ESCOLA POLITÉCNICA
DA
UNIVERSIDADE DE SÃO PAULO

DEPARTAMENTO DE ENGENHARIA DE ENERGIA
E AUTOMAÇÃO ELÉTRICAS

PROJETO DE FORMATURA / 2001
Relatório Final

Programa para Cálculo de Estratificação de Solo em Duas Camadas

Alunos: Fabiano Zemella Marques
Paulo Dessen Siqueira Brito
Orientador: Prof. Dr. Aderbal de Arruda Penteado Jr.
Prof. Dr. José Aquiles Baesso Grimone
Coordenador: Prof. Dr. Marco Antônio Saidel

SUMÁRIO

	PAGÍNA
1 – INTRODUÇÃO	04
2 – OBJETIVO	05
5.3 – DESCRIÇÃO DA TEORIA	06
5.3.1 – Resistividade Elétrica do Solo	06
5.3.1.1 – Características da Resistividade	06
5.3.1.2 – Medição da Resistividade	06
5.3.2 – Estratificação do solo em camadas	09
5.3.2.1 – Tratamento dos Dados	09
5.3.2.2 – Método de Estratificação	10
5.3.2.3 – Determinação da Curva ρ x a	11

5.3.2.4 – Equação Fundamental da Estratificação do Solo em Duas Camadas 11

5.3.2.5 – Método de Estratificação do Solo em Duas Camadas Usando Técnica de Otimização 14

5.4 – DESENVOLVIMENTO DO SOFTWARE 15

5.4.1 – Linguagem do Programa 15

5.4.2 – Entrada de Dados da Medição 15

5.4.3 – Análise das medidas 17

5.4.4 – Interpolação 18

5.4.5 – Estratificação em Duas Camadas 19

5.5 – ANEXOS 22

5.5.1 – Anexo 1 – Documentação do Software 22

1 – INTRODUÇÃO

Sabe-se que para um Sistema de Energia Elétrica operar corretamente, com uma adequada continuidade de serviços, com um desempenho seguro do sistema de proteção e, ainda poder garantir os níveis de segurança pessoal, é fundamental um adequado projeto de Sistema de Aterramento.

Os principais objetivos do aterramento são:

- Obter uma resistência de aterramento a mais baixa possível, para correntes de falta à terra;
- Manter os potenciais produzidos pelas correntes de falta dentro de limites de segurança de modo a não causar fibrilação do corpo humano;
- Fazer com que os equipamentos de proteção sejam mais sensibilizados e isolem rapidamente as faltas à terra;
- Proporcionar um caminho à terra para as descargas atmosféricas;
- Usar a terra como retorno de corrente para os sistemas MRT;
- Escoar as cargas estáticas geradas nas carcaças dos equipamentos para a terra.

São conhecidas varias maneiras para fazer o aterramento de um sistema elétrico, que vão desde uma simples haste, passando por placas de formas e tamanhos diversos, chegando às mais complicadas configurações de cabos enterrado no solo (malhas de aterramento).

Para se fazer um projeto de aterramento são necessários a obtenção de alguns dados e alguns parâmetros:

- Definir o local de aterramento;
- Providenciar vária medições no local;

- Fazer a estratificação do solo nas sua diversas camadas;
- Definir o tipo de sistema de aterramento desejado;
- Calcular a resistência aparente do solo para o respectivo sistema de aterramento;
- Dimensionar o sistema de aterramento, levando em conta a sensibilidade dos relés e os limites de segurança pessoal.

2 – OBJETIVO

Pelo que foi exposto acima verifica que o projeto de aterramento não é algo simples de ser feito. Sabe-se que nos dias de hoje não se encontra, ou não são de fácil acesso, programas destinados a calcular os parâmetros de um sistema de aterramento.

Mediante isto, este projeto teve a finalidade de desenvolver um software que permita o cálculo inicialmente da estratificação do solo em duas camadas e posteriormente que o programa possa ser ampliado para poder-se calcular a estratificação do solo em mais de duas camadas, quando necessário, pelo Método de Pirson e poder se calcular os parâmetros de uma Malha de Aterramento.

Este relatório tem a finalidade de apresentar um pouco sobre a teoria de aterramento utilizada para confecção do programa, bem como toda a maneira como o programa foi confeccionado.

5.3 – DESCRIÇÃO DA TEORIA

5.3.1 – Resistividade Elétrica do Solo

5.3.1.1 – Características da Resistividade

Vários fatores como tipo de solo, teor de umidade, temperatura, compactação e pressão, composição química e concentração de sais dissolvidos na água, influenciam na resistividade elétrica do solo. A combinação destes fatores resultam em solos com características diferentes, consequentemente com valores de resistividade distintos.

A maioria dos solos não são homogêneos, e sim formados por diversas camadas de resistividade e profundidade diferentes. Devido a formação geológica, estas camadas são geralmente horizontais e paralelas a superfície. Em alguns casos as camadas se encontram inclinadas e até verticais ao solo, devido a alguma falha geológica, no entanto os estudos apresentados para pesquisa do perfil do solo as consideram aproximadamente horizontais.

5.3.1.2 – Medição da Resistividade

Os métodos de medição de resistividade do solo são resultados da análise das características práticas das equações do eletromagnetismo de Maxwell, aplicadas ao solo.

Definido o local onde será instalado o sistema de aterramento, lugar este que depende da posição estratégica ocupada pelos equipamentos mais importantes

do sistema elétrico, deve-se efetuar medições no campo de forma a poder-se determinar a curva ρ x a (resistividade x espaçamento) característica do solo, sobre a qual são aplicados os métodos de estratificação do solo. Para efetuar estas medições, utiliza-se métodos de prospecção geoelétricos, sendo que o mais conhecido e utilizado é o método de Wenner.

O método de Wenner consiste em se cravar 4 hastes alinhadas, igualmente espaçadas e com mesma profundidade, no solo (figura 1). Um corrente elétrica I é injetada na haste A e coletada na haste D, de modo que esta corrente passando pelo solo entre os pontos A e D, produz potencial entre os pontos B e C.

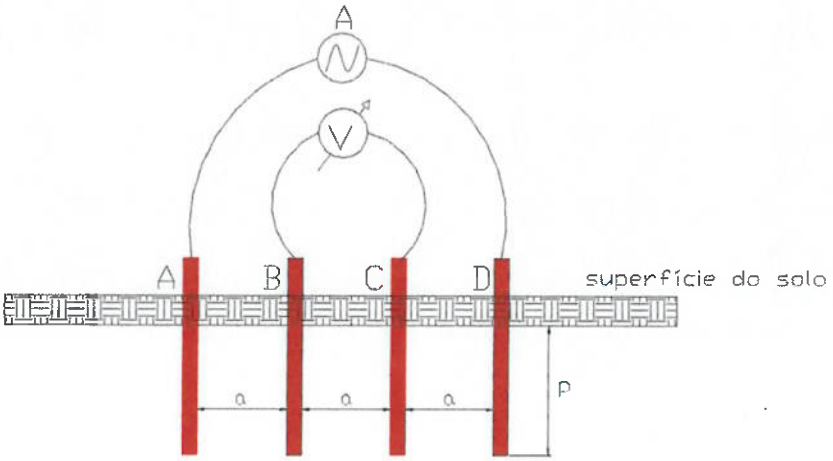

Figura 1

Utilizando o método de imagens [4], obtêm-se os potenciais nos pontos B e C, que são dados pelas duas equações seguintes:

$$V_B = \frac{\rho I}{4\pi}\left[\frac{1}{a} + \frac{1}{\sqrt{a^2+(2p)^2}} - \frac{1}{2a} - \frac{1}{\sqrt{(2a^2)+(2p)^2}}\right] \quad (1)$$

$$V_C = \frac{\rho I}{4\pi}\left[\frac{1}{2a} + \frac{1}{\sqrt{(2a)^2+(2p)^2}} - \frac{1}{a} - \frac{1}{\sqrt{a^2+(2p)^2}}\right] \quad (2)$$

Sendo a diferença de potencial entre os dois pontos dada por:

$$V_{BC} = \frac{\rho I}{4\pi}\left[\frac{1}{a} + \frac{2}{\sqrt{a^2+(2p)^2}} - \frac{2}{\sqrt{(2a)^2+(2p)^2}}\right] \quad (3)$$

Obtendo portanto a diferença de potencial entre os pontos B e C gerada pela corrente I, tem-se portanto a resistência elétrica do solo R, dada por (4), para uma profundidade proporcional a distância entre as hastes.

$$R = \frac{V_{BC}}{I} = \frac{\rho}{4\pi}\left[\frac{1}{a} + \frac{2}{\sqrt{a^2+(2p)^2}} - \frac{2}{\sqrt{(2a)^2+(2p)^2}}\right] \quad (4)$$

Através de (4), pode-se obter o valor da resistividade elétrica do solo através da fórmula de Palmer.

$$\rho = \frac{4\pi a R}{1 + \frac{2a}{\sqrt{a^2+(2p)^2}} - \frac{2a}{\sqrt{(2a)^2+(2p)^2}}} \quad (5)$$

O número de direções e espaçamentos entre hastes dependem da importância do local do aterramento, da dimensão do sistema de aterramento e da variação

acentuada nos valores medidos para os respectivos espaçamentos. Durante as medições devem ser tomados os seguintes cuidados:

- As hastes devem ser alinhadas e igualmente espaçadas;
- As hastes dever estar cravadas no solo a uma mesma profundidade, sendo recomendado de 20 a 30 cm;
- O aparelho deve estar posicionado simetricamente as hastes;
- As hastes devem estar limpas, para possibilitar bom contato com o solo;
- O estado de umidade do solo deve ser anotado durante a medição;
- Não devem ser feitas medições sobre condições de tempo adversas, devido a possibilidade de ocorrência de descargas atmosféricas;
- Verificar a integridade dos equipamentos utilizados;
- Utilizar equipamentos de segurança, como botas e luvas de isolação;
- Isolar a área deforma a evitar a presença de pessoas ou animais.

Para uma dada medição e preferível utilizar os valores de espaçamento padrões entre hastes, que são 50, 100, 200, 400, 800, 1600, 3200 e 6400 cm.

5.3.2 – Estratificação do solo em camadas

5.3.2.1 – Tratamento dos Dados

Antes de se iniciar os cálculos de estratificação, é necessário um tratamento do dados obtidos durante a medição para que os mesmos possam ser avaliados quanto a sua aceitação ou não.

O tratamento dos dados é feito de forma a se avaliar para cada espaçamento se os desvios de uma dada resistividade/resistência em relação a média aritmética da resistividades/resistências é maior que 50%. Se isto ocorrer, deve-se descartar este valor de resistividade/resistência.

O cálculo da média das resistividades/resistências para um determinado espaçamento e dado por:

$$\rho_M(a_j) = \frac{1}{n}\sum_{i=1}^{n}\rho_i(a_j) \qquad \forall\ j=1,q \text{ e } i=1,n \qquad (6)$$

Sendo o cálculo da porcentagem do desvio de cada medida em relação ao valor médio dado por:

$$\frac{\left|\rho_i(a_j) - \rho_M(a_j)\right|}{\rho_M(a_j)} * 100 \geq 50\% \qquad (7)$$

Após este tratamento de dados, tem-se então os valores de resistividade/resistência por espaçamento prontos para o estudo de estratificação do solo em camadas.

5.3.2.2 – Método de Estratificação

Existem alguns métodos de estratificação do solo em duas camadas, mas por se tratar de um software, e deste modo, pode-se trabalhar com algoritmos numéricos, optou-se por utilizar técnica de otimização.

5.3.2.3 – Determinação da Curva ρ x a

Os medidores utilizados para este tipo de medição fornecem os valores de resistência ou resistividade do solo. Quando o equipamento fornece os valores de resistividade do solo, pode-se diretamente trabalhar com estes dados, se o equipamento fornecer os valores de resistência do solo, deve-se primeiramente calcular os valores de resistividade a partir da fórmula (5) para poder-se então prosseguir com a estratificação.

Para se estratificar o solo em duas camadas utilizando algum tipo de método é necessário uma curva de resistividade x espaçamento (ρ x a). De posse dos valores de resistividade previamente calculados, pode-se traçar uma curva que rege os pontos obtidos em campo.

Esta curva fornecerá um valor aproximado de ρ(a), para qualquer valor de "a", que seria obtido na medição de campo.

5.3.2.4 – Equação Fundamental da Estratificação do Solo em Duas Camadas

Uma corrente elétrica I entrando no ponto A de um solo de duas camadas, figura 2, gera potenciais na primeira camada, que devem satisfazer a equação (8), conhecida como equação de Laplace.

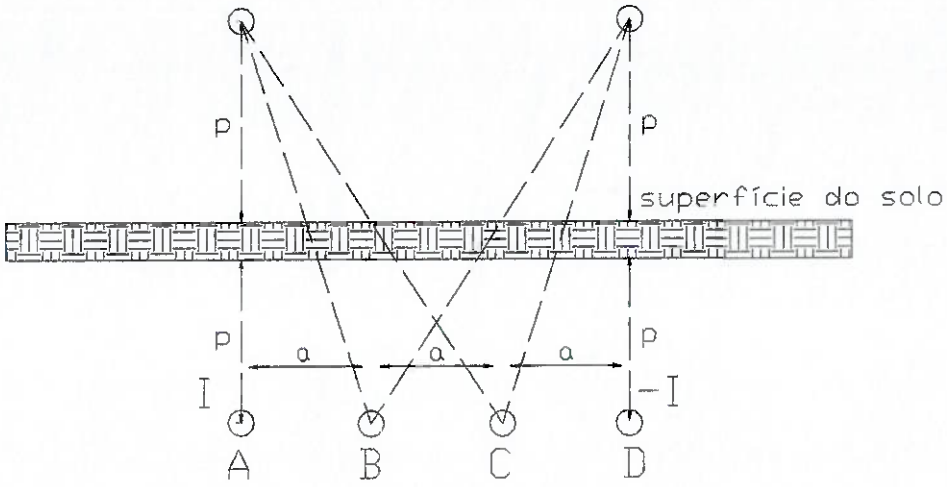

Figura 1

$$\nabla^2 V = 0 \qquad (8)$$

Desenvolvendo esta equação relativamente ao potencial V de qualquer ponto "p" da primeira camada do solo, distanciando de "r" da fonte de corrente situada no ponto A, chega-se a seguinte expressão:

$$V_p = \frac{Ip_1}{2\pi}\left[\frac{1}{r} + 2\sum_{n=1}^{\infty}\frac{K^n}{\sqrt{r^2 + (2nh)^2}}\right] \qquad (9)$$

Onde:

Vp é o potencial de um ponto p qualquer da primeira camada em relação ao infinito.

ρ1 é a resistência da primeira camada.

h é a profundidade da primeira camada.

r é a distância do ponto p à fonte de corrente A.

K é o coeficiente de reflexão, definido por (10):

$$K = \frac{\rho_2 - \rho_1}{\rho_2 + \rho_1} \qquad (10)$$

A expressão (9) definida acima aplicada na configuração de Wenner, para um solo de duas camadas resulta nos potenciais dos pontos B e C, (11) e (12).

$$V_B = \frac{I\rho_1}{2\pi}\left[\frac{1}{a} + 2\sum_{n=1}^{\infty}\frac{K^n}{\sqrt{a^2 + (2nh)^2}}\right] - \frac{I\rho_1}{2\pi}\left[\frac{1}{2a} + 2\sum_{n=1}^{\infty}\frac{K^n}{\sqrt{(2a)^2 + (2nh)^2}}\right] \qquad (11)$$

$$V_C = \frac{I\rho_1}{2\pi}\left[\frac{1}{2a} + 2\sum_{n=1}^{\infty}\frac{K^n}{\sqrt{(2a)^2 + (2nh)^2}}\right] - \frac{I\rho_1}{2\pi}\left[\frac{1}{a} + 2\sum_{n=1}^{\infty}\frac{K^n}{\sqrt{a^2 + (2nh)^2}}\right] \qquad (12)$$

Com estas duas expressões pode se chegar a diferença de potencial entre estes pontos, (13). E sabendo que VBC/I é o valor da resistência elétrica (R) obtida na medição em campo obtém-se (14):

$$V_{BC} = \frac{I\rho_1}{2\pi a}\left\{1 + 4\sum_{n=1}^{\infty}\left[\frac{K^n}{\sqrt{1 + (2n\frac{h}{a})^2}} - \frac{K^n}{\sqrt{4 + (2n\frac{h}{a})^2}}\right]\right\} \qquad (13)$$

$$2\pi aR = \rho_1\left\{1 + 4\sum_{n=1}^{\infty}\left[\frac{K^n}{\sqrt{1 + (2n\frac{h}{a})^2}} - \frac{K^n}{\sqrt{4 + (2n\frac{h}{a})^2}}\right]\right\} \qquad (14)$$

Através de uma aproximação da equação (5), tem-se que $\rho(a) = 2.\pi.a.R$ e substituindo em (14) tem-se:

$$\frac{\rho_a}{\rho_1} = 1 + 4\sum_{n=1}^{\infty}\left[\frac{K^n}{\sqrt{1+(2n\frac{h}{a})^2}} - \frac{K^n}{\sqrt{4+(2n\frac{h}{a})^2}}\right] \qquad (15)$$

Sendo esta ultima a expressão fundamental na elaboração da estratificação do solo em duas camadas.

5.3.2.5 – Método de Estratificação do Solo em Duas Camadas Usando Técnica de Otimização

A expressão (15) pode ser colocada na seguinte forma:

$$\rho_a = \rho_1\left\{1 + 4\sum_{n=1}^{\infty}\left[\frac{K^n}{\sqrt{1+(2n\frac{h}{a})^2}} - \frac{K^n}{\sqrt{4+(2n\frac{h}{a})^2}}\right]\right\} \qquad (16)$$

Esta expressão para um solo de duas camadas apresenta uma relação direta entre os espaçamentos entre hastes "a" da configuração de Wenner e o respectivo valor de ρ(a).

Em campo obtém-se valores de ρ para um determinado espaçamento entre hastes "a". Portanto os valores obtidos em campo para "a" e ρ(a) devem ser o mesmo que os obtidos pela formula (16).

A partir disto procura-se obter por técnicas de otimização o melhor solo em duas camadas regido pela curva (16) que mais se aproxima dos valores medidos. Para isto, deve-se ajustar a expressão variando ρ1, K e h de modo a

minimizar os valores de $\rho(a)$ medidos em campo dos $\rho(a)$ fornecidos pela expressão.

Uma alternativa para simplificar a otimização é achar inicialmente o valor da resistividade da primeira camada ($\rho 1$) através do prolongamento do polinômio interpolado até o eixo das abcissas, isto é, calcular o valor do polinômio para a distância igual a 0.

5.4 – DESENVOLVIMENTO DO SOFTWARE

5.4.1 – Linguagem do Programa

O software foi desenvolvido na linguagem de programação Visul Basic, proporcionando ao usuário um programa familiar aos disponíveis hoje no mercado pela Microsoft.

Os códigos de programação utilizados neste programa são em Basic.

5.4.2 – Entrada de Dados da Medição

Os dados de entrada do programa para que possa ser calculada a estratificação do solo em camadas são as resistências do solo por espaçamento, obtidas através de medições em campo.

O usuário pode configurar, frame configurações, a tabela de entrada dos valores de resistência do solo conforme a medição efetuada, isto é, ele pode

configurar o número de espaçamentos feitos entre as haste e o número de direções em cada ponto, como pode ser visto na figura 1.

Após ter configurado o grid de entrada de dados conforme sua necessidade, o usuário poderá entrar com os dados da medição efetuadas em campo no frame Coleta de Dados, o que pode ser visualizado na figura 2.

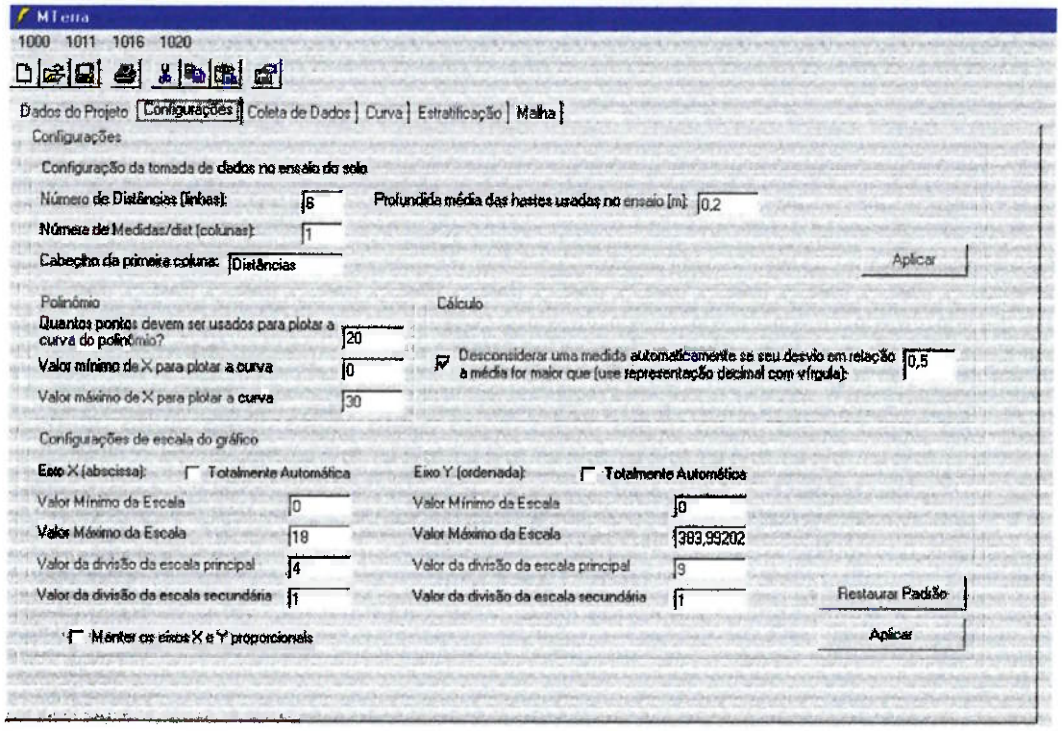

Figura 1

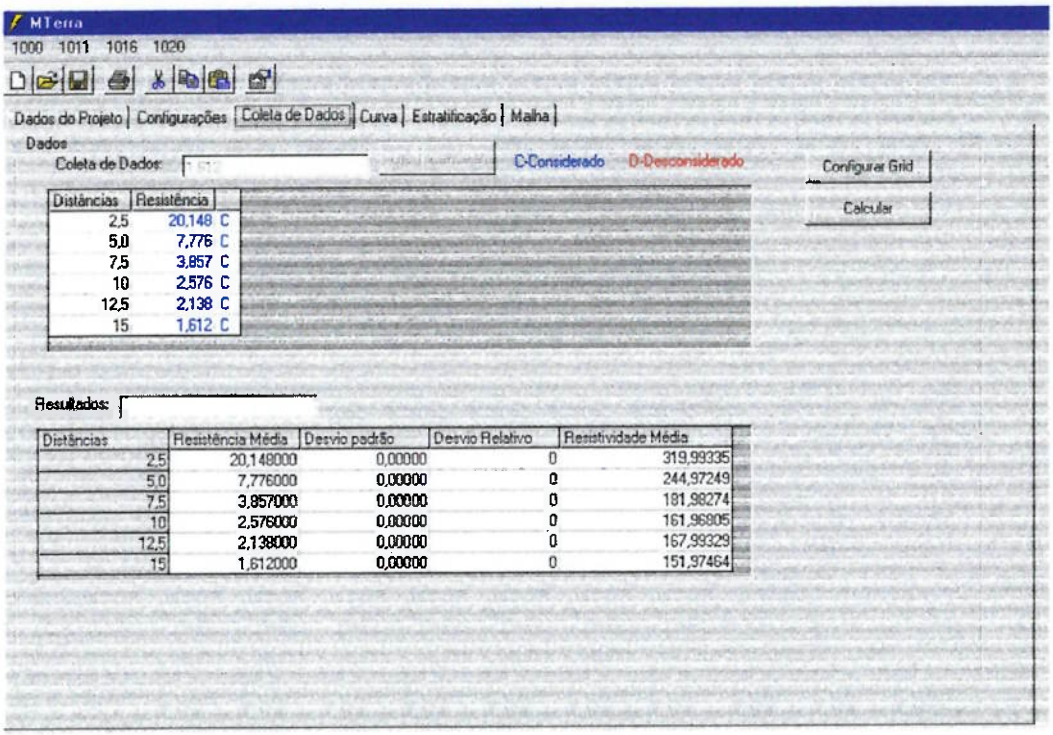

Figura 2

5.4.3 – Análise das medidas

De posse dos dados da medição, o programa faz automaticamente um tratamento dos dados conforme o procedimento descrito no item 5.3.2.1.

O programa permite ao usuário escolher, no frame configurações, o valor em porcentagem para os quais as medidas deverão ser desconsideradas automaticamente, sendo além destes cálculos automáticos, o usuário poderá selecionar, no frame coleta de dados, as medidas que ele deseje considerar ou desconsiderar.

Após os cálculos o programa fornece, no frame coleta de dados, os valores de resistência média calculada, os valores de desvios e os valores de resistividade média, sendo que este último e obtido a partir da fórmula (5).

5.4.4 – Interpolação

Primeiramente, são plotados pontos que representam os valores de resistividade calculados versos as suas respectivas distâncias, sendo estes posteriormente ligados por segmentos de reta de modo a formar uma curva. Para a estratificação do solo em camadas é necessário uma curva que modele estes pontos e se aproxime, da melhor forma possível, a curva formada por eles.

Para se obter uma curva que mais se aproxima dos pontos plotados utilizou-se interpolação polinomial pelo Método dos Mínimos Quadrados.

Este método gera um sistema de equações lineares resolvido através do Método de Gauss-Jordan. A solução deste sistema nós dá os valores dos coeficientes do polinômio.

Este polinômio encontrado é então plotado no mesmo local onde se encontra a curva dos "pontos medidos" para comparação.

O programa fornece a opção de altera os coeficientes calculados deste polinômio, de forma a melhorar o ajuste entre as curvas.

Estes procedimentos são efetuados no frame curva, e podem ser visualizados na figura 3.

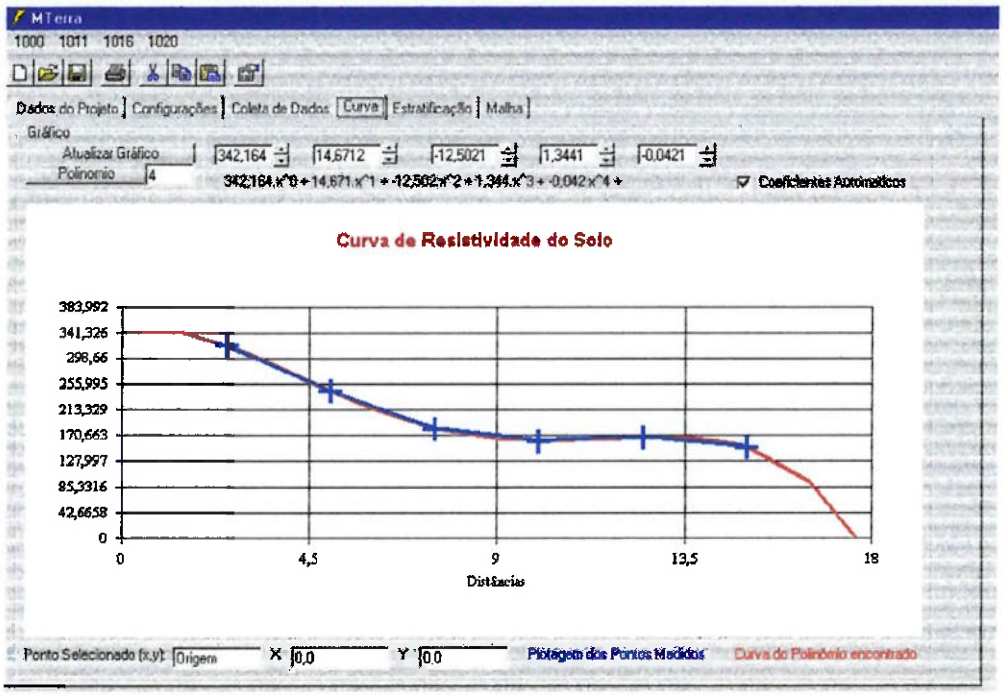

Figura 3

5.4.5 – Estratificação em Duas Camadas

De posse do polinômio previamente ajustado, no frame estratificação, figura 4, pode-se pedir para o programa calcular a estratificação em duas camadas.

Pode-se notar que há uma opção de estratificação para mais de duas camadas, tal função não esta ainda implementada, estando o programa preparado para recebe-la.

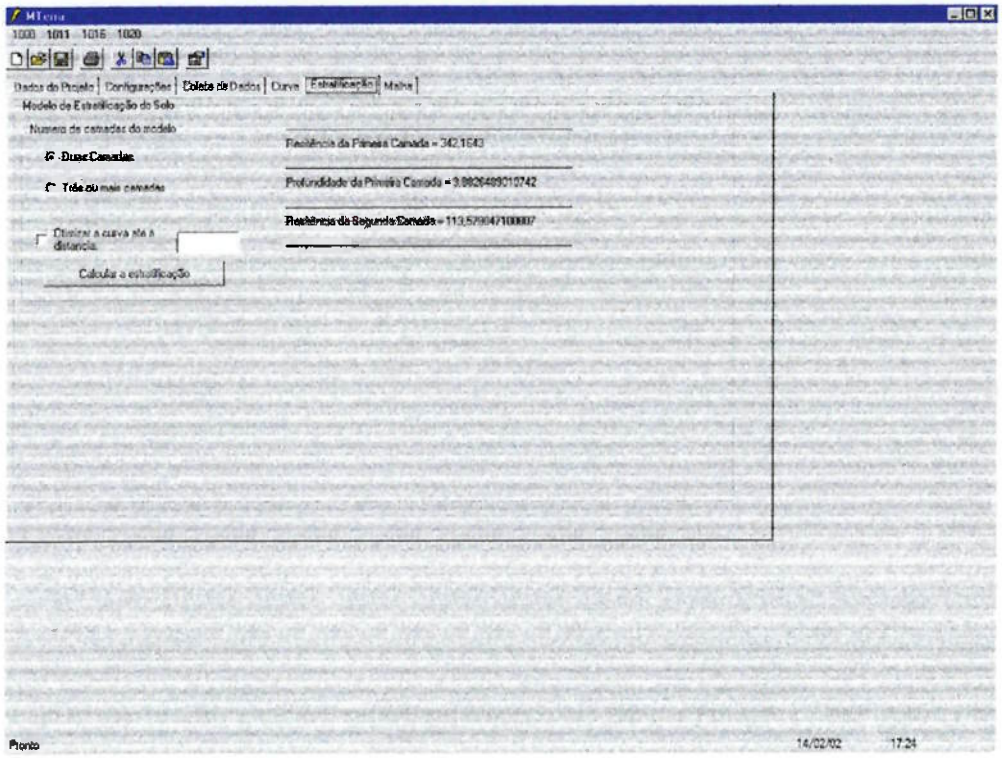

Figura 4

Para a estratificação em duas camadas foi utilizado, como descrito no item 5.3.2.5, o método de otimização.

Para otimizar a função (16), utilizou-se o Método de Newton para Seqüência de Otimização. Este método nada mais é que o Método de Newton estendido para mais de uma variável.

O Método de Newton encontra por derivações sucessivas qual valor de uma variável de uma função faz com que esta função tenha o menor valor possível.

No caso do Método estendido, o programa ajusta o valor de uma variável para posteriormente ajustar de outra e no final checa a convergência, iniciando novamente caso não haja uma boa convergência.

Para este método há necessidade de valores iniciais para as variáveis ajustadas, sendo que o programa tem inteligência para escolher o melhor valor inicial para as variáveis requisitadas.

Será otimizado somente os valores de K (constante de reflexão da primeira camada em relação a segunda) e o valor de h (profundidade da primeira camada), sendo que o valor da resistividade da primeira camada será o valor do polinômio para distância 0, isto é, prolongar a curva que rege o solo até encontrar o eixo das abcissas, como descrito no item 5.3.2.5.

Serão considerados para otimização os valores do polinômio até a maior distância que se entrou com dados de medição, sendo que o programa fornece a opção de se utilizar os valores do polinômio até um valor de distância escolhido pelo usuário.

Com os valores calculados de $\rho 1$, K e h calcula-se através da expressão (10) o valor da resistividade da segunda camada ($\rho 2$).

Os valores calculados são então apresentados no próprio frame estratificação.

5.5 – ANEXOS

5.5.1 – ANEXO 1 – DOCUMENTAÇÃO DO SOFTWARE

ANEXO 1

DOCUMENTAÇÃO DO SOFTWARE

frmMain - 1

```
Option Explicit
Private Declare Function OSWinHelp% Lib "user32" Alias "WinHelpA" (ByVal hwnd&, ByVal HelpFile$,
 ByVal wCommand%, dwData As Any)
Public Acolu As Integer 'Coluna atual
Public Alin As Integer 'Linha Atual
Public Polinomio As Integer
Public g_iGrauDoPolinomio As Integer
Private DadosDaMedicao() As Double
Public Profundidade As Double
Public DistanciaMaxima  As Double
Private coeff(8) As Double



Private Sub chkCfgConsid_Click()
If chkCfgConsid.Value = vbChecked Then
txtCfgConsid.Enabled = True
Else
txtCfgConsid.Enabled = False
End If
End Sub

Private Sub chkCfgEixosProporcionais_Click()
If chkCfgEixosProporcionais.Value = vbChecked Then _
grafCurva.Plot.UniformAxis = True Else grafCurva.Plot.UniformAxis = False
End Sub

Private Sub chkCfgEscalaEixoX_Click()
Dim index As Integer
If chkCfgEscalaEixoX.Value = vbChecked Then
    With grafCurva.Plot.Axis(VtChAxisIdX)
        .ValueScale.Auto = True
        .Intersection.Auto = True
    End With
    For index = 0 To 3
'     txtCfgEscalaX(0).Text = "Auto"
    txtCfgEscalaX(index).Enabled = False
    Next
Else
    For index = 0 To 3
    txtCfgEscalaX(index).Enabled = True
    Next
    With grafCurva.Plot.Axis(VtChAxisIdX).ValueScale
        .Auto = False
        If Not txtCfgEscalaX(0) = "Auto" Then .Minimum = txtCfgEscalaX(0)
        If Not txtCfgEscalaX(1) = "Auto" Then .Maximum = txtCfgEscalaX(1)
        If Not txtCfgEscalaX(2) = "Auto" Then .MajorDivision = txtCfgEscalaX(2)
        If Not txtCfgEscalaX(3) = "Auto" Then .MinorDivision = txtCfgEscalaX(3)
    End With

End If
End Sub

Private Sub chkCfgEscalaEixoY_Click()
Dim index As Integer
If chkCfgEscalaEixoY.Value = vbChecked Then
  grafCurva.Plot.Axis(VtChAxisIdY).ValueScale.Auto = True
For index = 0 To 3
  txtCfgEscalaY(index).Enabled = False
Next
Else
    For index = 0 To 3
    txtCfgEscalaY(index).Enabled = True
    Next
    With grafCurva.Plot.Axis(VtChAxisIdY).ValueScale
        .Auto = False
        If Not txtCfgEscalaY(0) = "Auto" Then .Minimum = txtCfgEscalaY(0)
        If Not txtCfgEscalaY(1) = "Auto" Then .Maximum = txtCfgEscalaY(1)
        If Not txtCfgEscalaY(2) = "Auto" Then .MajorDivision = txtCfgEscalaY(2)
        If Not txtCfgEscalaY(3) = "Auto" Then .MinorDivision = txtCfgEscalaY(3)
    End With
```

```
End If
End Sub

Private Sub chkTruncamento_Click()
If chkTruncamento = vbChecked Then
    txtDistTrunc.Enabled = True
Else
    txtDistTrunc.Enabled = False
    frmMain.DistanciaMaxima = gridDados.TextMatrix(gridDados.Rows - 1, 0)
    txtDistTrunc.Text = ""
    txtDistTrunc.Enabled = False
End If

End Sub

Private Sub cmdAplicDados_Click()
Dim altera As Integer
Dim i As Integer
altera = 0
gridDados.TextMatrix(0, 0) = txtCfgCabec

If 2 * txtCfgnumcol + 1 < gridDados.Cols Then
altera = MsgBox("Você inseriu um número de colunas menor que a configuração atual. Se prosseguir
 os dados serão perdidos. Deseja continuar?", vbYesNo, "MTerra - Atenção")
End If
If altera = vbYes Or altera = 0 Then
gridDados.Cols = txtCfgnumcol * 2 + 1
For i = 3 To gridDados.Cols Step 2
gridDados.ColWidth(i - 1) = 300
Next i
Else
txtCfgnumcol = (gridDados.Cols - 1) / 2
End If
altera = 0
If txtCfgNumlin + 1 < gridDados.Rows Then
altera = MsgBox("Você inseriu um número de linhas (distâncias) menor que a configuração atual. S
e prosseguir os dados serão perdidos. Deseja continuar?", vbYesNo, "MTerra - Atenção")
End If
If altera = vbYes Or altera = 0 Then
gridDados.Rows = txtCfgNumlin + 1
gridResult.Rows = txtCfgNumlin + 1
Else
txtCfgNumlin = (gridDados.Rows - 1)
End If
frmConfig.Visible = False
frameDados.Visible = True
Tabs1.Tabs(3).Selected = True


End Sub

Private Sub cmdAtualiz_Click()

'Cálculo das médias e desvios
Dim i As Integer, j As Integer
Dim soma As Double, Media As Double, dif As Double
Dim fat As Integer, desvp As Double
Dim Rest, Dist, Prof As Double
Dim flagrecalc As Boolean

On Error GoTo calcerr
With gridResult
    .TextMatrix(0, 0) = "Distâncias"
    .TextMatrix(0, 1) = "Resistência Média"
    .TextMatrix(0, 2) = "Desvio padrão"
    .TextMatrix(0, 3) = "Desvio Relativo"
    .TextMatrix(0, 4) = "Resistividade Média"
End With
ReDim DadosDaMedicao(txtCfgNumlin.Text, 0 To 1) As Double

For i = 1 To gridDados.Rows - 1
    gridResult.TextMatrix(i, 0) = gridDados.TextMatrix(i, 0)
```

```
    DadosDaMedicao(i, 0) = gridDados.TextMatrix(i, 0)
    Next i
    frmMain.DistanciaMaxima = gridDados.TextMatrix(i - 1, 0)
    txtDistTrunc.Text = ""
    txtDistTrunc.Enabled = False
    chkTruncamento.Value = vbUnchecked
For i = 1 To gridDados.Rows - 1
    soma = 0
    fat = 0
    Media = 0
    dif = 0
    desvp = 0

    'Média
Media:
    For j = 1 To gridDados.Cols - 1 Step 2
        If gridDados.TextMatrix(i, j + 1) = "C" Then
        soma = soma + gridDados.TextMatrix(i, j)
        fat = fat + 1
        End If
        Next j
        If fat > 0 Then Media = soma / fat
        gridResult.TextMatrix(i, 1) = Format(Media, "0.000000")

    'Avaliação dos valores
    flagrecalc = False
    For j = 1 To gridDados.Cols - 1 Step 2
    If chkCfgConsid.Value = vbChecked And _
        gridDados.TextMatrix(i, j + 1) = "C" And _
        Abs((gridDados.TextMatrix(i, j) - Media) / Media) > CDbl(txtCfgConsid) Then
    flagrecalc = True
    gridDados.TextMatrix(i, j + 1) = "D"
    gridDados.CellForeColor = vbRed
    gridDados.col = j + 1
    gridDados.CellForeColor = vbRed
    End If
    Next j
    If flagrecalc Then GoTo Media


    'Desvio Padrão
    soma = 0
    For j = 1 To gridDados.Cols - 1 Step 2
        If gridDados.TextMatrix(i, j + 1) = "C" Then
        dif = Abs(Media - gridDados.TextMatrix(i, j))
        soma = soma + dif
        End If
        Next j
    If fat > 0 Then desvp = soma / fat
        gridResult.TextMatrix(i, 2) = Format(desvp, "0.00000")

     'Desvio Relativo
     If desvp > 0 Then
     gridResult.TextMatrix(i, 3) = Format(desvp / Media, "0.0000%")
     Else
     gridResult.TextMatrix(i, 3) = 0
     End If

    'Resistividade do solo
     Dist = gridResult.TextMatrix(i, 0)
     Rest = gridResult.TextMatrix(i, 1)
     Prof = txtProfundidade
     gridResult.TextMatrix(i, 4) = (4 * Dist * 3.141592 * Rest) / _
        (1 + (2 * Dist / Sqr(Dist ^ 2 + (2 * Prof) ^ 2)) _
        - (2 * Dist / Sqr((2 * Dist) ^ 2 + (2 * Prof) ^ 2)))
     gridResult.TextMatrix(i, 4) = Format(gridResult.TextMatrix(i, 4), "0.00000")
    DadosDaMedicao(i, 1) = gridResult.TextMatrix(i, 4)
    Next i

txtResult.Text = ""
Exit Sub
calcerr:
```

```
    MsgBox "Erro de valor. Verifique os valores considerados e tente refazer o cálculo.", vbExcl
amation, "MTerra - Mensagem de erro no Calculo dos resultados"

End Sub

Private Sub cmdAtualizaGraf_Click()
'Atualiza o gráfico segundo as medidas de resistividade aparente do solo


'Primeiramente atualiza os dados do grid
cmdAtualiz_Click

'Tratamento de Erro
On Error GoTo errat

'Variáveis
Dim datasrc() As Double
Dim X As Integer
Dim dado As String

'redimensiona a matriz para os dados atuais
ReDim datasrc(txtCfgNumlin - 1, 2)

'Coleta os dados do grid e dos resultados calculados
For X = 0 To txtCfgNumlin - 1
    datasrc(X, 0) = gridDados.TextMatrix(X + 1, 0)
    datasrc(X, 1) = gridResult.TextMatrix(X + 1, 4)
If Polinomio = vbTrue Then
End If
Next X

'Alimenta o gráfico
grafCurva.ChartData = datasrc
grafCurva.AllowSelections = True
grafCurva.DoSetCursor = True
grafCurva.AllowDithering = True
grafCurva.Plot.UniformAxis = CBool(chkCfgEixosProporcionais.Value)

Exit Sub

errat:
MsgBox "Dados insuficientes. Descrição: " & Err.Description, vbCritical, "MTerra - Error"


End Sub

Private Sub cmdCalcEstratif_Click()
Dim i As Integer
'Leitura do numero de camadas dos modelos
Dim NumCamadas As Integer
If optNumCam2.Value = True Then
NumCamadas = 2
Else
NumCamadas = 3
MsgBox "Metodo ainda nao implementado!", vbInformation, "MTerra - Information"
Exit Sub
End If

'Leitura dos coeficientes a serem usados
For i = 0 To GrauPoli
CoefPol(i) = CDbl(Trim(txtCoefPoli(i)))
Next i

'Inicia o calculo do Método de Pirson
MetPirson

lblCmd1.Caption = "Resitência da Primeira Camada = " & resistcamada(0)

lblCmd2.Caption = "Profundidade da Primeira Camada = " & valKh(1)

lblCmd3.Caption = "Resitência da Segunda Camada = " & resistcamada(1)
```

```
frmMain - 5

End Sub

Private Sub cmdCfgEscalaAplic_Click()
cmdPoli_Click
End Sub

Private Sub cmdCfgEscalaDef_Click()
Dim check As Integer
Dim autoescala As Boolean

check = MsgBox("Deseja restaurar os valores padrão de escala? Você perderá os valor estabelecido
s até o momento." _
            , vbDefaultButton2, "MTerra - Confirmação")

If check = vbOK Then
chkCfgEixosProporcionais.Value = 0
autoescala = False
cmdPoli_Click
End If

End Sub

Private Sub cmdConfig_Click()
frameDados.Visible = False
frmConfig.Visible = True
End Sub

Private Sub cmdEditCanc_Click()
Voltadados
End Sub

Private Sub cmdEditOk_Click()
Dim res As Integer
If txtDados.Text = "" Then
res = MsgBox("Tem certeza que deseja apagar este valor?", vbYesNoCancel, "MTerra")
If res = vbYes Then
Trocadados (txtDados.Text)
ElseIf res = vbNo Then
Voltadados
Else
txtDados.Enabled = False
cmdEditOk.Enabled = False
cmdEditCanc.Enabled = False
Exit Sub
End If
Else
Trocadados (txtDados.Text)
End If
End Sub
Private Sub cmdPoli_Click()
'Gera o polinomio a partir dos dados inseridos no frmMain e armasenados na Matriz DadosUser()
Dim DadosUser() As Double 'Dados inseridos pelo usuário
Dim i As Integer
'ReDim coeff(txtGrauPoly1 + 1)

'Análise de erro - retirar o comentário
On Error GoTo erroPoly

If txtCfgNumlin <= 0 Or txtCfgnumcol <= 0 Then
MsgBox "Não há dados suficientes. Por favor insira os dados primeiro.", vbCritical, "MTerra"
Exit Sub
End If

cmdAtualizaGraf_Click
'ReDim DadosUser(txtGrauPoly1 + 2, 3)
'For i = 0 To txtCfgNumlin - 1
'    DadosUser(i, 0) = gridResult.TextMatrix(i + 1, 0)
'    DadosUser(i, 1) = gridResult.TextMatrix(i + 1, 4)
'Next i

Dim distancias() As Double
Dim resistividades() As Double
```

```
Dim graupoly As Integer
graupoly = txtGrauPoly1

ReDim distancias(txtCfgNumlin + 1)
ReDim resistividades(txtCfgNumlin + 1)

For i = 0 To txtCfgNumlin - 1
distancias(i) = gridResult.TextMatrix(i + 1, 0)
Next i
For i = 0 To txtCfgNumlin - 1
resistividades(i) = gridResult.TextMatrix(i + 1, 4)
Next i

'Chama a função que determina o polinomio
If chkInterpolacao = vbChecked Then
DeterPoly distancias, resistividades, txtCfgNumlin, graupoly, coeff
For i = 0 To 6
txtCoefPoli(i).Visible = False
updCoefPoli(i).Visible = False
Next i
End If

'Escreve o polinomio obtido
lblPoly.Caption = ""
For i = 0 To txtGrauPoly1
lblPoly.Caption = lblPoly.Caption & Format(coeff(i), "#0.000") & ".x^" & i & " + "
txtCoefPoli(i).Visible = True
updCoefPoli(i).Visible = True
txtCoefPoli(i).Text = Format(coeff(i), "#0.0000")
Next i
Polinomio = vbTrue

'Traça a curva do polinomio interpolado
Dim ncurva() As Double
ReDim ncurva(txtNumPontos - 1, 3)
'Alimenta os pontos da medição
For i = 0 To txtCfgNumlin - 1
ncurva(i, 0) = distancias(i)
ncurva(i, 1) = resistividades(i)
Next i
For i = txtCfgNumlin To txtNumPontos - 1
ncurva(i, 0) = distancias(txtCfgNumlin - 1)
ncurva(i, 1) = resistividades(txtCfgNumlin - 1)
Next i
'Plota a curva do Polinômio
For i = 0 To txtNumPontos - 1
ncurva(i, 2) = txtXPolyMin + i * ((txtXPolyMax - txtXPolyMin) / txtNumPontos)
If chkInterpolacao = vbChecked Then
ncurva(i, 3) = Polynomio(coeff, graupoly, ncurva(i, 2))
Else
ncurva(i, 3) = Polynomio(coeff, graupoly, ncurva(i, 2))
End If
Next i


'Alimenta o gráfico
grafCurva.ChartData = ncurva
grafCurva.Plot.UniformAxis = False
If Not chkCfgEscalaEixoX.Value = vbChecked Then

With grafCurva.Plot.Axis(VtChAxisIdX)
    .ValueScale.Auto = False
    .ValueScale.Maximum = 1.2 * MaiorValor(distancias, False)
    .ValueScale.Minimum = 0
    .ValueScale.MajorDivision = .ValueScale.Maximum * 0.2
    .AxisScale.Type = VtChScaleTypeLinear
    .AxisTitle.Text = txtCfgCabec.Text
    .AxisGrid.MajorPen.Join = VtPenJoinBevel
    txtCfgEscalaX(0) = .ValueScale.Minimum
    txtCfgEscalaX(1) = .ValueScale.Maximum
    txtCfgEscalaX(2) = .ValueScale.MajorDivision
    txtCfgEscalaX(3) = .ValueScale.MinorDivision
```

```
End With
End If
If Not chkCfgEscalaEixoY = vbChecked Then
With grafCurva.Plot.Axis(VtChAxisIdY).ValueScale
    .Auto = False
    .Minimum = 0
    .Maximum = 1.2 * MaiorValor(resistividades, False)
    txtCfgEscalaY(0) = .Minimum
    txtCfgEscalaY(1) = .Maximum
    txtCfgEscalaY(2) = .MajorDivision
    txtCfgEscalaY(3) = .MinorDivision
End With
grafCurva.Plot.SeriesCollection(3).DataPoints(-1).Brush.FillColor.Set 255, 0, 0
grafCurva.Plot.SeriesCollection(3).SeriesMarker.Show = False
grafCurva.Refresh
End If

Exit Sub
erroPoly:
    MsgBox "Ocorreu um erro durante o processamento. Verifique os dados." & Err.Description, vbC
ritical, "MTerra"

End Sub

Private Sub Form_Load()
    Dim i As Integer

    LoadResStrings Me
    Me.Left = GetSetting(App.Title, "Settings", "MainLeft", 1000)
    Me.Top = GetSetting(App.Title, "Settings", "MainTop", 1000)
    Me.Width = GetSetting(App.Title, "Settings", "MainWidth", 6500)
    Me.Height = GetSetting(App.Title, "Settings", "MainHeight", 6500)
    gridResult.ColWidth(0) = 1500
    gridResult.ColWidth(1) = 1500
    gridResult.ColWidth(2) = 1500
    gridResult.ColWidth(3) = 1500
    gridResult.ColWidth(4) = 2000
    gridDados.ColWidth(0) = 950
    i = 2
    While (i <= gridDados.Cols)
    gridDados.TextMatrix(0, i - 1) = "Resistência"
    i = i + 1
    gridDados.ColWidth(i - 1) = 300
    i = i + 1
    Wend
    txtCfgnumcol.Text = (gridDados.Cols - 1) / 2
    txtCfgNumlin.Text = gridDados.Rows - 1
    frameProj.Visible = True

End Sub


Private Sub Form_Unload(Cancel As Integer)
    Dim i As Integer


    'close all sub forms
    For i = Forms.count - 1 To 1 Step -1
        Unload Forms(i)
    Next
    If Me.WindowState <> vbMinimized Then
        SaveSetting App.Title, "Settings", "MainLeft", Me.Left
        SaveSetting App.Title, "Settings", "MainTop", Me.Top
        SaveSetting App.Title, "Settings", "MainWidth", Me.Width
        SaveSetting App.Title, "Settings", "MainHeight", Me.Height
    End If
End Sub

Private Sub grafCurva_PointSelected(Series As Integer, DataPoint As Integer, MouseFlags As Integ
er, Cancel As Integer)
Select Case Series
Case 1
```

```
txtDataPoint(2).Text = "Medição"
Case 3
txtDataPoint(2).Text = "Polinômio"
End Select
grafCurva.Column = Series
grafCurva.row = DataPoint
txtDataPoint(0) = grafCurva.Data
grafCurva.Column = Series + 1
grafCurva.row = DataPoint
txtDataPoint(1) = grafCurva.Data

End Sub

Private Sub grafCurva_SeriesActivated(Series As Integer, MouseFlags As Integer, Cancel As Intege
r)
'Altera as cores das curvas plotadas
If MouseFlags = 1 Then
grafCurva.Plot.SeriesCollection(Series).DataPoints(-1).Brush.FillColor.Set 255, 0, 0
ElseIf MouseFlags = 9 Then
grafCurva.Plot.SeriesCollection(Series).DataPoints(-1).Brush.FillColor.Set 0, 255, 0
Else
grafCurva.Plot.SeriesCollection(Series).DataPoints(-1).Brush.FillColor.Set 0, 0, 255
End If

End Sub
Sub GravarArquivo(Caminho As String, Nome As String, Extensao As String)

frmMain.Refresh
Open Caminho & Nome & "." & Extensao For Output As #1

Write #1, "Projeto de malha de terra do programa MTerra"
Write #1, "copyright©2002"
Write #1, "**************************************************"
Write #1, "*@*"
Write #1, txtProj
Write #1, txtData.Text
Write #1, txtHora.Text
Write #1, txtLocal.Text
Write #1, txtRegiao.Text
Write #1, txtCliente.Text
Write #1, txtContato.Text
Write #1, txtEnd.Text
Write #1, txtTel1.Text
Write #1, txtTel2.Text
Write #1, txtFax.Text
Write #1, txtEmail.Text
Write #1, txtTec.Text
Write #1, txtTecTel.Text
Write #1, txtObs.Text
Write #1, "End OK"
Close #1


End Sub
Private Sub gridDados_DblClick()
Dim i As Integer

'Marca a posição Atual
On Error GoTo mm
Acolu = gridDados.col
Alin = gridDados.row

    'Primeira coluna (nomes)
'    If Acolu = 0 Then
'        gridDados.TextMatrix(Alin, Acolu) = InputBox("Insira o nome desta linha.", "MTerra - En
trada de dados", _
'        gridDados.TextMatrix(Alin, Acolu))
'    End If

    'colunas de cosideração
    If Acolu > 0 And Acolu Mod 2 = 0 Then
        If gridDados.TextMatrix(Alin, Acolu) = "C" Then
```

```
        gridDados.TextMatrix(Alin, Acolu) = "D"
        gridDados.CellForeColor = vbRed
        gridDados.col = Acolu - 1
        gridDados.CellForeColor = vbRed
        ElseIf gridDados.TextMatrix(Alin, Acolu) = "D" Then
        gridDados.TextMatrix(Alin, Acolu) = "C"
        gridDados.CellForeColor = vbBlue
        gridDados.col = Acolu - 1
        gridDados.CellForeColor = vbBlue

        End If
    End If
    'colunas de dados
    If (Acolu Mod 2) <> 0 Or Acolu = 0 Then
        txtDados.Enabled = True
        txtDados.Text = gridDados.TextMatrix(Alin, Acolu)
        txtDados.SetFocus
        cmdEditOk.Enabled = True
        cmdEditCanc.Enabled = True
    End If
Exit Sub
mm: MsgBox "Não foi possível atualizar esta célula.", vbApplicationModal
End Sub

Private Sub gridResult_Click()
Dim Rcolu As Integer, Rlin As Integer

Rcolu = gridResult.col
Rlin = gridResult.row

txtResult = gridResult.TextMatrix(Rlin, Rcolu)

End Sub
Sub LeArquivo()

Dim str As String
Open "c:\Testfile.mtr" For Input As #1
Do While Not EOF(1)
    Input #1, str
    If str = "*@*" Then
    Input #1, str
    txtProj.Text = str
    Input #1, str
    txtData.Text = str
    Input #1, str
    txtHora.Text = str
    Input #1, str
    txtLocal.Text = str
    Input #1, str
    txtRegiao.Text = str
    Input #1, str
    txtCliente.Text = str
    End If
Loop
Close #1
End Sub

Private Sub Tabs1_Click()
frameProj.Visible = False
frameDados.Visible = False
frameGraf.Visible = False
frmConfig.Visible = False
frmEstratif.Visible = False
'esconde todos os frames
Select Case Tabs1.SelectedItem.index
Case 1
frameProj.Visible = True
Case 2
frmConfig.Visible = True
Case 3
frameDados.Visible = True
Case 4
```

```
frameGraf.Visible = True
Case 5
frmEstratif.Visible = True
Case 6
'exibe o frame 6
End Select

End Sub

Private Sub tbToolBar_ButtonClick(ByVal Button As MSComCtlLib.Button)
    On Error Resume Next
    Select Case Button.Key
        Case "New"
            'ToDo: Add 'New' button code.
            MsgBox "Add 'New' button code."
        Case "Open"
           LeArquivo
        Case "Save"
            GravarArquivo "C:\", "Testfile", "mtr"
        Case "Print"
            mnuFilePrint_Click
        Case "Cut"
            mnuEditCut_Click
        Case "Copy"
            mnuEditCopy_Click
        Case "Paste"
            mnuEditPaste_Click
        Case "Properties"
            mnuFileProperties_Click
    End Select
End Sub

Private Sub mnuHelpAbout_Click()
    frmAbout.Show vbModal, Me
End Sub

Private Sub mnuHelpContents_Click()
    Dim nRet As Integer


    'if there is no helpfile for this project display a message to the user
    'you can set the HelpFile for your application in the
    'Project Properties dialog
    If Len(App.HelpFile) = 0 Then
        MsgBox "Unable to display Help Contents. There is no Help associated with this project."
, vbInformation, Me.Caption
    Else
        On Error Resume Next
        nRet = OSWinHelp(Me.hwnd, App.HelpFile, 3, 0)
        If Err Then
            MsgBox Err.Description
        End If
    End If

End Sub


Private Sub mnuViewOptions_Click()
    frmOptions.Show vbModal, Me
End Sub

Private Sub mnuViewRefresh_Click()
    'ToDo: Add 'mnuViewRefresh_Click' code.
    MsgBox "Add 'mnuViewRefresh_Click' code."
End Sub

Private Sub mnuViewToolbar_Click()
    mnuViewToolbar.Checked = Not mnuViewToolbar.Checked
    tbToolBar.Visible = mnuViewToolbar.Checked
End Sub

Private Sub mnuEditPaste_Click()
```

```
    'ToDo: Add 'mnuEditPaste_Click' code.
    MsgBox "Add 'mnuEditPaste_Click' code."
End Sub

Private Sub mnuEditCopy_Click()
    'ToDo: Add 'mnuEditCopy_Click' code.
    MsgBox "Add 'mnuEditCopy_Click' code."
End Sub

Private Sub mnuEditCut_Click()
    'ToDo: Add 'mnuEditCut_Click' code.
    MsgBox "Add 'mnuEditCut_Click' code."
End Sub

Private Sub mnuEditUndo_Click()
    'ToDo: Add 'mnuEditUndo_Click' code.
    MsgBox "Add 'mnuEditUndo_Click' code."
End Sub

Private Sub mnuFileExit_Click()
    'unload the form
    Unload Me

End Sub

Private Sub mnuFilePrint_Click()
    'ToDo: Add 'mnuFilePrint_Click' code.
    MsgBox "Add 'mnuFilePrint_Click' code."
End Sub

Private Sub mnuFilePrintPreview_Click()
    'ToDo: Add 'mnuFilePrintPreview_Click' code.
    MsgBox "Add 'mnuFilePrintPreview_Click' code."
End Sub

Private Sub mnuFilePageSetup_Click()
    On Error Resume Next
'    With dlgCommonDialog
'        .DialogTitle = "Page Setup"
'        .CancelError = True
'        .ShowPrinter
'    End With

End Sub

Private Sub mnuFileProperties_Click()
    'ToDo: Add 'mnuFileProperties_Click' code.
    MsgBox "Add 'mnuFileProperties_Click' code."
End Sub

Private Sub mnuFileSaveAs_Click()
    'ToDo: Add 'mnuFileSaveAs_Click' code.
    MsgBox "Add 'mnuFileSaveAs_Click' code."
End Sub

Private Sub mnuFileSave_Click()
    'ToDo: Add 'mnuFileSave_Click' code.
    MsgBox "Add 'mnuFileSave_Click' code."
End Sub

Private Sub mnuFileClose_Click()
    'ToDo: Add 'mnuFileClose_Click' code.
    MsgBox "Add 'mnuFileClose_Click' code."
End Sub

Private Sub mnuFileOpen_Click()
    Dim sFile As String


'    With dlgCommonDialog
'        .DialogTitle = "Open"
'        .CancelError = False
```

```
'        'ToDo: set the flags and attributes of the common dialog control
'        .Filter = "All Files (*.*)|*.*"
'        .ShowOpen
'        If Len(.FileName) = 0 Then
'            Exit Sub
'        End If
'        sFile = .FileName
'    End With
    'ToDo: add code to process the opened file

End Sub


Private Sub mnuFileNew_Click()
    'ToDo: Add 'mnuFileNew_Click' code.
    MsgBox "Add 'mnuFileNew_Click' code."
End Sub

Sub Trocadados(ByVal novoValor)
gridDados.TextMatrix(Alin, Acolu) = novoValor
txtDados.Enabled = False
cmdEditOk.Enabled = False
cmdEditCanc.Enabled = False
If novoValor = 0 Or novoValor = "" Then
gridDados.TextMatrix(Alin, Acolu + 1) = ""
gridDados.CellForeColor = vbBlue
ElseIf Acolu > 0 Then
gridDados.TextMatrix(Alin, Acolu + 1) = "C"
gridDados.CellForeColor = vbBlue
gridDados.col = Acolu + 1
gridDados.CellForeColor = vbBlue
End If
End Sub
Sub Voltadados()
txtDados.Text = gridDados.TextMatrix(Alin, Acolu)
txtDados.Enabled = False
cmdEditOk.Enabled = False
cmdEditCanc.Enabled = False
End Sub

Private Sub txtCoefPoli_KeyPress(index As Integer, KeyAscii As Integer)
If KeyAscii = vbKeyReturn Then
coeff(index) = CDbl(Trim(txtCoefPoli(index).Text))
chkInterpolacao.Value = vbUnchecked
Call cmdPoli_Click
End If
End Sub


Private Sub txtDados_KeyDown(KeyCode As Integer, Shift As Integer)

If KeyCode = vbKeyReturn Then cmdEditOk_Click

End Sub

Private Sub txtDistTrunc_LostFocus()
frmMain.DistanciaMaxima = CDbl(Trim(txtDistTrunc.Text))
End Sub

'Private Sub txtGrauPoly1_KeyDown(KeyCode As Integer, Shift As Integer)
'If KeyCode = vbKeyReturn Then
'frmMain.g_iGrauDoPolinomio = CInt(Trim(txtGrauPoly1.Text))
'End If
'
'End Sub
Private Sub txtGrauPoly1_LostFocus()
If txtGrauPoly1 < 2 Then
MsgBox "Não é possível interpolar os pontos para um grau menor que 2.", vbInformation, "MTerra -
 ErroGrauPoli"
txtGrauPoly1.Text = 2
ElseIf txtGrauPoly1 >= 8 Then
MsgBox "O grau deve ser menor ou igual a 7. O programa nao aproxima para este ordem de polinomio
```

```
."
, vbInformation, "MTerra - Error"
txtGrauPoly1.Text = 7
End If
frmMain.g_iGrauDoPolinomio = CInt(Trim(txtGrauPoly1.Text))
End Sub

Private Sub txtProfundidade_LostFocus()
frmMain.Profundidade = CDbl(Trim(txtProfundidade))
End Sub

Private Sub updCoefPoli_MouseDown(index As Integer, Button As Integer, Shift As Integer, X As Si
ngle, Y As Single)
coeff(index) = CDbl(Trim(txtCoefPoli(index).Text))
chkInterpolacao.Value = vbUnchecked
Call cmdPoli_Click

End Sub
```

Pirson - 1

```
Option Explicit
Global resistcamada() As Double
Global GrauPoli As Integer
Global valKh(3) As Double


Function MetPirson()

 Dim numCamada As Integer
 Dim Toler As Double
 Dim ordemPolyDer As Integer
 Dim numInter As Integer
 Dim NumVars As Integer
 Dim roots(5) As polyRoot
 Dim NumCamadas As Integer
 Dim matder1() As Double
 Dim matder2() As Double
 Dim convergencia As Integer
 Dim k, p, i As Integer


'Dim rootsDer() As PolyRoots

' ValorPoly MatrizCoefPol, numRows, 0
' ResistividadeCamada(0) = ValordoPoly
' Escolher se havera mais de uma camada na estratificao
' Se for escolhido mais de uma camada numCamada = 1, caso contrario numCamada = 0
 Toler = 0.00000001
 numInter = 100
 GrauPoli = frmMain.g_iGrauDoPolinomio 'frmMain.txtGrauPoly1
 If frmMain.optNumCam2.Value = True Then NumCamadas = 2
' Else: numcamadas = 5
' End If

 ReDim matder1(GrauPoli + 1) As Double
 ReDim matder2(GrauPoli + 2) As Double
 ReDim resistcamada(NumCamadas) As Double

 resistcamada(0) = Polynomio(CoefPol(), GrauPoli, 0)

 DeriPoly CoefPol, matder1, GrauPoli + 1

 If matder1(GrauPoli - 1) >= 0 Then
  k = 1
 Else
  k = -1
 End If


 DeriPoly matder1, matder2, GrauPoli

 Select Case GrauPoli
 Case 2
 roots(0).real = Abs(matder2(0))
 Case 3
 roots(0).real = Abs(matder2(0) * -1 / matder2(1))
 Case Is > 3
 lbpolyroots matder2, roots, GrauPoli - 2, Toler
 End Select

 If frmMain.optNumCam2.Value = True Then
 NumVars = 2
  valKh(0) = 0



  valKh(1) = roots(0).real
  'funcao para pegar a menor raiz
  For i = 0 To count
   If valKh(1) > roots(i + 1).real And roots(i + 1).real <> 0 Then
    valKh(1) = roots(i + 1).real
   End If
  Next i
```

Pirson - 2

```
  convergencia = vbFalse
  p = 1

  While Not convergencia And Abs(valKh(0)) <= 1
   convergencia = NewtonMultiMin(valKh, NumVars, Toler, numInter)
   If k = 1 And (Not convergencia Or Abs(valKh(0)) > 1) Then
    valKh(0) = 0 + p * 0.1
    valKh(1) = roots(1).real
    convergencia = vbFalse
   ElseIf (Not convergencia Or Abs(valKh(0)) > 1) Then
    valKh(0) = 0 - p * 0.1
    valKh(1) = roots(1).real
    convergencia = vbFalse
   End If
    p = p + 1
    Wend
 If convergencia Then
  MsgBox "Convergiu! K=" & valKh(0) & " e h=" & valKh(1)
 Else
  MsgBox "Fodeu! K=" & valKh(0) & " e h=" & valKh(1)
 End If

 resistcamada(1) = resistcamada(0) * ((valKh(0) + 1) / (1 - valKh(0)))

 frmMain.Refresh


 End If

End Function
Function SolutionExp1(valKh() As Double, ValEsp As Integer, Optional ValProf As Integer) As D
ouble

Dim i As Integer

SolutionExp1 = 0
For i = 1 To 10
 SolutionExp1 = SolutionExp1 + ((valKh(0) ^ i) / Sqr(1 + (2 * i * valKh(1) / ValEsp) ^ 2)) -
((valKh(0) ^ i) / Sqr(4 + (2 * i * valKh(1) / ValEsp) ^ 2))
Next i

End Function

Function Exp2(valKh() As Double) As Double

Dim i As Integer

Exp2 = 0
For i = 1 To frmMain.DistanciaMaxima
Exp2 = Exp2 + (Polynomio(CoefPol(), GrauPoli, CDbl(i)) - (resistcamada(0) * (1 + (4 * Solutio
nExp1(valKh(), i, frmMain.Profundidade))))) ^ 2
Next i

End Function
```

```
Matrizes - 1

Option Explicit

Global Const OPTIM_BAD_RESULT# = -1E+31
Global Const MATVEC_EPSILON# = 0.000000000000001
Global Const MATVEC_BAD_RESULT# = -1E+30
Global Const matErr_Nome% = 0
Global Const matErr_Size% = 1
Global Const matErr_Singular% = 2
Global Const matErr_IllConditioned% = 3
Global Const matErr_IterLimit% = 4
Type polyRoot
 real As Double
 imag As Double
 isComplex As Integer
End Type
Function checkRowCol(Mat() As Double, row As Integer, col As Integer) As Integer

 If (row >= 0) And (col >= 0) And (row < UBound(Mat, 1)) And (col < UBound(Mat, 2)) Then
  checkRowCol = True
 Else
  checkRowCol = False
 End If

End Function

Function CopyMat(MatB() As Double, MatA() As Double, numRows As Integer, numCols As Integer)
As Integer

 'Copia a matriz A (matA) na matriz B (matB)

 Dim row As Integer, col As Integer
 If Not (checkRowCol(MatA(), numRows, numCols) And checkRowCol(MatB(), numRows, numCols)) The
n
  CopyMat = matErr_Size
  Exit Function
 End If
 For row = 0 To numRows - 1
  For col = 0 To numCols
   MatB(row, col) = MatA(row, col)
  Next col
 Next row
 CopyMat = matErr_Nome

End Function
Function GaussJordan(a() As Double, b() As Double, numRows As Integer, numCols As Integer) As
 Integer

 Dim rowIndex() As Integer
 Dim colIndex() As Integer
 Dim pivotIndex() As Integer
 Dim i As Integer, j As Integer, k As Integer
 Dim n As Integer, m As Integer
 Dim row As Integer, col As Integer
 Dim large As Double, z As Double, oneOverPiv As Double

 If Not checkRowCol(a(), numRows, numCols) Then
  GaussJordan = matErr_Size
  Exit Function
 End If

 'reside local dynamic arrays
 ReDim rowIndex(numRows)
 ReDim colIndex(numRows)
 ReDim pivotIndex(numRows)

 'initialize the row and column indices
 For i = 0 To numRows - 1
  rowIndex(i) = i
  colIndex(i) = i
 Next i

 'initialize the pivot index array
 For i = 0 To numRows - 1
  pivotIndex(i) = -1
```

```
    Next i

    For i = 0 To numRows - 1
     large = 0
     For j = 0 To numRows - 1
      If pivotIndex(j) <> 0 Then
       For k = 0 To numRows - 1
        If pivotIndex(k) = -1 Then
         If Abs(a(j, k)) >= large Then
          large = Abs(a(j, k))
          row = j
          col = k
         End If
        ElseIf pivotIndex(k) > 0 Then
         GaussJordan = matErr_Singular
         Exit Function
        End If
       Next k
      End If
     Next j
     pivotIndex(col) = pivotIndex(col) + 1
     If row <> col Then
      For n = 0 To numRows - 1
       swapDouble a(row, n), a(col, n)
      Next n
      For n = 0 To numCols - 1
       swapDouble b(row, n), b(col, n)
      Next n
     End If
     rowIndex(i) = row
     colIndex(i) = col
     If Abs(a(col, col)) < 0.0000000001 Then
      GaussJordan = matErr_Singular
      Exit Function
     End If
     oneOverPiv = 1 / a(col, col)
     a(col, col) = 1
     For n = 0 To numRows - 1
      a(col, n) = a(col, n) * oneOverPiv
     Next n
     For n = 0 To numCols - 1
      b(col, n) = b(col, n) * oneOverPiv
     Next n
     For m = 0 To numRows - 1
      If m <> col Then
       z = a(m, col)
       a(m, col) = 1
       For n = 0 To numRows - 1
        a(m, n) = a(m, n) - a(col, n) * z
       Next n
       For n = 0 To numCols - 1
        b(m, n) = b(m, n) - b(col, n) * z
       Next n
      End If
     Next m
    Next i
    For n = numRows - 1 To 0 Step -1
     If rowIndex(n) <> colIndex(n) Then
      For k = 0 To numRows - 1
       swapDouble a(k, rowIndex(n)), a(k, colIndex(n))
      Next k
     End If
    Next n
    GaussJordan = matErr_Nome

End Function



Function NewtonMultiMin(x() As Double, NumVars As Integer, tolerance As Double, maxIter As In
teger) As Integer

 Dim i As Integer, j As Integer
 Dim k As Integer
```

```
Matrizes - 3

 Dim diff As Double
 Dim ok As Integer

 Do
  ok = False
  For i = 0 To NumVars - 1
   If ExNewtonMin(x(), tolerance, diff, maxIter, i) Then
    If diff > tolerance Then
     ok = True
    End If
   Else
    NewtonMultiMin = False
    Exit Function
   End If
  Next i
 Loop While ok
 NewtonMultiMin = True

End Function
Function ExNewtonMin(x() As Double, tolerance As Double, diff As Double, maxIter As Integer,
index As Integer) As Integer

 Dim h As Double
 Dim Fm As Double, F0 As Double
 Dim Fp As Double
 Dim firstDeriv As Double, secondDeriv As Double
 Dim iter As Integer
 Dim xOld As Double
 Dim xx As Double

 iter = 0
 xOld = x(index)
 xx = xOld
 Do
  'calculate increment
  If Abs(xx) > 1 Then h = 0.01 * xx Else h = 0.01
  'calculate function values at X-h, X and X+h
  x(index) = xx - h
  Fm = Exp2(x())
  x(index) = xx + h
  Fp = Exp2(x())
  x(index) = xx
  F0 = Exp2(x())
  'calculate the first derivative
  firstDeriv = (Fp - Fm) / 2 / h
  'calculate the second derivative
  secondDeriv = (Fp - 2 * F0 + Fm) / h ^ 2
  'calculate the guess refinement
  diff = firstDeriv / secondDeriv
  'refine the guess
  xx = xx - diff
  x(index) = xx
  iter = iter + 1 'the increment iteration counter
 Loop While Abs(diff) > tolerance And iter < maxIter

 If iter <= maxIter Then
  ExNewtonMin = True
 Else
  x(index) = xOld
  ExNewtonMin = False
 End If

End Function
Function swapDouble(d1 As Double, d2 As Double)

Dim t As Double

t = d1
d1 = d2
d2 = t

End Function
```

Polinomios - 1

```
Option Explicit
Global CoefPol() As Double
Global count As Integer

Function DeriPoly(CoefPoly() As Double, PolyDer() As Double, numCoeff As Integer) As Integer

'Esta função deriva um polinômio de grau numCoeff apresentado no vetor de coeficientes MatDer
()


 Dim i As Integer
 For i = 0 To numCoeff - 1
  PolyDer(i) = CoefPoly(i) * i
 Next i
 For i = 0 To numCoeff - 1
  PolyDer(i) = PolyDer(i + 1)
 Next i

End Function


Function DeflatePolyRoots(coeff() As Double, initGuess As Double, roots() As Double, numRoots
 As Integer, polyOrder As Integer, maxIter As Integer, tolerance As Double) As Integer

 Dim a() As Double
 Dim b() As Double
 Dim c() As Double
 Dim diff As Double
 Dim z As Double, X As Double
 Dim i As Integer, iter As Integer, n As Integer

 iter = 1
 n = polyOrder + 1
 X = initGuess
 numRoots = 0

 'allocate dynamic coefficients
 ReDim a(n)
 ReDim b(n)
 ReDim c(n)

 For i = 0 To n - 1
  a(i) = coeff(i)
 Next i

 Do While (iter <= maxIter) And (n > 1)
  iter = iter + 1
  z = X
  b(n - 1) = a(n - 1)
  c(n - 1) = a(n - 1)
  For i = n - 2 To 1 Step -1
   b(i) = a(i) + z * b(i + 1)
   c(i) = b(i) + z * c(i + 1)
  Next i
  b(0) = a(0) + z * b(1)
  diff = b(0) / c(1)
  X = X - diff
  If Abs(diff) <= tolerance Then
   iter = 1 'reset iteration counter
   n = n - 1
   roots(numRoots) = X
   numRoots = numRoots + 1
   'update deflated roots
   For i = 0 To n - 1
    a(i) = b(i + 1)
    'get the last root
    If n = 2 Then
     n = n - 1
     roots(numRoots) = -a(0)
    End If
   Next i
  End If
Loop
DeflatePolyRoots = True
```

```
End Function

Function lbpolyroots(coeff() As Double, roots() As polyRoot, polyOrder As Integer, tolerance
As Double) As Integer

 'solves for the roots of the following polynomial
  'y = coeff(0) + coeff(1) * x + ... + coeff(n) * x^n
 'Parameters:
  'coeff must be an array wiyh a least polyOrder + 1 elements
  'roots output array of roots
  'polyOrder order of polynomial
  'tolerance tolerance of solutions

 Const SMALL# = 0.00000001
 Dim a() As Double
 Dim b() As Double
 Dim c() As Double
 Dim d() As Double
 Dim alfa1 As Double, alfa2 As Double
 Dim beta1 As Double, beta2 As Double
 Dim delta1 As Double, delta2 As Double, delta3 As Double
 Dim i As Integer, j As Integer, k As Integer
 Dim n As Integer
 Dim n1 As Integer
 Dim n2 As Integer

 n = polyOrder
 n1 = n + 1
 n2 = n + 2
 'is the coefficient of the highest term zero?
 If Abs(coeff(0)) < SMALL Then
  lbpolyroots = False
  Exit Function
 End If

 'allocate dynamic coefficients
 ReDim a(n1)
 ReDim b(n1)
 ReDim c(n1)
 ReDim d(n1)

 For i = 0 To n
  a(n1 - i) = coeff(i)
 Next i

 'is highest coeff no close to 1?
 If Abs(a(1) - 1) > SMALL Then
  'adjust coefficient because a(1) !=1
  For i = 2 To n1
   a(i) = a(i) / a(1)
  Next i
  a(1) = 1#
 End If

 'initialize roots counter
 count = 0
 Do

  'start the main Lin-Bairstow iteration loop
  'initialise the counter and the guesses for the coefficients of quadratic factor
  'p(x) = x^2 + alfa1 * x + beta1

  alfa1 = 1
  beta1 = 1

  Do
   b(0) = 0
   d(0) = 0
   b(1) = 1
   d(1) = 1
   j = 1
   k = 0
   For i = 2 To n1
```

```
    b(i) = a(i) - alfa1 * b(j) - beta1 * b(k)
    d(i) = b(i) - alfa1 * d(j) - beta1 * d(k)
    j = j + 1
    k = k + 1
   Next i
   j = n - 1
   k = n - 2
   delta1 = d(j) ^ 2 - (d(n) - b(n)) * d(k)
   alfa2 = (b(n) * d(j) - b(n1) * d(k)) / delta1
   beta2 = (b(n1) * d(j) - (d(n) - b(n)) * b(n)) / delta1
   alfa1 = alfa1 + alfa2
   beta1 = beta1 + beta2
  Loop While (Abs(alfa2) > tolerance) Or (Abs(beta2) > tolerance)

  delta1 = alfa1 ^ 2 - 4 * beta1

  If delta1 < 0 Then
   'imaginary roots
   delta2 = Sqr(Abs(delta1)) / 2
   delta3 = -alfa1 / 2
   For i = 0 To 1
    roots(count + i).isComplex = True
    roots(count + i).real = delta3
    roots(count + i).imag = Sign(i) * delta2
   Next i
  Else
   delta2 = Sqr(delta1)
   'roots are real
   For i = 0 To 1
    roots(count + i).isComplex = False
    roots(count + i).imag = 0
   Next i
   roots(count).real = (delta2 - alfa1) / 2
   roots(count + 1).real = (delta2 + alfa1) / (-2)
  End If
  'update root counter
  count = count + 2

  'reduce polynomial order
  n = n - 2
  n1 = n1 - 2
  n2 = n2 - 2

  'for n>=2 calculate coefficients of the new polynomial
  If n >= 2 Then
   For i = 1 To n1
    a(i) = b(i)
   Next i
  End If
 Loop While n >= 2

 If n = 1 Then 'obtain last single real root
  roots(count).isComplex = False
  roots(count).real = -b(2)
  roots(count).imag = 0
 End If
 lbpolyroots = True

End Function

Function DeterPoly(ValEspHas() As Double, ValResEsp() As Double, NumEsp As Integer, grau As I
nteger, Polinomio() As Double)

'Determina o polinomio a partir da matriz de dados colocada pelo usuário

 Dim esp() As Double 'espacamento entre hastes
 Dim res() As Double 'resistividade para os espacamentos entre hastes
 Dim som() As Double 'soma dos espacamentos elevados a uma determinada potencia
 Dim prod() As Double 'soma dos produtos dos espacamentos a uma determinada potencia e a resi
stividade
 Dim MatrizCoe1(20, 20) As Double 'matriz dos coeficientes do sistema de equacao para determi
nacao do polinomio
 Dim MatrizCoe2(20, 20) As Double
 Dim MatrizCons1(20, 20) As Double 'matriz das constantes do sistema para determinacao do pol
```

Polinomios - 4

```
inomio
 Dim MatrizCons2(20, 20) As Double
 Dim MatrizCoefPol(20, 20) As Double
 Dim MatrizCoefPolDer() As Double
 Dim i As Integer
 Dim j As Integer
 Dim k As Integer
 Dim numRows As Integer
 Dim numCols As Integer

 'ordemPoly = montar um botao para escolher a ordem do polinomio
 Dim ordemPoly As Integer
 ordemPoly = grau

'verificar
ReDim esp(frmMain.txtCfgNumlin + 1)
ReDim res(frmMain.txtCfgNumlin + 1)

 For i = 0 To NumEsp - 1
  esp(i) = ValEspHas(i)
  res(i) = ValResEsp(i)
 Next i

 'Verificar
 ReDim som(2 * grau + 1)
 ReDim prod(2 * grau + 1)
 'som pode ser unidimensional corrigir acima e abaixo
 'prod idem

 For i = 1 To 2 * ordemPoly
  som(i - 1) = 0
  For j = 0 To NumEsp - 1
   som(i - 1) = esp(j) ^ i + som(i - 1)
  Next j
 Next i

 For i = 0 To ordemPoly
  prod(i) = 0
  For j = 0 To NumEsp - 1
   prod(i) = esp(j) ^ i * res(j) + prod(i)
  Next j
 Next i

 For i = 0 To ordemPoly
  k = i - 1
  For j = 0 To ordemPoly
   If i = 0 Then
    If j = 0 Then
     MatrizCoe1(i, j) = ordemPoly + 2
    Else
     MatrizCoe1(i, j) = som(j - 1)
    End If
   Else
    MatrizCoe1(i, j) = som(k)
    k = k + 1
   End If
  Next j
  MatrizCons1(i, 0) = prod(i)
 Next i

 numRows = ordemPoly + 1
 numCols = 1

 CopyMat MatrizCoe2, MatrizCoe1, numRows, numRows - 1
 CopyMat MatrizCons2, MatrizCons1, numRows, numCols '-1

 GaussJordan MatrizCoe2, MatrizCons2, numRows, numCols

 CopyMat MatrizCoefPol, MatrizCons2, numRows, numCols '-1

 For i = 0 To ordemPoly
 Polinomio(i) = MatrizCoefPol(i, 0)
 Next i
```

Polinomios - 5

```
 ReDim CoefPol(ordemPoly + 1) As Double
 For i = 0 To ordemPoly
 CoefPol(i) = Polinomio(i)
 Next i


End Function


Function ValorPoly(CoefPoly() As Double, grau As Integer, valorVariavel As Double) As Double
'Esta função retorna o valor do polinomio no ponto escolhido
'Não está sendo usada
 Dim i As Integer

 ValorPoly = 0
 For i = 0 To frmMain.g_iGrauDoPolinomio - 1
  ValorPoly = CoefPoly(i) * valorVariavel ^ i + ValorPoly
 Next i

End Function

Function Polynomio(coeficientes() As Double, grau As Integer, absissa As Double) As Double
'Esta funcao retorna o valor do polinômio de grau "grau" e coeficientes expressos na matriz _
'coeficientes para o valor da absissa "absissa

Dim p As Integer
On Error GoTo Err_polinomio

Polynomio = 0
For p = 0 To grau
Polynomio = Polynomio + (coeficientes(p) * absissa ^ p)
Next p

Exit Function

Err_polinomio:
Polynomio = 0
MsgBox "Erro ao calcular o valor do Polinomio neste ponto (" & absissa & ").Verifique o grau
e os coeficientes.", vbOKOnly, "MTerra - Erro"

End Function

Function Sign(X As Integer) As Double
If X = 0 Then
 Sign = 1
 Else
 Sign = -1
 End If

End Function
```