Double, ByRef K As Integer,
ByRef Msg As String, X0 As Double, X1 As Double)
    Dim C As Double, Fa As Double, Fb As Double, Fc As Double, I As Integer
    Fa = Func(X0): Fb = Func(X1)
    Msg = ""
    If Abs(Fa) < E1 Then
      X = X0: F = Fa
      K = 0: SS = SS + Str(X) + ";"
      Exit Sub
    End If
    If Abs(Fb) < E1 Then
      X = X1: F = Fb
      K = 0: SS = SS + Str(X) + ";"
      Exit Sub
    End If
    For I = 1 To M
      C = (X0 + X1) / 2: Fc = Func(C)
      If Abs(C - X0)< E2 And Abs(Fc) < E1 Then
        X = C
        F = Fc
        K = I
        SS = SS + Str(X) + ";"
        Exit Sub
      End If
      If Fa * Fc < 0 Then
        X1 = C
        Fb = Fc
        SS = SS + Str(X1) + ";"
      Else
        X0 = C
        Fa = Fc
        SS = SS + Str(X0) + ";"
      End If
    Next
    X = 0
    F = 0
    K = M
    Msg = "二分法迭代了" + Str(M) + "次，没有得到符合要求的解"
```

```
End Sub
'输出求解结果子程序 OutputResult。输出根的近似值 X、方程的值 F 和迭代次数 K
  Private Sub OutputResult(X As Double, F As Double, K As Integer)
    With Memo
    .Text = .Text & "=======二分法=======" & vbCrLf
    .Text = .Text & "方程的根为：" + Str(Round(X / E2) * E2) & vbCrLf
    .Text = .Text & "方程的值为：" + Format(F, "0.####E+00") & vbCrLf
    .Text = .Text & "迭代次数：" & Str(K) & vbCrLf
    .Text = .Text & "====================" & vbCrLf
    End With
  End Sub
End Class
```

1.1.3 算例及结果

用二分法求函数 $f(x) = 2^{-x} - \cos(x)$在区间[4, 6]内的根。
参数输入及求解结果如图 1.3 所示。

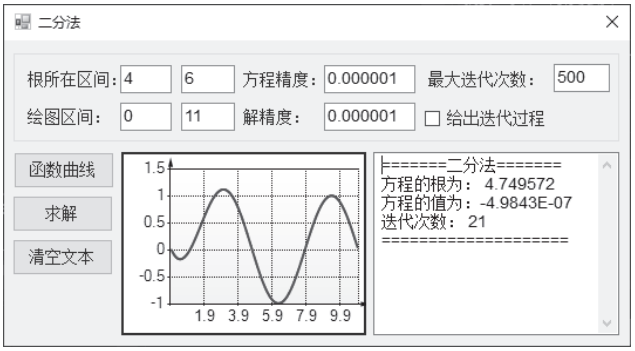

图 1.3　算例的二分法求解结果

1.2 不动点迭代法

1.2.1 算法原理与步骤

将方程 $f(x) = 0$ 变形为 $x = \varphi(x)$，先设法给出根的某个近似估计值 x_0（迭代初值），令 $x_1 = \varphi(x_0)$，并取 x_1 作为方程根新的近似值，令 $x_2 = \varphi(x_1)$，如此反复迭代，就有 $x_{k+1} = \varphi(x_k)$（$k = 0, 1, 2, \cdots$），据此，可以得到近似值序列$\{x_k\}$。如果序列$\{x_k\}$收敛，即 $x_k \to x^*$（$k \to \infty$），则 $x^* = \varphi(x^*)$。当 k 充分大时，有$|x_k - x^*| < \varepsilon$，就可得到满足事先给定精度$\varepsilon$的所求根 x^*的近似值 x_k。

求方程 $x=\varphi(x)$的根就是求曲线 $y=\varphi(x)$与直线 $y=x$ 交点的横坐标 x^*。迭代求解过程（见图 1.4）：$x_0 \to y=\varphi(x_0)(P_0$ 点$) \to x_1=\varphi(x_0) \to y=\varphi(x_1)(P_1$ 点$) \to x_2=\varphi(x_1) \to y=\varphi(x_2)(P_2$点$) \to \cdots$。当$\varphi(x)$的变化幅度小于 x 的变化幅度时，迭代公式收敛；否则，迭代公式发散。图 1.4（a）和 1.4（b）为不动点迭代法收敛示意图（x_k不断靠近 x^*），图 1.4（c）和 1.4（d）为迭代法发散示意图（x_k不断远离 x^*）。

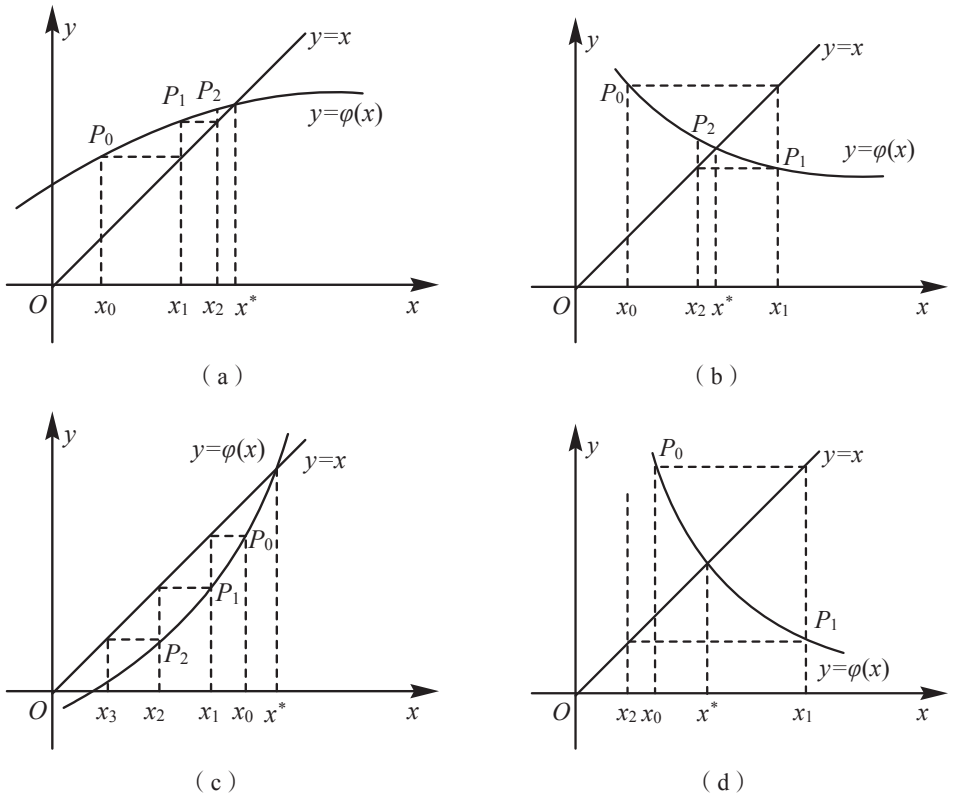

图 1.4　不动点迭代法收敛与发散示意图

用不动点迭代法求解非线性方程 $f(x)=0$ 的算法步骤如下：

步骤 1：输入迭代初值 x_0、控制精度 E、最大迭代次数 M。

步骤 2：计算 $f(x_0)$，若$|f(x_0)|<E$，则输出 x_0，计算结束；否则，往下进行。

对于 $K=1, 2, \cdots, M$，执行步骤 3 至步骤 5。

步骤 3：计算 $x_1=\varphi(x_0)$。

步骤 4：若$|x_1-x_0|<E$ 或$|x_1-f(x_1)|<E$，则输出 x_1，计算结束；否则，往下进行。

步骤 5：$x_0=x_1$，返回步骤 3。

步骤 6：输出“不动点迭代法已迭代 M 次，没有得到符合要求的解”，停止计算。

1.2.2　算法实现程序

不动点迭代法程序界面如图 1.5 所示。

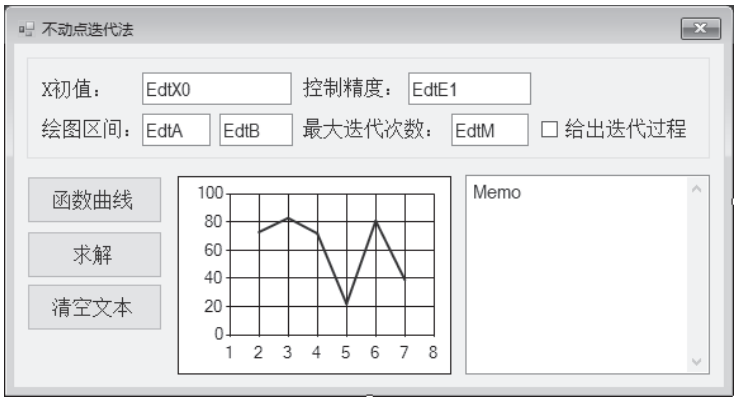

图 1.5　不动点迭代法程序界面

程序实现的主要代码如下：

```
Imports System.Math
Public Class frmMain
    Dim E1 As Double, M As Integer, SS As String, ShowGuoCheng As Boolean
    '不动点迭代函数φ(x)的表达式
    Private Function Fai(X As Double)
        Fai = X + Func(X)
    End Function
```

'按钮 BtnResult 的 **Click** 事件。本过程调用了输出计算结果子程序 **OutputResult** 和不动点迭代法求根子程序 **FixedP**

```
    Private Sub BtnResult_Click(sender As Object, e As EventArgs) Handles BtnResult.Click
        Dim X As Double, F As Double, K As Integer, Msg As String
        X = Val(EdtX0.Text)
        E1 = Val(EdtE1.Text)
        M = Val(EdtM.Text)
        ShowGuoCheng = CheckBox.Checked
        Msg = ""
        SS = "求解过程："
        FixedP(X, F, K, Msg)
        If (ShowGuoCheng = True) And (SS <> "") Then
            Memo.Text = SS & vbCrLf
        End If
        If Msg = "" Then
            OutputResult(X, F, K)
        Else
            MessageBox.Show(Msg)
        End If
```

```
    End Sub
    '不动点迭代法求根子程序 FixedP。输入和返回 X0、F、迭代次数 K 和迭代过程信息 Msg
    Private Sub FixedP(ByRef X0 As Double, ByRef F As Double, ByRef K As Integer, ByRef Msg As String)
        Dim I As Integer, X1 As Double
        Msg = ""
        F = Func(X0)
        If Abs(F) < E1 Then
            K = 0
            SS = SS + Str(X0) + ";"
            Exit Sub
        End If
        For I = 1 To M
            X1 = Fai(X0)
            F = Func(X1)
            SS = SS + Str(X1) + ";"
            If Abs(X1 - X0) < E1 Or Abs(X1 - F) < E1 Then
                K = I
                X0 = X1
                Exit Sub
            End If
            X0 = X1
        Next
        K = M
        Msg = "牛顿法迭代了" + Str(M) + "次，没有得到符合要求的解"
    End Sub
End Class
```

求函数 $f(x)$值的子程序 **Func(x)**、交换两个变量值的子程序 **SwapXY**、输出求解结果子程序 **OutputResult**、按钮 BtnCurve 的 **Click** 事件代码、按钮 BtnClear 的 **Click** 事件代码，参看“1.1 二分法”。

1.2.3 算例及结果

用不动点迭代法求函数 $f(x) = 2^{-x}-\cos(x)$在 5 附近的根。

本算例中，因所求根的近似值满足 $f(x_k)\to 0$，$x+f(x)\to x$，故取不动点迭代公式 $x = x+f(x)$，即 $\varphi(x) = x+f(x) = x+2^{-x}-\cos(x)$。参数输入及求解结果如图 1.6 所示。

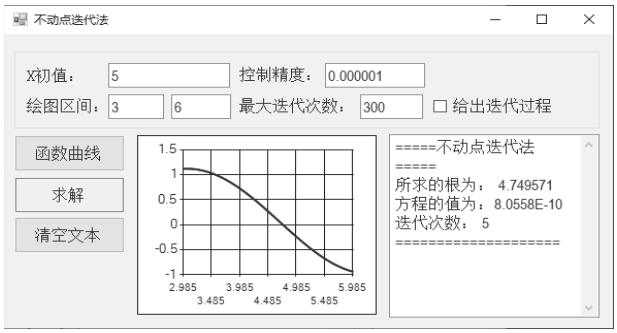

图 1.6　算例的不动点迭代法求解结果

1.3　布伦特方法

布伦特方法是求解区间[a, b]两端点处函数值异号的实函数方程 $f(x)=0$ 在[a, b]内的一个实根的方法。

1.3.1　算法原理与步骤

布伦特方法兼有二分法和反二次插值的优点，只要函数 $f(x)$在方程的有根区间内可求值，其收敛速度就比二分法快且对病态函数也能保证收敛。

设[a, b]为方程 $f(x)=0$ 的一个有根区间，即 $f(a)f(b)<0$，不妨设$|f(b)|\leqslant|f(a)|$，则算法步骤为：

（1）取 $c=a$，$f(c)=f(a)$。

（2）若 $0.5|c-b|<\varepsilon$或 $f(b)=0$，则 b 点为满足精度要求的根，计算结束；否则，执行下一步。

（3）若 $a=c$，则用二分法求出根的新的近似值 x；若 $a\neq c$，则用(a, $f(a)$)、(b, $f(b)$)、(c, $f(c)$)三点作反二次插值，得到根的新近似值：$X=b+P/Q$。其中，$P=S[T(R-T)(c-b)-(1-R)(b-a)]$，$Q=(T-1)(R-1)(S-1)$，$R=f(b)/f(c)$，$S=f(b)/f(a)$，$T=f(a)/f(c)$。

（4）将原来的 b 作为新的 a，用 x 代替原来的 b。

在上述过程中，b 总是当前根的最好的近似值，P/Q 为对 b 的一个微小修正值。当修正值 P/Q 使新的根的近似值 x 跑出了区间[c, b]，以及当有根区间用反二次插值计算衰减很慢时，用二分法求根的近似值。

（5）返回（2），重复执行上述步骤；只有当找到了满足精度要求的根或迭代次数已达到给定的最大迭代次数时，上述过程才停止。

用布伦特方法求解非线性方程 $f(x)$的算法步骤如下：

步骤 1：输入有根区间端点 a 和 b、方程精度 $E1$ 和解精度 $E2$、最大迭代次数 M。

步骤 2：计算 $f(x)$在区间[a, b]端点处的函数值 $f(a)$和 $f(b)$。

步骤 3：若$|f(a)|<E1$ 或$|f(b)|<E1$，则输出 a 或 b，计算结束；否则，往下进行。

步骤 4：若 $f(a)f(b)>0$，则用二分法找到有根区间$[a, b]$，使 $f(a)f(b)<0$。

步骤 5：若$|f(a)|<|f(b)|$，则 $a \leftrightarrow b$， $f(a) \leftrightarrow f(b)$。

步骤 6：取 $c=a$， $f(c)=f(a)$。

对于 $K=1, 2, \cdots, M$，执行步骤 7 至步骤 9。

步骤 7：若$|c-b|/2<E2$ 或 $f(b)<E1$，则 b 为满足精度要求的根，计算结束。

步骤 8：若 $a=c$，则用二分法求一根的新的近似值 X；若 $a\neq c$，则用$(a,f(a))$，$(b,f(b))$，$(c,f(c))$三点作反二次插值，得根的新的近似值：

$$R=f(b)/f(c)，\ S=f(b)/f(a)，\ T=f(a)/f(c)$$

$$P=S[T(R-T)(c-b)-(1-R)(b-a)]$$

$$Q=(T-1)(R-1)(S-1)$$

$$X=b+P/Q$$

步骤 9：$a=b$，$b=X$。

步骤 10：输出“布伦特方法已迭代 M 次，没有得到符合要求的解”，停止计算。

1.3.2 算法实现程序

布伦特方法程序界面参见图 1.2。

程序实现的主要代码如下：

```
Imports System.Math
Public Class frmMain
    Dim E1 As Double, E2 As Double, M As Integer, SS As String, ShowGuoCheng As Boolean
    '按钮 BtnResult 的 Click 事件。本过程调用了子程序 SwapXY、输出计算结果子程序 OutputResult 和布伦特方法求根子程序 Brent
    Private Sub BtnResult_Click(sender As Object, e As EventArgs) Handles BtnResult.Click
        Dim X As Double, F As Double, K As Integer, Msg As String, A As Double, B As Double, X0 As Double, X1 As Double
        X0 = Val(EdtX0.Text)
        X1 = Val(EdtX1.Text)
        E1 = Val(EdtE1.Text)
        E2 = Val(EdtE2.Text)
        M = Val(EdtM.Text)
        ShowGuoCheng = CheckBox.Checked
        If X0 > X1 Then SwapXY(X0, X1)
            Msg = ""
```

```
        SS = "求解过程："
        Brent(X, F, K, Msg, X0, X1)
        If (ShowGuoCheng = True) And (SS <> "") Then
            Memo.Text = Memo.Text + SS & vbCrLf
        End If
        If Msg = "" Then
            OutputResult(X, F, K)
        Else
            MessageBox.Show(Msg)
        End If
    End Sub
```

'布伦特方法求根子程序 **Brent**。使用控制精度 E1 和 E2、最大迭代次数 M，输入根所在区间端点 X0 和 X1，返回方程的近似解 **X**、方程的值 F、迭代次数 K 和迭代过程数据 SS 和消息 Msg

```
    Private Sub Brent(ByRef X As Double, ByRef F As Double, ByRef K As Integer, ByRef Msg As String, X0 As Double, X1 As Double)
        Dim I As Integer, A, B, C, Fa, Fb, Fc, Xm, D, E, P, Q, R, S, T, Tol, AA
        A = X0 : Fa = Func(A)
        B = X1 : Fb = Func(B)
        If Abs(Fa) < E1 Then
            X = A : F = Fa
            K = 0 : SS = SS + Str(X) + ";"
            Exit Sub
        End If
        If Abs(Fb) < E1 Then
            X = B : F = Fb
            K = 0 : SS = SS + Str(X) + ";"
            Exit Sub
        End If
        If Fa * Fb > 0 Then
        C = (A + B) / 2
            Fc = Func(C)
            If Abs(Fc) < E1 Then
        X = C : F = Fc
        K = 1 : SS = SS + Str(X) + ";"
        Exit Sub
            End If
            If Fa * Fc < 0 Then
```

```
B = C : Fb = Fc
   Else
A = C : Fa = Fc
   End If
End If
K = 1: Fc = Fb
For I = 1 To M
   If Fb * Fc > 0 Then
      C = A : Fc = Fa
      D = B - A : E = D
   End If
   If Abs(Fc) < Abs(Fb) Then
      A = B : B = C : C = A
      Fa = Fb : Fb = Fc : Fc = Fa
   End If
   Xm = 0.5 * (C - B)
   Tol = 2 * E2 * Abs(B) + 0.5 * E1
   If Abs(Xm) < E2 Or Abs(Fb) < E1 Then
      X = B: F = Fb
      SS = SS + Str(X) + ";"
      Exit Sub
   End If
   If Abs(E) > E2 And Abs(Fa) > Abs(Fb) Then
      S = Fb / Fa
      If A = C Then
         P = 2 * Xm * S
      Q = 1 - S
   Else
      Q = Fa / Fc
      R = Fb / Fc
      P = S * (2 * Xm * Q * (Q - R) + (B - A) * (R - 1))
      Q = (Q - 1) * (R - 1) * (S - 1)
   End If
   If P > 0 Then Q = -Q
   P = Abs(P)
   If 3 * Xm * Q - Abs(Tol * Q) < Abs(E * Q) Then
      AA = 3 * Xm * Q - Abs(Tol * Q)
   Else
```

```
            AA = Abs(E * Q)
        End If
        If 2 * P < AA Then
            E = D : D = P / Q
        Else
            D = Xm : E = D
        End If
    Else
        D = Xm : E = D
    End If
    A = B : Fa = Fb
    If Abs(D) > Tol Then
        B = B + D
    Else
        If Xm > = 0 Then
            B = B + Abs(Tol)
        Else
            B = B - Abs(Tol)
        End If
    End If
    X = B: Fb = Func(B)
    F = Fb:
    SS = SS + Str(X) + ";"
    K = K + 1
    Next
    K = M
    Msg = "布伦特方法迭代了" + Str(M) + "次，没有得到符合要求的解"
  End Sub
End Class
```

求函数 $f(x)$值的子程序 **Func(X)**、交换两个变量值的子程序 **SwapXY**、输出求解结果子程序 **OutputResult**、按钮 BtnCurve 的 **Click** 事件代码、按钮 BtnClear 的 **Click** 事件代码，参看“1.1 二分法”。

1.3.3 算例及结果

用布伦特方法求函数 $f(x) = 2^{-x} - \cos(x)$在区间[4, 6]内的根。

参数输入及求解结果如图 1.7 所示。可以看到，布伦特方法的收敛速度比二分法快很多。

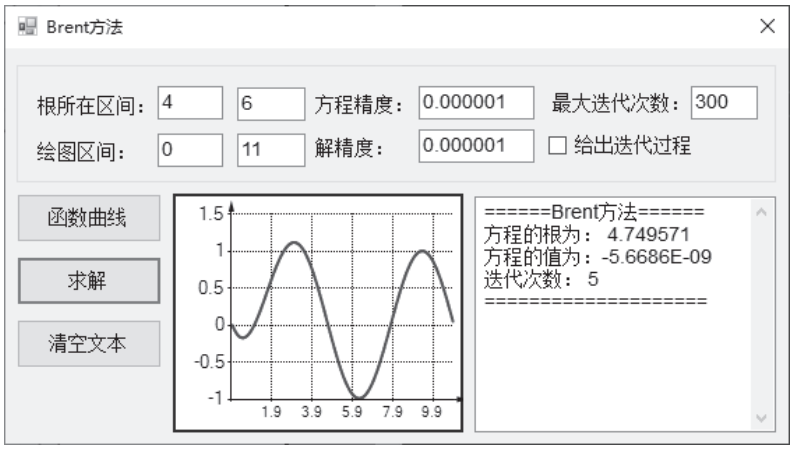

图 1.7　算例的布伦特方法求解结果

1.4　牛顿法

1.4.1　算法原理与步骤

解非线性方程 $f(x)=0$ 的牛顿法是将非线性方程逐次线性化的一种近似方法，其最大优点是在方程 $f(x)=0$ 的单根附近具有平方收敛。设 x^* 是 $f(x)=0$ 的根，选取 x_0 作为 x^* 的近似值。过点 $(x_0, f(x_0))$ 作曲线 $y=f(x)$ 的切线 L_1：$y=f(x_0)+f'(x_0)(x-x_0)$，L_1 与 x 轴交点的横坐标为 $x_1=x_0-f(x_0)/f'(x_0)$，称 x_1 为 x^* 的一次近似值；过点 $(x_1, f(x_1))$ 作曲线 $y=f(x)$ 的切线 L_2：$y=f(x_1)+f'(x_1)(x-x_1)$，L_2 与 x 轴交点的横坐标为 $x_2=x_1-f(x_1)/f'(x_1)$，称 x_2 为 x^* 的二次近似值；重复以上过程，得 x^* 的近似值序列 $\{x_k\}$。如果序列 $\{x_k\}$ 收敛，则 $x_k \to x^*$（$k \to \infty$）。

若给定 $f(x)=0$ 的有根区间 $[a, b]$，则可取其中点为 x^* 的初始近似值，即 $x_0=(a+b)/2$。

牛顿法具有二阶收敛速度，其求解非线性方程根的迭代过程如图 1.8 所示。

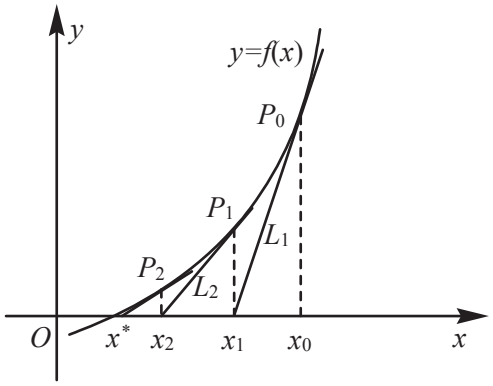

图 1.8　牛顿法迭代求解过程

牛顿法迭代公式为

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$$

用牛顿法求解非线性方程 $f(x) = 0$ 根的算法步骤如下：

步骤 1：输入方程根附近的初始解 x_0、解精度 $E1$ 和方程精度 $E2$、最大迭代次数 M。对于 $K = 1, 2, \cdots, M$，执行步骤 2 和步骤 3。

步骤 2：计算 $f(x_0)$和 $f'(x_0)$，$x_1 = x_0 - f(x_0) / f'(x_0)$。

步骤 3：若$|x_1 - x_0| < E1$ 且$|f(x_1)| < E2$，输出 x_1 和 $f(x_1)$，计算结束；否则，$x_0 = x_1$，返回到步骤 2。

步骤 4：输出“牛顿法已迭代 M 次，没有得到符合要求的解”，停止计算。

1.4.2 算法实现程序

牛顿法程序界面如图 1.9 所示。

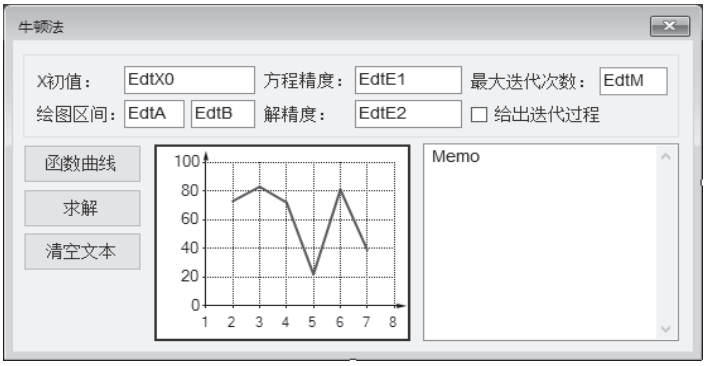

图 1.9　牛顿法程序界面

程序实现的主要代码如下：

```
Imports System.Math
Public Class frmMain
    Dim E1 As Double, E2 As Double, M As Integer, SS As String, ShowGuoCheng As Boolean
    '求函数 f(x)一阶导数的子程序 FuncP
    Private Function FuncP(X As Double)
        FuncP = –2 ^ (-X) +Sin(X)
    End Function
    '按钮 BtnResult 的 Click 事件。本过程调用了输出计算结果子程序 OutputResult 和牛顿法求根子程序 Newton
    Private Sub BtnResult_Click(sender As Object, e As EventArgs) Handles BtnResult. Click
        Dim X As Double, F As Double, K As Integer, Msg As String
```

```
    X = Val(EdtX0.Text): E1 = Val(EdtE1.Text)
    E2 = Val(EdtE2.Text): M = Val(EdtM.Text)
    ShowGuoCheng = CheckBox.Checked
    Msg = "": SS = "求解过程："
    Newton(X, F, K, Msg)
    If (ShowGuoCheng = True) And (SS <> "") Then Memo.Text = SS & vbCrLf
    If Msg = "" Then
      OutputResult(X, F, K)
    Else
      MessageBox.Show(Msg)
    End If
  End Sub
```

'牛顿法求根子程序 **Newton**。输入初始解 X、控制精度 E1 和 E2、最大迭代次数 M，返回根的近似解 X、方程的值 F、迭代次数 K、迭代过程数据 SS 和消息 Msg

```
  Private Sub Newton(ByRef X As Double, ByRef F As Double, ByRef K As Integer, ByRef Msg As String)
    Dim I As Integer, Fp, X1, X2, Fp1, Fp2, DX
    F = Func(X)
    If Abs(F) < E1 Then
      K = 0: SS = SS + Str(X) + ";"
      Exit Sub
    End If
    For I = 1 To M
      Fp = FuncP(X)
      Do While Abs(Fp) < = 0.00000001    '防止切线为水平线
        X1 = 0.99 * X: X2 = 1.01 * X
        Fp1 = FuncP(X1): Fp2 = FuncP(X2)
        If Abs(Fp1) > Abs(Fp2) Then
          X = X1: Fp = Fp1
        Else
          X = X2: Fp = Fp2
        End If
      Loop
      DX = Func(X) / Fp
      X = X – DX: F = Func(X)
      SS = SS + Str(X) + ";"
      If Abs(DX) < E2 And Abs(F) < E1 Then
        K = I
```

```
            Exit Sub
        End If
    Next
    K = M
    Msg = "牛顿法迭代了" + Str(M) + "次，没有得到符合要求的解"
End Sub
End Class
```

求函数 $f(x)$值的子程序 **Func(X)**、交换两个变量值的子程序 **SwapXY**、输出求解结果子程序 **OutputResult**、按钮 BtnCurve 的 **Click** 事件代码、按钮 BtnClear 的 **Click** 事件代码，参看“1.1 二分法”。

1.4.3 算例及结果

用牛顿法求函数 $f(x) = 2^{-x}-\cos(x)$在 5 附近的根。

参数输入及求解结果如图 1.10 所示。图 1.10 所示为包含迭代中间过程的求解结果。

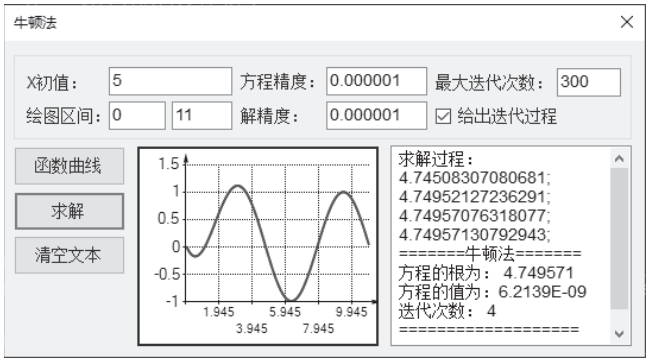

图 1.10　算例的牛顿法求解结果

1.5 牛顿下山法

1.5.1 算法原理与步骤

牛顿法只有局部收敛性，采用牛顿法求解非线性方程的根时，对初始值的选取要求较高，如果初始解近似值 x_0 偏离准确解 x^*较远，则牛顿法可能发散。牛顿下山法是牛顿法的一种变形，它是为减弱牛顿法对初始解近似值 x_0 的限制而提出的一种算法。

牛顿下山法迭代公式为

$$x_{k+1} = x_k - \lambda \frac{f(x_k)}{f'(x_k)}$$

其中，λ 为下山因子，由条件$|f(x_{k+1})|<|f(x_k)|$确定。如果条件$|f(x_{k+1})|<|f(x_k)|$成立，则

取 $\lambda = 1$(此时为牛顿法,可保证较快的收敛速度);否则,λ 减半,直到条件 $|f(x_{k+1})| < |f(x_k)|$ 成立。本方法的迭代序列是大范围收敛的，但总体收敛速度是线性的。

若给定 $f(x) = 0$ 的有根区间$[a, b]$，则可取其中点为 x^*的初始近似值，即 $x_0 = (a+b)/2$。

用牛顿下山法求解非线性方程 $f(x) = 0$ 根的算法步骤如下：

步骤 1：输入方程根附近的初始解 x_0、解精度 $E1$ 和方程精度 $E2$、最大迭代次数 M。

对于 $K = 1, 2, \cdots, M$，执行步骤 2 至步骤 5。

步骤 2：计算 $f(x_0)$和 $f'(x_0)$，取 $\lambda = 1$。

步骤 3：计算 $x_1 = x_0 - \lambda f(x_0) / f'(x_0)$ 和 $f(x_1)$。

步骤 4：若$|f(x_1)| \geqslant |f(x_0)|$，则 $\lambda = 0.5\lambda$，返回到步骤 3；否则，向下进行。

步骤 5：若$|x_1 - x_0| < E1$ 且$|f(x_1)| < E2$，输出 x_1 和 $f(x_1)$，计算结束；否则，$x_0 = x_1$，返回到步骤 2。

步骤 6：输出“牛顿下山法已迭代 M 次，没有得到符合要求的解”，停止计算。

1.5.2 算法实现程序

牛顿下山法程序界面参见图 1.9。

程序实现的主要代码如下：

```
Imports System.Math
Public Class frmMain
    Dim E1 As Double, E2 As Double, M As Integer, SS As String, ShowGuoCheng As Boolean
    '按钮 BtnResult 的 Click 事件。本过程调用了子程序 GetParameters、输出计算结果子程序 OutputResult 和牛顿法求根子程序 NewtonDown
    Private Sub BtnResult_Click(sender As Object, e As EventArgs) Handles BtnResult. Click
        Dim X As Double, F As Double, K As Integer, Msg As String
        X = Val(EdtX0.Text)
        E1 = Val(EdtE1.Text)
        E2 = Val(EdtE2.Text)
        M = Val(EdtM.Text)
        ShowGuoCheng = CheckBox.Checked
        Msg = "": SS = "求解过程："
        NewtonDown(X, F, K, Msg)
        If (ShowGuoCheng = True) And (SS <> "") Then
            Memo.Text = SS & vbCrLf
        End If
        If Msg = "" Then
            OutputResult(X, F, K)
        Else
```

```
        MessageBox.Show(Msg)
    End If
End Sub
```

'牛顿下山法求根子程序 **NewtonDown**。输入初始解 X、控制精度 E1 和 E2、最大迭代次数 M，返回根的近似解 X、方程的值 F、迭代次数 K 和迭代过程数据 SS 和消息 Msg

```
Private Sub NewtonDown(ByRef X As Double, ByRef F As Double, ByRef K As Integer, ByRef Msg As String)
    Dim I As Integer, Fp, X1, X2, Fp1, Fp2, DX, Lmd, F0
    Msg = ""
    F = Func(X)
    If Abs(F) < E1 Then
        K = 0
        SS = SS + Str(X) + ";"
        Exit Sub
    End If
    For I = 1 To M
        Fp = FuncP(X)
        Do While Abs(Fp) < = 0.00000001    '防止切线为水平线
            X1 = 0.99 * X: X2 = 1.01 * X
            Fp1 = FuncP(X1): Fp2 = FuncP(X2)
            If Abs(Fp1) > Abs(Fp2) Then
                X = X1: Fp = Fp1
            Else
                X = X2: Fp = Fp2
            End If
        Loop
        F0 = Func(X): DX = F0 / Fp
        Lmd = 1
    1:X = X - Lmd * DX
        F = Func(X)
        If Abs(F) > = Abs(F0) Then
            Lmd = 0.5 * Lmd
            GoTo 1
        End If
        SS = SS + Str(X) + ";"
        If Abs(DX) < E2 And Abs(F) < E1 Then
            K = I
```

```
            Exit Sub
        End If
    Next
    K = M
    Msg = "牛顿下山法迭代了" + Str(M) + "次，没有得到符合要求的解"
End Sub
End Class
```

求函数 $f(x)$值的子程序 **Func(X)**、交换两个变量值的子程序 **SwapXY**、输出求解结果子程序 **OutputResult**、按钮 BtnCurve 的 **Click** 事件代码、按钮 BtnClear 的 **Click** 事件代码，参看“1.1 二分法”；求函数 $f(x)$一阶导数的子程序 **FuncP**，参看“1.4 牛顿法”。

1.5.3 算例及结果

用牛顿下山法求函数 $f(x) = 2^{-x} - \cos(x)$在 5 附近的根。

参数输入及求解结果如图 1.11 所示。图 1.11 所示为包含迭代中间过程的求解结果。

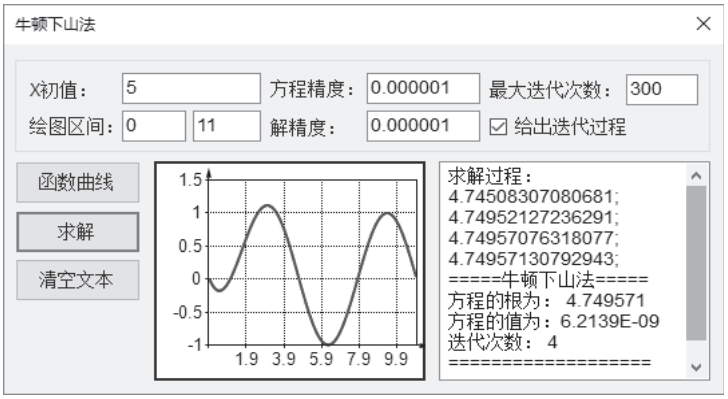

图 1.11　算例的牛顿下山法求解结果

1.6 牛顿-拉斐森法

1.6.1 算法原理与步骤

牛顿-拉斐森法是牛顿法和二分法的结合。牛顿法迭代公式为 $x_{k+1} = x_k - f(x_k)/f'(x_k)$，当初始解近似值 x_0距离准确解 x^*较近时，牛顿法具有二阶收敛速度；反之，当初始解近似值 x_0偏离准确解 x^*较远时，则牛顿法可能发散。设 $f(x) = 0$ 的有根区间为$[a, b]$，取其中点为 x^*的初始近似值，即 $x_0 = (a+b)/2$。设已求得 x^*的一个近似值 x_k，用牛顿迭代公式求出 x_{k+1}，如果 x_{k+1}不在有根区间内或区间长度缩小的过慢时，改用二分法求 x_{k+1}。迭代过程一直进行到迭代次数达到给定的最大迭代次数或求得满足精度要求的根为止。

x_{k+1} 在有根区间外可表示为 $DX = |x_{k+1} - x_k| = |f(x_k)/f'(x_k)| > 0.5DX0$，其中 $DX0$ 为原有根区间的长度，DX 为 x^*新的近似值与原近似值之间的距离。

用牛顿-拉斐森法求解非线性方程 $f(x) = 0$ 根的算法步骤如下：

步骤 1：输入有根区间端点 a 和 b、方程精度 $E1$ 和解精度 $E2$、最大迭代次数 M。

步骤 2：计算 $f(x)$在区间$[a, b]$端点处的函数值 $f(a)$和 $f(b)$。

步骤 3：若 $|f(a)| < E1$或 $|f(b)| < E1$，则输出 a 或 b，计算结束；否则，往下进行。

步骤 4：计算 $f(x)$在区间中点 $x = (a+b)/2$ 处的值 $f(x)$和 $f'(x)$。

步骤 5：若$|f(x)|<E1$，则输出 x 和 $f(x)$，计算结束；否则，确定新的有根区间端点 x_L，x_H，且 $f(x_L)<0$，计算 $DX = x_H - x_L$，$DX0 = DX$，往下进行。

对于 $K = 1, 2, \cdots, M$，执行步骤 6 至步骤 7。

步骤 6：若 x 在新的有根区间之内，且收敛速度较快，用牛顿法迭代求出新的 x、$DX = f(x)/f'(x)$，若$|DX|<E2$，则输出 x 和 $f(x)$，计算结束；否则，对原有根区间采用二分法，求出新的 $x = 0.5(x_H+x_L)$、$DX = 0.5(x_H - x_L)$，若$|DX|<E2$ 或$|f(x)|<E1$，则输出 x 和 $f(x)$，计算结束，否则，向下进行。

步骤 7：$DX0 = DX$，确定新的有根区间端点 x_L, x_H，且 $f(x_L)<0$。

步骤 8：输出“牛顿-拉斐森法已迭代 M 次，没有得到符合要求的解”，停止计算。

1.6.2 算法实现程序

牛顿-拉斐森法程序界面参看图 1.2。程序实现的主要代码如下：

```
Imports System.Math
Public Class frmMain
    Dim E1 As Double, E2 As Double, M As Integer, SS As String, ShowGuoCheng As Boolean
    '按钮 BtnResult 的 Click 事件。本过程调用了子程序 GetParameters、输出计算结果子程序 OutputResult 和牛顿法求根子程序 NewtonDown
    Private Sub BtnResult_Click(sender As Object, e As EventArgs) Handles BtnResult.Click
        Dim X As Double, F As Double, K As Integer, Msg As String
        X = Val(EdtX0.Text)
        E1 = Val(EdtE1.Text)
        E2 = Val(EdtE2.Text)
        M = Val(EdtM.Text)
        ShowGuoCheng = CheckBox.Checked
        Msg = "": SS = "求解过程："
        NewtonDown(X, F, K, Msg)
        If (ShowGuoCheng = True) And (SS <> "") Then Memo.Text = SS & vbCrLf
        If Msg = "" Then
```

```
            OutputResult(X, F, K)
        Else
            MessageBox.Show(Msg)
        End If
    End Sub
```

'牛顿-拉斐森法求根子程序 **NewtonRaphson**。输入初始解 X、控制精度 E1 和 E2、最大迭代次数 M，返回根的近似解 X、方程的值 F、迭代次数 K、迭代过程数据 SS 和消息 Msg

```
    Private Sub NewtonRaphson (ByRef X As Double, ByRef F As Double, ByRef K As Integer, ByRef Msg As String)
        Dim I As Integer, FL, FH, XL, XH, DX, Fp, DXOLD, Tem
        FL = Func(X0) : FH = Func(X1)
        If Abs(FL) < E1 Then
            X = X0 : F = FL
            K = 0 : SS = SS + Str(X) + ";"
            Exit Sub
        End If
        If Abs(FH) < E1 Then
            X = X1 : F = FH
            K = 0 : SS = SS + Str(X) + ";"
            Exit Sub
        End If
        If FL * FH > 0 Then
            X = 0.5 * (X0 + X1)
            F = Func(X)
            If Abs(F) < E1 Then
                K = 0 : SS = SS + Str(X) + ";"
                Exit Sub
            End If
            If FL * F > 0 Then
                X0 = X : FL = F
            Else
                X1 = X : FH = F
            End If
        End If
        If FL < 0 Then
            XL = X0 : XH = X1
        Else
```