

# Métodos Numéricos Computacionais

Teoria

Exemplos

Programas

Marco Antônio Rahal Sacoman

BCC e BSI - 2017

# Apresentação que se divide em três partes

## Parte 1 - Uma apresentação que mais se parece com uma advertência

A Parte 1 trata de um fato corriqueiro conhecido como *eu nunca leio as primeiras páginas de um livro*. Todos deveriam ler estas páginas introdutórias e, exatamente por este motivo, foi explicitado que se trata de uma advertência. É melhor ler do que futuramente pensar *fui traído, pois não era o livro que eu pensava ser*.

### Sobre o texto

*Este texto é parte integrante do material didático complementar da disciplina Métodos Numéricos Computacionais ministrada nos cursos BCC e BSI da FC-Unesp-Bauru.*

Se, ao ler a frase anterior, você sentiu alguma dificuldade em entender completamente seu teor, este texto não é para você, mas sendo o texto de apresentação e estando com muita paciência, achando que a vida é bela, vou detalhar: *este texto é parte integrante do material didático complementar da disciplina Métodos Numéricos Computacionais ministrada nos cursos BCC e BSI da FC-Unesp-Bauru*. Agora, com a explicação, ficou claro.

Então, se você é aluno do BCC ou do BSI e é aluno de MNC, pode ser que este texto seja para você. Se você preenche os requisitos da frase anterior e seu professor é o Marco Antônio, eureka, este texto é para você. E, se não for para você, nada impede que leia. Mas, nesta situação, qualquer livro de MNC será mais adequado para você, pois este depende das explanações realizadas em aula.

Este texto é realmente para alunos de MNC do BCC ou do BSI para ser utilizado como material complementar às aulas. É claro que está mais do que claro que é necessário frequentar as aulas. Usando desta maneira, você terá suporte gratuito, durante todo o semestre letivo, para qualquer dúvida sobre o material.

Espero que tenha apreciado a Parte 1 da Apresentação e pondere sobre utilizar ou não este texto.

### Só para alunos da disciplina

Consulte regularmente o endereço [sacoman.dco.fc.unesp.br](http://sacoman.dco.fc.unesp.br) e cutuque o item MNC.  
Foi feito para você.

## Parte 2 - Uma apresentação que se parece com apresentação mesmo

### Aulas: elas existem

Aula é aula e o resto é o resto.

Este texto contém um resumo que é razoavelmente completo, mas é um resumo, da teoria que é apresentada em aula. Sim, você tem que participar das aulas.

Contém exemplos com aquelas peculiaridades típicas de texto complementar: exemplos teóricos, exemplos práticos, mais exemplos e outros exemplos. Pratique, pois ajuda no aprendizado. Esta frase *pratique, pois ajuda no aprendizado*, está escrita aqui e somente aqui. Não será repetida em nenhum outro local deste texto.

Este texto contém, como já escrito em página anterior, que certamente você não pulou e leu, os itens Teoria, Exemplos e Programas.

## **Teoria**

Um resumo da teoria que é apresentada em aula. Não mais do que isto.

Na primeira aula você conhecerá a bibliografia e poderá decidir em ter um dos livros, copiar a teoria apresentada ou ... na primeira aula o assunto será discutido.

## **Exemplos**

Qualquer livro tem exemplos excelentes ou bons ou até medíocres.

Neste texto, você encontrará apenas uma quantidade pequena de exemplos que servirão para você verificar se sabe como resolver problemas que envolvam MNC.

## **Programas**

Isto é um capítulo à parte em sua formação.

Se você cursa BCC ou BSI e acha que MNC não requer ou não deve requerer que você faça programas de qualidade, então, este texto não é para você. E, com o devido respeito, nem a disciplina.

A avaliação contempla provas e trabalhos computacionais e é altamente recomendado que você faça todos os programas e os faça com vontade, determinação, entusiasmo, ânimo e excitação. É isto mesmo, com T. (de Trabalho). Aluno de computação que não tem entusiasmo ao programar, não merece um belo diploma de graduação em computação e nem o tão almejado adjetivo nerd. : - )

Agora você já sabe o que deverá fazer para seu sucesso na disciplina



---

Alunos normais de computação.  
Frequente as aulas e encontrará alguns.

## Parte 3 - Uma apresentação complementar

### Aluno de curso de computação tem que computar

A disciplina é Métodos Numéricos Computacionais.

Você é aluno de computação.

Então, você fará trabalhos computacionais eficientes e bonitos.

### Trabalhos Computacionais ou, simplesmente, Programas

Você poderá desenvolver seus trabalhos na linguagem que quiser, para serem executados em qualquer sistema operacional.

Contudo, só poderei verificar seus programas se eles executarem nos computadores que tenho à minha disposição.

Isto também é assunto para ser discutido na primeira aula. E é importante. Não perca.

### P&R

P - Mas eu não tenho muita intimidade com programação. Preciso mesmo fazer os programas?

R - Precisa

P - Mas eu não sei programar. Preciso mesmo fazer os programas?

R - Precisa

P - Mas eu preciso mesmo fazer os programas?

R - Precisa



---

Aluno com cara de ajustado e com pensamento determinado. Entenda esta piada sem graça lendo o Capítulo Ajuste de Curvas.

# Conteúdo

## Parte 1

1 - Escolha do ambiente de desenvolvimento para programar	10
2 - Cálculo Analítico, Cálculo Numérico, Métodos Numéricos e a Computação	13
3 - Erro, erros e mais erros, pois eles existem	16
4 - Cálculo de Funções por Séries de Potências	28
5 - Desenhar é preciso e desenhar funções é fundamental	34
6 - Avaliando funções e, como bônus, um avaliador de funções	52

## Parte 2

7 - Derivadas	65
8 - Raízes de funções de uma variável	87
9 - Sistemas de equações lineares	110
10 - Interpolação	147
11 - Ajuste de Curvas	167
12 - Integrais	201
13 - Sistemas de equações não-lineares	235
14 - Equações Diferenciais Ordinárias	245

## Parte 3

15 - Dicas, truques e quebra-galhos para desenvolver programas	246
16 - Seja curioso e leia isto agora	252

# Índice

## Parte 1

1 - Escolha do ambiente de desenvolvimento para programar	10
Uma lembrança histórica	10
Um lembrete histórico (e não se esqueça disto)	11
Obtendo o IDE Lazarus	12
Instalando o IDE Lazarus	12
Executando o Lazarus pela primeira vez	12
Uma palavra de alento	12
2 - Cálculo Analítico, Cálculo Numérico, Métodos Numéricos e a Computação	13
P&R	15
3 - Erro, erros e mais erros, pois eles existem	16
Erros do problema	16
Erros do método	16
Erros do algoritmo	17
Erros do programa	17
Erros do computador	17
Sistemas de numeração	17
Conversão decimal-binário	18
Conversão binário-decimal	19
Quando a conversão decimal-binário é finita	20
Sistema de ponto flutuante	20
Algarismos significativos	21
Exemplos de erros do programa e do computador	21
Exemplo de erro do autor do programa	21
Exemplo de erro do computador	24
Controlando os erros	26
4 - Cálculo de Funções por Séries de Potências	28
Séries de Taylor	28
Exemplos para algumas funções	29
Funções para testes computacionais de convergência	29
Programa para testar a convergências de algumas destas séries	30
5 - Desenhar é preciso e desenhar funções é fundamental	34
Usar a força bruta ou usar componentes adequados para desenhar gráficos	34
Desenhando em uma Fórmula	36
Desenhando em um Componente Caixa de Pintura (TPaintBox)	39
Desenhando em um Componente Gráfico (TChart)	41
Desenhando uma função em um TPaintBox	45
Desenhando funções em um TChart	48
Sobre Rosa e Azul, Renoir e MASP	50
P&R	51
Sugestão da Patricinha	51
6 - Avaliando funções e, como bônus, um avaliador de funções	52
O interpretador de funções	53

Breve descrição das rotinas da biblioteca Interpretador.dll	53
FxR1	54
FxRn	54
ErrorToStr	54
Exemplos de utilização	54
Um exemplo completo - versão para console (DOS)	54
Um exemplo completo - versão enjanelada (Windows)	57
P&R	62
Um exemplo prático - o programa da Patricinha	62
P&R	64

## Parte 2

7 - Derivadas	65
Derivada de funções de uma variável	66
Ensaizando um algoritmo	67
O tamanho de h	67
Diferenciabilidade de f	67
O motivo do cálculo da derivada	68
Um algoritmo bruto, curto e grosso, mas eficiente	68
Derivada de ordem superior de funções de uma variável	69
Derivada de função de várias variáveis	70
Derivada de ordem superior de função de várias variáveis	71
Sugestões para seus programas	73
Instruções claras e precisas sobre trabalhos computacionais para avaliação	75
Um exemplo de programa	75
8 - Raízes de funções de uma variável	87
Método da Busca Uniforme	88
Método da Divisão ao Meio	90
Método das Cordas	91
Método de Newton	93
Método das Cordas Modificado	96
Método de Newton Modificado	96
Sugestões para seus programas	97
Instruções claras e precisas sobre trabalhos computacionais para avaliação	97
Um exemplo de programa	98
9 - Sistemas de equações lineares	110
Queimando a mão com a chapa quente do Tião	110
Recordar é viver ou, de volta ao Colegial	111
Equação linear e sistema de equações lineares	111
Classificação de um sistema de equações lineares quanto à solução	111
Classificação da matriz A de um sistema de equações lineares	112
Classificação dos métodos para solução de sistemas de equações lineares	112
Solução de sistemas de equações lineares por Métodos Diretos	113
Sistema triangular superior	113
Sistema triangular inferior	113
Método da Eliminação de Gauss	113

Método de Gauss com Pivotamento Parcial	116
Método de Gauss com Pivotamento Total	117
Método de Decomposição L U	118
Seção Fofoca	120
Método de Gauss Compacto	121
Método de Cholesky	122
Cálculo do Determinante da Matriz	124
Cálculo da Matriz Inversa	124
Solução de sistemas de equações lineares por Métodos Iterativos	125
A escolha da norma para critério de parada	125
Método de Jacobi	126
Método de Gauss-Seidel	128
Galeria de Fotos	129
Sugestões para seus programas	130
Instruções claras e precisas sobre trabalhos computacionais para avaliação	131
Um exemplo de programa	131
10 - Interpolação	147
Interpolação polinomial	147
Interpolação polinomial utilizando grau menor que n	148
Método de Newton para interpolação polinomial	149
Método de Newton-Gregory para interpolação polinomial	152
Sugestões para seus programas	155
Instruções claras e precisas sobre trabalhos computacionais para avaliação	156
Um exemplo de programa	156
11 - Ajuste de curvas	167
Dona Mariquinha e seu aluno Joãozinho	167
Interpolar ou ajustar? Eis a questão	168
Ajuste e Tendência	168
Técnica dos Mínimos Quadrados	168
Ajuste de reta $\hat{y} = a + b \cdot x$	170
Coeficiente de Determinação	171
Técnica dos Mínimos Quadrados - continuando	171
Ajuste de polinômio $\hat{y} = a_0 + a_1 \cdot x + a_2 \cdot x^2 + \dots + a_m \cdot x^m$	171
Ajustes de equações exponenciais e outras mais	172
Exponencial do tipo $\hat{y} = a \cdot b^x$	173
Exponencial do tipo $\hat{y} = a \cdot e^{b \cdot x}$	173
Geométrica do tipo $\hat{y} = a \cdot x^b$	174
Hiperbólica do tipo $\hat{y} = 1 / (a + b \cdot x)$	174
Outras equações, para você desenvolver o procedimento	175
Sugestões para seus programas	175
Instruções claras e precisas sobre trabalhos computacionais para avaliação	176
Um exemplo de programa	176
12 - Integrais	201
Regra dos Retângulos à Esquerda	202
Regra dos Retângulos à Direita	203

Regra dos Trapézios	204
Regra de Simpson	205
Seção Fofoca	205
Regra de Simpson - Após Seção Fofoca	206
Regra 1/3 de Simpson	206
Regra 3/8 de Simpson	208
Quadratura de Gauss	210
Patricinha volta à cena	213
Sugestões para seus programas	216
Instruções claras e precisas sobre trabalhos computacionais para avaliação	217
Um exemplo de programa	217
Cálculo de Pesos e Pontos de Gauss	233
13 - Sistemas de equações não-lineares	235
Equações lineares x equações não-lineares	237
Método de Newton	238
Método de Newton Modificado	241
Sugestões para seus programas	242
Instruções claras e precisas sobre trabalhos computacionais para avaliação	243
Um exemplo de programa	243
14-Equações Diferenciais Ordinárias	245

## Parte 3

15 - Dicas, truques e quebra-galhos para desenvolver programas	246
DTQ 1 - Escolhendo a Fonte para a fôrma, textos e controles em geral	246
DTQ 2 - Prevenindo cintilação quando a fôrma é atualizada constantemente	246
DTQ 3 - Prevenindo que o programa congele por um longo tempo	246
DTQ 4 - Break não é pecado	247
DTQ 5 - Definindo as espessuras das barras de rolagem	248
DTQ 6 - Executando arquivos externos	248
Pensamento Recursivo	249
DTQ Extra - Menu Irritantemente Piscante da versão 1.6.4 do Lazarus	250
16 - Seja curioso e leia isto agora	252

# 1- Escolha do ambiente de desenvolvimento para programar

## **Uma lembrança histórica**

Em março de 1968, tudo parecia calmo na área da programação de computadores e, de repente, *Edsger Wybe Dijkstra (1930-2002)* escreve uma carta ao editor da conceituada revista *Communications of the Association for Computing Machinery* (CACM), tecendo algumas considerações a respeito da qualidade de alguns tipos de programas. Esta famosa carta causou um furor tão grande na comunidade científica que passou a influenciar, de forma decisiva, todo o pensamento posterior sobre programação.

Nesta carta, Dijkstra cita o artigo *Uma Contribuição para o Desenvolvimento do Algol*, publicado no número 9 da CACM de Junho de 1966, Seção 3.2.1.

*Niklaus Wirth (1934-)* que já havia se pronunciado a respeito destes fatos e, concordando com o ponto de vista de Dijkstra, desenvolveu, com sua equipe da Universidade Técnica de Zurique, a Linguagem Pascal, derivada da Linguagem Algol 60, com uma implementação mais simples e com estrutura de dados mais poderosa.

Pascal é uma linguagem de programação estruturada, que recebeu este nome em homenagem ao matemático e físico Blaise Pascal. Foi criada em 1970, tendo em mente encorajar o uso de código estruturado.

É uma linguagem que permite que o programador elabore o algoritmo e traduza para a linguagem simbólica de forma simples, direta e objetiva.

Niklaus Wirth diz que a linguagem Pascal foi criada simultaneamente para ensinar programação estruturada e para ser utilizada na sua fábrica de software. Simultaneamente, a linguagem reflete a liberdade pessoal de Wirth das restrições impostas após seu envolvimento com a especificação de ALGOL 68 e sua sugestão para essa especificação, o ALGOL W.

Pascal originou uma enorme gama de dialetos, podendo também ser considerada uma família de linguagens de programação. Grande parte de seu sucesso se deve à criação, na década de 1980, da linguagem Turbo Pascal, inicialmente disponível para computadores baseados na arquitetura 8086 (com versões para 8080 no seu início).

Pascal é normalmente a linguagem escolhida para ensinar programação já que permite uma natural transformação de um algoritmo em um programa. Comercialmente, a linguagem foi sucedida pela criação da linguagem Object Pascal, atualmente utilizada no IDE Embarcadero Delphi, IDE Kylix e IDE Lazarus.

Pascal possui padrões ISO, como o Pascal Standard e o Advanced Pascal.

Muitos pensaram, trabalharam e concluíram que era necessário que se criasse uma linguagem simples e eficiente para programação. Tudo isto e você imagina que para se iniciar em algoritmos e programação você deve utilizar C ou Java? Sinto muito por você.

Mais um pouquinho de doutrinação.

Alguém disse: quero fazer 10 vezes ...

- um matemático escreveu:  $i = 1, \dots, 10 \dots$

- tradução: quero fazer 10 vezes ...

- um programador escreveu em Pascal: for  $i := 1$  to 10 do ...

- tradução: quero fazer 10 vezes ...

- outro programador escreveu em C: for ( $i = 1; i \leq 10; i++$ ) ...

- tradução:  $i$  vale 1; quero dizer,  $i$  é menor ou igual a 10; esqueça isto,  $i$  é  $d+$  ...

Alguém disse: leia  $x$ , incremente  $x$  de uma unidade, escreva  $x$  ...

- um programador escreveu em Pascal: Read( $x$ ); Inc( $x, 1$ ); Write( $x$ ) ...

- tradução: leia  $x$ , incremente  $x$  de uma unidade, escreva  $x$  ...

- outro programador escreveu em C: scanf("%d", & $x$ );  $x++$ ; printf("%d",  $x$ ) ...

- tradução: nem vou tentar traduzir isto.

Fim da conversa sobre doutrinação.

### **Um lembrete histórico (e não se esqueça disto)**

Você poderá escrever seus programas em qualquer linguagem, mas neste texto os programas serão convertidos, a partir dos algoritmos, de forma simples e clara, utilizando a Linguagem Pascal.

Como é desejável que os programas sejam corretos e agradáveis de serem utilizados, deverão ser programas visualmente agradáveis. Para isto, pode-se utilizar Delphi ou Lazarus.

Considerando que a última versão de Delphi é mais sofisticada do que a última versão de Lazarus, alguns desejarão utilizar Delphi. Considerando que Delphi é um produto vendido e que Lazarus é um produto gratuito, alguns desejarão utilizar Lazarus.

É um dilema. Além disto, existe versão gratuita de Delphi para estudantes. Então todos desejarão utilizar Delphi. Mas esta versão gratuita é sensivelmente limitada. Então todos desejarão utilizar Lazarus. Além disto, quem diz que é partidário do software livre, deveria escolher Lazarus. Você decide.

Então, para acabar com a discussão, o produto eleito é Lazarus. E, para acabar mesmo, ligue os pontos 1, 2, 3, 4 aos pontos A, B, C, D, de preferência, corretamente.



## **Obtendo o IDE Lazarus**

Inicie seu navegador Web, solicite que o simpático navegador lhe apresente o conteúdo do endereço <http://www.lazarus-ide.org/> e leia a página. Se estiver muito ansioso, não leia nada e cutuque o botão Download Now. A versão disponível, no momento em este texto está sendo escrito, é a versão Lazarus 1.6.4, lançada em 1º de março de 2017, que utiliza o compilador FPC 3.0.2.

Para obter o instalador, haverá um redirecionamento <https://sourceforge.net>. Aguarde até que o instalador seja copiado e é conveniente que você o armazene em um diretório de fácil lembrança e acesso.

## **Instalando o IDE Lazarus**

Execute o instalador e aceite todas as sugestões, pressionando o botão para continuar a instalação. A única observação é a que segue e é extremamente importante. Não instale o Lazarus em Arquivos de Programas ou Program Files ou qualquer outro diretório. Instale na raiz, ou seja, em C:\. Se você considerar que isto é pouco profissional, saiba que concordo. Mas é assim que deve ser feito.

## **Executando o Lazarus pela primeira vez**

Ao executar o Lazarus pela primeira vez, será apresentada uma janela com várias abas e, para cada uma delas, existirão informações sobre os diretórios que o Lazarus deverá utilizar. Sempre que houver um OK ao final da exibição, aceite e não altere nada. Caso não haja um OK, o diretório correto deverá ser informado.

A seguir, o IDE será iniciado e poderá ser utilizado normalmente.

Não se assuste se o menu ficar piscando. É um erro da versão atual (1.6.4). Existe um remendo para resolver isto. Basta modificar o arquivo sourcefilemanager.pas do diretório C:\Lazarus\ide, e recomilar o Lazarus. A correção incorporada no instalador só deverá ocorrer na versão 1.8. Informações adicionais serão oferecidas em aula. Ou, use Delphi. :- ) Informações adicionais serão oferecidas em aula.

Divirta-se programando em um ambiente simpático.

## **Uma palavra de alento**

O melhor ambiente de programação é aquele que o programador conhece bem e a melhor linguagem é aquela que o programador conhece bem. Se alguém conhece bem um IDE e uma Linguagem, deve utilizá-los. Caso contrário, deve se iniciar, nesta arte, com um ambiente de fácil utilização e com uma linguagem de clareza absoluta.

Resumindo, decida-se sobre como vai desenvolver seus trabalhos e desenvolva-os. Pascal, C, C++, C#, Java, HTML com Javascript, HTML com PHP, PERL, HTML com PERL, Ruby, Ruby on Rails, Ruby fora dos Rayls, FORTRAN (minha primeira linguagem), LUA ou qualquer outro planeta. Pode até utilizar Flash ou Silverlight. E poderá fazer programas para Android, Windows Phone ou aiFone. A decisão é sua.

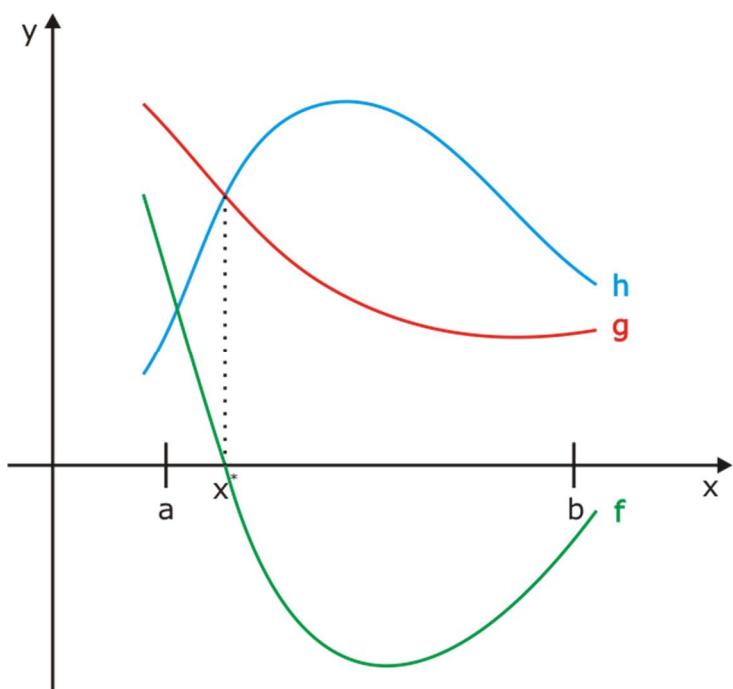
## 2- Cálculo Analítico, Cálculo Numérico, Métodos Numéricos e a Computação

Você já sabe que isto é apenas um resumo do que foi visto em aula.

Considere um problema qualquer, que envolva cálculos e que você tenha que oferecer a solução.

Como exemplo, o problema descreve um comportamento que pode ser (ou não) visto em um gráfico. Além disto, há um segundo comportamento. Deseja-se calcular o ponto onde os dois comportamentos resultam em um mesmo valor.

Graficamente, esta singela história pode ser vista como segue.



$g$  representa um comportamento e  $h$  representa outro comportamento.

Os dois comportamentos terão o mesmo valor nos pontos onde  $g(x) = h(x)$ .

Pode existir um único ponto  $x$  que satisfaça esta condição ou mais de um ou nenhum.

Para  $g(x) = h(x)$ , tem-se  $g(x)-h(x) = 0$ .

Se  $f(x) = g(x)-h(x)$ , então  $f(x) = 0$  oferece a solução do problema.

No exemplo, para a região apresentada, existe uma solução que é  $x^*$ .

Só para se divertir, imagine que  $g$  seja aquela função macabra que você fica estatelado ao vê-la. Imagine que  $h$  seja um pouco mais macabra. Se puder, imagine a cara de  $f$ . O seu problema é apenas encontrar o ponto onde  $f(x) = 0$ . Lembre-se que pode haver apenas um ponto, mais de um ou nenhum.

Use tudo aquilo que você aprendeu em Cálculo e resolva. Lembre-se que  $f$  é filha da mula sem cabeça com o curupira. Está difícil? Então continue lendo.

E agora, o grande marketing de MNC.

Olhe para a cara do gráfico e veja onde está a solução. É isto mesmo, olhe para a cara do gráfico. E se não houver gráfico com ou sem cara. Imagine como seria ou, nem imagine.

Considere que se procura uma ou mais raízes no intervalo  $[a, b]$ . Parta do ponto  $a$ , faça  $x_k = a$ , caminhe com pequenos passos de tamanho  $\Delta$  até o ponto  $b$  e, para cada novo ponto  $x_{k+1} = x_k + \Delta$ , verifique se  $f(x_{k+1})$  tem sinal diferente de  $f(x_k)$ . Se isto ocorrer, há uma raiz neste intervalo  $[x_{k-1}, x_k]$ . Simples

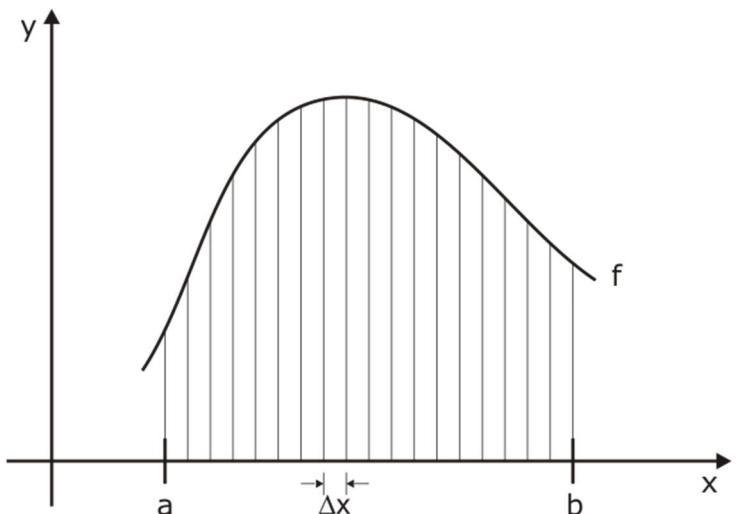
assim. É claro que para encontrar a raiz neste novo intervalo  $[x_{k-1}, x_k]$  deve-se refinar o procedimento, mas isto é assunto para o capítulo específico que trata do cálculo de raízes. Isto foi apenas uma ideia que mostra a simplicidade da busca numérica da solução.

É evidente que em um método de busca da raiz, como este proposto, haverá um erro proporcional ao valor de  $\Delta$  ou de outro valor utilizado no refinamento. E é sempre assim com os métodos numéricos, ou seja, há um erro, mas este erro é previsível e contornável através da seleção de parâmetros selecionáveis para a busca do resultado.

Mais um exemplo. Lembra-se daquela integral encapetada que você não resolveu nem por partes nem sem partes? E lembra-se, também, da definição de integral, não é mesmo?

Então, simplesmente utilize a definição e, onde está escrito  $n \rightarrow \infty$  ou  $\delta x \rightarrow 0$ , escreva  $\delta x = \Delta x$ , com  $\Delta x$  pequeno. A soma das faixas infinitesimais será substituída pela soma de faixas de largura  $\Delta x$  e o valor da integral será aproximado, onde a precisão será proporcional ao tamanho de  $\Delta x$ .

Graficamente, a brincadeira fica como segue.



Pode-se calcular o valor numérico da integral como  $\int_a^b f(x) dx \approx \sum_{i=1}^n f(\bar{x}) \cdot \Delta x$ .

Da conhecida definição de integral, sabe-se que  $\int_a^b f(x) dx = \lim_{n \rightarrow \infty} \sum_{i=1}^n f(\bar{x}) \cdot \Delta x$ , onde  $\bar{x}$  é o valor médio de  $f(x)$  para cada intervalo determinado por  $\Delta x$ .

Pode ser que você prefira a definição  $\int_a^b f(x) dx = \lim_{\delta x \rightarrow 0} \sum_{i=1}^n f(\bar{x}) \cdot \delta x$ , que dá tudo na mesma coisa.

O que interessa é que  $\delta x$  é infinitesimal e  $\Delta x$  não é.

Como descrito no caso da raiz, haverá um erro que poderá ser controlado fazendo  $\Delta x$  tão pequeno quanto se queira ou se possa fazer.

Mais um exemplo.

Era uma vez um grupo de pessoas que se candidatou a um maravilhoso emprego (deveria ser trabalho) e o entrevistador entregou para cada candidato uma lâmpada do tipo incandescente e pediu para que calculassem o volume da dita cuja. Ofereceu um laboratório com todo o material necessário, como escala triangular, paquímetro e outros medidores além de pipeta e bêquer, que muitos consideraram estranho, serra, que alguns riram ao ver, lápis, papel, calculadora, computador, wi-fi, facebook e por aí vai.

Os candidatos analíticos tomaram várias medidas, conjecturaram sobre a forma da superfície da lâmpada, estimaram a função desta superfície e calcularam a maravilhosa integral que resultou na

equação do volume da lâmpada. A seguir, aplicaram a esta equação os valores constantes para a lâmpada em questão e chegaram ao seu volume que, variou, dependendo do candidato, entre  $349,0568 \text{ cm}^3$  e  $351,0034 \text{ cm}^3$ . Levaram, em média, duas horas e 45 minutos para chegar à solução.

Um candidato numérico, um engenheiro caso não tenha entendido, procurou dentre o material oferecido uma serra e serrou o gargalo da lâmpada. Encheu a lâmpada com água, despejou a água em um recipiente graduado e, visualmente, verificou que o volume era  $350 \text{ cm}^3$ . O candidato demorou 15 minutos para resolver o problema, pois, caprichoso que era, serrou a lâmpada cuidadosamente.

Você pode considerar isto uma piada e rir ou, mesmo rindo, considerar que ... que nada, você já entendeu.

Reita mil vezes o que segue e já obterá mais 0,001 em sua média final.

Métodos analíticos servem para encontrar as equações que resolvem um problema ou uma classe de problemas e, ao se aplicar as constantes relativas a um determinado problema, encontra-se sua solução numérica.

Métodos numéricos servem para encontrar a solução numérica de um determinado problema.

Você quer a equação que resolve o problema? Então utilize um método analítico.

Você quer a solução numérica do problema? Então utilize um método numérico.

Você quer o volume da lâmpada? Então utilize serra, água e bêquer. :-)<sup>1</sup>

## P&R

P - Não seria mais esperto se o engenheiro mergulhasse a lâmpada no bêquer e verificasse a variação do volume?

R - Não. Mergulhar a lâmpada na água não é possível por ser leve. Forçar o mergulho poderia provocar erro devido ao instrumento utilizado para isto e, finalmente, a solicitação é para volume interno e, não, externo.



P - Mas e se...

R - Esqueça. A piada é como descrita acima. E agora que o problema foi resolvido, todo mundo quer inventar algo melhor. Ótimo, continue assim, um ser pensante, para resolver os próximos problemas.




---

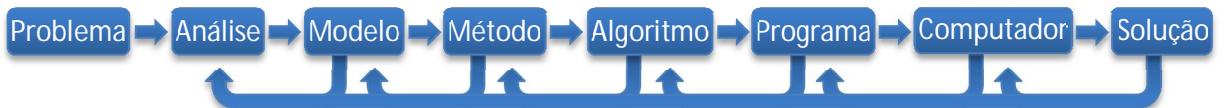
<sup>1</sup> Se ao ler este texto de (des)motivação para se interessar por Métodos Numéricos você entendeu que deve desprezar os Métodos Analíticos, você não entendeu nada e, por isto, deverá pagar com reprovações nas disciplinas Cálculo I, II, III, IV, V, VI, ..., Geometria Analítica I, II, ..., Álgebra Linear I, II, ..., ..., ... e até em MNC.

### 3- Erro, erros e mais erros, pois eles existem

Quem é que erra? Todo mundo erra. O problema erra, o método erra, o algoritmo erra e o computador erra. Então você não pode errar. Bastam os erros dos outros.

Imagine que em seu local de trabalho o seu chefe muito gentil lhe peça para resolver um sistema de equações lineares e lhe entrega uma folha com equações anotadas. Você só precisará resolver o sistema e devolver a folha com a solução anotada. Agora pare de sonhar e acorde para a realidade. Seu chefe, carrancudo, ordena que resolva um problema que está afetando o processo de expedição de produtos para exportação e, evidentemente, espera uma solução adequada e urgente. Este é o mundo real.

Um procedimento considerado adequado para tais situações é apresentado no esquema que segue.



A cada etapa do procedimento, se for detectado que existe erro ou existe possibilidade de erro, deve-se voltar à etapa anterior ou anteriores, até que se chegue à solução que, de fato, resolva o problema.

Quando isto acontecer, você poderá apagar todas estas setas com ligações arredondadas e dizer ao chefe carrancudo que resolveu o problema com o esquema linear que sai do problema e chega à solução. Poderá ganhar uma promoção ou mais problemas para resolver.

#### Erros do problema

O problema real existe e é chamado problema, pois é realmente um problema. Por vezes, um problema de difícil solução. Olha-se para aquele caos e tenta-se entender o que ocorre. É feita a análise do problema, onde sempre cabem diferentes interpretações. Com a interpretação que se acredita ser válida, pode-se optar por uma ou mais de uma técnica para resolver o problema interpretado.

Adota-se ou desenvolve-se um modelo matemático que se acredita ser representativo da interpretação do problema.

Até este instante, vários erros já podem fazer parte do problema. O que pode servir de consolo, para você que está estudando métodos numéricos, é que esta parte do esquema é desenvolvida por um especialista na área do problema e você passará a interagir na solução, a partir do modelo que será escolhido por você ou pelo especialista da área e você.

#### Erros do método

Estabelecido um modelo, podem existir diferentes métodos para resolvê-lo. Existe neste instante a possibilidade de acumular mais erros pela escolha de um método que não seja o mais adequado. Além disto, se for criado um método próprio para o modelo em questão, o próprio método poderá conter erros.

## Erros do algoritmo

Para o método selecionado, podem-se utilizar ou desenvolver diferentes algoritmos. Neste instante, há mais possibilidades de erros.

O algoritmo é o elo entre o método e o programa e deveria ser facilmente estabelecido, mas nem sempre é. Há, neste planeta, uma quantidade incontável de pessoas que escrevem algoritmos como se estivessem conversando em um batequim. E depois, alguém terá que ler o algoritmo e escrever o programa.

O algoritmo deve ser preciso, mas o programa gerado poderá não ser. Suponha que o algoritmo estabeleça que se  $(a - b) < \varepsilon$ , pare. Esta condição poderá nunca ser atingida se o programador não utilizar os tipos adequados para as variáveis e não conhecer os erros que o computador poderá cometer.

O algoritmo deve convergir e ser rápido e, assim, deverá levar a resultado adequado em tempo adequado.

## Erros do programa

Para o algoritmo selecionado, podem-se criar diferentes programas. E neste instante é vital que o autor do programa<sup>2</sup> saiba programar.

A escolha da linguagem deve ser adequada, o programador deve ser maior de idade, vacinado e portador de RG, CPF e CNH. E quase que me esqueço, deve ser alguém que aprendeu o que deveria aprender nas disciplinas introdutórias do curso. Em MNC este alguém terá a grande oportunidade de colocar em prática tudo isto.

## Erros do computador

Erros numéricos podem acontecer. E estes erros são devidos à forma como o computador armazena os números e efetua cálculos. Tudo aquilo que você aprendeu nas disciplinas introdutórias do curso e que não é apresentado neste texto. Contudo, em aula será demoradamente discutido. Lembre-se que isto é um resumo. Mas mesmo sendo um resumo, seguem lembretes sobre sistemas de numeração.

## Sistemas de numeração

Os humanos tem 10 dedos e, desde criancinhas, são ensinados a utilizarem os dedinhos e contarem de 1 até 10. Muito bom, mas se aprendessem a contar de 0 até 9, seria perfeito. Já cresceriam entendendo que de 0 até 9 existem 10 números que são os 10 algarismos do sistema decimal e que os próximos números são compostos dos algarismos já conhecidos. Assim funciona qualquer sistema de numeração e segue uma tabela do óbvio, com alguns valores expressos através dos sistemas binário, ternário, octal, decimal e hexadecimal, ou seja, sistemas com base 2, 3, 8, 10 e 16.

---

<sup>2</sup> Por autor do programa entenda programador, analista, engenheiro de software ou outra raça qualquer que se meta a desenvolver um programa.

Valor	Base				
	2	3	8	10	16
0	0	0	0	0	0
1	1	1	1	1	1
2	10	2	2	2	2
3	11	10	3	3	3
4	100	11	4	4	4
5	101	12	5	5	5
6	110	20	6	6	6
7	111	21	7	7	7
8	1000	22	10	8	8
9	1001	100	11	9	9
10	1010	101	12	10	A
11	1011	102	13	11	B
12	1100	110	14	12	C
13	1101	111	15	13	D
14	1110	112	16	14	E
15	1111	120	17	15	F
16	10000	121	20	16	10
17	10001	122	21	17	11
18	10010	200	22	18	12
19	10011	201	23	19	13
20	11000	202	24	20	14
...					
32	100000	1012	40	32	20
64	1000000	2101	100	64	40
128	10000000	11202	200	128	80
256	100000000	100111	400	256	100
512	1000000000	200222	1000	512	200
1024	10000000000	1101221	2000	1024	400
1234	10011010010	1200201	2322	1234	4D2
2047	11111111111	2210211	2147	3777	7FF

## Algarismos e números

### Números

Para qualquer base, o valor é representado por  $V = a_n.b^n + a_{n-1}.b^{n-1} + \dots + a_1.b^1 + a_0.b^0$ .

Usando o valor 19 como exemplo, tem-se as conversões que seguem.

Base 2

$$V = 1.2^4 + 0.2^3 + 0.2^2 + 1.2^1 + 1.2^0$$

$$V = 16 + 0 + 0 + 2 + 1 = 19$$

Base 3 (muito útil para bit de 3 estados) :-)

$$V = 2.3^2 + 0.3^1 + 1.3^0$$

$$V = 18 + 0 + 1 = 19$$

Base 8

$$V = 2.8^1 + 3.8^0$$

$$V = 16 + 3 = 19$$

Base 10

$$V = 1.10^1 + 9.10^0$$

$$V = 10 + 9 = 19$$

Base 16

$$V = 1.16^1 + 3.16^0$$

$$V = 16 + 3 = 19$$

Estendendo a representação do valor em qualquer base, para incluir os números reais com parte inteira e fracionária, tem-se  $V = a_n.b^n + a_{n-1}.b^{n-1} + \dots + a_1.b^1 + a_0.b^0 + a_{-1}.b^{-1} + a_{-2}.b^{-2} + \dots + a_{-m}.b^{-m}$ .

Considere o valor 13,8125 e sua representação binária 1101,1101. A conversão é apresentada como segue.  $V = 1.2^3 + 1.2^2 + 0.2^1 + 1.2^0 + 1.2^{-1} + 1.2^{-2} + 0.2^{-3} + 1.2^{-4} = 8 + 4 + 0 + 1 + 1/2 + 1/4 + 0 + 1/16 = 13,8125$ .

Com isto é possível converter números de uma base para outra. As bases de interesse neste texto são as bases 10 e 2, já que os humanos insistem em utilizar a base 10 e as máquinas, mais teimosas, insistem em utilizar a base 2.

## Conversão decimal-binário

A conversão de decimal inteiro para binário inteiro é efetuada dividindo-se o decimal por 2 e anotando-se o resto da divisão da direita para a esquerda até que o decimal seja nulo. Para a parte fracionária, multiplica-se o decimal por 2 e anota-se a parte inteira da multiplicação como sendo o próximo algarismo binário. Remove-se do decimal o valor 1 se a divisão resultou em valor maior ou igual a 1. Segue um exemplo para conversão do número decimal 13,8125 para binário.

Parte inteira	$13/2 = 6$ e resta <b>1</b>
	$6/2 = 3$ e resta <b>0</b>
	$3/2 = 1$ e resta <b>1</b>
	$1/2 = 0$ e resta <b>1</b>
	Os restos, de baixo para cima, formam o binário 1101.
Parte fracionária	$0,8125 \cdot 2 = 1,625$
	$0,625 \cdot 2 = 1,25$
	$0,25 \cdot 2 = 0,5$
	$0,5 \cdot 1 = 1,0$
	Os inteiros, de cima para baixo, formam o binário 0,1101.

Um algoritmo para converter de decimal para binário pode ser escrito como segue, onde D representa o número decimal e B representa o binário. No algoritmo, cada algarismo é adicionado ao número convertido, por concatenação à esquerda ou à direita, dependendo de conversão da parte inteira ou fracionária. Note que este algoritmo é literal, ou seja, para tratar cada algarismo como uma letra e brincar de conversão com papel, lápis e borracha. Ou, no caso de algoritmo numérico, deve-se atribuir o valor zero ou um ao bit correspondente. Uma boa tarefa de programação.

```

função DB(D): B
DI = Decimal inteiro, DF = Decimal fracionário, BI=vazio, BF = 0,
enquanto DI > 0 faça
    resto = resto de DI /2                                // será 0 ou 1
    DI = inteiro de DI /2
    BI = resto + BI                                     // concatenado à esquerda com BI
enquanto DF ≠ 0 faça
    DF = DF * 2
    se DF > 1 então DF = DF -1; BF = BF + 1          // concatenado à direita com BF
    senão, BF = BF + 0                                  // concatenado à direita com BF
  
```

Segue teste do algoritmo para o número decimal 13,8125. Tem-se D = 13,8125; DI = 13; DF = 0,8125.

DI	DI /2	resto	BI	DF	DF*2	BF	B
13	6	1	1				1
6	3	0	01				01
3	1	1	101				101
1	0, 5	1	1101				1101
				0, 8125	1, 625	0, 1	1101, 1
				0, 625	1, 25	0, 11	1101, 11
				0, 25	0, 5	0, 110	1101, 110
				0, 5	1	0, 1101	1101, 1101

### Conversão binário-decimal

A conversão de binário para decimal é efetuada multiplicando-se cada algarismo binário por potências de 2 correspondente à posição decimal do número. Para a parte fracionária, são efetuadas divisões das potências de 2. Segue um exemplo para conversão do número binário 1101,1101 para decimal.

Parte inteira	$1 \cdot 2^3 = 8$
	$1 \cdot 2^2 = 4$
	$0 \cdot 2^1 = 0$
	$1 \cdot 2^0 = 1$
	A soma dos produtos é o decimal 13.
Parte fracionária	$1 \cdot 2^{-1} = 0,5$
	$1 \cdot 2^{-2} = 0,25$
	$0 \cdot 2^{-3} = 0$
	$1 \cdot 2^{-4} = 0,0625$
	A soma dos produtos é o decimal 0,8125.

Neste caso não há algoritmo literal com concatenação e o algoritmo numérico é um simples somatório de produtos de potências dos algarismos do número binário. Então, fica para você exercitar seus conhecimentos de desenvolvimento de algoritmos. E depois de escrever o algoritmo, teste e verifique se número binário 1101,1101, convertido para decimal, resulta em 13,8125.

### **Quando a conversão decimal-binário é finita**

Considere o número decimal 12,34. Utilizando-se o algoritmo DB da seção **Conversão decimal-binário**, encontra-se o valor 1100,010101110000101000111101011100001010001111010111...

Basta seguir o algoritmo e você encontrará este resultado. É claro que o aluno esperto perguntará: Em que situação o resultado é finito? É a resposta é: quando o número real puder ser escrito como uma fração  $p/q$  com  $p$  inteiro e  $q$  sendo potência de 2.

Como exemplo, considere o decimal 13,5625 cuja representação binária é 1101,1001. A fração p/q é dada por  $217/16 = 217/2^4$ , que pode ser escrita como  $(208+9)/16 = 13+9/16 = 13+9/2^4$ . Considere o decimal 69,375 cuja representação binária é 1000101,011. A fração p/q é  $555/8 = 555/2^3$ , que pode ser escrita como  $(552+3)/8 = 69+3/8 = 69+3/2^3$ . Outro exemplo é o decimal 0,8125 cuja representação binária é 0,1101. A fração p/q é dada por  $13/16 = 13/2^4$ . O nome desta fração é fração diádica, ou seja, relativa à base 2. Todo número decimal que possa ser representado por uma fração diádica tem representação binária exata. Mas não fique muito feliz com isto, pois mesmo tendo representação binária exata, o valor decimal pode ser tal que o número de bits para a representação binária seja maior que a quantidade de bits disponível para armazenar o binário.

Resumo do resumo do resumo: o computador não representa corretamente todos os decimais que você utiliza e que continuará utilizando em seus programas.

Veja em [sacoman.dco.fc.unesp.br/mnc/decbin](http://sacoman.dco.fc.unesp.br/mnc/decbin) um conversor Decimal-Binário e Binário-Decimal que não se preocupa com o número de bits utilizados para armazenar decimais ou binários.

## Sistema de ponto flutuante

O nome já diz tudo. O ponto, que no caso deste texto é vírgula, flutua para a direita ou para a esquerda e em algum lugar existe a informação desta flutuação. O número  $12,34$  pode ser representado como  $0,01234 \cdot 10^3$ ,  $0,1234 \cdot 10^2$ ,  $1,234 \cdot 10^1$ , o próprio  $12,34 \cdot 10^0$ ,  $123,4 \cdot 10^{-1}$ ,  $1234 \cdot 10^{-2}$ ,  $12340 \cdot 10^{-3}$  e a forma normalizada é  $0,1234 \cdot 10^2$ . Alguém já deve estar perguntando: Qual é a vantagem disto? A resposta é simples. Se existe uma quantidade determinada de bits para armazenar um binário, um número muito pequeno ou um número muito grande não poderão ser armazenado, exceto se estiverem escritos com a notação de ponto flutuante.

O mesmo tipo de erro ocorre ao armazenar valores muito grandes. O decimal 123400000000000 é armazenado com 47 bits, como 1110000011101101001101001010100101000000000000, mas o decimal 123400000000000, 10 vezes o anterior, necessita, para seu armazenamento binário, 51 bits e é representado como 1000110001001010001000000110100111001000000000000.

A representação em ponto flutuante tem a forma  $F(b, t, m, n)$ , onde  $b$  é a base,  $t$  é o número de algarismos significativos,  $m$  e  $n$  são os limites dos expoentes. O número é dado por  $M \cdot b^e$  onde  $M$  é a mantissa e o expoente assume valores entre  $m$  e  $n$ .

### Algarismos significativos

Os algarismos significativos de um número são os algarismos diferentes de zero, contados à partir da esquerda, até o último algarismo diferente de zero à direita, para números sem vírgula inteiros e, até o último algarismo se houver vírgula. Seguem alguns exemplos.

Número	102	1,02	1,00	0,102	0,00102	0,01022	01,020
Algarismos significativos	3	3	3	3	3	4	4

### Exemplos de erros do programa e do computador

Seguem dois exemplos que apresentam erros que são cometidos frequentemente sem que o programador perceba a gravidade destes erros. O primeiro apresenta um programa que termina, mas não termina. Entendeu? Veja o programa. O segundo apresenta erros cometidos em simples somas.

Não perca o foco. Os exemplos contêm programas e, então, para chegar ao segundo exemplo, você terá o difícil trabalho de virar umas três páginas. Se optar por uma primeira leitura rápida, é recomendado que volte a ler cada exemplo de forma completa e atenta, além de reproduzi-los.

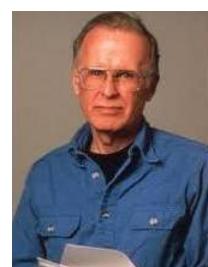
### Exemplo de erro do autor do programa

E depois de tudo isto, lembre-se que você tem que ter um mínimo de noção sobre o tempo computacional, para não escrever algoritmos que nunca terminam o processamento de seu problema. Suponha um algoritmo banal e que não termina nunca. Imaginou? Não? Então segue um exemplo.

Fazendo uma homenagem a John Backus<sup>3</sup>, autor da linguagem Fortran, considere que, implicitamente, as variáveis que iniciam com as letras i até n armazenem valores inteiros. Nada mais lógico, afinal **I**nteiro ... nem preciso explicar. Mas, para não exagerar, suponha as variáveis i, j, k, l, m, n sendo do tipo Byte. Armazenam apenas 256 valores diferentes. Faça um programa que tenha 6 ciclos aninhados com estas variáveis recebendo todos os seus possíveis valores. Captou a ideia?

---

<sup>3</sup> Backus queria ser químico, estudou alguns anos e resolveu ser médico. Estudou mais alguns anos e preferiu ser um cientista que pudesse criar algo novo. A programação de computadores, naquela época era algo incipiente e difícil. Backus criou a linguagem Fortran, de fácil utilização e de grande alcance para solução de problemas matemáticos. Foi assim que os engenheiros, físicos e matemáticos começaram a utilizar os computadores e o resto você já sabe, incluindo o fato de que os engenheiros ainda amam programar em Fortran. Backus é um dos autores da notação BNF (Backus-Naur Form). J. W. Backus nasceu em 3 de dezembro de 1924 e faleceu em 17 de março de 2007.

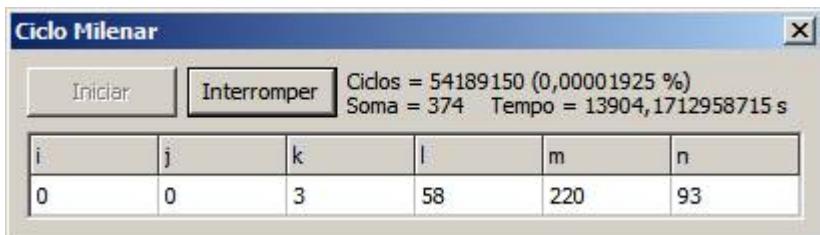


```

Para i = 0, ..., 255
  Para j = 0, ..., 255
    Para k = 0, ..., 255
      Para l = 0, ..., 255
        Para m = 0, ..., 255
          Para n = 0, ..., 255
            soma = i+j+k+l+m+n
            imprima soma
  
```

Faça este programa, com uma bela janela, compre um computador novo, compre um nobreak, execute o programa, mas não se esqueça de deixar um bilhete com os dizeres: Não desligue o computador nos próximos 1000 anos. Exagerei? Faça os cálculos e preveja o tempo que levará para o programa parar espontaneamente. Dá mais do que 1000 anos.

Se você é um incrédulo e está com preguiça de fazer o programa, simplesmente copie<sup>4</sup>. Segue uma receita grátis e a cara do programa é como segue. A fôrma é estilo diálogo, mas tem um belo ícone.



Crie um diretório chamado X:\meu diretório pessoal\NMC\Sou Teimoso, inicie o Lazarus<sup>5</sup>, inicie um novo projeto do tipo aplicação, salve o projeto neste diretório com o nome CicloMilenar.lpr e a unidade com o nome Unit1, atribua Caption = Ciclo Milenar, BorderStyle = bsDialog, Width = 404, Height = 92 e, para Font, Name = Tahoma e Size = 8. DTQ 1 Quase pronto.

Se não gostou do nome do diretório, troque por ...\\NMC\\1-Ciclo Milenar

Lance, na fôrma, alguns componentes como segue.

Componente	Left	Top	Caption	Width	Height	ColCount	RowCount	FixedCols
Button1	8	8	Iniciar					
Button2	88	8	Interromper					
Label1	168	8	Ciclos = 0 (0,0 %)					
Label2	168	20	Soma = 0					
Label3	240	20	Tempo = 0 s					
StringGrid1	8	40		388	44	6	2	0

Crie a rotina para o evento FormShow e escreva as linhas que seguem.

```

procedure TForm1.FormShow(Sender: TObject);
begin
  ControlStyle := ControlStyle + [csOpaque];
  DoubleBuffered := True;
  Button2.Enabled := False;
  Interrompe := False;
  with StringGrid1 do
  begin
    Cells[0, 0] := 'i';
    Cells[1, 0] := 'j';
    Cells[2, 0] := 'k';
    Cells[3, 0] := 'l';
    Cells[4, 0] := 'm';
    Cells[5, 0] := 'n';
  end;
end;
end;
  
```

DTQ 2

<sup>4</sup> E caso esteja com preguiça de copiar o programa, pegue um pronto no endereço informado na primeira aula.

<sup>5</sup> Qual o motivo de usar Lazarus? Se você fez esta pergunta, recomenda-se que volte ao texto introdutório e leia as explicações devidas. Contudo, você poderá reproduzir o exemplo utilizando qualquer linguagem com a qual esteja familiarizado.

Logo acima desta rotina defina a variável global Interrompe.

```

var
  Interrompe: Boolean;

Crie a rotina Button1Click para o evento onClick do Button1 e escreva as linhas que seguem.

procedure TForm1.Button1Click(Sender: TObject);
var
  i, j, k, l, m, n: Byte;
  TotalCiclos, Ciclos: Int64;
  Porcentagem: Real;
  Soma: Integer;
  Tempo1, Tempo2: TTime;
begin
  Button1.Enabled := False;
  Button2.Enabled := True;
  Button2.SetFocus;
  with StringGrid1 do
    begin
      Cells[0, 1] := '0';
      Cells[1, 1] := '0';
      Cells[2, 1] := '0';
      Cells[3, 1] := '0';
      Cells[4, 1] := '0';
      Cells[5, 1] := '0';
      TotalCiclos := 281474976710656; // ou := 256*256*256*256*256*256 ou := Power(256, 6);
      Ciclos := 0;
      Tempo1 := Now;
      for i := 0 to 255 do
        for j := 0 to 255 do
          for k := 0 to 255 do
            for l := 0 to 255 do
              for m := 0 to 255 do
                for n := 0 to 255 do
                  begin
                    Cells[0, 1] := IntToStr(i);
                    Cells[1, 1] := IntToStr(j);
                    Cells[2, 1] := IntToStr(k);
                    Cells[3, 1] := IntToStr(l);
                    Cells[4, 1] := IntToStr(m);
                    Cells[5, 1] := IntToStr(n);
                    Inc(Ciclos);
                    Porcentagem := 100*Ciclos/TotalCiclos;
                    Soma := i+j+k+l+m+n; // O máximo será 255*6 = 1530
                    Tempo2 := Now;
                    Label1.Caption := 'Ciclos = ' + IntToStr(Ciclos) +
                      ' (' + FloatToStrF(Porcentagem, ffNumber, 3, 8) + '%)';
                    Label2.Caption := 'Soma = ' + IntToStr(Soma);
                    Label3.Caption := 'Tempo = ' + FloatToStr(100000*(Tempo2-Tempo1)) + ' s';
                    Application.ProcessMessages; DTQ 3
                  if Interrompe then
                    begin
                      Button1.Enabled := True;
                      Button1.SetFocus;
                      Interrompe := False;
                      Exit;
                    end;
                end;
  end;
  Button1.Enabled := True; // Em mais de mil anos, chegará nesta linha :-
  Button1.SetFocus; // E, imediatamente a seguir, chegará nesta
end;

```

E, para usar em caso de pânico, crie a rotina Button2Click para o evento onClick do Button2 e preencha como segue.

```
procedure TForm1.Button2Click(Sender: TObject);
begin
  Button2.Enabled := False;
  Interrompe := True;
end;
```

Está pronto, mas faltam dois importantes itens. No menu principal do Lazarus, selecione Projeto, Opções de Projeto, atribua Título = Ciclo Milenar e carregue o Ícone MNC-CicloMilenar.ico.

Agora está pronto. Salve, compile e execute. E lembre-se de deixar um aviso pedindo para não interromperem o programa nos próximos 1000 anos.



Na figura da execução do programa, Ciclos = 0,00001925% e Tempo = 13904,1712958715 s. Considerando que o ano tem 31536000 s, o tempo para chegar a 100% será aproximadamente 2000 anos<sup>6</sup>.

Confira os cálculos que seguem, pois pode ser que eu tenha feito alguma coisa errada.

$$\frac{100\%}{0,00001925\%} \cdot \frac{13904,1712958715\text{ s}}{31536000 \frac{\text{s}}{\text{a}}} = 2290,3811922011207970112079701121 \text{ a}$$

Agora que você entendeu claramente que seu computador tem limitações de tempo de execução de programas mal feitos, não abuse dele e faça programas bem feitos.

### Exemplo de erro do computador

Você já sabe tudo sobre sistemas de numeração e representação binária utilizada pelo computador. Sabe quantos bits tem em cada byte e sabe que as variáveis têm tipos diferentes e cada tipo tem uma quantidade de bytes. Ótimo. E daí?

Daí, você sabe que, por exemplo, o número 1/10000 = 0,0001 não tem representação binária exata. Efetuando a conversão de decimal para binário obtém-se uma fração repetitiva a cada 500 bits, ou seja,  $0,0001_{(10)} = 0,000000000000110100011011011000101110101100011100010000110010\dots_{(2)}$ .

O valor binário terá maior ou menor precisão dependendo do número de bytes utilizados para seu armazenamento em ponto flutuante que podem ser Single, Real, Double, Extended ou outros oferecidos pelo compilador e suportados pelo processador.

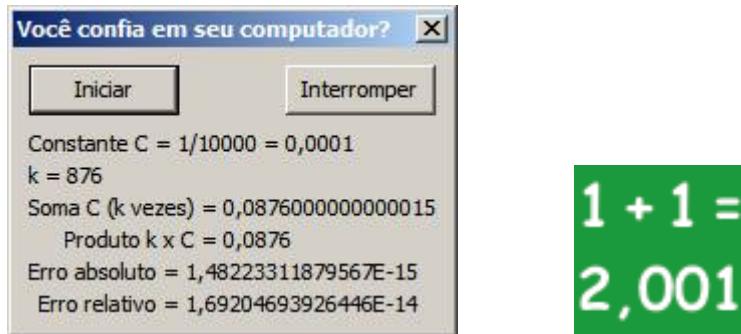
O exemplo proposto para brincar com 0,0001 é como segue.

Só um louco faz previsões sobre o futuro dos computadores.

---

<sup>6</sup> Em maquineta velha com processador Intel Core 2 Duo com 2,52 GHz e 2 GB de memória. É evidente que em sua máquina nova, muito melhor, poderá levar apenas 1000 anos ou 500 ou, quem sabe, apenas uns 200. Este texto foi escrito em 2015 e pode ser que em menos de 50 anos um programa como este demore apenas alguns anos para terminar. É só esperar.





O ícone do programa é uma piada com o fato de que errar é humano. O computador efetua  $1+1=2$ .

Crie um diretório chamado X:\meu diretório pessoal\MNC\Meu Computador Também Erra, inicie o Lazarus<sup>7</sup>, inicie um novo projeto do tipo aplicação, salve o projeto neste diretório com o nome ErroComputador.lpr e a unidade com o nome Unit1, atribua Caption = Você confia em seu computador?, BorderStyle = bsDialog, Width = 219, Height = 141 e, para Font, Name = Tahoma e Size = 8. Quase pronto.

Se não gostou do nome do diretório, troque por ...\\2-Erro do Computador

Lance, na forma, alguns componentes como segue.

Componente	Left	Top	Caption
Button1	8	8	Iniciar
Button2	136	8	Interromper
Label1	8	40	Constante C = 1/10000 = 0,0001
Label2	8	56	k =

Componente	Left	Top	Caption
Label3	8	72	Soma C (k vezes) =
Label4	26	88	Produto k x C =
Label5	8	104	Erro absoluto =
Label6	13	120	Erro relativo =

Crie a rotina para o evento Button1Click e escreva as linhas que seguem.

```
procedure TForm1.Button1Click(Sender: TObject);
var
  i, k: Integer;
  Soma: Real;
  Produto: Extended;
begin
  ControlStyle := ControlStyle + [csOpaque];
  DoubleBuffered := True;
  Pare := False;
  k := 0;
  Soma := 0;
  for i := 1 to 10000 do
begin
  k := k+1;
  Soma := Soma + 0.0001;
  Produto := k*0.0001;
  Label2.Caption := 'k = ' + IntToStr(k);
  Label3.Caption := 'Soma C (k vezes) = ' + FloatToStr(Soma);
  Label4.Caption := 'Produto k x C = ' + FloatToStr(Produto);
  Label5.Caption := 'Erro absoluto = ' + FloatToStr(Abs(Produto-Soma));
  Label6.Caption := 'Erro relativo = ' + FloatToStr(Abs((Produto-Soma)/(Produto)));
  Sleep(1);
  Application.ProcessMessages;
  if Abs(Soma-1.0) < 0.0000001 then
    Break;
  if Pare then
    Break;
end;
end;
```

DTQ 4

<sup>7</sup> Qual o motivo de usar Lazarus? Pergunta repetida. Veja nota de rodapé número 5.

Crie a rotina para o evento Button1Click e escreva apenas uma linha.

```
procedure TForm1.Button2Click(Sender: TObject);
begin
  Pare := True;
end;
```

E, claro, logo acima da rotina Button1Click defina a variável global Pare.

```
var
  Pare: Boolean;
```

Só faltam Título e Ícone. No menu principal do Lazarus, selecione Projeto, Opções de Projeto, atribua Título = Você confia em seu computador? e carregue o ícone MNC-ErroComputador.ico.

Compile, execute e aprecie o resultado. Em pouco tempo as 10000 somas são efetuadas, mas você poderá interromper quando perceber algum valor interessante, para contemplá-lo.

O programa apresentará resultados diferentes para diferentes variáveis para armazenamento de C.

Variável	bytes	Valor obtido	Valor apresentado
Single	4	1,0000535249710083008	1,000053525
Double	8	0,9999999999999061861544191743	0,999999999999906
Extended	10	0,99999999999999394473086666846	1

E essa história de Erro Absoluto e Erro Relativo que o programa apresenta? Se você leu o programa já entendeu do que se trata, mas como todo texto ortodoxo de MNC, segue uma explicação um pouco heterodoxa.

## Controlando os erros

Consideram-se dois valores,  $a$  e  $b$ , onde  $a$  é o valor correto e  $b$  é uma aproximação.

A distância absoluta entre os dois valores é medida como sendo  $|a-b|$ . Isto é o erro absoluto.

A distância relativa entre os dois valores, evidentemente relativa ao valor correto, é medida como sendo  $|a-b|/|a|$ . Isto é o erro relativo.

Em um procedimento iterativo onde não se conhece o valor correto, considera-se que a cada iteração obtém-se um valor melhor que o valor da iteração anterior. Isto ocorre se o procedimento converge, mas podem ocorrer exceções que são apresentadas ao longo do texto.

Então, se o procedimento inicia com  $x_1$  e, a partir de  $x_1$  calcula  $x_2$  e, a partir de  $x_2$  calcula  $x_3$  e, assim por diante, pode-se considerar que o último valor gerado é o melhor e será utilizado no lugar do valor correto para o cálculo do erro relativo.

Dada uma precisão ou tolerância aceitável ou erro máximo admissível  $\epsilon$ , para teste de parada em um procedimento convergente, pode-se parar quando:

- a)  $|x_{k+1}-x_k| < \epsilon$ , medindo o erro absoluto;
- b)  $|x_{k+1}-x_k| / |x_{k+1}| < \epsilon$ , medindo o erro relativo.

É claro que isto não basta, pois se o método ou o algoritmo ou o programa não garantem a convergência, há que se fazer uso de um contador de iterações que limitará o número de iterações.

Então, o procedimento pode-se parar quando:

- $|x_{k+1} - x_k| < \varepsilon$  ou *Iterações > Limite de iterações*, medindo o erro absoluto;
- $|x_{k+1} - x_k| / |x_{k+1}| < \varepsilon$  ou *Iterações > Limite de iterações*, medindo o erro relativo.

O erro absoluto mede apenas a distância. Para valores grandes de  $x_{k+1}$  e  $x_k$ , uma grande distância pode ser um erro grande ou não. O erro relativo relaciona a distância com o valor mais aproximado e, portanto, está medindo percentualmente o erro.

Medindo percentualmente fica mais fácil perceber se o erro é grande ou não. Parece, então que utilizar o erro relativo para teste de parada é melhor do que utilizar o erro absoluto.

Contudo, esta bela maneira de medir o erro relativo poderá não ser sempre adequada. Considere os casos que seguem, onde EA é o erro absoluto e ER é o Erro Relativo.

O erro relativo mostra que a relação entre os dois valores é pequena e, portanto, representa melhor o erro para um teste de parada.

$x_{k+1}$	$x_k$	EA	ER	Observações
10000000	9990000	10000	0,001	As distâncias assumem valores entre 10000 e 10, aparentemente grandes, mas de fato o erro percentual é sempre 0,001 ou 1%.
1000000	999000	1000	0,001	
100000	99900	100	0,001	
10000	9990	10	0,001	
1000	999	1	0,001	As distâncias assumem valores entre 1 e 0,001, aparentemente pequenas e, de fato o erro percentual é sempre 0,001 ou 1%.
100	99,9	0,1	0,001	
10	9,99	0,01	0,001	
1	0,999	0,001	0,001	
0,1	0,0999	0,0001	0,001	As distâncias assumem valores entre 0,0001 e 0,0000001, aparentemente muito pequenas, mas de fato o erro percentual é sempre 0,001 ou 1%.
0,01	0,00999	0,00001	0,001	
0,001	0,000999	0,000001	0,001	
0,0001	0,0000999	0,0000001	0,001	

Exagerando um pouco, considere a mesma proporção utilizada para medir o erro entre  $1 \cdot 10^{-20}$  e  $0,999 \cdot 10^{-20}$ . O erro relativo será o mesmo, ou seja, 1%. Porém, o erro absoluto envolverá cálculos com valores muito pequenos e o exemplo da seção **Exemplo de erro do computador** deixa claro que trabalhar com valores pequenos pode acarretar em grandes erros.

Outra consideração é que pode ser que um critério de parada aceite que o erro possa ser medido apenas com a distância, se o parâmetro de comparação  $\varepsilon$  for, por exemplo, da ordem de  $10^{-5}$ .

Então, um representante dessas várias situações é o erro medido da forma  $E = \frac{|x_{k+1} - x_k|}{\max\{|x_{k+1}|, 1\}}$ .

Mas, deve-se lembrar, que se os valores não forem abusivamente pequenos, ER é um bom medidor.

## 4- Cálculo de Funções por Séries de Potências

Após ver tantos erros de programação e de avaliação numérica, deve-se pensar em como o computador efetua cálculo de funções corriqueiramente utilizáveis. Pode-se utilizar um computador ou uma calculadora e solicitar o cálculo de seno(45). Antes de um piscar de olhos, a solução é apresentada e, em uma calculadora de telefone celular que faz de tudo e até pode ser usado para telefonar, a resposta será 0,85090352453411842486237967761804<sup>8</sup>. Será que o telefone tem uma tabela de senos para ver a resposta? :-)

O cálculo de funções por séries de potências permite obter expressões simples para a avaliação de funções com grau de complexidade maior. Esta é a forma utilizada pelos compiladores e calculadoras, para calcular diversas funções. Nas disciplinas de Cálculo esta teoria foi abordada e, neste texto, são apresentadas algumas dessas séries, para efeito de treinamento em programação e verificação de erros numéricos que ocorrem em suas avaliações.

Nas disciplinas de Cálculo esta teoria foi abordada e segue um breve resumo.

Uma série de potências em  $(x-x_0)$  é escrita como  $a_0+a_1.(x-x_0)+a_2.(x-x_0)^2+a_3.(x-x_0)^3+a_4.(x-x_0)^4+\dots$  que pode ser escrita como  $\sum_{n=0}^{\infty} a_n \cdot (x-x_0)^n$ . Para obter-se o valor da função em um ponto é necessário que os coeficientes  $a_n$  da série infinita sejam determinados.

### Séries de Taylor

Uma função  $f(x)$  é analítica em um ponto  $x_0$  se  $f(x)$  é a soma de uma série de potências para todo  $x$  tal que  $|x-x_0| < r$  com  $r > 0$ . Então,  $f(x) = \sum_{n=0}^{\infty} a_n \cdot (x-x_0)^n$ .

Toda função analítica em  $x_0$  também é analítica em sua vizinhança.

Uma função  $f(x)$  é analítica num ponto  $x_0$  se ela satisfizer as condições:

- 1- a função existe em  $x_0$  e vale  $f(x)$ ;
- 2- a função é contínua em  $x$ ;
- 3- a função é diferenciável em  $x_0$ , ou seja, suas derivadas  $f'(x)$ ,  $f''(x)$ ,  $f'''(x)$ , ...,  $f^n(x)$  existem no ponto.

Se  $f(x)$  é analítica para  $x_0$ , podem-se calcular os valores da função e de suas derivadas neste ponto:

$$f(x) = a_0 + a_1 \cdot (x-x_0) + a_2 \cdot (x-x_0)^2 + a_3 \cdot (x-x_0)^3 + a_4 \cdot (x-x_0)^4 + a_5 \cdot (x-x_0)^5 + \dots$$

$$f'(x) = a_1 + 2 \cdot a_2 \cdot (x-x_0) + 3 \cdot a_3 \cdot (x-x_0)^2 + 4 \cdot a_4 \cdot (x-x_0)^3 + 5 \cdot a_5 \cdot (x-x_0)^4 + \dots$$

$$f''(x) = 2 \cdot a_2 + 6 \cdot a_3 \cdot (x-x_0) + 12 \cdot a_4 \cdot (x-x_0)^2 + 20 \cdot a_5 \cdot (x-x_0)^3 + \dots$$

$$f'''(x) = 6 \cdot a_3 + 24 \cdot a_4 \cdot (x-x_0) + 60 \cdot a_5 \cdot (x-x_0)^2 + \dots$$

Para  $x=x_0$ , tem-se  $f(x_0) = a_0$ ,  $f'(x_0) = a_1$ ,  $f''(x_0) = 2 \cdot a_2$ ,  $f'''(x_0) = 6 \cdot a_3$ , que podem ser escritas como  $f(x_0) = 0! \cdot a_0$ ,  $f'(x_0) = 1! \cdot a_1$ ,  $f''(x_0) = 2! \cdot a_2$ ,  $f'''(x_0) = 3! \cdot a_3$ , ...,  $f^n(x_0) = n! \cdot a_n$ .

---

<sup>8</sup>Na sua calculadora seno(45) = 0,70710678118654752440084436210485? O que será que está acontecendo?

$$\text{Então, } a_0 = \frac{f(x_0)}{0!}, \quad a_1 = \frac{f'(x_0)}{1!}, \quad a_2 = \frac{f''(x_0)}{2!}, \quad a_3 = \frac{f'''(x_0)}{3!}, \dots, \quad a_n = \frac{f^n(x_0)}{n!}.$$

Considerando  $f(x) = a_0 + a_1 \cdot (x - x_0) + a_2 \cdot (x - x_0)^2 + a_3 \cdot (x - x_0)^3 + \dots + a_n \cdot (x - x_0)^n$  e substituindo os coeficientes conforme calculados, tem-se:  $f(x) = \frac{f(x_0)}{0!} + \frac{f'(x_0)}{1!} \cdot (x - x_0) + \frac{f''(x_0)}{2!} \cdot (x - x_0)^2 + \frac{f'''(x_0)}{3!} \cdot (x - x_0)^3 + \dots + \frac{f^n(x_0)}{n!} \cdot (x - x_0)^n$

$$\text{ou, } f(x) = \sum_{n=0}^{\infty} \frac{f^n(x_0)}{n!} \cdot (x - x_0)^n.$$

### Exemplos para algumas funções

$$1- f(x) = e^x$$

Expandindo a série  $f(x) = \sum_{n=0}^{\infty} \frac{f^n(x_0)}{n!} \cdot (x - x_0)^n$ , em torno de  $x_0 = 0$ , para a função  $f(x) = e^x$ , tem-se:

$$f(x) = e^x; \quad f'(x) = e^x; \quad f''(x) = e^x; \quad f'''(x) = e^x; \quad f^4(x) = e^x; \dots$$

Para  $x_0 = 0$  tem-se:  $f(0) = e^0 = 1; \quad f'(0) = e^0 = 1; \quad f''(0) = e^0 = 1; \quad f'''(0) = e^0 = 1; \quad f^4(0) = e^0 = 1; \dots$

Substituindo os valores na expressão geral, tem-se  $e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \dots + \frac{x^n}{n!} = \sum_{n=0}^{\infty} \frac{x^n}{n!}$ .

$$2- f(x) = \sin(x)$$

Expandindo a série  $f(x) = \sum_{n=0}^{\infty} \frac{f^n(x_0)}{n!} \cdot (x - x_0)^n$ , em torno de  $x_0 = 0$ , para a função  $f(x) = \sin(x)$ , tem-se:

$$f(x) = \sin(x); \quad f'(x) = \cos(x); \quad f''(x) = -\sin(x); \quad f'''(x) = -\cos(x); \quad f^4(x) = \sin(x).$$

Para  $x_0 = 0$  tem-se:  $f(0) = \sin(0) = 0; \quad f'(0) = \cos(0) = 1; \quad f''(0) = -\sin(0) = 0; \quad f'''(0) = -\cos(0) = -1; \quad f^4(0) = \sin(0) = 0; \dots$

Substituindo os valores na expressão geral, tem-se  $\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \dots = \sum_{n=0}^{\infty} (-1)^n \cdot \frac{x^{2n+1}}{(2n+1)!}$ .

$$3- f(x) = \cos(x)$$

Expandindo a série  $f(x) = \sum_{n=0}^{\infty} \frac{f^n(x_0)}{n!} \cdot (x - x_0)^n$ , em torno de  $x_0 = 0$ , para a função  $f(x) = \cos(x)$ , tem-se:

$$f(x) = \cos(x); \quad f'(x) = -\sin(x); \quad f''(x) = -\cos(x); \quad f'''(x) = \sin(x); \quad f^4(x) = \cos(x).$$

Para  $x_0 = 0$  tem-se:  $f(0) = \cos(0) = 1; \quad f'(0) = -\sin(0) = 0; \quad f''(0) = -\cos(0) = -1; \quad f'''(0) = \sin(0) = 0; \quad f^4(0) = \cos(0) = 1; \dots$

Substituindo os valores na expressão geral, tem-se  $\cos(x) = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \dots = \sum_{n=0}^{\infty} (-1)^n \cdot \frac{x^{2n}}{(2n)!}$ .

### Funções para testes computacionais de convergência

Seguem algumas dessas séries, para efeito de treinamento em programação e verificação de erros numéricos que ocorrem em suas avaliações.

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \dots + \frac{x^n}{n!} = \sum_{n=0}^{\infty} \frac{x^n}{n!}, \forall x$$

$$\ln(x) = \left\{ (x-1) - \frac{(x-1)^2}{2} + \frac{(x-1)^3}{3} - \frac{(x-1)^4}{4} + \dots \right\} = \sum_{n=1}^{\infty} (1)^{n-1} \cdot \frac{(x-1)^n}{n}, 0 < x \leq 2.$$

$$\ln(x) = 2 \cdot \left\{ \left( \frac{(x-1)}{(x+1)} \right) + \frac{1}{3} \left( \frac{(x-1)}{(x+1)} \right)^3 + \frac{1}{5} \left( \frac{(x-1)}{(x+1)} \right)^5 + \dots \right\} = 2 \cdot \sum_{n=0}^{\infty} \frac{1}{2n+1} \cdot \left( \frac{x-1}{x+1} \right)^{2n+1}, x > 0.$$

$$\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \dots = \sum_{n=0}^{\infty} (-1)^n \cdot \frac{x^{2n+1}}{(2n+1)!}, \forall x.$$

$$\cos(x) = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \dots = \sum_{n=0}^{\infty} (-1)^n \cdot \frac{x^{2n}}{(2n)!}, \forall x.$$

$$\operatorname{senh}(x) = x + \frac{x^3}{3!} + \frac{x^5}{5!} + \frac{x^7}{7!} + \dots = \sum_{n=0}^{\infty} \frac{x^{2n+1}}{(2n+1)!}, \infty < x < \infty.$$

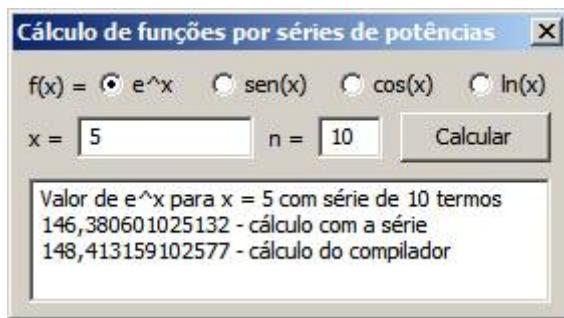
$$\cosh(x) = 1 + \frac{x^2}{2!} + \frac{x^4}{4!} + \frac{x^6}{6!} + \dots = \sum_{n=0}^{\infty} \frac{x^{2n}}{(2n)!}, \infty < x < \infty.$$

### Programa para testar a convergências de algumas destas séries

Você que já se divertiu fazendo um programa que soma mil vezes um milésimo, poderá agora fazer, ou copiar descaradamente, um programa que testa algumas destas séries.

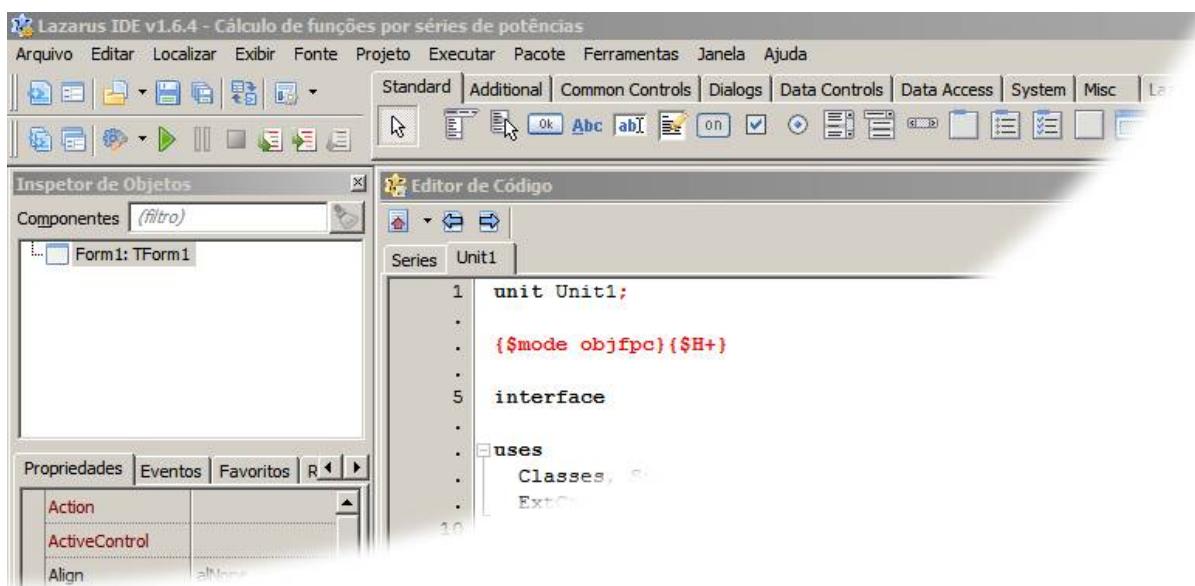
O programa que segue, testa as entradas de dados, efetua os cálculos e os apresenta. Apenas isto. Fica por sua conta efetuar todos os outros testes de exceção e, quando isto ocorrer, emitir mensagem de erro e interromper os cálculos. Se você nunca fez isto, é uma boa oportunidade para aprender. Basta verificar como isto foi feito para gerenciar as exceções nas entradas de dados.

Mãos à obra e, ao terminar, seu programa terá a cara que segue, com um belo ícone de Séries.



Crie um diretório chamado X:\meu diretório pessoal\MNC\Series ou outro qualquer que você quiser. Mas crie um diretório, caso contrário, seu computador ficará tão desorganizado, que ao final de todos os programas que desenvolver em MNC, dificilmente encontrará cada um deles.

Inicie o Lazarus, inicie um novo projeto e salve os arquivos, neste diretório, com os nomes Unit1.pas e Series.lpr. Salve antes de compilar pela primeira vez, caso contrário, o Lazarus vai criar vários diretórios para compilação que dificilmente você os encontrará. A cara do ambiente de desenvolvimento será semelhante à que segue. Se não for, pressione Ctrl F12 e carregue todos os módulos de seu programa no Editor de Código. Mova as abas dos módulos do código, para ficar ordenado logicamente.



Selecione a aba da Unit1 e, se necessário, pressione F12 para ver a fórmula Form1 e, nas propriedades, atribua Height = 131, Width = 276, Caption = Cálculo de funções por séries de potências e, para Font, Face = Tahoma e Size = 8. Está quase pronto. Salve. Se não salvar e a CPFL interromper o fornecimento de energia, como faz regularmente, além de esperar a energia ser restabelecida, você vai perder mais 10 minutos para fazer tudo que já está feito.

Lance, na forma, os componentes que seguem, com suas devidas propriedades.

Componente	Caption	Left	Top	Height	Width	ReadOnly	Text	Lines	Checked
Label1	f(x) =	8	10						
Label2	x =	8	36						
Label3	n =	128	36						
RadioButton1	e^x	42	8						True
RadioButton2	sen(x)	98	8						
RadioButton3	cos(x)	162	8						
RadioButton4	ln(x)	226	8						
Edit1		32	32		88		(remova)		
Edit2		152	32		32		(remova)		
Button1		193	30						
Memo1		8	63	62	260	True		(remova)	

Dê um toque duplo no componente Button1, para criar a rotina Button1Click que será executada quando ocorrer o evento OnClick e, logo após a cláusula Implementation, copie tudo o que segue.

### implementation

```

{$R *.lfm}
{ TForm1 }

uses
  Math; // Para utilizar a função Power

var
  x, y, z: Real;
  n: Integer;

```



```

try
  n := StrToInt(Edit2.Text)                                // - converter texto em número Inteiro
except
  ShowMessage('Valor inválido para n');
  Edit1.SetFocus;
  Exit;
end;
if RadioButton1.Checked then                               // Botão da função e^x marcado
begin
  y := CalculaExp(x, n);                                // Calcular e^x com série
  z := Exp(x);                                         // Calcular e^x do compilador
  Memo1.Lines.Add('Valor de e^x para x = ' + FloatToStr(x) + ' com série de ' + IntToStr(n) + ' termos');
  Memo1.Lines.Add(FloatToStr(y) + ' - cálculo com a série');
  Memo1.Lines.Add(FloatToStr(z) + ' - cálculo do compilador');
end;
if RadioButton2.Checked then
begin
  y := CalculaSen(x, n);
  z := Sin(x);
  Memo1.Lines.Add('Valor de sen(x) para x = ' + FloatToStr(x) + ' com série de ' + IntToStr(n) + ' termos');
  Memo1.Lines.Add(FloatToStr(y) + ' - cálculo com a série');
  Memo1.Lines.Add(FloatToStr(z) + ' - cálculo do compilador');
end;
if RadioButton3.Checked then
begin
  y := CalculaCos(x, n);
  z := Cos(x);
  Memo1.Lines.Add('Valor de cos(x) para x = ' + FloatToStr(x) + ' com série de ' + IntToStr(n) + ' termos');
  Memo1.Lines.Add(FloatToStr(y) + ' - cálculo com a série');
  Memo1.Lines.Add(FloatToStr(z) + ' - cálculo do compilador');
end;
if RadioButton4.Checked then
begin
  y := CalculaLn(x, n);
  z := Ln(x);
  Memo1.Lines.Add('Valor de ln(x) para x = ' + FloatToStr(x) + ' com série de ' + IntToStr(n) + ' termos');
  Memo1.Lines.Add(FloatToStr(y) + ' - cálculo com a série');
  Memo1.Lines.Add(FloatToStr(z) + ' - cálculo do compilador');
end;
end;
end.

```

Para terminar, no menu principal do Lazarus, selecione Projeto, Opções de Projeto, atribua Título = Cálculo de funções por séries de potências e carregue o belo ícone MNC-SériesPotências.ico que foi feito só para você.

Salve, compile, execute e teste.

Leia atentamente as rotinas, verifique os algoritmos para geração do programa, e perceba que para determinados valores de x e n, os resultados são absolutamente incorretos.

Qual o motivo disto? Pense, pense e pense. Ao concluir, informe ao seu professor, na próxima aula, para deixá-lo orgulhoso de ter um aluno como você.

E tendo entendido o motivo, você poderá melhorar o programa.

Mais alguma coisa para pensar. O factorial precisa ser calculado a cada iteração? Não poderia ser calculado na rotina de cada série, sendo aumentado do valor necessário para a parcela em questão?

E tem mais algumas coisas para você pensar sobre o cálculo das funções. Pense e inicie o debate na próxima aula.

## 5- Desenhar é preciso e desenhar funções é fundamental

Brincando de desenhar.

Como você já sabe, poderá fazer seus programas na linguagem que quiser para executar no sistema operacional que quiser.

Contudo, o amado mestre fornecerá algumas instruções em uma linguagem que é a ideal para se programar de forma limpa.

Apenas para deixar claro para quem já está viciado em coisas que não deveriam ser viciantes para criaturas tão novas como meus amados alunos, escrevo 3 vezes a mesma frase, em diferentes línguas, com traduções.

Língua	Grafi a	Tradução
Matemática	$i = 1, \dots, n$	para $i$ variando de 1 até $n$ faça ...
Pascal	for $i := 1$ to $10$ do ...	para $i$ variando de 1 até $n$ faça ...
Outras	for ( $i=1; i <=10; i++$ ) ...	$i$ vale 1, ops, $i$ é menor ou igual a 10, me enganei, $i$ é d+



Se você realmente considera que a terceira frase representa a primeira de forma mais clara do que a segunda a representa, não leia mais nada. Meus exemplos serão todos em Pascal e ... e ... e mais nada. Não há como fazer a cabeça de quem já tem a cabeça feita.

Então, vamos ao que interessa, utilizando Pascal com Delphi, Lazarus, ou outro IDE. O eleito para MNC foi Lazarus.

### Usar a força bruta ou usar componentes adequados para desenhar gráficos

Desenhar gráficos é algo útil e nem sempre fácil. A seguir são apresentadas três maneiras de se fazer desenhos. Uma maneira que serve apenas para estudo e que o desenho se perde em algumas situações<sup>9</sup>. Outra maneira onde o desenho não é volátil. A maneira de desenhar nos dois casos não é difícil, mas pode ser trabalhosa. A terceira maneira é utilizando um componente apropriado para desenhar gráficos que se chama TChart. Após apresentar as diferentes maneiras de desenhar, o exemplo apresenta como desenhar uma função em uma caixa de pintura e como fazer isto de forma muito mais fácil e eficiente com um componente para gráficos. Basta copiar, salvar, compilar, executar e, a seguir, você entenderá tudo.

Crie um diretório chamado X:\meu diretório pessoal\MNC\Brincando de Desenhar, inicie o Lazarus, inicie um novo projeto do tipo aplicação. Atribua BorderStyle=bsDialog, Caption=Brincando de Desenhar, Height=93, Width=313 e Name=FormaPrincipal. Salve o projeto neste diretório com o nome Graficos.lpr e a unidade com o nome Principal.pas. Para manter compatibilidade entre diferentes versões do Windows que utilizam diferentes fontes padrão<sup>10</sup>, em Font, atribua Name=Tahoma e Size=8. Quase pronto.

<sup>9</sup> A figura deixa de ser visível em diferentes situações para diferentes versões do Windows. No Windows XP e versões anteriores basta que outra fórmula se sobreponha à fórmula da figura ou parte da fórmula é arrastada para fora do vídeo. Nas versões posteriores ocorre quando parte da fórmula é arrastada para fora do vídeo.

<sup>10</sup> Leia sobre isto no capítulo 15.

Ao ler parágrafo anterior você entendeu que deve salvar o projeto antes de compilar? Ótimo. Assim o executável não será criado no diretório de projetos do IDE e, sim, no seu diretório pessoal.

Lance, na fôrma, alguns componentes como segue.

Name	Caption	Left	Top	Cursor
Label1	Desenhando em uma Fôrma (TForm)	8	8	crHandPoint
Label2	Desenhando em um Componente Caixa de Pintura (TPaintBox)	8	24	crHandPoint
Label3	Desenhando em um Componente Gráfico (TChart)	8	40	crHandPoint
Label4	Desenhando uma função (TPaintBox)	8	56	crHandPoint
Label5	Desenhando funções (TChart)	8	72	crHandPoint

E, agora, salve. Pressione o botão para salvar. Salve sempre. Salvar é a salvação. SALVE. Nunca mais vou repetir que você deve salvar sempre. Lembre-se que a CPFL já foi uma empresa que prestava serviços de boa qualidade. E se não quer salvar sempre, compre um nobreak.

Na aba Eventos, dê um toque duplo em OnShow, para criar a rotina FormShow e preencha como segue.

```
procedure TFormPrincipal . FormShow(Sender: TObject);
begin
  Left := 100;
  Top := 100;
end;
```

Dê um duplo toque no Label1 para criar a rotina Label1Click e preencha com o comentário **// Desenhando na Fôrma**. Repita este procedimento para os componentes Label2 até Label5, preenchendo as rotinas com os comentários **// Desenhando na Caixa de Pintura**, **// Desenhando no Gráfico (Chart)**, **// Desenhando uma função na Caixa de Pintura** e **// Desenhando funções no Gráfico (Chart)**.

Marque o Label1 e, em Eventos, dê um duplo toque em OnMouseEnter para criar a rotina Label1MouseEnter. Preencha com a linha **(Sender as TLabel)**. **Font.Color := clBlue;**. Repita a operação para o evento OnMouseLeave e preencha a rotina Label1MouseLeave com a linha **(Sender as TLabel)**. **Font.Color := clWindowText;**.

Compile e veja o efeito ao mover o mouse sobre o Label1.

Para que os outros componentes TLabel funcionem da mesma forma, NÃO repita tudo isto para todos eles. Marque os componentes Label2 até Label5 e, em Eventos, descortine os eventos já existentes para OnMouseEnter e escolha a rotina Label1MouseEnter. Em seguida descortine os eventos já existentes para OnMouseLeave e escolha a rotina Label1MouseLeave. Desta forma estas duas rotinas funcionarão para os eventos OnMouseEnter e OnMouseLeave de todos os componentes TLabel.

Compile, move o mouse sobre cada Label e veja seu comportamento.

Mesmo com bom planejamento, quando se desenvolve um programa, geralmente a criação de novas fôrmas ocorre ao longo da programação. Mas neste programa, como você está copiando algo que já foi feito, é possível criar todas as fôrmas e testar se a fôrma principal executa todas as outras corretamente. Então, crie cinco novas formas como segue, lembrando-se que a principal tem Name = FormaPrincipal e a unidade foi salva com o nome Principal.pas. Assim, as próximas fôrmas e unidades serão criadas a partir de Form1 e Unit1. Crie cada uma e atribua as seguintes propriedades.

Name	Caption	BorderStyle	Height	Width	Position
Form1	Desenhando na Fôrma	bsDialog	550	450	poDesktopCenter
Form2	Desenhando no PaintBox	bsDialog	550	450	poDesktopCenter
Form3	Desenhando no Chart	bsDialog	450	450	poDesktopCenter
Form4	Desenhando função em um TPaintBox	bsSizeable	608	856	poDesktopCenter
Form5	Desenhando funções em um TChart	bsSizeable	608	856	poDesktopCenter

Salve tudo, lembrando-se que as unidades deverão se chamar, pela ordem de criação, Unit1.pas até Unit5.pas.

Na rotina Label1Click, do FormPrincipal, logo abaixo do comentário **// Desenhando na Fórmula**, escreva a linha `Form1.Show();`. Faça o mesmo para as rotinas Label2Click até Label5Click, escrevendo `Form2.Show();` até `Form5.Show();`.

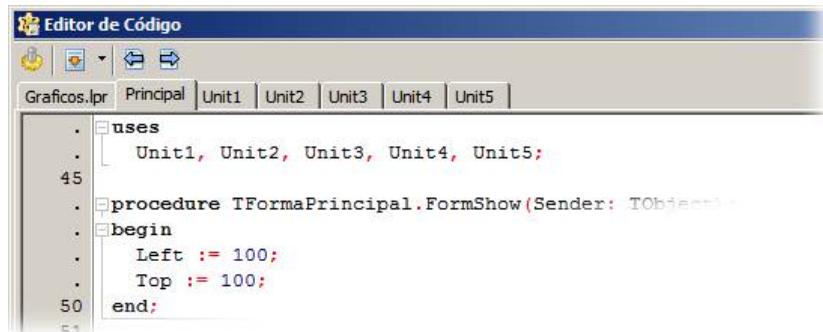
Salve e compile. O compilador reclamou? No caso do Delphi, ele resolveria isto sozinho, mas o Lazarus precisa de uma pequena ajuda. Na unidade Principal, Logo acima da rotina FormShow, introduza a cláusula uses, como segue.

```
{ TFormPrincipal }  
uses  
  Unit1, Unit2, Unit3, Unit4, Unit5;  
procedure TFormPrincipal.FormShow(Sender: TObject);  
begin
```

Salve e compile. Teste o Label1 e verifique se o Form1 é aberto. Faça isto para todos os outros. Se tudo estiver correto, só falta fazer quase tudo. Mas está quase pronto.

Se você é razoavelmente organizado, o Editor de Código deverá estar com as abas dos programas como apresentadas na figura ao lado.

Se não estiver, arraste para movê-las que ficará organizado e mais fácil de usá-las.



Alguma dificuldade para ver todas as abas? Pressione Ctrl+F12 para carregar todos os programas do projeto. Pressione F12 para comutar entre programa e fórmula.

## Desenhando em uma Fórmula

Cutuque a aba da Unit1, pressione F12 para comutar entre programa e fórmula e comece a desenhar.

No Form1 coloque um TButton (Button1) na posição (8,8) e atribua Caption=Desenhar. Dê um toque duplo para criar a rotina Button1Click.

Preencha com o trecho de programa que segue.

```

procedure TForm1.Button1Click(Sender: TObject);
var
  i: Integer;
begin
  // Desenhando na Fórmula
  // Use Alt Tab para colocar outra janela sobre esta e o desenho desaparecerá
  //Canvas.Pen.Width := 1;
  //Canvas.Pen.Color := clBlack;
  Canvas.Brush.Color := clRed;
  Canvas.Brush.Style := bsDiagCross;
  Canvas.Ellipse(50, 50, 200, 100);

  // Para não ter que repetir Canvas em todos os comandos use with Canvas do
  with Canvas do
    begin
      Pen.Width := 4;
      //Pen.Color := clBlack;
      Brush.Color := clBlue;
      Brush.Style := bsSolid;
      Rectangle(50, 150, 200, 200);

      Pen.Width := 2;
      Pen.Color := clRed;
      MoveTo(75, 140);
      LineTo(75, 110);
      LineTo(150, 110);
      LineTo(150, 210);
      LineTo(75, 210);
      LineTo(15, 290);
      for i := 1 to 60 do
        LineTo(15+4*i, 290+Round(i*Sqrt(60-i)/2)+Round(5*Sin(i)));
      Pen.Width := 4;
      Pen.Color := clYellow;
      Brush.Color := clBlue;
      Brush.Style := bsSolid;
      Rectangle(50, 250, 200, 300);

      Pen.Width := 7;
      Pen.Color := clGreen;
      Brush.Color := clBlack;
      Brush.Style := bsDiagonal;
      RoundRect(50, 350, 200, 500, 20, 20);
    end;
  end;

```



Rosa e Azul - Renoir  
Óleo sobre canvas  
MASP

Canvas é a película imaginária na qual se desenha. Funciona tal como uma tela de pintura também chamada de canvas. Pen é a pena para traçados, Brush é a textura para preenchimento e Width, Color e Style são tão óbvios que não informarei e continue lendo e acompanhando o programa que você vai entender tudo.

Nas funções de desenho, o primeiro par de valores representa o canto superior esquerdo e o segundo, o canto inferior direito. Assim, Ellipse(50, 50, 200, 100) desenha uma elipse contida em um retângulo com origem em (50,50), e fim em (200,100), ou seja, a elipse tem comprimento 150 e altura 50. MoveTo(X, Y) move a pena para as coordenadas e LineTo(X, Y) risca da coordenada anterior até a atual. Para traçar retas ou curvas com pequenos segmentos de retas utilizam-se estas duas funções.

Compile, pressione o Label1 de FormPrincipal, pressione o Botão da Unit1 e veja a série de desenhos.

Utilize Alt Tab para colocar outro programa sobre este ou minimize e restaure este programa ou arraste esta fórmula para a direita e arraste o Programa Principal sobre esta fórmula (parecerá uma borra-

cha que apagará a figura). Lembre-se da informação anterior sobre versões do Windows. No caso de versões mais recentes, para a figura desaparecer é necessário arrastar a fórmula para fora do vídeo.

Para desenhar novamente, pressione o Botão. Percebeu que as bordas de algumas figuras não são mais as originais? As bordas das duas primeiras figuras ficaram diferentes. Isto é lógico, pois a pena e o preenchimento foram alterados nas duas últimas figuras e essas informações são utilizadas para as próximas vezes que se desenha. No caso deste programa, as bordas das duas primeiras figuras que eram com Pen.Width=1 e Pen.Color=clBlack, que são os valores padrão, ficaram com Pen.Width=7 e Pen.Color=clGreen que foi a última alteração feita. Na primeira vez, funcionou como esperado, pois o padrão é Pen.Width=1 e Pen.Color=clBlack. Remova os comentários destas duas linhas no início da construção da primeira figura e tudo voltará a ser como o esperado. No caso da segunda figura você também deveria retirar os comentários de Pen.Color := clBlack, para garantir o resultado no caso de modificar o programa, mas neste programa, se não retirar continua funcionando corretamente, pois, na primeira figura foi atribuído Pen.Color=clBlack, que funcionará para a segunda.

Agora que está tudo entendido, faça o mesmo desenho sem que ele seja apagado quando a fórmula é escondida e reapresentada. Para isto deve-se utilizar o evento OnPaint. O Windows trabalha com mensagens e toda vez que um desenho deve ser feito ou refeito, por ter sido apagado ao ficar sob outra janela, o evento OnPaint é tratado e, se houver uma rotina associada a ele, ela executará.

Cutuque a forma e, em Eventos, dê um toque duplo em OnPaint para criar a rotina FormPaint. Preencha com o trecho de programa que segue. É o mesmo que o anterior, com a construção de todas as figuras dentro do laço with Canvas do e com as figuras deslocadas de 200 pontos à direita das figuras anteriores.

```
procedure TForm1.FormPaint(Sender: Tobject);
var
  i: Integer;
begin
  // Desenhando na Fórmula
  // Use Alt Tab para colocar outro programa sobre este e o desenho NÃO desaparecerá
  // As figuras são deslocadas de 200 pontos à direita.
  with Canvas do
  begin
    Pen.Width := 1;
    Pen.Color := clBlack;
    Brush.Color := clRed;
    Brush.Style := bsDiagCross;
    Ellipse(50+200, 50, 200+200, 100);

    Pen.Width := 4;
    Canvas.Pen.Color := clBlack;
    Brush.Color := clBlue;
    Brush.Style := bsSolid;
    Rectangle(50+200, 150, 200+200, 200);

    Pen.Width := 2;
    Pen.Color := clRed;
    MoveTo(75+200, 140);
    LineTo(75+200, 110);
    LineTo(150+200, 110);
    LineTo(150+200, 210);
    LineTo(75+200, 210);
    LineTo(15+200, 290);
    for i := 1 to 60 do
      LineTo(15+200+4*i, 290+Round(i*Sqrt(60-i)/2)+Round(5*Sin(i)));
    Pen.Width := 4;
```

```

Pen. Color := clYellow;
Brush. Color := clBlue;
Brush. Style := bsSolid;
Rectangle(50+200, 250, 200+200, 300);

Pen. Width := 7;
Pen. Color := clGreen;
Brush. Color := clBlack;
Brush. Style := bsBDiagonal;
RoundRect(50+200, 350, 200+200, 500, 20, 20);
end;
end;

```

Compile e teste. Arraste a forma Principal sobre esta e veja que as figuras da esquerda são apagadas e as da direita não. Pressione o botão para desenhar, novamente as figuras da esquerda. As figuras da direita são redesenhadadas toda vez que são apagadas e suficientemente rápidas para que não se perceba que estão sendo redesenhadadas.

Pronto. Você pintou na forma. Ops, no canvas da fórmula. Um perfeito Renoir. Para saber mais sobre Rosa e Azul, Renoir e MASP, veja a última página deste capítulo, onde encontrará informações úteis.

### **Desenhando em um Componente Caixa de Pintura (TPaintBox)**

Desenhar em uma fórmula ou em uma caixa de pintura é a mesma coisa. A vantagem da caixa de pintura é que não há necessidade de fazer tantos cálculos como na forma, pois as posições e medidas serão relativas à caixa e a caixa pode ser posicionada em qualquer lugar. Com estas informações é claro que você já entendeu que será tudo igual, mutatis mutandis<sup>11</sup>. Então aproveite a oportunidade para utilizar a conhecida técnica de copiar e colar, com Ctrl C e Ctrl V para os calouros ou Ctrl Ins e Shift Ins para os veteranos.

Cutuque a aba da Unit2, pressione F12 para comutar entre programa e fórmula e comece a desenhar.

No Form2 coloque um TButton (Button1) na posição (8,8) e atribua Caption=Desenhar. Dê um toque duplo para criar a rotina Button1Click e preencha com o comentário **// Desenhando no PaintBox**.

Lance um TPaintBox (PaintBox1) e atribua Left=0, Top=0, Height=550 e Width=225. Marque o PaintBox1, pressione Ctrl C e Ctrl V para criar o PaintBox2 e atribua Left=225 e Top=0.

Na rotina Button1Click que tem apenas o comentário escreva as linhas que seguem.

```

procedure TForm2.Button1Click(Sender: TObject);
var
  i: Integer;
begin
  // Desenhando no PaintBox
  { Preenchendo o fundo com branco }
  with PaintBox1.Canvas do
  begin
    Brush. Color := clWhite;
    Brush. Style := bsSolid;
    FillRect(PaintBox1.BoundsRect);
  end;

```

---

<sup>11</sup> Não sabe o que é mutatis mutandis? Então vá à luta procurando neste emaranhado que é a Web ou veja aqui: [http://en.wikipedia.org/wiki/Mutatis\\_mutandis](http://en.wikipedia.org/wiki/Mutatis_mutandis).

```

with PaintBox1 do
  with Canvas do
    begin
      Pen.Width := 1;
      Pen.Color := clBlack;
      Brush.Color := clRed;
      Brush.Style := bsDiagCross;
      Ellipse(50, 50, 200, 100);

      Pen.Width := 4;
      Canvas.Pen.Color := clBlack;
      Brush.Color := clBlue;
      Brush.Style := bsSolid;
      Rectangle(50, 150, 200, 200);

      Pen.Width := 2;
      Pen.Color := clRed;
      MoveTo(75, 140);
      LineTo(75, 110);
      LineTo(150, 110);
      LineTo(150, 210);
      LineTo(75, 210);
      LineTo(15, 290);
      for i := 1 to 60 do
        LineTo(15+4*i, 290+Round(i*Sqrt(60-i)/2)+Round(5*Sin(i)));
      Pen.Width := 4;
      Pen.Color := clYellow;
      Brush.Color := clBlue;
      Brush.Style := bsSolid;
      Rectangle(50, 250, 200, 300);

      Pen.Width := 7;
      Pen.Color := clGreen;
      Brush.Color := clBlack;
      Brush.Style := bsDiagonal;
      RoundRect(50, 350, 200, 500, 20, 20);
    end;
  end;

```

Salve, compile e teste. Funcionará como as figuras pintadas na fórmula, ou seja, desaparecerão quando sobrepostas. A diferença fica por conta a invasão do traço vermelho que não mais ocorre, como ocorre ao pintar na fórmula.

Agora que está tudo entendido, faça o mesmo desenho sem que ele seja apagado quando a fórmula é escondida e rerepresentada. Para isto deve-se utilizar o evento OnPaint, como no caso do Form1, mas agora, o evento será para o PaintBox2.

Cutique o PaintBox2 e, em Eventos, dê um toque duplo em OnPaint para criar a rotina PaintBox2Paint. Preencha com o trecho de programa que segue.

A grande diferença entre esta rotina e a rotina equivalente do Form1 é que não é mais necessário calcular a posição relativa entre as duas coleções de figuras. As posições são absolutas. O que muda é a posição do PaintBox2 com relação ao PaintBox1.

Como já inventaram os gêmeos Ctrl C e Ctrl V ao lado, segue a rotina completa. Mas você poderá simplesmente copiar a rotina anterior apenas trocando **with** PaintBox1.Canvas **do** por **with** PaintBox2.Canvas **do**.



```

procedure TForm2.PaintBox2Paint(Sender: Tobject);
var
  i: Integer;
begin
  // Desenhando no PaintBox
  { Preenchendo o fundo com azul claro }
  with PaintBox2.Canvas do
  begin
    Brush.Color := clSkyBlue;
    Brush.Style := bsSolid;
    FillRect(PaintBox1.BoundsRect);
  end;

  with PaintBox2 do
    with Canvas do
    begin
      Pen.Width := 1;
      Pen.Color := clBlack;
      Brush.Color := clRed;
      Brush.Style := bsDiagCross;
      Ellipse(50, 50, 200, 100);

      Pen.Width := 4;
      Canvas.Pen.Color := clBlack;
      Brush.Color := clBlue;
      Brush.Style := bsSolid;
      Rectangle(50, 150, 200, 200);

      Pen.Width := 2;
      Pen.Color := clRed;
      MoveTo(75, 140);
      LineTo(75, 110);
      LineTo(150, 110);
      LineTo(150, 210);
      LineTo(75, 210);
      LineTo(15, 290);
      for i := 1 to 60 do
        LineTo(15+4*i, 290+Round(i*Sqrt(60-i)/2)+Round(5*Sin(i)));

      Pen.Width := 4;
      Pen.Color := clYellow;
      Brush.Color := clBlue;
      Brush.Style := bsSolid;
      Rectangle(50, 250, 200, 300);

      Pen.Width := 7;
      Pen.Color := clGreen;
      Brush.Color := clBlack;
      Brush.Style := bsBDiagonal;
      RoundRect(50, 350, 200, 500, 20, 20);
    end;
  end;

```

### Desenhando em um Componente Gráfico (TChart)

O componente TChart possui diversas habilidades e, consequentemente, é complexo. Permite desenhar gráficos de vários tipos (linhas, barras, pizza, ...), com efeitos 2D ou 3D, com cores discretas ou super bregas, como alguns gostam, além de muito mais. Para conhecer o TChart e obter manuais do produto, utilize o endereço <http://www.steema.com/>. Neste endereço são encontrados os manuais para Delphi e outros ambientes. No caso do Lazarus, deve-se utilizar a documentação do diretório

X:\Lazarus\components\tachart. Há vários diretórios, incluindo demo e tutorials, além dos arquivos fonte do TChart. Com estas considerações, o exemplo se limita a apresentar a habilidade de desenhar pontos e curvas. Então, mãos à obra.

Cutuque a aba da Unit3, pressione F12 para comutar entre programa e fôrma e comece a desenhar.

No Form3 coloque um TButton (Button1) na posição (8,8) e atribua Caption=Desenhar. Dê um toque duplo para criar a rotina Button1Click e preencha com o comentário // Desenhando no Chart.

Lance um TChart (Chart1) e atribua Left=0, Top=40, Height=371 e Width=441.

Na rotina Button1Click que tem apenas o comentário escreva as linhas que seguem.

Cutuque o Chart1 com o botão direito do mouse e escolha Edit Series ou dê um duplo toque com o mouse, para abrir o editor de séries. Agora siga a receita e não faça perguntas. Simplesmente siga a receita. Ao terminar, você terá entendido quase tudo.

1- Em Edit Series, pressione Add. Serão desenhados pontos, mas não existe a opção Point. Escolha Line series. Será adicionado o componente Chart1LineSeries1. Cutuque-o para alterar algumas de suas propriedades. Atribua LineType = ltNone, ShowPoints = True e Title = Alguns pontos. Abra a propriedade Pointer e veja que Style = psRectangle. Quando seu programa desenhar pontos, eles serão retangulares. São úteis para marcar alguns pontos de diferentes séries de dados, mas neste programa, será dada preferência a pontos redondos. Atribua Style = psCircle. Podem-se alterar várias outras propriedades, incluindo as cores do gráfico, mas isto será feito no programa. Pode-se, também, criar uma série de pontos em Source DataPoints, mas isto também será feito no programa.

Salve, compile e execute. Nenhum ponto deverá aparecer, exceto se você preenchesse Source DataPoints com alguns valores. Isto é apenas para testar se as coisas ainda estão funcionando e se o Chart1 está visível.

2- Abra o editor de séries e, em Edit Series, pressione Add. Escolha Line series. Será adicionado o componente Chart1LineSeries2. Cutuque-o para alterar apenas uma de suas propriedades. Atribua Title = Uma reta.

Salve, compile e execute. Nenhum ponto ou segmento de reta deverá aparecer.

3- Abra o editor de séries e, em Edit Series, pressione Add. Escolha Line series. Será adicionado o componente Chart1LineSeries3. Cutuque-o para alterar apenas uma de suas propriedades. Atribua Title = Uma curva.

Salve, compile e execute. Nenhum ponto ou segmento de reta ou curva deverá aparecer.

Lembre-se que as cores para os pontos, para a reta e para a curva poderiam ser escolhidas nas propriedades de cada série, mas isto pode ser feito no próprio programa e, se as cores não forem agradáveis, pode-se alterá-las com facilidade. Então continue a fazer o programa que está quase no final.

Os títulos Alguns pontos, Uma reta e Uma curva, correspondem aos componentes Chart1LineSeries1, Chart1LineSeries2 e Chart1LineSeries3 que são utilizados para armazenar os dados que são apresentados no gráfico. Lá no início da Unit, pode-se ver que além do componente Chart1: TChart;, foram criados os componentes Chart1LineSeries1, Chart1LineSeries2, Chart1LineSeries3: TLineSeries;.

Só faltam construir as Séries 1, 2 e 3 e solicitar que o Chart1 gentilmente as apresente.

A função AddXY(**Const X, Y: Double; xlabel: String; Color: TColor**): Integer, preenche os dados a serem apresentados, permitem um rótulo para a série (que pode ser vazio) e atribui uma cor (que é opcional).

Portanto, como exemplo, para Chart1LineSeries1 (série dos pontos), podem-se escolher diferentes formas para preenchimento dos dados, com diferentes resultados. Seguem alguns exemplos.

```
// Desenha 10 pontos aleatórios entre (0,0) e (20,40), com a cor clBlack
for i := 1 to 10 do
    Chart1LineSeries1.AddXY(Random(20), Random(40), '', clBlack);

// Desenha 10 pontos aleatórios entre (0,0) e (20,40), com a cor definida no Editor do TChart
// ou com a cor definida em linha anterior do programa
for i := 1 to 10 do
    Chart1LineSeries1.AddXY(Random(20), Random(40), '');

// Desenha 10 pontos aleatórios entre (0,0) e (20,40), com a cor clRed
// com o ridículo título PT, que significa Ponto Tolo
for i := 1 to 10 do
    Chart1LineSeries1.AddXY(Random(20), Random(40), 'PT', clRed);
```

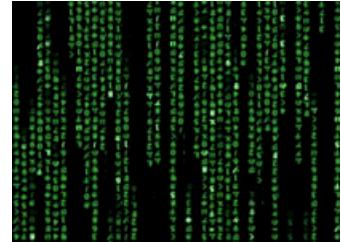
Na rotina Button1Click que tem apenas o comentário escreva as linhas que seguem.

```
procedure TForm3.Button1Click(Sender: TObject);
begin
    // Desenhando no Chart
    // Salpicando um punhado de pontos
    with Chart1LineSeries1 do
        for i := 1 to 10 do
            AddXY(Random(20), Random(40), '', clRed);
end;
```

Salve, compile e teste. Pressione o botão e veja os 10 pontos aleatórios. Pressione novamente. Mais 10 pontos foram lançados. Pressione várias vezes e verá dezenas de pontos vermelhos.



A primeira coisa que você, aluno esperto deve ter notado, é que se faz necessário limpar o Chart1 em algumas situações. Neste caso, pode ser quando se pressiona o botão. Em certas situações é conveniente quando se apresenta ou quando se fecha a fórmula, utilizando os eventos OnShow ou OnClose. Então, logo após o comentário **// Desenhando no Chart**, escreva `Chart1LineSeries1.Clear;`. Você já pode aproveitar e escrever `Chart1LineSeries2.Clear;` e `Chart1LineSeries3.Clear;`, também.



Agora, teste as variações, compilando e executando cada uma:

- `AddXY(Random(20), Random(40), '', clBlack);`
- `AddXY(Random(20), Random(40), '');`
- `AddXY(Random(20), Random(40));`

Na primeira variação os pontos serão totalmente pretos. Lembre-se que quando eram vermelhos, a borda era preta. Isto porque a propriedade Pointer Pen é `clBlack`, por padrão.

Na segunda variação os pontos são brancos com borda preta. Isto porque a propriedade Pointer Brush é `clWhite`, por padrão.

Na última variação, a apresentação é idêntica à anterior, ou seja, a o parâmetro Label pode ser omitido. Contudo, mesmo que não haja valor para Label, deve-se utilizar `''`, se houver o parâmetro Color.

Dê um toque no componente Chart1, abra a propriedades Legend e veja que `Visible = False`. Altere para `Visible = True`. Agora você tem a legenda dos gráficos e todos os símbolos são pretos.

Volte à situação inicial AddXY(Random(20), Random(40), '', clRed);, compile e execute. os pontos do gráfico são vermelhos e as figuras da legenda continuam pretas. Durante a criação das Séries foi sugerido que não se atribuissem cores para nada que isto seria feito de forma mais fácil no programa. Troque o trecho do programa que desenha os pontos pelo que segue.

```
// Salpicando um punhado de pontos
Chart1LIneSeries1.Pointer.Brush.Color := clRed;
with Chart1LIneSeries1 do
  for i := 1 to 10 do
    AddXY(Random(20), Random(40));
```

Compile, execute e veja que com Pointer Brush Color = clRed, não é mais necessário escrever a cor na função AddXY.

E agora, sem enrolação, modifique sua rotina Button1Click, para ficar como segue.

```
procedure TForm3.Button1Click(Sender: TObject);
var
  X, Y: array[1..101] of Real;
  Xmin, Xmax, DeltaX: Real;
  i, n: Integer;
begin
  // Desenhando no Chart
  Chart1LIneSeries1.Clear;
  Chart1LIneSeries2.Clear;
  Chart1LIneSeries3.Clear;
  // Salpicando um punhado de pontos
  Chart1LIneSeries1.Pointer.Brush.Color := clRed;
  with Chart1LIneSeries1 do
    for i := 1 to 10 do
      AddXY(Random(20), Random(40));
  // Desenhando 3 segmentos de reta
  with Chart1LIneSeries2 do
  begin
    SeriesColor := clGreen;
    AddXY(0, 10);
    AddXY(10, 20);
    AddXY(15, 5);
    AddXY(20, 40);
  end;
  // Desenhando uma curva
  // Os dados podem ser carregados diretamente com a função AddXY
  // Apenas para demonstração, cria-se a coleção de dados e depois se carrega a série
  Xmin := 0;
  Xmax := 20;
  n := 100;
  DeltaX := (Xmax-Xmin)/n;
  for i := 1 to n+1 do
  begin
    X[i] := Xmin+(i-1)*DeltaX;
    Y[i] := Sqr(X[i])/10 * Sin(X[i]);
  end;
  // Depois que tudo está calculado, carrega-se a série
  Chart1LIneSeries3.SeriesColor := clBlue;
  with Chart1LIneSeries3 do
    for i := 1 to n+1 do
      AddXY(X[i], Y[i]);
end;
```

Faça um último teste. Coloque as 3 últimas linhas da rotina como comentário, compile e execute. O gráfico deverá ter, no eixo das ordenadas, valores de 0 até 40 ou de 5 até 40, dependendo de existir ou não pontos entre 0 e 5, pois os segmentos de reta ocupam a região de 5 até 40.

Remova os comentários, compile e execute. O gráfico deverá ter, no eixo das ordenadas, valores de -30 até 40.

Menino esperto este TChart, não é mesmo? Ele gradua o eixo de forma a utilizar toda a área de gráfico para desenhar as figuras, sem deixar áreas vazias. Caso seja necessário usar escala não variável, devem-se utilizar as propriedades adequadas para tal empreitada que ficará a seu cargo.

Considere que o último teste foi o penúltimo e faça o último. Coloque comentário em `// Chart1LineSeries2.Clear;` e `// Chart1LineSeries3.Clear;`. Deixe `Chart1LineSeries1.Clear;` sem comentário para não encher o gráfico com bolotas vermelhas.

Compile, execute e pressione o botão Desenhar e olhe bem para a cara do gráfico. Pressione outra vez e veja a diferença. O que você está vendo? Um absurdo, não é? Para não ver mais este absurdo, não se esqueça de limpar o gráfico em situações específicas como sugeridas no início desta seção. Aproveite e remova os comentários.

Só não gostei de uma coisa que você se esqueceu de fazer. Não tem título neste gráfico. Cutuque o Chart1, abra a propriedade Title e mude Visible de False para True. Agora ficou pior, escrito TAChart, com cor azul. Arrume isto. Abra a propriedade Font da Propriedade Title e mude Color de clBlue para clBlack. Na propriedade Title Text, escreva meigamente, Meu primeiro gráfico com o TChart. Agora ficou coisa de profissional, mesmo.

### **Desenhando uma função em um TPaintBox**

Cutuque a aba da Unit4, pressione F12 para comutar entre programa e fôrma e comece a desenhar.

Lance um TPaintBox (PaintBox1) na posição (8,8) e estique o máximo que puder, com múltiplos de 8 pontos. A caixa de pintura deverá ficar com Height=592 e Width=840.

Agora só falta desenhar e redesenhar quando a fôrma alterar de tamanho. Note que esta fôrma permite Maximizar, Restaurar e Redimensionar. Portanto, a cada vez que o tamanho da fôrma modificar, é necessário redesenhar. Você já sabe utilizar o evento OnPaint para desenhar de forma não volátil. Agora verá como redesenhar quando a fôrma tiver seu tamanho modificado.

Ao invés de longas explicações antes de apresentar cada rotina, este texto apresenta as rotinas, com comentários, que são suficientes. Quem leu atentamente o texto das seções anteriores deverá entender.

O exemplo é de uma função seno. Há uma rotina para calcular todos os pontos para que se possa desenhar a função e as outras para desenhar ou redesenhar.

No início da Unit4 você verá a estrutura do Form4, como segue.

```
type
  { TForm4 }
TForm4 = class(TForm)
  Pai ntBox1: TPai ntBox;
private
  { private declarations }
public
  { public declarations }
end;
```

Acrescente uma nova variável e uma nova rotina, como seguem.

```

private
  { private declarations }
PontosFuncao: array [0..144] of TPoint;
procedure CalculaGrafico;

```

No início da seção **implementation**, escreva a rotina CalculaGrafico, como segue.

```

{ TForm4 }
procedure TForm4.CalculaGrafico;
var
  FaixaPontosX, FaixaPontosY: Integer;
  Origem: TPoint;
  Radios, Intervalo: Double;
  i: Integer;
begin
  {
    O seno é calculado no intervalo entre -2Pi..+2Pi, com resolução
    de 4Pi/144 como uma série de pontos que são conectados por uma polyline.
    A escala do gráfico é calculada para caber e preencher o PaintBox.
    A origem do sistema de coordenadas é o centro do PaintBox.
    O sistema de coordenadas padrão é invertido, pois o eixo Y,
    na tela, é medido de cima para baixo.
  }
  FaixaPontosX := (PaintBox1.Width - 2) div 4; { pontos em Pi unidades em x }
  FaixaPontosY := (PaintBox1.Height - 2) div 2; { pontos em 1 unidade em y }
  Origem := Point(PaintBox1.Width div 2, PaintBox1.Height div 2);
  Radios := -2.0 * Pi;
  Intervalo := 4.0 * Pi / 144.0;
  for i := 0 to High(PontosFuncao) do
  begin
    PontosFuncao[i].X := Origem.X + Round(Radios * FaixaPontosX / Pi);
    PontosFuncao[i].Y := Origem.Y - Round(Sin(Radios) * FaixaPontosY);
    Radios := Radios + Intervalo;
  end;
end;

```

Com os pontos calculados, pode-se desenhar a função.

No Form4, dê um toque no PaintBox e, em Eventos, um duplo toque em OnPaint para criar a rotina PaintBox1Paint e preencha com o trecho de programa que segue.

```

procedure TForm4.PaintBox1Paint(Sender: TObject);
var
  Origem: TPoint;
  FaixaPontosX, FaixaPontosY: Integer;
begin
  with PaintBox1.Canvas do
  begin
    { Desenha traços das coordenadas }
    Origem := Point(PaintBox1.Width div 2, PaintBox1.Height div 2);
    Pen.Color := clBlack;
    Pen.Style := psSolid;
    Pen.Width := 1;
    MoveTo(1, Origem.Y);
    LineTo(PaintBox1.Width - 1, Origem.Y);
    MoveTo(Origem.X, 1);
    LineTo(Origem.X, PaintBox1.Height - 1);

    { Desenha algumas marcas e rótulos para os eixos }
    Font.Name := 'Symbol';
    Font.Size := 8;
    Font.Color := clBlack;
    FaixaPontosX := (PaintBox1.Width - 2) div 4; { pontos em Pi unidades em x }
    FaixaPontosY := (PaintBox1.Height - 2) div 2; { pontos em 1 unidade em y }

    { Eixo X }
    MoveTo(Origem.X - 2 * FaixaPontosX, Origem.Y - 4);
  
```

```

LineTo(Origem.x - 2 * Fai xaPontosX, Origem.y + 4);
TextOut(Origem.x - 2 * Fai xaPontosX + 2, Origem.y + 2, '-2p');
MoveTo(Origem.x - Fai xaPontosX, Origem.y - 4);
LineTo(Origem.x - Fai xaPontosX, Origem.y + 4);
TextOut(Origem.x - Fai xaPontosX + 2, Origem.y + 2, '-p');
MoveTo(Origem.x + Fai xaPontosX, Origem.y - 4);
LineTo(Origem.x + Fai xaPontosX, Origem.y + 4);
TextOut(Origem.x + Fai xaPontosX - 2 - TextWidth('p'), Origem.y + 2, 'p');
MoveTo(Origem.x + 2 * Fai xaPontosX, Origem.y - 4);
LineTo(Origem.x + 2 * Fai xaPontosX, Origem.y + 4);
TextOut(Origem.x + 2 * Fai xaPontosX - 2 - TextWidth('2p'), Origem.y + 2, '2p');

{ Eixo Y }
MoveTo(Origem.x - 4, Origem.y - Fai xaPontosY);
LineTo(Origem.x + 4, Origem.y - Fai xaPontosY);
TextOut(Origem.x + 4, Origem.y - Fai xaPontosY, '1.0');
MoveTo(Origem.x - 4, Origem.y - Fai xaPontosY div 2);
LineTo(Origem.x + 4, Origem.y - Fai xaPontosY div 2);
TextOut(Origem.x + 4, Origem.y - (Fai xaPontosY + TextHeight('1')) div 2, '0.5');
MoveTo(Origem.x - 2, Origem.y + Fai xaPontosY div 2);
LineTo(Origem.x + 2, Origem.y + Fai xaPontosY div 2);
TextOut(Origem.x + 3, Origem.y + (Fai xaPontosY - TextHeight('1')) div 2, '-0.5');
MoveTo(Origem.x - 2, Origem.y + Fai xaPontosY);
LineTo(Origem.x + 2, Origem.y + Fai xaPontosY);
TextOut(Origem.x + 3, Origem.y + Fai xaPontosY - TextHeight('1'), '-1.0');

{ Desenha o Gráfico }
Pen.Color := clBlue;
Polyline(PontosFuncao);
end;
end;

```

Salve, compile e teste. Só viu os eixos? É lógico. A rotina PaintBox1Paint é disparada pelo evento OnPaint do PaintBox1 e desenha eixos, marcas, rótulos e executa Polyline que não recebeu os dados dos pontos da função.

Dê um toque na fôrma e, em Eventos, dê um duplo toque em OnCreate e em OnResize para criar as rotinas FormCreate e FormResize. Digite, na rotina FormCreate, a linha `CalculaGrafico;` e digite, na rotina FormResize, a linha `CalculaGrafico;`. Desta forma, os pontos serão calculados quando a fôrma for criada e quando for redimensionada.

Salve, compile e teste. Maximize e restaure a fôrma. Pense sobre o que você viu. O que é que está faltando? No Form4, marque o PaintBox1 e, em Propriedades, atribua para Anchors, `akRight=True` e `akBottom=True`. Agora, o PaintBox1 está ancorado à fôrma, em seu canto superior esquerdo e, também, em seu canto inferior direito. Portanto, será redimensionado proporcionalmente à fôrma.

Salve, compile e teste. Maximize e restaure a fôrma. Nem pense sobre o que você viu. Ficou lindo. Se pensar estraga.

Informações importante, para quem ainda não percebeu (ou seja, só para os nós-cegos).

- 1- CalculaGrafico simplesmente calcula os pontos necessários.
- 2- FormCreate executa CalculaGrafico quando a fôrma é criada e, quando ela é apresentada pela primeira vez, PaintBox1Paint executa e apresenta o desenho de forma adequada.
- 3- FormResize executa CalculaGrafico quando a fôrma é redimensionada e, quando ela é redesenhada, PaintBox1Paint executa e apresenta o desenho de forma adequada.

Tudo entendido? Já que sim, faça os seguintes testes.

- 1- Coloque em comentário apenas o comando `//CalculaGrafico;` da rotina FormResize. Execute o programa, maximize e restaure. Você verá que os pontos não foram recalculados para a nova área de desenho e, portanto, a função foi apresentada de forma incorreta.

- 2- Coloque em comentário apenas o comando `//CalculaGrafico;` da rotina FormCreate. Execute o programa. Você verá que os pontos foram calculados e, portanto, a função foi apresentada de forma correta.

Então não é necessário calcular os pontos antes de apresentar pela primeira vez? Claro que precisa. E onde eles foram calculados? Onde está a mágica? Há mais coisas na relação entre os eventos do Windows do que pode imaginar sua vã filosofia. Veja o Help do Lazarus ou do Delphi ou de qualquer IDE para Windows e leia sobre a sequência dos eventos. Quando uma fórmula é criada e apresentada os eventos ocorrem na ordem OnCreate, OnShow, OnActivate e OnPaint. Portanto, OnPaint sempre ocorrerá na primeira apresentação da fórmula e não é necessário calcular os pontos em OnCreate ou em OnShow. Então não é necessário calcular os pontos em FormCreate, mas eles são calculados antes de apresentar pela primeira vez na rotina chamada pelo evento OnPaint, que é PaintBox1Paint.

Tudo entendido? Já que sim, fazer o mesmo com o TChart é um pouquinho mais fácil.

### **Desenhando funções em um TChart**

Cutuque a aba da Unit5, pressione F12 para comutar entre programa e fórmula e comece a desenhar.

Lance um TChar (Chart1) na posição (8,8) e estique o máximo que puder, com múltiplos de 8 pontos. O gráfico deverá ficar com Height=592 e Width=840. Atribua, para Anchors, akRight=True e akBottom=True, como no PaintBox1 do caso anterior.

Agora só falta desenhar e redesenhar quando a forma alterar de tamanho.

Não minha santa. Nada disto. Preste atenção. Agora você está utilizando o TChart. Basta definir as séries que serão utilizadas, calcular os pontos da série e solicitar que o TChart apresente o resultado.

Cutuque o Chart1 com o botão direito do mouse e escolha Edit Series ou dê um duplo toque com o mouse, para abrir o editor de séries. Agora, siga a receita mantendo o editor Edit Series aberto.

- 1- Pressione Add. Escolha Line series. Será adicionado o componente Chart1LineSeries1. Cutuque-o para alterar algumas de suas propriedades. Atribua SeriesColor = clRed e Title = Função seno.
- 2- Pressione Add. Escolha Line series. Será adicionado o componente Chart1LineSeries2. Cutuque-o para alterar algumas de suas propriedades. Atribua SeriesColor = clGreen e Title = Função coseno. Para quem não está acostumado com as convenções de cores, saiba que clGreen é verde escuro discreto RGB(0,128,0) e clLime é verde claro cheguei RGB(0,255,0).
- 3- Pressione Add. Escolha Line series. Será adicionado o componente Chart1LineSeries3. Cutuque-o para alterar algumas de suas propriedades. Atribua SeriesColor = clBlue e Title = Função doidinha.

Acredite se quiser, está quase pronto. Cara, você está virando um programador supimpa<sup>12</sup>.

Feche o editor, dê um toque no componente Chart1, abra a propriedade Legend e atribua Visible = True. Agora você tem a legenda dos gráficos e os símbolos são coloridos, pois as cores foram atribuídas nas propriedades das Séries.

Abra a propriedade Title e atribua Visible = True. Em Text, abra o editor e escreva Minhas funções:, cutuque a tecla Enter e escreva Seno, Cosseno e Doidinha. Pressione Ok para salvar e fechar o editor. Abra a propriedade Font e atribua Color = clBlack.

---

<sup>12</sup>Se não sabe o que é supimpa, não deve ser supimpa. Então veja: <http://priberam.pt/dlpo/supimpa>. Espero que na próxima aula, todos já tenham adquirido o estado de programador supimpa.

Cutuque o Form5. Se estiver difícil, cutuque o Chart1 e a tecla Esc. A tecla Esc é uma tecla mágica. Agora o controle está no Form5. Vivendo e aprendendo.

Em Eventos, dê um toque duplo em OnShow, para criar a rotina FormShow.

Preencha com o trecho de programa que segue.

```
procedure TForm5. FormShow(Sender: TObj ect);
var
  x, y: array[0..201] of Real;
  Del taX: Real;
  i: Integer;
begin
  // Limpando para não ligar o último ao primeiro ponto na próxima vez que executar esta fórmula
  Chart1Li neSeri es1. Clear;
  Chart1Li neSeri es2. Clear;
  Chart1Li neSeri es3. Clear;
  // Definindo o passo
  Del taX := 4*Pi /200;
  // Criando a função seno entre -2Pi e 2Pi, com 200 intervalos (201 pontos)
  for i := 1 to 201 do
  begin
    x[i] := -2*Pi + (i-1)*Del taX;
    y[i] := Sin(x[i]);
    Chart1Li neSeri es1. AddXY(x[i], y[i]);
  end;
end;
```

Você sabe que poderia fazer como segue, mas para efeito didático foi feito como foi feito.

```
for i := 1 to 201 do
  Chart1Li neSeri es1. AddXY(-2*Pi + (i-1)*Del taX, Sin(-2*Pi + (i-1)*Del taX));
```

Compile, execute e babe. Depois que você limpar a baba, continue.

```
// Criando a função seno entre -2Pi e 2Pi, com 200 intervalos (201 pontos)
for i := 1 to 201 do
  ...
end;
// Criando a função cosseno entre -2Pi e 2Pi, com 200 intervalos (201 pontos)
for i := 1 to 201 do
begin
  x[i] := -2*Pi + (i-1)*Del taX;
  y[i] := Cos(x[i]);
  Chart1Li neSeri es2. AddXY(x[i], y[i]);
end;
```

Compile, execute e babe, novamente. Já pensou o trabalho que daria fazer isto na força bruta, utilizando o TPaitBox?. Depois que você limpar a baba, continue.

Em certas situações é conveniente destacar uma ou mais funções. Por exemplo, pode-se desenhar a função seno com linha mais grossa que a função cosseno. Isto pode ser feito de duas formas. A primeira é dar um toque duplo em Chart1 para abrir o editor de séries, selecionar Chart1LineSeries1, abrir a propriedade LinePen e atribuir Width =2. Outra forma é escrever, no programa, a linha Chart1Li neSeri es1. Li nePen. Wi dth := 2; abaixo da linha **// Criando a função seno entre -2Pi ....**

E, para terminar, desenha-se a função doidinha. Mas como criar a tal da função doidinha?

Vou criar uma, mas fique à vontade para criar a sua. Vou usar a mesma de exemplo anterior (Form3), com uma leve modificação. No exemplo anterior, a variação de x é entre 0 e 20 e, neste, entre -6 e 6. Para que a função doidinha fique com o mesmo aspecto neste gráfico e no anterior, será feita a transformação  $(x+6)^*20/12$ .

```

// Criando a função cosseno entre -2Pi e 2Pi , com 200 intervalos (201 pontos)
for i := 1 to 201 do
  ...
end;
// Criando a função doidinha entre -6 e 6, com 200 intervalos (201 pontos)
Del taX := 12/200;
for i := 1 to 201 do
begin
  x[i] := -6 + (i-1)*Del taX;
  y[i] := Sqr((x[i]+6)*20/12)/10 * Sin((x[i]+6)*20/12);
  Chart1LinetSeries3.AddXY(x[i], y[i]/36);
end;

```

Compile, execute e babe, mais uma vez. Já pensou o trabalho que daria fazer isto na força bruta, utilizando o TPaitBox?. Depois que você limpar a baba, mais uma vez, continue. Ou pare, pois chegou ao fim.

Veja, no trecho de programa, que não foi necessário utilizar a função Round, como em exemplo anterior (Form3), para obter valores inteiros. O TChart aceita valores reais e trata estes valores ao desenhar.

Note que ao carregar a série 3, os valores de y foram divididos por 36. Remova esta divisão e veja que a função doidinha assume valores aproximadamente entre -30 e 36 no eixo das ordenadas, reduzindo muito a visão das funções seno e cosseno. Portanto, uma forma de padronizar as funções é dividir esta função por 36.

Finalmente, vá até o menu principal do Lazarus e selecione Projeto, Opções de Projeto, preencha Título com Brincando de Desenhar, pressione o botão Carregar ícone e carregue o ícone MNC-Gráficos.ico, gentilmente cedido pelo seu amado professor.

Pronto. Você fez um belo programa. Acho que você ficou orgulhoso por fazer *sozinho* este seu programa, mesmo copiando tudo. Eu também estou. Afinal, tenho certeza que ainda vou ganhar aquele notebook de presente de aniversário, que foi em 9 de abril.

## Sobre Rosa e Azul, Renoir e MASP



O Museu de Arte de São Paulo possui um acervo de excelente qualidade e deve ser visitado por qualquer pessoa que tenha um mínimo de cultura e bom gosto. E eu acho que é o seu caso, afinal, alunos de Computação estão até um pouco acima deste mínimo. :-)

Pierre-Auguste Renoir (1841-1919) foi um pintor francês impressionista e, para conhecer um pouco sobre sua vida e obras, você deveria dar uma espiada em [http://pt.wikipedia.org/wiki/Pierre-Auguste\\_Renoir](http://pt.wikipedia.org/wiki/Pierre-Auguste_Renoir).

Rosa e Azul, As Meninas Cahen d'Anvers, é uma obra importante de Renoir e encontra-se desde 1952 no MASP. Leia mais em:

- [http://pt.wikipedia.org/wiki/Rosa\\_e\\_Azul\\_\(Renoir\)](http://pt.wikipedia.org/wiki/Rosa_e_Azul_(Renoir))
- [http://www.maspa.art.br/masp2010/acervo\\_detalheobra.php?id=272](http://www.maspa.art.br/masp2010/acervo_detalheobra.php?id=272)

E apenas para relembrar, programar é sempre a mesma coisa, mutatis mutandis.



Será que Renoir colou na prova?

## P&R

P - Gostei da fórmula 5 com os desenhos do seno, do cosseno e da tal da doidinha. 😊

Mas para que serve este programa?

Joãozinho - 2 dias, 4 horas e 18 minutos

R - Que bom que você gostou.

Serve apenas para desenhar as funções seno, cosseno e doidinha.

P - Mas para que serve este programa? 😕

Joãozinho - 2 dias, 4 horas e 16 minutos

R - Serve apenas para desenhar as funções seno, cosseno e doidinha.

P - E se eu quiser desenhar a função  $x^2+3*x$ ? 😕

Joãozinho - 2 dias, 4 horas e 14 minutos

R - Agora a pergunta foi boa.

Edite o programa, altere tudo que tem que ser alterado, compile e execute.

P - Mas isto é um absurdo. 😬

Joãozinho - 2 dias, 4 horas e 12 minutos

R - Concordo com você e, para resolver este problema, é necessário um interpretador de funções.

P - E onde eu encontro um interpretador de funções? 😕

Joãozinho - 2 dias, 4 horas e 10 minutos

R - Muito simples. Veja o próximo capítulo e assista a próxima aula.

P - E como eu utilizo um interpretador de funções? 😕

Joãozinho - 2 dias, 4 horas e 8 minutos

R - Muito simples. Veja a resposta anterior.

P - Fessôr. O Renoá colou mesmo do Maurício de Souza? 🦁

Patricinha - 2 dias, 4 horas e 6 minutos

R - Por motivos técnicos o P&R foi encerrado antes que fosse possível responder a última pergunta.

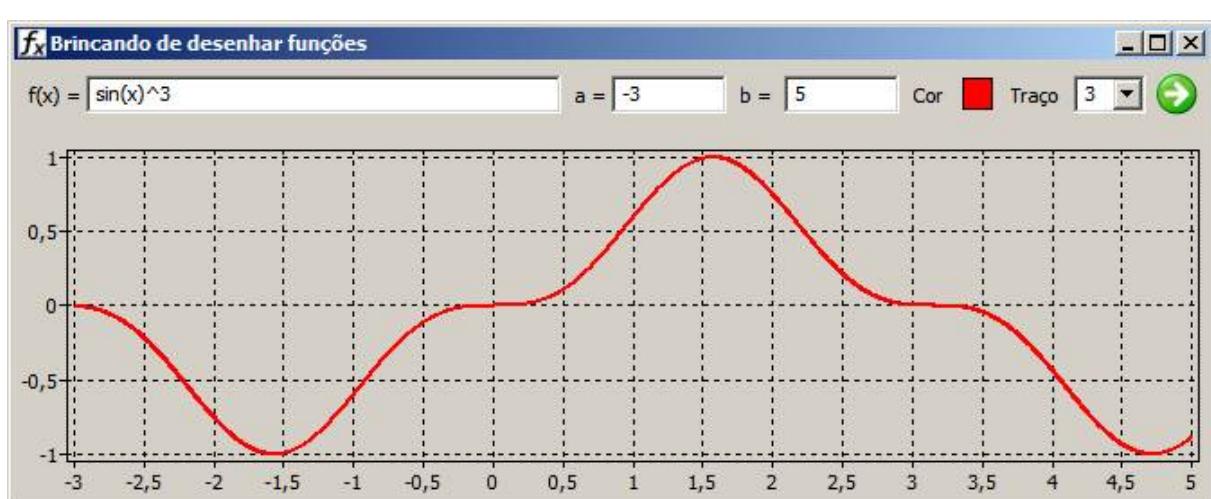
P&R encerrado

Todo mundo já entendeu que não se edita programa para trocar funções de usuário.

## Sugestão da Patricinha

O P&R encerrou, mas a matraca da Patricinha não para de falar.

Se o P&R não estivesse encerrado, eu perguntaria se é possível fazer, com o interpretador, um programa onde o usuário entra com a função, com o intervalo, com a cor e com a grossura do traço. Mas como está encerrado, vou só ficar sonhando com o programa. 🧑‍🦰 Não sou loira, não.



## 6- Avaliando funções e, como bônus, um avaliador de funções

Todo método numérico utiliza funções e, então, avaliar funções é absolutamente necessário.

Lembra-se do que eu disse na primeira aula? Eu minto e espero que você, aluno, esteja acordado, perceba a mentira e discuta sobre o assunto.

Nem todo método numérico utiliza funções, mas avaliar funções é absolutamente necessário.

É claro que o texto se refere às funções dadas explicitamente na forma  $f(x)$  onde se queria avaliar o valor de  $y=f(x)$ , para determinado valor de  $x$ . Nem sempre isto ocorre. Mas quando ocorre, alguns programadores fazem programa que merecem ir ao lixo, ops, de forma menos grosseira, ao recycle bin. É isto mesmo, lixeira. E, em seguida, esvazia-se a lixeira, pois não é para reciclar.



E qual o motivo de toda esta coleção de lixeiras? Muito simples. Imagine um programa onde se define uma função  $f$ . O programa é compilado e executado. Tudo funciona perfeitamente. A seguir, deseja-se alterar a função. E aí acontece o previsível. Edita-se o programa fonte, para alterar a função, compila-se novamente e executa-se o dito cujo. Super profissional.

Imagine se para cada novo nível daquele joguinho que você vive jogando, fosse necessário compilar novamente, após trocar todo o cenário e personagens do jogo. Super profissional. Quero um deste.

Então, é necessário ter um avaliador de funções que calcule o valor de uma função em um determinado ponto, onde a função e o ponto sejam dados através de entrada de dados em formato de texto, sem a necessidade de editar o código fonte e recompilar. Agora a coisa ficou com cara profissional.

Você poderá fazer o seu interpretador ou utilizar um pronto que pode ser obtido na Web. Basta procurar. Mas para você, há um interpretador já testado, que avalia funções com uma ou várias variáveis e que é distribuído em forma de biblioteca (dll). O endereço para obter esta biblioteca é aquele informado em aula e que também está lá no começo deste texto.

Seguem as instruções para sua utilização.

## O interpretador de funções

Como descrito em página anterior, este texto contém **exemplos** escritos em Pascal e compilados com Lazarus.

O **interpretador** poderia ser feito em C, C+, C++, C+++, C# ou C##, mas foi feito em Pascal. A biblioteca (dll) poderia ser feita de forma a ser fácil de utilizar com programa feitos em Pascal e, também, em C. Mas foi feita para ser facilmente utilizada em Pascal. Pura maldade, para você aprender a programar de forma clara, acompanhando o algoritmo de forma simples e, depois que você crescer, faça seu interpretador e utilize outras linguagens.

A versão da dll oferecida ao amado leitor é específica para **Windows 32 bits**.



### Breve descrição das rotinas da biblioteca Interpretador.dll

O interpretador tem uma quantidade de rotinas para uso interno e outras que são exportadas para serem utilizadas externamente. As rotinas e outros parâmetros para utilização externa são descritos a seguir.

A documentação completa encontra-se em Interpretador.chm que pode ser obtida no pacote l6-Interpretador Exemplo Win.rar

Tipo	Representando
Vetor	Vetor de Extended com tamanho variável

Constantes	Representando
pi	3,1415926535897932384626433832795
e	2,7182818284590452353602874713527

Operadores	+	,	-	*	/	^	**	!
Representando	Sinal	Parênteses	Soma	Subtração	Multiplicação	Divisão	Potência	Fatorial

Funções	Representando
Trigonométricas	
cos	cosseno
sin	seno
tan, tg	tangente
cotan, cot, cotg, ctg	cotangente
secant, sec	secante
cosecant, cosec, csc	cossecante
arccos	arcosseno
arcsin	arcseno
arctan, arctg	arctangente
Hiperbólicas	
cosh	cosseno
sinh	seno
tanh, tgh	tangente
arccosh, arcosh	arcosseno
arsinh, arcsinh	arcseno
artanh, arctanh, arctgh, artgh	arctangente

Funções	Representando
Exponenciais	
exp	
In	
log10	
log2	
Inxp1	
Potência	não implementada
power	utilize o operador ^
intpower	utilize o operador ^
Módulos	
abs	valor absoluto
sqr	quadrado
sqrt	raiz quadrada
Arredondamentos	
round	arredonda
frac	parte fracionária
int	parte inteira
trunc	trunca

Função	Interface
FxR1	function FxR1(f: string; x: Extended; var y: Extended): Word;
FxRn	function FxRn(f: string; x: Vector; Brackets: Boolean; var y: Extended): Word;
ErrorToStr	function ErrorToStr(Error: Word; Lang: string): string;

Código de erro (const)	Valor	Representando (en-us)	Representando (pt-br)
C_NOERROR	0	No error	Sem erro
C_ERR_BRACKETS	1	Could not parse brackets	Não foi possível interpretar os parênteses
C_ERR_EMPTY_EXPR	2	Empty expression.	Expressão vazia
C_ERR_UNKNOWN_FUNC	3	Unknown function	Função desconhecida
C_ERR_FUNCTION_ERR	4	Error while evaluating function	Erro ao avaliar a função
C_ERR_DEV_BY_ZERO	5	Division by zero	Divisão por zero
C_ERR_CANNOT_SOLVE	6	Error while evaluating expression	Erro ao avaliar a expressão
C_ERR_BAD_NUMBER	7	Incorrect number	Número incorreto
C_ERR_EXCEPTION	8	Exception	Exceção

**FxR1** *function FxR1(f: string; x: Extended; var y: Extended): Word;*

Avalia *f* para dado *x* e o resultado é armazenado em *y*. Devolve C\_NOERROR se não houver erro e, caso contrário, devolve um dos códigos de erro e o valor de *y* será zero.

**FxRn** *function FxRn(f: string; x: Vetor; Colchetes: Boolean; var y: Extended): Word;*

Avalia *f* para dado *x* e o resultado é armazenado em *y*. Devolve C\_NOERROR se não houver erro e, caso contrário, devolve um dos códigos de erro e o valor de *y* será zero.

**ErrorToStr** *function ErrorToStr(Erro: Word; Lingua: string): String;*

Converte o código de erro devolvido por FxR1 ou FxRn para sua representação em cadeia de caracteres, nas línguas en-us e pt-br.

## Exemplos de utilização

Para aqueles que ainda não fizeram programas que utilizam dll, recomenda-se que reproduzam os exemplos que seguem e, a seguir, verificarão que é algo extremamente fácil.

Para aqueles que não entenderam as descrições sobre as rotinas contidas na dll, recomenda-se que releiam o texto da seção anterior, *Breve descrição das rotinas da biblioteca Interpretador.dll*, até entenderem. Depois de entendido, passem aos exemplos.

### Um exemplo completo - versão para console (DOS)

Hoje eu estou com aquela preguiça característica de alunos de graduação em computação e vou me limitar a desenvolver um programa para DOS. Você acreditou nisto? Se sim, é uma múmia. Vou desenvolver um programa para DOS para que (1) você veja como fazer um deles utilizando ambiente de desenvolvimento de programa que, em geral, é utilizado para fazer programas visuais; (2) você veja que utilizar uma DLL é algo extremamente comum com programas visuais, mas é (ou deveria ser) tão comum em programas para console. Mas não vou me limitar apenas a isto. Continue lendo e verá.

Crie um diretório com nome apropriado, como X:\meu diretório pessoal\MMC\Interpretador e, dentro dele crie os diretórios, com nomes tais como DOS e Windows, para fazer duas versões diferentes de exemplos que utilizam o interpretador.

Mãos à obra. Inicie o Lazarus e escolha, no menu, Projeto , Novo Projeto ... , Programa simples e será criado um novo projeto que não passa de um programa Pascal, mas que é tratado como projeto.

**program** Proj ect1;

```
begin
end.
```

Salve, no recém-criado diretório DOS, com o nome InterpretadorExemploDOS.ipi e copie o que segue. Leia cada linha atentamente e entenderá tudo.

```

program InterpretadorExempl oDOS;
type
  VetorX = array of Extended;
function FxR1(f: string; x: Extended; var y: Extended): Word; stdcall; external 'Interpretador.dll';
function FxRn(f: string; xVetor: VetorX; Colchetes: Boolean; var y: Extended): Word; stdcall;
  external 'Interpretador.dll';
function ErrorToStr(Error: Word; Lang: string): string; stdcall; external 'Interpretador.dll';

var
  Funcao: string;
  x, y: Extended;
  xVetor: VetorX;
  Colchetes: Boolean; // Não utilizado nas duas primeiras chamadas (utilizados False e True)
                      // Utilizado na terceira chamada com Colchetes = True
  ResultadoFxR1: Word; // Não utilizado na primeira chamada (serviria para tratar erros)
                      // Utilizado na segunda chamada
  ResultadoFxRn: Word; // Não utilizado nas duas primeiras chamadas (serviria para tratar erros)
                      // Utilizado na terceira chamada
  TextoErro: string;

begin
  // f: R -> R sem tratamento de erro
  x := 2;
  Funcao := 'x^2 + sin(x)'; // sin(x) grafado corretamente
  FxR1(Funcao, x, y);
  Writeln('Executando f: R -> R');
  Writeln('x = ', x, '#10, 'f(x) = ', Funcao, '#10, 'y = ', y);
  Writeln;

  // f: R -> R com tratamento de erro
  x := 2;
  Funcao := 'x^2 + seno(x)'; // seno(x) grafado incorretamente
  ResultadoFxR1 := FxR1(Funcao, x, y); // y será nulo e erro será 3
  Writeln('Executando f: R -> R');
  Writeln('x = ', x, '#10, 'f(x) = ', Funcao, '#10, 'y = ', y);
  if ResultadoFxR1 <> 0 then
    begin
      // Tratamento de erro
      TextoErro := ErrorToStr(ResultadoFxR1, 'pt-br');
      Writeln('Erro ', ResultadoFxR1, ' - ', TextoErro);
    end;
  Writeln;

  // f: Rn -> R sem tratamento de erro (sem colchetes na expressão)
  SetLength(xVetor, 4); // Para usar variáveis x1, x2 e x3 (x0 é desprezada)
  xVetor[1] := 2; xVetor[2] := Pi / 2; xVetor[3] := 0.5;
  Funcao := '(x1^2 + sin(x2)) / x3';
  FxRn(Funcao, xVetor, False, y);
  Writeln('Executando f: Rn -> R');
  Writeln('x1 = ', xVetor[1], '#10, 'x2 = ', xVetor[2], '#10, 'x3 = ', xVetor[3], '#10, 'f(x) = ', Funcao, '#10, 'y = ', y);
  Writeln;

  // f: Rn -> R sem tratamento de erro (com colchetes na expressão)
  SetLength(xVetor, 4); // Para usar variáveis x[1], x[2] e x[3] (x[0] é desprezada)
  xVetor[1] := 2; xVetor[2] := Pi / 2; xVetor[3] := 0.5;
  Funcao := '(x[1]^2 + sin(x[2])) / x[3]';
  FxRn(Funcao, xVetor, True, y);
  Writeln('Executando f: Rn -> R');
  Writeln('x1 = ', xVetor[1], '#10, 'x2 = ', xVetor[2], '#10, 'x3 = ', xVetor[3], '#10, 'f(x) = ', Funcao, '#10, 'y = ', y);
  Writeln;

  // f: Rn -> R com tratamento de erro (com colchetes na expressão)
  Colchetes := True;
  SetLength(xVetor, 4); // Para usar variáveis x[1], x[2] e x[3] (x[0] é desprezada)

```

Informe:

- o que será utilizado da dll;
- qual o tipo de chamada;
- qual o nome da dll.

```

xVetor[1] := 2; xVetor[2] := Pi /2; xVetor[3] := 0.5;
Funcao := '(x[1]^2 + seno(x[2])) / x[3]';           // seno(x) grafado incorretamente
Resul tadoFxRn := FxRn(Funcao, xVetor, Col chetes, y); // y será nulo e erro será 3
Wri teLn(' Executando f: Rn -> R');
Wri teLn('x1 = ',xVetor[1],#10,'x2 = ',xVetor[2],#10,'x3 = ',xVetor[3],#10,'f(x) = ',Funcao,#10,'y = ',y);
if Resul tadoFxRn <> 0 then
begin
  // Tratamento de erro
  TextoErro := ErrorToStr(Resul tadoFxRn, 'en-us');
  Wri teLn(' Erro ', Resul tadoFxRn, ' - ', TextoErro);
end;
Wri teLn;
Wri teLn(' Pressione Return para cair fora.');
ReadLn;
end.

```

Salvou? Legal. Então compile e execute. A solução deverá ser apresentada naquela simpática janelinha preta com a qual é possível fazer um monte de coisas<sup>13</sup> e terá a seguinte cara:

```

Executando f: R -> R
x = 2.0000000000000000000000000000E+0000
f(x) = x^2 + sin(x)
y = 4.90929742682567999985E+0000

Executando f: R -> R
x = 2.0000000000000000000000000000E+0000
f(x) = x^2 + seno(x)
y = 0.0000000000000000000000000000E+0000
Erro 3 - Função desconhecida
                                         função incorreta
                                         resul tado nulo
                                         mensagem de erro

Executando f: Rn -> R
x1 = 2.0000000000000000000000000000E+0000
x2 = 1.57079632679489661926E+0000
x3 = 5.0000000000000000000000000000E-0001
f(x) = (x1^2 + sin(x2)) / x3
y = 1.0000000000000000000000000000E+0001

Executando f: Rn -> R
x1 = 2.0000000000000000000000000000E+0000
x2 = 1.57079632679490E+0000
x3 = 5.0000000000000000000000000000E-0001
f(x) = (x[1]^2 + sen(x[2])) / x[3]
y = 1.0000000000000000000000000000E+0001
                                         função incorreta
                                         resul tado nulo
                                         mensagem de erro

Executando f: Rn -> R
x1 = 2.0000000000000000000000000000E+0000
x2 = 1.57079632679489661926E+0000
x3 = 5.0000000000000000000000000000E-0001
f(x) = (x[1]^2 + seno(x[2])) / x[3]
y = 0.0000000000000000000000000000E+0000
Erro 3 - Unknown function
                                         função incorreta
                                         resul tado nulo
                                         mensagem de erro

Pressione Return para cair fora.

```

Pronto. Um legítimo programa 4 DOS feito com Lazarus. E utilizando uma DLL. Como eu já escrevi em todos os textos anteriores, você está se transformando em um excelente programador. Já pode anotar na linha do tempo no Facebook.

E agora, você, viciado que está em receber programa para copiar e colar, espera que um exemplo com o sugestivo nome InterpretadorExemploWin seja apresentado, não é? Ainda bem que você respondeu que não. Então, o InterpretadorExemploWin está disponível no endereço que você já conhece e espero que seja útil.

---

<sup>13</sup>Caso queira fazer, abra sua janelinha do DOS e digite Calc para iniciar sua calculadora, digite Notepad para iniciar seu bloco de notas, digite AppWiz.cpl, para iniciar o programa de instalar e desinstalar programas e, finalmente, Shutdown -r -t 5 -c "Asta la vista, baby.", para se divertir um pouco.

Tudo bem. Não reclame. Segue a receita, bem no estilo chutado, que será o suficiente. Se tiver alguma dúvida, me pergunte na próxima aula. Mas lembre-se, só pergunte se não entendeu alguma coisa após tentar, tentar, tentar, tentar entender. E acho que você não perguntará nada, afinal, já é um excelente programador. Ou não? Todas as figurinhas necessárias para fazer o programa ficar bonito estão anexadas ao exemplo que se encontra no endereço já conhecido.

### **Um exemplo completo - versão enjanelada (Windows)**

Mãos à obra. Inicie o Lazarus e salve seu projeto em um diretório apropriado, como X:\meu diretório pessoal\MNC\Interpretador\Windows. Salve o programa principal como InterpretadorExemploWin.dpr ou um nome que seja de fácil identificação e a unidade principal como Principal.pas ou deixe Unit1.pas pois será única.

Atribua, para a fórmula, as propriedades BorderStyle=bsDialog, Caption=Brincando com Interpretador de Funções - DLL, Height=162, Width=520, Name=FormaPrincipal ou deixe o padrão Form1, Position=poDesktopCenter e ShowHint=True. Para manter compatibilidade entre diferentes versões do Windows que utilizam diferentes fontes padrão, em Font, atribua Name=Tahoma e Size=8.

Lance um componente TGroupBox (GroupBox1) e atribua as propriedades Left=8, Top=8, Height=113, Width=169 e Caption=Exemplo de cálculo de f: R -> R. Todos os componentes que seguem devem ser lançados no GroupBox1. Lance dois componentes TLabel. Atribua, para Label1, Left=6, Top=10, Caption=f(x) =. Atribua, para Label2, Left=6, Top=38, Caption=x =. Lance três componentes TEdit. Atribua, para Edit1, Left=38, Top=6, Width=120 e apague o valor de Text. Atribua, para Edit2, Left=38, Top=34, Width=48 e apague o valor de Text. Atribua, para Edit3, Left=38, Top=62, Width=120, apague o valor de Text e atribua ReadOnly=True. Lance um componente TSpeedButton (SpeedButton1) e atribua Flat=True, Hint=Calcular valor de f(x) para o ponto dado, Left=6, Top=62 e, em Glyph, carregue o arquivo forward.bmp.

Lance, na fórmula, um componente TGroupBox (GroupBox2) e atribua as propriedades Left=184, Top=8, Height=145, Width=329 e Caption=Exemplo de cálculo de f: Rn -> R. Todos os componentes que seguem devem ser lançados no GroupBox2. Lance dois componentes TLabel. Atribua, para Label3, Left=6, Top=10, Caption=Número de variáveis:. Atribua, para Label4, Left=6, Top=66, Caption=f(x) =. Lance três componentes TEdit. Atribua, para Edit4, Left=111, Top=6, Width=48 e Text=1. Atribua, para Edit5, Left=39, Top=62, Width=120 e apague o valor de Text. Atribua, para Edit6, Left=39, Top=90, Width=120, apague o valor de Text e atribua ReadOnly=True. Lance dois componentes TSpeedButton. Atribua, para SpeedButton2, Flat=True, Hint=Grade para entrada de valores de x[i], Left=111, Top=33 e, em Glyph, carregue o arquivo grid.bmp. Atribua, para SpeedButton3, Flat=True, Hint=Calcular valor de f(x) para o ponto dado, Left=6, Top=90 e, em Glyph, carregue o arquivo forward.bmp. Lance um componente TCheckBox (CheckBox1) e atribua as propriedades Left=6, Top=38 e Caption=Usar colchetes. Lance um componente TStringGrid e atribua Left=168, Top=6, Height=109, Width=132, ColCount=2, RowCount=5, DefaultColWidth=64, DefaultRowHeight=21, FixedCols=1, FixedRows=0 e, em Options, atribua goEditing=True.

Lance, na fórmula, um componente TLabel (Label5) e atribua Left=48, Top=132, Visible=False e Caption=Manual de utilização.

Lance, na fórmula, um componente TSpeedButton (SpeedButton4) e atribua Flat=True, Hint=Informações sobre uso e autoria, Left=16, Top=128, Visible=False e, em Glyph, carregue o arquivo help2.bmp.

Lance, na fórmula, um componente TImage (Image1) e atribua Left=152, Top=128, em Picture carregue o arquivo fx-24x24.bmp, atribua AutoSize=True e Transparent=True.

Lance, na fórmula, um componente TTimer (Timer1) e atribua Interval=10.

Sua fórmula deve ter ficado com a cara que segue e, agora, só falta programar.



Você salvou várias vezes enquanto salpicava este monte de componentes em sua fórmula? Não? Então tomara que o apagão chegue onde você está e..., e..., nem vou explicar o que poderá acontecer.

Crie todas as rotinas necessárias e coloque comentários para documentar suas utilidades.

Cutuque a forma e, em Eventos, dê um toque duplo em OnShow para criar a rotina FormShow. Preencha com o comentário **// Dimensionar StringGrid1 e colocar foco em Edit1.**

Dê um duplo toque no CheckBox1 para criar a rotina CheckBox1Change e preencha com o comentário **// Dimensionar StringGrid1.**

Dê um duplo toque no SpeedButton1 para criar a rotina SpeedButton1Click e preencha com o comentário **// Calcular f(x) com FxR1.**

Dê um duplo toque no SpeedButton2 para criar a rotina SpeedButton2Click e preencha com o comentário **// Dimensionar StringGrid1 e colocar foco em Edit5.**

Dê um duplo toque no SpeedButton3 para criar a rotina SpeedButton3Click e preencha com o comentário **// Calcular f(x) com FxRn.**

Dê um duplo toque no SpeedButton4 para criar a rotina SpeedButton4Click e preencha com o comentário **// Apresentar arquivo de ajuda.**

E já que você inventou de colocar um temporizador em seu programa, dê um duplo toque no Timer1 para criar a rotina Timer1Timer e preencha com o comentário **// Vou brincar de temporizar.**

Toda a estrutura do programa foi montada e agora só falta preencher as rotinas com os códigos adequados.

No início da seção implementation, antes da rotina FormShow, defina tipos, variáveis e importação de rotinas da DLL, escrevendo o código que segue.

```

type
  Vetor = array of Extended;

var
  n: Integer;

function FxR1(f: String; x: Extended; var y: Extended): Word; stdcall; external 'Interpretador.dll';
function FxRn(f: String; x: Vetor; colchetes: Boolean; var y: Extended): Word; stdcall;
  external 'Interpretador.dll';

```

Como você percebeu, já que é um programador superatento, não vou utilizar a rotina ErrorToStr.

Não salve agora. Espere para mais tarde, assim, se der zica e o computador travar, você poderá perder tudo. Claro que não tem nenhum problema, pois basta digitar tudo novamente. :-)

Na rotina FormShow, após os comentários, escreva:

```
SpeedButton2. Click;
Edit1. SetFocus;
```

FormShow solicita que SpeedButton2Click seja executada e esta redimensionará a grade, de acordo com o valor de Edit4.Text, que é o número de variáveis (ou componentes do vetor  $x$ ), para  $f: \mathbb{R}^n \rightarrow \mathbb{R}$  (GroupBox2).

Na rotina CheckBox1Change, após os comentários, escreva:

```
SpeedButton2. Click;
```

CheckBox1Change também solicita que SpeedButton2Click seja executada e esta redimensionará a grade e escreverá os nomes das variáveis com ou sem colchetes, na grade.

Na rotina SpeedButton1Click, antes de begin, defina as variáveis.

```
var
  Erro: Word;
  x, y: Extended;
```

E, após os comentários, escreva:

```
if (Trim(Edit1.Text) = '') then
begin
  ShowMessage('Informe uma função.');
  Exit;
end;
x := StrToFloat(Edit2.Text);
Erro := FxR1(Edit1.Text, x, y);
if (Erro <> 0) then
begin
  ShowMessage('Erro ao avaliar expressão.');
  Exit;
end;
Edit3.Text := FloatToStrF(y, ffGeneral, 10, 6);
```

Caso queira, defina uma variável f do tipo string e, no início, escreva  $f := (\text{Trim(Edit1.Text)})$ . Com isto, você poderá escrever, mais adiante,  $\text{Erro} := \text{FxR1}(f, x, y)$ . Ficará mais claro, para entender o código, para leitura e manutenção futura. Afinal, recordar é viver.

Na rotina SpeedButton2Click, antes de begin, defina a variável.

```
var
  i : Integer;
```

E, após os comentários, escreva:

```
try
  n := StrToInt(Edit4.Text);
except
begin
  ShowMessage('Número de componentes de X inválido.');
  Edit4.SetFocus;
  Exit;
end;
StringGrid1.RowCount := n;
for i := 0 to n-1 do
  if CheckBox1.Checked then
    StringGrid1.Cells[0, i] := 'x[' + IntToStr(i+1) + ']'
  else
    StringGrid1.Cells[0, i] := 'x' + IntToStr(i+1);
```

```

if n <= 5 then
begin
  StringGrid1.Height := 25+(n-1)*21;
  StringGrid1.Width := 132;
end
else
begin
  StringGrid1.Height := 25+4*21;
  StringGrid1.Width := 132+GetSystemMetrics(SM_CXVSCROLL); // Largura do ScrollBar
end;
Edi t5.SetFocus;

```

É esta rotina que redimensiona a grade e escreve os nomes das variáveis com ou sem colchetes, na grade. Diferentes versões do Windows têm diferentes larguras de ScrollBar. Para compatibilidade entre versões, utiliza-se a largura em tempo de execução com GetSystemMetrics. Para isto, é necessário acrescentar Windows na cláusula uses, no início do programa ou no início da interface. Leia sobre isto no capítulo 13.

Na rotina SpeedButton3Click, antes de begin, defina as variáveis.

```

var
  i: Integer;
  Erro: Word;
  y: Extended;
  x: Vetor;

```

E, após os comentários, escreva:

```

if (Trim(Edi t5.Text) = '') then
begin
  ShowMessage('Informe uma função.');
  Exit;
end;
SetLength(x, n+1); // Lembre-se que a posição [0] de Vetor é desprezada
for i := 1 to n do
  try
    x[i] := StrToInt(GridView1.Cells[1, i-1]);
  except
    begin
      if CheckBox1.Checked then
        ShowMessage('Valor inválido para x[' + IntToStr(i) + ']')
      else
        ShowMessage('Valor inválido para x' + IntToStr(i));
      Exit;
    end;
  end;
end;
Erro := FxRn(Edi t5.Text, x, CheckBox1.Checked, y);
//if (Erro <> C_NOERROR) then
// Lembre-se que as constantes de erro não estão na interface
// Então, utilize seus valores numéricos
if (Erro <> 0) then
begin
  ShowMessage('Erro ao avaliar expressão.');
  Exit;
end;
Edi t6.Text := FloatToStrF(y, ffGeneral, 10, 6);

```

Na rotina SpeedButton1Click, pode-se definir uma variável f do tipo string e, no início, escrever f := (Trim(Edi t1.Text)); e, mais adiante, escrever Erro := FxR1(f, x, y);. O mesmo, em SpeedButton3Click.

Na rotina SpeedButton4Click, após os comentários, escreva:

```

if not OpenDocument('Interpretador.chm') then
  ShowMessage('Não foi possível abrir o arquivo de ajuda Interpretador.chm.');

```

Esta instrução executará o arquivo de ajuda chamado Interpretador.chm. Para que a função OpenDocument seja utilizada, é necessário acrescentar LCLIntf na cláusula uses, no início do programa ou no início da interface. Leia sobre isto no capítulo 15.

E agora você deve estar pensando "como fazer um arquivo chm?". Qualquer dia conversaremos sobre isto. Lembre-se que este texto é para a disciplina Métodos Numéricos Computacionais e, pelo que dizem por aí, chm não é assunto da disciplina. Mas nada impede de conversarmos sobre isto se você for um aluno muito, muito, muito legal. Vá treinando. :-)

Eu imagino que você já salvou, compilou e testou. Funcionou? Se sim, que bom. Se não, procure o erro.

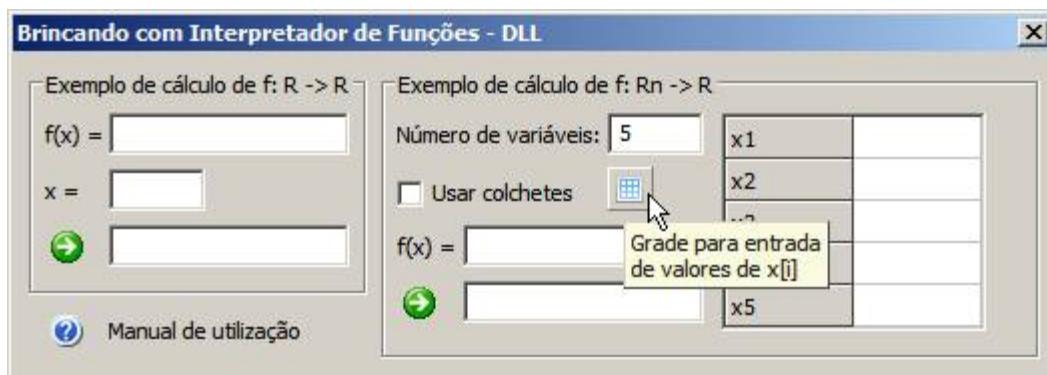
Mas e o temporizador? Para que é que este sujeito no programa? Para fazer uma gracinha temporizada.

Na rotina Timer1Timer, após os comentários, escreva:

```
Image1.Left := Image1.Left - 1;
if Image1.Left = -24 then
begin
  Timer1.Enabled := False;
  SpeedButton4.Visible := True;
  Label5.Visible := True;
end;
```

Finalmente, vá até o menu principal do Lazarus e selecione Projeto, Opções de Projeto, preencha Título com Brincando com Interpretador de Funções - DLL, pressione o botão Carregar ícone e carregue o ícone MNC-InterpretadorExemploWin.ico, gentilmente cedido pelo seu amado professor.

E, depois do finalmente, compare seu programa com o meu e verifique as janelas de informações sobre os botões, ao estacionar o apontador do mouse sobre cada um deles.



Se seus textos estiverem longos e estreitos, provavelmente você escreveu o texto no campo de edição da propriedade Hint sem usar o editor de Hint. Para que a descrição ocupe mais de uma linha, cutuque o SpeedButton1 e dê um toque duplo no campo de edição da propriedade Hint ou pressione o botão com reticências. Quebre o texto onde considerar adequado e pressione o botão OK do editor. Volte ao campo de edição, percorra os caracteres do texto e você verá que na quebra de linha existe caractere não legível. Por exemplo, para o SpeedButton1, Calcular valor de f(x) para o ponto dado. Estes caracteres são #13#10, ou seja, CR e LF, que você bem conhece. Caso queiravê-los, proceda como segue, mas tome cuidado para não danificar a estrutura dos dados da forma. Cutuque a forma com o botão direito do mouse e escolha o item Exibir Fonte (.lfm). No caso do SpeedButton1, você verá algo como Hint = 'Calcular valor de f(x)'#13#10'para o ponto dado'. Agora feche o arquivo lfm para não estragar seu projeto.

Caso queira, repita a operação de quebra de linha para os outros componentes para os quais você atribuiu valor para Hint.

No programa usei o esquema que segue, onde os textos com quebra me parecem mais simpáticos.

Componente	Texto longo	Texto com quebra de linha
SpeedButton1	Calcular valor de $f(x)$ para o ponto dado	Calcular valor de $f(x)$ para o ponto dado
SpeedButton2	Grade para entrada de valores de $x[i]$	Grade para entrada de valores de $x[i]$
SpeedButton3	Calcular valor de $f(x)$ para o ponto dado	Calcular valor de $f(x)$ para o ponto dado
SpeedButton4	Informações sobre uso e autoria	Informações sobre uso e autoria

E, desta forma, você poderá escrever um Hint Romance com várias linhas. Mas não exagere, pois o tempo do Hint é curto e, se o texto for longo, o usuário não terá tempo suficiente para ler.

Pronto. Você fez um belo programa.

Acho que você ficou orgulhoso por fazer *sozinho* este seu programa, mesmo copiando tudo. Eu também estou. Afinal, tenho certeza que ainda vou ganhar aquele notebook de presente de aniversário, que é em 9 de abril.

## P&R

P - Amado professor. Agora ficou ótimo. Com o interpretador, vou tentar fazer o programa que apresenta gráficos para as funções que o usuário digitar. 😊

Joãozinho - 1 dia, 2 horas e 18 minutos

R - Que bom que você gostou.

Faça o programa e você aprenderá o suficiente para fazer vários outros bons programas.

P - Fessôr. Você não vai fazer o programa que eu sugeri e que sonho com ele depois que o P&R foi fechado por motivos técnicos? Vou ter que continuar sonhando com ele? 🤖

Patricinha - 1 dia, 2 horas e 16 minutos

R - Por motivos técnicos o P&R foi encerrado antes que fosse possível responder a última pergunta.

P&R encerrado

Todo mundo já entendeu que está na hora de começar a fazer seus próprios programas.

Sempre que eu faço uma pergunta inteligente o P&R tem problemas técnicos. E agora? Quem vai fazer o programa de meus sonhos? 🎭🎭

É de matar (de raiva) ver a Patricinha chorando. Então, só para a Patricinha, segue a construção do programa de seus sonhos, sugerido ao final do Capítulo anterior. Se você não é a Patricinha, não leia. Afinal, já está na hora de começar a estudar os Métodos Numéricos.

## Um exemplo completo - o programa da Patricinha

Acompanhe o texto, Patricinha. Crie um diretório chamado X:\meu diretório pessoal\MNC\Programa da Patricinha, inicie o Lazarus, inicie um novo projeto do tipo aplicação. Atribua Caption=Brincando de desenhar funções, Height=246, Width=664, Position=poDesktopCenter e salve o projeto neste diretório com o nome GraficosPatricinha.lpr e a unidade com o nome Unit.pas. Quase pronto.

Lance, na fôrma, alguns componentes como segue.

Componente	Left	Top	Caption	Width	Height	Text	Flat
Label1	8	12	f(x) =				
Label2	312	12	a =				
Label3	404	12	b =				
Label4	500	12	Cor				
Label5	554	12	Traço				
Edit1	40	8		264		(remova)	
Edit2	332	8		64		(remova)	
Edit3	428	8		64		(remova)	
ColorButton1	524	6		24	25		True
ComboBox1	589	8		40			
SpeedButton1	634	6		25	24		True
Chart1	0	43		662	200		

Selecione o ComboBox1 e, em Propriedades, corte Items e abra o editor. Digite valores de 1 até 5, cada um em uma linha. Feche o editor. Atribua para a propriedade ItemIndex o valor zero.

Arcore o Chart atribuindo Anchors akBottom=True e akRight=True.

Dê um toque duplo no Chart1 e, em Edit Series, pressione Add e selecione Line series para criar Chart1LineSeries1. Feche o editor.

Crie a rotina para o evento OnClick de SpeedButton1 e escreva as linhas que seguem.

```

function FxR1(f: string; x: Extended; var y: Extended): Word; stdcall; external 'Interpretador.dll';
procedure TForm1.SpeedButton1Click(Sender: TObject);
var
  i, n: Integer;
  a, b, delta, x, y: Extended;
  xVet, yVet: array[1..2001] of Extended;
  f: string;
begin
  f := Trim(Edit1.Text);
  if f = '' then
    begin
      ShowMessage('Digite uma função válida.');
      Edit1.SetFocus;
      Exit;
    end;
  try
    a := StrToFloat(Edit2.Text);
  except
    ShowMessage('Digite um valor válido para a.');
    Edit2.SetFocus;
    Exit;
  end;
  try
    b := StrToFloat(Edit3.Text);
  except
    ShowMessage('Digite um valor válido para b.');
    Edit3.SetFocus;
    Exit;
  end;
  if a >= b then
    begin
      ShowMessage('b deve ser maior que a.');
      Edit3.SetFocus;
      Exit;
    end;

```

```

n := 2001; // Escrevi como variável para colocar na entrada de dados, caso a Patricinha queira
del ta := (b-a)/n;
for i := 1 to n do
begin
  x := a+(i-1)*del ta;
  if FxR1(f, x, y) <> 0 then
  begin
    ShowMessage('Ai guma coisa saiu errada.');
    Exit;
  end;
  xVet[i] := x;
  yVet[i] := y;
end;
Chart1LieneSeries1.Clear;
Chart1LieneSeries1.SeriesColor := ColorButton1.ButtonColor;
Chart1LieneSeries1.LinePen.Width := ComboBox1.ItemIndex;
for i := 1 to n do
  Chart1LieneSeries1.AddXY(xVet[i], yVet[i]);
end;

```

Acredite se quiser: acabou. Ainda acho que mereço o notebook de aniversário.

Cutuque o SpeedButton1 e carregue, em Glyph, Botão Avançar.bmp, ofertado pelo amado professor. Cutuque, no menu, Projeto, Opções de Projeto, atribua Título=Brincando de desenhar funções e carregue o ícone MNC-GraficosPatricinha.ico, também ofertado pelo amado professor.

Claro que você já salvou várias vezes enquanto escrevia o programa e, agora, basta compilar e executar. Ficou como a Patricinha sugeriu.

Pode ficar melhor. No programa, o número de pontos é n (variável), mas é constante (2001), que é um tremendo exagero. Pode-se colocar mais um TLabel e um TEdit, para o usuário entrar com o número de pontos. Pode-se limitar o valor de n entre dois valores, como exemplo, 101 e 1001 e caso o usuário digitar valores abaixo do mínimo ou acima do máximo, o programa seleciona o limite mais próximo. Além disto, se o usuário não digitar o número de pontos, o programa adota um valor que pode ser proporcional à distância (b-a). Assim sobra alguma coisinha para você fazer para melhorar o belo programa sugerido pela Patty.

## P&R

P - Fessôr, amei, mas dá pau quando eu executo. 🤷‍♀️ 😊 ❤️

Aparece uma mensagem informando que não encontrou a dll e que preciso instalar o programa novamente. Mas eu não instalei o programa. Eu copiei o que você fez.

Patricinha - 1 dia, 1 hora e 16 minutos

R - Patricinha, minha santa.

Você precisa copiar o arquivo Interpretador.dll para o diretório de cada programa que utiliza a dll ou manter apenas uma cópia da dll no diretório do Windows.

P - Eu não sabia que precisava. Precisa uma cópia em cada diretório de programa que utiliza a dll ou apenas uma cópia no diretório do Windows? Como a gente iria estar podendo saber disto? 🤔

Patricinha - 1 dia, 1 hora e 14 minutos

R - Por motivos técnicos o P&R foi encerrado antes que fosse possível responder a última pergunta.

P&R encerrado

Todo mundo já entendeu que é necessário ter uma cópia da dll em cada diretório de programa que utiliza a dll ou, apenas uma cópia, em diretório contido no Path, por exemplo, X:\Windows.

Iria estar podendo saber = Iria poder saber = Poderia saber = Saberia. A Patty é bonitinha, mas gosta de gerundismo, que a torna feinha. Então a Patricinha *deveria estar podendo dizer*: Como saberia disto? E tem tanta gente grande que fala *gerundiando*. Que tristeza.

## 7- Derivadas

Em geral, os livros de MNC iniciam com cálculo de raízes de funções de uma variável. Capítulos após, apresentam cálculo de derivadas e, depois, sistemas de equações não-lineares que significa cálculo de raízes de funções com várias variáveis. Mas há métodos para cálculo de raízes de funções de uma variável que utilizam derivadas. A ordem descrita pouco importa no aprendizado do Cálculo Numérico, mas importa e muito, ao se programar os Métodos Numéricos. Então, independentemente de quem nasceu antes, o ovo ou a galinha, neste texto, o cálculo de derivadas é apresentado antes do cálculo de raízes.

Antes de começar a leitura, pegue seus cadernos de Cálculo I e de Cálculo II que espero que tenha guardado e que se lembre de onde os guardou e dê uma olhada no texto relativo às derivadas. Aproveite e veja em seus cadernos de Física I e de Física II os problemas que envolvem derivadas. Se não tem os cadernos, a figura que segue resume quase tudo.



Os itens que seguem são:

- derivada de função de uma variável  $\frac{d}{dx} f(x)$
- derivada de ordem superior de função de uma variável  $\frac{d^k}{dx^k} f(x)$
- derivada de função de várias variáveis  $\nabla f(x) = \left( \frac{\partial}{\partial x_i} f(x) \right)$
- derivada de ordem superior de função de várias variáveis  $Hf(x) = \left[ \frac{\partial^2}{\partial x_i \partial x_j} f(x) \right]$

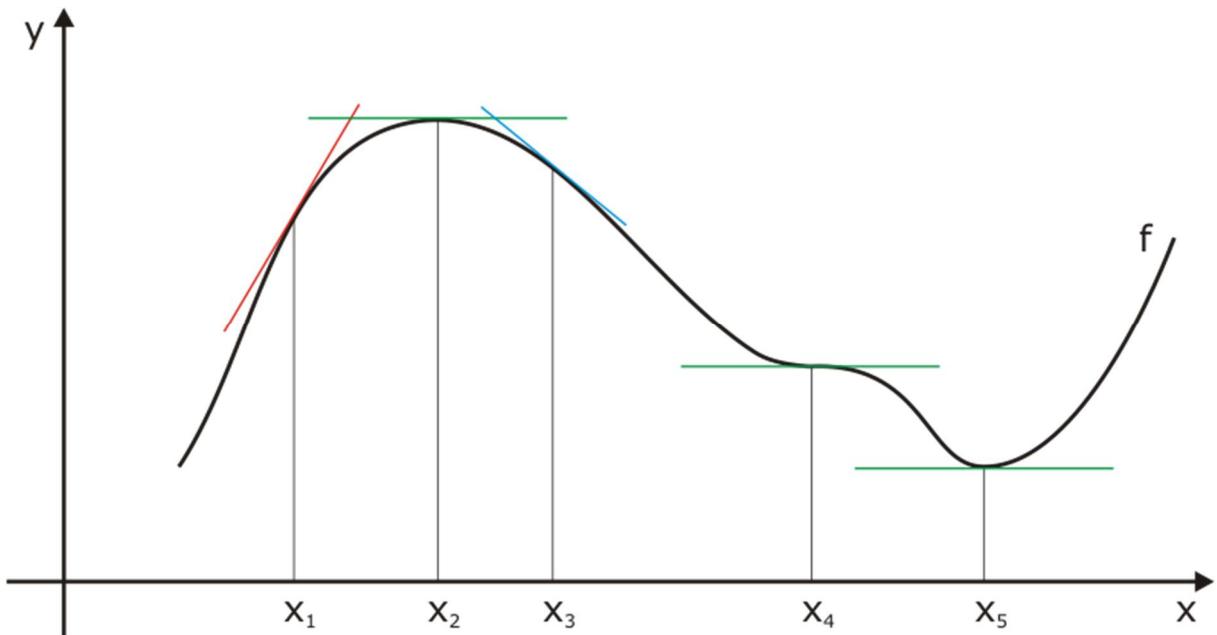
Mãos à obra. Mas antes de colocar as mãos à obra, uma singela observação. A notação utilizada é, como exemplo,  $\frac{d}{dx} f(x)$  e, não,  $\frac{df(x)}{dx}$ . Qual o motivo disto? Derivada é uma operação efetuada sobre uma função. Portanto o operador é  $\frac{d}{dx}$  e a função que sofre a operação é  $f(x)$ . Não é nem frescura, nem rigor matemático. É apenas um costume (saudável) do autor deste texto.

Agora que você já deu uma olhadela em seus cadernos de Cálculo, outra olhadela em seus cadernos de Física, leu quais itens são abordados e verificou a notação, basta ver os algoritmos que seguem.

Nada disto. Tem que ir à aula do assunto, entender os métodos e depois se divertir com os algoritmos. Vejo você na aula. E lembre-se que o que segue não passa de um resumo.

## Derivada de funções de uma variável

Nada como uma figurinha para *entender* o problema.



No ponto  $x_1$  a derivada é crescente. No ponto  $x_3$  é decrescente, Nos pontos  $x_2$ ,  $x_4$  e  $x_5$  é nula.

A interpretação matemática, física, ortodoxa, heterodoxa e exotérica você conhece. Conhece, também, a definição para encontrar a equação da derivada da função, ou seja, para encontrar  $f'(x)$ . Considere o conceito, conjecture e invente uma forma de encontrar  $f'(x)$ , numericamente, para dado valor de  $x$ .

$$\text{Pela definição, } f'(x) = \lim_{\Delta x \rightarrow 0} \frac{f(x + \Delta x) - f(x)}{\Delta x}.$$

Resolvendo este limite, encontra-se a equação de  $f'(x)$ . Mas o que se deseja é encontrar o valor de  $f'(x)$  para determinado  $x$ , ou seja, para  $\bar{x}$ .

Suponha que seja aceitável calcular  $f'(\bar{x})$  com  $\Delta x = h$ , onde  $h$  é pequeno, mas longe de tender a zero. A forma para resolver o problema seria  $f'(\bar{x}) = \frac{f(\bar{x} + \Delta x) - f(\bar{x})}{\Delta x}$  ou  $f'(\bar{x}) = \frac{f(\bar{x} + h) - f(\bar{x})}{h}$ .

Você acha que ficou bom? Eu não acho, não. Pela definição, como  $\Delta x \rightarrow 0$ , não há problema de simetria. Mas para  $\Delta x = h$ , haverá este tipo de problema.

Então, pode-se escrever  $f'(\bar{x}) = \frac{f(\bar{x} + h) - f(\bar{x} - h)}{2h}$ . Agora ficou bonito.

Mas como  $h$  está longe de tender a zero, o valor obtido é impreciso. Então, pode-se calcular o valor suposto da derivada para um dado  $h$  e, a seguir, reduzir o valor de  $h$  e calcular novamente. Repete-se este procedimento e quando dois valores consecutivos para a suposta derivada forem próximos, considera-se como sendo um bom representante do verdadeiro valor da derivada.

## Ensaiando um algoritmo

```
Dados f, x, h, ε
p = ( f(x+h) - f(x-h) ) / (2·h)
Repi ta
    q = p                                // q = aproximação da derivada
    h = h/2                               // ou h/5 ou h/10 ou outro padrão
    p = ( f(x+h) - f(x-h) ) / (2·h)      // p = aproximação da derivada (supostamente melhor)
Até que |p-q| < ε                      // erro admissível de convergência
d = p                                    // d = valor aceito como derivada de f em x
```

Este algoritmo funciona se h for adequado para a função e para o ponto em questão. Mas como saber isto? Há uma série de técnicas que podem ser utilizadas para contornar problemas conhecidos.

## O tamanho de h

O valor inicial de h pode ser dado ou ser uma constante da rotina que calcula a derivada. Se h for inicialmente muito grande, haverá uma quantidade muito grande de iterações e, neste caso, deve-se limitar o número de iterações. Se após várias iterações h for muito pequeno, o quociente p poderá tender a infinito (overflow). Portanto, devem-se utilizar comandos próprios da linguagem computacional que verificam exceções. Além disto, quanto menor for h, maior poderá ser o erro numérico, sendo que, após um número de iterações, o método poderá divergir. Neste caso, deve-se verificar se o erro está aumentando e interromper. Segue um algoritmo que prevê estas várias situações.

```
Dados f, x, h, ε, maxIt
it = 0                                  // iterações
erro = ∞                                 // número muito grande
p = ( f(x+h) - f(x-h) ) / (2·h)
Repi ta para sempre                     // algo como while True do ou, repeat ... until False
    it = it + 1
    q = p                                // q = aproximação da derivada
    h = h/2                               // ou h/5 ou h/10 ou outro padrão
    p = ( f(x+h) - f(x-h) ) / (2·h)      // p = aproximação da derivada (supostamente melhor)
    se |p-q| < erro
        erro = |p-q|
    senão
        d = q                                // está divergindo; parar com estimativa anterior
        interrompe                           // o velho e conhecido Break           ①
    se |p-q| < ε ou it > maxIt
        d = p                                // d = valor aceito como derivada de f em x
        interrompe                           // o velho e conhecido Break           ②
{ fim do Repita para sempre }
```

A instrução repita será interrompida ② quando  $|p-q| < \epsilon$  ou quando  $it > maxIt$ , com  $d = p$  (última estimativa) ou será interrompido ① quando  $|p-q| \geq erro$ , com  $d = q$  (estimativa anterior).

## Diferenciabilidade de f

Se a função não for contínua, pode ser que uma das avaliações de f provoque exceção. Contudo, tenha em mente que, um método numérico nunca investigará se a função é ou não é diferenciável, ou seja, derivável em todos os pontos da região onde se deseja calcular derivadas. Se a derivada é requerida, por exemplo, em um método para encontrar raízes de f em um intervalo pré-determinado, pode ocorrer uma exceção em um ponto onde se calcula a derivada. Matematicamente, poder-se-ia considerar que o método para cálculo da raiz não é apropriado, mas computacionalmente, pode-se contornar a situação, evitando o ponto onde ocorre a exceção.

## O motivo do cálculo da derivada

Se o problema é simplesmente para calcular a derivada, esta não poderá ter um erro que possa ser considerado inadmissível. Mas se o cálculo da derivada for para auxiliar em cálculo de raízes ou de mínimos, será utilizado como método auxiliar. Mesmo que haja um erro que possa parecer inadmissível, ele será diluído através das várias iterações do problema principal.

Os itens anteriores descrevem verdades computacionais que espero não provoquem desconforto aos matemáticos ortodoxos que o leiam.

## Um algoritmo bruto, curto e grosso, mas eficiente

```
Dados f, x, ε                                // ou f, x, h, ε, se h não é função de ε
h = 1000.ε                                    // ou h = 1024.ε (Qual é melhor: 1000 ou 1024?)
p = ( f(x+h) - f(x-h) ) / (2. h)           // estimativa da derivada primeira
Para k = 1, 2, ..., 10                         // no máximo 10 iterações
    q = p                                       // recebe estimativa anterior
    h = h/2                                     // reduz h
    p = ( f(x+h) - f(x-h) ) / (2. h)           // nova estimativa da derivada primeira (esperado que seja melhor)
    se |p-q| < ε                             // ou se |p-q|/|p| < ε
        interrompe                            // o velho e conhecido Break
    { fim do Para k = 1, 2, ..., 10}
d = p                                         // valor aceito para derivada
```

Como h inicia com 1000.ε (ou 1024.ε) e o número máximo de iterações é 10, pode-se considerar que não haverá divergência, pois na décima iteração, h será 0,9765625.ε (ou 1.ε) e é esperado que o valor de ε não seja absurdo.

Para testar o algoritmo, calcule a derivada primeira de  $\sin(x)$ , para  $x = 0,5$ . Utilize  $h = 1$  e  $\epsilon = 0,001$ .

Se  $f(x) = \sin(x)$ ,  $f'(x) = \cos(x)$  e  $f'(0,5) = 0,87758256189037271611628158260383$ . Lembrei-me de um colega de faculdade que, sempre que brincava com  $\pi$ , utilizava 32 algarismos significativos.

Inicialmente  $h = 1$  e, portanto,  $p = 0,738460262604129$ .

k	q	h	p	p-q
0		1	0,738460262604129	
1	0,738460262604129	0,5	0,841470984807897	0,103010722203768000
2	0,841470984807897	0,25	0,868469601537622	0,026998616729725800
3	0,868469601537622	0,125	0,875298975417659	0,006829373880036240
4	0,875298975417659	0,0625	0,877011330656657	0,001712355238998150
5	0,877011330656657	0,03125	0,877439733161178	0,000428402504521141 < 0,001

$d = 0,877439733161178$  com  $|p-q| < 0,001$

Escreva seu algoritmo considerando todas as possibilidades, como no item *O tamanho de h* ou da forma simplificada, como no item *Um algoritmo bruto, curto e grosso, mas eficiente*. Isto é importante e, para os itens subsequentes, a sugestão é feita de forma sucinta, simplesmente como segue.

Escreva seu algoritmo prevendo o controle de exceções, escreva o programa, teste e reserve para utilizar no programa geral proposto no final do capítulo.

## Derivada de ordem superior de funções de uma variável

Se você considera que  $f'(\bar{x}) = \frac{f(\bar{x} + h) - f(\bar{x} - h)}{2.h}$  para  $h$  pequeno e que é possível utilizar um procedimento repetitivo reduzindo  $h$  até que dois valores sucessivos representantes da derivada tenham uma diferença menor que um valor estabelecido, então basta derivar a derivada para encontrar o valor de  $f''(x)$ , para um dado valor de  $x$ .

Então meta a derivada dentro da derivada e resolva o problema.

$$f''(\bar{x}) = \frac{f'(\bar{x} + h) - f'(\bar{x} - h)}{2.h} = \frac{\frac{f(\bar{x} + 2.h) - f(\bar{x})}{2.h} - \frac{f(\bar{x}) - f(\bar{x} - 2.h)}{2.h}}{2.h} = \frac{f(\bar{x} + 2.h) - 2.f(\bar{x}) + f(\bar{x} - 2.h)}{(2.h)^2}$$

Um possível algoritmo é como o anterior, onde  $p$  é calculado com esta nova equação.

```
Dados f, x, ε
h = 1000.ε
p = ( f(x+2.h) - 2.f(x) + f(x-2.h) ) / (4.h.h)
Para k = 1, 2, ..., 10
    q = p
    h = h/2
    p = ( f(x+2.h) - 2.f(x) + f(x-2.h) ) / (4.h.h)
    se |p-q| < ε
        interrompe
    { fim do Para k = 1, 2, ..., 10}
d = p
// ou f, x, h, ε, se h não é função de ε
// ou h = 1024.ε
// estimativa da derivada segunda
// no máximo 10 iterações
// recebe estimativa anterior
// reduz h
// nova estimativa da derivada segunda
// ou se |p-q|/|p| < ε
// o velho e conhecido Break
// valor aceito para derivada
```

Para testar o algoritmo, calcule a derivada segunda de  $\sin(x)$ , para  $x = 0,5$ . Utilize  $h = 1$  e  $\epsilon = 0,001$ .

Se  $f(x) = \sin(x)$ ,  $f''(x) = -\sin(x)$  e  $f''(0,5) = -0,47942553860420300027328793521557$ .

Inicialmente  $h = 1$  e, portanto,  $p = -0,339468479927126$ .

k	q	h	p	p-q
0		1	-0,339468479927126	
1	-0,339468479927126	0,5	-0,440781629208555	0,101313149281429000
2	-0,440781629208555	0,25	-0,469520369602038	0,028738740393483500
3	-0,469520369602038	0,125	-0,476933726888783	0,007413357286745190
4	-0,476933726888783	0,0625	-0,478801611641359	0,001867884752575490
5	-0,478801611641359	0,03125	-0,479269495922651	0,000467884281292186

< 0,001

$d = -0,479269495922651$  com  $|p-q| < 0,001$

Escreva seu algoritmo prevendo o controle de exceções, escreva o programa, teste e reserve para utilizar no programa geral proposto no final do capítulo.

Estendendo o conceito, tem-se:

$$f'''(\bar{x}) = \frac{f(\bar{x} + 3.h) - 3.f(\bar{x} + h) + 3.f(\bar{x} - h) - f(\bar{x} - 3.h)}{(2.h)^3}$$

$$f''''(\bar{x}) = \frac{f(\bar{x} + 4.h) - 4.f(\bar{x} + 2.h) + 6.f(\bar{x}) - 4.f(\bar{x} - 2.h) + f(\bar{x} - 4.h)}{(2.h)^4}$$

Os algoritmos serão como os anteriores, alterando a equação de p.

Para testar o algoritmo, calcule a derivada terceira de  $\sin(x)$ , para  $x = 0,5$ . Utilize  $h = 1$  e  $\varepsilon = 0,001$ .

Se  $f(x) = \sin(x)$ ,  $f'''(x) = -\cos(x)$  e  $f'''(0,5) = -0,87758256189037271611628158260383$ .

Inicialmente  $h = 1$  e, portanto,  $p = -0,522884082401304$ .

k	q	h	p	p-q
0		1	-0,522884082401304	
1	-0,522884082401304	0,5	-0,773644542790111	0,250760460388807000
2	-0,773644542790111	0,25	-0,850526589570596	0,076882046780485400
3	-0,850526589570596	0,125	-0,870749613596450	0,020223024025853600
4	-0,870749613596450	0,0625	-0,875869983415157	0,005120369818707000
5	-0,875869983415157	0,03125	-0,877154145437316	0,001284162022159310
6	-0,877154145437316	0,015625	-0,877475440773196	0,000321295335879768 < 0,001

$d = -0,877475440773196$  com  $|p-q| < 0,001$

Para testar o algoritmo, calcule a derivada quarta de  $\sin(x)$ , para  $x = 0,5$ . Utilize  $h = 1$  e  $\varepsilon = 0,001$ .

Se  $f(x) = \sin(x)$ ,  $f''''(x) = \sin(x)$  e  $f''''(0,5) = 0,47942553860420300027328793521557$ .

Inicialmente  $h = 1$  e, portanto,  $p = 0,240368606978133$ .

k	q	h	p	p-q
0		1	0,240368606978133	
1	0,240368606978133	0,5	0,405252597125714	0,164883990147581000
2	0,405252597125714	0,25	0,459819846295737	0,054567249170022800
3	0,459819846295737	0,125	0,474454866351692	0,014635020055955300
4	0,474454866351692	0,0625	0,478178496658870	0,003723630307177930
5	0,478178496658870	0,03125	0,479113504043198	0,000935007384327946 < 0,001

$d = 0,479113504043198$  com  $|p-q| < 0,001$

Todos os algoritmos foram escritos e testados com o critério de parada  $|p-q| < \varepsilon$ . Faça seus programas e teste com o critério de parada  $|p-q| < \varepsilon$  e com  $|p-q|/|p| < \varepsilon$ . Antes mesmo de fazer os programas, você pode testar o algoritmo em uma planilha.

Outro teste é forçar as iterações a um número grande. O que poderá ocorrer após várias iterações?

Teste tudo isto e traga informações para discutir em aula.

## Derivada de função de várias variáveis

Considere  $f: \mathbb{R}^n \rightarrow \mathbb{R}$ . Para cada variável  $x_i$ , pode ser obtida a derivada parcial  $\frac{\partial}{\partial x_i} f(x)$ .

Pela definição,  $\frac{\partial}{\partial x_i} f(x_1, x_2, \dots, x_n) = \lim_{\Delta x \rightarrow 0} \frac{f(x_1, x_2, \dots, x_i + \Delta x, \dots, x_n) - f(x_1, x_2, \dots, x_i, \dots, x_n)}{\Delta x}$ , ou seja, apenas a componente  $x_i$  sofre variação para o cálculo da derivada parcial.

Considerando o desenvolvimento apresentado para o cálculo numérico de  $f'(\bar{x})$ , tem-se:

$$\frac{\partial}{\partial x_i} f(\bar{x}_1, \bar{x}_2, \dots, \bar{x}_n) = \frac{f(\bar{x}_1, \bar{x}_2, \dots, \bar{x}_i + h, \dots, \bar{x}_n) - f(\bar{x}_1, \bar{x}_2, \dots, \bar{x}_i - h, \dots, \bar{x}_n)}{2.h}$$

O vetor gradiente, que contém todas as derivadas parciais, é dado por  $\nabla f(\bar{x}) = \left( \frac{\partial}{\partial x_i} f(\bar{x}) \right)$ .

Um singelo algoritmo para calcular  $\frac{\partial}{\partial x_i} f(\bar{x})$  pode ser escrito como segue.

```
Dados f, x, i, ε                                // Lembre-se que x é um vetor
h = 1000.ε                                       // ou h = 1024.ε
xi = x[i];                                         // armazena x[i] que será alterado, para restaurá-lo
x[i] = xi + h;                                    // modifica apenas componente x[i] do vetor
f1 = f(x);                                         // calcula a primeira parte da expressão
x[i] = xi - h;                                    // modifica apenas componente x[i] do vetor
f2 = f(x);                                         // calcula a segunda parte da expressão
p = (f1-f2) / (2.h)                               // estimativa da derivada
Para k = 1, 2, ..., 10                            // no máximo 10 iterações
  q = p                                            // recebe estimativa anterior
  h = h/2                                          // reduz h
  x[i] = xi + h;                                 // modifica apenas componente x[i] do vetor
  f1 = f(x);                                     // calcula a primeira parte da expressão
  x[i] = xi - h;                                 // modifica apenas componente x[i] do vetor
  f2 = f(x);                                     // calcula a segunda parte da expressão
  p = (f1-f2) / (2.h)                            // nova estimativa da derivada (esperado que seja melhor)
  se |p-q| < ε                                  // ou se |p-q|/|p| < ε
    interrompe                                    // o velho e conhecido Break
{ fim do Para k = 1, 2, ..., 10}
d = p                                              // valor aceito para derivada
```

Este algoritmo é uma cópia descarada do algoritmo que calcula derivada de  $f: R \rightarrow R$ , com as modificações necessárias para somar e subtrair  $h$  apenas na componente  $x_i$  do vetor  $x$ .

Lembre-se que se  $x$  for uma variável global, no final da rotina é necessário restaurar a componente  $i$  do vetor  $x$ , fazendo  $x[i] = xi$ .

Considere uma rotina chamada `DerivadaParcial(f, x, i, ε)` e pode-se escrever um algoritmo para calcular o gradiente de  $f$  como segue.

```
Dados f, x, i, ε                                // Lembre-se que x é um vetor
Para i = 1, ..., n faça
  Gradi ente[i] = DerivadaParcial(f, x, i, ε)
```

Agora faça a sua parte que é programar as rotinas para cálculo das derivadas parciais e do gradiente.

## Derivada de ordem superior de função de várias variáveis

Considere  $f: R^n \rightarrow R$ . Para cada variável  $x_i$ , pode ser obtida a derivada parcial segunda  $\frac{\partial^2}{\partial x_i \partial x_j} f(x)$ .

Utilizando os mesmos conceitos descritos nas seções anteriores, pode-se calcular a derivada parcial segunda como segue.

Para  $x_i \neq x_j$

$$\frac{\partial^2}{\partial x_i \partial x_j} f(\bar{x}_1, \bar{x}_2, \dots, \bar{x}_n) = \frac{1}{(2.h)^2} \cdot \left[ f(\bar{x}_1, \bar{x}_2, \dots, \bar{x}_i + h, \dots, \bar{x}_j + h, \dots, \bar{x}_n) - f(\bar{x}_1, \bar{x}_2, \dots, \bar{x}_i + h, \dots, \bar{x}_j - h, \dots, \bar{x}_n) \right. \\ \left. - f(\bar{x}_1, \bar{x}_2, \dots, \bar{x}_i - h, \dots, \bar{x}_j + h, \dots, \bar{x}_n) + f(\bar{x}_1, \bar{x}_2, \dots, \bar{x}_i - h, \dots, \bar{x}_j - h, \dots, \bar{x}_n) \right]$$

Para  $x_i = x_j$

$$\frac{\partial^2}{\partial x_i \partial x_j} f(\bar{x}_1, \bar{x}_2, \dots, \bar{x}_n) = \frac{f(\bar{x}_1, \bar{x}_2, \dots, \bar{x}_i + 2.h, \dots, \bar{x}_n) - 2.f(\bar{x}_1, \bar{x}_2, \dots, \bar{x}_i, \dots, \bar{x}_n) + f(\bar{x}_1, \bar{x}_2, \dots, \bar{x}_i - 2.h, \dots, \bar{x}_n)}{(2.h)^2}$$

A matriz hessiana, que contém todas as derivadas parciais, é dada por  $Hf(x) = \left[ \frac{\partial^2}{\partial x_i \partial x_j} f(x) \right]$ .

Um singelo algoritmo para calcular  $\frac{\partial^2}{\partial x_i \partial x_j} f(x)$  pode ser escrito como segue.

```
Dados f, x, i, j, ε // Lembre-se que x é um vetor
h = 1000.ε // ou h = 1024.ε
xi = x[i] // armazena x[i] que será alterado, para restaurá-lo
xj = x[j] // armazena x[j] que será alterado, para restaurá-lo
se i ≠ j // estimativa da derivada para i ≠ j
    x[i] = xi + h; x[j] = xj + h; f1 = f(x); // altera x[i], x[j] e calcula primeira parte da expressão
    x[j] = xj - h; f2 = f(x); // altera x[j] e calcula segunda parte da expressão
    x[i] = xi - h; f4 = f(x); // altera x[i] e calcula quarta parte da expressão
    x[j] = xj + h; f3 = f(x); // altera x[j] e calcula terceira parte da expressão
    p = (f1-f2-f3+f4) / (4. h. h) // estimativa da derivada
senão // estimativa da derivada para i = j
    x[i] = xi + 2h; f1 = f(x) // altera x[i] e calcula primeira parte da expressão
    x[i] = xi - 2h; f3 = f(x) // altera x[i] e calcula terceira parte da expressão
    x[i] = xi; f2 = f(x) // altera x[i] e calcula segunda parte da expressão
    p = (f1-2. f2+f3) / (4. h. h) // estimativa da derivada
Para k = 1, 2, ..., 10 // no máximo 10 iterações
    q = p // recebe estimativa anterior
    h = h/2 // reduz h
    se i ≠ j
        x[i] = xi + h; x[j] = xj + h; f1 = f(x)
        x[j] = xj - h; f2 = f(x)
        x[i] = xi - h; f4 = f(x)
        x[j] = xj + h; f3 = f(x)
        p = (f1-f2-f3+f4) / (4. h. h)
    senão
        x[i] = xi + 2h; f1 = f(x)
        x[i] = xi - 2h; f3 = f(x)
        x[i] = xi; f2 = f(x)
        p = (f1-2. f2+f3) / (4. h. h) // nova estimativa da derivada (esperado que seja melhor)
        se |p-q| < ε // ou se |p-q|/|p| < ε
            interrompe // o velho e conhecido Break
    { fim do Para k = 1, 2, ..., 10}
    d = p // valor aceito para derivada
```

Este algoritmo é uma cópia desacarada do algoritmo anterior, modificando o cálculo de p, para calcular a derivada parcial segunda.

Lembre-se que se x for uma variável global, no final da rotina é necessário restaurar as componentes i e j do vetor x, fazendo  $x[i] = xi$  e  $x[j] = xj$ .

Considere uma rotina chamada DerivadaParcialSegunda( $f, x, i, j, \varepsilon$ ) e pode-se escrever um algoritmo para calcular a hessiana de  $f$  como segue.

```
Dados f, x, i, j, ε // Lembre-se que x é um vetor
Para i = 1, ..., n faça
  Para j = 1, ..., n faça
    Hessiana[i, j] = DerivadaParcialSegunda(f, x, i, j, ε)
```

Como a hessiana é simétrica, pode-se melhorar o algoritmo como segue.

```
Dados f, x, i, j, ε // Lembre-se que x é um vetor
Para i = 1, ..., n faça
  Para j = i, ..., n faça
    Hessiana[i, j] = DerivadaParcialSegunda(f, x, i, j, ε)
  Para i = 2, ..., n faça
    Para j = 1, ..., i-1 faça
      Hessiana[i, j] = Hessiana[j, i]
```

Agora faça a sua parte que é programar as rotinas para cálculo das derivadas parciais segunda e da hessiana.

E a matriz do Jacobiano? Bem lembrado. Um dia você precisará desta matriz.

Dadas  $m$  funções  $f_i: \mathbb{R}^n \rightarrow \mathbb{R}$ , com  $i = 1, \dots, m$ , a matriz do Jacobiano contém, em cada linha, o gradiente transposto de cada função, como segue.

$$Jf(x) = \begin{bmatrix} \nabla^T f_1(x) \\ \vdots \\ \nabla^T f_m(x) \end{bmatrix} = \begin{bmatrix} \frac{\partial}{\partial x_1} f_1(x) & \dots & \frac{\partial}{\partial x_n} f_1(x) \\ \dots & \dots & \dots \\ \frac{\partial}{\partial x_1} f_m(x) & \dots & \frac{\partial}{\partial x_n} f_m(x) \end{bmatrix}$$

Um singelo algoritmo para calcular  $Jf(x)$  pode ser escrito como segue.

```
Dados fi, x, i, j, ε // Lembre-se que x é um vetor
Para i = 1, ..., m faça
  Para j = i, ..., n faça
    Jacobiano[i, j] = DerivadaParcial(fi, x, j, ε)
```

Agora faça a sua parte que é programar a rotina para cálculo da matriz Jacobiana.

## Sugestões para seus programas

Considere um programa com o qual seja possível:

- calcular derivada primeira, segunda, terceira e quarta para  $f: \mathbb{R} \rightarrow \mathbb{R}$ ;
- calcular derivada primeira e segunda  $f: \mathbb{R}^n \rightarrow \mathbb{R}$ ;
- calcular o Gradiente de  $f: \mathbb{R}^n \rightarrow \mathbb{R}$ ;
- calcular a Hessiana de  $f: \mathbb{R}^n \rightarrow \mathbb{R}$ ;
- calcular o Jacobiano de um sistema  $f_i: \mathbb{R}^n \rightarrow \mathbb{R}$ , com  $i=1, \dots, m$ .

Um simpático programa, para resolver derivadas com todos os métodos descritos, poderia ter a cara que segue.

**Brincando com Derivadas**

Derivadas, Gradiente, Hessiana | Jacobiano |

Função Real com uma variável Real -  $f : \mathbb{R} \rightarrow \mathbb{R}$

$f(x) = x^2 - 7$     $x = 3$     $\epsilon = 0,001$

$f'(x) = 6$     $f''(x) = 2$

Função Real com  $n$  variáveis Reais -  $f : \mathbb{R}^n \rightarrow \mathbb{R}$

$n = 6$     $f(x) = [1]^2 + x[2]^2 + x[3]^4 + \sin(x[4]) - x[5] \cdot x[6]$     $\epsilon = 0,000001$

i	1	2	3	4	5	6	Calcula
$x[i]$	1	1	1	1	1	1	

i	1	2	3	4	5	6
$G[i]$	2	2	4,00000025	0,54030228	-1	-1

$H[i,j]$	1	2	3	4	5	6
1	2,00000000	0	0	0	0	0
2	0	2,00000000	0	0	0	0
3	0	0	12,00000000	0	0	0
4	0	0	0	-0,8414709	0	0
5	0	0	0	0	2,16840434	-0,99999999
6	0	0	0	0	-0,99999999	2,16840434

**Brincando com Derivadas**

Derivadas, Gradiente, Hessiana | Jacobiano |

Funções Reais com  $n$  variáveis Reais -  $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$

$m = 3$     $n = 3$     $\epsilon = 0,00001$

i	1	2	3
$x[i]$	1	1	1

i	$f[i]$
1	$\sin(x[1]) + \sin(x[2]) + \sin(x[3])$
2	$x[1] \cdot x[2] + x[1] \cdot x[3] + x[2] \cdot x[3]$
3	$x[1]^3 + x[2] \cdot x[3]^2$

$J[i,j]$	1	2	3
1	0,54030005	0,54030005	0,54030005
2	2	2	2
3	3,00000156	1	2

Lindo, não é? E você vai fazer um assim. Ou não vai. Isto é algo que você decidirá. E como este é o seu primeiro trabalho de MNC para entregar para avaliação, segue uma série de informações já dadas em sala de aula, mas que são repetidas aqui, para ficar tudo muito bem documentado.

### **Instruções claras e precisas sobre trabalhos computacionais para avaliação**

Você já sabe que pode fazer programas em qualquer linguagem, para qualquer sistema operacional, mas eu só vou avaliar os que executarem no computador que tenho a minha disposição, que tem sistema Win32. Se você fizer um programa para Linux, Solaris, iOS, Android, ou outro qualquer, traga seu dispositivo para me apresentar o programa executando. Além disto, se você utilizar linguagens interpretadas, que eu não tenha o interpretador instalado no computador que utilizo, você deverá trazer seu dispositivo e me apresentar o programa executando. Como exemplo, se fizer um programa para executar através do navegador Web, utilizando javascript, eu poderei avaliar. Se for feito com PHP ou Perl, também. Mas se for feito com Ruby ou Python, você deverá me apresentar o programa executando em seu computador. Acho que ficou claro. Dizem as más línguas que eu só gosto de programas feitos com Delphi. Pura maldade, tanto que estou fazendo os programas deste texto com Lazarus. :- ) Você poderá fazer seus programas com qualquer linguagem. Poderá até fazer programas que executam na janelinha do DOS. Não é nada profissional, mas eu medirei se seu programa executa corretamente. Você não precisa utilizar o interpretador e avaliador de funções que deixei à sua disposição. Poderá utilizar outro ou fazer o seu. Poderá até fazer um programa tão pouco profissional que o usuário não consiga entrar com a função desejada e, neste caso, introduza no programa algumas funções para o usuário escolher dentre elas qual irá executar. Poderá fazer programas para executar em ambiente operacional gráfico, que tem um apelo visual mais agradável. Poderá utilizar, neste caso, qualquer ambiente de desenvolvimento, tal como, Delphi, Lazarus, Visual Studio, utilizando Pascal, C++, C#, Visual Basic, dentre outros. Pode até fazer com aquela coisa sem graça conhecida como Java. Mas faça. Leia mais sobre isto em <http://sacoman.dco.fc.unesp.br/mnc/pet/>.

Se você optar em fazer um programa que tenha a aparência das figuras apresentadas ou similar, poderá continuar lendo para ter algumas explicações que poderão ser úteis para você.

### **Um exemplo de programa**

Minha ajuda será *apenas na fabricação da interface*. O resto é com você. Depois de pronto, limpe todo lixo gerado durante a elaboração do projeto, compacte com Zip, Rar ou qualquer outro compactador, com o sugestivo nome T1-RA-V1.ext, onde RA é o número de seu Registro Acadêmico, ext é a extensão fornecida pelo seu compactador, V1 é a primeira versão e entregue via endereço eletrônico de recepção que você já sabe qual é. Não sabe qual é? Então leia tudo sobre o assunto no endereço <http://sacoman.dco.fc.unesp.br/mnc/pet/>.

Então, mãos a obra.

Crie um diretório apropriado para desenvolver seu projeto (e.g.: X:\MNC-Trabalhos\T1). Organize seu ambiente de trabalho e crie subdiretórios necessários para o projeto, como, por exemplo, um diretório chamado Figuras, para as figuras dos botões e ícone e outro chamado Exemplo, para o exemplo executável que está à sua disposição.

Copie de <http://sacoman.dco.fc.unesp.br/mnc/material/> o arquivo M1-Derivadas.rar e descompacte.

Mova Derivadas.exe e Interpretador.dll para o diretório Exemplo, para executar quando quiser. Mantenha uma cópia do Interpretador.dll no diretório principal, onde desenvolverá o projeto. Mova as figuras para o diretório Figuras.

Inicie seu ambiente de projeto, no meu caso, Lazarus, ajeite as janelas da forma como achar conveniente e salve o projeto com o nome Derivadas.lpr e a unidade principal com o nome Principal ou Unit1, já que é a única que existirá.

Atribua Height=534, Width=512, Caption = Brincando com Derivadas, BorderStyle=bsSingle, BorderIcons com biMaximize=False e biMinimize=False, Font com Name=Tahoma e Size=8, Position=poDesktopCenter e, finalmente, ShowHint=True.

No menu Projeto, selecione Opções de Projeto e preencha Título com Brincando com Derivadas e carregue o ícone MNC-Derivadas.ico.

Lance na fôrma um componente TPageControl (PageControl1), atribua Align=alClient, pressione o botão direito do mouse, selecione Adicionar página, para criar TabSheet1 e repita a operação para criar TabSheet2. Pressione a aba TabSheet1, pressione o corpo do componente (abaixo da aba) e atribua Caption=Derivadas, Gradiente, Hessiana. Repita a operação e atribua Caption=Jacobiano para TabSheet2.

Salve, compile e execute. Verifique se a fôrma se parece com fôrma tipo diálogo, mas com ícone e se ao pressionar as abas de PageControl1 elas funcionam corretamente. Se tudo funcionar, continue.

Selecione a aba TabSheet1 e lance dois componentes TGroupBox, como segue.

Componente	Caption	Left	Top	Height	Width
GroupBox1	Função Real com uma variável Real - f : R ---> R	8	8	88	488
GroupBox2	Função Real com n variáveis Reais - f : Rn ---> R	8	104	396	488

Lance, no GroupBox1, alguns componentes como segue.

Componente	Caption	Left	Top	Height	Width	ReadOnly	Text	Flat	Glyph
Label1	f(x) =	8	12						
Label2	x =	262	12						
Label3	$\varepsilon$ =	346	12						
Label4	f'(x) =	8	44						
Label5	f''(x) =	243	44						
Edit1		45	8	192			(remova)		
Edit2		282	8	56			(remova)		
Edit3		362	8	56			(remova)		
Edit4		45	40	192	True		(remova)		
Edit5		282	40	192	True		(remova)		
SpeedButton1		426	8					True	forward.bmp
SpeedButton2		453	8					True	delete.bmp

Atribua, para SpeedButton1, Hint=Calcula e, para SpeedButton2, Hint=Limpa os dados.

Lance, no GroupBox2, alguns componentes como segue.

Componente	Caption	Left	Top	Height	Width	Text	Flat	Glyph	ColCount	RowCount
Label6	n	8	12							
Label7	f(x) =	78	12							
Label8	$\varepsilon$ =	346	12							
Edit6		28	8		20	3				
Edit7		110	8		228	(remova)				
Edit8		362	8		56	(remova)				
SpeedButton3		52	8				True	grid.bmp		
SpeedButton4		426	8				True	forward.bmp		
SpeedButton5		453	8				True	delete.bmp		
StringGrid1		8	44	62	452				7	2
StringGrid2		8	124	62	452				8	2
StringGrid3		8	204	167	468				8	8

Atribua, para SpeedButton3, Hint=Redimensiona as grades, para SpeedButton4, Hint=Calcula e, para SpeedButton5, Hint=Limpas os dados. Atribua DefaultRowHeight=21 para StringGrid1, StringGrid2 e StringGrid3. Para StringGrid1 atribua Options goEditing=True.

Selecione a aba TabSheet2 e lance um componente TGroupBox, como segue.

Componente	Caption	Left	Top	Height	Width
GroupBox3	Funções Reais com n variáveis Reais - f : Rn ---> Rm	8	8	492	488

Lance, no GroupBox3, alguns componentes como segue.

Componente	Caption	Left	Top	Height	Width	Text	Flat	glyph	ColCount	RowCount
Label9	m =	8	12							
Label10	n =	58	12							
Label11	$\varepsilon$ =	138	12							
Edit9		30	8		20	3				
Edit10		78	8		20	3				
Edit11		154	8		56	(remova)				
SpeedButton6		106	8				True	grid.bmp		
SpeedButton7		426	8				True	forward.bmp		
SpeedButton8		453	8				True	delete.bmp		
StringGrid4		8	44	62	452				7	2
StringGrid5		8	124	151	468				2	8
StringGrid6		8	292	167	468				8	8

Atribua, para SpeedButton6, Hint=Redimensiona as grades, para SpeedButton7, Hint=Calcula e, para SpeedButton8, Hint=Limpas os dados. Atribua DefaultRowHeight=21 para StringGrid4, StringGrid5 e StringGrid6. Para StringGrid4 e StringGrid5 atribua Options goEditing=True.

Salve, compile, execute e, se tudo funcionar, continue.

Agora só falta você fazer a sua parte que é programar os algoritmos para solução dos problemas de calcular as derivadas. Só isto. Divirta-se.

Lá vem a Patricinha.

Fessôr, dá uma força, vai. Faça algumas partes do programa para ajudar quem tem pouca intimidade com programação. Não é o meu caso, mas tem uns colegas que precisam de ajuda. 

No início da seção implementation, logo após {\$R \*.lfm}, introduza unit Windows à cláusula uses e variáveis para medidas das barras de rolagem. A seguir, defina tipos, variáveis e funções do problema matemático. A seguir, escreva todas as rotinas matemáticas que serão necessárias ao programa. Uma sugestão é como segue.

```

uses
  Windows; // WinAPI para GetSystemMetrics
var
  ScrollBarAltura, ScrollBarLargura: Integer; // Altura e largura da barra de rolagem
  m, n: Integer; // n = número de variáveis; m = número de funções para Jacobiano
  RepassaErro: Boolean; // Informa erro entre diferentes rotinas

type
  Vetor = array of Extended;
  Matriz = array of array of Extended;

function FxR1(f: String; x: Extended; var y: Extended): Word; stdcall; external 'Interpretador.dll';
function FxRn(f: String; x: Vetor; colchetes: Boolean; var y: Extended): Word; stdcall;
external 'Interpretador.dll';

procedure CopiaVetor(V1, V2: Vetor);
var
  i: Integer;
begin
  for i := 1 to n do
    V1[i] := V2[i];
end;

// Nas rotinas que seguem, var d, G ou H devolvem os valores de Derivada, Gradiente ou Hessiana
// Se não houver erro ao avaliar f(x), o valor da rotina (Word) será 0
// Se houver erro, o valor da rotina (Word) será 1

function DerivadaPrimeira(f: string; x, Epsilon: Extended; var d: Extended): Word;
var
  ...
begin
  ...
end;

function DerivadaSegunda(f: string; x, Epsilon: Extended; var d: Extended): Word;
var
  ...
begin
  ...
end;

function DerivadaParcialPrimeira(f: string; var x: Vetor; i: Byte; Epsilon: Extended; var d: Extended): Word;
var
  ...
begin
  ...
end;

function DerivadaParcialSegunda(f: string; var x: Vetor; i, j: Byte; Epsilon: Extended; var d: Extended): Word;
var
  ...
begin
  ...
end;

function Gradiente(f: string; x: Vetor; Epsilon, d: Extended; var G: Vetor): Word;
var
  ...
begin
  ...
end;

function Hessiana(f: string; x: Vetor; Epsilon, d: Extended; var H: Matriz): Word;
var
  ...
begin
  ...
end;

```

Diferentes versões do Windows têm diferentes espessuras para as barras de rolagem.

A rotina CopiaVetor serve para armazenar cópia de vetor para restaurá-lo quando necessário. Dependendo da maneira como o programa for feito, ela não é necessária.

Os cabeçalhos das rotinas são meras sugestões. Faça como considerar mais conveniente. Os tipos Vetor e Matriz são para armazenamento dinâmico, ou seja, deve-se utilizar SetLength(x, n+1), se x for um vetor de ordem n, para utilizar valores de x[1] até x[n], desprezando x[0]. O mesmo ocorre com variável do tipo matriz. Se não quiser utilizar tamanhos variáveis para vetores e matrizes, utilize tamanho fixo.

Fessôr. E aquela coisa de ler dados da grade e escrever respostas na grade? Dá uma dica, vai. Faça algumas partes do programa para ajudar quem tem pouca intimidade com programação. Não é o meu caso, mas tem uns colegas que precisam de ajuda. 

Cutuque a fôrma. Difícil? Cutuque qualquer componente e pressione ESC quantas vezes forem necessárias até chegar à fôrma. Ou, selecione a fôrma no Inspetor de Objetos.

Em Eventos, dê um toque duplo em OnShow para criar a rotina FormShow. Dê um toque duplo em SpeedButton1 para criar a rotina SpeedButton1Click. Repita a operação para SpeedButton2 até SpeedButton5. Selecione a aba TabSheet2. Repita a operação de criação de rotinas, para SpeedButton6, SpeedButton7 e SpeedButton8. Preencha como segue.

```

procedure TForm1.FormShow(Sender: TObject);
begin
  // Medir largura e altura das barras de rolagem
  // Dimensionar StringGrid1, StringGrid3 e StringGrid5 e colocar foco em Edit1
  // Dimensionar StringGrid4, StringGrid5 e StringGrid6
  ScrollBarAltura := GetSystemMetrics(SM_CYHSCROLL);
  ScrollBarLargura := GetSystemMetrics(SM_CXVSCROLL);
  PageControl1.ActivePage := TabSheet1;
  SpeedButton3.Click;
  Edit1.SetFocus;
  SpeedButton6.Click;
end;

procedure TForm1.SpeedButton1Click(Sender: TObject);
var
  f: string;
  x, Epsilon, d: Extended;
begin
  // Calcular derivada primeira e derivada segunda de f: R -> R
  f := Trim(Edit1.Text);
  if f = '' then
    begin
      ShowMessage('Seria bom se você informasse a função.');
      Edit1.SetFocus;
      Exit;
    end;
  try
    x := Strtofloat(Edit2.Text)
  except
    ShowMessage('Valor inválido para x.');
    Edit2.SetFocus;
    Exit;
  end;
  try
    Epsilon := Strtofloat(Edit3.Text)
  except
    ShowMessage('Valor inválido para Epsilon.');
    Edit3.SetFocus;
    Exit;
  end;
end;
```

```

if (Epsilon > 0.001+10E-10) or (Epsilon < 0.000001-10E-10) then
begin
  ShowMessage('Minha santa.' +#10+'A precisão deve estar entre 1E-3 e 1E-7.' +#10+
              'Para precisões maiores, compre a versão PRO.' +#10+
              'Para precisões menores, compre a versão JR.' +#10+
              'Para precisão nula ou negativa, compre a versão Exotérica.');
  Edi t3.SetFocus;
  Exit;
end;
if DerivadaPrimeira(f, x, Epsilon, d) <> 0 then
begin
  Edi t4.Text := '';
  ShowMessage('Alguma coisa saiu errada no cálculo da derivada primeira.');
end
else
  Edi t4.Text := FloatToStr(d);
if DerivadaSegunda(f, x, Epsilon, d) <> 0 then
begin
  Edi t5.Text := '';
  ShowMessage('Alguma coisa saiu errada no cálculo da derivada segunda.');
end
else
  Edi t5.Text := FloatToStr(d);
end;

procedure TForm1.SpeedButton2Click(Sender: TObject);
begin
  // Limpa tudo ( f: R --> R )
  Edi t1.Text := '';
  Edi t2.Text := '';
  Edi t3.Text := '';
  Edi t4.Text := '';
  Edi t5.Text := '';
  Edi t1.SetFocus;
end;

procedure TForm1.SpeedButton3Click(Sender: TObject);
var
  i : Integer;
begin
  // Ler n em Edit6 e, se houver erro, informar, colocar foco em Edit6 e sair
  // Dimensionar StringGrid1, StringGrid2 e StringGrid3 e colocar foco em Edit7
  // Rotina executada em OnShow que ativa TabSheet1 e, então, pode colocar foco em Edit7
  try
    n := StrToInt(Edit6.Text);
  except
    begin
      ShowMessage('Número de componentes de x inválido.');
      Edit6.SetFocus;
      RepassaErro := True;
      Exit;
    end;
  end;
  if n < 1 then
  begin
    ShowMessage('Minha santa.' +#10+'Leia com atenção:' +#10+
                'Funções ortodoxas possuem ao menos uma variável.' +#10+
                'Se você quiser utilizar funções heterodoxas, não várias +' +
                'ou do aléim, este programa não é o recomendado.');
    Edit6.SetFocus;
    RepassaErro := True;
    Exit;
  end;
  if n < 2 then
  begin
    ShowMessage('Minha santa.' +#10+'Leia com atenção:' +#10+
                'Para funções de uma variável, utilize a parte superior deste programa.');
    Edit6.SetFocus;
    RepassaErro := True;
  end;

```

```

    Exi t;
end;
if n > 20 then
begin
  ShowMessage('Minha santa.' +#10+'Leia com atenção:' +#10+
              'Sinto muito, mas só resolvo problemas com vetores com, no máximo, 20 componentes.');
  Edi t6.SetFocus;
  RepassaErro := True;
  Exi t;
end;
//Grade do vetor x (transposto)
StringGrid1.ColCount := n+1;
StringGrid1.Cells[0, 0] := 'i';
StringGrid1.Cells[0, 1] := 'x[i]';
for i := 1 to n do
  StringGrid1.Cells[i, 0] := IntToStr(i);
if n <= 6 then
begin
  StringGrid1.Height := 46;
  StringGrid1.Width := 68+n*64;
end
else
begin
  StringGrid1.Height := 46+ScrollBar1.Tura;
  StringGrid1.Width := 68+6*64;
end;
//Grade do vetor Gradiente (transposto)
StringGrid2.ColCount := n+1;
StringGrid2.Cells[0, 0] := 'i';
StringGrid2.Cells[0, 1] := 'G[i]';
for i := 1 to n do
  StringGrid2.Cells[i, 0] := IntToStr(i);
if n <= 6 then
begin
  StringGrid2.Height := 46;
  StringGrid2.Width := 68+n*64;
end
else
begin
  StringGrid2.Height := 46+ScrollBar1.Tura;
  StringGrid2.Width := 68+6*64;
end;
//Grade da matriz Hessiana
StringGrid3.ColCount := n+1;
StringGrid3.RowCount := n+1;
StringGrid3.Cells[0, 0] := 'H[i,j]';
for i := 1 to n do
begin
  StringGrid3.Cells[i, 0] := IntToStr(i);
  StringGrid3.Cells[0, i] := IntToStr(i);
end;
if n <= 6 then
begin
  StringGrid3.Height := 25+n*21;
  StringGrid3.Width := 68+n*64;
end
else
begin
  StringGrid3.Height := 151+ScrollBar1.Tura;
  StringGrid3.Width := 452+ScrollBar1.Largura;
end;
Edi t7.SetFocus;
end;

procedure TForm1.SpeedButton4Click(Sender: TObject);
var
  f: string;
  x, G: Vetor;
  H: Matriz;

```

```

Epsilon, d: Extended;
i, j: Integer;
begin
  // Calcular derivada primeira e derivada segunda de f: Rn -> R (gradiente e hessiana)
  // Rever entrada de dados n caso o usuário tenha feito a gracinha de modificar n
  RepassaErro := False;
  SpeedButton3.Click;
  if RepassaErro then
    begin
      RepassaErro := False;
      Exit;
    end;
  f := Trim(Edit7.Text);
  if f = '' then
    begin
      ShowMessage('Seria bom se você informasse a função.');
      Edit7.SetFocus;
      Exit;
    end;
  try
    Epsilon := StrToFloat(Edit8.Text)
  except
    ShowMessage('Valor inválido para Epsilon.');
    Edit8.SetFocus;
    Exit;
  end;
  if (Epsilon > 0.001+10E-10) or (Epsilon < 0.000001-10E-10) then
    begin
      ShowMessage('Minha santa.'+#10+'A precisão deve estar entre 1E-3 e 1E-7.'+#10+
                  'Para precisões maiores, compre a versão PRO.'+#10+
                  'Para precisões menores, compre a versão JR.'+#10+
                  'Para precisão nula ou negativa, compre a versão Exotérica.');
      Edit8.SetFocus;
      Exit;
    end;
  SetLength(x, n+1);
  for i := 1 to n do
    try
      x[i] := StrToFloat(StringGrid1.Cells[i, 1]);
    except
      ShowMessage('Valor inválido para x['+IntToStr(i)+'].' );
      StringGrid1.Col := i;
      StringGrid1.Row := 1;
      StringGrid1.SetFocus;
      Exit;
    end;
  end;
  if Gradi ente(f, x, Epsilon, d, G) <> 0 then
    begin
      for i := 1 to n do
        StringGrid2.Cells[i, 1] := '';
      ShowMessage('Alguma coisa saiu errada no cálculo do Gradi ente.');
    end
    else
      for i := 1 to n do
        StringGrid2.Cells[i, 1] := FloatToStr(G[i]);
  if Hessiana(f, x, Epsilon, d, H) <> 0 then
    begin
      for i := 1 to n do
        for j := 1 to n do
          StringGrid3.Cells[j, i] := '';
      ShowMessage('Alguma coisa saiu errada no cálculo da Hessiana.');
    end
    else
      for i := 1 to n do
        for j := 1 to n do
          StringGrid3.Cells[j, i] := FloatToStr(H[i, j]);
    end;
procedure TForm1.SpeedButton5Click(Sender: TObject);

```

```

var
  i, j: Integer;
begin
  // Limpa tudo ( f: Rn --> R )
  Edi t6.Text := '3';
  Edi t7.Text := '';
  Edi t8.Text := '';
  SpeedButton3.Click; // Restaura StringGrid 1, 2 e 3 ao padrão inicial com n = 3
  for j := 1 to n do
  begin
    StringGrid1.Cells[j, 1] := '';
    StringGrid2.Cells[j, 1] := '';
    for i := 1 to n do
      StringGrid3.Cells[j, i] := '';
  end;
  // Apenas para deixar a casela (1,1) marcada (não é necessário)
  StringGrid1.Col := 1; StringGrid1.Row := 1;
  StringGrid2.Col := 1; StringGrid2.Row := 1;
  StringGrid3.Col := 1; StringGrid3.Row := 1;
end;

procedure TForm1.SpeedButton6Click(Sender: TObject);
var
  i : Integer;
begin
  // Ler m em Edit9 e, se houver erro, informar, colocar foco em Edit9 e sair
  // Ler n em Edit10 e, se houver erro, informar, colocar foco em Edit10 e sair
  // Dimensionar StringGrid4, StringGrid5 e StringGrid6 e colocar foco em StringGrid5
  // Rotina executada em OnShow e, então, só colocar foco em StringGrid5, se ActivePage = TabSheet2
  try
    m := StrToInt(Edit9.Text);
  except
    begin
      ShowMessage('Número de funções inválido.');
      Edit9.SetFocus;
      RepassaErro := True;
      Exit;
    end;
  end;
  if m < 2 then
  begin
    ShowMessage('Minha santa.' +#10+'Leia com atenção:' +#10+
                'O menor número de funções para cálculo' +#10+
                'do Jacobiano, neste programa, é 2.');
    Edit9.SetFocus;
    RepassaErro := True;
    Exit;
  end;
  if m > 20 then
  begin
    ShowMessage('Minha santa.' +#10+'Leia com atenção:' +#10+
                'Sinto muito, mas só resolvo problemas com, no máximo, 20 funções.');
    Edit9.SetFocus;
    RepassaErro := True;
    Exit;
  end;
  try
    n := StrToInt(Edit10.Text);
  except
    begin
      ShowMessage('Número de componentes de x inválido.');
      Edit10.SetFocus;
      RepassaErro := True;
      Exit;
    end;
  end;
  if n < 2 then
  begin
    ShowMessage('Minha santa.' +#10+'Leia com atenção:' +#10+

```

```

        'O menor número de variáveis para cálculo' +#10+
        'do Jacobiano, neste programa, é 2.' );
RepassaErro := True;
Edi t10.SetFocus;
Exi t;
end;
if n > 20 then
begin
  ShowMessage('Minha santa.' +#10+'Leia com atenção:' +#10+
              'Sinto muito, mas só resolvo problemas com vetores com, no máximo, 20 componentes.' );
  Edi t10.SetFocus;
  RepassaErro := True;
  Exi t;
end;
//Grade do vetor x (transposto)
StringGrid4.ColCount := n+1;
StringGrid4.Cells[0, 0] := 'i';
StringGrid4.Cells[0, 1] := 'x[i]';
for i := 1 to n do
  StringGrid4.Cells[i, 0] := IntToStr(i);
if n <= 6 then
begin
  StringGrid4.Height := 46;
  StringGrid4.Width := 68+n*64;
end
else
begin
  StringGrid4.Height := 46+ScrollBarAltura;
  StringGrid4.Width := 68+6*64;
end;
//Grade das m funções
StringGrid5.ColWidths[1] := 384;
StringGrid5.RowCount := m+1;
StringGrid5.Cells[0, 0] := 'i';
StringGrid5.Cells[1, 0] := 'f[i]';
for i := 1 to m do
  StringGrid5.Cells[0, i] := IntToStr(i);
if m <= 6 then
begin
  StringGrid5.Height := 25+m*21;
  StringGrid5.Width := 452;
end
else
begin
  StringGrid5.Height := 151;
  StringGrid5.Width := 452+ScrollBarLargura;
end;
//Grade da matriz Jacobiana
StringGrid6.ColCount := n+1;
StringGrid6.RowCount := m+1;
StringGrid6.Cells[0, 0] := 'J[i,j]';
for i := 1 to m do
  StringGrid6.Cells[0, i] := IntToStr(i);
for i := 1 to n do
  StringGrid6.Cells[i, 0] := IntToStr(i);
if (m <= 6) and (n <= 6) then
begin
  StringGrid6.Height := 25+m*21;
  StringGrid6.Width := 68+n*64;
end;
if (m <= 6) and (n > 6) then
begin
  StringGrid6.Height := 25+m*21+ScrollBarAltura;
  StringGrid6.Width := 452;
end;
if (m > 6) and (n <= 6) then
begin
  StringGrid6.Height := 151;
  StringGrid6.Width := 68+n*64+ScrollBarLargura;
end;

```

```

end;
if (m > 6) and (n > 6) then
begin
  StringGrid6.Height := 151+ScrollBar1Height;
  StringGrid6.Width := 452+ScrollBar1Width;
end;
if PageControl1.ActivePage = TabSheet2 then
  StringGrid5.SetFocus;
end;

procedure TForm1.SpeedButton7Click(Sender: TObject);
var
  f: string;
  x, G: Vetor;
  Epsilon, d: Extended;
  i, j: Integer;
begin
  // Calcular derivada primeira de cada f[i]: Rn -> R (linhas do jacobiano)
  // Rever entrada de dados n caso o usuário tenha feito a gracinha de modificar m ou n
  RepassaErro := False;
  SpeedButton6.Click;
  if RepassaErro then
    begin
      RepassaErro := False;
      Exit;
    end;
  try
    Epsilon := StrToFloat(Edit11.Text);
  except
    ShowMessage('Valor inválido para Epsilon.');
    Edit11.SetFocus;
    Exit;
  end;
  if (Epsilon > 0.001+10E-10) or (Epsilon < 0.000001-10E-10) then
    begin
      ShowMessage('Minha santa.'+#10+'A precisão deve estar entre 1E-3 e 1E-7.'+#10+
                  'Para precisões maiores, compre a versão PRO.'+#10+
                  'Para precisões menores, compre a versão JR.'+#10+
                  'Para precisão nula ou negativa, compre a versão Exotérica.');
      Edit11.SetFocus;
      Exit;
    end;
  SetLength(x, n+1);
  for i := 1 to n do          // Ler vetor x
    try
      x[i] := StrToFloat(StringGrid4.Cells[i, 1]);
    except
      ShowMessage('Valor inválido para x['+IntToStr(i)+'].');
      StringGrid4.Col := i;
      StringGrid4.Row := 1;
      StringGrid4.SetFocus;
      Exit;
    end;
  for i := 1 to m do
  begin
    f := Trim(StringGrid5.Cells[1, i]);
    if f = '' then
      begin
        ShowMessage('Seria bom se você informasse a função.');
        StringGrid5.Row := i;
        StringGrid5.SetFocus;
        Exit;
      end;
    if Gradiente(f, x, Epsilon, d, G) <> 0 then
      begin
        for j := 1 to n do
          StringGrid6.Cells[j, i] := '';
        ShowMessage('Alguma coisa saiu errada no cálculo do Jacobiano, na linha ' + IntToStr(i) + '.');
      end;
  end;
end;

```

```

    else
      for j := 1 to n do
        StringGrid6.Cells[j, i] := FloatToStr(G[j]);
    end;
end;

procedure TForm1.SpeedButton8Click(Sender: TObject);
var
  i, j: Integer;
begin
  // Limpa tudo ( f: Rn --> Rm )
  Edit9.Text := '3';
  Edit10.Text := '3';
  Edit11.Text := '';
  SpeedButton6.Click; // Restaura StringGrid 4, 5 e 6 ao padrão inicial com m = 3 e n = 3
  for j := 1 to n do
  begin
    StringGrid4.Cells[j, 1] := '';
    for i := 1 to m do
      StringGrid6.Cells[j, i] := '';
  end;
  for i := 1 to m do
    StringGrid5.Cells[1, i] := '';
  // Apenas para deixar a casela (1,1) marcada (não é necessário)
  StringGrid4.Col := 1; StringGrid4.Row := 1;
  StringGrid5.Col := 1; StringGrid5.Row := 1;
  StringGrid6.Col := 1; StringGrid6.Row := 1;
end;

```

Tudo isto não passa de sugestão. Faça o seu programa como considerar conveniente.

Tudo isto não passa de sugestão. Faça o seu programa como considerar conveniente.

Tudo isto não passa de sugestão. Faça o seu programa como considerar conveniente.

Tudo isto não passa de sugestão. Faça o seu programa como considerar conveniente.

Tudo isto não passa de sugestão. Faça o seu programa como considerar conveniente.

Fessôr. Dá mais uma força, aí. Faça as rotinas de MNC lá do começo do programa, para ajudar quem tem pouca intimidade com programação. Não é o meu caso, mas tem uns colegas que precisam de ajuda. 

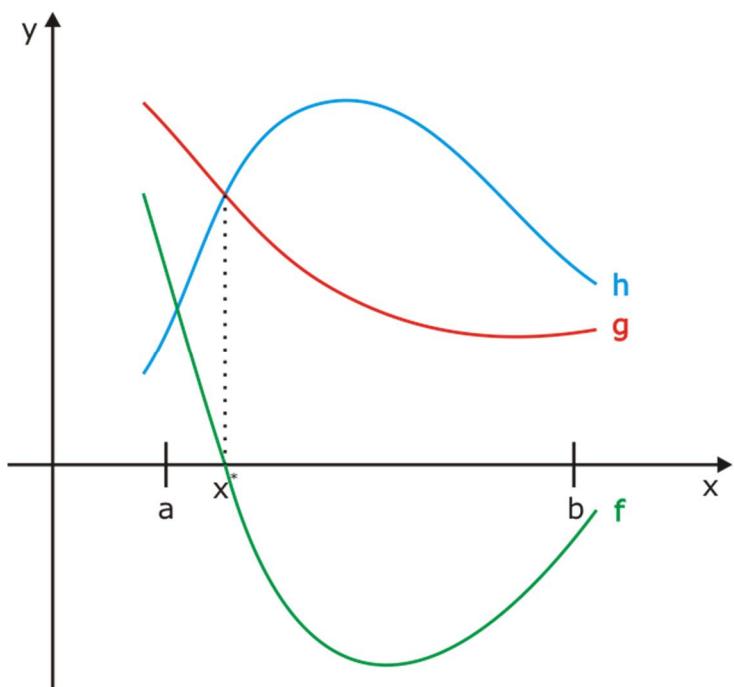


## 8- Raízes de funções de uma variável

Um nome mais completo para o capítulo poderia ser raízes de funções reais com uma variável real, ou seja, com  $f: \mathbb{R} \rightarrow \mathbb{R}$  tal que  $x \mapsto f(x)$ .

Você abre seu livro de Cálculo e tem um capítulo com centenas de exercícios para determinar raízes de funções das mais singelas até as mais cabeludas. E você resolve todas. É dia de prova de Cálculo e você resolve exercícios onde é solicitado encontrar uma ou mais raízes de algumas funções. Tudo muito lindo. Você termina a graduação, começa a trabalhar em uma grande empresa e ninguém pede para você determinar qualquer raiz.

Mas o seu chefe, com cara de poucos amigos, lhe entrega uma folha com algumas equações e outras informações e pede para você determinar alguma coisa que você não entendeu e se desesperou. É assim mesmo que funciona a vida real. Ninguém vai lhe entregar uma função e solicitar as raízes. Volte ao exemplo do Capítulo 2, onde duas funções representam dois comportamentos e deseja-se calcular o ponto onde os dois comportamentos são iguais. Poder-se-ia requerer a região, ou intervalo, onde um comportamento tem valores numéricos superiores ao outro e, neste caso, são requeridas duas raízes que determinam o intervalo  $[x_p, x_q]$  ou informar que o intervalo é  $(-\infty, x_p]$  ou  $[x_p, \infty)$ . Esta é a verdade dos fatos, mas em MNC, você estudará métodos para encontrar as raízes da função que representa o problema. Então, volta-se aos problemas onde é dada uma função e tudo que você terá que fazer é encontrar as raízes.



$g$  representa um comportamento e  $h$  representa outro comportamento.

Os dois comportamentos terão o mesmo valor nos pontos onde  $g(x) = h(x)$ .

Pode existir um único ponto  $x$  que satisfaça esta condição ou mais de um ou nenhum.

Para  $g(x) = h(x)$ , tem-se  $g(x)-h(x) = 0$ .

Se  $f(x) = g(x)-h(x)$ , então  $f(x) = 0$  oferece a solução do problema.

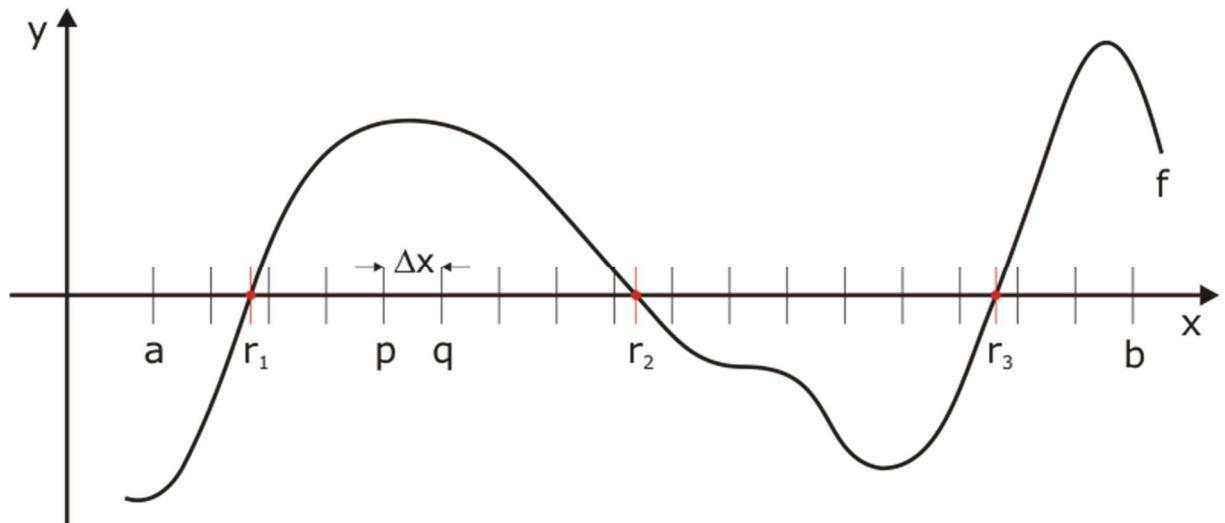
No exemplo, para a região apresentada, existe uma solução que é  $x^*$ .

No texto inicial do Capítulo 2, é apresentada esta figura com este mesmo texto lateral e uma ideia sobre como caminhar entre  $a$  e  $b$  encontrar raízes, comparando o sinal da função entre dois pontos consecutivos do caminhamento.

Este método, geralmente omitido em textos de Métodos Numéricos Computacionais, é chamado aqui, de Método da Busca Uniforme, pois efetua uma busca entre  $a$  e  $b$  com passos constantes  $\Delta$ .

Para caminhar de  $a$  até  $b$  com passos  $\Delta$  pode ser demorado, mas este método sempre encontrará todas as raízes reais do intervalo, se  $\Delta$  for bem escolhido. Isto é intuitivo e visível na figura que segue.

### Método da Busca Uniforme



O texto apresentado na primeira página do Capítulo 2 e repetido neste capítulo deixa clara a ideia geradora deste método.

Utilizando-se a notação da figura,  $a$ ,  $b$ ,  $p$ ,  $q$ ,  $\Delta x$  e  $r_i$ , pode-se escrever o método como segue.

Parte-se de  $a$ , fazendo  $p=a$  e  $q=p+\Delta x$ . Avalia-se a função em  $p$  e  $q$ . Se  $f(p)$  e  $f(q)$  tem sinais diferentes, há ao menos uma raiz no intervalo  $[p, q]$  e se houver mais do que uma será um número ímpar de raízes. Se  $f(p)$  e  $f(q)$  tem sinais iguais, não há raízes no intervalo ou, se houver, será um número par de raízes. Há, também, a remota possibilidade, em problemas reais, de  $p$  ou  $q$  ser raiz. A cada vez que  $f(p)$  e  $f(q)$  forem avaliados, faz-se  $p=q$  e  $q=p+\Delta x$ , avançando um passo. Repete-se o procedimento até atingir o limite superior  $b$ .

Pensando nesta joia de algoritmo para verificar se não passa de uma bijuteria.

Se  $\Delta x$  for um valor escolhido de forma apropriada, sempre que  $f(p)$  e  $f(q)$  tiverem sinais diferentes, haverá uma raiz no intervalo e quando tiverem sinais iguais, nenhuma raiz. Para afirmar que não há raiz, deve-se verificar anteriormente se  $f(p)$  ou  $f(q)$  é nulo. Já está ficando melhor. Ou não, pois o que é um  $\Delta x$  escolhido de forma adequada?

A resposta poderá ferir os sentimentos dos puristas, mas é necessário afirmar mais uma vez (não me lembro de já afirmado isto anteriormente) que os MNC existem para resolver problemas práticos e, não, para efetuar análise matemática. Então, o usuário dos programas elaborados com os algoritmos baseados nos métodos é uma pessoa (espera-se que seja) que sabe que tipo de problema está resolvendo e, portanto, tem uma noção razoável a respeito da solução do problema e, consequentemente, do valor adequado para  $\Delta x$ .

Dependendo do tipo de problema, basta que  $\Delta x$  seja um décimo do intervalo cujo tamanho é  $b-a$ , mas dependendo do problema, poderá ser mais adequado que seja um centésimo. O usuário saberá

escolher, Acredite nisto. E caso o usuário escolha um valor inadequado, muito pequeno ou muito grande para o intervalo considerado, o programa poderá selecionar outro valor.

Para que o método encontre ou separe todas as raízes, deveria fazer uma varredura no intervalo  $[a, b]$  com  $\Delta x \rightarrow 0$ , que seria absolutamente impraticável, pelo tempo computacional.

Além disto, a função poderá assumir o valor zero em um ponto e ter seus valores à direita e à esquerda deste ponto com sinais iguais. Isto é fácil de verificar em equações do tipo  $a.x^2+b.x+c$ , cujas raízes são  $x_1 = (-b + \sqrt{b^2 - 4.a.c})/(2.a)$  e  $x_2 = (-b - \sqrt{b^2 - 4.a.c})/(2.a)$ . Se  $\sqrt{b^2 - 4.a.c}$  for nulo,  $x_1 = x_2$  e não há troca de sinal entre valores à esquerda e à direita de cada raiz. Contudo, pode-se de verificar a possibilidade da existência de raiz, verificando-se, além do sinal, a declividade da função.

Desta forma, o método consegue separar todos os intervalos onde existem raízes. É necessário, a seguir, determinar cada uma das raízes. Pode-se utilizar, para cada intervalo, o mesmo método, utilizando  $\delta x \ll \Delta x$  para encontrar as raízes ou utilizar outro método e, neste caso, a Busca Uniforme servirá apenas para separar as raízes.

A grande vantagem deste método sobre outros é que só efetua avaliações da função, sem usar derivadas e separa todas as raízes. Com as raízes já separadas, outros métodos podem ser mais eficientes, para determinar a raiz de cada intervalo separado.

Chegou a hora de você, amado leitor escrever os algoritmos para os seguintes casos:

- programa para separar as raízes;
- programa para separar a primeira raiz e refinar com  $\delta x \ll \Delta x$  e encontrar a primeira raiz;
- programa para separar as raízes e refinar com  $\delta x \ll \Delta x$  e encontrar todas as raízes.

E após os algoritmos, evidentemente, fazer os correspondentes programas.

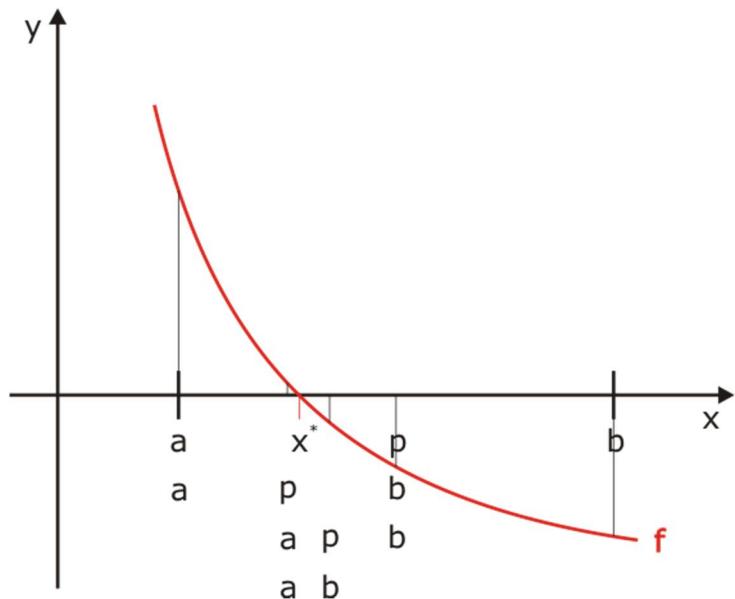
## Método da Divisão ao Meio

O Método da Divisão ao Meio também é conhecido como Método da Bissecção. Mas bissecção significa dividir em duas partes e o método divide exatamente ao meio. Então sem querer ser ortodoxo, cartesiano ou, simplesmente importuno, neste texto, o método se chama Divisão ao Meio.

É razoável que se utilize este método quando se tem certeza que existe uma raiz no intervalo de busca e existe apenas uma. Isto pode ser feito utilizando o Método da Busca Uniforme para separar as raízes. Caso contrário, o método poderá falhar.

A ideia do método é avaliar a função nos extremos do intervalo de busca e se  $f(a)$  e  $f(b)$  tiverem sinais diferentes, considera-se que há raiz no intervalo. Divide-se o intervalo ao meio, fazendo  $p=(a+b)/2$  e avalia-se  $f(p)$ . Se  $f(a)$  tiver o mesmo sinal que  $f(p)$ , não há raiz no intervalo  $[a, p]$  e descarta-se este intervalo fazendo  $a=p$ . Se  $f(a)$  tiver sinal diferente de  $f(p)$ , há raiz no intervalo  $[a, p]$  e descarta-se o intervalo  $[p, b]$ , fazendo  $b=p$ . Sempre que houver raiz em  $[a, p]$  é considerado que não há raiz em  $[p, b]$  e sempre que não houver raiz em  $[a, p]$  é considerado que há raiz em  $[p, b]$ . Assim, um dos intervalos é descartado. O procedimento é repetido até que o intervalo  $[a, b]$  seja suficientemente pequeno para se considerar que um ponto pertencente ao intervalo seja o representante da raiz.

A figura que representa esta ideia é apresentada a seguir. Para efeito de descrição da sequência de passos, considere que  $p=(a+b)/2$  seja descrito como  $p=MA(a,b)$ , ou seja Média Aritmética entre  $a$  e  $b$ .



Inicia-se com  $p = MA(a,b)$ .

Na primeira avaliação,  $f(a).f(p) < 0$  então,  $b = p$  e  $p = MA(a,b)$ .

Na segunda avaliação,  $f(a).f(p) > 0$  então,  $a = p$  e  $p = MA(a,b)$ .

Na terceira avaliação,  $f(a).f(p) < 0$  então,  $b = p$  e, como  $b-a$  é suficientemente pequeno, um ponto do intervalo  $[a, b]$  pode representar  $x^*$ .

Outro critério de parada é verificar se  $f(p)$  é suficientemente próximo de zero.

Chegou a hora de você, amado leitor escrever o algoritmo para este método e após o algoritmo, evidentemente, fazer o correspondente programa.

## Método das Cordas

O Método das Cordas também é conhecido como Método da Secante ou da Posição Falsa, sendo que o último nome é uma tradução de Regula Falsi, descrito no papiro de Rhind (1833-1863), escrito por Ahmes (1650 aC) e de autoria desconhecida. Então, depois de 3665 anos, segue o método.

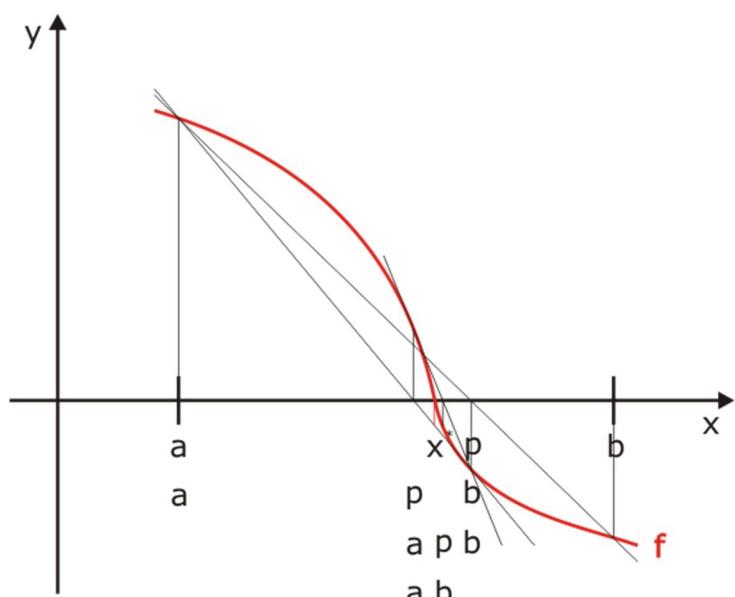
Como informado no método anterior, é razoável que se utilize este método quando se tem certeza que existe uma raiz no intervalo de busca e existe apenas uma. Isto pode ser feito utilizando o Método da Busca Uniforme para separar as raízes. Caso contrário, o método poderá falhar.

O Método da Divisão ao Meio divide o intervalo sem considerar nenhuma outra informação sobre a raiz estar mais próxima de  $a$  ou de  $b$ . Se a função não tiver inflexões, ou seja, se for côncava ou convexa no intervalo  $[a, b]$ , a divisão ao meio, que descarta metade do intervalo, poderá auxiliar na velocidade de convergência. Caso contrário, não passa de conjectura. O Método das Cordas não descarta metade do intervalo, mas procura um ponto que seja mais próximo da raiz.

Considerando a existência de uma raiz no intervalo  $[a, b]$  e sendo  $f(a).f(b) < 0$ , pode-se escolher um ponto  $p$  que fique mais próximo de  $a$  ou de  $b$ , dependendo de  $|f(a)|$  ser menor ou maior do que  $|f(b)|$ . Assim,  $p$  ficará mais próximo da raiz do que se fosse escolhido como o ponto médio do intervalo. Resumindo, no Método das Cordas,  $p$  é selecionado por média ponderada e, não, por média aritmética.

Então,  $p$  será dado por  $(a.f(b) + b.f(a)) / (|f(b)| + |f(a)|)$ . Como  $f(a)$  e  $f(b)$  tem sinais opostos, tem-se  $p = (a.f(b) - b.f(a)) / (f(b) - f(a))$ .

A figura que representa esta ideia é apresentada a seguir. Para efeito de descrição da sequência de passos, considere que  $p = (a.f(b) - b.f(a)) / (f(b) - f(a))$  seja descrito como  $p=MAP(a, f(b), b, f(a))$ , ou seja, Média Aritmética Ponderada entre  $a$  e  $b$  com pesos  $f(b)$  e  $f(a)$ .



Inicia-se com  $p=MAP(a, f(b), b, f(a))$ .

Na primeira avaliação,  $f(a).f(p) < 0$  então,  $b = p$  e  $p=MAP(a, f(b), b, f(a))$ .

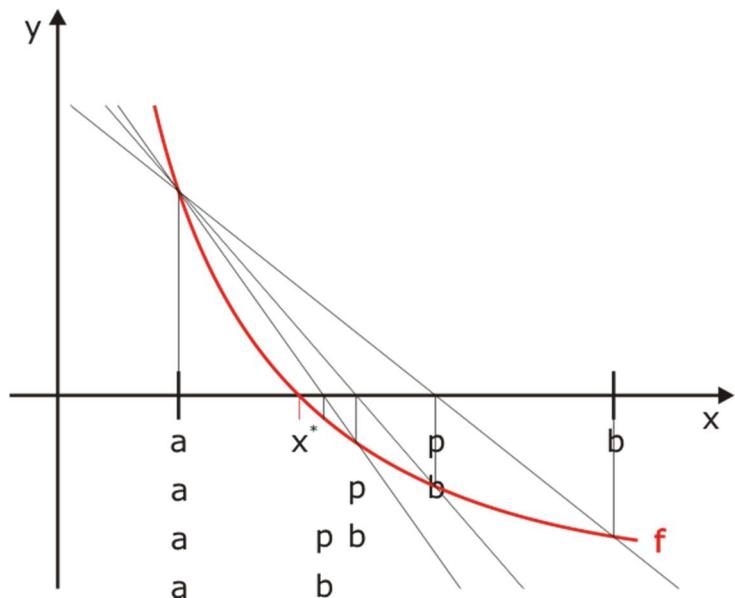
Na segunda avaliação,  $f(a).f(p) > 0$  então,  $a = p$  e  $p=MAP(a, f(b), b, f(a))$ .

Na terceira avaliação,  $f(a).f(p) < 0$  então,  $b = p$  e como  $b-a$  é suficientemente pequeno, um ponto do intervalo  $[a, b]$  pode representar  $x^*$ .

Outro critério de parada é verificar se  $f(p)$  é suficientemente próximo de zero.

Considere que a função seja convexa ou côncava no intervalo  $[a, b]$ . Neste caso, o critério de parada que mede a distância entre  $a$  e  $b$  poderá não ser atingido.

A figura que representa esta ideia é apresentada a seguir. Para efeito de descrição da sequência de passos, considere que  $p = (a.f(b) - b.f(a)) / (f(b) - f(a))$  seja descrito como  $p = \text{MAP}(a, f(b), b, f(a))$ , ou seja, Média Aritmética ponderada entre  $a$  e  $b$  com pesos  $f(b)$  e  $f(a)$ .



Inicia-se com  $p = \text{MAP}(a, f(b), b, f(a))$ .

Na primeira avaliação,  $f(a).f(p) < 0$  então,  $b = p$  e  $p = \text{MAP}(a, f(b), b, f(a))$ .

Na segunda avaliação,  $f(a).f(p) < 0$  então,  $b = p$  e  $p = \text{MAP}(a, f(b), b, f(a))$ .

Na terceira avaliação,  $f(a).f(p) < 0$  então,  $b = p$  e mesmo continuando o procedimento, o intervalo  $[a, b]$  nunca será tão pequeno para que qualquer ponto nele contido possa representar  $x^*$ .

Neste caso é necessário o critério de parada que verifica se  $f(p)$  é suficientemente próximo de zero.

Para a figura, o critério de parada  $|b-a| < \varepsilon$  não é atingível. Porém, após algumas iterações,  $|f(p)| < \varepsilon$ .

No caso onde  $f$  é convexa e decrescente, apenas  $b$  se move e quando é convexa e crescente, apenas  $a$  se move. No caso onde  $f$  é côncava e decrescente, apenas  $a$  se move e quando é côncava e crescente, apenas  $b$  se move. Portanto, para que este método tenha convergência garantida, é necessário que se teste se a distância entre  $a$  e  $b$  é suficiente pequena **ou** se  $f(p)$  é suficientemente próximo de zero.

Como algoritmo, a ideia do método é exatamente a mesma do método anterior, exceto pela forma de se calcular o valor de  $p$  e testar  $|b-a|$  ou  $f(p)$ .

Chegou a hora de você, amado leitor escrever o algoritmo para este método e após o algoritmo, evidentemente, fazer o correspondente programa.

## Método de Newton

O Método de Newton, também conhecido como Método das Tangentes, diferentemente dos anteriores, parte de um ponto e procura por uma raiz que poderá ou não estar no intervalo de busca  $[a, b]$ . O sucesso da busca dependerá do ponto de partida e do comportamento da função.

O método procura, sucessivamente, um ponto mais próximo da raiz do que o ponto corrente, fazendo uso da ideia de aproximações, como segue.

Considere que  $\bar{x}$  satisfaça  $f(x) = 0$ , no intervalo  $[a, b]$  finito e que  $f'(x)$  e  $f''(x)$  sejam contínuas e que preservem o sinal no intervalo. Considere  $x_k$  tal que  $x_k \approx \bar{x}$ , com  $x_k \in [a, b]$  e  $h_k$  uma tolerância positiva tal que  $\bar{x} = x_k + h_k$ .

A série de Taylor, aplicada em  $\bar{x}$ , resulta em  $f(\bar{x}) = f(x_k + h_k) = f(x_k) + h_k \cdot f'(x_k) + \frac{(h_k)^2}{2!} \cdot f''(x_k) + \dots + \text{Erro}$ .

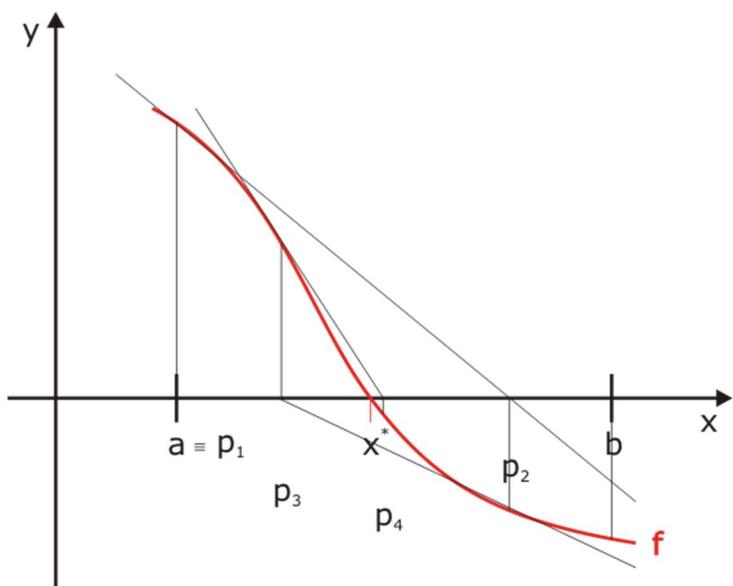
Truncando a série a partir do termo de ordem 2, que produzem valores muito pequenos, tem-se uma aproximação linear dada por  $f(\bar{x}) \approx f(x_k) + h_k \cdot f'(x_k)$ .

Se  $\bar{x}$  é raiz,  $f(\bar{x}) = 0$  e, então,  $f(x_k) \approx -h_k \cdot f'(x_k)$ , portanto,  $h_k \approx \frac{-f(x_k)}{f'(x_k)}$ . Como  $\bar{x} = x_k + h_k$ ,

$$\bar{x} \approx x_k - \frac{f(x_k)}{f'(x_k)}.$$

O valor de  $\bar{x}$  será raiz quando  $h_k$  for próximo de zero e pode-se obter esta situação de forma iterativa, fazendo  $x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$ .

Então o método de Newton é uma sequência de pontos onde  $p = p - f(p)/f'(p)$ . Repete-se o procedimento até que a distância entre os dois últimos pontos da seqüência<sup>14</sup> seja menor que a tolerância admissível.



Inicia-se com  $p=a$ .

Calcula-se  $f(p)$  e sua interseção no eixo das abscissas dará o novo ponto  $p$ .

Repete-se o procedimento até que dois valores de  $p$  consecutivos estejam muito próximos entre si e o último deles possa representar  $x^*$ .

Deve-se armazenar o valor anterior de  $p$  em  $q$  ou testar sua diferença que é  $f(x)/f'(x)$ .

<sup>14</sup> Roubaram o trema do u. Então secuência se tornou sekênciia. Escrevendo com trema sou um fora da lei?  
<http://www.brasil.gov.br/educacao/2014/12/acordo-ortografico-so-entrara-em-vigor-em-2016>

A interpretação geométrica do método é visível na figura anterior, onde se pode considerar, para qualquer das iterações, o triângulo formado pelos catetos  $|p_{k+1} - p_k|$  e  $f(p)$  e pela hipotenusa que tem a inclinação de  $f'(p)$ .

Considere  $\tan(\alpha) = \frac{f(p)}{p_k - p_{k+1}}$  e  $\tan(\alpha) = f'(p_k)$ . Então,  $\frac{f(p)}{p_k - p_{k+1}} = f'(p_k)$  e  $p_{k+1} = p_k - \frac{f(p_k)}{f'(p_k)}$ .

Ou, para menores de idade, pela equação da reta, onde  $y_{k+1} - y_k = m \cdot (x_{k+1} - x_k)$  com  $x_{k+1} = p_{k+1}$ ,  $x_k = p_k$ ,  $y_{k+1} = f(p_{k+1}) = 0$ ,  $y_k = f(p_k)$ ,  $m = f'(p_k)$ . Então,  $0 - f(p_k) = f'(p_k) \cdot (p_{k+1} - p_k)$  e obtém-se  $p_{k+1} = p_k - f(p_k)/f'(p_k)$ .

Um possível algoritmo simplificado para o método é como segue.

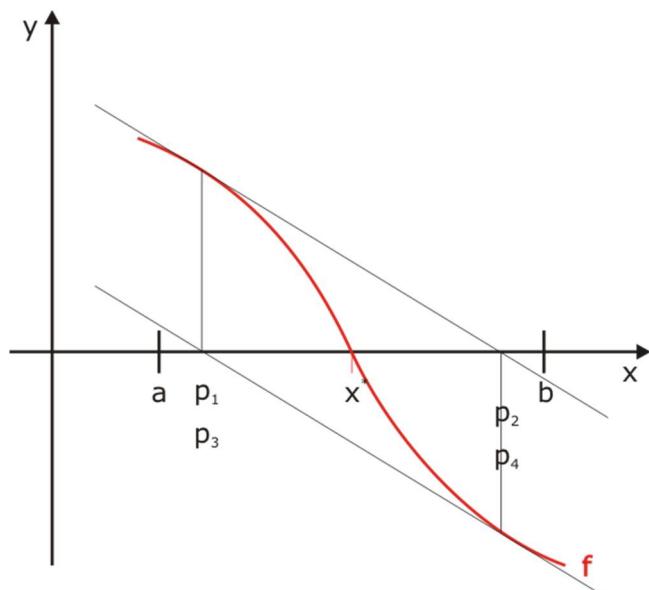
```
Dados f, a, b, ε
p = a                                // ou p = b ou p = um ponto do intervalo [a, b]
Repetir
    q = p
    p = p - f(p)/f'(p)                // se f'(x) = 0, reinicie com outro p ou recalcule f' com p+ε
Até que |p-q| < ε
```

Outro possível algoritmo simplificado para o método é como segue, considerando que  $p-q = f(x)/f'(x)$ .

```
Dados f, a, b, ε
p = a                                // ou p = b ou p = um ponto do intervalo [a, b]
Repetir
    p = p - f(p)/f'(p)                // se f'(x) = 0, reinicie com outro p ou recalcule f' com p+ε
Até que |f(p)/f'(p)| < ε
```

O primeiro algoritmo utiliza uma variável extra, mas o segundo efetua uma avaliação de  $f'$  a mais, a cada iteração. Portanto, o primeiro é muito melhor que o segundo, em minha opinião. É sempre bom lembrar-se que quando você jodia do computador os duendes escovadores de bits que moram dentro dele poderão se vingar de você.

Há casos onde o método não converge. Um exemplo claro é apresentado na figura que segue onde há simetria na função.



Inicia-se com  $p_1$  no intervalo  $[a, b]$ .

Calcula-se  $f(p_1)$  e sua interseção no eixo das abscissas dará o novo ponto  $p_2$ .

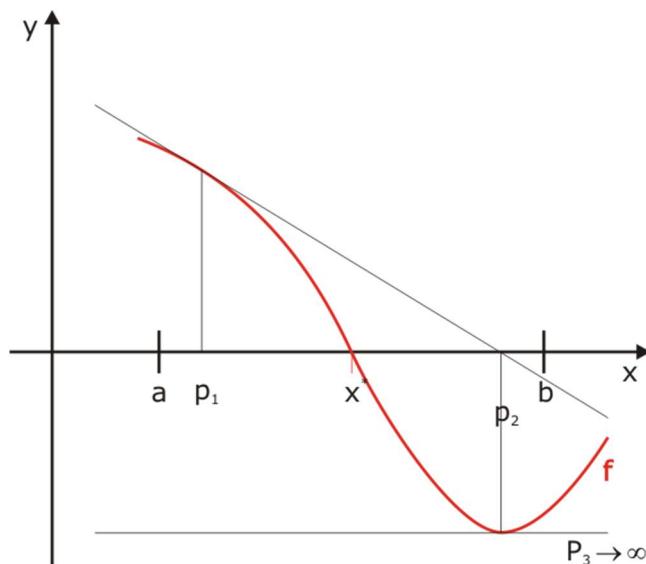
Repete-se o procedimento e encontra-se  $p_3 = p_1$ .

Repete-se o procedimento e encontra-se  $p_4 = p_2$ .

E assim, sucessivamente.

Portanto, neste caso, o método não converge.

Outra situação é apresentada na figura que segue.



Inicia-se com  $p_1$  no intervalo  $[a, b]$ .

Calcula-se  $f(p_1)$  e sua interseção no eixo das abscissas dará o novo ponto  $p_2$ .

Repete-se o procedimento e encontra-se  $p_3 \rightarrow \infty$ .

Portanto, neste caso, o método não converge.

Este método requer, por não convergir sempre, um contador de iterações para ser interrompido.

Assim, um possível algoritmo simplificado para o método, prevendo o limite de iterações, é como segue.

```
Dados f, a, b, ε, maxI t
k = 0
p = a // ou p = b ou p = um ponto do intervalo [a, b]
Repetir
    k = k + 1
    q = p
    p = p -f(p)/f'(p) // se f'(x) = 0, reinicie com outro p ou recalcule f' com p+ε
Até que |p-q| < ε ou k = maxI t
```

Chegou a hora de você, amado leitor escrever o **seu** algoritmo para este método e após o algoritmo, evidentemente, fazer o correspondente programa.

## Método das Cordas Modificado

Nas situações onde o Método das Cordas progride alterando sucessivamente a posição de a ou sucessivamente a posição de b, no caso da função ser convexa ou côncava, da região próxima da raiz, pode-se efetuar uma das modificações que seguem.

Se o ponto a se mantiver fixo ao longo de algumas iterações e apenas o ponto b mudar de posição, pode-se alterar a MAP  $(a.f(b) - b.f(a)) / (f(b) - f(a))$  por  $(a.f(b)/2 - b.f(a)) / (f(b)/2 - f(a))$ .

Se o ponto b se mantiver fixo ao longo de algumas iterações e apenas o ponto a mudar de posição, pode-se alterar a MAP  $(a.f(b) - b.f(a)) / (f(b) - f(a))$  por  $(a.f(b) - b.f(a)/2) / (f(b) - f(a)/2)$ .

Isto significa que se altera a inclinação da reta secante pela metade de seu valor e, com isto, tenta-se fazer com que o próximo ponto mude do intervalo  $[a, p]$  para  $[p, b]$  ou mude do intervalo  $[p, b]$  para  $[a, p]$ .

No algoritmo, utiliza-se um contador de iterações para verificar se apenas um dos extremos, a ou b, está se movendo. Se após 2 ou 3 iterações apenas um dos extremos mudar de posição, utiliza-se na próxima iteração a modificação. A seguir, volta-se ao procedimento usual e reinicia-se a contagem de iterações referente ao movimento do ponto extremo.

Chegou a hora de você, amado leitor escrever o algoritmo para este método e após o algoritmo, evidentemente, fazer o correspondente programa.

## Método de Newton Modificado

Ô mania que esse povo tem de modificar as coisas.

Para acelerar o Método de Newton, pode-se utilizar o mesmo valor de uma derivada previamente calculada, para as próximas iterações. Após algumas iterações, deve-se calcular novamente a derivada para o ponto corrente.

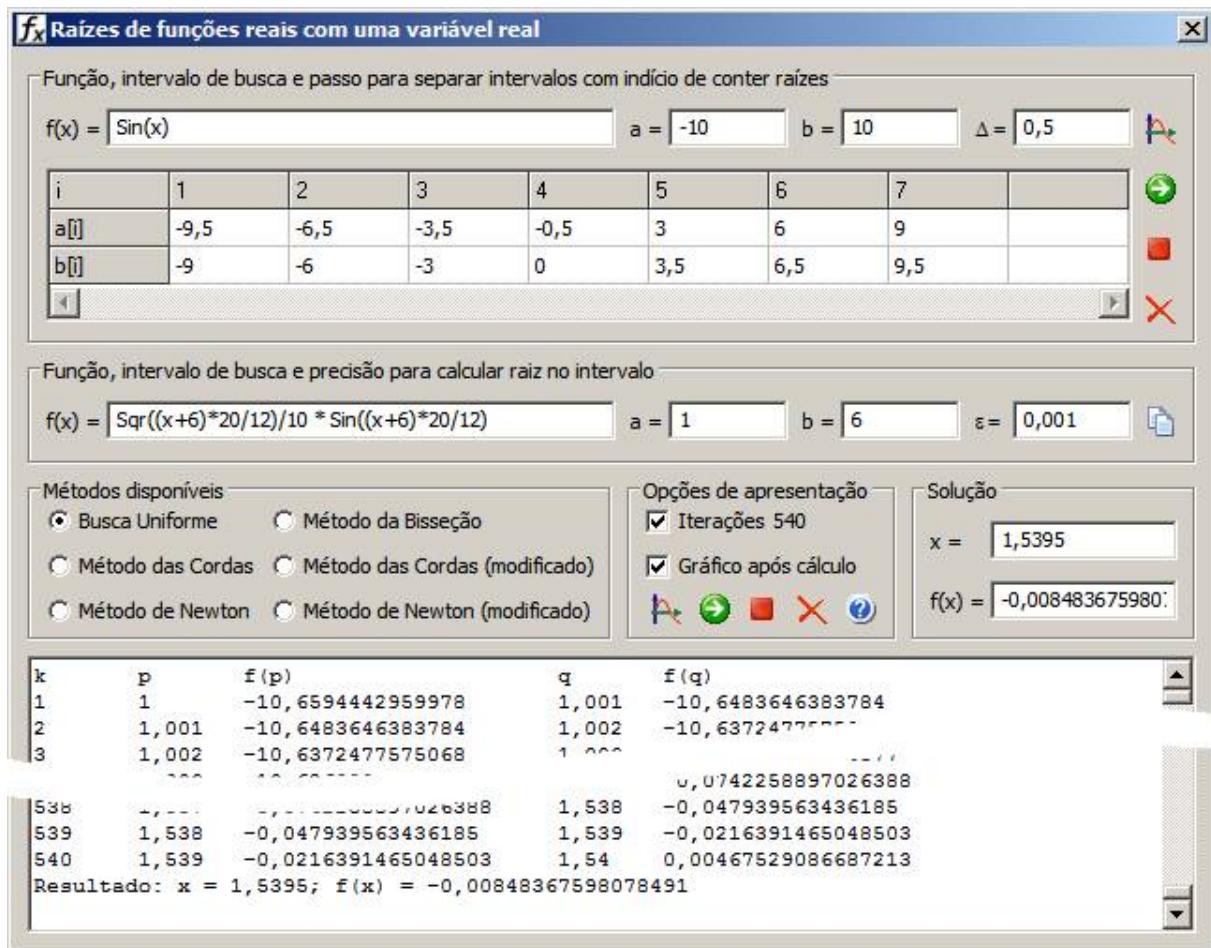
Assim, um possível algoritmo simplificado para o método, calculando a derivada a cada 5 iterações e prevendo o limite de iterações, é como segue.

```
Dados f, a, b, ε, maxI t
k = 0
p = a // ou p = b ou p = um ponto do intervalo [a, b]
Repetir
    se resto da divisão k/5 = 0 então // k mod 5
        derivada = f'(p) // se f'(x) = 0, reinicie com outro p ou ...
    k = k + 1
    q = p
    p = p - f(p)/derivada
Até que |p-q| < ε ou k = maxI t
```

Chegou a hora de você, amado leitor escrever o algoritmo para este método e após o algoritmo, evidentemente, fazer o correspondente programa.

## Sugestões para seus programas

Um simpático programa, para resolver  $f(x) = 0$  com todos os métodos descritos, poderia ter a cara que segue.



Com este programa é possível:

- apresentar o gráfico da função no intervalo  $[a, b]$ ;
- separar todas as raízes do intervalo  $[a, b]$ ;
- calcular a raiz de cada intervalo com cada um dos métodos;
- apresentar uma tabela com os cálculos a cada iteração;
- apresentar o gráfico da função e sua raiz;
- interromper o programa caso não converja em tempo suportável pelo usuário.

### Instruções claras e precisas sobre trabalhos computacionais para avaliação

Todas as instruções necessárias estão descritas no mesmo item, do capítulo anterior. Copiar e colar é muito fácil, mas não é nada elegante, agradável, simpático ou conveniente repetir tudo aquilo que já está escrito no capítulo anterior. Além disto, se você já entregou o primeiro trabalho, já sabe como deve proceder. Caso contrário, veja as instruções no capítulo anterior. E antes de entregar, leia as instruções em <http://sacomam.dco.fc.unesp.br/mnc/pet/>.

Se você optar em fazer um programa que tenha a aparência da figura apresentadas ou similar, pode-

rá continuar lendo para ter algumas explicações que poderão ser úteis para você.

### Um exemplo de programa

Minha ajuda será *apenas na fabricação da interface*. O resto é com você. Depois de pronto, limpe todo lixo gerado durante a elaboração do projeto, compacte com Zip, Rar ou qualquer outro compactador, com o sugestivo nome T2-RA-V1.ext, onde RA é o número de seu Registro Acadêmico, ext é a extensão fornecida pelo seu compactador, V1 é a primeira versão e entregue via endereço eletrônico de recepção que você já sabe qual é. Não sabe qual é? Então leia tudo sobre o assunto no endereço <http://sacoman.dco.fc.unesp.br/mnc/pet/>.

Então, mãos a obra.

Crie um diretório apropriado para desenvolver seu projeto (e.g.: X:\MNC-Trabalhos\T2). Organize seu ambiente de trabalho e crie subdiretórios necessários para o projeto, como, por exemplo, um diretório chamado Figuras, para as figuras dos botões e ícone e outro chamado Exemplo, para o exemplo executável que está à sua disposição.

Copie de <http://sacoman.dco.fc.unesp.br/mnc/material/> o arquivo M2-Raízes - R1.rar e descompacte. Mova Raizes.exe e Interpretador.dll para o diretório exemplo, para executar quando quiser. Mantenha uma cópia do Interpretador.dll no diretório principal, onde desenvolverá o projeto. Mova as figuras para o diretório Figuras.

Inicie seu ambiente de projeto, no meu caso, Lazarus, ajeite as janelas da forma como achar conveniente e salve o projeto com o nome Raizes.lpr e a unidade principal com o nome Principal ou Unit1. Pressione o botão Nova Fórmula, e salve novamente, dando o nome Unit2 ou Grafico para esta nova unidade.

Volte à fórmula principal e logo após a declaração **implementation**, escreva as linhas que seguem.

```
uses
  Uni t2, Windows, LCLIntf;           // para usar rotinas GetSystemMetrics e OpenDocument
```

Logo após a declaração **public**, escreva as linhas que seguem.

```
public
  { public declarations }
  function VerificaDados(k: Byte): Boolean;
  procedure DesenhaGrafico(k: Byte);
  procedure DesenhaRaiz;
  procedure Derivada;
  procedure UniformeSeparar;
  procedure Uniforme;
  procedure Bissecao;
  procedure Cordas;
  procedure CordasModificado;
  procedure Newton;
  procedure NewtonModificado;
end;
```

Se estas rotinas forem implementadas e não forem definidas como membros da fórmula, sempre que a rotina utilizar um componente da fórmula é necessário escrever, por exemplo, Form1.Edi t1.Text. Definindo como rotinas da fórmula, elas recebem a visibilidades de todos os componentes da fórmula

sem necessidade de explicitar. Então, as rotinas poderão utilizar, por exemplo, Edi t1.Text.

Pressione F12 para ver a cara da fórmula, atribua Height=480, Width=640, Caption = Raízes de funções reais com uma variável real, BorderStyle=bsSingle, BorderIcons com biMaximize=False e biMinimize=False, Font com Name=Tahoma e Size=8, Position=poDesktopCenter e ShowHint=True.

No menu Projeto, selecione Opções de Projeto e atribua Título=Raízes de funções reais com uma variável real e carregue o ícone MNC-Raízes.ico.

Lance na fórmula 5 componentes TGroupBox e atribua as propriedades que seguem.

Componente	Caption	Left	Top	Height	Width
GroupBox1	Função, intervalo de busca e passo para separar intervalos com índice de conter raízes	8	8	150	624
GroupBox2	Função, intervalo de busca e precisão para calcular raiz no intervalo	8	164	58	624
GroupBox3	Métodos disponíveis	8	228	88	312
GroupBox4	Opções de apresentação	320	228	88	144
GroupBox5	Solução	480	228	88	152

Lance, no GroupBox1, alguns componentes como segue.

Componente	Caption	Left	Top	Height	Width	Text	Flat	Glyph	ColCount	RowCount
Label1	f(x) =	8	12							
Label2	a =	320	12							
Label3	b =	412	12							
Label4	$\Delta$ =	504	12							
Edit1		40	8		272	(remova)				
Edit2		340	8		64	(remova)				
Edit3		432	8		64	(remova)				
Edit4		524	8		64	(remova)				
SpeedButton1		592	8				True	raiz.bmp		
SpeedButton2		592	40				True	forward.gif		
SpeedButton3		592	72				True	stop2.gif		
SpeedButton4		592	104				True	delete.gif		
StringGrid1		8	40	81	580				9	3

Para StringGrid1 atribua ScrollBars=ssHorizontal.

Lance, no GroupBox2, alguns componentes como segue.

Componente	Caption	Left	Top	Width	Text	Flat	Glyph
Label5	f(x) =	8	12				
Label6	a =	320	12				
Label7	b =	412	12				
Label8	$\varepsilon$ =	504	12				
Edit5		40	8	272	(remova)		
Edit6		340	8	64	(remova)		
Edit7		432	8	64	(remova)		
Edit8		524	8	64	(remova)		
SpeedButton5		592	8			True	copy.gif

Lance, no GroupBox3, alguns componentes como segue.

Componente	Caption	Left	Top
RadioButton1	Busca Uniforme	8	0
RadioButton2	Método da Bisseção	128	0
RadioButton3	Método das Cordas	8	24
RadioButton4	Método das Cordas (modificado)	128	24
RadioButton5	Método de Newton	8	48
RadioButton6	Método de Newton (modificado)	128	48

Lance, no GroupBox4, alguns componentes como segue.

Componente	Caption	Left	Top	Flat	Glyph
CheckBox1	Iterações	8	0		
CheckBox2	Gráfico após cálculo	8	24		
Label9	0	76	1		
SpeedButton6		8	44	True	raiz.bmp
SpeedButton7		34	44	True	forward.gif
SpeedButton8		60	44	True	stop2.gif
SpeedButton9		86	44	True	delete.gif
SpeedButton10		112	44	True	help2.gif

Lance, no GroupBox5, alguns componentes como segue.

Componente	Caption	Left	Top	Width	Text
Label10	x =	8	12		
Label11	f(x) =	8	44		
Edit9		40	8	100	(remova)
Edit10		40	40	100	(remova)

Lance, na fôrma, um componente TMemo e atribua Left=8, Top=324, Height=148, Width=624, ScrollBars=ssAutoVertical e, para Font, Name=Courier New e Pitch=fpFixed.

Salve, compile e execute, para verificar se a fôrma se parece com fôrma tipo diálogo, mas com ícone. Se tudo funcionar, continue.

Mova para Unit2, pressione F12 para ver a cara da fôrma, atribua Height=300, Width=450, Caption=Raízes de funções reais com uma variável real, Font com Name=Tahoma e Size=8 e Position=poDesktopCenter.

Lance, na fôrma, um componente TChart e atribua Align=alClient, Legend Visible=True e, para Title, Visible=True e Font Color=clBlack.

Cutuque o botão direito do mouse, selecione Edit series, pressione o botão Add, selecione Line series, para adicionar Chart1LineSeries1, atribua SeriesColor=clRed e Title=Função. No editor de series pressione o botão Add, selecione Line series, para adicionar Chart1LineSeries2, atribua ShowPoints=True, Pointer Style=psCircle, Pointer Brush Color=clGreen e Title=Raiz.

Não salve. Espere a CPFL interromper a energia e perca tudo que fez. **Salve essa @#\$% de programa.**

Agora só falta criar as rotinas que serão disparadas pelos eventos, recheá-las convenientemente e estará tudo pronto. Mas antes disto, defina as variáveis necessárias e a função que será chamada de programa externo.

Mova para a Unit1 e logo após {\$R \*.lfm} escreva as linhas que seguem.

```
{$R *.lfm}
var
  Metodo: Byte;
  f: string;
  a, b, Delta, Epsilon: Extended;
  //p, q, f_p, f_q: Extended;           // pode ser que você use em seu programa
  Raiz, f_Raiz: Extended;
  //Sinal1, Sinal2, Iteracao: Integer;  // pode ser que você use em seu programa
  //EncontrouRaiz: Boolean;            // pode ser que você use em seu programa
  //ResultadoFxR1: Word;              // pode ser que você use em seu programa
  VerIteracoes, VerGrafico, Interrompe: Boolean;
function FxR1(f: String; x: Extended; var y: Extended): Word; stdcall external 'Interpretador.dll';
```

Algumas das variáveis acima estão como comentários, pois não usei em meu programa, mas pode ser que você use no seu. Outras estão como comentários, pois usei como variáveis locais em minhas rotinas, mas pode ser que você queira usar como variáveis globais. Faça como achar melhor.

Cutuque a fórmula e, em eventos, dê um toque duplo em OnShow para criar a rotina FormShow e preencha como segue.

```
procedure TForm1.FormShow(Sender: Tobject);
begin
  // Atribuir tamanhos e posições          { uses Windows para GetSystemMetrics }
  ClientHeight := 324;
  StringGrid1.Height := 62+GetSystemMetrics(SM_CYHSCROLL)+GetSystemMetrics(SM_CYFRAME);
  StringGrid1.Cells[0,0] := 'i';
  StringGrid1.Cells[0,1] := 'a[i]';
  StringGrid1.Cells[0,2] := 'b[i]';
end;
```

Cutuque o RadioButton1 e, em eventos, dê um toque duplo em OnClick para criar a rotina RadioButton1Click e altere seu nome para RadioButtonQualquerClick. Marque o RadioButton2 e, com Shift pressionado, marque de RadioButton3 até RadioButton6. Em eventos, descontine o ComboBox do evento OnClick e selecione a rotina RadioButtonQualquerClick. Desta forma, quando o usuário selecionar qualquer RadioButton, a rotina disparada será a mesma. Isto evita ter que fazer 6 diferentes rotinas para verificar o método selecionado. Preencha esta rotina como segue.

```
procedure TForm1.RadioButtonQualquerClick(Sender: Tobject);
begin
  // Seleciona o Método
  Metodo := 0;
  if RadioButton1.Checked then Metodo := 1;
  if RadioButton2.Checked then Metodo := 2;
  if RadioButton3.Checked then Metodo := 3;
  if RadioButton4.Checked then Metodo := 4;
  if RadioButton5.Checked then Metodo := 5;
  if RadioButton6.Checked then Metodo := 6;
end;
```

Cutuque o CheckBox1 e, em eventos, dê um toque duplo em OnClick para criar a rotina CheckBox1Click. Faça o mesmo com CheckBox2 para criar a rotina CheckBox2Click e preencha como segue.

```

procedure TForm1.CheckBox1Click(Sender: TObject);
begin
  // Apresentar ou não as iterações
  VerIteracoes := not VerIteracoes;
end;

procedure TForm1.CheckBox2Click(Sender: TObject);
begin
  // Apresentar ou não o gráfico
  VerGrafico := not VerGrafico;
end;

```

Crie as rotinas que atenderão ao evento OnClick para cada SpeedButton, dando um toque duplo em cada um, pela ordem, ou seja, de SpeedButton1 até SpeedButton10. Respire fundo e preencha como segue.

```

procedure TForm1.SpeedButton1Click(Sender: TObject);
begin
  // Existem dados necessários para desenhar gráfico ?
  // Verifica entrada de dados (f, a, b)
  if not VerificaDados(1) then
    Exit;
  Form2.Chart1LineSeries2.Clear;
  DesenhaGrafico(1);
  Form2.Show;
end;

procedure TForm1.SpeedButton2Click(Sender: TObject);
begin
  // Existem dados necessários para separar as raízes?
  // Verifica entrada de dados (f, a, b, Delta)
  if not VerificaDados(2) then
    Exit;
  UniformeSepara;
end;

procedure TForm1.SpeedButton3Click(Sender: TObject);
begin
  // Botão do pânico
  Interrompe := True;
end;

procedure TForm1.SpeedButton4Click(Sender: TObject);
var
  i: Integer;
begin
  Edit1.Text := '';
  Edit2.Text := '';
  Edit3.Text := '';
  Edit4.Text := '';
  with StringGrid1 do
  begin
    ColCount := 9;
    for i := 1 to ColCount-1 do
    begin
      Cells[i, 0] := '';
      Cells[i, 1] := '';
      Cells[i, 2] := '';
    end;
  end;
end;

procedure TForm1.SpeedButton5Click(Sender: TObject);
begin
  Edit5.Text := Edit1.Text;
end;

```

```

procedure TForm1.SpeedButton6Click(Sender: TObject);
begin
  // Existem dados necessários para desenhar gráfico ?
  // Verifica entrada de dados (f, a, b)
  if not VerificaDados(3) then
    Exit;
  Form2.Chart1LineSerieses2.Clear;
  DesenhaGrafico(2);
  Form2.Show;
end;

procedure TForm1.SpeedButton7Click(Sender: TObject);
begin
  // Calcular a raiz
  Interrompe := False;      // Utilizado para controlar o Botão do pânico
  // Limpa resposta anterior
  Edit9.Text := '';
  Edit10.Text := '';
  // Verifica entrada de dados
  // Existem dados necessários para calcular a raiz ?
  if not VerificaDados(4) then
    Exit;
  ClientHeight := 324;
  // Existe Método selecionado ?                                // Este trecho não é necessário se um
  // dos botões já estiver marcado                            // durante o desenvolvimento do programa
  // Por exemplo, atribuir, no                                // Inspetor de Objetos,
  // RadioButton1.Checked = True
  if Metodo = 0 then
  begin
    ShowMessage('É necessário escolher um Método');
    Exit;
  end;
  // Executa método selecionado - se você prefere utilizar case ou switch, use
  if Metodo = 1 then      //Busca Uniforme
  begin
    UniForme;
    Exit;
  end;
  if Metodo = 2 then      //Método da Bissecção
  begin
    Bissecao;
    Exit;
  end;
  if Metodo = 3 then      //Método das Cordas
  begin
    Cordas;
    Exit;
  end;
  if Metodo = 4 then      //Método das Cordas Modificado
  begin
    CordasModificado;
    Exit;
  end;
  if Metodo = 5 then      //Método de Newton
  begin
    Newton;
    Exit;
  end;
  if Metodo = 6 then      //Método de Newton Modificado
  begin
    NewtonModificado;
    Exit;
  end;
end;

procedure TForm1.SpeedButton8Click(Sender: TObject);
begin
  // Botão do pânico
  Interrompe := True;
end;

```

```

procedure TForm1.SpeedButton9Click(Sender: TObject);
begin
  Edit5.Text := '';
  Edit6.Text := '';
  Edit7.Text := '';
  Edit8.Text := '';
  Edit9.Text := '';
  Edit10.Text := '';
  Label9.Caption := '0';
  ClientHeight := 324;
end;

procedure TForm1.SpeedButton10Click(Sender: TObject);
begin
  // Apresentar arquivo de ajuda { uses LCLIntf para OpenDocument }
  if not OpenDocument('Raízes.chm') then
    ShowMessage('Arquivo de Ajuda não foi encontrado.');
end;

```

Para copiar um intervalo  $[a, b]$  da grade de separação das raízes para os campos de edição de  $a$  e  $b$  da busca da raiz, cutoque a StringGrid1 e, em eventos, dê um toque duplo em OnMouseDown, para criar a rotina StringGrid1MouseDown. Preencha como segue.

```

procedure TForm1.StringGrid1MouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
var
  Cell1Col, Cell1Row: Integer;
begin
  StringGrid1.MouseToCell(X, Y, Cell1Col, Cell1Row);
  if Cell1Col = 0 then
    Exit;
  Edit6.Text := StringGrid1.Cells[Cell1Col, 1];
  Edit7.Text := StringGrid1.Cells[Cell1Col, 2];
end;

```

Com isto, o usuário poderá digitar a função e o intervalo de busca da raiz ou, simplesmente copiar os dados da separação para os dados de cálculo da raiz. Com essa dica fantástica, eu acho que eu ainda mereço aquele microcomputador que deveria ter recebido em 9 de abril. E você, também acha? Deixe isso prá lá.

Salve, compile e veja o compilador reclamar. Se você copiou as rotinas prestando atenção no que fazia, viu que tem uma série de rotinas para verificar a entrada de dados, desenhar o gráfico, desenhar a raiz, calcular a derivada, separar as raízes e calcular as raízes por 6 métodos diferentes. Todas já foram definidas na cláusula public. Só falta escrever as rotinas. Tudo isto é trabalho seu, mas eu escrevi as rotinas para verificar os dados, separar as raízes, desenhar o gráfico, calcular a raiz pelo Método da Busca Uniforme e desenhar a raiz. Se você fizer sem copiar, a nota será só sua. Se copiar, me avise e eu ficarei com metade da nota. De qualquer forma, para você ter alguma coisa para fazer sozinho, sobraram as outras rotinas para calcular as raízes.

Logo abaixo da linha onde foi definida a rotina FxR1 externa ao programa e, antes de qualquer outra rotina, escreva o que segue. Respire fundo, novamente, e vá em frente.

```

function FxR1(f: String; x: Extended; var y: Extended): Word; stdcall; external 'Interpretador.dll';
function TForm1.VerificaDados(k: Byte): Boolean;
begin
  // Parte superior da fórmula (k = 1 => Gráfico e k = 2 => Separação)
  if (k = 1) then      // Verifica entrada de dados para o gráfico (f, a, b)
  begin
    Result := False;
  end;

```

```

f := Trim(Edit1.Text);
if (f = '') then
begin
  ShowMessage('Informe uma função.');
  Edit1.SetFocus;
  Exit;
end;
try
  a := StrToFloat(Edit2.Text);
except
  ShowMessage('Valor de a incorreto');
  Edit2.SetFocus;
  Exit;
end;
try
  b := StrToFloat(Edit3.Text);
except
  ShowMessage('Valor de b incorreto');
  Edit3.SetFocus;
  Exit;
end;
if b <= a then
begin
  ShowMessage('Valor de b deve ser maior que valor de a.');
  Edit3.SetFocus;
  Exit;
end;
// Se passou por tudo e k = 1 => f, a, b estão corretos
Result := True;
Exit;
end;

if k = 2 then          // Verificar f, a, b anteriores e Delta
begin
  Result := False;
  if not VerificaDados(1) then
    Exit;
  try
    Delta := StrToFloat(Edit4.Text);
  except
    ShowMessage('Valor de Delta incorreto');
    Edit4.SetFocus;
    Exit;
  end;
  if ((Delta - (b-a)/1000) < 0) or ((Delta - (b-a)/20) > 1E-10) then // if (Delta < (b-a)/100) or (Delta > (b-a)/20) then
begin
    ShowMessage('0 menor valor admissível para Delta é (b-a)/1000 (1000 avaliações da função). '+#10+
                '0 maior valor admissível para Delta é (b-a)/20 (20 avaliações da função).');
    Edit4.SetFocus;
    Exit;
  end;
  // Se passou por tudo e k = 2 => f, a, b, Delta estão corretos
  Result := True;
end;

// Parte inferior da fórmula (k = 3 => Gráfico e k = 4 => Raiz)
if (k = 3) then        // Verifica entrada de dados para o gráfico (f, a, b)
begin
  Result := False;
  f := Trim(Edit5.Text);
  if (f = '') then
begin
  ShowMessage('Informe uma função.');
  Edit5.SetFocus;
  Exit;
end;
try
  a := StrToFloat(Edit6.Text);
except

```

```

ShowMessage('Valor de a incorreto');
Edi t6. SetFocus;
Exit t;
end;
try
  b := StrToFloat(Edit7.Text);
except
  ShowMessage('Valor de b incorreto');
  Edit7. SetFocus;
  Exit t;
end;
if b <= a then
begin
  ShowMessage('Valor de b deve ser maior que valor de a.');
  Edit7. SetFocus;
  Exit t;
end;
// Se passou por tudo e k = 3 => f, a, b estão corretos
Result := True;
Exit t;
end;

if k = 4 then          // Verificar f, a, b anteriores e Epsilon
begin
  Result := False;
  if not VerificaDados(3) then
    Exit t;
  try
    Epsilon := StrToFloat(Edit8.Text);
  except
    ShowMessage('Valor de Epsilon incorreto');
    Edit8. SetFocus;
    Exit t;
  end;
  if (Epsilon - (b-a)/20) > 1E-10 then // if Epsilon > (b-a)/20 then
begin
  ShowMessage('O máximo admissível para Epsilon é (b-a)/20.');
  Edit4. SetFocus;
  Exit t;
end;
// Se passou por tudo e k = 4 => f, a, b, Epsilon estão corretos
Result := True;
end;
end;

procedure TForm1.DesenhaGrafico(k: Byte);
var
  i: Integer;
  x, y: Extended;
begin
  // Desenhando o gráfico de uma função
  // Existem dados necessários para desenhar gráfico ?
  // Gráfico da parte superior (separação das raízes)
  if k = 1 then
    if not VerificaDados(1) then
      Exit t;
  // Gráfico da parte inferior (cálcula de uma raiz)
  if k = 2 then
    if not VerificaDados(3) then
      Exit t;
  with Form2 do
begin
  Chart1LineSeries1.Clear;
  Chart1.Title.Text[0] := 'Gráfico da função ' + f;
  Delta := (b-a)/1000;
  for i := 0 to 1000 do
begin
  try

```

```

        x := a+i*Del ta;
        FxR1(f, x, y);
        Chart1LineSeries1.AddXY(x, y, '');
    except
        //Exit;      // ou interrompe ou deixa continuar até completar o gráfico
    end;
end;
end;
end;

procedure TForm1.DesenhaRaiz;
begin
    // Desenhando a raiz de uma função
    with Form2 do
    begin
        Chart1LineSeries2.Clear;
        Chart1LineSeries2.AddXY(Rai z, f_Rai z, '');
    end;
end;

procedure TForm1.Derivada;
begin
    // Calcular derivada requerida nos Métodos de Newton (pode ser procedure ou function)
end;

procedure TForm1.UniformeSepara;
var
    i, j: Integer;
    p, q, f_p, f_q: Extended;
    Erro: Word;
begin
    // Método da Busca Uniforme para separar raízes
    Interrompe := False;           // Utilizado para controlar o Botão do pânico
    with StringGrid1 do
    begin
        Col Count := 9;
        for i := 1 to Col Count-1 do
        begin
            Cells[i, 0] := '';
            Cells[i, 1] := '';
            Cells[i, 2] := '';
        end;
    end;
    i := 0;
    j := 1;
    p := a;
    FxR1(f, p, f_p);
    q := p+Del ta;
    FxR1(f, q, f_q);
    while (p <= b) do
    begin
        Application.ProcessMessages;
        if Interrompe then
        begin
            ShowMessage('Usuário cancelou.');
            Exit;
        end;
        if f_p * f_q <= 0 then
        begin
            Inc(i);
            with StringGrid1 do
            begin
                if Col Count < i+1 then
                    Col Count := i+1;
                Cells[i, 0] := IntToStr(i);
                Cells[i, 1] := FloatToStr(p);
                Cells[i, 2] := FloatToStr(q);
            end;
            if f_q = 0 then

```

```

begin
  Inc(j);
  q := a+j *Delta;
  FxR1(f, q, f_q);
end;
end;
p := q;
f_p := f_q;
Inc(j);
q := a+j *Delta; // q := p+Delta;
try
  Erro := FxR1(f, q, f_q);
except
  ShowMessage(IntToStr(Erro) + ' - ' + FloatToStr(q) + ' - ' + FloatToStr(f_q));
end;
end;
if i = 0 then
begin
  ShowMessage('Não foram encontradas regiões com índice de conter raízes.');
  Exit;
end;
end;

procedure TForm1.Uniforme;
var
  k: Integer;
  p, q, f_p, f_q: Extended;
  Erro: Word;
begin
  // Método da Busca Uniforme para calcular raízes
  Interrompe := False; // Utilizado para controlar o Botão do pânico
  Memo1.Clear;
  Memo1.Lines.Add('k' +#09+ 'p' +#09+ 'f(p)' +#09#09#09+ 'q' +#09+ 'f(q)' );
  k := 1;
  p := a;
  FxR1(f, p, f_p);
  q := p+Epsilon;
  FxR1(f, q, f_q);
  Memo1.Lines.Add(IntToStr(k)+#09+FloatToStr(p)+#09+FloatToStr(f_p)+#09+
    FloatToStr(q)+#09+FloatToStr(f_q));
  while (p <= b) and (f_p * f_q > 0) do
  begin
    Application.ProcessMessages;
    if Interrompe then
    begin
      ShowMessage('Usuário cancelou.');
      Exit;
    end;
    Inc(k);
    p := q;
    f_p := f_q;
    q := p+Epsilon;
    FxR1(f, q, f_q);
    Label9.Caption := IntToStr(k); // Como comentário, não exibe a cada iteração
    Memo1.Lines.Add(IntToStr(k)+#09+FloatToStr(p)+#09+FloatToStr(f_p)+#09+
      FloatToStr(q)+#09+FloatToStr(f_q));
  end;
  Raiz := (p+q)/2;
  FxR1(f, Raiz, f_Raiz);
  Memo1.Lines.Add('Resultado: x = ' +FloatToStr(Raiz)+'; f(x) = ' +FloatToStr(f_Raiz));
  Edit9.Text := FloatToStr(Raiz);
  Edit10.Text := FloatToStr(f_Raiz);
  //Label9.Caption := IntToStr(k); // Sem comentário, exibe ao terminar
  if VerIteracoes then
    ClientHeight := 480;
  if VerGrafico then
  begin
    // Calcula e apresenta o gráfico da função e a raiz
    DesenhaGrafico(2);
  end;
end;

```

```

    DesenhaRai z;
    Form2. Show;
end;
end;

procedure TForm1. Bissicao;
begin
  // Método da Bisseção
  ShowMessage('Implementar Bisseção');
end;

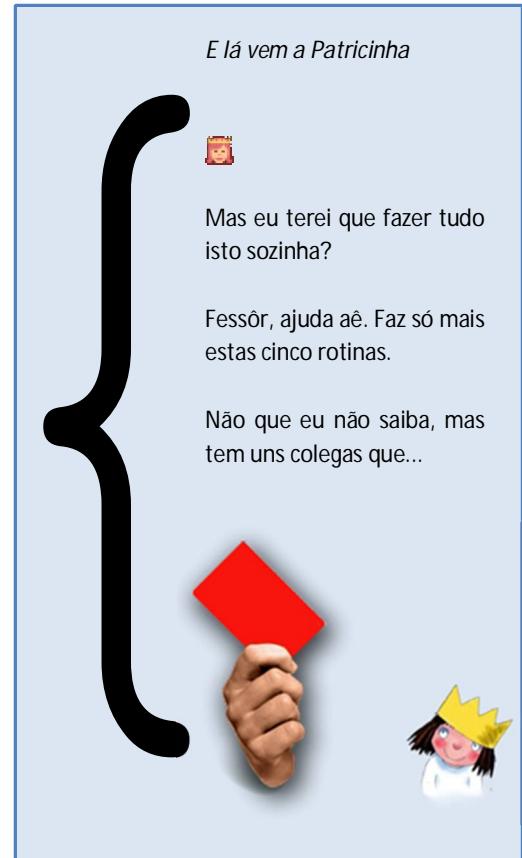
procedure TForm1. Cordas;
begin
  // Método das Cordas
  ShowMessage('Implementar Cordas');
end;

procedure TForm1. CordasModificado;
begin
  // Método das Cordas Modificado
  ShowMessage('Implementar Cordas modificado');
end;

procedure TForm1. Newton;
begin
  // Método de Newton
  ShowMessage('Implementar Newton');
end;

procedure TForm1. NewtonModificado;
begin
  // Método de Newton Modificado
  ShowMessage('Implementar Newton Modificado');
end;

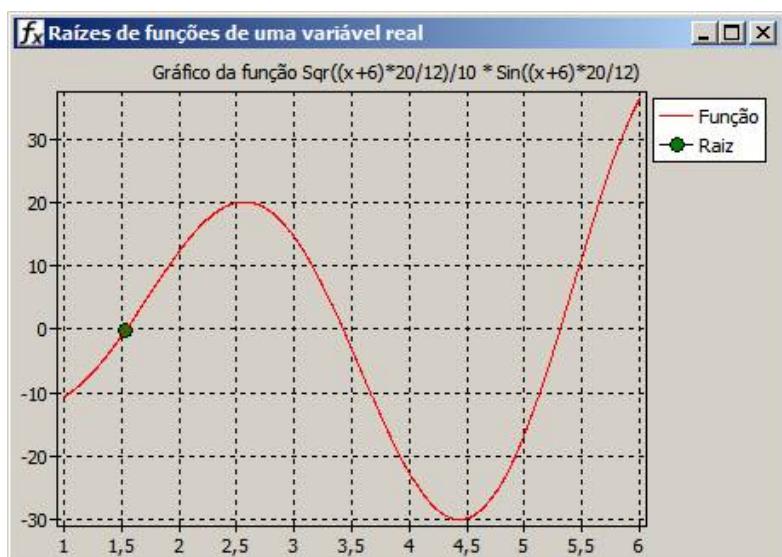
```



Salve, compile e veja a maravilha que você criou. Claro que estou imaginando que você já preencheu as rotinas de cálculo de raiz para todos os métodos. Parabéns. Agora é só compactar e entregar.

E se não preencheu as rotinas, respire fundo, pois agora só falta você fazer a sua parte que é programar os algoritmos para solução do problema de encontrar as raízes. Só isto. Divirta-se.

Caso queira, pode salpicar Hint para vários componentes. No início, foi atribuído ShowHint=True para a fôrma e todos os componente herdam essa propriedade, já que a propriedade ParentShowHint é True para todos eles. Veja exemplos de Hint executando o exemplo que está disponível para você e crie os seus. Ou copie.



Já que sobrou espaço, segue uma fotografia da fôrma do gráfico.

## 9- Sistemas de equações lineares

Vários livros de MNC apresentam, a título de motivação, problemas de determinação de quantidades de alimentos para fornecer determinadas quantidades de vitaminas, horas de máquinas para produzir determinadas quantidades de produtos e outros problemas similares. E sempre com constantes onde o sistema só tem uma solução. Isto não existe. Estes problemas, na prática, têm mais de uma solução e devem ser resolvidos com técnicas de otimização onde se busca a melhor solução possível para problemas que admitem várias soluções. Além disto, problemas de motivação devem conter uma dose de terrorismo como estes que gosto de apresentar aos meus amados alunos.

### Queimando a mão com a chapa quente do Tião

Seu gentil chefe, feliz e sorridente, foi até sua sala, levou um cafezinho e lhe entregou uma folha com um sistema de equações lineares com 6 equações e 6 variáveis para serem determinadas. Depois do cafezinho, ele perguntou se você poderia resolver o problema, sem nenhuma pressa e lhe entregar, se possível, corretamente. Agora deixe de sonhar.



Seu chefe, carrancudo, deu um berro lhe chamando para ir até sua sala (sua dele, claro), ordenou que aproveitasse a ida para levar um café e lhe passou um problema para determinar a temperatura em 6 pontos de uma chapa metálica. Adicionalmente lhe informou que a temperatura se distribui uniformemente e que eram conhecidas as temperaturas em 10 pontos conhecidos da borda da chapa. Rabiscou o problema e lhe entregou o rabisco. Terminou dizendo: quero a solução em 15 minutos. E antes que me esqueça, o nome do chefe é Tião Porrada que, além de nome, traduz a personalidade do indivíduo. Então, ao trabalho.

	$20^{\circ}\text{C}$	$20^{\circ}\text{C}$	$20^{\circ}\text{C}$	
$10^{\circ}\text{C}$	$T_1$	$T_2$	$T_3$	
				$40^{\circ}\text{C}$
$10^{\circ}\text{C}$	$T_4$	$T_5$	$T_6$	$40^{\circ}\text{C}$
	$20^{\circ}\text{C}$	$20^{\circ}\text{C}$	$20^{\circ}\text{C}$	

Com as informações dadas, você começou escrevendo  $T_1 = (10+20+T_2+T_4)/4$ , então  $4.T_1 - T_2 - T_4 = 30$ .

E assim, fez o mesmo para os outros pontos e concluiu que deveria resolver o sistema de equações lineares que segue.

$$\begin{cases} 4.T_1 - 1.T_2 & -1.T_4 & = 30 \\ -1.T_1 & 4.T_2 - 1.T_3 & -1.T_5 & = 20 \\ & -1.T_2 & 4.T_3 & -1.T_6 = 60 \\ -1.T_1 & & 4.T_4 - 1.T_5 & = 30 \\ & -1.T_2 & -1.T_4 & 4.T_5 - 1.T_6 = 20 \\ & & -1.T_3 & -1.T_5 & 4.T_6 = 60 \end{cases}$$

Você gastou 12 dos 15 minutos que tinha para modelar o problema e, a seguir, começou a chorar. Como resolver o sistema em apenas 3 minutos? Então, lembre-se que você deve aprender os métodos e fazer um excelente programa que, se estivesse disponível, sobraria tempo. Você entregaria ao Tião Porrada esperando uma promoção e ele lhe diria: não fez mais do que a obrigação.

Você perdeu o emprego e merece, ao menos, saber a solução.  $T_1 = T_4 = 17,14^{\circ}\text{C}$ ,  $T_2 = T_5 = 21,43^{\circ}\text{C}$  e  $T_3 = T_6 = 27,14^{\circ}\text{C}$ . Agora que você está supermotivado, vire a página (nos dois sentidos) e continue.

Recordar é viver ou, de volta ao Colegial

### **Equação linear e sistema de equações lineares**

Equação linear com n variáveis é a equação escrita na forma  $a_1.x_1+a_2.x_2+\dots+a_n.x_n = b$ , onde as variáveis são  $x_1, x_2, \dots, x_n$  e  $a_1, a_2, \dots, a_n$  e  $b$  são constantes.

Sistema de equações lineares é um conjunto de m equações lineares escrito na forma que segue.

$$\begin{cases} a_{11}.x_1 + a_{12}.x_2 + a_{13}.x_3 + \dots + a_{1n}.x_n = b_1 \\ a_{21}.x_1 + a_{22}.x_2 + a_{23}.x_3 + \dots + a_{2n}.x_n = b_2, \text{ onde } a_{ij} \text{ e } b_i \text{ são constantes, com } i = 1, \dots, m \text{ e } j = 1, \dots, n. \\ \vdots \\ a_{m1}.x_1 + a_{m2}.x_2 + a_{m3}.x_3 + \dots + a_{mn}.x_n = b_m \end{cases}$$

O sistema pode ser escrito na forma matricial, representado por  $A.x = b$ , como segue.

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2n} \\ \vdots & & & & \\ a_{m1} & a_{m2} & a_{m3} & \dots & a_{mn} \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{pmatrix}, \text{ onde } A \in \mathbb{R}^{m \times n}, x \in \mathbb{R}^n \text{ e } b \in \mathbb{R}^m.$$

A solução do sistema  $A.x = b$  é  $x = (x_1, x_2, x_3, \dots, x_n)^T$  que satisfaz simultaneamente todas as equações, ou seja,  $x_1, x_2, x_3, \dots, x_n$  são as raízes do sistema.

### **Classificação de um sistema de equações lineares quanto à solução**

Sistema possível ou compatível

- admite solução

Sistema possível e determinado

- possui uma única solução
- a matriz deve ser não singular, ou seja, seu determinante deve ser diferente de zero
- se  $b$  for nulo, a solução será trivial, ou seja,  $x$  será nulo

Sistema possível e indeterminado

- possui infinitas soluções
- a matriz é singular, ou seja, seu determinante é nulo
- $b$  é nulo ou múltiplo de uma das colunas de  $A$

Sistema impossível ou incompatível

- não possui solução
- a matriz é singular, ou seja, seu determinante é nulo
- $b$  não é nulo nem múltiplo de uma das colunas de  $A$

Sistema homogêneo

- $A.x = b$  é homogêneo se  $b$  é nulo
- um sistema homogêneo é consistente, uma vez que  $x$  nulo sempre será solução do sistema.

## Classificação da matriz A de um sistema de equações lineares

Matriz transposta

- para  $A \in R^{n \times n}$  com coeficientes  $a_{ij}$ , a matriz transposta  $A^T$  tem seus coeficientes  $\bar{a}_{ij} = a_{ji}$

Matriz simétrica

- a matriz  $A \in R^{n \times n}$  é simétrica se seus coeficientes  $a_{ij}$  forem tais que  $a_{ij} = a_{ji}$

Matriz triangular superior

- a matriz  $A \in R^{n \times n}$  é chamada triangular superior se seus elementos abaixo da diagonal principal forem nulos, ou seja,  $a_{ij} = 0$  para  $i > j$  com  $i = 1, \dots, n$  e  $j = 1, \dots, n$

Matriz triangular inferior

- a matriz  $A \in R^{n \times n}$  é chamada triangular inferior se seus elementos acima da diagonal principal forem nulos, ou seja,  $a_{ij} = 0$  para  $i < j$  com  $i = 1, \dots, n$  e  $j = 1, \dots, n$

Matriz diagonal

- a matriz  $A \in R^{n \times n}$  é chamada diagonal se seus elementos acima e abaixo da diagonal principal forem nulos, ou seja,  $a_{ij} = 0$  para  $i \neq j$  com  $i = 1, \dots, n$  e  $j = 1, \dots, n$

Sistemas com matriz diagonal tem solução imediata, ou seja,  $x_i = b_i/a_{ii}$ ,  $i = 1, \dots, n$

Sistemas com triangular superior podem ser resolvidos facilmente, partindo de  $x_n = b_n/a_{nn}$  e, por substituição, calcular as outras variáveis de baixo para cima.

Sistemas com triangular inferior podem ser resolvidos facilmente, partindo de  $x_1 = b_1/a_{11}$  e, por substituição, calcular as outras variáveis de cima para baixo.

Para se obter sistemas particulares como triangular superior, triangular inferior ou diagonal pode-se utilizar o conceito de sistemas equivalentes.

Sistemas equivalentes

- dois sistemas S1 e S2 são equivalentes se S2 for obtido, a partir de S1, com operações elementares em S1, que são:
  - 1- trocar posição de linha ou coluna do sistema
  - 2- multiplicar uma linha do sistema por um escalar não nulo
  - 3- multiplicar uma linha do sistema por um escalar não nulo e somar com outra linha do sistema
 Com estas operações em S1 obtém-se S2 e, se S2 é equivalente a S1, sua solução é solução de S1.

## Classificação dos métodos para solução de sistemas de equações lineares

Métodos diretos

- fornecem solução exata para o sistema, caso exista solução, com um número finito de operações

Métodos iterativos

- fornecem solução aproximada para o sistema, caso exista solução, a partir de uma aproximação

inicial, após um número de iterações onde a distância entre a solução corrente e a anterior seja menor que uma tolerância dada e a convergência é garantida sob certas condições

## Solução de sistemas de equações lineares por Métodos Diretos

Um sistema de equações lineares de dimensão  $n \times n$  e que seja possível e determinado, tem como solução  $\mathbf{x} = \mathbf{A}^{-1} \cdot \mathbf{b}$ . Contudo a inversão da matriz A demanda uma grande quantidade de operações e para determinar a solução, foram desenvolvidos métodos que não exijam a inversão da matriz A. Além disto, se para a solução de algum problema prático, a inversa da matriz A for requerida a inversa poderá ser calculada através de alguns desses métodos, por solução de  $n$  sistemas onde o vetor b, em cada solução, representa cada uma das  $n$  colunas da matriz identidade. Essa técnica é apresentada na seção *Inversão de Matrizes*.

### Sistema triangular superior

Considere o sistema de equações lineares onde a matriz A tenha todos os elementos abaixo da diagonal principal nulos e que  $a_{ii} \neq 0$  para  $i = 1, \dots, n$ .

$$\left\{ \begin{array}{l} a_{11} \cdot x_1 + a_{12} \cdot x_2 + a_{13} \cdot x_3 + \dots + a_{1n} \cdot x_n = b_1 \\ a_{22} \cdot x_2 + a_{23} \cdot x_3 + \dots + a_{2n} \cdot x_n = b_2 \\ \vdots \\ a_{nn} \cdot x_n = b_n \end{array} \right.$$

A solução é obtida fazendo

$$x_n = \frac{b_n}{a_{nn}} \text{ e } x_i = \frac{b_i - \sum_{j=i+1}^n a_{ij} \cdot x_j}{a_{ii}}, \quad i = n-1, \dots, 1$$

### Sistema triangular inferior

Considere o sistema de equações lineares onde a matriz A tenha todos os elementos acima da diagonal principal nulos e que  $a_{ii} \neq 0$  para  $i = 1, \dots, n$ .

$$\left\{ \begin{array}{l} a_{11} \cdot x_1 = b_1 \\ a_{21} \cdot x_1 + a_{22} \cdot x_2 = b_2 \\ a_{n1} \cdot x_1 + a_{n2} \cdot x_2 + a_{n3} \cdot x_3 + \dots + a_{nn} \cdot x_n = b_n \end{array} \right.$$

A solução é obtida fazendo

$$x_1 = \frac{b_1}{a_{11}} \text{ e } x_i = \frac{b_i - \sum_{j=1}^{i-1} a_{ij} \cdot x_j}{a_{ii}}, \quad i = 2, \dots, n$$

### Método da Eliminação de Gauss

Considere o sistema de equações lineares  $\mathbf{A} \cdot \mathbf{x} = \mathbf{b}$ , onde todas as submatrizes principais de A sejam não singulares.

O método transforma o sistema original em um sistema triangular superior, através da aplicação de operação que transforma o sistema original em um sistema equivalente, preservando a solução do sistema original.

A operação é subtrair de uma equação outra equação multiplicada por uma constante não nula.

Considere o sistema que segue com sua matriz A e vetor b ou matriz A aumentada ( $A|b$ ).

$$\begin{cases} a_{11} \cdot x_1 + a_{12} \cdot x_2 + a_{13} \cdot x_3 + \cdots + a_{1n-1} \cdot x_{n-1} + a_{1n} \cdot x_n = b_1 \\ a_{21} \cdot x_1 + a_{22} \cdot x_2 + a_{23} \cdot x_3 + \cdots + a_{2n-1} \cdot x_{n-1} + a_{2n} \cdot x_n = b_2 \\ a_{31} \cdot x_1 + a_{32} \cdot x_2 + a_{33} \cdot x_3 + \cdots + a_{3n-1} \cdot x_{n-1} + a_{3n} \cdot x_n = b_3 \\ \vdots \\ a_{n1} \cdot x_1 + a_{n2} \cdot x_2 + a_{n3} \cdot x_3 + \cdots + a_{nn-1} \cdot x_{n-1} + a_{nn} \cdot x_n = b_n \end{cases}$$

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n-1} & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n-1} & a_{2n} \\ a_{31} & a_{32} & a_{33} & \cdots & a_{3n-1} & a_{3n} \\ \vdots & & & & & \\ a_{n1} & a_{n2} & a_{n3} & \cdots & a_{nn-1} & a_{nn} \end{pmatrix}, \quad b = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_n \end{pmatrix}$$

Para gerar zeros na primeira coluna, para todos os elementos abaixo da diagonal, pode-se proceder como segue, onde  $L_i$  significa i-ésima linha do sistema,  $m_{ij} = a_{ij}/a_{ii}$  é o multiplicador e  $a_{ii}$  é o pivô.

$$m_{21} = a_{21}/a_{11}$$

$$L_2 = L_2 - m_{21} \cdot L_1 \quad a_{21} \text{ se torna nulo e todos os elementos da linha 2 são alterados em } A \text{ e } b$$

$$m_{31} = a_{31}/a_{11}$$

$$L_3 = L_3 - m_{31} \cdot L_1 \quad a_{31} \text{ se torna nulo e todos os elementos da linha 3 são alterados em } A \text{ e } b$$

...

$$m_{n1} = a_{n1}/a_{11}$$

$$L_n = L_n - m_{n1} \cdot L_1 \quad a_{n1} \text{ se torna nulo e todos os elementos da linha } n \text{ são alterados em } A \text{ e } b$$

Com estas operações o sistema fica bom A e b como seguem.

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n-1} & a_{1n} \\ 0 & a_{22} & a_{23} & \cdots & a_{2n-1} & a_{2n} \\ 0 & a_{32} & a_{33} & \cdots & a_{3n-1} & a_{3n} \\ \vdots & & & & & \\ 0 & a_{n2} & a_{n3} & \cdots & a_{nn-1} & a_{nn} \end{pmatrix}, \quad b = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_n \end{pmatrix}$$

onde apenas a primeira linha se mantém inalterada com relação ao sistema original

Para gerar zeros na segunda coluna, para todos os elementos abaixo da diagonal, pode-se proceder como segue.

$$m_{32} = a_{32}/a_{22}$$

$$L_3 = L_3 - m_{32} \cdot L_2 \quad a_{32} \text{ se torna nulo e todos os elementos da linha 3 são alterados em } A \text{ e } b$$

$$m_{42} = a_{42}/a_{22}$$

$$L_4 = L_4 - m_{42} \cdot L_2 \quad a_{42} \text{ se torna nulo e todos os elementos da linha 4 são alterados em } A \text{ e } b$$

...

$$m_{n2} = a_{n2}/a_{22}$$

$$L_n = L_n - m_{n2} \cdot L_2 \quad a_{n2} \text{ se torna nulo e todos os elementos da linha } n \text{ são alterados em } A \text{ e } b$$

Com estas operações o sistema fica bom A e b como seguem.

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n-1} & a_{1n} \\ 0 & a_{22} & a_{23} & \cdots & a_{2n-1} & a_{2n} \\ 0 & 0 & a_{33} & \cdots & a_{3n-1} & a_{3n} \\ \vdots & & & & & \\ 0 & 0 & a_{n3} & \cdots & a_{nn-1} & a_{nn} \end{pmatrix}, \quad b = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_n \end{pmatrix}$$

onde apenas a primeira e a segunda linhas se mantêm inalteradas com relação à iteração anterior

Repete-se o procedimento para as outras colunas, até a coluna  $n-1$  e obtém-se o sistema triangular.

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n-1} & a_{1n} \\ 0 & a_{22} & a_{23} & \cdots & a_{2n-1} & a_{2n} \\ 0 & 0 & a_{33} & \cdots & a_{3n-1} & a_{3n} \\ \vdots & & & & & \\ 0 & 0 & 0 & \cdots & 0 & a_{nn} \end{pmatrix}, b = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_n \end{pmatrix}$$

onde as linhas 1 até  $n-1$  se mantém inalteradas com relação à iteração anterior

Um possível algoritmo pode ser escrito como segue.

```
Para j = 1, ..., n-1 faça
  Para i = j+1, ..., n faça
    m = aij/ajj
    Para k = j, ..., n faça
      aik = aik-m. ajk
    bi = bi-m. bj
```

E a solução, como segue.

```
xn = bn/ann
Para i = n-1, ..., 1 faça
  soma = 0
  Para j = i+1, ..., n
    soma = soma+aij. xj
  xi = (bi-soma)/aii
```

Se for utilizada a forma matricial com matriz  $A$  aumentada<sup>15</sup> ( $A|b$ ), a coluna contendo  $b$  será a coluna  $n+1$  e  $b_i$  será  $a_{in+1}$ . Então o algoritmo ficam como segue.

Um possível algoritmo pode ser escrito como segue.

```
Para j = 1, ..., n-1 faça
  Para i = j+1, ..., n faça
    m = aij/ajj
    Para k = j, ..., n+1 faça
      aik = aik-m. ajk
```

E a solução, como segue.

```
xn = ann+1/ann
Para i = n-1, ..., 1 faça
  soma = 0
  Para j = i+1, ..., n
    soma = soma+aij. xj
  xi = (ain+1-soma)/aii
```

O algoritmo apresentado, com ou sem matriz aumentada, só funciona se a hipótese de que todas as submatrizes principais de  $A$  sejam não singulares. Contudo, se o programa se limitar ao algoritmo apresentado, poderá causar exceção, pois o usuário poderá entrar com dados que não respeitem a hipótese. Então é necessário escrever o programa prevendo exceções e, caso ocorra exceção, informar ao usuário e interromper.

Outra forma, mais elegante, é verificar antes do cálculo de  $m$ , se  $a_{jj}$  é nulo e trocar a linha atual por qualquer outra linha  $p$  abaixo desta e que tenha  $a_{pj}$  não nulo. Mesmo assim, se  $a_{jj}$  for muito pequeno, poderá ocorrer overflow ou, ainda, poderá ser impossível esta troca de linhas. Então, tratar exceções, também é necessário mesmo com esta técnica de troca de linhas.

É uma boa hora para você escrever o seu algoritmo prevendo exceções, com ou sem a troca de linhas sugerida e escrever o correspondente programa.

Aproveite e escreva o algoritmo para transformar o sistema dado em um sistema triangular inferior e o algoritmo de sua solução.

---

<sup>15</sup>N. do A.

Não vejo beleza, graça ou simplificação de nada utilizando a matriz aumentada.

As próximas seções citam a dita cuja, mas a proposta de algoritmo será para  $A$  e  $b$  não agrupados na matriz aumentada.

## Método de Gauss com Pivoteamento Parcial

Considere o sistema que  $A \cdot x = b$ , como do caso anterior, com sua matriz A e vetor b ou matriz A aumentada  $(A | b)$ .

Este método, como o anterior, consiste em transformar o sistema original em um sistema triangular superior, através da aplicação de operação que transforma o sistema original em um sistema equivalente, preservando a solução do sistema original.

A operação é subtrair de uma equação outra equação multiplicada por uma constante não nula.

Contudo, o pivô será o maior elemento em valor absoluto, considerados os elementos da diagonal para baixo, na coluna em questão, ou seja, para a coluna j, pivô =  $\max|a_{rj}|, j \leq r \leq n$ .

No método anterior,  $m = a_{ij}/a_{jj}$  e a operação sobre a linha i é  $a_{ik} = a_{ik} - m \cdot a_{jk}$ . Com a alteração acima descrita, se  $a_{pj} = \max|a_{rj}|, j \leq r \leq n$ , a linha do pivô será p, o multiplicador será  $m = a_{ij}/a_{pj}$  e a operação sobre a linha de A e de b serão  $a_{ik} = a_{ik} - m \cdot a_{pk}$  e  $b_i = b_i - m \cdot b_p$ .

Para manter o algoritmo anterior, com  $m = a_{ij}/a_{jj}$ , basta introduzir a verificação do maior elemento e, se a linha p não for a própria linha j, trocar a linha p com a linha j. Neste caso, o método pode ser chamado de Método de Gauss com Pivoteamento Parcial com Troca de Linhas.

Efetuando a troca da linha p com a linha j,  $a_{jj}$  não será nulo, exceto se todos os elementos da coluna j, da diagonal para baixo, forem nulos. Neste caso o sistema não terá solução com este método.

Caso não se queira trocar a linha p com a linha j, o elemento  $a_{ij}$  que será anulado não poderá ser elemento da linha p, pois não poderá ser o pivô. Então, deve-se verificar o maior valor absoluto como sendo  $a_{pj} = \max|a_{rj}|, j \leq r \leq n, r \neq i$ . Se forem todos nulos, o sistema não terá solução com este método.

Trocar as linhas provoca um tempo computacional maior, mas cria um algoritmo de entendimento mais fácil. Um possível algoritmo pode ser escrito como segue, onde a parte destacada é diferente do algoritmo da seção anterior, ou seja, do Método da Eliminação de Gauss.

Trocando linha e utilizando  $m = a_{ij}/a_{jj}$

Para  $j = 1, \dots, n-1$  faça

**Determinar**  $a_{pj} = \max|a_{rj}|, j \leq r \leq n$   
**Se**  $p \neq j$ , **trocar** linha p com linha j  
 Para  $i = j+1, \dots, n$  faça  
 $m = a_{ij}/a_{jj}$   
 Para  $k = j, \dots, n$  faça  
 $a_{ik} = a_{ik} - m \cdot a_{jk}$   
 $b_i = b_i - m \cdot b_j$

Sem trocar linha e utilizando  $m = a_{ij}/a_{pj}$

Para  $j = 1, \dots, n-1$  faça

Para  $i = j+1, \dots, n$  faça

**Determinar**  $a_{pj} = \max|a_{rj}|, j \leq r \leq n, r \neq i$   
**// O pivô não pode ser**  $a_{ij}$   
 $m = a_{ij}/a_{pj}$   
 Para  $k = j, \dots, n$  faça  
 $a_{ik} = a_{ik} - m \cdot a_{pk}$   
 $b_i = b_i - m \cdot b_p$

Escolha entre utilizar o pivô da linha p ou efetuar a troca de linha e utilizar o pivô da linha j e escreva o seu algoritmo prevendo exceções e o correspondente programa. E lembre-se que a troca de linhas é no sistema de equações e, portanto, no seu algoritmo, ao trocar a linha p com a linha j da matriz A, não se esqueça de trocar  $b_p$  com  $b_j$ .

## Método de Gauss com Pivoteamento Total

Considere o sistema que  $A \cdot x = b$ , como do caso anterior, com sua matriz A e vetor b ou matriz A aumentada  $(A|b)$ .

Este método, como os anteriores, consiste em transformar o sistema original em um sistema triangular superior, através da aplicação de operação que transforma o sistema original em um sistema equivalente, preservando a solução do sistema original.

A operação é subtrair de uma equação outra equação multiplicada por uma constante não nula.

Contudo, o pivô será o maior elemento em valor absoluto, considerados os elementos da submatriz da diagonal para baixo e da diagonal para a direita, na coluna em questão, ou seja, para a coluna j, pivô =  $\max |a_{pq}|, j \leq p \leq n, j \leq q \leq n$ .

Considere o sistema onde se inicia a eliminação de valores abaixo da diagonal da coluna 2, como na figura ao lado. O pivô no Método da Eliminação de Gauss seria  $a_{22}$ .

Considera-se a submatriz a partir da linha 2 e da coluna 2. Suponha que o maior elemento em valor absoluto seja  $a_{34}$ . Neste caso, pode-se efetuar a troca da linha 3 com a linha 2 e da coluna 4 com a coluna 2. A partir desta troca efetua-se a eliminação de todos os elementos abaixo da diagonal.

Pode-se, também, de uma maneira muito mais sofrida, criar uma matriz de apontadores e anotar as trocas sem efetuá-las na matriz e nos vetores. Mas já foi avisado que é uma maneira para quem gosta de sofrer.

Repete-se o procedimento para cada nova coluna onde se inicia a eliminação.

Para simplicidade do algoritmo, como no caso anterior, pode-se efetuar troca de linha e, neste caso, troca de coluna, ou seja, troca-se a linha p com a linha j e a coluna q com a coluna j.

Ao se efetuar troca de colunas, deve-se lembrar de trocar a posição dos correspondentes valores do vetor x, ou seja, devem-se anotar suas novas posições para informar a solução na ordem correta.

Trocar as linhas e as colunas provoca um tempo computacional maior, mas cria um algoritmo de entendimento mais fácil. Contudo, para quem gosta de sofrer, deixo para os amados alunos decidir o que preferem e escreverem seus próprios algoritmos.

Escreva o seu algoritmo prevendo exceções e o correspondente programa.

$$\left( \begin{array}{cccccc} a_{11} & a_{12} & a_{13} & a_{14} & \cdots & a_{1n} \\ 0 & a_{22} & a_{23} & a_{24} & \cdots & a_{2n} \\ 0 & a_{32} & a_{33} & a_{34} & \cdots & a_{3n} \\ 0 & a_{42} & a_{43} & a_{44} & \cdots & a_{4n} \\ & & \vdots & \ddots & & \\ 0 & a_{n2} & a_{n3} & a_{n4} & \cdots & a_{nn} \end{array} \right) \left( \begin{array}{c} x_1 \\ x_2 \\ x_3 \\ x_4 \\ \vdots \\ x_n \end{array} \right) = \left( \begin{array}{c} b_1 \\ b_2 \\ b_3 \\ b_4 \\ \vdots \\ b_n \end{array} \right)$$

$$\left( \begin{array}{cccccc} a_{11} & a_{12} & a_{13} & a_{14} & \cdots & a_{1n} \\ 0 & a_{32} & a_{33} & a_{34} & \cdots & a_{3n} \\ 0 & a_{22} & a_{23} & a_{24} & \cdots & a_{2n} \\ 0 & a_{42} & a_{43} & a_{44} & \cdots & a_{4n} \\ & & \vdots & \ddots & & \\ 0 & a_{n2} & a_{n3} & a_{n4} & \cdots & a_{nn} \end{array} \right) \left( \begin{array}{c} x_1 \\ x_2 \\ x_3 \\ x_4 \\ \vdots \\ x_n \end{array} \right) = \left( \begin{array}{c} b_1 \\ b_3 \\ b_2 \\ b_4 \\ \vdots \\ b_n \end{array} \right)$$

$$\left( \begin{array}{cccccc} a_{11} & a_{14} & a_{13} & a_{12} & \cdots & a_{1n} \\ 0 & a_{34} & a_{33} & a_{32} & \cdots & a_{3n} \\ 0 & a_{24} & a_{23} & a_{22} & \cdots & a_{2n} \\ 0 & a_{44} & a_{43} & a_{42} & \cdots & a_{4n} \\ & & \vdots & \ddots & & \\ 0 & a_{n4} & a_{n3} & a_{n2} & \cdots & a_{nn} \end{array} \right) \left( \begin{array}{c} x_1 \\ x_4 \\ x_3 \\ x_2 \\ \vdots \\ x_n \end{array} \right) = \left( \begin{array}{c} b_1 \\ b_3 \\ b_2 \\ b_4 \\ \vdots \\ b_n \end{array} \right)$$

## Método de Decomposição L U

Considere o sistema de equações lineares  $A \cdot x = b$ , onde todas as submatrizes principais de  $A$  sejam não singulares.

O método decompõe a matriz  $A$  do sistema original em duas matrizes,  $L$  e  $U$ <sup>16</sup>, onde  $A = L \cdot U$ , com  $L$  sendo uma matriz triangular inferior e  $U$  uma matriz triangular superior.

Considere a matriz  $A$  do sistema e a decomposição  $L \cdot U$ , como segue.

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \cdots & a_{3n} \\ \vdots & & & & \\ a_{n1} & a_{n2} & a_{n3} & \cdots & a_{nn} \end{pmatrix}, L = \begin{pmatrix} l_{11} & 0 & 0 & \cdots & 0 \\ l_{21} & l_{22} & 0 & \cdots & 0 \\ l_{31} & l_{32} & l_{33} & \cdots & 0 \\ \vdots & & & & \\ l_{n1} & l_{n2} & l_{n3} & \cdots & l_{nn} \end{pmatrix}, U = \begin{pmatrix} u_{11} & u_{12} & u_{13} & \cdots & u_{1n} \\ 0 & u_{22} & u_{23} & \cdots & u_{2n} \\ 0 & 0 & u_{33} & \cdots & u_{3n} \\ \vdots & & & & \\ 0 & 0 & 0 & \cdots & u_{nn} \end{pmatrix}$$

Pode-se efetuar a decomposição onde a matriz  $L$  tenha a diagonal principal unitária e o procedimento é conhecido como decomposição de Doolittle ou, efetuar a decomposição onde a matriz  $U$  tenha a diagonal principal unitária e o procedimento é conhecido como decomposição de Crout.

Efetuando a decomposição  $A = L \cdot U$ , a solução do sistema é dada como segue.

Se  $A \cdot x = b$ , então  $L \cdot U \cdot x = b$ . Considera-se  $U \cdot x = y$ . Então, pode-se calcular  $L \cdot y = b$ , obtendo-se  $y$ . Com  $y$  calculado, pode-se calcular  $U \cdot x = y$ , obtendo-se  $x$ .

Para deduzir o algoritmo, considera-se a decomposição de Doolittle e o produto  $A = L \cdot U$ , como segue.

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \cdots & a_{3n} \\ \vdots & & & & \\ a_{n1} & a_{n2} & a_{n3} & \cdots & a_{nn} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ l_{21} & 1 & 0 & \cdots & 0 \\ l_{31} & l_{32} & 1 & \cdots & 0 \\ \vdots & & & & \\ l_{n1} & l_{n2} & l_{n3} & \cdots & 1 \end{pmatrix} \cdot \begin{pmatrix} u_{11} & u_{12} & u_{13} & \cdots & u_{1n} \\ 0 & u_{22} & u_{23} & \cdots & u_{2n} \\ 0 & 0 & u_{33} & \cdots & u_{3n} \\ \vdots & & & & \\ 0 & 0 & 0 & \cdots & u_{nn} \end{pmatrix}$$

Multiplicando a linha 1 de  $L$ , pelas colunas 1 até  $n$  de  $U$ .

$$\begin{aligned} a_{11} &= l_{11} \cdot u_{11} \Rightarrow u_{11} = a_{11} \\ a_{12} &= l_{11} \cdot u_{12} \Rightarrow u_{12} = a_{12} \\ \dots \\ a_{1n} &= l_{11} \cdot u_{1n} \Rightarrow u_{1n} = a_{1n} \end{aligned} \quad \left. \begin{aligned} u_{ij} &= a_{ij}, j=1, \dots, n \end{aligned} \right\}$$

Multiplicando as linhas 2 até  $n$  de  $L$ , pela coluna 1 de  $U$ .

$$\begin{aligned} a_{21} &= l_{21} \cdot u_{11} \Rightarrow l_{21} = u_{11} / a_{21} \\ a_{31} &= l_{31} \cdot u_{11} \Rightarrow l_{31} = u_{11} / a_{31} \\ \dots \\ a_{n1} &= l_{n1} \cdot u_{11} \Rightarrow l_{n1} = u_{11} / a_{n1} \end{aligned} \quad \left. \begin{aligned} l_{ii} &= u_{11} / a_{ii}, i=2, \dots, n \end{aligned} \right\}$$

Continua-se o procedimento para determinar, alternadamente, elementos de linhas de  $U$ , a partir do elemento da diagonal e elementos de coluna de  $L$ , a partir do elemento abaixo da diagonal.

<sup>16</sup> Lower e Upper. Cuidado com textos que as chamam de Least e Upper. Já vi alguns por aí. : - )

$$\left. \begin{array}{l} a_{22} = l_{21} \cdot u_{12} + u_{22} \Rightarrow u_{22} = a_{22} - l_{21} \cdot u_{12} \\ a_{23} = l_{21} \cdot u_{13} + u_{23} \Rightarrow u_{23} = a_{23} - l_{21} \cdot u_{13} \\ \dots \\ a_{2n} = l_{21} \cdot u_{1n} + u_{2n} \Rightarrow u_{2n} = a_{2n} - l_{21} \cdot u_{1n} \end{array} \right\} u_{2j} = a_{2j} - l_{21} \cdot u_{1j}, j=1, \dots, n$$

$$\left. \begin{array}{l} a_{32} = l_{31} \cdot u_{12} + l_{32} \cdot u_{22} \Rightarrow l_{32} = \frac{a_{32} - l_{31} \cdot u_{12}}{a_{22}} \\ a_{42} = l_{41} \cdot u_{12} + l_{42} \cdot u_{22} \Rightarrow l_{42} = \frac{a_{42} - l_{41} \cdot u_{12}}{a_{22}} \\ \dots \\ a_{n2} = l_{n1} \cdot u_{12} + l_{n2} \cdot u_{22} \Rightarrow l_{n2} = \frac{a_{n2} - l_{n1} \cdot u_{12}}{a_{22}} \end{array} \right\} l_{i2} = \frac{a_{i2} - l_{i1} \cdot u_{12}}{a_{22}}, i=3, \dots, n$$

Repete-se o procedimento para determinar as outras linhas de U e colunas de L e obtém-se as equações gerais que seguem.

$$\text{Linha de } U: u_{ij} = a_{ij} - \sum_{k=1}^{i-1} l_{ik} \cdot u_{kj}, i \leq j. \quad \text{Coluna de } L: l_{ij} = \frac{a_{ij} - \sum_{k=1}^{j-1} l_{ik} \cdot u_{kj}}{u_{jj}}, i > j.$$

```
// Decomposição de A em L.U
Para k = 1, ..., n faça
    l_{kk} = 1
Para i = 1, ..., n faça
    Para j = i, ..., n faça
        soma = 0
        Para k = 1, ..., i-1 faça
            soma = soma + l_{ik} \cdot u_{kj}
        u_{ij} = a_{ij} - soma
    Para j = i+1, ..., n faça
        soma = 0
        Para k = 1, ..., i-1 faça
            soma = soma + l_{jk} \cdot u_{ki}
        l_{ji} = (a_{ji} - soma) / u_{ii} // usar cláusula de exceção e, caso ocorra, informar e interromper
    // Solução de L.y = b por Substituição
    y_1 = b_1 / l_{11} // não deve ocorrer exceção pois l_{11} = 1 (pode-se escrever y_1 = b_1)
    Para i = 2, ..., n faça
        soma = 0
        Para j = 1, ..., i-1 faça
            soma = soma + l_{ij} \cdot y_j
        y_i = (b_i - soma) / l_{ii} // não deve ocorrer exceção pois l_{ii} = 1 (pode-se escrever y_i = b_i - soma)
    // Solução de U.x = y por Retrosubstituição
    x_n = y_n / u_{nn} // usar cláusula de exceção e, caso ocorra, informar e interromper
    Para i = n-1, ..., 1 faça
        soma = 0
        Para j = i+1, ..., n faça
            soma = soma + u_{ij} \cdot x_j
        x_i = (y_i - soma) / u_{ii} // usar cláusula de exceção e, caso ocorra, informar e interromper
```

É uma boa hora para você escrever o seu algoritmo prevendo exceções e escrever o correspondente programa.

Seguem informações que você poderá adotar em seu programa, caso queira.

Se  $\det(A) \neq 0$ , mas  $\det(A_k) = 0$ , para algum  $k$  entre 1 e  $n$ , é possível utilizar o Método de Decomposição L U, se for efetuada a troca da linha  $k$  com uma linha abaixo, de tal forma que  $\det(A_k) \neq 0$ , para todo  $k$  entre 1 e  $n$ .

A matriz L do Método de Decomposição L U é a matriz dos multiplicadores do Método da Eliminação de Gauss para matriz triangular superior, onde  $l_{ij} = m_{ij}$  e  $l_{kk} = 1$ .

$$L = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ l_{21} & 1 & 0 & \cdots & 0 \\ l_{31} & l_{32} & 1 & \cdots & 0 \\ \vdots & & & & \vdots \\ l_{n1} & l_{n2} & l_{n3} & \cdots & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ m_{21} & 1 & 0 & \cdots & 0 \\ m_{31} & m_{32} & 1 & \cdots & 0 \\ \vdots & & & & \vdots \\ m_{n1} & m_{n2} & m_{n3} & \cdots & 1 \end{pmatrix}$$

Os elementos da matriz U podem ser obtidos através do Método da Eliminação de Gauss para matriz triangular superior.

## Seção Fofoca

- Com eliminação de Gauss podem-se escrever as matrizes L e U, como apresentado acima.  
(Johann Carl Friedrich Gauß, 1777-1855)
- Doolittle particularizou para L com diagonal unitária e Crout para U com diagonal unitária.
- Para matriz A simétrica e definida positiva, tem-se o algoritmo de Cholesky, onde  $L = U^T$ .  
(André-Louis Cholesky, 1875-1918)  
E Cholesky era Engenheiro Civil. :-)
- Alan Turing apresentou, em 1948, o artigo *Rounding-off errors in matrix processes*.

- Não é necessário escrever um algoritmo para Decomposição L U de Crout.

Se  $A = L.U$ ,  $A^T = (L.U)^T = U^T.L^T$ .

Então, dada a matriz A, o algoritmo de Dolittle produz a decomposição de Crout, para  $A^T$ .  
E vice-versa, claro. Ops, e a recíproca é verdadeira, claro.

- Pode-se utilizar o algoritmo para decomposição  $A = L U$ , sem utilizar 3 matrizes.

Basta utilizar apenas a matriz A e armazenar os valores de L e U na própria matriz A.

É claro que a diagonal será da matriz U e a diagonal de L, unitária, será virtual, mental, transcendental ou nirvânica. Altere, em seu programa, todo  $L[i,j]$  e  $U[i,j]$  por  $A[i,j]$  e, se Deus quiser, dará tudo certo. Mas não atribua valor para  $L[i,i]$  que, implicitamente, é 1. E lembre-se que a substituição  $L.y = b$  deve ser modificada, pois  $A[i,i]$  conterá  $U[i,i]$  e, não,  $L[i,i]$ . Então,  $L[i,i]$ , ou  $A[i,i]$  na solução  $L.y = b$  deve ser substituído por 1.

- Um bom motivo para utilizar a linguagem Fortran, é que se pode escrever um programa com A, L e U e, no início do programa, informar que todas as matrizes utilizam a mesma área de memória.

Equivalence ((A,L), (A,U))

Responda se for capaz: afinal, quem inventou tudo isto?



## Método de Gauss Compacto

Considere o sistema de equações lineares  $A \cdot x = b$ , onde todas as submatrizes principais de  $A$  sejam não singulares.

O método decompõe a matriz  $A$  do sistema original em duas matrizes,  $L$  e  $U$ , onde  $A = L \cdot U$ , com  $L$  sendo uma matriz triangular inferior e  $U$  uma matriz triangular superior, tal como no Método de Decomposição LU, apresentado na seção anterior, com  $l_{ii} = 1$ , ou seja, a decomposição de Doolittle.

Contudo, neste método, as matrizes  $L$  e  $U$  são armazenadas em uma única matriz, onde a diagonal contém os elementos  $u_{ii}$  e os elementos  $l_{ij}$  que são unitários ficam implícitos.

Além disto, os cálculos efetuados para calcular os elementos  $u_{ij}$  são estendidos aos elementos do vetor  $b$ . Desta forma, os valores de  $b$  transformados serão os valores de  $y$  da solução  $L \cdot y = b$ , utilizada no Método de Decomposição LU.

Assim, após a decomposição com transformação dos elementos de  $b$ , basta calcular o sistema  $U \cdot x = y$  que é equivalente a calcular  $U \cdot x = b$ , com  $b$  transformado. Ou seja, basta efetuar a retrosubstituição.

O algoritmo pode ser escrito para armazenar  $L$  e  $U$  em uma única matriz, fazendo jus ao termo compacto do nome do método. Mas pode ser ainda mais econômico, armazenando a decomposição na própria matriz  $A$ , como descrito na Seção Fofoca, logo após a seção Método de Decomposição LU.

Considere a matriz  $A$  do sistema e a decomposição LU, armazenada em uma única matriz LU, como segue. Além disto, considere que  $b$  será operado tal como os elementos  $u_{ij}$ .

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \cdots & a_{3n} \\ \vdots & & & & \\ a_{n1} & a_{n2} & a_{n3} & \cdots & a_{nn} \end{pmatrix}, \quad LU = \begin{pmatrix} u_{11} & u_{12} & u_{13} & \cdots & u_{1n} \\ l_{21} & u_{22} & u_{23} & \cdots & u_{2n} \\ l_{31} & l_{32} & u_{33} & \cdots & u_{3n} \\ \vdots & & & & \\ l_{n1} & l_{n2} & l_{n3} & \cdots & u_{nn} \end{pmatrix}$$

Como no Método de Decomposição LU, calcula-se alternadamente linha de  $U$  e coluna de  $L$ , operando, também,  $b$ , com as equações que seguem.

$$\text{Linha 1 a } n: u_{ij} = a_{ij} - \sum_{k=1}^{i-1} l_{ik} \cdot u_{kj}, \quad i \leq j; \quad b_i = b_i - \sum_{k=1}^{i-1} l_{ik} \cdot b_k$$

$$\text{Coluna 1 a } n: l_{ij} = \frac{a_{ij} - \sum_{k=1}^{j-1} l_{ik} \cdot u_{kj}}{u_{jj}}, \quad i > j.$$

Pode-se, também, utilizar a matriz aumentada  $(A|b)$  que terá  $n$  linhas e  $n+1$  colunas e utilizar as equações que seguem, onde  $u_{in+1}$  representa  $b$ .

$$\text{Linha 1 a } n: u_{ij} = a_{ij} - \sum_{k=1}^{i-1} l_{ik} \cdot u_{kj}, \quad i \leq j;$$

$$\text{Coluna 1 a } n+1: l_{ij} = \frac{a_{ij} - \sum_{k=1}^{j-1} l_{ik} \cdot u_{kj}}{u_{jj}}, \quad i > j.$$

Reescreva o programa do Método de Decomposição LU, trocando  $L[i,j]$  e  $U[i,j]$  por  $LU[i,j]$ , para armazenar  $L$  e  $U$  em uma única matriz ou, melhor ainda, trocando  $L[i,j]$  e  $U[i,j]$  por  $A[i,j]$ , para armazenar  $L$  e  $U$  na própria matriz  $A$ . Em qualquer caso, não se preenche  $l_{ii}$  com 1, já que fica implícito e quem ocupa a diagonal são os elementos  $u_{ii}$ . A substituição  $L \cdot y = b$  não é efetuada, pois  $b$  já será  $y$  e a retro-substituição  $U \cdot x = y$  passa a ser  $U \cdot x = b$ , que já foi operado se transformando em  $y$ .

Sua parte é só isto.

## Método de Cholesky

Considere o sistema de equações lineares com a matriz A simétrica e definida positiva. Pode-se efetuar uma decomposição  $A = L \cdot U$ , onde  $L = U^T$  ou, para simplicidade do algoritmo,  $A = G \cdot G^T$ .

Para deduzir o algoritmo, considera-se a decomposição  $A = G \cdot G^T$ , como segue, com  $g_{ij} = g_{ji}$ .

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \cdots & a_{3n} \\ \vdots & & & & \\ a_{n1} & a_{n2} & a_{n3} & \cdots & a_{nn} \end{pmatrix} = \begin{pmatrix} g_{11} & 0 & 0 & \cdots & 0 \\ g_{21} & g_{22} & 0 & \cdots & 0 \\ g_{31} & g_{32} & g_{33} & \cdots & 0 \\ \vdots & & & & \\ g_{n1} & g_{n2} & g_{n3} & \cdots & g_{nn} \end{pmatrix} \cdot \begin{pmatrix} g_{11} & g_{21} & g_{31} & \cdots & g_{n1} \\ 0 & g_{22} & g_{32} & \cdots & g_{n2} \\ 0 & 0 & g_{33} & \cdots & g_{n3} \\ \vdots & & & & \\ 0 & 0 & 0 & \cdots & g_{nn} \end{pmatrix}$$

Multiplicando a linha 1 de G, pelas colunas 1 até n de  $G^T$ .

$$\left. \begin{array}{l} a_{11} = g_{11} \cdot g_{11} \Rightarrow g_{11} = \sqrt{a_{11}} \\ a_{12} = g_{11} \cdot g_{21} \Rightarrow g_{21} = \frac{a_{12}}{g_{11}} \\ \dots \\ a_{1n} = g_{11} \cdot g_{n1} \Rightarrow g_{n1} = \frac{a_{1n}}{g_{11}} \end{array} \right\} \begin{array}{l} g_{11} = \sqrt{a_{11}} \\ g_{i1} = \frac{a_{i1}}{g_{11}}, i=2, \dots, n \\ \text{e, por simetria,} \\ g_{1i} = g_{i1}, i=2, \dots, n \end{array}$$

Multiplicando a linha 2 de G, pelas colunas 2 até n de  $G^T$ .

$$\left. \begin{array}{l} a_{22} = g_{21} \cdot g_{21} + g_{22} \cdot g_{22} \Rightarrow g_{22} = \sqrt{a_{22} - g_{21}^2} \\ a_{23} = g_{21} \cdot g_{31} + g_{22} \cdot g_{32} \Rightarrow g_{32} = \frac{a_{23} - g_{21} \cdot g_{31}}{g_{22}} \\ \dots \\ a_{2n} = g_{21} \cdot g_{n1} + g_{22} \cdot g_{n2} \Rightarrow g_{n2} = \frac{a_{2n} - g_{21} \cdot g_{n1}}{g_{22}} \end{array} \right\} \begin{array}{l} g_{22} = \sqrt{a_{22} - g_{21}^2} \\ g_{i2} = \frac{a_{i2} - g_{21} \cdot g_{i1}}{g_{22}}, i=3, \dots, n \\ \text{e, por simetria,} \\ g_{j1} = g_{1j}, j=3, \dots, n \end{array}$$

Multiplicando a linha 3 de G, pelas colunas 3 até n de  $G^T$ .

$$\left. \begin{array}{l} a_{33} = g_{31} \cdot g_{31} + g_{32} \cdot g_{32} + g_{33} \cdot g_{33} \Rightarrow g_{33} = \sqrt{a_{33} - g_{31}^2 - g_{32}^2} \\ a_{34} = g_{31} \cdot g_{41} + g_{32} \cdot g_{42} + g_{33} \cdot g_{43} \Rightarrow g_{43} = \frac{a_{34} - g_{31} \cdot g_{41} - g_{32} \cdot g_{42}}{g_{33}} \\ \dots \\ a_{3n} = g_{31} \cdot g_{n1} + g_{32} \cdot g_{n2} + g_{33} \cdot g_{n3} \Rightarrow g_{n3} = \frac{a_{3n} - g_{31} \cdot g_{n1} - g_{32} \cdot g_{n2}}{g_{33}} \end{array} \right\} \begin{array}{l} g_{33} = \sqrt{a_{33} - g_{31}^2 - g_{32}^2} \\ g_{i3} = \frac{a_{i3} - g_{31} \cdot g_{i1} - g_{32} \cdot g_{i2}}{g_{33}}, i=4, \dots, n \\ \text{e, por simetria,} \\ g_{j1} = g_{1j}, j=4, \dots, n \end{array}$$

Continua-se o procedimento para determinar, alternadamente, elementos da  $g_{ii}$  da diagonal e elementos  $g_{ij}$  fora da diagonal e copiá-los para a transposta.

$$g_{11} = \sqrt{a_{11}}, g_{i1} = \frac{a_{i1}}{g_{11}}, i=2, \dots, n, g_{ii} = \sqrt{a_{ii} - \sum_{k=1}^{i-1} g_{ik}^2}, i=2, \dots, n, g_{ij} = \frac{a_{ji} - \sum_{k=1}^{j-1} g_{jk} \cdot g_{ik}}{g_{jj}}, j=2, \dots, n, i=j+1, \dots, n$$

e, para cada  $g_{ij}$  calculado,  $g_{ji} = g_{ij}, j=2, \dots, n$

Efetuando a decomposição  $A = G \cdot G^T$ , a solução do sistema é dada como segue.

Se  $A \cdot x = b$ , então  $G \cdot G^T \cdot x = b$ . Considera-se  $G^T \cdot x = y$ . Então, pode-se calcular  $G \cdot y = b$ , obtendo-se  $y$ . Com  $y$  calculado, pode-se calcular  $G^T \cdot x = y$ , obtendo-se  $x$ .

```

// Verificar se A é simétrica
Para i = 1, ..., n-1 faça
    Para j = i+1, ..., n faça
        Se aij ≠ aji então
            informar e interromper

// Decomposição de A em G.GT
Para k = 1, ... n faça
    soma = 0
    Para j = 1, ..., k-1 faça
        soma := soma + gkj2
    gkk = Raiz(akk-soma) // usar cláusula de exceção e, caso ocorra, informar e interromper
    Para i = k+1, ..., n faça
        soma = 0
        Para j = 1, ..., k-1 faça
            soma = soma + gij · gkj
        gik = (aik-soma)/gkk

// Preenchendo a GT
Para i = 1, ..., n-1 faça
    Para j = i+1, ..., n faça
        gij = gji

// Solução de G.y = b por Substituição
y1 = b1/g11 // usar cláusula de exceção e, caso ocorra, informar e interromper
Para i = 2, ..., n faça
    soma = 0;
    Para j = 1, ..., i-1 faça
        soma := soma + gij · yj
    yi = (bi-soma)/gii // usar cláusula de exceção e, caso ocorra, informar e interromper

// Solução de GT.x = y por Retrosubstituição
xn = yn/gnn // usar cláusula de exceção e, caso ocorra, informar e interromper
Para i = n-1, ..., 1 faça
    soma = 0
    Para j = i+1, ..., n faça
        soma = soma + gij · xj
    xi = (yi-soma)/gii // usar cláusula de exceção e, caso ocorra, informar e interromper

```

É uma boa hora para você escrever o seu algoritmo prevendo exceções e escrever o correspondente programa.

E lembre-se que substituindo  $G[i,j]$  por  $A[i,j]$ , a decomposição será armazenada sobre a matriz  $A$  e, se Deus quiser, deverá dar certo.

## Cálculo do Determinante da Matriz

Para os métodos apresentados, o cálculo do determinante da matriz é imediato após obtenção da matriz triangular, por eliminação, ou das matrizes triangulares, por decomposição.

Para uma matriz triangular superior ou inferior, o determinante da matriz é o produto dos elementos da diagonal.

Então, para métodos de eliminação,  $\det(A) = a_{11} \cdot a_{22} \cdot \dots \cdot a_{nn} = \prod_{i=1}^n a_{ii}$ .

Para métodos de decomposição, se  $A = P \cdot Q$ ,  $\det(A) = \det(P) \cdot \det(Q) = p_{11} \cdot p_{22} \cdot \dots \cdot p_{nn} \cdot q_{11} \cdot q_{22} \cdot \dots \cdot q_{nn}$ .

Então, para  $A = L \cdot U$ ,  $\det(A) = \prod_{i=1}^n \prod_{j=1}^n u_{ij} = \prod_{i=1}^n u_{ii}$  e, para  $A = G \cdot G^T$ ,  $\det(A) = \prod_{i=1}^n g_{ii} \cdot \prod_{i=1}^n g_{ii} = \prod_{i=1}^n g_{ii}^2 = \prod_{i=1}^n g_{ii}^2$ .

## Cálculo da Matriz Inversa

Considere  $A$  uma matriz de ordem  $n$ . Se  $\det(A) \neq 0$ , existe uma matriz  $C$  que satisfaz  $A \cdot C = C \cdot A = I$ , onde  $I$  é a matriz identidade.

A matriz  $C$  é a matriz inversa de  $A$  e pode ser representada por  $A^{-1}$ . Então,  $A \cdot A^{-1} = A^{-1} \cdot A = I$ .

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \cdots & a_{3n} \\ \vdots & & & & \\ a_{n1} & a_{n2} & a_{n3} & \cdots & a_{nn} \end{pmatrix} \cdot \begin{pmatrix} c_{11} & c_{12} & c_{13} & \cdots & c_{1n} \\ c_{21} & c_{22} & c_{23} & \cdots & c_{2n} \\ c_{31} & c_{32} & c_{33} & \cdots & c_{3n} \\ \vdots & & & & \\ c_{n1} & c_{n2} & c_{n3} & \cdots & c_{nn} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & & & & \\ 0 & 0 & 0 & \cdots & 1 \end{pmatrix}$$

Multiplicando-se a matriz  $A$  pela primeira coluna de  $A^{-1}$  tem-se a primeira coluna de  $I$ . Chamando de  $C^1$  a coluna 1 de  $A^{-1}$  e de  $I^1$  a coluna 1 de  $I$ , tem-se  $A \cdot C^1 = I^1$ . De forma geral, tem-se  $A \cdot C^j = I^j$ ,  $j = 1, \dots, n$ .

Basta, então, resolver o sistema  $A \cdot C^j = I^j$ ,  $j = 1, \dots, n$  para obter-se a matriz inversa de  $A$ . De forma prática, basta utilizar qualquer método apresentado, e resolver  $n$  vezes o sistema  $A \cdot x = b$ . Um algoritmo geral pode ser escrito como segue.

```
Para r = 1, ..., n faça
  b = I^r // b = I^r = (0 0 ... 0 1 0 ... 0 0)^T com 1 na r-ésima posição
  Resol ver A.x = b
  C^r = x // C^r é a r-ésima coluna de A^-1
```

Esta proposta gera grande esforço computacional. Podem-se, então, modificar os algoritmos de eliminação já desenvolvidos para que a operação de pivotamento sobre a linha  $i$  efetue a operação  $b_i = b_i - m_i \cdot b_j$  para uma matriz  $B$  com as colunas de  $B$  sendo as colunas de  $I$ .

No caso dos algoritmos de decomposição, deve-se inicialmente decompor a matriz  $A$  e, a seguir, resolver o sistema  $n$  vezes, pois nestes métodos não há pivotamento sobre o vetor  $b$  e as  $n$  soluções serão obtidas com uma única decomposição.

## Solução de sistemas de equações lineares por Métodos Iterativos

Os métodos iterativos se baseiam na construção de uma sequência de aproximações onde, a cada iteração, os valores da iteração anterior são utilizados e devem produzir um resultado mais aproximado da solução. Portanto, requerem uma solução aproximada<sup>17</sup> inicial.

O critério de parada, tal como em outros métodos apresentados, podem mediar a distância ou a distância relativa. No caso dos problemas com variáveis  $x \in \mathbb{R}$ , com uma sequência de pontos  $x^k$ , pode-se utilizar  $|x_{k+1} - x_k|$  ou  $|x_{k+1} - x_k|/|x_{k+1}|$ . No caso de problemas com variáveis  $x \in \mathbb{R}^n$ ,  $\|x^{k+1} - x^k\|$  ou  $\|x^{k+1} - x^k\|/\|x^{k+1}\|$ .

Norma de um vetor,  $\|x\|$ , com  $x \in \mathbb{R}^n$  é qualquer função  $\|x\|: \mathbb{R}^n \rightarrow \mathbb{R}$ , que satisfaça as condições:

- $\|x\| \geq 0$  e  $\|x\| = 0 \Leftrightarrow x = 0$
- $\|\lambda \cdot x\| = |\lambda| \cdot \|x\|$ ,  $\forall \lambda$
- $\|x + y\| \leq \|x\| + \|y\|$ , ou seja, atende à desigualdade triangular

As normas usuais para vetores são:

- $\|x\|_\infty = \max_{1 \leq i \leq n} |x_i|$

- $\|x\|_1 = \sum_{i=1}^n |x_i|$

- $\|x\|_E = \sqrt{\sum_{i=1}^n x_i^2}$

As normas usuais para matrizes são:

- $\|A\|_\infty = \max_{1 \leq i \leq n} \sum_{j=1}^n |a_{ij}|$ , chamada norma linha

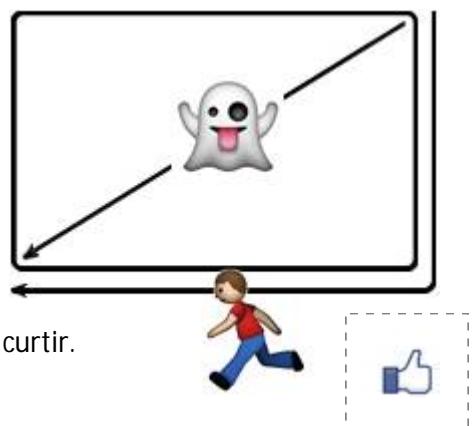
- $\|A\|_1 = \max_{1 \leq j \leq n} \sum_{i=1}^n |a_{ij}|$ , chamada norma coluna

- $\|A\|_E = \sqrt{\sum_{i=1}^n \sum_{j=1}^n a_{ij}^2}$ , chamada norma Euclidiana

### A escolha da norma para critério de parada

Considere um retângulo e um ser que sai de um de seus vértices e deve chegar até o vértice diametralmente oposto. A medida de distância mais natural para seres do além é a norma Euclidiana, mas para seres mortais, é mais adequado que se meça a distância percorrendo dois lados do retângulo.

Uma boa norma é  $\|x\|_E = (x^T \cdot x)^{1/2}$ . E se gostou não se esqueça de curtir.



<sup>17</sup>Solução aproximada inicial é uma forma dos matemáticos dizerem aquilo que os engenheiros chamam de estimativa inicial e os alunos chamam de chute inicial. Escolha seu dialeto. :-)

## Método de Jacobi

Considere o sistema de equações lineares escrito na forma  $A.x = b$ . A matriz A pode ser escrita como a soma de uma matriz diagonal D e uma matriz com os elementos restantes R, ou seja,  $A = D+R$ .

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \cdots & a_{3n} \\ \ddots & & & & \\ a_{n1} & a_{n2} & a_{n3} & \cdots & a_{nn} \end{pmatrix} = \begin{pmatrix} a_{11} & 0 & 0 & \cdots & 0 \\ 0 & a_{22} & 0 & \cdots & 0 \\ 0 & 0 & a_{33} & \cdots & 0 \\ \ddots & & \ddots & & \\ 0 & 0 & 0 & \cdots & a_{nn} \end{pmatrix} + \begin{pmatrix} 0 & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & 0 & a_{23} & \cdots & a_{2n} \\ a_{31} & a_{32} & 0 & \cdots & a_{3n} \\ \ddots & & \ddots & & \\ a_{n1} & a_{n2} & a_{n3} & \cdots & 0 \end{pmatrix}$$

Desta forma, o sistema pode ser reescrito como  $(D+R).x = b$  ou,  $D.x+R.x = b$ . Então,  $D.x = b-R.x$  e  $x = D^{-1}.(b-R.x)$ . Em um procedimento iterativo pode-se escrever  $x^{k+1} = D^{-1}.(b-R.x^k)$ .

Considerando as matrizes D e R como definidas, a equação matricial  $x^{k+1} = D^{-1}.(b-R.x^k)$  equivale a

$$x_i^{k+1} = \frac{1}{a_{ii}} \left( b_i - \sum_{\substack{j=1 \\ j \neq i}}^n a_{ij} \cdot x_j^k \right), \quad i = 1, 2, \dots, n.$$

Em resumo, sem nenhuma elegância, pode-se afirmar que para cada linha do sistema de equações, a variável  $x_i$  é isolada e calculada a partir dos valores das outras variáveis da iteração anterior.

$$\begin{cases} a_{11} \cdot x_1 + a_{12} \cdot x_2 + a_{13} \cdot x_3 + \cdots + a_{1n} \cdot x_n = b_1 \\ a_{21} \cdot x_1 + a_{22} \cdot x_2 + a_{23} \cdot x_3 + \cdots + a_{2n} \cdot x_n = b_2 \\ \vdots \\ a_{n1} \cdot x_1 + a_{n2} \cdot x_2 + a_{n3} \cdot x_3 + \cdots + a_{nn} \cdot x_n = b_n \end{cases} \quad \begin{cases} x_1^{k+1} = b_1 - (a_{12} \cdot x_2^k + a_{13} \cdot x_3^k + \cdots + a_{1n} \cdot x_n^k) / a_{11} \\ x_2^{k+1} = b_2 - (a_{21} \cdot x_1^k + a_{23} \cdot x_3^k + \cdots + a_{2n} \cdot x_n^k) / a_{22} \\ \vdots \\ x_n^{k+1} = b_n - (a_{n1} \cdot x_1^k + a_{n2} \cdot x_2^k + \cdots + a_{n-1} \cdot x_{n-1}^k) / a_{nn} \end{cases}$$

Como  $x_i^{k+1}$  é calculado com os valores de x da iteração anterior, para quem gosta de hi-tech, o algoritmo pode ser considerado um algoritmo paralelo. Mas não invente moda em seu programa. : - )

A convergência do método não depende do ponto de partida  $x^0$ . Partindo-se de um vetor inicial  $x^0$ , calcula-se  $x^k$ , para  $k = 1, 2, 3, \dots$  até que  $\|x^{k+1} - x^k\| / \|x^{k+1}\| < \varepsilon$ .

A convergência do método é garantida se um dos critérios for satisfeito:

- $\max_{1 \leq i \leq n} \sum_{\substack{j=1 \\ j \neq i}}^n \left| \frac{a_{ij}}{a_{ii}} \right| < 1$ , chamado critério das linhas
- $\max_{1 \leq j \leq n} \sum_{\substack{i=1 \\ i \neq j}}^n \left| \frac{a_{ij}}{a_{jj}} \right| < 1$ , chamado critério das colunas

O algoritmo requer um teste de parada com medida de distância absoluta ou relativa entre os dois últimos pontos.

Pode-se utilizar qualquer das normas apresentadas para isto. Segue uma sugestão de algoritmo utilizando a Norma Euclidiana.

```
// Rotina para cálculo de distância relativa para critério de parada
Numerador = 0
Denominador = 0
Para i = 1, ..., n faça
    Numerador = Numerador + (xi - xInii)2
    Denominador = Denominador + xi2
Distancia = Raiz(Numerador)/Raiz(Denominador)
```

Um algoritmo para solução de sistemas lineares pelo método de Jacobi pode ser escrito como segue.

```
// Verificar critério das linhas
Maximo = 0
Para i = 1, ..., n faça
    soma = 0
    Para j = 1, ..., n faça
        Se i ≠ j então
            soma = soma + |aij|
        soma = soma / |aii|
    Se Maximo < soma então
        Maximo = soma
    CDL = Maximo < 1 // CDL é Verdade se Maximo < 1

// Verificar critério de colunas
Maximo = 0
Para j = 1, ..., n faça
    soma = 0
    Para i = 1, ..., n faça
        Se i ≠ j então
            soma = soma + |aij|
        soma = soma / |ajj|
    Se Maximo < soma então
        Maximo = soma
    CDC = Maximo < 1 // CDC é Verdade se Maximo < 1

Se CDL e CDC Falsos // Se um dos critérios é Verdade continua e, se não, ...
    informar e interromper // ... pode-se tentar a solução, mas vou interromper

// Iterações do Método (a solução inicial é xIni que deve estar disponível)
Iteracao = 0
Repetir indefinidamente
    Incrementar Iteracao
    Se Iteracao > LimiteIteracao
        informar e interromper
    Para i = 1, ..., n faça
        soma = 0
        Para j = 1, ..., n faça
            Se j ≠ i então
                soma = soma + aij * xInij
            xi = (bi - soma) / aii
        Se Distancia < ε então
            Solução encontrada e fim da repetição
        Para i = 1, ..., n faça
            xInii = xi
```

É uma boa hora para você escrever o seu algoritmo prevendo exceções e escrever o correspondente programa.

E lembre-se que é necessário ter o vetor  $x$  da iteração  $k$  e da iteração  $k+1$ , tanto para o cálculo de  $x_i^{k+1}$  como para o cálculo de  $\|x^{k+1} - x^k\|$ .

No algoritmo apresentado, fica claro que  $x^k$  é  $x_{Ini}$  e  $x^{k+1}$  é  $x$ .

## Método de Gauss-Seidel

Considere o sistema de equações lineares escrito na forma  $A.x = b$ . A matriz A pode ser escrita como a soma de uma matriz diagonal D, uma matriz triangular estritamente inferior L e uma matriz triangular estritamente superior U, ou seja,  $A = D+L+U$ .

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \cdots & a_{3n} \\ \vdots & & & & \\ a_{n1} & a_{n2} & a_{n3} & \cdots & a_{nn} \end{pmatrix} = \begin{pmatrix} a_{11} & 0 & 0 & \cdots & 0 \\ 0 & a_{22} & 0 & \cdots & 0 \\ 0 & 0 & a_{33} & \cdots & 0 \\ \vdots & & & & \\ 0 & 0 & 0 & \cdots & a_{nn} \end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 & \cdots & 0 \\ a_{21} & 0 & 0 & \cdots & 0 \\ a_{31} & a_{32} & 0 & \cdots & 0 \\ \vdots & & & & \\ a_{n1} & a_{n2} & a_{n3} & \cdots & 0 \end{pmatrix} + \begin{pmatrix} 0 & a_{12} & a_{13} & \cdots & a_{1n} \\ 0 & 0 & a_{23} & \cdots & a_{2n} \\ 0 & 0 & 0 & \cdots & a_{3n} \\ \vdots & & & & \\ 0 & 0 & 0 & \cdots & 0 \end{pmatrix}$$

Desta forma, o sistema pode ser reescrito como  $(D+L+U).x = b$  ou,  $D.x+L.x+U.x = b$ . Então,  $D.x = b-L.x-U.x$  e  $x = D^{-1}.(b-L.x-U.x)$ . Em um procedimento iterativo, considerando que os valores de  $x^{k+1}$  já estão calculados para  $j < i$ , pode-se escrever  $x^{k+1} = D^{-1}.(b-L.x^{k+1}-U.x^k)$ .

Considerando as matrizes D, L e U como definidas, a equação matricial  $x^{k+1} = D^{-1}.(b-L.x^{k+1}-U.x^k)$  equivale a  $x_i^{k+1} = \frac{1}{a_{ii}} \left( b_i - \sum_{j=1}^{i-1} a_{ij} \cdot x_j^{k+1} - \sum_{j=i+1}^n a_{ij} \cdot x_j^k \right)$ ,  $i = 1, 2, \dots, n$ .

Em resumo, sem nenhuma elegância, pode-se afirmar que para cada linha do sistema de equações, a variável  $x_i$  é isolada e calculada a partir dos valores das outras variáveis da iteração anterior se ainda não calculadas na iteração atual e com valores atualizados se já calculados na iteração atual.

$$\begin{cases} a_{11} \cdot x_1 + a_{12} \cdot x_2 + a_{13} \cdot x_3 + \cdots + a_{1n} \cdot x_n = b_1 \\ a_{21} \cdot x_1 + a_{22} \cdot x_2 + a_{23} \cdot x_3 + \cdots + a_{2n} \cdot x_n = b_2 \\ \vdots \\ a_{n1} \cdot x_1 + a_{n2} \cdot x_2 + a_{n3} \cdot x_3 + \cdots + a_{nn} \cdot x_n = b_n \end{cases} \quad \begin{cases} x_1^{k+1} = b_1 - (a_{12} \cdot x_2^k + a_{13} \cdot x_3^k + \cdots + a_{1n} \cdot x_n^k) / a_{11} \\ x_2^{k+1} = b_2 - (a_{21} \cdot x_1^{k+1} + a_{23} \cdot x_3^k + \cdots + a_{2n} \cdot x_n^k) / a_{22} \\ \vdots \\ x_n^{k+1} = b_n - (a_{n1} \cdot x_1^{k+1} + a_{n2} \cdot x_2^{k+1} + \cdots + a_{nn-1} \cdot x_{n-1}^{k+1}) / a_{nn} \end{cases}$$

Como  $x_i^{k+1}$  é calculado com os valores de x da iteração anterior e da iteração atual, para quem gosta de hi-tech, o algoritmo não é um algoritmo paralelo. Então, neste caso, não dá para inventar moda em seu programa. :-)

A convergência do método não depende do ponto de partida  $x^0$ . Partindo-se de um vetor inicial  $x^0$ , calcula-se  $x^k$ , para  $k = 1, 2, 3, \dots$  até que  $\|x^{k+1} - x^k\| / \|x^{k+1}\| < \varepsilon$ .

A convergência do método é garantida se um dos critérios for satisfeito:

- $\max_{1 \leq i \leq n} \sum_{\substack{j=1 \\ j \neq i}}^n |a_{ij}| / |a_{ii}| < 1$ , chamado critério das linhas
- $\max_{1 \leq i \leq n} \beta_i < 1$ , chamado critério de Sassenfeld, com  $\beta_i = \sum_{j=1}^{i-1} |a_{ij}| \cdot \beta_j - \sum_{j=i+1}^n |a_{ij}|$ .

Um algoritmo para solução de sistemas lineares pelo método de Jacobi pode ser escrito como segue.

```

// Verificar critério das linhas
... (veja trecho de programa na seção Método de Jacobi)

// Verificar critério de Sassenfeld
Maximo = 0
Para i = 1, ..., n faça
     $\beta_i = 0$ 
    Para j = 1, ..., i-1 faça
         $\beta_i = \beta_i + |a_{ij}/a_{ii}| \cdot \beta_j$  // Soma da linha de j = 1 até i-1
    Para j = i+1, ..., n faça
         $\beta_i = \beta_i + |A_{ij}/A_{ii}|$  // Soma da linha de j = i+1 até n
    Se Maximo <  $\beta_i$  então
        Maximo =  $\beta_i$ 
    CDS = Maximo < 1 // CDS é True se Maximo < 1
Se CDL e CDS Falsos
    informar e interromper // Se um dos critérios é Verdade continua, se não, ...
                           // ... pode-se tentar a solução, mas vou interromper

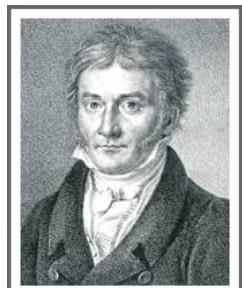
// Iterações do Método (a solução inicial é x e, xIni que deve estar disponível, deve ser copiado em x)
Para i = 1, ..., n faça
     $x_i = x_{\text{Ini}_i}$ 
Iteracao = 0
Repetir indefinidamente
    Incrementar Iteracao
    Se Iteracao > LimiteIteracao
        informar e interromper
    Para i = 1, ..., n faça
        soma = 0
        Para j = 1, ..., n faça
            Se  $j \neq i$  então
                soma = soma +  $a_{ij} \cdot x_j$  // Diferente de Jacobi que usa  $x_{\text{Ini}}[j]$ 
             $x_i = (b_i - soma) / a_{ii}$ 
        Se Distancia <  $\epsilon$  então // Veja rotina Distância na seção Método de Jacobi
            Solução encontrada e fim da repetição
        Para i = 1, ..., n faça
             $x_{\text{Ini}_i} = x_i$ 

```

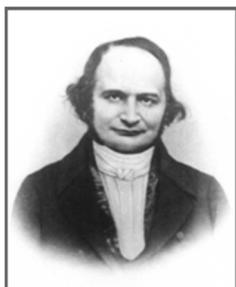
É uma boa hora para você escrever o seu algoritmo prevendo exceções e escrever o correspondente programa.

E lembre-se que é necessário ter o vetor  $x$  da iteração  $k$  e da iteração  $k+1$ , tanto para o cálculo de  $x_i^{k+1}$  como para o cálculo de  $\|x^{k+1} - x^k\|$ .

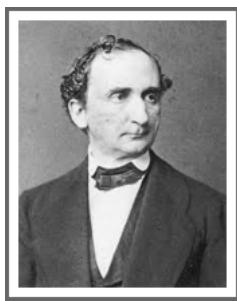
## Galeria de Fotos



Johann Carl Friedrich Gauß  
Braunschweig, 30 de abril de 1777  
Göttingen, 23 de fevereiro de 1855



Carl Gustav Jakob Jacobi  
Potsdam, 10 de dezembro de 1804  
Berlim, 18 de fevereiro de 1851



Philipp Ludwig von Seidel  
Zweibrücken, 23 de outubro de 1821  
Munique, 13 de agosto de 1896



André-Louis Cholesky  
Montguyon, 15 de outubro de 1875  
31 de agosto de 1918



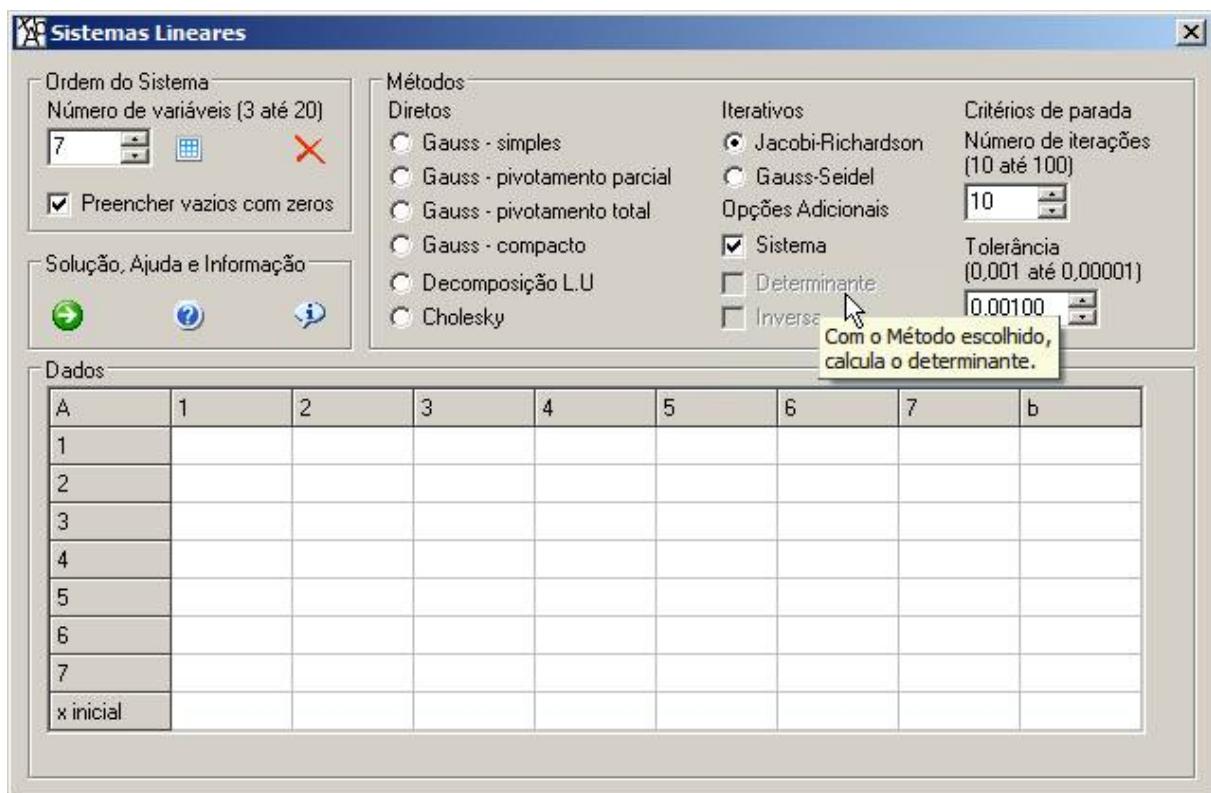
## Sugestões para seus programas

Um simpático programa, para resolver sistemas de equações lineares deverá permitir a definição da ordem do sistema, a seleção do método, a definição do critério de parada para métodos iterativos e opções adicionais que permitam selecionar solução do sistema, cálculo do determinante da matriz e cálculo da matriz inversa.

Além disto, o programa deve ajudar e policiar o usuário, este ser que não comprehende como é difícil fazer um programa e sempre quer que o programa faça o que ele deveria fazer.

Como exemplo, redimensionar a grade com a ordem do sistema, sem apagar os dados, pois o usuário poderá testar problemas com a mesma estrutura da matriz e ao aumentar ou diminuir a ordem do sistema poderá aproveitar os dados anteriores. É claro que, neste caso, deverá existir um botão para apagar os dados quando for necessário. Outro exemplo, é não permitir que o usuário queira o cálculo do determinante ou a inversa da matriz em métodos iterativos. E por aí vai. Usuário é um ser exigente e, muitas vezes, sem noção. E você trabalha para ele. :-)

Um programa com as características descritas poderia ter a cara que segue.



Com este programa é possível:

- definir a ordem do sistema;
- preencher caselas vazias com zeros;
- resolver o sistema com métodos diretos e iterativos;
- calcular o determinante e a inversa da matriz com um ou mais métodos;
- definir critérios de parada para os métodos iterativos;
- interromper o programa caso não converja em tempo suportável pelo usuário.

## Instruções claras e precisas sobre trabalhos computacionais para avaliação

Todas as instruções necessárias estão descritas no mesmo item, do Capítulo 7. Copiar e colar é muito fácil, mas não é nada elegante, agradável, simpático ou conveniente repetir tudo aquilo que já está escrito em capítulo anterior. Além disto, se você já entregou algum trabalho, já sabe como deve proceder. Caso contrário, veja as instruções no Capítulo 7. E antes de entregar, leia as instruções em <http://sacoman.dco.fc.unesp.br/mnc/pet/>.

Se você optar em fazer um programa que tenha a aparência da figura apresentada ou similar, poderá continuar lendo para ter algumas explicações que poderão ser úteis para você.

### Um exemplo de programa

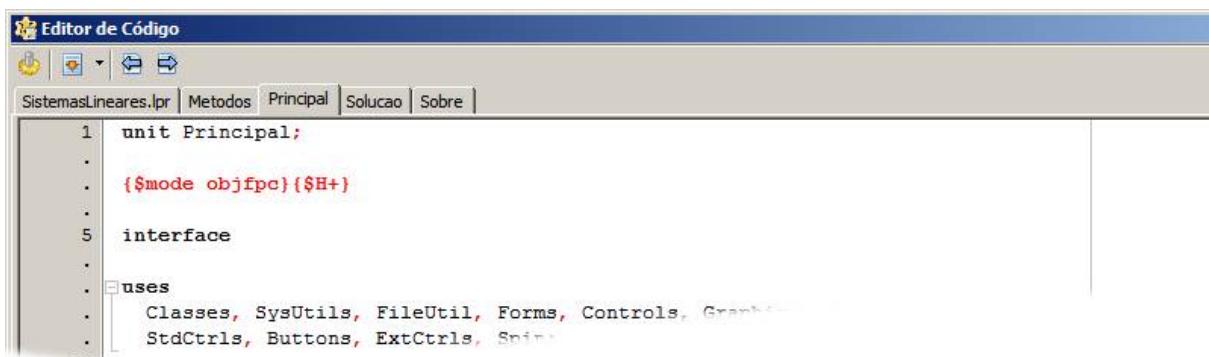
Minha ajuda será *apenas na fabricação da interface*. O resto é com você. Depois de pronto, limpe todo lixo gerado durante a elaboração do projeto, compacte com Zip, Rar ou qualquer outro compactador, com o sugestivo nome T3-RA-V1.ext, onde RA é o número de seu Registro Acadêmico, ext é a extensão fornecida pelo seu compactador, V1 é a primeira versão e entregue via endereço eletrônico de recepção que você já sabe qual é. Não sabe qual é? Então leia tudo sobre o assunto no endereço <http://sacoman.dco.fc.unesp.br/mnc/pet/>.

Então, mãos a obra.

Crie um diretório apropriado para desenvolver seu projeto (e.g.: X:\MNC-Trabalhos\T3). Organize seu ambiente de trabalho e crie subdiretórios necessários para o projeto, como, por exemplo, um diretório chamado Figuras, para as figuras dos botões e ícone e outro chamado Exemplo, para o exemplo executável que está à sua disposição.

Copie de <http://sacoman.dco.fc.unesp.br/mnc/material/> o arquivo M3-Sistemas de Equações Lineares.rar e descompacte. Mova SistemasLineares.exe e Ajuda.chm para o diretório exemplo, para executar quando quiser. Mova as figuras para o diretório Figuras.

Inicie seu ambiente de projeto, no meu caso, Lazarus, ajeite as janelas da forma como achar conveniente e salve o projeto com o nome SistemasLineares.lpr e a unidade principal com o nome Principal. Pressione o botão Nova Fórmula e salve dando o nome Solucao para esta nova unidade. Crie mais uma e salve com o nome Sobre. Pressione o botão Nova Unidade e salve com o nome Metodos. Esta unidade conterá todos os métodos de solução de sistemas lineares, além das variáveis globais que são utilizadas em mais de uma unidade. Ajeite as abas, para facilitar a utilização, como segue.



```

Editor de Código
SistemasLineares.lpr | Metodos | Principal | Solucao | Sobre |
1 unit Principal;
.
.
.
5 interface
.
.
.
uses
.
.
.
Classes, SysUtils, FileUtil, Forms, Controls, Graphics,
StdCtrls, Buttons, ExtCtrls, Sma...

```

Pressione a aba Sobre, pressione F12 para ver a fórmula, atribua Caption=Sobre o programa Sistemas Lineares, BorderStyle=bsDialog, para Font, Name=Tahoma e Size=8, Height=237, Width=177 e Position=poDesktopCenter.

Lance, na fórmula, um componente TLabel e atribua Caption=Quase tudo feito por Gauss, Left=24 e Top=8. Lance outro componente TLabel e atribua Caption=Johann Carl Friedrich Gauß, Left=24 e Top=216. E, neste buraco que ficou na fórmula, lance um componente TImage e atribua Height=183, Width=161, Left=8, Top=28 e, em Picture, carregue o arquivo Gauss.bmp.

Seu About Box está pronto. Não faça mais nada. Mas, quando utilizar o meu programa, cutuque a imagem do Gauss, para ouvir a Patricinha reclamando. É claro que você se lembra da Patricinha.

Pressione a aba Solucao, pressione F12 para ver a fórmula, atribua Caption=Sistemas Lineares, BorderStyle=bsSingle, para BorderIcons, biMaximize=False e biMinimize=False, para Font, Name=Tahoma e Size=8, Height=493, Width=632 e Position=poDesktopCenter.

Lance, na fórmula, dois componentes TGroupBox e atribua, para GroupBox1, as propriedades Caption=Solução do sistema, Left=8, Top=8, Height=224 e Width=617. Para GroupBox2 atribua Caption=Matriz inversa, Left=8, Top=261, Height=224 e Width=617. Lance, na fórmula, um componente TLabel e atribua Caption=Determinante = , Left=18 e Top=240.

Lance, no GroupBox1, um componente TStringGrid e atribua ColCount=16, RowCount=15, Height=200 e Width=597. Lembre-se que meu projeto é desenvolvido em máquina Windows XP. Se você estiver projetando em máquina Windows 7 ou superior, ajuste as dimensões para ficar agradável ou, deixe como está, pois, em tempo de execução, as medidas serão corrigidas. Caso queira editar na grade, atribua Options goEditing=True. Para um efeito de rolagem mais rápido, atribua Options goSmoothScroll=False.

Lance, no GroupBox2, um componente TStringGrid e repita os atributos da grade anterior. Ou marque a grade anterior, copie, marque o GroupBox2 e cole.

Basta isto na fórmula da Unit Solucao, mas pode-se refinar. Não é necessário, mas é adequado. Caso queira abrir a fórmula sempre na mesma posição em que foi fechada anteriormente, deve-se marcar a posição em que estava ao ser fechada e restaurar a posição ao ser reaberta. Então, algumas linhas de código. Cutuque a fórmula e, em Eventos, dê um duplo toque em OnCreate, em OnShow e em OnCloseQuery. Crie variáveis necessárias e preencha as rotinas como segue.

```
implementation
{$R *.lfm}
{ TForm2 }

var
  PrimeiraVez: Boolean; // Primeira vez que a fórmula é aberta
  F2L, F2T: Integer; // Left e Top da fórmula 2, para abrir na mesma posição em que foi fechada

procedure TForm2.FormCreate(Sender: TObject);
begin
  PrimeiraVez := True;
end;
```

```

procedure TForm2.FormShow(Sender: TObject);
begin
  if PrimeiraVez then
    Exit;
  Left := F2L;
  Top := F2T;
end;

procedure TForm2.FormCloseQuery(Sender: TObject; var CanClose: boolean);
begin
  PrimeiraVez := False;
  F2L := Left;
  F2T := Top;
  CanClose := True;
end;

```

Sua fórmula de Solução está pronta. Não faça mais nada. O preenchimento da solução será programado na fórmula Principal.

Duas unidades de programação já foram feita. Só faltam duas, que são Unit Metodos e Unit Principal. Então, está quase pronto. Você ainda acredita nesta frase?

Pressione a aba Principal, pressione F12 para ver a fórmula, atribua Caption=Sistemas Lineares, BorderStyle=bsSingle, para BorderIcons, biMaximize=False e biMinimize=False, para Font, Name=MS Sans Serif e Size=8, Height=390, Width=632, Position=poDesktopCenter e Show.

Comece, com calma, a montar a interface principal de seu programa. Volte algumas páginas e reveja a interface proposta. Deve ter mais de 30 componentes. Então, comece a salpicá-los na fórmula.

Lance, na fórmula, alguns componentes como segue.

Componente	Caption	Left	Top	Height	Width
GroupBox1	Ordem do Sistema	8	8	88	172
GroupBox2	Solução, Ajuda e Informação	8	104	52	172
GroupBox3	Métodos	188	8	148	437
GroupBox4	Dados	8	158	224	617

Para facilitar o lançamento de componente, segue a lista de componentes por tipo, independentemente do local onde será lançado. O local de lançamento consta da tabela, onde GB é GroupBox.

GB	Componente	Caption	Left	Top
1	Label1	Número de variáveis (3 até 20)	8	0
3	Label2	Diretos	8	0
3	Label3	Iterativos	184	0
3	Label4	Opções Adicionais	184	52
3	Label5	Critérios de parada	312	0
3	Label6	Número de iterações   (10 até 100)	312	16
3	Label7	Tolerância   (0,001 até 0,00001)	312	72

O caractere | significa quebra de linha. Utilize o editor de Caption para editar texto com mais de uma linha.

GB	Componente	Left	Top	Heighth	Width	MinValue	MaxValue	Value	Increment	DecimalPlaces
1	SpinEdit1	8	16	21	56	3	20	3	1	0
3	SpinEdit2	312	45	21	56	10	100	10	1	0
3	FloatSpinEdit1	312	101	21	72	0,00001	0,001	0,001	0,00001	5

GB	Componente	Left	Top	Flat	Glyph
1	SpeedButton1	72	16	True	grid.gif
1	SpeedButton2	136	16	True	delete.gif
2	SpeedButton3	8	8	True	forward.gif
2	SpeedButton4	72	8	True	help2.gif
2	SpeedButton5	136	8	True	inform1.gif

GB	Componente	Caption	Left	Top	Checked
1	CheckBox1	Preencher vazios com zeros	8	48	True
3	CheckBox2	Sistema	184	70	True
3	CheckBox3	Determinante	184	88	False
3	CheckBox4	Inversa	184	106	False

GB	Componente	Caption	Left	Top	Checked
3	RadioButton1	Gauss - simples	8	16	True
3	RadioButton2	Gauss - pivotamento parcial	8	34	False
3	RadioButton3	Gauss - pivotamento total	8	52	False
3	RadioButton4	Gauss - compacto	8	70	False
3	RadioButton5	Decomposição L.U	8	88	False
3	RadioButton6	Cholesky	8	106	False
3	RadioButton7	Jacobi-Richardson	184	16	False
3	RadioButton8	Gauss-Seidel	184	34	False

Pressione a aba Solucao, pressione F12 para ver a fórmula, selecione a TStringGrid1 que reside em GroupBox1, pressione Ctrl+C, pressione a aba Principal, pressione F12 para ver a fórmula, selecione GroupBox4, pressione Ctrl+V e, se *tudo der certo*, você terá uma cópia da grade da Unit Solucao na Unit Principal. Se você decidiu não editar a grade da fórmula Solucao, atribua Options goEditing=True, pois nesta grade será necessário editar.

E, se *não der certo*, faça como fez para criar as outras grades da fórmula Solucao, ou seja, lance, no GroupBox4, um componente TStringGrid e atribua ColCount=16, RowCount=15, Height= 200 e Width=597. Esta grade deve ser editada e, então, atribua Options goEditing=True. Para um efeito de rolagem mais rápido, atribua Options goSmoothScroll=False.

A propriedade Hint de cada componente será atribuída na rotina FormCreate, que fica mais fácil e visível para editar e dar manutenção.

Acredite, se quiser, que agora só falta fazer o programa. Siga a receita para criar as rotinas que serão disparadas pelos eventos e, depois, copie descaradamente o programa. Ou faça o seu.

Cutucando:

- Cutoque a forma e, em Eventos, dê um toque duplo em OnCreate e em OnShow, para criar as rotinas FormCreate e FormShow.

- Selecione SpinEdit1 e, em Eventos, dê um toque duplo em OnExit e em OnKeyPress, para criar as rotinas SpinEdit1Exit e SpinEdit1KeyPress.
- Selecione SpinEdit2 e, em Eventos, dê um toque duplo em OnExit, para criar a rotina SpinEdit2Exit.
- Selecione FloatSpinEdit1 e, em Eventos, dê um toque duplo em OnExit, para criar a rotina FloatSpinEdit1Exit.
- Selecione RadioButton1 e, em Eventos, dê um toque duplo em OnClick, para criar a rotina RadioButton1Click. No campo de edição altere seu nome para RadioButtonMetodosDiretosClick.
- Selecione apenas os RadioButton2 até RadioButton6 e, em Eventos, descontine OnClick e selecione RadioButtonMetodosDiretosClick. Assim, todos os botões de rádio para métodos diretos acionarão a mesma rotina.
- Selecione RadioButton7 e, em Eventos, dê um toque duplo em OnClick, para criar a rotina RadioButton7Click. No campo de edição altere seu nome para RadioButtonMetodosIterativosClick.
- Selecione apenas o RadioButton8 e, em Eventos, descontine OnClick e selecione RadioButtonMetodosIterativosClick. Assim, todos os botões de rádio para métodos iterativos acionarão a mesma rotina.
- Selecione CheckBox2 e, em Eventos, dê um toque duplo em OnClick, para criar a rotina CheckBox2Click. No campo de edição altere seu nome para CheckBoxOpcoesClick.
- Selecione apenas os CheckBox3 e CheckBox4 e, em Eventos, descontine OnClick e selecione CheckBoxOpcoesClick. Assim, as caixas de seleção para Sistema, Determinante e Inversa acionarão a mesma rotina.
- Selecione StringGrid1 e, em Eventos, dê um toque duplo em OnSelectCell, para criar a rotina StringGrid1SelectCell.
- Selecione SpeedButton1 e, em Eventos, dê um toque duplo em OnClick, para criar a rotina SpeedButton1Click.
- Selecione SpeedButton2 e, em Eventos, dê um toque duplo em OnClick, para criar a rotina SpeedButton2Click.
- Selecione SpeedButton3 e, em Eventos, dê um toque duplo em OnClick, para criar a rotina SpeedButton3Click.
- Selecione SpeedButton4 e, em Eventos, dê um toque duplo em OnClick, para criar a rotina SpeedButton4Click.
- Selecione SpeedButton5 e, em Eventos, dê um toque duplo em OnClick, para criar a rotina SpeedButton5Click.

Você criou 15 rotinas em uma ordem lógica e o Lazarus as colocou em ordem alfabética. No meu projeto, eu as reordenei, para continuarem na ordem lógica. Se considerar que deve reordená-las, reordene de acordo com a lista de criação, de FormCreate até SpeedButton5, que ficará mais fácil para entender o programa e, principalmente, para ~~descaradamente~~ copiá-lo.

Agora é só copiar ou fazer o seu. Inicialmente são definidas algumas unidades necessárias na cláusula uses, alguma variáveis usadas apenas nesta Unit e, depois as rotinas. Lembre-se que a Unit Metodos contém a definição de várias variáveis globais que você verá nas rotinas, mas que ainda não estão disponíveis.

Então, **SALVE SEMPRE**, mas não compile, a menos que queira ver um monte de erros apontados pelo compilador. Vire a página e inicie a cola. São apenas seis páginas. Não terá mais nenhum texto explicativo, mas as rotinas estão ampla, geral, irrestrita e completamente documentadas. Mão a obra.

**implementation**

```

{$R *.lfm}

{ TForm1 }

uses
  Metodos, Solucao, Sobre, Windows, LCLIntf; // Windows e LCLIntf para usar GetSystemMetrics e OpenDocument

// Variáveis utilizadas nesta Unit - As Globais para várias Units estão definidas na Unit Metodos
var
  EFH, EFV, ESH, ESV: Byte; // Valores para Grade que são diferentes em diferentes versões do Windows
  DRH, DCW: Integer;
  GLX: Boolean;           // GLX = Grade com Linha extra para entrada de x inicial em Método Iterativo
  n_atual: Integer;
  bCopia, xIniCopia: array[1..20] of string;

procedure TForm1.FormCreate(Sender: TObject);
begin
  // Hint para vários componentes
  SpinEdit1.Hint := 'Entre com o número de variáveis ou' #13' cutuque para cima e para baixo para' #13+
    'definir o número de variáveis,' #13' que deve ser entre 3 e 20.';
  SpinEdit2.Hint := 'Entre o número máximo de iterações' #13' ou cutuque para cima e para baixo para' #13+
    'definir com o numero máximo de iterações,' #13' que deve ser entre 10 e 100.';
  FloatSpinEdit1.Hint := 'Entre com a tolerância e ou' #13' cutuque para cima e para baixo para' #13+
    'definir a tolerância e, que deve' #13'ser entre 0,00001 e 0,001.';
  SpeedButton1.Hint := 'Cria grade para entrada de dados.';
  SpeedButton2.Hint := 'Limpa grade.';
  SpeedButton3.Hint := 'Resolve o sistema.';
  SpeedButton4.Hint := 'Apresenta o arquivo de ajuda.';
  SpeedButton5.Hint := 'Informações sobre direitos e autoria.';
  RadioButton1.Hint := 'Método de Gauss com escalonamento simples.';
  RadioButton2.Hint := 'Método de Gauss com pivotamento' #13' parcial: permutação entre linhas.';
  RadioButton3.Hint := 'Método de Gauss com pivotamento total:' #13' permutação entre linhas e entre colunas.';
  RadioButton4.Hint := 'Método de Gauss compacto.';
  RadioButton5.Hint := 'Método de Decomposição L.U.';
  RadioButton6.Hint := 'Método de Cholesky.';
  RadioButton7.Hint := 'Método de Jacobi - Richardson.';
  RadioButton8.Hint := 'Método de Gauss-Seidel.';
  CheckBox2.Hint := 'Com o Método escolhido,' #13' calcula o sistema.';
  CheckBox3.Hint := 'Com o Método escolhido,' #13' calcula o determinante.';
  CheckBox4.Hint := 'Com o Método escolhido,' #13' calcula a matriz inversa.';

  // Medidas para moldura e scrollbar da Grade (são diferentes para cada versão do Windows)
  EFH := GetSystemMetrics(SM_CYBORDER)+GetSystemMetrics(SM_CFYXEDFRAME); // Espessura Frame Horizontal
  EFV := GetSystemMetrics(SM_CXBORDER)+GetSystemMetrics(SM_CFYXEDFRAME); // Espessura Frame Vertical
  ESH := GetSystemMetrics(SM_CYHSCROLL); // Espessura Scroll Horizontal
  ESV := GetSystemMetrics(SM_CXVSCROLL); // Espessura Scroll Horizontal (não utilizado neste programa)

  // Medidas da altura da linha e comprimento da coluna de cada casela
  // Mesmo definindo os valores no projeto, eles mudam de acordo com a versão do Windows
  // Forçar altura da casela ser a mesma em diferentes versões do Windows
  StringGrid1.DefaultRowHeight := 20;
  DRH := StringGrid1.DefaultRowHeight; // Default Row Height
  DCW := StringGrid1.DefaultColWidth; // Default Col Width
end;

procedure TForm1.FormShow(Sender: TObject);
begin
  n_atual := 0; // Variável para verificar se grade altera de tamanho
  n := SpinEdit1.Value; // Valor inicial de n
  ItMax := SpinEdit2.Value; // Valor inicial de ItMax
  Epsilon := FloatSpinEdit1.Value; // Valor inicial de Epsilon
  SpeedButton1.Click; // Dimensionar grade com tamanho (A|b) = 3x4
  RadioButtonMetodosDirectosClick(Self); // Habilitar ou desabilitar Opções Adicionais
  CheckBoxOpcoesClick(Self); // Valores para CalculaSistema, CalculaDeterminante e CalculaInversa
end;

procedure TForm1.SpinEdit1Exit(Sender: TObject);
begin
  n := SpinEdit1.Value;
  SpeedButton1.Click;
end;

procedure TForm1.SpinEdit1KeyPress(Sender: TObject; var Key: char);
begin
  if Key = #13 then
  begin
    Key := #0;
    SpeedButton1.Click;
  end;
end;

```

```

procedure TForm1.SpinEdit2Exit(Sender: TObject);
begin
  l1Max := SpinEdit2.Value;
end;

procedure TForm1.FloatSpinEdit1Exit(Sender: TObject);
begin
  Epsilon := FloatSpinEdit1.Value;
end;

procedure TForm1.RadioButtonMetodosDeterminantesClick(Sender: TObject);
begin
  // Evento onClick para RadioButton1 até RadioButton6
  // Só vou calcular a Inversa com Decomposição A = LU
  CheckBox3.Enabled := True; // Determinante com todos os métodos
  CheckBox4.Enabled := RadioButton5.Checked; // Inversa só com LU
  // Executa a rotina do SpeedButton1 para verificar se grade está correta
  SpeedButton1.Click;
  // Executa a rotina CheckBoxOpcoesClick para rever os valores das
  // variáveis CalculaSistema, CalculaDeterminante e CalculaInversa
  CheckBoxOpcoesClick(Self);
end;

procedure TForm1.RadioButtonMetodosIterativosClick(Sender: TObject);
begin
  // Evento onClick para RadioButton7 e RadioButton8
  // Não calcula Determinante nem Inversa, então tem que calcular Sistema (lógico)
  CheckBox2.Checked := True;
  CheckBox3.Enabled := False;
  CheckBox4.Enabled := False;
  // Executa a rotina do SpeedButton1 para verificar se grade está correta
  SpeedButton1.Click;
  // Executa a rotina CheckBoxOpcoesClick para rever os valores das
  // variáveis CalculaSistema, CalculaDeterminante e CalculaInversa
  CheckBoxOpcoesClick(Self);
end;

procedure TForm1.CheckBoxOpcoesClick(Sender: TObject);
begin
  // Evento onClick para CheckBox2 até CheckBox4
  CalculaSistema := CheckBox2.Checked and CheckBox2.Enabled;
  CalculaDeterminante := CheckBox3.Checked and CheckBox3.Enabled;
  CalculaInversa := CheckBox4.Checked and CheckBox4.Enabled;
end;

procedure TForm1.StringGrid1SelectCell(Sender: TObject; aCol, aRow: Integer;
  var CanSelect: Boolean);
begin
  // Impedir que o usuário digite dados na última posição da grade em Métodos Iterativos
  if GLX then
    with StringGrid1 do
      if (aCol = RowCount - 1) and (aRow = RowCount - 1) then
        CanSelect := False;
end;

procedure TForm1.SpeedButton1Click(Sender: TObject);
var
  k: Integer;
begin
  // Verificar se será Método Iterativo
  GLX := RadioButton7.Checked or RadioButton8.Checked;

  // Para ser legal com o usuário, copiar o vetor b e o vetor xini, se existir,
  // para restaurar para a nova posição após modificar o tamanho da grade
  // Além disto, deve-se limpar as posições atuais de b e xini
  with StringGrid1 do
    begin
      for k := 1 to n_atual do
        begin
          bCopia[k] := Cells[n_atual + 1, k];
          Cells[n_atual + 1, k] := '';
        end;
      // Se GLX foi alterado no início da rotina, verificar se era GLX, verificando número de linhas
      if GLX and (RowCount = n_atual + 2) then // É Método Iterativo
        for k := 1 to n_atual do
          begin
            xiniCopia[k] := Cells[k, n_atual + 1];
            Cells[k, n_atual + 1] := '';
          end;
    end;
end;

```

```

// Rever valor de n
n := SpinEdit1.Value;
// Se alterou n, redimensionar grade
if n <> n_atual then
  with StringGrid1 do
    begin
      // Altera o tamanho da grade
      RowCount := n+1; // Linha 0 de texto + n linhas para matriz A + 1 linha se Método Iterativo
      ColCount := n+2; // Coluna 0 de texto + n colunas para matriz A + 1 coluna para vetor b
      n_atual := n;
      // Escrever títulos da grade
      Cells[0,0] := 'A';
      for k := 1 to n do
        begin
          Cells[k,0] := IntToStr(k);
          Cells[0,k] := IntToStr(k);
        end;
      Cells[n+1,0] := 'b';
      // Dimensionar a grade, lembrando que para diferentes versões do Windows, há diferentes valores
      // para borda, scrollbar e até altura da casela, mesmo que seja criada com tamanho determinado
      Width := DCW*(n+2) + EFV;
      Height := DRH*(n+1) + EFH;
      if n > 7 then begin Width := DCW*9 + EFV; Height := Height + ESH; end;
      if n > 8 then begin Width := DCW*9 + EFV + ESV; Height := DRH*9 + EFH + ESH; end;
      SetFocus;
    end;
    // Com n alterado ou não, verificar se o tipo de método foi alterado
    // Verificar linha extra para entrada de dados de métodos iterativos
  with StringGrid1 do
    if GLX then // Se método iterativo, acrescentar linha extra se não houver
      begin
        if RowCount = n+1 then // Não tem linha Extra (acrescentar)
          begin
            RowCount := RowCount+1;
            if n < 8 then
              Height := Height+DRH;
            Cells[0,RowCount-1] := 'x inicial';
            if n > 7 then
              Width := DCW*9 + EFV + ESV;
          end;
        end
      else // Se método direto, remover linha extra se houver
        begin
          if RowCount = n+2 then // Tem linha Extra (remover)
            begin
              RowCount := RowCount-1;
              Width := DCW*(n+2) + EFV;
              Height := DRH*(n+1) + EFH;
              if n > 7 then begin Width := DCW*9 + EFV; Height := Height + ESH; end;
              if n > 8 then begin Width := DCW*9 + EFV + ESV; Height := DRH*9 + EFH + ESH; end;
            end;
          end;
        // Para ser legal com o usuário, restaurar o vetor b e o vetor xIni, se existir,
        // que foram copiados antes de modificar o tamanho da grade, para a nova posição
        for k := 1 to n do
          begin
            StringGrid1.Cells[n+1,k] := bCopia[k];
            if GLX then
              StringGrid1.Cells[k,n+1] := xIniCopia[k];
          end;
        end;
      procedure TForm1.SpeedButton2Click(Sender: TObject);
      var
        i, j: Integer;
      begin
        // Limpar Grade
        for i := 1 to n do
          for j := 1 to n+1 do
            StringGrid1.Cells[j,i] := '';
      end;
      procedure TForm1.SpeedButton3Click(Sender: TObject);
      var
        i, j, k: Integer;
      begin
        // Para não cansar o usuário, preencher caselas vazias com zeros
        if CheckBox1.Checked then // Usuário marcou a opção preencher com zeros

```

```

begin
  for i := 1 to n do
    for j := 1 to n do
      if (Trim(StringGrid1.Cells[j, i])) = '' then
        StringGrid1.Cells[j, i] := '0';
  // Em geral há poucos nulos no vetor b, mas se o usuário for preguiçoso, ...
  for i := 1 to n do
    if (Trim(StringGrid1.Cells[n+1, i])) = '' then
      StringGrid1.Cells[n+1, i] := '0';
  // Em geral, quando há ponto inicial, nem sempre são nulos, mas se o usuário for preguiçoso, ...
  if GLX then
    for j := 1 to n do
      if (Trim(StringGrid1.Cells[j, n+1])) = '' then
        StringGrid1.Cells[j, n+1] := '0';
end;
// Ler os dados literais da grade e transformá-los em números mas matrizes
for i := 1 to n do
begin
  for j := 1 to n do
    try
      A[i, j] := StrToIntFloat(StringGrid1.Cells[j, i]);
    except
      MessageDlg('O valor de A['+IntToStr(i)+','+IntToStr(j)+'] não é um número real válido', mtError, [mbOk], 0);
      StringGrid1.Col := j;
      StringGrid1.Row := i;
      StringGrid1.SetFocus;
      Exit;
    end;
    try
      b[i] := StrToIntFloat(StringGrid1.Cells[n+1, i]);
    except
      MessageDlg('O valor de b['+IntToStr(i)+'] não é um número real válido', mtError, [mbOk], 0);
      StringGrid1.Col := n+1;
      StringGrid1.Row := i;
      StringGrid1.SetFocus;
      Exit;
    end;
  end;
// Se a grade tem linha extra (Métodos iterativos), existe ponto inicial xini
if GLX then
  for j := 1 to n do
    try
      xini[j] := StrToIntFloat(StringGrid1.Cells[j, n+1]);
    except
      MessageDlg('O valor de x['+IntToStr(j)+'] não é um número real válido', mtError, [mbOk], 0);
      StringGrid1.Col := j;
      StringGrid1.Row := n+1;
      StringGrid1.SetFocus;
      Exit;
    end;
// Se o usuário não escolheu o que quer ...
if not (CalculaSistema or CalculaDeterminante or CalculaInversa) then
begin
  MessageDlg('Não foi solicitado resolver o sistema'+#10+'nem calcular o determinante'+#10+
             'nem calcular a inversa.'+#10+'Então, nada será feito.', mtInformation, [mbOk], 0);
  Exit;
end;
// Fechar a forma 2 e só reabrir se a solução existir (Erro = Falso)
Form2.Close;
// Definir Erro = Verdade - Se o método escolhido falhar, Erro será Verdade
// Se o método escolhido terminar corretamente será atribuído, antes de retornar, Erro = Falso
Erro := True;
if RadioButton1.Checked then Gauss;
if RadioButton2.Checked then GaussPivotamentoParcial;
if RadioButton3.Checked then GaussPivotamentoTotal;
if RadioButton4.Checked then GaussCompacto;
if RadioButton5.Checked then DecomposicaoLU;
if RadioButton6.Checked then Cholesky;
if RadioButton7.Checked then JacobiRichardson;
if RadioButton8.Checked then GaussSeidel;
if Erro then
  Exit;
// Se não teve erro, basta dar a resposta
with Form2 do          // Tudo que segue é para forma 2 que apresenta a solução
begin
  // Ajustar o tamanho da grade de solução
  with StringGrid1 do

```

```

begin
  RowCount := n+2;      // Linha 0 de texto + n linhas para matriz A + 1 linha para vetor x Solução
  ColCount := n+2;       // Coluna 0 de texto + n colunas para matriz A + 1 coluna para vetor b
  for i := 1 to n do
    for j := 1 to n+1 do
      Cells[j,i] := '';
  Cells[0,0] := 'A';
  for k := 1 to n do
  begin
    Cells[k,0] := IntToStr(k);
    Cells[0,k] := IntToStr(k);
  end;
  Cells[n+1,0] := 'b';
  Cells[0,n+1] := 'x';
  Width := DCW*(n+2) + EFV;
  Height := DRH*(n+2) + EFH;
  if n > 7 then begin Width := DCW*9 + EFV + ESV; Height := DRH*9 + EFH + ESH; end;
end;
// Ajustar o tamanho da grade da inversa (1 linha e 1 coluna menos que a grade do sistema)
with StringGrid2 do
begin
  RowCount := n+1;      // Linha 0 de texto + n linhas para matriz A
  ColCount := n+1;       // Coluna 0 de texto + n colunas para matriz A
  for i := 1 to n do
    for j := 1 to n do
      Cells[j,i] := '';
  Cells[0,0] := 'A';
  for k := 1 to n do
  begin
    Cells[k,0] := IntToStr(k);
    Cells[0,k] := IntToStr(k);
  end;
  Width := DCW*(n+1) + EFV;
  Height := DRH*(n+1) + EFH;
  if n > 8 then begin Width := DCW*9 + EFV + ESV; Height := DRH*9 + EFH + ESH; end;
end;
// Apresentar Solução do Sistema
if CalculaSistema then
begin
  GroupBox1.Caption := 'Solução do sistema';
  if GLX then
    GroupBox1.Caption := 'Solução do sistema - Iterações = '+IntToStr(Iteracao);
  with StringGrid1 do
  begin
    for i := 1 to n do
    begin
      for j := 1 to n do
        Cells[j,i] := FloatToStr(A[i,j]);
        Cells[j+1,i] := FloatToStr(b[i]);
      end;
      for j := 1 to n do
        Cells[j,n+1] := FloatToStr(x[j]);
      // Limpar a casela da última posição da grade (pode ter valor de solução anterior)
      Cells[n+1,n+1] := '';
    end;
  end
  else
  begin
    GroupBox1.Caption := 'Solução do sistema - Não foi solicitado';
    for i := 1 to n+1 do
      for j := 1 to n+1 do
        StringGrid1.Cells[j,i] := '';
  end;
// Apresentar Determinante
if CalculaDeterminante then
  Label1.Caption := 'Determinante = '+FloatToStr(Determinante)
else
  Label1.Caption := 'Determinante - Não foi solicitado';
// Apresentar Inversa da Matriz
if CalculaInversa then
begin
  GroupBox2.Caption := 'Matriz inversa';
  with StringGrid2 do
    for i := 1 to n do
      for j := 1 to n do
        Cells[j,i] := FloatToStr(AlInv[i,j]);
end;

```

```

else
begin
  GroupBox2.Caption := 'Matriz inversa - Não foi solicitado';
  for i := 1 to n do
    for j := 1 to n do
      StringGrid2.Cells[j, i] := '';
end;
// Tornar Form2 visível
Show;
end;
end;

procedure TForm1.SpeedButton4Click(Sender: TObject);
begin
  // Ajuda
  if not OpenDocument('SistemasLineares.chm') then
    MessageDlg('O arquivo de ajuda #10' + 'SistemasLineares.chm' + '#10' + 'não foi encontrado.', mtInformation, [mbOk], 0);
  // Se quiser o título da janela em língua pátria, substitua a linha acima pela linha abaixo
  //MessageDlg('Informação', 'O arquivo de ajuda #10' + 'SistemasLineares.chm' + '#10' + 'não foi encontrado.', mtInformation, [mbOk], 0);
end;

procedure TForm1.SpeedButton5Click(Sender: TObject);
begin
  // Sobre
  Form3.ShowModal();
end;

```

Enganei-me. Foram apenas 5,37 páginas. Agora só falta fazer a sua parte que é criar as rotinas para calcular a solução do sistema, determinante e inversa. Vai lá uma ajuda, antes que a Patricinha venha recheiar a mochila.

Pressione a aba Metodos, pressione F12 para ver a fórmula, pare para pensar, lembre-se que esta Unit não tem fórmula, vá descansar e volte. Após voltar, comece pela definição das variáveis que serão utilizadas nesta Unit e na Unit Principal, defina as rotinas e siga com a implementação. Escreva o trecho que segue, em todas as rotinas, de Gauss até GaussSeidel, para ter um programa compilável e executável, para testar enquanto desenvolve as rotinas.

```

MessageDlg('Ainda não foi implementado', mtInformation, [mbOk], 0);
Erro := True;

```

Mãos a obra.

```

unit Metodos;
{$mode objfpc}{$H+}

interface

uses
  Classes, SysUtils, Dialogs;

var
  n, Iteracao, ItMax: Integer;
  Epsilon, Determinante: Real;
  Erro, CalculaSistema, CalculaDeterminante, CalculaInversa: Boolean;
  // A, L, U, G, AlInv: array of array of Real;
  // b, x, xIni, y, bj, xj: array of Real;
  // Poderia utilizar matrizes dinâmicas, mas como o programa foi
  // limitado a sistemas de até 20x20, vou usar matrizes estáticas
  A, L, U, G, AlInv: array [1..20, 1..20] of Real;
  b, x, xIni, y, bj, xj: array [1..20] of Real;

procedure Gauss;
procedure GaussPivoteamentoParcial;
procedure GaussPivoteamentoTotal;
procedure GaussCompacto;
procedure DecomposicaoLU;
procedure Cholesky;
procedure Jacobi Richardson;
procedure GaussSeidel;

```

**implementation**

```
procedure Gauss;
var
  i, j, k: Integer;
  m, soma: Real;
begin
  Erro := True; // Já foi definido em Principal - só está também aqui para você se lembrar
```

**// Triangularizando a matriz (Gauss)**

```
for j := 1 to n-1 do
```

```
•••
```

Escreva o trecho de programa que triangulariza a Matriz A.

Lembre-se de utilizar try except end ao calcular  $m = a_{ij}/a_{ii}$  e se houver exceção informar que o determinante é nulo e interromper.

```
•••
```

**// Solução por Retrosubstituição**

```
•••
```

Escreva o trecho de programa que resolve o sistema.

Lembre-se de utilizar try except end ao calcular  $x_n = b_n/a_{nn}$  e  $x_i = (b_i - \text{soma})/a_{ii}$  e se houver exceção informar e interromper.

Ao terminar, atribua Erro = Falso, pois na chamada da rotina, na forma Principal, foi atribuído Erro = Verdade

```
•••
```

```
Erro := False;
```

**// Determinante**

```
if not CalculaDeterminante then
```

```
  Exit;
```

```
•••
```

Se não é para calcular o determinante, interromper.

Se for para calcular, calcule o Determinante que é o produto dos elementos da diagonal de A que já foi triangularizada.

```
•••
```

```
end;
```

```
procedure GaussPivotamentoParcial;
```

```
begin
```

```
  MessageDlg('Ainda não foi implementado', mtInformation, [mbOK], 0);
```

```
  Erro := True;
```

```
end;
```

```
procedure GaussPivotamentoTotal;
```

```
begin
```

```
  MessageDlg('Ainda não foi implementado', mtInformation, [mbOK], 0);
```

```
  Erro := True;
```

```
end;
```

```
procedure GaussCompacto;
```

```
begin
```

```
  MessageDlg('Ainda não foi implementado', mtInformation, [mbOK], 0);
```

```
  Erro := True;
```

```
end;
```

```
procedure DecomposicaoLU;
```

```
var
```

```
  i, j, k: Integer;
```

```
  soma: Real;
```

```
begin
```

```
  Erro := True; // Já foi definido em Principal - só está também aqui para você se lembrar
```

**// Decompondo A em L.U**

```
for k := 1 to n do
```

```
  L[k, k] := 1;
```

```
•••
```

Escreva o trecho de programa que decompõe A em L.U.

Se você utilizar matrizes L e U deverá, no início, preencher  $L_{kk}$  com a unidade, como apresentado acima.

Se você utilizar apenas a matriz A, armazenando os valores de  $l_{ij}$  e  $u_{ij}$  na própria matriz A, remova o trecho acima.

Lembre-se de utilizar try except end ao calcular  $l_{ji} = a_{ji} - \text{soma}/a_{ii}$  e se houver exceção informar que o determinante é nulo e interromper.

```
•••
```

**// Solução de L.y = b por Substituição**

•••

*Escreva o trecho de programa que calcula y com a relação L.y = b.*

*Se você utilizar apenas a matriz A, armazenando os valores de  $l_{ij}$  e  $u_{ij}$  na própria matriz A, lembre-se que  $l_{ii}$  não está armazenado em A e deve ser substituído por 1, no programa.*

*Lembre-se de utilizar try except end ao calcular  $y_1 = b_1/l_{11}$  e  $y_i = (b_i - \text{soma})/l_{ii}$  e se houver exceção informar e interromper.*

•••

**// Solução de U.x = y por Retrosubstituição**

•••

*Escreva o trecho de programa que calcula x com a relação U.x = y.*

*Lembre-se de utilizar try except end ao calcular  $x_n = y_n/u_{nn}$  e  $x_i = (y_i - \text{soma})/u_{ii}$  e se houver exceção informar e interromper.*

•••

•••

*Se você utilizar apenas a matriz A para armazenar os valores de L e U, remova o trecho abaixo.*

*Se você utilizar as matrizes L e U, o trecho abaixo armazena L e U em A, para apresentar a solução da decomposição.*

*Como lembrete, isto não foi necessário na rotina Gauss, pois nela, A é triangularizada na própria matriz A.*

•••

**// Colocando L e U em A**

```
for i := 1 to n do
    for j := i to n do
        A[i, j] := U[i, j];
for i := 2 to n do
    for j := 1 to i - 1 do
        A[i, j] := L[i, j];
```

**// Determinante**

```
if CalculaDeterminante then
begin
```

•••

*Se não é para calcular o determinante, interromper.*

*Se for para calcular, calcule o Determinante que é o produto dos elementos da diagonal de U ou de A, que já contém o valor de U.*

*Note que na rotina Gauss, se não é para calcular o Determinante, a rotina termina. Neste caso, não pode terminar, pois a seguir é feito o cálculo da inversa.*

*Você pode fazer seu programa Principal diferente e alterar as rotinas desta Unit Metodos.*

•••

end;

**// Inversa**

```
if CalculaInversa then
begin
```

```
// Cálculo da Inversa (resolver o sistema n vezes, com vetores b nulos, exceto na posição j)
// Os vetores b serão chamados de bj, para não perder o valor de b do sistema
// O vetor x será chamado de xj, para não perder o valor de x do sistema
```

```
for k := 1 to n do
```

```
begin
```

```
// Preencher o vetor b referente à coluna j da Identidade
```

```
FillChar(bj, SizeOf(bj), 0);
```

```
bj[k] := 1;
```

```
// Solução de L.y = b por Substituição
```

•••

*Repetir o mesmo trecho de programa anterior que calcula y com a relação L.y = b, com o vetor bj.*

*Lembre-se de utilizar try except end como no trecho anterior.*

•••

```
// Solução de U.x = y por Retrosubstituição
```

•••

*Escreva o mesmo trecho de programa anterior que calcula x com a relação U.x = y, com o vetor xj.*

*Lembre-se de utilizar try except end como no trecho anterior.*

•••

```
// Preencher as colunas da matriz inversa de A com os vetores xj
```

```
for i := 1 to n do
```

```
    Alnv[i, k] := xj[i];
```

```
end;
```

end;

```
Erro := False;
```

```
end;
```

```

procedure Cholesky;
var
  i, j, k: Integer;
  soma: Real;
begin
  Erro := True; // Já foi definido em Principal - só está também aqui para você se lembrar

  // Verificar se A é simétrica
  for i := 1 to n-1 do
    for j := i+1 to n do
      if (A[i,j] <> A[j,i]) then
        begin
          MessageDlg('A matriz A não é Simétrica.', mtInformation, [mbOk], 0);
          Exit;
        end;

  // Decompondo A em G.GT

  // Cálculo de G
  •••
Escreva o trecho de programa que calcula G.
Você sabe que pode utilizar uma matriz chamada G ou armazenar os valores calculados na própria matriz A.

Lembre-se de utilizar try except end ao calcular  $G_{kk} = Raiz(A_{kk}-soma)$  e se houver exceção informar que a matriz A não é Definida Positiva.
  •••

  // Preenchendo a GT
  for i := 1 to n-1 do
    for j := i+1 to n do
      G[i,j] := G[j,i];
  •••
Se você utilizar apenas a matriz A, o trecho acima deve ser modificado.
  •••

  // Solução de G.y = b por Substituição
  •••
Similar ao Método de Decomposição A = L.U.
  •••

  // Solução de GT.x = y por Retrosubstituição
  •••
Similar ao Método de Decomposição A = L.U.
  •••

  // Colocando G em A
  for i := 1 to n do
    for j := 1 to n do
      A[i,j] := G[i,j];

  Erro := False;

  // Determinante
  if not CalculaDeterminante then
    Exit;
  •••
Se não é para calcular o determinante, interromper.
Se for para calcular, calcule o Determinante que é o produto dos elementos da diagonal de A ao quadrado, que já contém o valor de G.
  •••

end;

function Distancia: Extended;
var
  Numerador, Denominador: Extended;
  i: Integer;
begin
  Numerador := 0;
  Denominador := 0;
  for i := 1 to n do
    begin
      Numerador := Numerador+Sqr(x[i]-x[ini[i]]);
      Denominador := Denominador+Sqr(x[i]);
    end;
  Result := Sqr(Numerador)/Sqr(Denominador);
end;

```

•••

Fessor, explique essa rotina do final da página anterior.

Não é que eu não tenha entendido, mas tem colegas que ficaram olhando, olhando, olhando e não entenderam nada.

Querida Patricinha (ô guria chata), trata-se da rotina que calcula a norma da diferença relativa entre dois vetores.

Para isto, utilizando a Norma Euclidiana, foi calculado o erro relativo e a rotina foi chamada de Distancia. Poderia ser chamada de DistanciaRelativa. Ela será utilizada nas rotinas JacobiRichardson e GaussSeidel, que são iterativas e servirá como teste de parada.

Fessor, não podia chamar Distância, com chapeuzinho no a?

E JacobiRichardson e GaussSeidel não ficariam melhor se fossem Jacobi-Richardson e Gauss-Seidel?

Querida Patty. Em programas, o tracinho significa subtração. Alguém sabe onde está o cartão vermelho?

•••

```
procedure Jacobi Richardson;
var
```

```
    i, j : Integer;
    soma, Maximo: Real ;
    CDL, CDC: Boolean;
```

```
begin
```

```
    Erro := True;
```

**// Verificar critério das linhas**

```
    Maximo := 0;
```

•••

Escreva o algoritmo para calcular o valor máximo do critério das linhas.

Se este máximo for menor que um, CDL é Verdade.

•••

```
    CDL := Maximo < 1;           // CDL é True se Maximo < 1
```

**// Verificar critério de colunas**

```
    Maximo := 0;
```

•••

Escreva o algoritmo para calcular o valor máximo do critério das colunas.

Se este máximo for menor que um, CDC é Verdade.

•••

```
    CDC := Maximo < 1;           // CDC é True se Maximo < 1
```

```
if not (CDL or CDC) then      // Se um dos critérios é True continua
```

```
begin                         // Pode-se tentar a solução, mas vou interromper
```

```
    MessageDlg('A matriz não atende ao Critério das' +#10+'Linhas nem '+
               'ao Critério das Colunas.', mtInformation, [mbOK], 0);
```

```
    Exit;
```

```
end;
```

**// Jacobi usa xlni para calcular x e o vetor xlni já foi lido da grade e está na memória**

**// Iterações do método**

```
Iteracao := 0;
```

```
while True do
```

```
begin
```

•••

Escreva o algoritmo descrito na seção **Método de Jacobi**, utilizando a rotina Distancia para calcular a distância entre a solução x e a anterior xlni, interrompendo quando ocorrer uma das situações:

- Iteracao > ItMax e Erro continua Verdade

- Distancia < Epsilon e Erro deve receber o valor Falso

•••

```
end;
```

```
end;
```

```
procedure GaussSeidel ;
```

```
var
```

```
    i, j : Integer;
```

```
    Soma, Maximo: Real ;
```

```
    CDL, CDS: Boolean;
```

```
    Beta: array [1..20] of Real ;
```

```
begin
```

```
    Erro := True;
```

**// Verificar critério das linhas**

```
    Maximo := 0;
```

•••

Escreva o algoritmo para calcular o valor máximo do critério das linhas.

Se este máximo for menor que um, CDL é Verdade.

•••

```
    CDL := Maximo < 1;           // CDL é True se Maximo < 1
```

Cuidado com o ponto de partida.  
Neste método é xlni. Compare com o Método de Gauss-Seidel.

## // Verificar critério de Sassenfeld

•••

Escreva o algoritmo para calcular o valor máximo do critério de Sassenfeld.

Se este máximo for menor que um, CDS é Verdade.

•••

CDS := Maximo &lt; 1; // CDS é True se Maximo &lt; 1

if not (CDL or CDS) then // Se um dos critérios é True continua

begin // Pode-se tentar a solução, mas vou interromper

MessageDI g('A matriz não atende ao Critério das '+#10+' Linhas nem '+  
'ao Critério de Sassenfeld.', mtInformation, [mbOk], 0);

Exit;

end;

// Seidel usa x para calcular novo x e, então, deve-se iniciar x com xIni já foi lido da grade e está na memória

for i := 1 to n do

x[i] := xIni[i];

// Iterações do método

Iteracao := 0;

while True do

begin

•••

Escreva o algoritmo descrito na seção **Método de Gauss-Seidel**, utilizando a rotina Distancia para calcular a distância entre a solução x e a anterior xIni, interrompendo quando ocorrer uma das situações:

- Iteracao &gt; ItMax e Erro continua Verdade

- Distancia &lt; Epsilon e Erro deve receber o valor Falso

•••

end;

end;

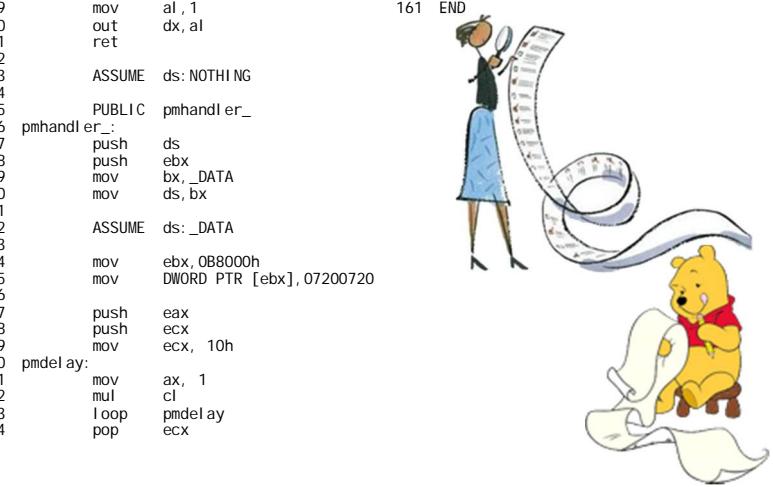
end.

Que maravilha. Você fez um programa com mais de mil linhas. Para encerrar o Capítulo, segue um programa com menos de 160. Se você perder seu tempo lendo isto, traduza e me informe.

```

001 . 286          063    EXTRN _com_port: WORD      125
002             064    EXTRN _com_id: BYTE      126      mov   BYTE PTR [ebx+2], 'P'
003 _TEXT16 SEGMENT BYTE PUBLIC 'CODE'  065    EXTRN _l_owmem_seg: WORD      127
004     ASSUME cs:_TEXT16   066    EXTRN _pm_count: DWORD      128      push  edx
005     maxbuf EQU 1024   067    _DATA ENDS      129      mov   bx, [_com_port]
006     PUBLIC rmhandl er_, _com_port_low 068    _TEXT SEGMENT BYTE PUBLIC 'CODE'  130      lea   dx, [bx+2]
007     rmhandl er_:      069    ASSUME cs:_TEXT, ds:_DATA  131      in   al , dx
008     push ds           070    PUBLIC com_ini_t_  132      mov   dx, bx
009     push bx           071    com_ini_t_      133      in   al , dx
010     mov  bx, 0B800h      072    PUBLIC com_ini_t_  134      inc  [_pm_count]
011     mov  ds, bx         073    com_ini_t_      135      mov   bx, [_l_owmem_seg]
012     sub  bx, bx         074    mov  ax, 0F3h      136      mov   ds, bx
013     mov  WORD PTR [bx], 0720h  075    mov  dl , [_com_id]  137
014     mov  WORD PTR [bx+2], 0720h  076    sub  dh, dh      138
015     sub  bx, bx         077    dec  dx      139
016     dec  dx             078    int  14h      140      ASSUME ds:NOTHING
017     int  14h             079    mov  bx, [_com_port]  141      mov   ebx, ds:[128]
018     lea   dx, [bx+5]      080    lea   dx, [bx+5]  142      mov   ds:[132+ebx], al
019     in   al , dx         081    in   al , dx      143      inc  ebx
020     mov  cx, 10h          082    lea   dx, [bx+4]  144      cmp  ebx, maxbuf+1
021 rmdelay:        083    in   al , dx      145      jl   no_ov2
022     mov  ax, 1            084    or   al , 8       146      sub  ebx, ebx
023     mul  cl             085    out  dx, al      147      no_ov2:
024     loop rmdelay        086    lea   dx, [bx+2]  148      mov  ds:[128], ebx
025     pop  ax             087    in   al , dx      149
026     pop  cx             088    mov  dx, bx      150      mov  dx, 020h
027     pop  cx             089    in   al , dx      151      mov  al , dl
028     mov  BYTE PTR [bx], 'R' 090    mov  dl , NOT 10h  152      out  dx, al
029             091    cmp  [_com_id], 1  153      pop  edx
030     db   0BBh           092    je   maskit      154      pop  eax
031 _com_port_low DW ?      093    mov  dl , NOT 8h  155      pop  ebx
032     push ax             094    maskit:      156      pop  ds
033     push dx             095    in   al , 21h      157      iretd
034     lea   dx, [bx+2]      096    and  al , dl      158
035     in   al , dx         097    out  21h, al      159 _TEXT ENDS
036     mov  dx, bx         098    lea   dx, [bx+1]  160
037     in   al , dx         099    mov  al , 1       161 END
038             100    out  dx, al
039     mov  bx, cs:[128]      101    ret
040     mov  cs:[132+bx], al  102
041     inc  bx             103     ASSUME ds:NOTHING
042     cmp  bx, maxbuf+1   104
043     jl   no_ov1          105     PUBLIC pmhandl er_
044     sub  bx, bx         106     pmhandl er_:
045 no_ov1:          107     push  ds
046     mov  cs:[128], bx    108     push  ebx
047             109     mov  bx, _DATA
048     mov  dx, 020h          110     mov  ds, bx
049     mov  al , dl          111
050     out  dx, al          112     ASSUME ds:_DATA
051     pop  dx             113
052     pop  ax             114     mov  ebx, 0B8000h
053     pop  bx             115     mov  DWORD PTR [ebx], 07200720
054     pop  ds             116
055     iret               117     push  eax
056     ASSUME cs:NOTHING   118     push  ecx
057             119     mov  ecx, 10h
058 _TEXT16 ENDS          120     pmdelay:
059             121     mov  ax, 1
060 . 386p              122     mul  cl
061             123     loop pmdelay
062 _DATA SEGMENT DWORD PUBLIC 'DATA'  124     pop  ecx

```

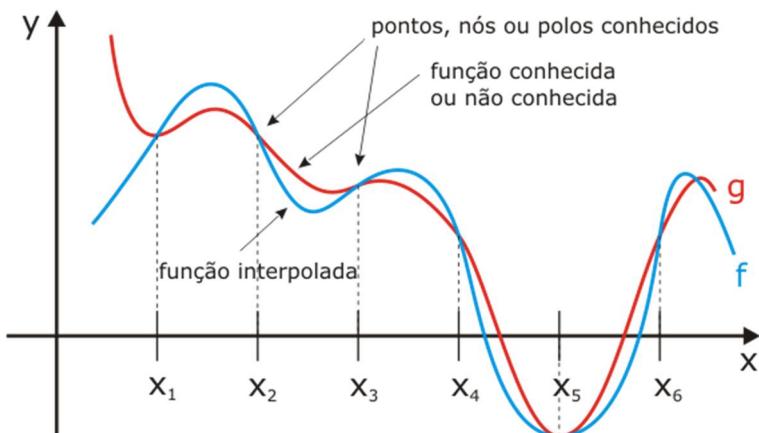


## 10- Interpolação

Interpolação é o ato de interpolar e, interpolar é composto de inter + polar. Pronto, é isto. :-)

Mas é isto mesmo, ou seja, colocar valores entre dois outros conhecidos. Então, dada uma nuvem de pontos, deseja-se conhecer uma equação que passe por estes pontos e forneça os valores intermediários não conhecidos.

Esta nuvem de pontos pode ser obtida experimentalmente ou de uma série histórica de dados ou de uma função conhecida. E lá vem pergunta: qual a vantagem de se interpolar uma função a pontos de uma função conhecida. E a resposta é: se a função conhecida for do mal e de difícil trato, uma nova função interpolada, do bem, será mais gentil ao ser manipulada, por exemplo, para calcular uma derivada ou para efetuar outro tipo de cálculo. Portanto, não se utiliza interpolação apenas para nuvem de pontos. É comum o uso de interpolação para pontos de uma função conhecida.



Então, o procedimento de interpolação determina uma função que passa pelos pontos conhecidos, permitindo calcular valores entre estes pontos, com erro que pode ser estimado.

Permite, também, calcular valores que estejam fora da região delimitada pelos pontos, mas neste caso, sem permitir que se façam considerações sobre o erro.

### Interpolação polinomial

A interpolação polinomial procura um polinômio que passa pelos pontos conhecidos. Dados  $n$  intervalos, pode-se determinar um polinômio de grau  $n$ . Para  $n$  intervalos, tem-se  $n+1$  pontos. Na prática, ninguém tenha 10 pontos disponíveis diz que tem 9+1 pontos, mas é assim que os milhões de textos tratam o assunto. O motivo é quase óbvio. Com um ponto, tem-se um polinômio de grau zero e, então o zero é um gentil e simpático ser presente nos cálculos. Na figura anterior, tem-se 5 intervalos e  $5+1=6$  pontos. Para usar esta notação, os pontos da figura serão rebatizados de  $x_0$  até  $x_5$ .

Então, dados  $n+1$  pontos, pode-se determinar um polinômio de grau  $n$ . Para 1 ponto,  $(x_0, y_0)$ , tem-se o polinômio  $y = y_0$ , ou seja, uma reta horizontal com cota  $y_0$ . Para 2 pontos,  $(x_0, y_0)$  e  $(x_1, y_1)$ , tem-se o polinômio  $y = (y_1 - y_0)/(x_1 - x_0) \cdot x + y_0 - (y_1 - y_0)/(x_1 - x_0) \cdot x_0$  ou seja, uma reta que passa pelos 2 pontos, que pode ser escrita como  $y = a \cdot x + b$ , com  $a = (y_1 - y_0)/(x_1 - x_0)$  e  $b = y_0 - a \cdot x_0$ . Para 3 pontos, uma equação do segundo grau e, assim por diante.

Para determinar a forma geral, deve-se partir da definição de interpolação, ou seja, determinar uma função polinomial de grau  $n$  que passe por  $n+1$  pontos. Então, com  $n+1$  pontos pode-se escrever as relações  $f(x_0) = g(x_0)$ ,  $f(x_1) = g(x_1)$ , ...,  $f(x_n) = g(x_n)$ , onde  $g(x_0) = y_0$ ,  $g(x_1) = y_1$ , ...,  $g(x_n) = y_n$  e são conhecidos.

Deve-se, então determinar um polinômio  $f(x) = p_n(x) = a_0 \cdot x^0 + a_1 \cdot x^1 + a_2 \cdot x^2 + \dots + a_n \cdot x^n$  ou, sem escrever o que é óbvio, mas sabendo que existem,  $p_n(x) = a_0 + a_1 \cdot x + a_2 \cdot x^2 + \dots + a_n \cdot x^n$ .

$$\begin{cases} a_0 + a_1 \cdot x_0 + a_2 \cdot x_0^2 + \dots + a_n \cdot x_0^n = y_0 \\ a_0 + a_1 \cdot x_1 + a_2 \cdot x_1^2 + \dots + a_n \cdot x_1^n = y_1 \\ a_0 + a_1 \cdot x_2 + a_2 \cdot x_2^2 + \dots + a_n \cdot x_2^n = y_2 \\ \vdots \\ a_0 + a_1 \cdot x_n + a_2 \cdot x_n^2 + \dots + a_n \cdot x_n^n = y_n \end{cases}$$

$$\left[ \begin{array}{cccc|c} 1 & x_0 & x_0^2 & \cdots & x_0^n & a_0 \\ 1 & x_1 & x_1^2 & \cdots & x_1^n & a_1 \\ 1 & x_2 & x_2^2 & \cdots & x_2^n & a_2 \\ \vdots & & & & & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^n & a_n \end{array} \right] \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}$$

A matriz A é conhecida como matriz de Vandermonde e, se os pontos forem distintos, seu determinante é não nulo e, então, o sistema tem solução única.

O algoritmo é tão simples que não vou deixar você fazer antes que eu o faça. Vou fazer já. Mas antes disto, vou pensar um pouco, para não fazer uma enorme confusão de nomes de variáveis. No sistema  $A \cdot x = b$ , a matriz A contém valores de x, o vetor b, valores de y e o vetor solução x fornecerá valores dos coeficientes a do polinômio. Baita confusão. Vou supor que exista uma rotina Gauss(n, A, x, b) que resolve sistemas do tipo  $A \cdot x = b$ . Vou criar uma matriz chamada Mx, um vetor chamado Vy e quero calcular um vetor chamado Vx. Pronto, agora ficou fácil.

Dados  $n \geq 1$  e  $n+1$  pontos  $(x_i, y_i)$ , com  $i = 0, 1, 2, \dots, n$

Verificar se há  $x_i = x_j$ , com  $i \neq j$

Se houver, então

Descartar um deles e reduzir n de uma unidade

Se  $n < 1$ , informar e interromper

**Montar matriz Mx e vetor Vy como segue, considerando índices a partir de 1, com ordem n+1**

Para  $i = 1, \dots, n+1$  faça

Para  $j = 1, \dots, n+1$  faça

$M_{ij} = x_{i-1}^{j-1}$  *Pode-se preencher a primeira coluna com 1 e a segunda com x, sem efetuar  $x^0$  e  $x^1$*

$Vy_i = y_{i-1}$

Gauss(n+1, Mx, Vx, Vy) *Resolver o sistema A \cdot x = b, executando a rotina*

Para  $i = 1, \dots, n+1$  faça

$a_{i-1} = Vx_i$  *Os coeficiente  $a_n, \dots, a_1$  do polinômio são  $Vx_1, \dots, Vx_{n+1}$  da solução do sistema*

Se gostou, ótimo. Se não, se vire, pois você já é um excelente programador e saberá o que fazer.

E, ao se virar, decida se em seu algoritmo n deve ser maior ou igual a 1 ou 2 ou 3. Pode ser que você considere que as equações de grau zero ou um não devam ser contempladas em seu algoritmo.

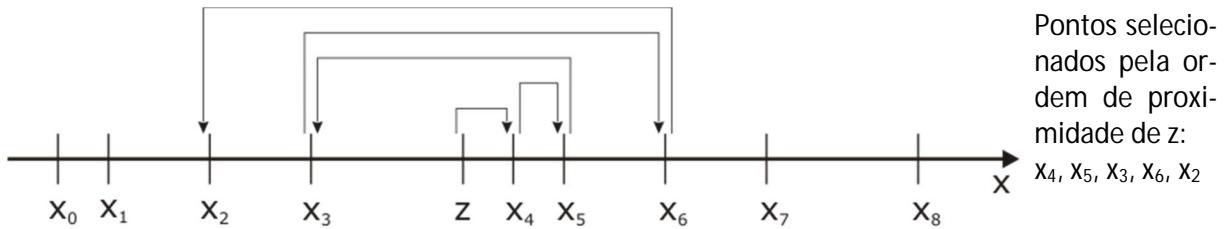
Se virando mais um pouco, o algoritmo matemático apresentado faz j variar de 1 a n+1 e seria mais conveniente variar de 3 até n+1, preenchendo as colunas 1 e 2 sem efetuar as potências  $x^0$  e  $x^1$ . Claro que isto só vale se n for maior ou igual a 2.

## Interpolação polinomial utilizando grau menor que n

Se para  $n+1$  pontos o desejo incontido do usuário for a obtenção de um polinômio de grau k, é necessário que se escolha  $k+1$  pontos adequados. Esses pontos serão tão mais adequados quanto mais próximos estiverem da região onde se usará o polinômio para estimar valores interpolados.

Então, deve-se ter um valor z centrado nesta região e, a partir de z, seleciona-se os  $k+1$  valores de x que mais se aproximam de z, como na figura que segue.

Considere que são dados 9 pontos e, com isto, é possível determinar um polinômio de grau 8. Contudo, deseja-se um polinômio de grau 4. Seleciona-se 5 dos 9 pontos, mais próximos de  $z$ .



Claro que você já fez o algoritmo mental. Só falta escrevê-lo.

### Método de Newton para interpolação polinomial

O método de Newton utiliza o conceito de diferenças divididas, já que Newton era amante das derivadas.

Considere  $f(x)$  uma função contínua,  $n+1$  vezes diferenciável e definida em  $n+1$  pontos distintos,  $x_0, x_1, \dots, x_n$ , de um intervalo  $[a, b]$ .

Define-se diferença dividida como:

$$\begin{aligned} \Delta^0 y_i &= f[x_i] = f(x_i) = y_i && \text{Ordem 0} \\ \Delta^1 y_i &= f[x_i, x_{i+1}] = \frac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i} = \frac{y_{i+1} - y_i}{x_{i+1} - x_i} && \text{Ordem 1} \\ \Delta^2 y_i &= f[x_i, x_{i+1}, x_{i+2}] = \frac{f[x_{i+1}, x_{i+2}] - f[x_i, x_{i+1}]}{x_{i+2} - x_i} = \frac{\Delta^1 y_{i+1} - \Delta^1 y_i}{x_{i+2} - x_i} && \text{Ordem 2} \\ &\dots \\ \Delta^n y_i &= f[x_i, x_{i+1}, x_{i+2}, \dots, x_n] = \frac{f[x_{i+1}, x_{i+2}, \dots, x_n] - f[x_i, x_{i+1}, x_{i+2}, \dots, x_{n-1}]}{x_n - x_i} = \frac{\Delta^{n-1} y_n - \Delta^{n-1} y_i}{x_n - x_i} && \text{Ordem } n \end{aligned}$$

Para simplicidade de visão e cálculo, podem-se tabelar os valores como segue.

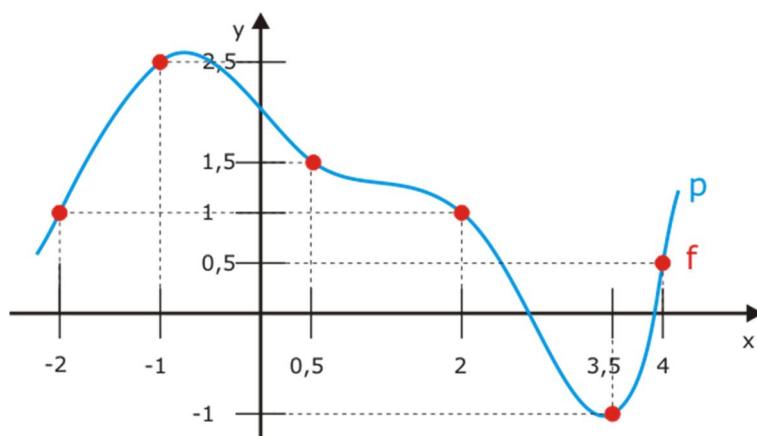
$x$	$\Delta^0 y_i$	$\Delta^1 y_i$	$\Delta^2 y_i$	$\Delta^3 y_i$	$\Delta^n y_i$
$x_0$	$y_0$				
$x_1$	$y_1$	$\Delta^1 y_0$			
$x_2$	$y_2$	$\Delta^1 y_1$	$\Delta^2 y_0$		
$x_3$	$y_3$	$\Delta^1 y_2$	$\Delta^2 y_1$	$\Delta^3 y_0$	
$x_4$	$y_4$	$\Delta^1 y_3$	$\Delta^2 y_2$	$\Delta^3 y_1$	$\Delta^n y_0$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	
$x_n$	$y_n$		$\Delta^1 y_{n-1}$	$\Delta^2 y_{n-2}$	

Computacionalmente, utilize o esquema que segue, ou seja, não deixe buracos na tabela. :-)

$x$	$\Delta^0 y_i$	$\Delta^1 y_i$	$\Delta^2 y_i$	$\Delta^3 y_i$	$\dots$	$\Delta^n y_i$
$x_0$	$y_0$	$\Delta^1 y_0$	$\Delta^2 y_0$	$\Delta^3 y_0$	$\dots$	$\Delta^n y_0$
$x_1$	$y_1$	$\Delta^1 y_1$	$\Delta^2 y_1$	$\Delta^3 y_1$	$\ddots$	
$x_2$	$y_2$	$\Delta^1 y_2$	$\Delta^2 y_2$	$\vdots$	$\ddots$	
$x_3$	$y_3$	$\Delta^1 y_3$	$\vdots$	$\Delta^3 y_{n-2}$		
$x_4$	$y_4$	$\vdots$	$\Delta^2 y_{n-1}$			
$\vdots$	$\vdots$	$\Delta^1 y_{n-1}$				
$x_n$	$y_n$					

Os exemplos numéricos que existem por aí são tão viciados que ajudam a atrapalhar. Segue um que poderá ajudar em seu treino algorítmico.

Considere  $f$  dada por uma nuvem de pontos e uma antevisão de uma função polinomial interpolada.



Dados os pontos, calcular as diferenças divididas.

$x$	$y$
-2	1
-1	2,5
0,5	1,5
2	1
3,5	-1
4	0,5

Seguem os cálculos para tabela manual e computacional.

Calculando com tabela manual.

$x$	$\Delta^0 y_i$	$\Delta^1 y_i$	$\Delta^2 y_i$	$\Delta^3 y_i$	$\Delta^4 y_i$	$\Delta^5 y_i$
-2	1					
-1	2,5	1,5				
0,5	1,5	-0,666667	-0,866667	0,244444		
2	1	-0,333333	0,111111	-0,098765	-0,062402	
3,5	-1	-1,333333	-0,333333	0,714286	0,162610	0,037502
4	0,5	3	2,166667			

Calculando com tabela computacional.

$x$	$\Delta^0 y_i$	$\Delta^1 y_i$	$\Delta^2 y_i$	$\Delta^3 y_i$	$\Delta^4 y_i$	$\Delta^5 y_i$
-2	1	1,5	-0,866667	0,244444	-0,062402	0,037502
-1	2,5	-0,666667	0,111111	-0,098765	0,162610	
0,5	1,5	-0,333333	-0,333333	0,714286		
2	1	-1,333333	2,166667			
3,5	-1	3				
4	0,5					

Certamente você já escreveu o algoritmo, o programa e testou. Ótimo. Só falta determinar o polinômio.

Considere  $f(x)$  uma função contínua e definida em  $n+1$  pontos distintos,  $x_0, x_1, \dots, x_n$ , de um intervalo  $[a, b]$ .

Determinadas as diferenças divididas e considerando que os pontos pertencem a um polinômio de grau  $n$ , tem-se que:

$$\Delta^1 y = \frac{p_n(x) - p_n(x_0)}{x - x_0} = \frac{p_n(x) - y_0}{x - x_0} \quad (1)$$

$$\Delta^2 y = \frac{\Delta^1 y - \Delta^1 y_0}{x - x_1} \quad (2)$$

...

$$\Delta^n y = \frac{\Delta^{n-1} y - \Delta^{n-1} y_0}{x - x_{n-1}} \quad (3)$$

$$\text{De (1) tem-se } p_n(x) = y_0 + (x - x_0) \cdot \Delta^1 y \quad (4)$$

$$\text{De (2) tem-se } \Delta^1 y = \Delta^1 y_0 + (x - x_1) \Delta^2 y \quad (5)$$

$$\text{Aplicando (5) em (4) tem-se } p_n(x) = y_0 + (x - x_0) \cdot \Delta^1 y_0 + (x - x_0) \cdot (x - x_1) \Delta^2 y.$$

Repete-se o procedimento até (3).

De (3) tem-se  $\Delta^{n-1} y = \Delta^{n-1} y_0 + (x - x_{n-1}) \Delta^n y$  que aplicada à equação do polinômio resulta em

$$p_n(x) = y_0 + (x - x_0) \cdot \Delta^1 y_0 + (x - x_0) \cdot (x - x_1) \Delta^2 y_0 + (x - x_0) \cdot (x - x_1) \cdot (x - x_2) \Delta^3 y_0 + \dots + (x - x_0) \cdot (x - x_1) \cdot (x - x_2) \dots (x - x_{n-1}) \Delta^n y_0 + E_n(x) \quad (6)$$

$E_n(x) = (x - x_0) \cdot (x - x_1) \cdot (x - x_2) \dots (x - x_n) \Delta^{n+1} y_0$  e seu valor é nulo para os pontos utilizados para determinar o polinômio.

Pode-se, então considerar que  $p_n(x)$  é dado como segue e que  $E_n(x)$  é o erro.

$$p_n(x) = y_0 + (x - x_0) \cdot \Delta^1 y_0 + (x - x_0) \cdot (x - x_1) \cdot \Delta^2 y_0 + (x - x_0) \cdot (x - x_1) \cdot (x - x_2) \cdot \Delta^3 y_0 + \dots + (x - x_0) \cdot (x - x_1) \cdot (x - x_2) \dots (x - x_{n-1}) \cdot \Delta^n y_0$$

Para acelerar a avaliação de  $p_n(x)$ , é conveniente tratar a equação como segue.

$$p_n(x) = y_0 + (x - x_0) \cdot \left\{ \Delta^1 y_0 + (x - x_1) \cdot \left\{ \Delta^2 y_0 + (x - x_2) \Delta^3 y_0 + \dots + \left\{ \Delta^{n-1} y_0 + (x - x_{n-1}) \Delta^n y_0 \right\} \dots \right\} \right\}$$

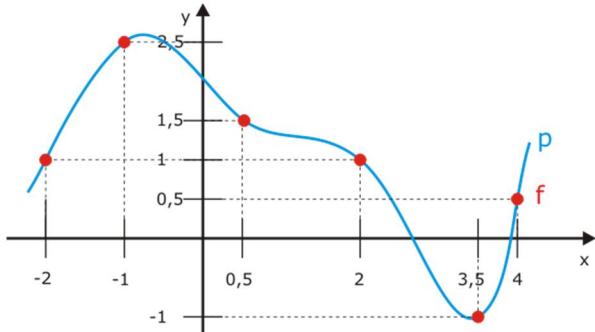
Utilizando o exemplo anterior, o polinômio é escrito como segue.

$x$	$\Delta^0 y_i$	$\Delta^1 y_i$	$\Delta^2 y_i$	$\Delta^3 y_i$	$\Delta^4 y_i$	$\Delta^5 y_i$
-2	1	1,5	-0,866667	0,244444	-0,062402	0,037502
-1	2,5	-0,666667	0,111111	-0,098765	0,162610	
0,5	1,5	-0,333333	-0,333333	0,714286		
2	1	-1,333333	2,166667			
3,5	-1	3				
4	0,5					

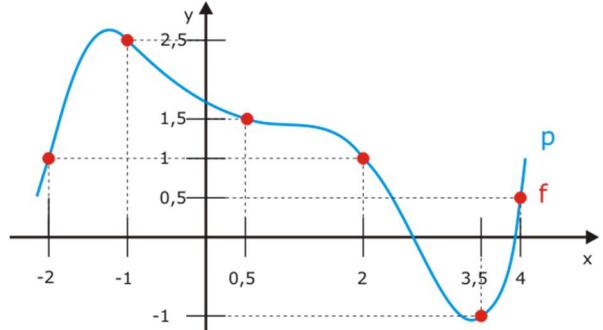
$$p_5(x) = 1 + (x+2) \cdot \{1,5 + (x+1) \cdot \{-0,866667 + (x-0,5) \cdot \{0,244444 + (x-2) \cdot \{-0,062402 + (x-3,5) \cdot \{0,037502\}\}\}\}\}$$

E agora, o tira-teima. O Programa da Patricinha vai entrar em ação. É só virar a página.

Minha antevisão



Outra possibilidade

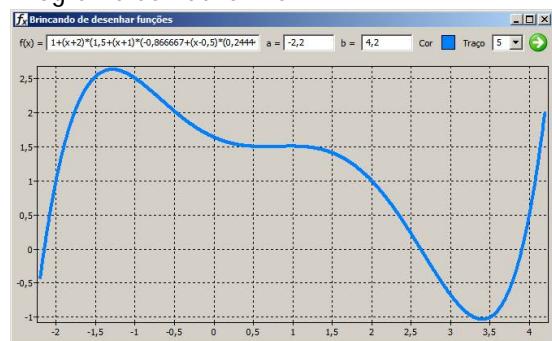


Comentário do Chapolin Colorado



O professor chutou errado.  
Suspeitei desde o princípio.

Programa da Patrincinha



## Método de Newton-Gregory para interpolação polinomial

Quando os ponto, nós ou polos  $(x_i, y_i)$  tiverem seus valores  $x_i$  igualmente espaçados, pode-se simplificar o método anterior, pois as diferenças  $(x_{k+1}-x_k)$  são constantes. Assim, o método de Newton-Gregory utiliza o conceito de diferenças finitas.

Considere  $f(x)$  uma função contínua,  $n+1$  vezes diferenciável e definida em  $n+1$  pontos distintos,  $x_0, x_1, \dots, x_n$ , de um intervalo  $[a, b]$ , com espaçamento constante  $h$  e, portanto,  $x_j = x_0 + j.h$ .

Define-se diferença finita como:

$$\Delta^0 y_i = y_i$$

$$\Delta^1 y_i = y_{i+1} - y_i$$

$$\Delta^2 y_i = \Delta^1 y_{i+1} - \Delta^1 y_i$$

...

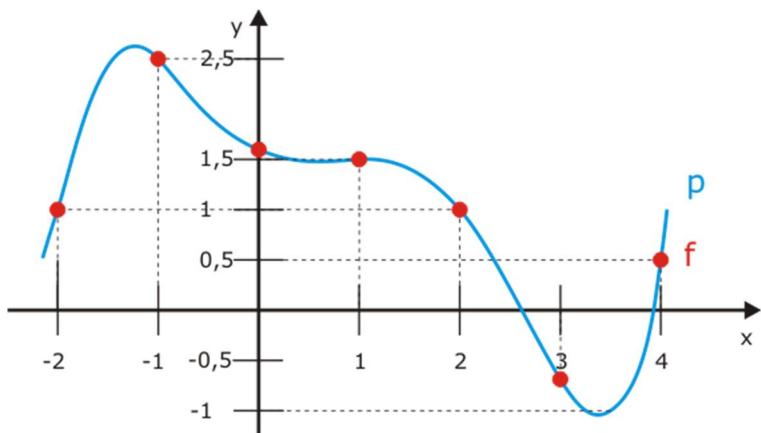
$$\Delta^n y_i = \Delta^{n-1} y_n - \Delta^{n-1} y_i$$

Computacionalmente, pode-se utilizar o esquema que segue, para calcular as diferenças finitas.

$x$	$\Delta^0 y_i$	$\Delta^1 y_i$	$\Delta^2 y_i$	$\Delta^3 y_i$	$\dots$	$\Delta^n y_i$
$x_0$	$y_0$	$\Delta^1 y_0$	$\Delta^2 y_0$	$\Delta^3 y_0$	$\dots$	$\Delta^n y_0$
$x_1$	$y_1$	$\Delta^1 y_1$	$\Delta^2 y_1$	$\Delta^3 y_1$		
$x_2$	$y_2$	$\Delta^1 y_2$	$\Delta^2 y_2$	$\vdots$		
$x_3$	$y_3$	$\Delta^1 y_3$	$\vdots$	$\Delta^3 y_{n-2}$		
$x_4$	$y_4$	$\vdots$	$\Delta^2 y_{n-1}$			
$\vdots$	$\vdots$	$\Delta^1 y_{n-1}$				
$x_n$	$y_n$					

Os exemplos numéricos que existem por aí são tão viciados que ajudam a atrapalhar. Segue um que poderá ajudar em seu treino algorítmico.

Considere  $f$  dada por uma nuvem de pontos e uma antevisão de uma função polinomial interpolada.



Dados os pontos, calcular as diferenças divididas.

x	y
-2	1
-1	2,5
0	1,6
1	1,5
2	1
3	-0,7
4	0,5

Seguem os cálculos para tabela computacional.

Calculando com tabela computacional.

x	$\Delta^0 y_i$	$\Delta^1 y_i$	$\Delta^2 y_i$	$\Delta^3 y_i$	$\Delta^4 y_i$	$\Delta^5 y_i$	$\Delta^6 y_i$
-2	1	1,5	-2,4	3,2	-4,4	4,8	-0,3
-1	2,5	-0,9	0,8	-1,2	0,4	4,5	
0	1,6	-0,1	-0,4	-0,8	4,9		
1	1,5	-0,5	-1,2	4,1			
2	1	-1,7	2,9				
3	-0,7	1,2					
4	0,5						

Certamente você já escreveu o algoritmo, o programa e testou. Ótimo. Só falta determinar o polinômio.

Para utilizar a formulação do polinômio do método anterior, é necessário que se transforme o cálculo das diferenças finitas em diferenças divididas. Foi utilizado o cálculo das diferenças finitas, pois o quociente para cada operação é constante e isto pode ser incorporado após os cálculos iniciais.

Comparando os cálculos das diferenças divididas com os cálculos das diferenças finitas, e efetuando as transformações, tem-se:

Diferenças divididas	Diferenças finitas	Transformação
$\Delta^0 y_i = y_i$	$\Delta^0 y_i = y_i$	$y_i$
$\Delta^1 y_i = \frac{y_{i+1} - y_i}{x_{i+1} - x_i}$	$\Delta^1 y_i = y_{i+1} - y_i$	$\frac{y_{i+1} - y_i}{h}$
$\Delta^2 y_i = \frac{\Delta^1 y_{i+1} - \Delta^1 y_i}{x_{i+2} - x_i}$	$\Delta^2 y_i = \Delta^1 y_{i+1} - \Delta^1 y_i$	$\frac{\Delta^1 y_{i+1} - \Delta^1 y_i}{h} = \frac{\Delta^2 y_i}{2.h}$
$\Delta^3 y_i = \frac{\Delta^2 y_{i+1} - \Delta^2 y_i}{x_{i+3} - x_i}$	$\Delta^3 y_i = \Delta^2 y_{i+2} - \Delta^2 y_i$	$\frac{\Delta^2 y_{i+1} - \Delta^2 y_i}{2.h^2} = \frac{\Delta^3 y_i}{3.h}$
$\Delta^n y_i = \frac{\Delta^{n-1} y_n - \Delta^{n-1} y_i}{x_n - x_i}$	$\Delta^n y_i = \Delta^{n-1} y_n - \Delta^{n-1} y_i$	$\frac{\Delta^{n-1} y_{i+1} - \Delta^{n-1} y_i}{(n-1).h^{n-1}} = \frac{\Delta^n y_i}{n.h}$

Pode-se, então utilizar a equação polinomial do método anterior com estas transformações.

$$p_n(x) = y_0 + (x-x_0) \cdot \frac{\Delta^1 y_0}{1! \cdot h^1} + (x-x_0) \cdot (x-x_1) \cdot \frac{\Delta^2 y_0}{2! \cdot h^2} + (x-x_0) \cdot (x-x_1) \cdot (x-x_2) \cdot \frac{\Delta^3 y_0}{3! \cdot h^3} + \dots + \\ (x-x_0) \cdot (x-x_1) \cdot (x-x_2) \dots (x-x_{n-1}) \cdot \frac{\Delta^n y_0}{n! \cdot h^n} \quad (\text{A.1})$$

Para acelerar a avaliação de  $p_n(x)$ , é conveniente tratar a equação como segue.

$$p_n(x) = y_0 + (x-x_0) \cdot \left\{ \frac{\Delta^1 y_0}{1! \cdot h^1} + (x-x_1) \cdot \left\{ \frac{\Delta^2 y_0}{2! \cdot h^2} + (x-x_2) \frac{\Delta^3 y_0}{3! \cdot h^3} + \dots \left\{ \frac{\Delta^{n-1} y_0}{(n-1)! \cdot h^{n-1}} + (x-x_{n-1}) \frac{\Delta^n y_0}{n! \cdot h^n} \right\} \dots \right\} \right\} \quad (\text{A.2})$$

Como  $x_{k+i}-x_k$  é constante, pode-se simplificar a equação do polinômio, como segue.

Faz-se  $s = (x-x_0)/h$  e, então,  $x = s \cdot h + x_0$ . Como  $x_j = x_0 + j \cdot h$ ,  $x-x_j = h \cdot (s-j)$ .

Então, com  $s = (x-x_0)/h$ , o polinômio interpolador de Newton-Gregory pode ser escrito como:

$$p_n(s) = y_0 + s \cdot \frac{\Delta^1 y_0}{1!} + s \cdot (s-1) \cdot \frac{\Delta^2 y_0}{2!} + s \cdot (s-1) \cdot (s-2) \cdot \frac{\Delta^3 y_0}{3!} + \dots + \\ s \cdot (s-1) \cdot (s-2) \cdot (s-3) \dots (s-n+1) \cdot \frac{\Delta^n y_0}{n!} \quad (\text{B.1})$$

Para acelerar a avaliação de  $p_n(x)$ , é conveniente tratar a equação como segue.

$$p_n(s) = y_0 + s \cdot \left\{ \frac{\Delta^1 y_0}{1!} + (s-1) \cdot \left\{ \frac{\Delta^2 y_0}{2!} + (s-2) \frac{\Delta^3 y_0}{3!} + \dots \left\{ \frac{\Delta^{n-1} y_0}{(n-1)!} + (s-n+1) \frac{\Delta^n y_0}{n!} \right\} \dots \right\} \right\} \quad (\text{B.2})$$

Cuidado ao programar. As equações (A.1) e (A.2) calculam pontos da interpolação para  $x$  e as equações (B.1) e (B.2) calculam pontos da interpolação para  $s$ , onde  $s = (x-x_0)/h$ . Decida qual utilizar e utilize corretamente.

Utilizando o exemplo anterior, o polinômio é escrito como segue.

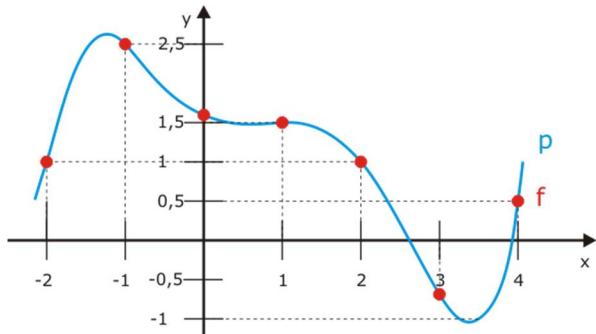
$x$	$\Delta^0 y_i$	$\Delta^1 y_i$	$\Delta^2 y_i$	$\Delta^3 y_i$	$\Delta^4 y_i$	$\Delta^5 y_i$	$\Delta^6 y_i$
-2	1	1, 5	-2, 4	3, 2	-4, 4	4, 8	-0, 3
-1	2, 5	-0, 9	0, 8	-1, 2	0, 4	4, 5	
0	1, 6	-0, 1	-0, 4	-0, 8	4, 9		
1	1, 5	-0, 5	-1, 2	4, 1			
2	1	-1, 7	2, 9				
3	-0, 7	1, 2					
4	0, 5						

Utilizando a equação A.2

$$p_6(x) = 1 + (x+2) \cdot \{1,5/1+(x+1) \cdot \{-2,4/2+(x-0) \cdot \{3,2/6+(x-1) \cdot \{-4,4/24+(x-2) \cdot \{4,8/120+(x-3) \cdot \{-0,3/720\}\}\}\}\}$$

E agora, o tira-teima. O Programa da Patricinha vai entrar em ação. É só virar a página.

Minha antevisão



Comentário do Chaves



Programa da Patricinha



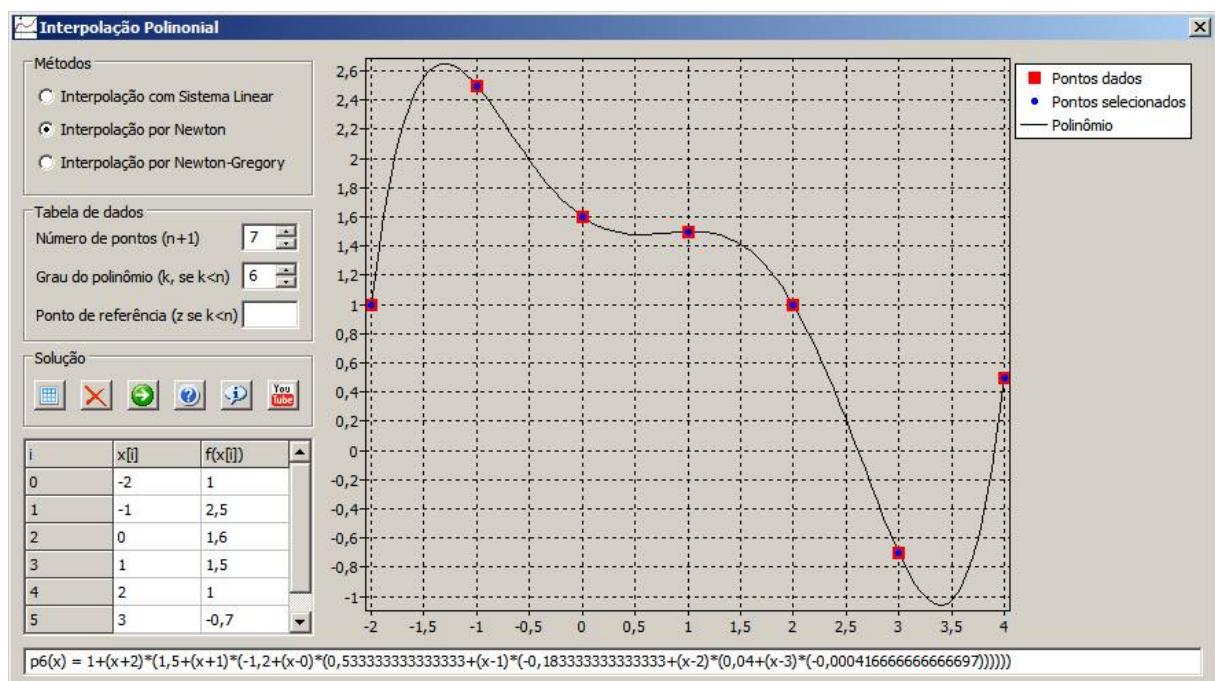
Isso, isso, isso. Agora o chute foi bom.

No exemplo foi utilizada a equação (A.2), para utilizar o Programa da Patricinha, com  $p_6(x)$ , no intervalo  $[-2,2; 4,2]$ .

Se você se decidir utilizar a equação (B.2) em seu programa de interpolação, lembre-se que ao calcular  $p_6(s)$ , com  $s = (x-x_0)/h$ , o valor de  $s$  deve ser calculado para cada valor de  $x$ .

### Sugestões para seus programas

Um simpático programa, para calcular interpolação polinomial, poderia ter a cara que segue.



Com este programa é possível:

- definir a ordem do polinômio;
- selecionar o método para determinação do polinômio;
- calcular os coeficientes do polinômio;
- apresentar a forma explícita do polinômio;
- desenhar os pontos dados, os pontos selecionados e o polinômio.

### **Instruções claras e precisas sobre trabalhos computacionais para avaliação**

Todas as instruções necessárias estão descritas no mesmo item, do Capítulo 7. Copiar e colar é muito fácil, mas não é nada elegante, agradável, simpático ou conveniente repetir tudo aquilo que já está escrito em capítulo anterior. Além disto, se você já entregou algum trabalho, já sabe como deve proceder. Caso contrário, veja as instruções no Capítulo 7. E antes de entregar, leia as instruções em <http://sacoman.dco.fc.unesp.br/mnc/pet/>.

Se você optar em fazer um programa que tenha a aparência da figura apresentada ou similar, poderá continuar lendo para ter algumas explicações que poderão ser úteis para você.

### **Um exemplo de programa**

Minha ajuda será *apenas na fabricação da interface*. O resto é com você. Depois de pronto, limpe todo lixo gerado durante a elaboração do projeto, compacte com Zip, Rar ou qualquer outro compactador, com o sugestivo nome T4-RA-V1.ext, onde RA é o número de seu Registro Acadêmico, ext é a extensão fornecida pelo seu compactador, V1 é a primeira versão e entregue via endereço eletrônico de recepção que você já sabe qual é. Não sabe qual é? Então leia tudo sobre o assunto no endereço <http://sacoman.dco.fc.unesp.br/mnc/pet/>.

Então, mãos a obra.

Crie um diretório apropriado para desenvolver seu projeto (e.g.: X:\MNC-Trabalhos\T4). Organize seu ambiente de trabalho e crie subdiretórios necessários para o projeto, como, por exemplo, um diretório chamado Figuras, para as figuras dos botões e ícone e outro chamado Exemplo, para o exemplo executável que está à sua disposição.

Copie de <http://sacoman.dco.fc.unesp.br/mnc/material/> o arquivo M4-Interpolação.rar e descomponha. Mova Interpol.exe e Interpol.chm para o diretório exemplo, para executar quando quiser. Mova as figuras para o diretório Figuras.

Inicie seu ambiente de projeto, no meu caso, Lazarus, ajeite as janelas da forma como achar conveniente e salve o projeto com o nome Interpol.lpr e a unidade principal com o nome Principal. Pressione o botão Nova Fórmula e salve dando o nome Sobre para esta nova unidade.

No Editor de Código, pressione a aba Sobre, pressione F12 para ver a fórmula, atribua Caption=Sobre o programa Interpolação Polinomial, BorderStyle=bsDialog, para Font, Name=Tahoma e Size=8, Height=275, Width=177 e Position=poDesktopCenter.

Lance, na fórmula, um componente TLabel e atribua Caption=Quase tudo feito por Newton, Left=19 e

Top=8. Lance outro componente TLabel e atribua Caption=Isaac Newton (1642-1727), Left=24 e Top=254. E, neste buraco que ficou na fórmula, lance um componente TImage e atribua Height=221, Width=161, Left=8, Top=28 e, em Picture, carregue o arquivo Newton-1689.png.

Seu About Box está pronto. Não faça mais nada. Mas, quando utilizar o meu programa, cutuque a imagem do Newton, para ver outra foto dele.

Agora, só falta montar a fórmula Principal e fazer o programa.

No Editor de Código, pressione a aba Principal, pressione F12 para ver a fórmula, atribua Caption=Interpolação Polinomial, BorderStyle=bsSingle, para BorderIcons, biMaximize=False e biMinimize=False, para Font, Name=Tahoma e Size=8, Height=468, Width=875, Position=poDesktopCenter e ShowHint=True.

Lance, na fórmula, alguns componentes como segue.

Componente	Caption	Left	Top	Height	Width
GroupBox1	Métodos	8	8	105	212
GroupBox2	Tabela de dados	8	116	103	212
GroupBox3	Solução	8	222	59	212

No GroupBox1, lance alguns componentes como segue.

Componente	Caption	Left	Top	Checked
RadioButton1	Interpolação com Sistema Linear	8	8	True
RadioButton2	Interpolação por Newton	8	32	False
RadioButton3	Interpolação por Newton-Gregory	8	56	False

No GroupBox2, lance alguns componentes como segue.

Componente	Caption	Left	Top	Text	Width	MinValue	.MaxValue	Value
Label1	Número de pontos (n+1)	8	4					
Label2	Grau do polinômio (k, se k<n)	8	32					
Label3	Ponto de referência (z se k<n)	8	60					
SpinEdit1		157	0		42	3	20	4
SpinEdit2		157	28		42	1	19	3
Edit1		157	56	Remova	42			

No GroupBox3, lance alguns componentes como segue.

Componente	Left	Top	Flat	Glyph
SpeedButton1	6	8	True	grid.gif
SpeedButton2	40	8	True	delete.gif
SpeedButton3	74	8	True	forward.gif
SpeedButton4	108	8	True	help2.gif
SpeedButton5	142	8	True	inform1.gif
SpeedButton6	176	8	True	youtube.gif

Pode ser que você não queira o botão do YouTube e, neste caso, faça espaçamentos adequados.

Uma sugestão é Left iguais a 8, 50, 92, 134, 176, para os cinco primeiros botões.

*No meu programa, desta vez, deixei Flat=False, só para variar.*

Lance, na fórmula, mais três componentes como segue.

Um componente TStringGrid (StringGrid1) e atribua Left=8, Top=288, ColCount=3, RowCount=7, Height=144, Width=212, ScrollBars=ssVertical e, para Options, goEditing=True.

Um componente TChart (Chart1) e atribua Left=228, Top=8, Height=426, Width=639 e, para Legend, Visible=True.

Um componente TEdit (Edit2) e atribua Left=8, Top=440, Width=859 e remova o valor de Text.

Marque o Chart1, pressione o botão direito do mouse, para abrir o Editor de Séries, pressione o botão Add, adicione uma Line series (Chart1LineSeries1), atribua LineType=ltNone, ShowPoints=True, Title=Pontos dados e, para Pointer, Style=psRectangle, HorizSize=4, VertSize=4, Brush Color=clRed e Pen Color=clRed.

No Editor de Séries, pressione o botão Add, adicione uma Line series (Chart1LineSeries2), atribua LineType=ltNone, ShowPoints=True, Title=Pontos selecionados e, para Pointer, Style=psCircle, HorizSize=2, VertSize=2, Brush Color=clBlue e Pen Color=clBlue.

No Editor de Séries, pressione o botão Add, adicione uma Line series (Chart1LineSeries3) e atribua Title=Polinômio.

A propriedade Hint de cada componente será atribuída na rotina FormShow, que fica mais fácil e visível para editar e dar manutenção.

Acredite, se quiser, que agora só falta fazer o programa. Siga a receita para criar as rotinas que serão disparadas pelos eventos e, depois, copie descaradamente o programa. Ou faça o seu.

Cutucando:

- Cutuque a fôrma e, em Eventos, dê um toque duplo em OnShow, para criar a rotina FormShow.
- Selecione SpinEdit1 e, em Eventos, dê um toque duplo em OnEditingDone, para criar a rotina SpinEdit1EditingDone.
- Selecione SpeedButton1 e, em Eventos, dê um toque duplo em OnClick, para criar a rotina SpeedButton1Click.
- Selecione SpeedButton2 e, em Eventos, dê um toque duplo em OnClick, para criar a rotina SpeedButton2Click.
- Selecione SpeedButton3 e, em Eventos, dê um toque duplo em OnClick, para criar a rotina SpeedButton3Click.
- Selecione SpeedButton4 e, em Eventos, dê um toque duplo em OnClick, para criar a rotina SpeedButton4Click.
- Selecione SpeedButton5 e, em Eventos, dê um toque duplo em OnClick, para criar a rotina SpeedButton5Click.
- Selecione SpeedButton6 e, em Eventos, dê um toque duplo em OnClick, para criar a rotina SpeedButton6Click, somente se for utilizar o botão do YouTube.

Você criou 8 rotinas em uma ordem lógica e o Lazarus as colocou em ordem alfabética. No meu projeto, eu as reordenei, para continuarem na ordem lógica. Se considerar que deve reordená-las, reordene de acordo com a lista de criação, de FormShow até SpeedButton5 ou SpeedButton6, que ficará mais fácil para entender o programa e, principalmente, para *descaradamente* copiá-lo.

Agora é só copiar ou fazer o seu. Inicialmente são definidas algumas unidades necessárias na cláusula uses, alguma variáveis globais, rotinas de cálculos auxiliares rotinas de cálculos de interpolação e, depois, as rotinas dos eventos.

Continue digitando sem salvar, para perder tudo se a CPFL ajudar interrompendo o fornecimento de energia elétrica. **SALVE essa "!@#\$%&\* de Programa**. Agora que salvou, inicie a cola. São apenas seis páginas. Não terá mais nenhum texto explicativo, mas as rotinas estão ampla, geral, irrestrita e completamente documentadas. Mão a obra.

```

implementation

{$R *.lfm}

{ TForm1 }

uses
  Sobre,
  Math, Windows, LCLIntf; // Math, Windows e LCLIntf para usar Power, GetSystemMetrics e OpenDocument

var
  EFH, EFV, ESH, ESV: Byte; // Valores para Grade que são diferentes em diferentes versões do Windows
  DRH, DCW: Integer;
  n, m, k: Integer;
  z: Real;
  x, y, c, Delta0y: array[0..20] of Real;
  xTodos, yTodos: array[0..20] of Real;

{ ***** Início das rotinas de cálculo ***** }
{ Auxiliares: OrdenarDados, DadosRepetidos e PontosProximos }
{ Interpolação: SistemaLinear, Newton e NewtonGregory }

procedure OrdenarDados(n: Integer);
var
  Trocou: Boolean;
  Auxiliar: Real;
  i, j: Integer;
begin
  // Ordena valores de x de forma crescente, mantendo os dados da grade como estão
  Trocou := True;
  j := n-1;
  while Trocou do
    begin
      Trocou := False;
      for i := 0 to j do
        if x[i] > x[i+1] then
          begin
            Auxiliar := x[i]; x[i] := x[i+1]; x[i+1] := Auxiliar;
            Auxiliar := y[i]; y[i] := y[i+1]; y[i+1] := Auxiliar;
            Trocou := True;
          end;
      j := j -1;
    end;
end;

function DadosRepetidos(var p, q: Integer): Boolean;
var
  i, j: Integer;
begin
  // Verifica se x[i]=x[j], i<>j e, se verdade, p e q são os índices das variáveis iguais
  Result := False;
  for i := 0 to n-1 do
    for j := i+1 to n do
      if x[i] = x[j] then
        begin
          p := i;
          q := j;
          Result := True;
          Exit;
        end;
  end;
end;

```

```

procedure PontosProximos;
var
  Indices: array[0..20] of Byte;
  Distancias: array[0..20] of Real;
  i, j: Integer;
  Trocou: Boolean;
  AuxiliarD: Real;
  AuxiliarI: Byte;
begin
  // Seleciona k pontos x mais próximos de z
  // Lembre-se que os pontos já estão ordenados e x[0] <= z <= x[n]
  for i := 0 to n do
  begin
    Distancias[i] := Abs(z-x[i]);
    Indices[i] := i;
  end;
  // Ordenar Distancias e Indices
  Trocou := True;
  j := n-1;
  while Trocou do
  begin
    Trocou := False;
    for i := 0 to j do
      if Distancias[i] > Distancias[i+1] then
      begin
        AuxiliarD := Distancias[i]; Distancias[i] := Distancias[i+1]; Distancias[i+1] := AuxiliarD;
        AuxiliarI := Indices[i]; Indices[i] := Indices[i+1]; Indices[i+1] := AuxiliarI;
        Trocou := True;
      end;
    j := j-1;
  end;
  // Preencher vetores x e y com os valores selecionados
  // Os vetores originais já foram copiados para xTodos e yTodos antes de chamar esta rotina
  FillChar(x, SizeOf(x), 0);
  FillChar(y, SizeOf(y), 0);
  for i := 0 to k do
  begin
    x[i] := xTodos[Indices[i]];
    y[i] := yTodos[Indices[i]];
  end;
end;

```

```

procedure SistemaLinear(Ordem: Integer);
var
  A: array[1..21, 1..21] of Real;
  b: array[1..21] of Real;
  xLocal: array[1..21] of Real; // para não conflitar com x Global
  i, j, n: Integer;
  pivo, soma: Real;
begin
  // Recebe Ordem que pode ser n Global ou k Global
  // Não confunda com n e k Locais da rotina
  // Portanto, resolve sistema para todos os pontos ou para os selecionados
  n := Ordem+1;
  // Montando a matriz A e o vetor b
  FillChar(A, SizeOf(A), 0);
  FillChar(b, SizeOf(b), 0);
  FillChar(xLocal, SizeOf(xLocal), 0);
  FillChar(c, SizeOf(c), 0);
  •••

```

Escreva o trecho de programa que monta a Matriz A e o Vetor b.

```

•••
// Solução por Gauss (triangular superior)
// Triangularização
•••

```

Escreva o trecho de programa que triangulariza a Matriz A.

Lembre-se de utilizar try except end ao calcular  $m = a_{ij}/a_{ii}$  e se houver exceção informar que o determinante é nulo e interromper. Mas, se isto acontecer, há erro na montagem da Matriz A, pois a matriz de Vandermonde tem determinante não nulo.

```

•••
// Retrosubstituição
•••

```

Escreva o trecho de programa que resolve o sistema.

Lembre-se de utilizar try except end ao calcular  $xLocal_n = b_n/a_{nn}$  e  $xLocal_i = (b_i - soma)/a_{ii}$  e se houver exceção informar e interromper. Mas, se isto acontecer, há erro na montagem da Matriz A, pois a matriz de Vandermonde tem determinante não nulo.

```

•••

```

```

•••
Ao terminar, a solução xLocal contém os coeficientes do polinômio.
Copie xLocal na variável global c dos coeficientes.

•••
for i := 1 to n+1 do
  c[i-1] := xLocal [i];
end;

procedure Newton(Ordem: Integer);
var
  i, j, n: Integer;
•••
Defina as variáveis locais que você utilizará para calcular as Diferenças Divididas  $\Delta^k y_i$ .
•••

begin
  n := Ordem;
•••
  Calcular os  $\Delta^k y_i$ .
  No meu programa, os cálculos dos  $\Delta^k y_i$  estão em Del.
•••

  Copie os valores locais para o vetor global Delta0y.
•••
  FILLChar(Delta0y, Si zeOf(Delta0y), 0);
  for i := 0 to n do
    Delta0y[i] := Del [i];
end;

procedure NewtonGregory(Ordem: Integer);
var
  i, j, n: Integer;
•••
Defina as variáveis locais que você utilizará para calcular as Diferenças Finitas  $\Delta^k y_i$ .
•••

begin
  n := Ordem;
•••
  Calcular os  $\Delta^k y_i$ .
  No meu programa, os cálculos dos  $\Delta^k y_i$  já foram divididos por  $k! \cdot h^k$ .
  No meu programa, os cálculos dos  $\Delta^k y_i / k! \cdot h^k$  estão em Del.
•••

  Copie os valores locais para o vetor global Delta0y.
•••
  FILLChar(Delta0y, Si zeOf(Delta0y), 0);
  for i := 0 to n do
    Delta0y[i] := Del [i];
end;

{ ***** Fim das rotinas de cálculo ***** }

procedure TForm1.FormShow(Sender: TObject);
begin
  // Tempo para leitura dos textos informativos (padrão = 2500 ms = 2,5 s - pouco para longos textos)
  // Modificando para 20 segundos = 20000 ms
  // Mesmo sem este comando, no meu computador, o Hint fica visível por mais de 30 segundos
  // Teste seu programa e utilize se necessário
  Application.HintDePause := 20000;
  // Textos para Informações (Hint) de alguns componentes
  RadioButon1.Hint := 'Interpolação com Sistema Linear:'#10#10+
    'Calcula os coeficientes do polinômio'#10+
    'através de solução de sistema linear.'#10;
  RadioButon2.Hint := 'Interpolação por Newton:'#10#10+
    'Calcula os coeficientes do polinômio'#10+
    'através de diferenças finitas.'#10;
  RadioButon3.Hint := 'Interpolação por Newton-Gregory:'#10#10+
    'Calcula os coeficientes do polinômio'#10+
    'através de diferenças finitas com '#10+
    'espacamentos constantes.'#10;
  SpinEdit1.Hint := 'Número de pontos (n+1):'#10#10+
    'Para a seqüência de pontos (x[i], y[i]), '#10+
    'i = 0, ..., n, o total de pontos é n+1.'#10#10+
    'Entendeu, ou preciso fazer um desenho?'#10;

```

```

SpinEdit2.Hint := 'Grau do polinômio ( k<n+1 ):'#10#10+
  'Para n+1 pontos, o grau máximo' #10+
  'do polinômio é n.' #10;
Edit1.Hint := 'Ponto de referência z ( k<n ):'#10#10+
  'Se o grau determinado para o polinômio for menor que n,'#10+
  'deve-se fornecer um valor de referência entre x[0] e x[n].'#10#10+
  'O Polinômio será calculado para as proximidades do valor de z.'#10;
SpeedButton1.Hint := 'Cria ou ajusta grade:'#10#10+
  'Cria grade para entrada dos dados' #10+
  '( x[i], y[i] ), i = 0, ..., n, ou seja,'#10+
  'grade com n+1 caselas para x e para y.' #10;
SpeedButton2.Hint := 'Limpa a grade:'#10#10+
  'Os dados não são apagados quando se altera seu tamanho' #10+
  'para permitir ao usuário aproveitar dados já digitados.' #10;
SpeedButton3.Hint := 'Calcula polinômio interpolador:'#10#10+
  'Resolve o problema, através do método' #10+
  'determinado, com os dados oferecidos.' #10;
SpeedButton4.Hint := 'Ajuda:'#10#10+
  'Apresenta o arquivo de ajuda, caso exista.'#10#10+
  'Pois é, se não existir, não apresenta. :-)' #10;
SpeedButton5.Hint := 'Informação:'#10#10+
  'Apresenta informações sobre autoria e direitos.'#10#10+
  'Cuidado: Pirataria é crime. :-(' #10;
SpeedButton6.Hint := 'Iú Tube:'#10#10+
  'Veja no YouTube.' #10+
  'Veja em tela cheia.' #10+
  'Veja em tela vazia.' #10+
  'Não veja.' #10#10+
  'A Internet virou uma palhaçada.'#10+
  'Os programas de MNC, também. :-)' #10;

// Medidas para moldura e scrollbar da Grade (são diferentes para cada versão do Windows)
EFH := GetSystemMetrics(SM_CYBORDER)+GetSystemMetrics(SM_CFYXEDFRAME); // Espessura Frame Horizontal
EFV := GetSystemMetrics(SM_CXBORDER)+GetSystemMetrics(SM_CFYXEDFRAME); // Espessura Frame Vertical
ESH := GetSystemMetrics(SM_CYHSCROLL); // Espessura Scroll Horizontal
ESV := GetSystemMetrics(SM_CXVSCROLL); // Espessura Scroll Vertical (não utilizado neste programa)
// Medidas da altura da linha e comprimento da coluna de cada casela
// Mesmo definindo os valores no projeto, eles mudam de acordo com a versão do Windows
// Forçar altura da casela ser a mesma em diferentes versões do Windows
StringGrid1.DefaultRowHeight := 20;
DRH := StringGrid1.DefaultRowHeight; // Default Row Height
DCW := StringGrid1.DefaultColWidth; // Default Col Width

// Ajustar a grade
SpeedButton1.Click;
end;

procedure TForm1.SpinEdit1EditingDone(Sender: TObject);
begin
  // Sempre que n+1 alterar, fazer grau do polinômio = n
  // Se o usuário quiser grau menor, ele deverá alterar valor de k
  // Ao alterar n+1, alterar grade
  SpinEdit2.Value := SpinEdit1.Value-1;
  SpeedButton1.Click;
end;

procedure TForm1.SpeedButton1Click(Sender: TObject);
var
  i: Integer;
begin
  // Ajustar a grade (em função do número de linhas e de diferentes versões de Windows)
  m := SpinEdit1.Value;
  n := m-1;
  with StringGrid1 do
  begin
    RowCount := m+1;
    if RowCount > 7 then
    begin
      Width := DCW*3 + EFH + ESH; // Com barra de rolagem (necessária)
      Height := DRH*7 + EFV;
    end
    else
    begin
      Width := DCW*3 + EFH + ESH; // Caso não queira, remova +ESH e projete com ScrollBars ssAutoBoth
      Height := DRH*(m+1) + EFV;
    end;
    Cells[0,0] := 'i';
  end;
end;

```

```

Cells[1,0] := 'x[i]';
Cells[2,0] := 'f(x[i])';
for i := 0 to n do
  Cells[0,i+1] := IntToStr(i);
end;
end;

procedure TForm1.SpeedButton2Click(Sender: TObject);
var
  i, j: Byte;
begin
  // Limpar a grade, o gráfico e a solução
  for i := 1 to StringGrid1.RowCount-1 do
    for j := 1 to 2 do
      StringGrid1.Cells[j,i] := '';
  Chart1LineSeries1.Clear;
  Chart1LineSeries2.Clear;
  Chart1LineSeries3.Clear;
  Edit2.Clear;
end;

procedure TForm1.SpeedButton3Click(Sender: TObject);
var
  i, j, p, q: Integer;
  passo: Real;
  xGrafico, yGrafico: Real;
begin
  // Calcular polinômio
  // Ao utilizar SpinEdit1 (n+1 = m), SpinEdit2 (k) é alterado e a grade é redimensionada
  // Então, m é conhecido e a grade está de tamanho correto
  // Mas o usuário pode ter alterado k (SpinEdit2)
  // Testar valores de k e z
  k := SpinEdit2.Value;
  if (k < 1) or (k >= m) then
  begin
    ShowMessage('A ordem do polinômio deve ser entre 1 e n.');
    SpinEdit2.SetFocus;
    Exit;
  end;
  // Se k < n tem que existir z
  if k < n then
    try
      z := StrToFloat(Edit1.Text);
    except
      ShowMessage('Verifique o valor de z.');
      Edit1.SetFocus;
      Exit;
    end;
  // Ler pares (x,y) da grade
  with StringGrid1 do
  begin
    // Ler coluna x da grade e armazenar no vetor x
    for i := 0 to n do
    begin
      try
        x[i] := StrToFloat(Cells[1,i+1]);
      except
        ShowMessage('Verifique o valor de x[' + IntToStr(i) +'].');
        Col := 1;
        Row := i+1;
        SetFocus;
        Exit;
      end;
    end;
    // Ler coluna y da grade e armazenar no vetor y
    for i := 0 to n do
    begin
      try
        y[i] := StrToFloat(Cells[2,i+1]);
      except
        ShowMessage('Verifique o valor de y[' + IntToStr(i) +'].');
        Col := 2;
        Row := i+1;
        SetFocus;
        Exit;
      end;
    end;
  end;
end;

```

```

// Verificar se há valores de x repetidos
if DadosRepetidos(p, q) then
begin
  ShowMessage('Os valores de x[' +IntToStr(p)+'] e de x[' +IntToStr(q)+'] são iguais.'#10+
  'Remova ou altere os dados repetidos para valores de x.');
  Exit;
end;
// Ordenar os dados com x crescente (x[0], ..., x[n])
OrdenarDados(n);
// Se k < n existe z e z deve estar entre x[0] e x[n] (com x já ordenado)
if k < n then
  if (z < x[0]) or (z > x[n]) then
begin
  ShowMessage('z deve estar entre '+FloatToStr(x[0])+' e '+FloatToStr(x[n])+'.');
  Edi t1.SetFocus;
  Exit;
end;
// Se k < n, armazenar (x,y) em (xTodos,yTodos) e escolher
// k valores de (xTodos,yTodos) para ser o novo (x,y)
// Isto não é necessário, mas quero desenha todos os
// pontos com uma cor e os k pontos com outra cor
for i := 0 to n do
begin
  xTodos[i] := x[i];
  yTodos[i] := y[i];
end;
// Se k < n, selecionar k valores de x mais próximos de z
PontosProximos;
// Ordenar os k dados com x crescente (x[0], ..., x[k])
OrdenarDados(k);

// Resolver Interpolação

if RadioButon1.Checked then          // Interpolação por Sistemas Lineares
begin
  if k < n then
    SistemaLinear(k)
  else
    SistemaLinear(n);
  // Escrever equação do polinômio
  Edi t2.Text := 'p' +IntToStr(n) +'(x) = ';
  for i := 0 to n do
    if c[i] < 0 then
      Edi t2.Text := Edi t2.Text + FloatToStr(c[i]) + '*x^' + IntToStr(i)
    else
      Edi t2.Text := Edi t2.Text + '+' + FloatToStr(c[i]) + '*x^' + IntToStr(i);
  //DesenharGrafico
  Chart1LineSeries1.Clear;
  Chart1LineSeries2.Clear;
  Chart1LineSeries3.Clear;
  passo := (xTodos[n]-xTodos[0])/100; // Se não gostou da cara da curva,
  for i := 0 to 100 do              // troque 100 por 1000 nestas linhas
begin
  xGrafico := xTodos[0]+i *passo;
  yGrafico := 0;
  for j := 0 to n do
    yGrafico := yGrafico + c[j]*Power(xGrafico,j);
  Chart1LineSeries3.AddXY(xGrafico,yGrafico);
end;
  for i := 0 to n do
    Chart1LineSeries1.AddXY(xTodos[i],yTodos[i]);
  for i := 0 to k do
    Chart1LineSeries2.AddXY(x[i],y[i]);
end;

if RadioButon2.Checked then          // Interpolação por Newton
begin
  if k < n then
    Newton(k)
  else
    Newton(n);
  // Escrever equação do polinômio
  Edi t2.Text := 'p' +IntToStr(n) +'(x) = ' +FloatToStr(Delta0y[0]);
  for i := 0 to n-1 do
    if x[i] < 0 then
      Edi t2.Text := Edi t2.Text + '+(' +FloatToStr(-x[i]) + ')*' +FloatToStr(Delta0y[i+1])
    else

```

```

if x[i] = 0 then
    Edi t2.Text := Edi t2.Text + '(x-' +FloatToStr(-x[i]) + ')*' +FloatToStr(Delta0y[i+1])
else
    Edi t2.Text := Edi t2.Text + '(x' +FloatToStr(-x[i]) + ')*' +FloatToStr(Delta0y[i+1]);
for i := 0 to n-1 do
    Edi t2.Text := Edi t2.Text + ')';
//DesenharGrafico
Chart1LineSeries1.Clear;
Chart1LineSeries2.Clear;
Chart1LineSeries3.Clear;
passo := (xTodos[n]-xTodos[0])/100; // Se não gostou da cara da curva,
for i := 0 to 100 do // troque 100 por 1000 nestas linhas
begin
    xGrafico := xTodos[0]+i *passo;
    yGrafico := Delta0y[k]; // k seira n, se não houvesse a história do k, z
    for j := k-1 downto 0 do // k-1 seria n-1
        yGrafico := yGrafico * (xGrafico-x[j]) + Delta0y[j];
    Chart1LineSeries3.AddXY(xGrafico,yGrafico);
end;
for i := 0 to n do
    Chart1LineSeries1.AddXY(xTodos[i],yTodos[i]);
for i := 0 to k do
    Chart1LineSeries2.AddXY(x[i],y[i]);
end;

if RadioButton3.Checked then // Interpolação por Newton-Gregory
begin
    // Verificar se valores de x são equidistantes
    passo := x[1]-x[0];
    for i := 2 to n do
        if (x[i]-x[i-1]-passo)/passo > 1E-7 then // x[i]-x[i-1] = passo com erro relativo 1E-7
        begin
            ShowMessage('Com o vetor x já ordenado, os valores de x[' +IntToStr(i-1)+'] e de x[' +IntToStr(i)+'] não '#10+
                'tem o mesmo espaçamento existente entre x[0] e x[1].');
            Exit;
        end;
    if k < n then
        NewtonGregory(k)
    else
        NewtonGregory(n);
    // Escrever equação do polinômio
    Edi t2.Text := 'p' +IntToStr(n) +'(x) = ' +FloatToStr(Delta0y[0]);
    for i := 0 to n-1 do
        if x[i] < 0 then
            Edi t2.Text := Edi t2.Text + '(x+' +FloatToStr(-x[i]) + ')*' +FloatToStr(Delta0y[i+1])
        else
            if x[i] = 0 then
                Edi t2.Text := Edi t2.Text + '(x-' +FloatToStr(-x[i]) + ')*' +FloatToStr(Delta0y[i+1])
            else
                Edi t2.Text := Edi t2.Text + '(x' +FloatToStr(-x[i]) + ')*' +FloatToStr(Delta0y[i+1]);
    for i := 0 to n-1 do
        Edi t2.Text := Edi t2.Text + ')';
//DesenharGrafico
Chart1LineSeries1.Clear;
Chart1LineSeries2.Clear;
Chart1LineSeries3.Clear;
passo := (xTodos[n]-xTodos[0])/100; // Se não gostou da cara da curva,
for i := 0 to 100 do // troque 100 por 1000 nestas linhas
begin
    xGrafico := xTodos[0]+i *passo;
    yGrafico := Delta0y[k]; // n se não houvesse a história do k, z
    for j := k-1 downto 0 do // n-1
        yGrafico := yGrafico * (xGrafico-x[j]) + Delta0y[j];
    Chart1LineSeries3.AddXY(xGrafico,yGrafico);
end;
for i := 0 to n do
    Chart1LineSeries1.AddXY(xTodos[i],yTodos[i]);
for i := 0 to k do
    Chart1LineSeries2.AddXY(x[i],y[i]);
end;

```

```

procedure TForm1.SpeedButton4Click(Sender: TObject);
begin
  // Ajuda
  if not OpenDocument('Ajuda.chm') then
    MessageDlg('Informação', 'O arquivo de ajuda \'Ajuda.chm\' não foi encontrado.', mtInformation, [mbOk], 0);
end;

procedure TForm1.SpeedButton5Click(Sender: TObject);
begin
  // Informação
  Form2.ShowModal();
end;

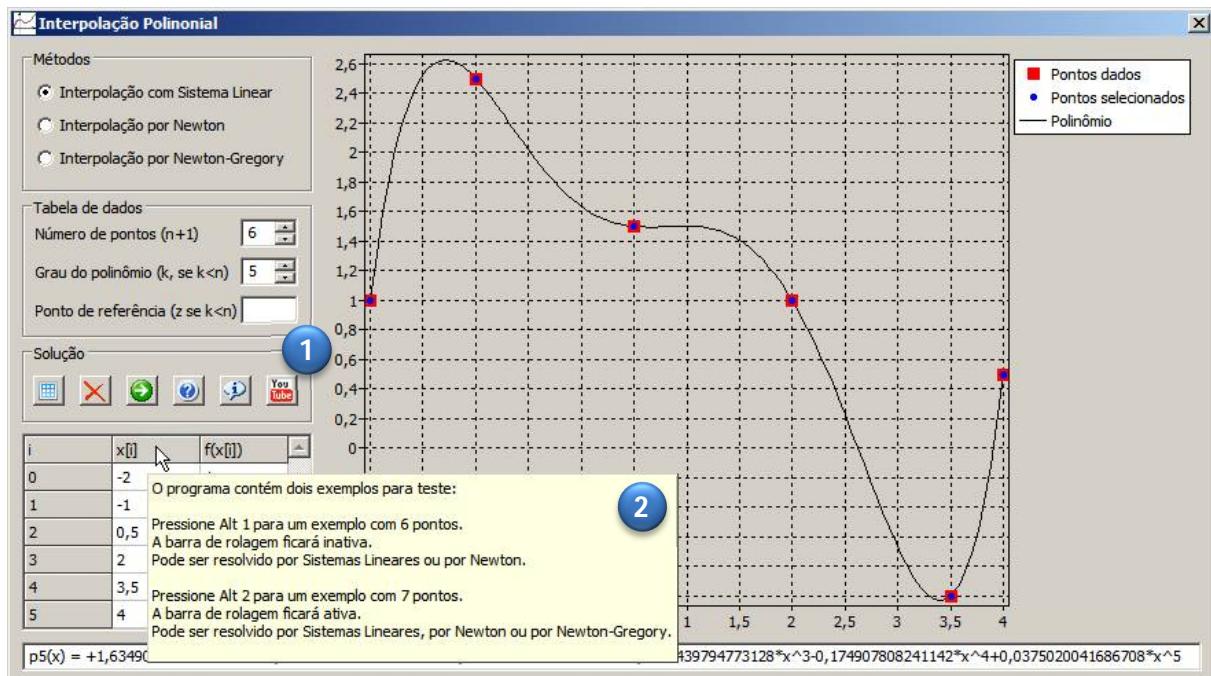
procedure TForm1.SpeedButton6Click(Sender: TObject);
begin
  ...
  Aqui eu fiz a minha gracinha do YouTube.
  ...
end;

```

E agora quero me desculpar, pois eu fiz quase tudo, tirando sua chance de fazer sozinho. Só de raiva, você deveria preencher rapidamente os blocos cinza que eu deixei para você fazer e entregar ontem.

Ao utilizar o programa exemplo, não deixe de cutucar o botão YouTube. 1

Ao utilizar o programa exemplo, estacione o apontador do mouse sobre a grade de dados e leia sobre dois exemplos já embutidos no programa. 2



## 11- Ajuste de curvas

Dona Mariquinha e seu aluno Joãozinho

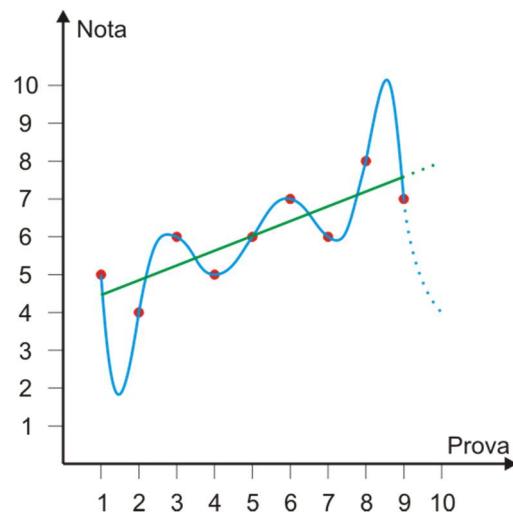


Pode ser que você não saiba que, há algum tempo, os professores ensinavam e os alunos aprendiam. Mas isto foi há algum tempo, pois hoje, tudo mudou. Há algum tempo, quando os alunos não aprendiam, seus pais chamavam sua atenção na cara da professora. Agora, ridicularizam com a *tia* na cara do ~~futuro delinquente~~ filho. Pois é, mas a pergunta ainda é a mesma: *que notas são essas?*

Então, aproveitando essa coisa de professora e tia, segue a lista de notas do Joãozinho, em uma disciplina qualquer, ao longo de um tempo qualquer.

Prova	Nota
1	5
2	4
3	6
4	5
5	6
6	7
7	6
8	8
9	7
10	?

Quanto o pestinha deverá ter na P10?



Os pontos da tabela, a interpolação e uma reta que representa a tendência destes pontos são apresentados no gráfico.

A interpolação destes pontos indica que na Prova 10 o Joãozinho deverá ter uma Nota próxima de 4.

A reta de tendência indica que a Nota deverá ser próxima de 8.

Vamos ser justos com o Joãozinho. Olhe para a figura, contemple a série de dados e diga qual é o valor mais representativo para a nota da P10. Próximo de 4 ou próximo de 8? Mas é um pestinha.

## Interpolar ou ajustar? Eis a questão

A interpolação requer que o polinômio interpolador tenha, nos pontos dados, o mesmo valor da função que gerou os pontos, seja a função conhecida ou não.

O primeiro fato facilmente perceptível é que se a quantidade de pontos for muito grande, o polinômio terá um grau elevado. Isto não é conveniente.

Outro fato é que se os dados foram obtidos experimentalmente ou por observações, podem conter erros e o polinômio interpolador poderá não representar adequadamente a função geradora dos pontos, pois passará por todos os pontos, incluindo os que estejam carregados de erros.

Nada melhor, nestes casos, do que encontrar uma função que represente os pontos da melhor maneira possível, sem que esta função seja um polinômio de grau elevado e sem que a construção desta função se preocupe exageradamente com alguns pontos coletados erradamente.

Certamente você já entendeu tudo, ou seja, o que é Interpolação e o que é Ajuste e para o que serve cada uma destas técnicas.

## Ajuste e Tendência

Dada uma nuvem de pontos obtida experimentalmente ou por observação ou a partir de uma função conhecida, pode-se ajustar uma ou mais funções à nuvem de pontos. Ajustadas várias funções, algumas representarão os pontos de forma mais adequada. Com a melhor função ajustada é possível calcular valores intermediários aos pontos dados ou, até calcular valores para pontos fora da faixa de dados conhecida, ou seja, extrapolar. Claro que a função ajustada representa, da melhor maneira possível os pontos, mas não se pode exagerar com a extração, pois a tendência oferecida pela função ajustada poderá facilmente ser descaracterizada por fatores não conhecidos que modifiquem o comportamento da função ajustada para pontos fora da faixa de pontos conhecidos. Quer um bom exemplo? Pegue a série dos 10 últimos anos do índice Bovespa e estime o valor para os próximos 12 meses. Acho que não vai dar certo.

## Técnica dos Mínimos Quadrados

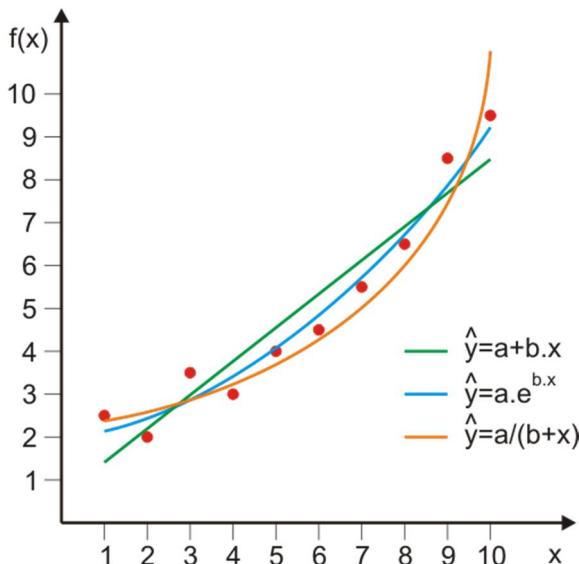
Para facilitar o desenvolvimento das funções ajustadas, considera-se a seguinte notação.

Os pontos conhecidos são chamados  $(x_i, y_i)$  e são provenientes de uma função  $f(x)$ , ou  $y(x)$ , conhecida ou não.

A função ajustada é conhecida como  $\hat{f}(x)$  ou  $\hat{y}(x)$  e os pontos estimados são chamados  $(x_i, \hat{y}_i)$ .

A distância entre  $y_i$  e  $\hat{y}_i$  é chamado de  $e_i$ , representando o desvio ou erro entre o valor conhecido de  $f(x_i)$  e o valor estimado de  $\hat{f}(x_i)$ .

Considere uma nuvem de pontos e diferentes ajustes efetuados para representar adequadamente estes pontos. A figura que segue apresenta esta situação.



Qual o melhor ajuste?

Visualmente, podem-se passar horas contemplando as três curvas e nada concluir.

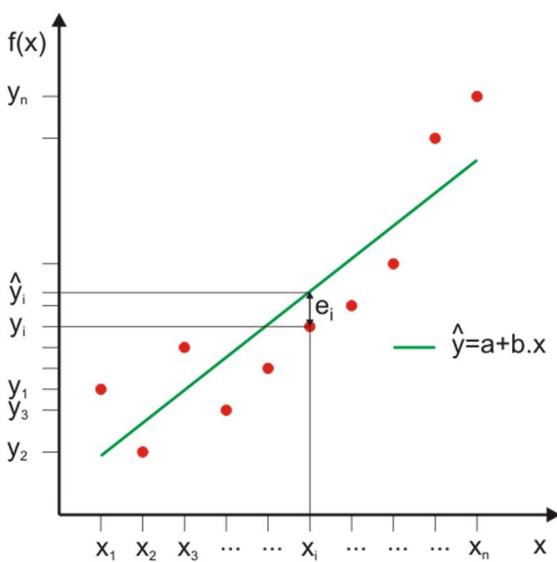
Há que existir um critério para comparação.

Considere que a soma dos desvios entre os pontos dados e os pontos calculados em cada uma das curvas seja a mínima possível.

Pronto. Menor soma dos desvios ou dos erros. Boa ideia.

O critério para comparação do melhor ajuste é, também, um conceito para geração dos ajustes.

Considere que a curva ajustada tenha, para cada ponto dado, um desvio, ou um erro, ou uma distância, como apresentado na figura que segue.



Na figura é apresentado um ajuste de reta, ou seja,  $\hat{y} = a + b \cdot x$ .

Então, para cada ponto  $x_i$ , há um desvio  $e_i$ , que pode ser medido como  $y_i - \hat{y}_i$  e o que se quer é que a soma dos desvios seja mínima.

Quando  $y_i > \hat{y}_i$ ,  $y_i - \hat{y}_i$  é positivo e quando  $y_i < \hat{y}_i$ ,  $y_i - \hat{y}_i$  é negativo. Minimizar a soma dos desvios  $e_i = y_i - \hat{y}_i$  seria absurdo. Há infinitas curvas que fornecerão desvios tais que sua soma seja nula e que não representam os pontos dados.

Alguém já deve estar pensando: *bota módulo*. Não é uma boa ideia. Primeiramente não conheço o animal que bota módulos e, além disto, minimizar indica derivar e derivar a função módulo não é uma tarefa agradável.

Outro alguém já deve estar pensando: *bota quadrado*. Boa ideia, mas sem botar. Para tornar os desvios positivos e minimizar sua soma, podem-se utilizar os quadrados dos desvios. E assim a técnica é conhecida como Técnica dos Mínimos Quadrados.

Para qualquer tipo de equação  $\hat{y}$  que se queira ajustar, escreve-se o somatório S dos desvios, onde

$S = \sum_{i=1}^n e_i^2 = \sum_{i=1}^n (y_i - \hat{y}_i)^2$  e, para minimizar, deriva-se este somatório em relação às variáveis que se deseja conhecer e iguala-se a zero. As variáveis que se deseja conhecer são os coeficientes da equação que se quer ajustar. Deve-se, então, minimizar a soma dos desvios quadráticos. Só isto.

### Ajuste de reta $\hat{y} = a + b \cdot x$

Dados  $n$  pontos  $(x_i, y_i)$ , deseja-se ajustar uma reta  $\hat{y} = a + b \cdot x$ , considerando que a soma dos desvios quadráticos seja mínima. Deve-se, então, minimizar a soma dos desvios quadráticos.

Esta soma é  $S = \sum_{i=1}^n e_i^2 = \sum_{i=1}^n (y_i - \hat{y}_i)^2$  e, no caso da reta  $\hat{y}_i = a + b \cdot x_i$ , tem-se  $S = \sum_{i=1}^n (y_i - a - b \cdot x_i)^2$ .

Como as variáveis são  $a$  e  $b$ , deriva-se  $S$  com relação a estas variáveis, iguala-se as expressões a zero e resolve-se o sistema, ou seja, escreve-se  $\frac{\partial S}{\partial a} = 0$ ,  $\frac{\partial S}{\partial b} = 0$  e resolve-se o sistema de equações, obtendo  $a$  e  $b$ .

$$\frac{\partial S}{\partial a} = \frac{\partial}{\partial a} \sum_{i=1}^n (y_i - a - b \cdot x_i)^2 = 2 \cdot \sum_{i=1}^n (y_i - a - b \cdot x_i) \cdot (-1) = 0$$

$$\frac{\partial S}{\partial b} = \frac{\partial}{\partial b} \sum_{i=1}^n (y_i - a - b \cdot x_i)^2 = 2 \cdot \sum_{i=1}^n (y_i - a - b \cdot x_i) \cdot (-x_i) = 0$$

$$\begin{cases} \sum_{i=1}^n (y_i - a - b \cdot x_i) = 0 \\ \sum_{i=1}^n (y_i - a - b \cdot x_i) \cdot (x_i) = 0 \end{cases} \Rightarrow \begin{cases} \sum_{i=1}^n y_i - \sum_{i=1}^n a - \sum_{i=1}^n b \cdot x_i = 0 \\ \sum_{i=1}^n x_i \cdot y_i - \sum_{i=1}^n a \cdot x_i - \sum_{i=1}^n b \cdot x_i^2 = 0 \end{cases} \Rightarrow \begin{cases} n \cdot a + \sum_{i=1}^n x_i \cdot b = \sum_{i=1}^n y_i \\ \sum_{i=1}^n x_i \cdot a + \sum_{i=1}^n x_i^2 \cdot b = \sum_{i=1}^n x_i \cdot y_i \end{cases}$$

Pode-se isolar  $a$  e  $b$ , para resolver o sistema, ou fazer isto como deve ser feito, através de um método para solução de sistemas lineares. Assim, a forma matricial é como segue.

$$\begin{bmatrix} n & \sum_{i=1}^n x_i \\ \sum_{i=1}^n x_i & \sum_{i=1}^n x_i^2 \end{bmatrix} \cdot \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^n y_i \\ \sum_{i=1}^n x_i \cdot y_i \end{pmatrix}$$

Para testar o método, considere o problema do pestinha do Joãozinho.

i	1	2	3	4	5	6	7	8	9	n	9
$x_i$	1	2	3	4	5	6	7	8	9	$\sum_{i=1}^n x_i$	45
$y_i$	5	4	6	5	6	7	6	8	7	$\sum_{i=1}^n y_i$	54
$x_i^2$	1	4	9	16	25	36	49	64	81	$\sum_{i=1}^n x_i^2$	285
$x_i \cdot y_i$	5	8	18	20	30	42	42	64	63	$\sum_{i=1}^n x_i \cdot y_i$	292

$$\begin{bmatrix} 9 & 45 \\ 45 & 285 \end{bmatrix} \cdot \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} 54 \\ 292 \end{pmatrix}$$

$$b = 0,3666667$$

$$a = 4,1666667$$

$$\hat{y} = 4,1666667 + 0,3666667 \cdot x$$

E como era esperado o pestinha do Joãozinho tem, para P10, o valor estimado 7,83.

Foi bom para você? Foi bom para o Joãozinho? A pergunta não é esta. Segue a pergunta correta.

E o ajuste foi bom?

## Coeficiente de Determinação

Dados  $n$  valores  $(x_i, y_i)$ , pode-se determinar uma função de ajuste que fornecerá os pontos  $(x_i, \hat{y}_i)$ . Os desvios deste ajuste são os  $n$  valores  $e_i$ . O coeficiente de Determinação é dado como segue.

$$R^2 = 1 - \left( \frac{n \cdot \sum_{i=1}^n e_i^2}{n \cdot \sum_{i=1}^n y_i^2 - \left( \sum_{i=1}^n y_i \right)^2} \right), \text{ onde } \sum_{i=1}^n e_i^2 = \sum_{i=1}^n (y_i - \hat{y}_i)^2 \text{ e } 0 \leq R^2 \leq 1.$$

O ajuste será tanto melhor quanto mais próximo de 1 for o Coeficiente de Determinação.

Pode-se utilizar o Coeficiente de Aderência que é  $1-R^2$ . Ou seja, os Coeficiente de Determinação e de Aderência somam 1. O ajuste será tanto melhor quanto mais próximo de 0 for o Coeficiente de Aderência.

No famoso caso do Joãozinho, tem-se o que segue.

i	1	2	3	4	5	6	7	8	9	n	9
$y_i$	5	4	6	5	6	7	6	8	7	$\sum_{i=1}^n y_i$	54
$y_i^2$	25	16	36	25	36	49	36	64	49	$\sum_{i=1}^n y_i^2$	336
$\hat{y}_i$	4,5333	4,9000	5,2666	5,6333	6,0000	6,3666	6,7333	7,1000	7,4666		
$e_i$	0,4666	-0,9	0,7333	-0,6333	0	0,6333	-0,7333	0,9	-0,4666		
$e_i^2$	0,2177	0,81	0,5377	0,4011	0	0,4011	0,5377	0,81	0,2177	$\sum_{i=1}^n e_i^2$	3,9333

$$R^2 = 1 - \left( \frac{9 \cdot 3,9333}{9,336 - 54^2} \right) = 0,6722.$$

Um bom ajuste deve ter um Coeficiente de Determinação maior que 0,8. O Joãozinho é um peste, mesmo. Mas vai acabar obtendo a média necessária nesta disciplina.

## Técnica dos Mínimos Quadrados - continuando

Para apresentar a técnica, foi utilizado um ajuste de reta, mas a técnica pode ser utilizada para qualquer tipo de função que se queira ajustar.

Considere que se queira ajustar um polinômio onde o grau é determinado pelo projetista do polinômio, isto é, dados  $n$  pontos, deseja-se ajustar um polinômio de grau  $m$ .

$$\text{Ajuste de polinômio } \hat{y} = a_0 + a_1 \cdot x + a_2 \cdot x^2 + \dots + a_m \cdot x^m$$

Dados  $n$  pontos  $(x_i, y_i)$ , deseja-se ajustar um polinômio  $\hat{y} = a_0 + a_1 \cdot x + a_2 \cdot x^2 + \dots + a_m \cdot x^m$ , considerando que a soma dos desvios quadráticos seja mínima. Deve-se, então, minimizar a soma dos desvios quadráticos.

Esta soma é  $S = \sum_{i=1}^n e_i^2 = \sum_{i=1}^n (y_i - \hat{y}_i)^2$  e, no caso do polinômio  $\hat{y} = a_0 + a_1 \cdot x + a_2 \cdot x^2 + \dots + a_m \cdot x^m$ , tem-se

$$S = \sum_{i=1}^n (y_i - a_0 - a_1 \cdot x_i - a_2 \cdot x_i^2 - \dots - a_m \cdot x_i^m)^2.$$

Como as variáveis são  $a_0, a_1, a_2, \dots, a_m$ , deriva-se  $S$  com relação a estas variáveis, iguala-se as expressões a zero e resolve-se o sistema.

Desenvolvendo, exatamente como desenvolvido para o caso da reta, o sistema é como segue.

$$\left[ \begin{array}{ccccc} n & \sum_{i=1}^n x_i & \sum_{i=1}^n x_i^2 & \dots & \sum_{i=1}^n x_i^m \\ \sum_{i=1}^n x_i & \sum_{i=1}^n x_i^2 & \sum_{i=1}^n x_i^3 & \dots & \sum_{i=1}^n x_i^{m+1} \\ \sum_{i=1}^n x_i^2 & \sum_{i=1}^n x_i^3 & \sum_{i=1}^n x_i^4 & \dots & \sum_{i=1}^n x_i^{m+2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \sum_{i=1}^n x_i^m & \sum_{i=1}^n x_i^{m+1} & \sum_{i=1}^n x_i^{m+2} & \dots & \sum_{i=1}^n x_i^{m+m} \end{array} \right] \cdot \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_m \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^n y_i \\ \sum_{i=1}^n x_i \cdot y_i \\ \sum_{i=1}^n x_i^2 \cdot y_i \\ \vdots \\ \sum_{i=1}^n x_i^m \cdot y_i \end{pmatrix}$$

Resolvendo o sistema, tem-se os coeficientes  $a_0, a_1, a_2, \dots, a_m$  do polinômio.

Observa-se que o sistema  $2x2$  contido neste sistema para solução dos coeficientes do polinômio, é exatamente igual ao sistema para solução dos coeficientes da reta, onde  $a$  e  $b$  são identificados por  $a_0$  e  $a_1$ . Isto é óbvio, pois a reta é um polinômio de grau 1, ou seja,  $\hat{y} = a_0 + a_1 \cdot x$ .

Exagerando só um pouquinho, pode-se escrever o sistema para o polinômio de grau zero, que dará uma reta horizontal com cota equivalente à média dos valores de  $y_i$ . Neste caso,  $\hat{y} = a_0$  e o sistema fornecerá  $a_0 = \sum_{i=1}^n y_i / n$ .

Então chegou a hora de você fazer um belo programa para calcular ajustes de reta e polinômio.

Mas o assunto continua, com ajustes de outros tipos de equações, utilizando a mesma técnica dos mínimos quadrados.

### Ajustes de equações exponenciais e outras mais

Considere equações como  $\hat{y} = ab^x$ ,  $\hat{y} = ae^{bx}$ ,  $\hat{y} = e^{ax+b}$ , ou  $\hat{y} = \frac{a}{b+x}$ .

Efetua-se uma transformação de variáveis para que a equação escolhida tenha a forma de uma equação de reta. A seguir resolve-se o sistema obtido com a equação transformada e, a seguir, efetua-se a transformação inversa, para obter os valores das variáveis para a equação original.

Seguem alguns exemplos e, depois, você poderá construir vários outros. Poderá fazer isto por amor à arte ou por necessidade. Necessidade? Sim, tem uma prova vindo por aí.

### Exponencial do tipo $\hat{y} = a \cdot b^x$

Considere um ajuste para uma equação do tipo  $\hat{y} = ab^x$ .

Efetua-se a transformação  $\ln(\hat{y}) = \ln(ab^x)$ . Desenvolvendo, tem-se  $\ln(\hat{y}) = \ln(a) + x \cdot \ln(b)$ . Pode-se reescrever como  $\hat{Y} = A + x \cdot B$  ou  $\hat{Y} = A + B \cdot x$ .

Para cada  $y_i$  dado, calcula-se  $Y_i = \ln(y_i)$ . Com os pontos  $(x_i, Y_i)$ , resolve-se o sistema que encontra  $A$  e  $B$  da reta. A seguir, efetua-se a transformação inversa para obter  $a$  e  $b$  da equação  $\hat{y} = ab^x$ .

A operação inversa é como segue. Como  $A = \ln(a)$ , então  $a = e^A$ . Como  $B = \ln(b)$ , então  $b = e^B$ .

Usando novamente o problema do cara dura do Joãozinho e ajustando uma equação do tipo  $\hat{y} = ab^x$ , tem-se o que segue, lembrando que  $Y_i = \ln(y_i)$  e que  $x_i \cdot Y_i = x_i \cdot \ln(y_i)$ .

i	1	2	3	4	5	6	7	8	9	$\sum_{i=1}^n x_i$	9
$x_i$	1	2	3	4	5	6	7	8	9	$\sum_{i=1}^n x_i$	45
$y_i$	5	4	6	5	6	7	6	8	7		
$Y_i$	1,609438	1,386294	1,791759	1,609438	1,791759	1,94591	1,791759	2,079442	1,94591	$\sum_{i=1}^n Y_i$	15,95171
$x_i^2$		1	4	9	16	25	36	49	64	$\sum_{i=1}^n x_i^2$	285
$x_i \cdot Y_i$	1,609438	2,772589	5,375278	6,437752	8,958797	11,67546	12,54232	16,63553	17,51319	$\sum_{i=1}^n x_i \cdot Y_i$	83,52035

$$\begin{bmatrix} 9 & 45 \\ 45 & 285 \end{bmatrix} \cdot \begin{pmatrix} A \\ B \end{pmatrix} = \begin{pmatrix} 15,95171 \\ 83,52035 \end{pmatrix} \quad B = 0,0626967, \quad b = 1,06470387 \\ A = 1,4589289, \quad a = 4,30134989 \quad \hat{y} = 4,30134989 \cdot 1,06470387^x$$

A estimativa para a nota da P10 é 8,05. Estou começando ficar com raiva deste moleque. Merece ser reprovado.

### Exponencial do tipo $\hat{y} = a \cdot e^{bx}$

Considere um ajuste para uma equação do tipo  $\hat{y} = a \cdot e^{bx}$ .

Efetua-se a transformação  $\ln(\hat{y}) = \ln(e^{bx})$ . Desenvolvendo, tem-se  $\ln(\hat{y}) = \ln(a) + b \cdot x \cdot \ln(e)$  e, portanto,  $\ln(\hat{y}) = \ln(a) + b \cdot x$ . Pode-se reescrever como  $\hat{Y} = A + b \cdot x$ .

Para cada  $y_i$  dado, calcula-se  $Y_i = \ln(y_i)$ . Com os pontos  $(x_i, Y_i)$ , resolve-se o sistema que encontra  $A$  e  $b$  da reta. A seguir, efetua-se a transformação inversa para obter  $a$  e  $b$  da equação  $\hat{y} = a \cdot e^{bx}$ .

A operação inversa é como segue. Como  $A = \ln(a)$ , então  $a = e^A$ .

Para facilitar os cálculos, sugere-se a tabela que segue, lembrando que  $Y_i = \ln(y_i)$ .

i	1	2	3	...	n	n
$X_i$	$x_1$	$x_2$	$x_3$	...	$x_n$	$\sum_{i=1}^n x_i$
$y_i$	$y_1$	$y_2$	$y_3$	...	$y_n$	
$Y_i$	$Y_1$	$Y_2$	$Y_3$	...	$Y_n$	$\sum_{i=1}^n Y_i$
$X_i^2$	$x_1^2$	$x_2^2$	$x_3^2$	...	$x_n^2$	$\sum_{i=1}^n X_i^2$
$X_i \cdot Y_i$	$x_1 \cdot Y_1$	$x_2 \cdot Y_2$	$x_3 \cdot Y_3$	...	$x_n \cdot Y_n$	$\sum_{i=1}^n X_i \cdot Y_i$

$$\begin{bmatrix} n & \sum_{i=1}^n X_i \\ \sum_{i=1}^n X_i & \sum_{i=1}^n X_i^2 \end{bmatrix} \cdot \begin{pmatrix} A \\ b \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^n Y_i \\ \sum_{i=1}^n X_i \cdot Y_i \end{pmatrix}$$

Calcula-se A e b

Transforma-se  $a = e^A$

Obtém-se  $\hat{y} = a \cdot e^{b \cdot x}$

### Geométrica do tipo $\hat{y} = a \cdot x^b$

Considere um ajuste para uma equação do tipo  $\hat{y} = ax^b$ .

Efetua-se a transformação  $\ln(\hat{y}) = \ln(ax^b)$ . Desenvolvendo, tem-se  $\ln(\hat{y}) = \ln(a) + b \cdot \ln(x)$ . Pode-se reescrever como  $\hat{Y} = A + b \cdot X$ .

Para cada  $y_i$  dado, calcula-se  $Y_i = \ln(y_i)$  e para cada  $x_i$  dado, calcula-se  $X_i = \ln(x_i)$ . Com os pontos  $(X_i, Y_i)$ , resolve-se o sistema que encontra A e b da reta. A seguir, efetua-se a transformação inversa para obter a e b da equação  $\hat{y} = ax^b$ .

A operação inversa é como segue. Como  $A = \ln(a)$ , então  $a = e^A$ .

Para facilitar os cálculos, sugere-se a tabela que segue, lembrando que  $Y_i = \ln(y_i)$ .

i	1	2	3	...	n	n
$X_i$	$x_1$	$x_2$	$x_3$	...	$x_n$	
$y_i$	$y_1$	$y_2$	$y_3$	...	$y_n$	
$X_i$	$X_1$	$X_2$	$X_3$	...	$X_n$	$\sum_{i=1}^n X_i$
$Y_i$	$Y_1$	$Y_2$	$Y_3$	...	$Y_n$	$\sum_{i=1}^n Y_i$
$X_i^2$	$X_1^2$	$X_2^2$	$X_3^2$	...	$X_n^2$	$\sum_{i=1}^n X_i^2$
$X_i \cdot Y_i$	$X_1 \cdot Y_1$	$X_2 \cdot Y_2$	$X_3 \cdot Y_3$	...	$X_n \cdot Y_n$	$\sum_{i=1}^n X_i \cdot Y_i$

$$\begin{bmatrix} n & \sum_{i=1}^n X_i \\ \sum_{i=1}^n X_i & \sum_{i=1}^n X_i^2 \end{bmatrix} \cdot \begin{pmatrix} A \\ b \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^n Y_i \\ \sum_{i=1}^n X_i \cdot Y_i \end{pmatrix}$$

Calcula-se A e b

Transforma-se  $a = e^A$

Obtém-se  $\hat{y} = ax^b$

### Hiperbólica do tipo $\hat{y} = \frac{1}{a+b \cdot x}$

Considere um ajuste para uma equação do tipo  $\hat{y} = \frac{1}{a+b \cdot x}$ .

Efetua-se a transformação  $\frac{1}{\hat{y}} = a + b \cdot x$ . Pode-se reescrever como  $\hat{Y} = a + b \cdot x$ .

Para cada  $y_i$  dado, calcula-se  $Y_i = 1/y_i$ . Com os pontos  $(x_i, Y_i)$ , resolve-se o sistema que encontra a e b da reta que também serão a e b da hipérbole e obtém-se a equação  $\hat{y} = \frac{1}{a+b \cdot x}$ .

Para facilitar os cálculos, sugere-se a tabela que segue, lembrando que  $Y_i = 1/y_i$ .

i	1	2	3	...	n	$\sum_{i=1}^n x_i$
$x_i$	$x_1$	$x_2$	$x_3$	...	$x_n$	
$y_i$	$y_1$	$y_2$	$y_3$	...	$y_n$	
$Y_i$	$Y_1$	$Y_2$	$Y_3$	...	$Y_n$	$\sum_{i=1}^n Y_i$
$x_i^2$	$x_1^2$	$x_2^2$	$x_3^2$	...	$x_n^2$	$\sum_{i=1}^n x_i^2$
$x_i \cdot Y_i$	$x_1 \cdot Y_1$	$x_2 \cdot Y_2$	$x_3 \cdot Y_3$	...	$x_n \cdot Y_n$	$\sum_{i=1}^n x_i \cdot Y_i$

$$\begin{bmatrix} n & \sum_{i=1}^n x_i \\ \sum_{i=1}^n x_i & \sum_{i=1}^n x_i^2 \end{bmatrix} \cdot \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^n Y_i \\ \sum_{i=1}^n x_i \cdot Y_i \end{pmatrix}$$

Calcula-se a e b

$$\text{Obtém-se } \hat{y} = \frac{1}{a+b \cdot x}$$

### Outras equações, para você desenvolver o procedimento

Aposto que você está muito empolgado com a brincadeira e com um desejo incontido de desenvolver o procedimento para outros tipos de equações.

Então, segue uma lista para você se divertir.

$$1- \hat{y} = ab^{ex}$$

$$2- \hat{y} = \frac{x}{a+b \cdot x}$$

$$3- \hat{y} = \frac{1}{1+e^{a+b \cdot x}}$$

$$4- \hat{y} = 1+ae^{bx}$$

$$5- \hat{y} = a + b \cdot \ln(x)$$

$$6- \hat{y} = a + \frac{b}{x}$$

$$7- \hat{y} = \frac{a}{b+x}$$

$$8- \hat{y} = \frac{ax}{b+x}$$

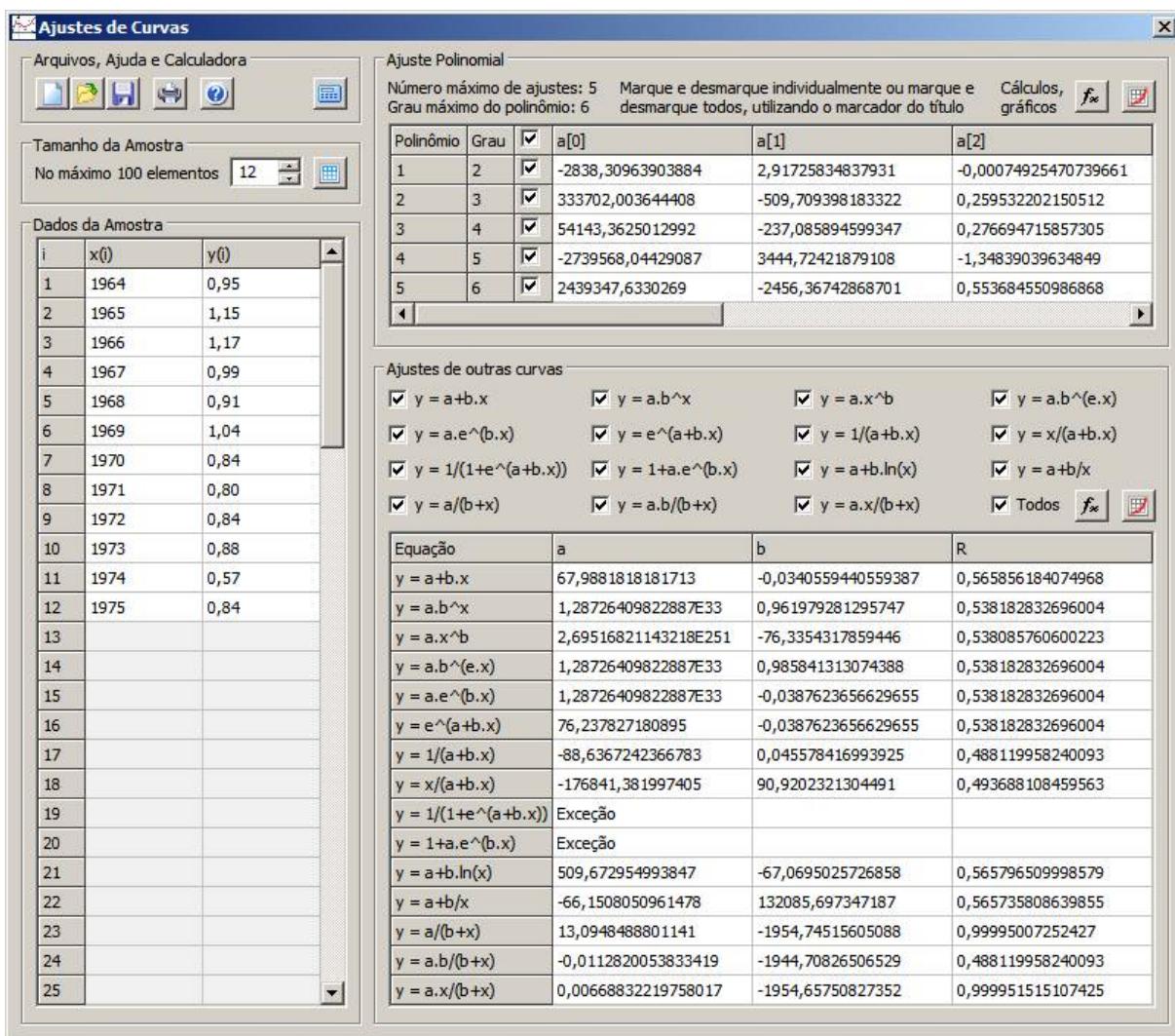
E, novamente, chegou a hora de você fazer um belíssimo programa para efetuar vários tipos de ajustes, mostrando as equações obtidas o Coeficiente de Determinação ou de Aderência e os gráficos das equações.

### Sugestões para seus programas

Um simpático programa, para resolver ajustes de curvas deverá permitir a definição da quantidade de pontos dados, a seleção do tipo de ajuste, efetuar os ajustes e desenhar o gráfico dos ajustes.

Além disto, o programa deve permitir que o usuário salve os dados digitados para uso posterior, pois a quantidade de dados pode ser muito grande e se o usuário quiser utilizar os mesmos dados para um novo ajuste, em outra ocasião, poderá carregar os dados que salvou anteriormente.

Um programa com as características descritas poderia ter a cara que segue.



Com este programa é possível:

- determinar diferentes tipos de ajustes para uma nuvem de pontos dada;
- calcular o coeficiente de determinação ou de aderência para cada ajuste;
- desenhar os pontos dados e os ajustes efetuados.

### Instruções claras e precisas sobre trabalhos computacionais para avaliação

Todas as instruções necessárias estão descritas no mesmo item, do Capítulo 7. Copiar e colar é muito fácil, mas não é nada elegante, agradável, simpático ou conveniente repetir tudo aquilo que já está escrito em capítulo anterior. Além disto, se você já entregou algum trabalho, já sabe como deve proceder. Caso contrário, veja as instruções no Capítulo 7. E antes de entregar, leia as instruções em <http://sacoman.dco.fc.unesp.br/mnc/pet/>.

Se você optar em fazer um programa que tenha a aparência da figura apresentada ou similar, poderá continuar lendo para ter algumas explicações que poderão ser úteis para você.

### Um exemplo de programa

Minha ajuda será apenas na fabricação da interface. O resto é com você. Depois de pronto, limpe todo lixo gerado durante a elaboração do projeto, compacte com Zip, Rar ou qualquer outro compac-

tador, com o sugestivo nome T5-RA-V1.ext, onde RA é o número de seu Registro Acadêmico, ext é a extensão fornecida pelo seu compactador, V1 é a primeira versão e entregue via endereço eletrônico de recepção que você já sabe qual é. Não sabe qual é? Então leia tudo sobre o assunto no endereço <http://sacoman.dco.fc.unesp.br/mnc/pet/>.

Então, mãos a obra.

Crie um diretório apropriado para desenvolver seu projeto (e.g.: X:\MNC-Trabalhos\T5). Organize seu ambiente de trabalho e crie subdiretórios necessários para o projeto, como, por exemplo, um diretório chamado Figuras, para as figuras dos botões e ícone e outro chamado Exemplo, para o exemplo executável que está à sua disposição.

Copie de <http://sacoman.dco.fc.unesp.br/mnc/material/> o arquivo M5-Ajustes de Curvas.rar e descompacte. Mova AjustesCurvas.exe e AjustesCurvas.chm para o diretório exemplo, para executar quando quiser. Mova as figuras para o diretório Figuras.

Inicie seu ambiente de projeto, no meu caso, Lazarus, ajeite as janelas da forma como achar conveniente e salve o projeto com o nome AjustesCurvas.lpr e a unidade principal com o nome Principal. Pressione o botão Nova Fórmula e salve dando o nome Grafico para esta nova unidade. Pressione o botão Nova Fórmula e salve dando o nome Sobre para esta nova unidade.

No Editor de Código, pressione a aba Sobre, pressione F12 para ver a fórmula, atribua Caption=Sobre o programa Ajustes de Curvas, BorderStyle=bsDialog e o resto, faça como considerar conveniente, pois você já leu este tipo de instrução em todos os outros exemplos anteriores. Você pode colocar um ou mais componentes TLabel com os textos que quiser e um componente TImage com sua foto, afirmindo que você fez o programa sozinho. Mas não deixe de ver o do programa exemplo que está à sua disposição.

No Editor de Código, pressione a aba Grafico, pressione F12 para ver a fórmula, atribua Height=560, Width=840, Caption=Ajustes de Curvas e Position=poDesktopCenter.

Lance, na fórmula, um componente TChart e atribua Align=alClient e Legend Visible=True.

Cutique o botão direito do mouse, selecione Edit series, pressione o botão Add, selecione Line series, para adicionar Chart1LineSeries1. Na aba de propriedades, mude o nome para Chart1LineSeries0, para utilizar os nomes Chart1LineSeries1 até Chart1LineSeries15 para as funções. Atribua LineType=ltNone, ShowPoints=True, Pointer Style=psCircle, Brush Color=clBlack e Title=Pontos da amostra.

Pressione o botão Add, selecione Line series, para adicionar Chart1LineSeries1. Repita a operação para criar Chart1LineSeries2 até Chart1LineSeries15. As propriedades de cada série serão atribuídas no programa, pois fica mais fácil.

No programa, além das propriedades das séries, será verificado o local onde a fórmula é fechada, para ser aberta na mesma posição, sem ficar incomodando o usuário.

Pressione ESC para selecionar a fórmula e, em eventos, dê um toque duplo em OnCreate, OnShow,

OnCloseQuery e OnClose, para criar as rotinas acionadas por esses eventos e, a seguir, copie descaradamente o trecho de programa que segue.

```

implementation
{$R *.lfm}
{ TForm2 }

var
  PrimeiraVez: Boolean; // Primeira vez que a fôma é aberta
  F2L, F2T: Integer; // Left e Top da fôma 2, para abrir na mesma posição em que foi fechada

procedure TForm2.FormCreate(Sender: TObject);
begin
  // Controle do local onde a fôma se localiza
  PrimeiraVez := True;
  // Atribuir cores para as séries (mais fácil fazer no programa do que no componente)
  {
    // Escolhidas pela ordem, na paleta de 16 cores, sem considerar se são ou não simpáticas
    Chart1LineSeries1.SeriesColor := clMaroon; Chart1LineSeries2.SeriesColor := clGreen;
    Chart1LineSeries3.SeriesColor := clOlive; Chart1LineSeries4.SeriesColor := clNavy;
    Chart1LineSeries5.SeriesColor := clPurple; Chart1LineSeries6.SeriesColor := clTeal;
    Chart1LineSeries7.SeriesColor := clGray; Chart1LineSeries8.SeriesColor := clSilver;
    Chart1LineSeries9.SeriesColor := clRed; Chart1LineSeries10.SeriesColor := clLime;
    Chart1LineSeries11.SeriesColor := clYellow; Chart1LineSeries12.SeriesColor := clBlue;
    Chart1LineSeries13.SeriesColor := clFuchsia; Chart1LineSeries14.SeriesColor := clAqua;
    Chart1LineSeries15.SeriesColor := clWhite;
  }
  // Escolhidas no cubo de cores de http://sacoman.dco.fc.unesp.br/aulas/html/07b.html
  Chart1LineSeries1.SeriesColor := $00FFFF00; Chart1LineSeries1.LinePen.Style := psDash;
  Chart1LineSeries2.SeriesColor := $00FFCC00;
  Chart1LineSeries3.SeriesColor := $00FF9900; Chart1LineSeries3.LinePen.Style := psDash;
  Chart1LineSeries4.SeriesColor := $00FF3300;
  Chart1LineSeries5.SeriesColor := $003300FF; Chart1LineSeries5.LinePen.Style := psDash;
  Chart1LineSeries6.SeriesColor := $009900FF;
  Chart1LineSeries7.SeriesColor := $00FF00FF; Chart1LineSeries7.LinePen.Style := psDash;
  Chart1LineSeries8.SeriesColor := $00FF0099;
  Chart1LineSeries9.SeriesColor := $00FF0033; Chart1LineSeries9.LinePen.Style := psDash;
  Chart1LineSeries10.SeriesColor := $00FF3300;
  Chart1LineSeries11.SeriesColor := $00FF9900; Chart1LineSeries11.LinePen.Style := psDash;
  Chart1LineSeries12.SeriesColor := $00FFF000;
  Chart1LineSeries13.SeriesColor := $0099FF00; Chart1LineSeries13.LinePen.Style := psDash;
  Chart1LineSeries14.SeriesColor := $0033FF00;
  Chart1LineSeries15.SeriesColor := $0000FF33; Chart1LineSeries15.LinePen.Style := psDash;
  // Tamanho da pena
  Chart1LineSeries1.LinePen.Width := 2; Chart1LineSeries2.LinePen.Width := 2;
  Chart1LineSeries3.LinePen.Width := 2; Chart1LineSeries4.LinePen.Width := 2;
  Chart1LineSeries5.LinePen.Width := 2; Chart1LineSeries6.LinePen.Width := 2;
  Chart1LineSeries7.LinePen.Width := 2; Chart1LineSeries8.LinePen.Width := 2;
  Chart1LineSeries9.LinePen.Width := 2; Chart1LineSeries10.LinePen.Width := 2;
  Chart1LineSeries11.LinePen.Width := 2; Chart1LineSeries12.LinePen.Width := 2;
  Chart1LineSeries13.LinePen.Width := 2; Chart1LineSeries14.LinePen.Width := 2;
  Chart1LineSeries15.LinePen.Width := 2;
  // Limpar legendas com traço colorido
  Chart1LineSeries1.Legend.Visible := False; Chart1LineSeries2.Legend.Visible := False;
  Chart1LineSeries3.Legend.Visible := False; Chart1LineSeries4.Legend.Visible := False;
  Chart1LineSeries5.Legend.Visible := False; Chart1LineSeries6.Legend.Visible := False;
  Chart1LineSeries7.Legend.Visible := False; Chart1LineSeries8.Legend.Visible := False;
  Chart1LineSeries9.Legend.Visible := False; Chart1LineSeries10.Legend.Visible := False;
  Chart1LineSeries11.Legend.Visible := False; Chart1LineSeries12.Legend.Visible := False;
  Chart1LineSeries13.Legend.Visible := False; Chart1LineSeries14.Legend.Visible := False;
  Chart1LineSeries15.Legend.Visible := False;
end;

procedure TForm2.FormShow(Sender: TObject);
begin
  if PrimeiraVez then
    Exit;
  Left := F2L;
  Top := F2T;
end;

```

```

procedure TForm2.FormCloseQuery(Sender: TObject; var CanClose: boolean);
begin
  PrimeiraVez := False;
  F2L := Left;
  F2T := Top;
  CanClose := True;
end;

procedure TForm2.FormClose(Sender: TObject; var CloseAction: TCloseAction);
begin
  // Limpar gráficos anteriores
  Chart1LinesSeries0.Clear; // Pontos
  Chart1LinesSeries1.Clear; Chart1LinesSeries2.Clear; Chart1LinesSeries3.Clear; // Curvas
  Chart1LinesSeries4.Clear; Chart1LinesSeries5.Clear; Chart1LinesSeries6.Clear;
  Chart1LinesSeries7.Clear; Chart1LinesSeries8.Clear; Chart1LinesSeries9.Clear;
  Chart1LinesSeries10.Clear; Chart1LinesSeries11.Clear; Chart1LinesSeries12.Clear;
  Chart1LinesSeries13.Clear; Chart1LinesSeries14.Clear; Chart1LinesSeries15.Clear;
  // Limpar Legendas com traço colorido e funções anteriores
  Chart1LinesSeries1.Legend.Visible := False; Chart1LinesSeries2.Legend.Visible := False;
  Chart1LinesSeries3.Legend.Visible := False; Chart1LinesSeries4.Legend.Visible := False;
  Chart1LinesSeries5.Legend.Visible := False; Chart1LinesSeries6.Legend.Visible := False;
  Chart1LinesSeries7.Legend.Visible := False; Chart1LinesSeries8.Legend.Visible := False;
  Chart1LinesSeries9.Legend.Visible := False; Chart1LinesSeries10.Legend.Visible := False;
  Chart1LinesSeries11.Legend.Visible := False; Chart1LinesSeries12.Legend.Visible := False;
  Chart1LinesSeries13.Legend.Visible := False; Chart1LinesSeries14.Legend.Visible := False;
  Chart1LinesSeries15.Legend.Visible := False;
end;

```

Só isto. O preenchimento dos dados para desenhar os gráficos será feito na Unit Principal.

Agora, só falta montar a fôrma Principal e fazer o programa.

No Editor de Código, pressione a aba Principal, pressione F12 para ver a fôrma, atribua Caption=Ajustes de Curvas, BorderStyle=bsSingle, para BorderIcons, biMaximize=False e biMinimize=False, para Font, Name=Tahoma e Size=8, Height=676, Width=796, Position=poDesktopCenter e ShowHint=True.

Lance, na fôrma, alguns componentes como segue.

Componente	Caption	Left	Top	Height	Width
GroupBox1	Arquivos, Ajuda e Calculadora	8	4	52	232
GroupBox2	Tamanho da Amostra	8	62	49	232
GroupBox3	Dados da Amostra	8	116	552	232
GroupBox4	Ajuste Polinomial	248	4	204	542
GroupBox5	Ajustes de outras curvas	248	214	454	542

No GroupBox1, lance alguns componentes como segue.

Componente	Left	Top	Glyph	Componente	Left	Top	Glyph
SpeedButton1	8	4	new.gif	SpeedButton4	88	4	printe3.gif
SpeedButton2	32	4	open.gif	SpeedButton5	120	4	help2.gif
SpeedButton3	56	4	save.gif	SpeedButton6	197	4	calcul1.gif

No GroupBox2, lance alguns componentes como segue.

Componente	Caption	Left	Top	Glyph	Width	MinValue	MaxValue	Value
Label1	No máximo 100 elementos	8	4					
SpinEdit1		140	0		50	3	100	5
SpeedButton7		197	0	grid.gif				

No GroupBox3, lance um componente TStringGrid (StringGrid1) e atribua Left=8, Top=0, Height=524, Width=212, ColCount=3, RowCount=26 e Options goEditing=True.

No GroupBox4, lance alguns componentes como segue.

Componente	Caption	Left	Top	Glyph
Label2	Número máximo de ajustes: 5 Grau máximo do polinômio: 6	8	4	
Label3	Marque e desmarque individualmente ou marque e desmarque todos, utilizando o marcador do título	166	4	
Label4	Cálculos, gráficos	425	4	
SpeedButton8		475	6	function.gif
SpeedButton9		507	6	Gráfico.bmp

Lance um componente TStringGrid (StringGrid2) e atribua Left=8, Top=35, Height=141, Width=522, ColCount=11, RowCount=6, DefaultColWidth=80 e FixedCols=3.

Sobre a grade (StringGrid2) lance seis componentes TCheckBox, alterando seus nomes, como segue.

Name	Caption	Left	Top
CBP0	Remova	98	37
CBP1	Remova	98	57
CBP2	Remova	98	77

Name	Caption	Left	Top
CBP3	Remova	98	97
CBP4	Remova	98	117
CBP5	Remova	98	137

No GroupBox5, lance alguns componentes como segue.

Name	Caption	Left	Top
CBF1	$y = a+b.x$	8	4
CBF2	$y = a.b^x$	146	4
CBF3	$y = a.x^b$	284	4
CBF4	$y = a.b^{(e.x)}$	418	4
CBF5	$y = a.e^{(b.x)}$	8	28
CBF6	$y = e^{(a+b.x)}$	146	28
CBF7	$y = 1/(a+b.x)$	284	28
CBF8	$y = x/(a+b.x)$	418	28

Name	Caption	Left	Top	Glyph
CBF9	$y = 1/(1+e^{(a+b.x)})$	8	52	
CBF10	$y = 1+a.e^{(b.x)}$	146	52	
CBF11	$y = a+b.\ln(x)$	284	52	
CBF12	$y = a+b/x$	418	52	
CBF13	$y = a/(b+x)$	8	76	
CBF14	$y = a.b/(b+x)$	146	76	
CBF15	$y = a.x/(b+x)$	284	76	
CBF0	Todos	418	76	
SpeedButton10		475	74	function.gif
SpeedButton11		507	74	Gráfico.bmp

Lance um componente TStringGrid (StringGrid3) e atribua Left=8, Top=102, Height=324, Width=522, ColCount=4, RowCount=16 e DefaultColWidth=80.

Lance mais três componentes, em qualquer lugar que queira, pois não são visíveis. Os componentes são TOpenDialog, TSaveDialog e TPopupMenu.

Selecione OpenDialog1 e, em Propriedade, atribua os valores que seguem.

Propriedade	Valor
DefaultExt	.txt
Filter	Arquivo texto de dados para ajuste (*.txt) Todos os arquivos (*.*)
InitialDir	.
Title	Abrir arquivo existente (com um espaço antes do texto)

Selecione SaveDialog1 e, em Propriedade, atribua os valores que seguem.

Propriedade	Valor
DefaultExt	.txt
Filter	Arquivo texto de dados para ajuste (*.txt) Todos os arquivos (*.*)
InitialDir	.
Title	Salvar arquivo como (com um espaço antes do texto)

Selecione PopupMenu1, dê um duplo toque, para abrir o Editor de Menu, selecione o primeiro item e atribua Caption=Ajuda e, em Bitmap, carregue help.gif.

Pressione o primeiro item com o botão direito do mouse, selecione Inserir Novo Item (após) e atribua Caption=Sobre o programa e, em Bitmap, carregue inform1.gif.

As propriedades Hint dos componentes TSpeedButton são longas e é melhor escrevê-las na rotina disparada pelo evento Oncreate da fôrma. Se você não utilizar Hint nem precisará desta rotina e, se escrever textos curtos, pode ser que prefira escrever na aba de propriedades.

A fôrma está preparada e pode-se iniciar a programação das rotinas.

No início do programa, defina algumas rotinas na seção private, para que elas pertençam ao objeto Form1. Desta forma, estas rotinas poderão utilizar os componentes da fôrma sem nenhuma informação extra.

```

private
  { private declarations }
  function LerDadosXYGrade: Boolean;
  procedure AjustePolí (m: Integer);
  procedure Ajuste1;
  procedure Ajuste2;
  procedure Ajuste3;
  procedure Ajuste4;
  procedure Ajuste5;
  procedure Ajuste6;
  procedure Ajuste7;
  procedure Ajuste8;
  procedure Ajuste9;
  procedure Ajuste10;
  procedure Ajuste11;
  procedure Ajuste12;
  procedure Ajuste13;
  procedure Ajuste14;
  procedure Ajuste15;
public
  { public declarations }
end;

```

Cutucando:

- Cutuque a fôrma e, em Eventos, dê um toque duplo em OnCreate e em OnShow, para criar as rotinas FormCreate e FormShow.
- Selecione o TCheckBox CBP0 e, em Eventos, dê um toque duplo em OnClick, para criar a rotina CBP0Click.
- Selecione todos os outros componentes TCheckBox, ou seja, CBP1 até CBP5, tomando o cuidado de não selecionar CBP0 e, em Eventos, dê um toque duplo em OnClick, para criar a rotina CBP1Click. Modifique o nome desta rotina para CBPTodosClick. Selecione cada um deles, separadamente, para verificar se todos eles executam a mesma rotina CBPTodosClick.

- Selecione o TCheckBox CBF0 e, em Eventos, dê um toque duplo em OnClick, para criar a rotina CBF0Click.
- Selecione todos os outros componentes TCheckBox, ou seja, CBF1 até CBF15, tomando o cuidado de não selecionar CBF0 e, em Eventos, dê um toque duplo em OnClick, para criar a rotina CBF1Click. Modifique o nome desta rotina para CBFTodosClick. Selecione cada um deles, separadamente, para verificar se todos eles executam a mesma rotina CBFTodosClick.
- Selecione PopupMenu1, dê um toque duplo para abrir o Editor de Menu, selecione o item Ajuda e, em Eventos, dê um toque duplo em OnClick, para criar a rotina MenuItem1Click.
- Com o Editor de Menu ainda aberto, selecione o item Sobre o programa e, em Eventos, dê um toque duplo em OnClick, para criar a rotina MenuItem2Click.
- Selecione SpeedButton1 e, em Eventos, dê um toque duplo em OnClick, para criar a rotina SpeedButton1Click.
- Repita a operação anterior com os botões SpeedButton2 até SpeedButton11, para criar as rotinas SpeedButton2Click até SpeedButton11Click.
- Selecione SpinEdit1 e, em Eventos, dê um toque duplo em OnExit, para criar a rotina SpinEdit1Exit.
- Selecione StringGrid1 e, em Eventos, dê um toque duplo em OnClick, para criar a rotina StringGrid1Click.
- Com a grade ainda selecionada, em Eventos, dê um toque duplo em OnPrepareCanvas, para criar a rotina StringGrid1PrepareCanvas.
- Com a grade ainda selecionada, em Eventos, dê um toque duplo em OnSelectCell, para criar a rotina StringGrid1SelectCell.

Você criou 8 rotinas em uma ordem lógica e o Lazarus as colocou em ordem alfabética. No meu projeto, eu as reordenei, para continuarem na ordem lógica. Se considerar que deve reordená-las, reordene de acordo com a lista de criação, de FormShow até SpeedButton5 ou SpeedButton6, que ficará mais fácil para entender o programa e, principalmente, para *descaradamente copiá-lo*.

Agora é só copiar ou fazer o seu. Inicialmente são definidas algumas unidades necessárias na cláusula uses, alguma variáveis globais, rotinas auxiliares para salvar e ler dados e, depois, as rotinas dos eventos. Além disto, as rotinas de cálculos dos ajustes estão ao final do programa. Fica mais fácil de encontrá-las para modificar durante a elaboração e teste do programa.

```
implementation
{$R *.lfm}
{ TForm1 }

uses
  Grafi co, Sobre,
  Math, Windows, LCLIntf; // Math, Windows e LCLIntf para usar Power, GetSystemMetrics e OpenDocument

var
  EFH, EFV, ESH, ESV: Byte; // Valores para Grade que são diferentes em diferentes versões do Windows
  DRH, DCW: Integer;
  n: Integer; // Número de pontos dados para Ajustes
  aj1, aj2, aj3, aj4, aj5, aj6, aj7, aj8, aj9,
  aj10, aj11, aj12, aj13, aj14, aj15: Boolean; // Tipos de Ajustes (Form2)
  xGrade, yGrade: array[1..100] of Real;

// Salva Grade em arquivo (somente linhas e colunas selecionadas)
procedure SalvaGrade(Grade: TStringGrid; Linhas, Colunas: Integer; const NomeArquivo: TFileName);
var
  f: TextFile;
  i, j: Integer;
begin
  AssignFile(f, UTF8ToSys(NomeArquivo));
```

```

Rewrite(f);
with Grade do
begin
  // Número de Colunas e Linhas
  Writeln(f, Colunas);
  Writeln(f, Linhas);
  // Valores das Caselas
  for i := 0 to Colunas-1 do
    for j := 0 to Linhas-1 do
      Writeln(f, Cells[i, j]);
end;
CloseFile(f);
end;

// Carrega Grade de arquivo (somente linhas e colunas selecionadas)
procedure CarregaGrade(Grade: TStringGrid; var Linhas, Colunas: Integer; const NomeArquivo: TFileName);
var
  f: TextFile;
  iTmp, i, j: Integer;
  strTemp: String;
begin
  AssignFile(f, UTF8ToSys(NomeArquivo));
  Reset(f);
  with Grade do
  begin
    // Número de Colunas
    Readln(f, iTmp);
    Colunas := iTmp;
    // Número de Linhas
    Readln(f, iTmp);
    Linhas := iTmp;
    // Valores das Caselas
    for i := 0 to Colunas-1 do
      for j := 0 to Linhas-1 do
        begin
          Readln(f, strTemp);
          Cells[i, j] := strTemp;
        end;
  end;
  CloseFile(f);
end;

procedure TForm1.FormCreate(Sender: TObject);
begin
  SpeedButton1.Hint := 'Novo' +#10+
    'Apaga todos os dados e resultados' +#10+
    'para iniciar novo problema de ajuste.' ;
  SpeedButton2.Hint := 'Abrir' +#10+
    'Permite selecionar arquivo de dados' +#10+
    'para preencher a grade dos pontos.' ;
  SpeedButton3.Hint := 'Salvar' +#10+
    'Permite salvar dados da grade' +#10+
    'dos pontos de um problema.' ;
  SpeedButton4.Hint := 'Imprimir' +#10+
    'Aprenda como imprimir cursando PAW' +#10+
    'Programação Avançada para Windows.' ;
  SpeedButton5.Hint := 'Ajuda' +#10+
    'Apresenta a ajuda do programa' +#10+
    'e informações sobre o programa.' ;
  SpeedButton6.Hint := 'Calculadora' +#10+
    'Precisando de uma calculadora para' +#10+
    'ajudar no preenchimento dos dados' +#10+
    'ou para conferir os resultados?' +#10+
    'Cutuque e execute a calculadora do Windows.' ;
  SpeedButton7.Hint := 'Prepara a grade para entrada de dados' +#10+
    'A grade tem tamanho fixo de 100 linhas.' +#10+
    'Cutuque este botão para definir a' +#10+
    'região da grade para entrada de pontos.' ;
  SpeedButton8.Hint := 'Ajuste polinomial' +#10+
    'Calcula os coeficientes dos polinômios' +#10+
    'selecionados utilizando os dados da' +#10+
    'grade de entrada.' ;

```

```

SpeedButton9.Hint := 'Ajuste polinomials' +#10+
'Desenha os gráficos dos polinômios' +#10+
'selecionados e os dados da grade' +#10+
'de entrada.' +#10+
'É necessário calcular os coeficientes' +#10+
'dos polinômios antes de desenhar o' +#10+
'gráfico.' ;
SpeedButton10.Hint := 'Ajuste de funções' +#10+
'Calcula os coeficientes das funções' +#10+
'selecionadas utilizando os dados da' +#10+
'grade de entrada.' ;
SpeedButton11.Hint := 'Ajuste de funções' +#10+
'Desenha os gráficos das funções' +#10+
'selecionadas e os dados da grade' +#10+
'de entrada.' +#10+
'É necessário calcular os coeficientes' +#10+
'das funções antes de desenhar o' +#10+
'gráfico.' ;
end;

procedure TForm1.FormShow(Sender: TObject);
var
  i: Byte;
begin
  //StringGrid1.Visible := False;
  //StringGrid1.ColWidths[0] := 32;
  // Medidas para moldura e scrollbar da Grade (são diferentes para cada versão do Windows)
  EFH := GetSystemMetrics(SM_CYBORDER)+GetSystemMetrics(SM_CXFRAME); // Espessura Frame Horizontal
  EFV := GetSystemMetrics(SM_CXBORDER)+GetSystemMetrics(SM_CYFRAME); // Espessura Frame Vertical
  ESH := GetSystemMetrics(SM_CYHSCROLL); // Espessura Scroll Horizontal
  ESV := GetSystemMetrics(SM_CXVSCROLL); // Espessura Scroll Vertical
  // Medidas da altura da linha e comprimento da coluna de cada casela
  // Mesmo definindo os valores no projeto, elas mudam de acordo com a versão do Windows
  // Forçar altura da casela ser a mesma em diferentes versões do Windows
  StringGrid1.DefaultRowHeight := 20;
  StringGrid2.DefaultRowHeight := 20;
  StringGrid3.DefaultRowHeight := 20;
  DRH := StringGrid1.DefaultRowHeight; // Default Row Height
  DCW := StringGrid1.DefaultColWidth; // Default Col Width
  with StringGrid1 do
    begin
      RowCount := 101;
      DefaultColWidth := 80;
      ColWidths[0] := 32;
      Width := 192+EFH+ESH;
      Cells[0,0] := 'i'; Cells[1,0] := 'x(i)'; Cells[2,0] := 'y(i)';
      for i := 1 to RowCount-1 do
        Cells[0,i] := IntToStr(i);
      Visible := True;
    end;
  with StringGrid2 do
    begin
      DefaultColWidth := 136;
      ColWidths[0] := 52;
      ColWidths[1] := 32;
      ColWidths[2] := 26;
      Width := 518+EFH;
      Cells[0,0] := 'Polinômio';
      Cells[0,1] := '1';
      Cells[0,2] := '2';
      Cells[0,3] := '3';
      Cells[0,4] := '4';
      Cells[0,5] := '5';
      Cells[1,0] := 'Grau';
      Cells[1,1] := '2';
      Cells[1,2] := '3';
      Cells[1,3] := '4';
      Cells[1,4] := '5';
      Cells[1,5] := '6';
      Cells[3,0] := 'a[0]';
      Cells[4,0] := 'a[1]';
    end;
end;

```

```

Cells[5, 0] := 'a[2]';
Cells[6, 0] := 'a[3]';
Cells[7, 0] := 'a[4]';
Cells[8, 0] := 'a[5]';
Cells[9, 0] := 'a[6]';
Cells[10, 0] := 'R';
end;
with StringGrid3 do
begin
  DefaultColWidth := 136;
  ColWidths[0] := 110;
  Width := 518+EFH;
  Cells[0, 0] := 'Equação';
  Cells[1, 0] := 'a';
  Cells[2, 0] := 'b';
  Cells[3, 0] := 'R';
  Cells[0, 1] := 'y = a+b. x';
  Cells[0, 2] := 'y = a. b^x';
  Cells[0, 3] := 'y = a. x^b';
  Cells[0, 4] := 'y = a. b^(e. x)';
  Cells[0, 5] := 'y = a. e^(b. x)';
  Cells[0, 6] := 'y = e^(a+b. x)';
  Cells[0, 7] := 'y = 1/(a+b. x)';
  Cells[0, 8] := 'y = x/(a+b. x)';
  Cells[0, 9] := 'y = 1/(1+e^(a+b. x))';
  Cells[0, 10] := 'y = 1+a. e^(b. x)';
  Cells[0, 11] := 'y = a+b. ln(x)';
  Cells[0, 12] := 'y = a+b/x';
  Cells[0, 13] := 'y = a/(b+x)';
  Cells[0, 14] := 'y = a. b/(b+x)';
  Cells[0, 15] := 'y = a. x/(b+x)';
end;
// Ajustar a grade
SpeedButton7.Click;
end;

procedure TForm1.CBPOClick(Sender: TObject);
begin
  // Se CBPO estiver marcado, marcar todos; se estiver desmarcado, desmarcar todos
  if CBPO.Checked then
    begin
      CBP1.Checked := True; CBP2.Checked := True; CBP3.Checked := True;
      CBP4.Checked := True; CBP5.Checked := True;
    end
  else
    begin
      CBP1.Checked := False; CBP2.Checked := False; CBP3.Checked := False;
      CBP4.Checked := False; CBP5.Checked := False;
    end;
end;

procedure TForm1.CBPTodosClick(Sender: TObject);
begin
  // Marcar CBPO se todos estiverem marcados e desmarcar se todos estiverem desmarcados
  // Desnecessário - Foi feito só para dar um ar profissional na coisa
  if CBP1.Checked and CBP2.Checked and CBP3.Checked and CBP4.Checked and CBP5.Checked then
    CBPO.State := cbChecked
  else
    CBPO.State := cbUnchecked;
end;

procedure TForm1.CBF0Click(Sender: TObject);
begin
  // Se CBF0 estiver marcado, marcar todos; se estiver desmarcado, desmarcar todos
  if CBF0.Checked then
    begin
      CBF1.Checked := True; CBF2.Checked := True; CBF3.Checked := True;
      CBF4.Checked := True; CBF5.Checked := True; CBF6.Checked := True;
      CBF7.Checked := True; CBF8.Checked := True; CBF9.Checked := True;
      CBF10.Checked := True; CBF11.Checked := True; CBF12.Checked := True;
      CBF13.Checked := True; CBF14.Checked := True; CBF15.Checked := True;
    end
end;

```

```

else
begin
  CBF1.Checked := False;  CBF2.Checked := False;  CBF3.Checked := False;
  CBF4.Checked := False;  CBF5.Checked := False;  CBF6.Checked := False;
  CBF7.Checked := False;  CBF8.Checked := False;  CBF9.Checked := False;
  CBF10.Checked := False; CBF11.Checked := False; CBF12.Checked := False;
  CBF13.Checked := False; CBF14.Checked := False; CBF15.Checked := False;
end;
end;

procedure TForm1.CBFTodpsClick(Sender: TObject);
begin
  // Marcar CBP0 se todos estiverem marcados e desmarcar se todos estiverem desmarcados
  // Desnecessário - Foi feito só para dar um ar profissional na coisa
  if CBF1.Checked and CBF2.Checked and CBF3.Checked and CBF4.Checked and CBF5.Checked and
    CBF6.Checked and CBF7.Checked and CBF8.Checked and CBF9.Checked and CBF10.Checked and
    CBF11.Checked and CBF12.Checked and CBF13.Checked and CBF14.Checked and CBF15.Checked then
    CBP0.State := cbChecked
  else
    CBP0.State := cbUnchecked;
end;

procedure TForm1.Menu1Click(Sender: TObject);
begin
  // Ajuda
  if not OpenDocument('Aj ustesCurvas.chm') then
    MessageDlg('Informação', 'Arquivo de Aj uda' +#10+'Aj ustesCurvas.chm' +#10+'não foi encontrado',
               mtInformation, [mbOk], 0);
end;

procedure TForm1.Menu2Click(Sender: TObject);
begin
  // Sobre
  Form3.ShowModal;
end;

procedure TForm1.SpeedButton1Click(Sender: TObject);
begin
  // Novo
  // Limpa Grade de pontos dados
  StringGrid1.Clean(1, 1, 2, 100, [gzNormal]);
  SpinEdit1.Value := 5;
  SpeedButton7.Click;
  // Limpa Grade de coeficientes dos polinômios
  StringGrid2.Clean(3, 1, 10, 5, [gzNormal]);
  // Limpa CheckBox de Ajustes Polinomiais;
  CBP0.State := cbUnchecked; // Desmarca CBP0
  CBP0Click(Self); // Executa rotina CBP0Click que desmarcará todos
  // Limpa Grade de coeficientes dos ajustes
  StringGrid3.Clean(1, 1, 3, 15, [gzNormal]);
  // Limpa CheckBox de Ajustes não Polinomiais;
  CBF0.State := cbUnchecked; // Desmarca CBF0
  CBF0Click(Self); // Executa rotina CBF0Click que desmarcará todos
end;

procedure TForm1.SpeedButton2Click(Sender: TObject);
var
  Linhas, Colunas: Integer;
begin
  // Abrir
  if OpenDialog1.Execute then
  begin
    // Limpa Grades e CheckBox acionando o botão Novo
    SpeedButton1.Click;
    CarregaGrade(StringGrid1, Linhas, Colunas, OpenDialog1.FileName);
    n := Linhas-1;
    SpinEdit1.Value := n;
    SpeedButton7.Click;
  end;
end;

```

```

procedure TForm1.SpeedButton3Click(Sender: TObject);
begin
  // Salvar
  if SaveDialog1.Execute then
    SaveGrade(StringGrid1, n+1, 3, SaveDialog1.FileName);
end;

procedure TForm1.SpeedButton4Click(Sender: TObject);
begin
  // Imprimir
  ...
  Possibilidades:
  - Aprender como imprimir e fazer a rotina;
  - Executar uma rotina informando que não foi implementada (ShowMessage, MessageDlg, ...);
  - Tirar o botão Imprimir do programa.

  Veja o meu programa.
  Fiz uma rotina só para você.
  ...
end;

procedure TForm1.SpeedButton5Click(Sender: TObject);
begin
  // Abrir menu Popup
  PopupMenu1.Popup;
end;

procedure TForm1.SpeedButton6Click(Sender: TObject);
begin
  // Calculadora
  OpenDocument('Calc.exe');
end;

procedure TForm1.SpeedButton7Click(Sender: TObject);
begin
  n := SpinEdit1.Value;
  StringGrid1.SetFocus;
  StringGrid1.Refresh;
end;

...
A próxima rotina verifica quais são os polinômios selecionados na grade e executa a rotina necessária para as soluções.
Para cada polinômio selecionado, a rotina de solução é executada, passando como parâmetro o grau do polinômio.

A rotina de solução está no final do programa.
...

procedure TForm1.SpeedButton8Click(Sender: TObject);
begin
  // Ler dados (x,y) da grade, verificar ajustes selecionados e efetuar ajustes

  // Verificar se dados de StringGrid1 são válidos
  if not LerDadosXYGrade then
    Exit;
  // Tipos de Ajustes
  aj1 := CBP1.Checked; aj2 := CBP2.Checked; aj3 := CBP3.Checked;
  aj4 := CBP4.Checked; aj5 := CBP5.Checked;
  // Tem ajuste selecionado?
  if not aj1 and not aj2 and not aj3 and not aj4 and not aj5 then
  begin
    MessageDlg('Advertência', 'Selecione ao menos um tipo de ajuste', mtWarning, [mbOk], 0);
    Exit;
  end;
  // Limpa Grade de coeficientes dos ajustes;
  StringGrid2.Clear(3, 1, 10, 5, [gzNormal]);
  // Efetua ajustes selecionados
  if aj1 then try AjustePol(2) except StringGrid2.Cells[3, 1] := 'Exceção' end;
  if aj2 then try AjustePol(3) except StringGrid2.Cells[3, 2] := 'Exceção' end;
  if aj3 then try AjustePol(4) except StringGrid2.Cells[3, 3] := 'Exceção' end;
  if aj4 then try AjustePol(5) except StringGrid2.Cells[3, 4] := 'Exceção' end;
  if aj5 then try AjustePol(6) except StringGrid2.Cells[3, 5] := 'Exceção' end;
end;

```



• • •

A próxima rotina verifica, na grade, quais são os polinômios que foram calculados e cria o gráfico correspondente.

É longa, mas não passa de repetição para cada polinômio que deve ser desenhado.

• • •

```

procedure TForm1.SpeedButton9Click(Sender: TObject);
var
  i: Integer;
  TemAjuste: array[0..5] of Boolean;
  x, y: array[1..100] of Real;
  Xmin, Xmax, DeltaX: Real;
  novoX: array[1..101] of Real;
  novon: Integer;
  a: array[0..6] of Real;
begin
  // Gráficos das funções polinomiais

  // Se fôrma de gráficos estiver aberta, fechar
  Form2.Close;

  // Tem alguma função ajustada? E quais são?
  FillChar(TemAjuste, SizeOf(TemAjuste), False);
  for i := 1 to 5 do
    if (StringGrid2.Cells[3, i] <> '') and (StringGrid3.Cells[3, i] <> 'Exceção') then
      begin
        TemAjuste[0] := True;
        TemAjuste[i] := True;
      end;
  if not TemAjuste[0] then
    begin
      MessageDlg('Advertência', 'Não há curvas ajustadas para desenhar gráficos', mtWarning, [mbOk], 0);
      Exit;
    end;

  // Se tem ajuste, tem dados em xGrade e yGrade
  for i := 1 to n do
  begin
    x[i] := xGrade[i];
    y[i] := yGrade[i];
  end;

  // Traçar gráficos para os ajustes existentes
  with Form2 do
  begin
    // Atribuir cores para as séries (feito na Unit Grafico)
    // Limpar gráficos anteriores (feito na Unit Grafico)
    // Limpar títulos anteriores (feito na Unit Grafico)

    // (x,y)
    with Chart1LineSeries0 do
      for i := 1 to n do
        AddXY(x[i], y[i]);

    // Cálculo de 101 pontos ordenados para desenhar as curvas
    Xmin := x[1];
    for i := 1 to n do
      if x[i] < Xmin then
        Xmin := x[i];
    Xmax := x[1];
    for i := 1 to n do
      if x[i] > Xmax then
        Xmax := x[i];
    DeltaX := (Xmax - Xmin)/100;
    for i := 1 to 101 do
      novoX[i] := Xmin + (i - 1)*DeltaX;
    novon := 101;
  end;
end;

```

```

// p2(x) = a0 + a1.x + a2.x^2
if TemAjuste[1] then
begin
  a[0] := StrToFloat(StringGrid2.Cells[3, 1]);
  a[1] := StrToFloat(StringGrid2.Cells[4, 1]);
  a[2] := StrToFloat(StringGrid2.Cells[5, 1]);
  with Chart1LineSeries1 do
    try
      Legend.Visible := True;
      for i := 1 to novoN do
        AddXY(novoX[i], a[0]+a[1]*novoX[i]+a[2]*Sqr(novoX[i]));
      Title := 'p2(x)';
    except
      Title := 'p2(x) (Erro)';
    end;
end;

// p3(x) = a0 + a1.x + a2.x^2 + a3.x^3
if TemAjuste[2] then
begin
  a[0] := StrToFloat(StringGrid2.Cells[3, 2]);
  a[1] := StrToFloat(StringGrid2.Cells[4, 2]);
  a[2] := StrToFloat(StringGrid2.Cells[5, 2]);
  a[3] := StrToFloat(StringGrid2.Cells[6, 2]);
  with Chart1LineSeries2 do
    try
      Legend.Visible := True;
      for i := 1 to novoN do
        AddXY(novoX[i], a[0]+a[1]*novoX[i]+a[2]*Sqr(novoX[i])+a[3]*Power(novoX[i], 3));
      Title := 'p3(x)';
    except
      Title := 'p3(x) (Erro)';
    end;
end;

// p4(x) = a0 + a1.x + a2.x^2 + a3.x^3 + a4.x^4
if TemAjuste[3] then
begin
  a[0] := StrToFloat(StringGrid2.Cells[3, 3]);
  a[1] := StrToFloat(StringGrid2.Cells[4, 3]);
  a[2] := StrToFloat(StringGrid2.Cells[5, 3]);
  a[3] := StrToFloat(StringGrid2.Cells[6, 3]);
  a[4] := StrToFloat(StringGrid2.Cells[7, 3]);
  with Chart1LineSeries3 do
    try
      Legend.Visible := True;
      for i := 1 to novoN do
        AddXY(novoX[i], a[0]+a[1]*novoX[i]+a[2]*Sqr(novoX[i])+a[3]*Power(novoX[i], 3)+a[4]*Power(novoX[i], 4));
      Title := 'p4(x)';
    except
      Title := 'p4(x) (Erro)';
    end;
end;

// p5(x) = a0 + a1.x + a2.x^2 + a3.x^3 + a4.x^4 + a5.x^5
if TemAjuste[4] then
begin
  a[0] := StrToFloat(StringGrid2.Cells[3, 4]);
  a[1] := StrToFloat(StringGrid2.Cells[4, 4]);
  a[2] := StrToFloat(StringGrid2.Cells[5, 4]);
  a[3] := StrToFloat(StringGrid2.Cells[6, 4]);
  a[4] := StrToFloat(StringGrid2.Cells[7, 4]);
  a[5] := StrToFloat(StringGrid2.Cells[8, 4]);
  with Chart1LineSeries4 do
    try
      Legend.Visible := True;
      for i := 1 to novoN do
        AddXY(novoX[i], a[0]+a[1]*novoX[i]+a[2]*Sqr(novoX[i])+a[3]*Power(novoX[i], 3)+a[4]*Power(novoX[i], 4)+a[5]*Power(novoX[i], 5));
      Title := 'p5(x)';
    except
      Title := 'p5(x) (Erro)';
    end;
end;

```

```

// p6(x) = a0 + a1.x + a2.x^2 + a3.x^3 + a4.x^4 + a5.x^5 + a6.x^6
if TemAjuste[5] then
begin
  a[0] := StrToInt(GridView1.Cells[3, 5]);
  a[1] := StrToInt(GridView1.Cells[4, 5]);
  a[2] := StrToInt(GridView1.Cells[5, 5]);
  a[3] := StrToInt(GridView1.Cells[6, 5]);
  a[4] := StrToInt(GridView1.Cells[7, 5]);
  a[5] := StrToInt(GridView1.Cells[8, 5]);
  a[6] := StrToInt(GridView1.Cells[9, 5]);
  with Chart1LineSeries5 do
    try
      Legend.Visible := True;
      for i := 1 to novoN do
        AddXY(novoX[i], a[0]+a[1]*novoX[i]+a[2]*Sqr(novoX[i])+a[3]*Power(novoX[i], 3)+a[4]*Power(novoX[i], 4)+a[5]*Power(novoX[i], 5)+a[6]*Power(novoX[i], 6));
      Title := 'p6(x)';
    except
      Title := 'p6(x) (Erro)';
    end;
  end;
  Show; // Form2.Show;
end;

```

•••

A próxima rotina verifica quais são as funções selecionados na grade e executa as rotinas necessárias para as soluções.

A rotina de solução está no final do programa.

•••

```

procedure TForm1.SpeedButton10Click(Sender: TObject);
begin
  // Ler dados (x,y) da grade, verificar ajustes selecionados e efetuar ajustes

  // Verificar se dados de StringGrid1 são válidos
  if not LerDadosXYGrade then
    Exit;
  // Tipos de Ajustes
  aj1 := CBF1.Checked; aj2 := CBF2.Checked; aj3 := CBF3.Checked;
  aj4 := CBF4.Checked; aj5 := CBF5.Checked; aj6 := CBF6.Checked;
  aj7 := CBF7.Checked; aj8 := CBF8.Checked; aj9 := CBF9.Checked;
  aj10 := CBF10.Checked; aj11 := CBF11.Checked; aj12 := CBF12.Checked;
  aj13 := CBF13.Checked; aj14 := CBF14.Checked; aj15 := CBF15.Checked;
  // Tem ajuste selecionado?
  if not aj1 and not aj2 and not aj3 and not aj4 and not aj5 and
    not aj6 and not aj7 and not aj8 and not aj9 and not aj10 and
    not aj11 and not aj12 and not aj13 and not aj14 and not aj15 then
  begin
    MessageDlg('Advertência', 'Selecionou ao menos um tipo de ajuste', mtWarning, [mbOk], 0);
    Exit;
  end;
  // Limpa Grade de coeficientes dos ajustes;
  StringGrid3.Clear(1, 1, 3, 15, [gzNormal]);
  // Efetua ajustes selecionados
  if aj1 then try Ajuste1 except StringGrid3.Cells[1, 1] := 'Exceção' end;
  if aj2 then try Ajuste2 except StringGrid3.Cells[1, 2] := 'Exceção' end;
  if aj3 then try Ajuste3 except StringGrid3.Cells[1, 3] := 'Exceção' end;
  if aj4 then try Ajuste4 except StringGrid3.Cells[1, 4] := 'Exceção' end;
  if aj5 then try Ajuste5 except StringGrid3.Cells[1, 5] := 'Exceção' end;
  if aj6 then try Ajuste6 except StringGrid3.Cells[1, 6] := 'Exceção' end;
  if aj7 then try Ajuste7 except StringGrid3.Cells[1, 7] := 'Exceção' end;
  if aj8 then try Ajuste8 except StringGrid3.Cells[1, 8] := 'Exceção' end;
  if aj9 then try Ajuste9 except StringGrid3.Cells[1, 9] := 'Exceção' end;
  if aj10 then try Ajuste10 except StringGrid3.Cells[1, 10] := 'Exceção' end;
  if aj11 then try Ajuste11 except StringGrid3.Cells[1, 11] := 'Exceção' end;
  if aj12 then try Ajuste12 except StringGrid3.Cells[1, 12] := 'Exceção' end;
  if aj13 then try Ajuste13 except StringGrid3.Cells[1, 13] := 'Exceção' end;
  if aj14 then try Ajuste14 except StringGrid3.Cells[1, 14] := 'Exceção' end;
  if aj15 then try Ajuste15 except StringGrid3.Cells[1, 15] := 'Exceção' end;
end;

```

•••

A próxima rotina verifica, na grade, quais são as funções que foram calculadas e cria o gráfico correspondente.

É longa, mas não passa de repetição para cada função que deve ser desenhada.

•••

```

procedure TForm1.SpeedButton11Click(Sender: TObject);
var
  i: Integer;
  TemAjuste: array[0..15] of Boolean;
  x, y: array[1..100] of Real;
  Xmin, Xmax, DeltaX: Real;
  novoX: array[1..101] of Real;
  novon: Integer;
  a, b: Real;
begin
  // Gráficos das funções não polinomiais

  // Se fôrma de gráficos estiver aberta, fechar
  Form2.Close;

  // Tem alguma função ajustada? E quais são?
  FillChar(TemAjuste, SizeOf(TemAjuste), False);
  for i := 1 to 15 do
    if (StringGrid3.Cells[1, i] <> '') and (StringGrid3.Cells[1, i] <> 'Exceção') then
      begin
        TemAjuste[0] := True;
        TemAjuste[i] := True;
      end;
  if not TemAjuste[0] then
    begin
      MessageDlg('Advertência', 'Não há curvas ajustadas para desenhar gráficos', mtWarning, [mbOK], 0);
      Exit;
    end;

  // Se tem ajuste, tem dados em xGrade e yGrade
  for i := 1 to n do
    begin
      x[i] := xGrade[i];
      y[i] := yGrade[i];
    end;

  // Traçar gráficos para os ajustes existentes
  with Form2 do
    begin
      // Atribuir cores para as séries (feito na Unit Grafico)
      // Limpar gráficos anteriores (feito na Unit Grafico)
      // Limpar títulos anteriores (feito na Unit Grafico)

      // (x,y)
      with Chart1LineSeries0 do
        for i := 1 to n do
          AddXY(x[i], y[i]);

      // Cálculo de 101 pontos ordenados para desenhar as curvas
      Xmin := x[1];
      for i := 1 to n do
        if x[i] < Xmin then
          Xmin := x[i];
      Xmax := x[1];
      for i := 1 to n do
        if x[i] > Xmax then
          Xmax := x[i];
      DeltaX := (Xmax - Xmin) / 100;
      for i := 1 to 101 do
        novoX[i] := Xmin + (i - 1) * DeltaX;
      novon := 101;

      // y = a + b.x (dizem por aí que para reta bastam 2 pontos)
      if TemAjuste[1] then
        begin
          a := StrToFloat(StringGrid3.Cells[1, 1]);
          b := StrToFloat(StringGrid3.Cells[2, 1]);
        end;
    end;
end;

```

```

with Chart1LineSeries1 do
begin
  AddXY(novoX[1], a+b*novoX[1]);
  AddXY(novoX[100], a+b*novoX[100]);
  Title := 'y = a + b.x';
  Legend.Visible := True;
end;
end;

// y = a.b^x
if TemAjuste[2] then
begin
  a := StrToFloat(StringGrid3.Cells[1, 2]);
  b := StrToFloat(StringGrid3.Cells[2, 2]);
  with Chart1LineSeries2 do
    try
      Legend.Visible := True;
      for i := 1 to novoN do
        AddXY(novoX[i], a*Math.Power(b, novoX[i]));
      Title := 'y = a.b^x';
    except
      Title := 'y = a.b^x (Error)';
    end;
end;

// y = a.x^b
if TemAjuste[3] then
begin
  a := StrToFloat(StringGrid3.Cells[1, 3]);
  b := StrToFloat(StringGrid3.Cells[2, 3]);
  with Chart1LineSeries3 do
    try
      Legend.Visible := True;
      for i := 1 to novoN do
        AddXY(novoX[i], a*Math.Power(novoX[i], b));
      Title := 'y = a.x^b';
    except
      Title := 'y = a.x^b (Error)';
    end;
end;

// y = a.b^(e.x)
if TemAjuste[4] then
begin
  a := StrToFloat(StringGrid3.Cells[1, 4]);
  b := StrToFloat(StringGrid3.Cells[2, 4]);
  with Chart1LineSeries4 do
    try
      Legend.Visible := True;
      for i := 1 to novoN do
        AddXY(novoX[i], a*Math.Power(b, (Exp(1)*novoX[i])));
      Title := 'y = a.b^(e.x)';
    except
      Title := 'y = a.b^(e.x) (Error)';
    end;
end;

// y = a.e^(b.x)
if TemAjuste[5] then
begin
  a := StrToFloat(StringGrid3.Cells[1, 5]);
  b := StrToFloat(StringGrid3.Cells[2, 5]);
  with Chart1LineSeries5 do
    try
      Legend.Visible := True;
      for i := 1 to novoN do
        AddXY(novoX[i], a*Math.Power(Exp(1), (b*novoX[i])));
      Title := 'y = a.e^(b.x)';
    except
      Title := 'y = a.e^(b.x) (Error)';
    end;
end;

```

```

// y = e^(a+b.x)
if TemAjuste[6] then
begin
  a := StrToFloat(StringGrid3.Cells[1, 6]);
  b := StrToFloat(StringGrid3.Cells[2, 6]);
  with Chart1LineSeries6 do
    try
      Legend.Visible := True;
      for i := 1 to novoN do
        AddXY(novoX[i], Math.Power(Exp(1), (a+b*novoX[i])));
      Title := 'y = e^(a+b.x)';
    except
      Title := 'y = e^(a+b.x) (Error)';
    end;
end;

// y = 1/(a+b.x)
if TemAjuste[7] then
begin
  a := StrToFloat(StringGrid3.Cells[1, 7]);
  b := StrToFloat(StringGrid3.Cells[2, 7]);
  with Chart1LineSeries7 do
    try
      Legend.Visible := True;
      for i := 1 to novoN do
        AddXY(novoX[i], 1/(a+b*novoX[i]));
      Title := 'y = 1/(a+b.x)';
    except
      Title := 'y = 1/(a+b.x) (Error)';
    end;
end;

// y = x/(a+b.x)
if TemAjuste[8] then
begin
  a := StrToFloat(StringGrid3.Cells[1, 8]);
  b := StrToFloat(StringGrid3.Cells[2, 8]);
  with Chart1LineSeries8 do
    try
      Legend.Visible := True;
      for i := 1 to novoN do
        AddXY(novoX[i], novoX[i]/(a+b*novoX[i]));
      Title := 'y = x/(a+b.x)';
    except
      Title := 'y = x/(a+b.x) (Error)';
    end;
end;

// y = 1/(1+e^(a+b.x))
if TemAjuste[9] then
begin
  a := StrToFloat(StringGrid3.Cells[1, 9]);
  b := StrToFloat(StringGrid3.Cells[2, 9]);
  with Chart1LineSeries9 do
    try
      Legend.Visible := True;
      for i := 1 to novoN do
        AddXY(novoX[i], 1/(1+Math.Power(Exp(1), (a+b*novoX[i)))));
      Title := 'y = 1/(1+e^(a+b.x))';
    except
      Title := 'y = 1/(1+e^(a+b.x)) (Error)';
    end;
end;

// y = 1+a.e^(b.x)
if TemAjuste[10] then
begin
  a := StrToFloat(StringGrid3.Cells[1, 10]);
  b := StrToFloat(StringGrid3.Cells[2, 10]);
  with Chart1LineSeries10 do
    try

```

```

Legend.Visible := True;
for i := 1 to novoN do
  AddXY(novoX[i], 1+a*Math.Power(Exp(1), (b*novoX[i])));
  Title := 'y = 1+a. e^(b. x)';
except
  Title := 'y = 1+a. e^(b. x) (Error)';
end;
end;

// y = a+b. ln(x)
if TemAjuste[11] then
begin
  a := StrToFloat(StringGrid3.Cells[1, 11]);
  b := StrToFloat(StringGrid3.Cells[2, 11]);
  with Chart1LineSeries11 do
    try
      Legend.Visible := True;
      for i := 1 to novoN do
        AddXY(novoX[i], a+b*Ln(novoX[i]));
        Title := 'y = a+b. ln(x)';
    except
      Title := 'y = a+b. ln(x) (Error)';
    end;
end;

// y = a+b/x
if TemAjuste[12] then
begin
  a := StrToFloat(StringGrid3.Cells[1, 12]);
  b := StrToFloat(StringGrid3.Cells[2, 12]);
  with Chart1LineSeries12 do
    try
      Legend.Visible := True;
      for i := 1 to novoN do
        AddXY(novoX[i], a+b/novoX[i]);
        Title := 'y = a+b/x';
    except
      Title := 'y = a+b/x (Error)';
    end;
end;

// y = a/(b+x)
if TemAjuste[13] then
begin
  a := StrToFloat(StringGrid3.Cells[1, 13]);
  b := StrToFloat(StringGrid3.Cells[2, 13]);
  with Chart1LineSeries13 do
    try
      Legend.Visible := True;
      for i := 1 to novoN do
        AddXY(novoX[i], a/(b+novoX[i]));
        Title := 'y = a/(b+x)';
    except
      Title := 'y = a/(b+x) (Error)';
    end;
end;

// y = a.b/(b+x)
if TemAjuste[14] then
begin
  a := StrToFloat(StringGrid3.Cells[1, 14]);
  b := StrToFloat(StringGrid3.Cells[2, 14]);
  with Chart1LineSeries14 do
    try
      Legend.Visible := True;
      for i := 1 to novoN do
        AddXY(novoX[i], a*b/(b+novoX[i]));
        Title := 'y = a. b/(b+x)';
    except
      Title := 'y = a. b/(b+x) (Error)';
    end;
end;

```

```

// y = a.x/(b+x)
if TemAjuste[15] then
begin
  a := StrToFloat(StringGrid3.Cells[1, 15]);
  b := StrToFloat(StringGrid3.Cells[2, 15]);
  with Chart1LineSeries15 do
    try
      Legend.Visible := True;
      for i := 1 to novoN do
        AddXY(novoX[i], a*novoX[i]/(b+novoX[i]));
      Title := 'y = a.x/(b+x)';
    except
      Title := 'y = a.x/(b+x) (Erro)';
    end;
  end;
  Show; // Form2.Show;
end;

procedure TForm1.SpinEdit1Exit(Sender: TObject);
begin
  SpeedButton7.Click;
end;

procedure TForm1.StringGrid1Click(Sender: TObject);
begin
  with StringGrid1 do
    begin
      if Row > n then
        Row := n;
      //Refresh;
    end;
end;

procedure TForm1.StringGrid1PrepareCanvas(Sender: TObject; aCol, aRow: Integer;
  aState: TGridDrawState);
begin
  with StringGrid1 do
    if (aCol > 0) and (aRow > n) then
      Canvas.Brush.Color := $00FOFOFO;
end;

procedure TForm1.StringGrid1SelectCell(Sender: TObject; aCol, aRow: Integer;
  var CanSelect: Boolean);
begin
  if aRow > n then
    CanSelect := False;
end;

{ -----
{           Rotinas para solução dos ajustes
{ ----- }

function TForm1.LerDadosXYGrade: Boolean;
var
  i: Integer;
begin
  Result := False;
  for i := 1 to n do
    try
      xGrade[i] := StrToFloat(StringGrid1.Cells[1, i]);
    except
      MessageDlg('Erro na entrada de dados', 'Dado inválido para x[' + IntToStr(i) + ']', mtError, [mbOk], 0);
      StringGrid1.Col := 1;
      StringGrid1.Row := i;
      StringGrid1.SetFocus;
      Exit;
    end;
  for i := 1 to n do
    try

```

```

yGrade[i] := StrToFloat(StringGrid1.Cells[2, i]);
except
  MessageDlg('Erro na entrada de dados', 'Dado inválido para y[' + IntToStr(i) + ']', mtError,
  [mbOk], 0);
  StringGrid1.Col := 2;
  StringGrid1.Row := i;
  StringGrid1.SetFocus;
  Exit;
end;
Result := True;
end;

procedure TForm1.AjustePol (m: Integer);
var
  A: array[1..7, 1..7] of Real; // Polinômios de no máximo ordem 6
  x, b: array[1..7] of Real; // Para montar A, usar xGrade e yGrade
  i, j, k, nLocal: Integer; // nLocal que é ordem do sistema (m+1)
  Soma, pivô: Real;
  SomaY, SomaY2, SomaE2, YChapeu, R2: Real;
begin
  // Lembre-se que a rotina é chamada com m = grau do polinômio
  // Montar matriz A de ordem m+1xm+1 e vetor b de ordem m+1 (nLocal=m+1)
  // Resolver sistema A.x = b
  // A solução x1, x2, ..., xm+1 são os coeficientes a0, a1, ..., am do polinômio

  FillChar(A, SizeOf(A), 0);
  FillChar(b, SizeOf(b), 0);

  nLocal := m+1;
  •••
  Montar a matriz A com os somatórios de x (xGrade).
  Montar o vetor b com os somatórios de y (yGrade).
  Resolver o sistema A.x=b.
  A solução x fornece os coeficientes do polinômio.
  No meu programa, montei a matriz A triangular superior e copiei os valores para a triangular inferior, já que A é simétrica.
  •••
  // Matriz simétrica - copiar triangular inferior da superior
  for i := 2 to nLocal do
    for j := 1 to i-1 do
      A[i, j] := A[j, i];

  // Solução de A.x=b

  // Solução por Gauss (triangular superior)
  // Triangularização
  •••
  Calcular a triangularização por Gaus ou efetuar decomposição L.U.
  •••
  // Retrosubstituição
  •••
  Resolver o sistema por retrosubstituição se usou triangularização de Gauss.
  Resolver o sistema por substituição e retrosubstituição se usou decomposição L.U.
  •••

  •••
  A solução x fornece os coeficientes do polinômio.
  Armazenar estes valores na grade dos coeficientes do polinômio correspondente.
  •••
  for i := 1 to nLocal do
    StringGrid2.Cells[i+2, m-1] := FloatToStr(x[i]);

  // Coeficiente de Determinação
  SomaY := 0; SomaY2 := 0; SomaE2 := 0;
  •••
  Calcular R2.
  Lembre-se de usar os valores xGrade e yGrade.
  Armazenar R2 na grade, na linha do polinômio correspondente.
  •••
  R2 := 1 - (n * SomaE2 / (n * SomaY2 - Sqr(SomaY)));
  StringGrid2.Cells[10, m-1] := FloatToStr(R2);
end;

```

```

procedure TForm1.Ajuste1;      // Ajuste Linear y = a + b.x
var
  •••
  Definir as variáveis locais necessárias.
  •••
begin
  •••
  Efetuar o ajuste, calculando a, b e R2.
  Armazenar estes valores na grade dos coeficientes da função correspondente.
  •••
  with StringGrid3 do
  begin
    Cells[1, 1] := FloatToStr(a);
    Cells[2, 1] := FloatToStr(b);
    Cells[3, 1] := FloatToStr(R2);
  end;
end;

procedure TForm1.Ajuste2;      // Ajuste Exponencial y = a.bx
var
  •••
  Definir as variáveis locais necessárias.
  •••
begin
  •••
  Efetuar o ajuste, calculando a, b e R2.
  Armazenar estes valores na grade dos coeficientes da função correspondente.
  •••
  with StringGrid3 do
  begin
    Cells[1, 2] := FloatToStr(a);
    Cells[2, 2] := FloatToStr(b);
    Cells[3, 2] := FloatToStr(R2);
  end;
end;

procedure TForm1.Ajuste3;      // Ajuste Exponencial y = a.xb
var
  •••
  Definir as variáveis locais necessárias.
  •••
begin
  •••
  Efetuar o ajuste, calculando a, b e R2.
  Armazenar estes valores na grade dos coeficientes da função correspondente.
  •••
  with StringGrid3 do
  begin
    Cells[1, 3] := FloatToStr(a);
    Cells[2, 3] := FloatToStr(b);
    Cells[3, 3] := FloatToStr(R2);
  end;
end;

procedure TForm1.Ajuste4;      // Ajuste Exponencial y = a.b^(e.x)
var
  •••
  Definir as variáveis locais necessárias.
  •••
begin
  SomaX := 0; SomaY := 0; SomaXY := 0; SomaX2 := 0; SomaY2 := 0;
  for i := 1 to n do
  begin
  •••
  Efetuar o ajuste, calculando a, b e R2.
  Armazenar estes valores na grade dos coeficientes da função correspondente.
  •••
  with StringGrid3 do
  begin
    Cells[1, 4] := FloatToStr(a);
    Cells[2, 4] := FloatToStr(b);
  end;
end;

```

```

    Cells[3, 4] := FloatToStr(R2);
end;
end;

procedure TForm1.Ajuste5; // Ajuste Exponencial y = a.e^(b.x)
var
  ...
  Definir as variáveis locais necessárias.
  ...
begin
  ...
  Efetuar o ajuste, calculando a, b e R2.
  Armazenar estes valores na grade dos coeficientes da função correspondente.
  ...
  with StringGrid3 do
begin
  Cells[1, 5] := FloatToStr(a);
  Cells[2, 5] := FloatToStr(b);
  Cells[3, 5] := FloatToStr(R2);
end;
end;

procedure TForm1.Ajuste6; // Ajuste Exponencial y = e^(a+b.x)
var
  ...
  Definir as variáveis locais necessárias.
  ...
begin
  ...
  Efetuar o ajuste, calculando a, b e R2.
  Armazenar estes valores na grade dos coeficientes da função correspondente.
  ...
  with StringGrid3 do
begin
  Cells[1, 6] := FloatToStr(a);
  Cells[2, 6] := FloatToStr(b);
  Cells[3, 6] := FloatToStr(R2);
end;
end;

procedure TForm1.Ajuste7; // Ajuste Hiperbólico y = 1/(a+b.x)
var
  ...
  Definir as variáveis locais necessárias.
  ...
begin
  ...
  Efetuar o ajuste, calculando a, b e R2.
  Armazenar estes valores na grade dos coeficientes da função correspondente.
  ...
  with StringGrid3 do
begin
  Cells[1, 7] := FloatToStr(a);
  Cells[2, 7] := FloatToStr(b);
  Cells[3, 7] := FloatToStr(R2);
end;
end;

procedure TForm1.Ajuste8; // Ajuste Hiperbólico y = x/(a+b.x)
var
  ...
  Definir as variáveis locais necessárias.
  ...
begin
  ...
  Efetuar o ajuste, calculando a, b e R2.
  Armazenar estes valores na grade dos coeficientes da função correspondente.
  ...
  with StringGrid3 do
begin
  Cells[1, 8] := FloatToStr(a);

```

```

Cells[2, 8] := FloattoStr(b);
Cells[3, 8] := FloattoStr(R2);
end;
end;

procedure TForm1.Ajuste9; // Ajuste Hiperbólico y = 1/(1+e^(a+b.x))
var
  •••
  Definir as variáveis locais necessárias.
  •••
begin
  •••
  Efetuar o ajuste, calculando a, b e R2.
  Armazenar estes valores na grade dos coeficientes da função correspondente.
  •••
  with StringGrid3 do
    begin
      Cells[1, 9] := FloattoStr(a);
      Cells[2, 9] := FloattoStr(b);
      Cells[3, 9] := FloattoStr(R2);
    end;
end;

procedure TForm1.Ajuste10; // Ajuste Exponencial y = 1+a.e^(b.x)
var
  •••
  Definir as variáveis locais necessárias.
  •••
begin
  •••
  Efetuar o ajuste, calculando a, b e R2.
  Armazenar estes valores na grade dos coeficientes da função correspondente.
  •••
  with StringGrid3 do
    begin
      Cells[1, 10] := FloattoStr(a);
      Cells[2, 10] := FloattoStr(b);
      Cells[3, 10] := FloattoStr(R2);
    end;
end;

procedure TForm1.Ajuste11; // Ajuste Exponencial y = a+b.ln(x)
var
  •••
  Definir as variáveis locais necessárias.
  •••
begin
  •••
  Efetuar o ajuste, calculando a, b e R2.
  Armazenar estes valores na grade dos coeficientes da função correspondente.
  •••
  with StringGrid3 do
    begin
      Cells[1, 11] := FloattoStr(a);
      Cells[2, 11] := FloattoStr(b);
      Cells[3, 11] := FloattoStr(R2);
    end;
end;

procedure TForm1.Ajuste12; // Ajuste Hiperbólico y = a+b/x
var
  •••
  Definir as variáveis locais necessárias.
  •••
begin
  •••
  Efetuar o ajuste, calculando a, b e R2.
  Armazenar estes valores na grade dos coeficientes da função correspondente.
  •••
  with StringGrid3 do
    begin

```

```

Cells[1, 12] := FloattoStr(a);
Cells[2, 12] := FloattoStr(b);
Cells[3, 12] := FloattoStr(R2);
end;
end;

procedure TForm1.Ajuste13; // Ajuste Hiperbólico y = a/(b+x)
var
  ...
  Definir as variáveis locais necessárias.
  ...
begin
  ...
  Efetuar o ajuste, calculando a, b e R2.
  Armazenar estes valores na grade dos coeficientes da função correspondente.
  ...
  with StringGrid3 do
    begin
      Cells[1, 13] := FloattoStr(a);
      Cells[2, 13] := FloattoStr(b);
      Cells[3, 13] := FloattoStr(R2);
    end;
end;

procedure TForm1.Ajuste14; // Ajuste Hiperbólico y = a.b/(b+x)
var
  ...
  Definir as variáveis locais necessárias.
  ...
begin
  ...
  Efetuar o ajuste, calculando a, b e R2.
  Armazenar estes valores na grade dos coeficientes da função correspondente.
  ...
  with StringGrid3 do
    begin
      Cells[1, 14] := FloattoStr(a);
      Cells[2, 14] := FloattoStr(b);
      Cells[3, 14] := FloattoStr(R2);
    end;
end;

procedure TForm1.Ajuste15; // Ajuste Hiperbólico y = a.x/(b+x)
var
  ...
  Definir as variáveis locais necessárias.
  ...
begin
  ...
  Efetuar o ajuste, calculando a, b e R2.
  Armazenar estes valores na grade dos coeficientes da função correspondente.
  ...
  with StringGrid3 do
    begin
      Cells[1, 15] := FloattoStr(a);
      Cells[2, 15] := FloattoStr(b);
      Cells[3, 15] := FloattoStr(R2);
    end;
end;
end.

```

Agora só falta você fazer a sua parte que é programar os algoritmos para solução dos problemas de ajuste de curvas. Só isto. Divirta-se.

## 12- Integrais

No início do Capítulo 2, são apresentadas as principais noções relativas aos Métodos Numéricos Computacionais.

Uma delas é que não se deseja a solução analítica, para encontrar a solução numérica, aplicando-se as condições de contorno numéricas à solução analítica.

Outra é que os métodos são, evidentemente, compatíveis com os métodos matemáticos analíticos, ou seja, utilizam os mesmos conceitos, mas isto é feito para encontrar apenas a solução numérica.

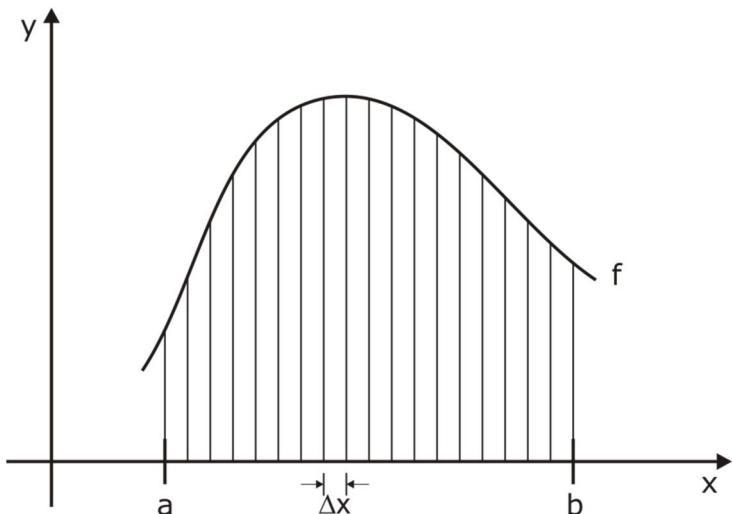
Além disto, em certas situações, utilizar serra, água e bêquer é mais sensato. :-)

Refrescada sua memória, é reproduzido, a seguir, o trecho de motivação ou desmotivação do Capítulo 2, sobre cálculo de integrais.

Lembra-se daquela integral encapetada que você não resolveu nem por partes nem sem partes? E lembra-se, também, da definição de integral, não é mesmo?

Então, simplesmente utilize a definição e, onde está escrito  $n \rightarrow \infty$  ou  $\delta x \rightarrow 0$ , escreva  $\delta x = \Delta x$ , com  $\Delta x$  pequeno. A soma das faixas infinitesimais será substituída pela soma de faixas de largura  $\Delta x$  e o valor da integral será aproximado, onde a precisão será proporcional ao tamanho de  $\Delta x$ .

Graficamente, a brincadeira fica como segue.



Pode-se calcular o valor numérico da integral como  $\int_a^b f(x) dx \approx \sum_{i=1}^n f(\bar{x}) \cdot \Delta x$ .

Da conhecida definição de integral, sabe-se que  $\int_a^b f(x) dx = \lim_{n \rightarrow \infty} \sum_{i=1}^n f(\bar{x}) \cdot \Delta x$ , onde  $\bar{x}$  é o valor médio de  $f(x)$  para cada intervalo determinado por  $\Delta x$ .

Pode ser que você prefira a definição  $\int_a^b f(x) dx = \lim_{\delta x \rightarrow 0} \sum_{i=1}^n f(\bar{x}) \cdot \delta x$ , que dá tudo na mesma coisa.

O que interessa é que  $\delta x$  é infinitesimal e  $\Delta x$  não é.

Com  $\Delta x$  pequeno, porém infinitamente maior que  $\delta x \rightarrow 0$ , haverá um erro que poderá ser controlado fazendo  $\Delta x$  tão pequeno quanto se queira ou se possa fazer.

Partindo do conceito de que  $\int_a^b f(x) dx \approx \sum_{i=1}^n f(\bar{x}) \cdot \Delta x$  e sabendo que  $\Delta x \gg \delta x \rightarrow 0$ , o que se faz é somar as áreas dos retângulos que se pode formar em cada faixa da figura. Segue o desenvolvimento deste conceito.

Para não haver confusão de notação, fica combinado que o número de faixas é  $n$  e, portanto, o número de pontos é  $n+1$ . Para facilitar a notação, considera-se que os pontos são chamados de  $x_0$  até  $x_n$ . Além disto,  $\Delta x$  será chamado de  $h$ , velho conhecido passo para distâncias pequenas. Então, deseja-se calcular  $\int_a^b f(x) \cdot dx \approx \sum_{i=0}^n f(x_i) \cdot h$ , onde a largura da faixa é  $h = (b-a)/n$ .

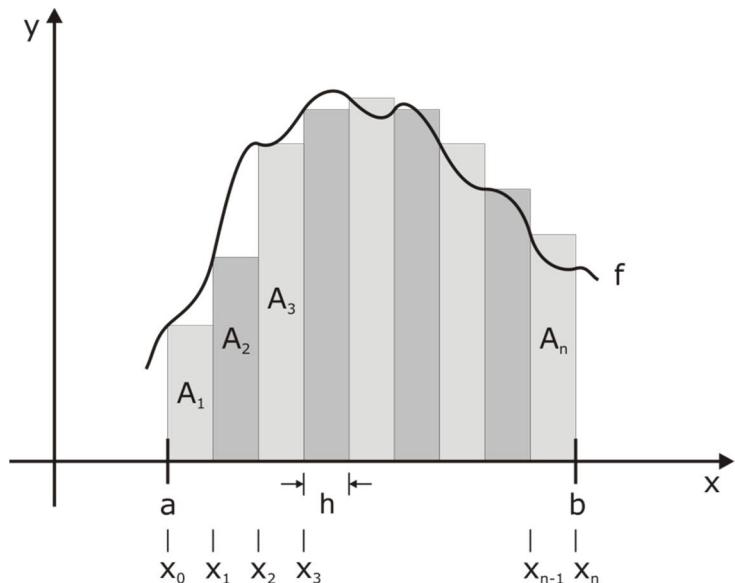
Para verificar o resultado numérico dos métodos que seguem, é utilizado o exemplo como segue.

Considere  $f(x) = x^2$ ,  $a = 2$ ,  $b = 5$ .

$$\text{Analiticamente, } \int_2^5 x^2 \cdot dx = \frac{x^3}{3} \Big|_2^5 = \frac{125 - 8}{3} = \frac{117}{3} = 39.$$

## Regra dos Retângulos à Esquerda

Considere a figura que segue e pense no óbvio ou, mais ainda, pense no óbvio ululante.



O valor da integral será a soma das áreas dos retângulos, ou seja,  $A = A_1 + A_2 + A_3 + \dots + A_n$ .

Neste caso,  $\int_a^b f(x) \cdot dx \approx \sum_{i=0}^{n-1} f(x_i) \cdot h$ , com  $h = (b-a)/n$ .

Nas regiões onde  $f(x)$  é convexa, a área de cada faixa é menor que a área verdadeira e, nas regiões onde  $f(x)$  é côncava, a área de cada faixa é maior que a área verdadeira.

Calculando numericamente, com a equação precedente, com  $n = 8$ , tem-se o que segue.

$f(x) = x^2$ ,  $a = 2$ ,  $b = 5$ . Se  $n = 8$ , então,  $h = (5-2)/8 = 0,375$ .

Os pontos para cálculo dos retângulos à esquerda são  $x_0 = 2, \dots, x_7 = 4,625$ .

O valor da integral é a soma das áreas dos retângulos, ou seja,  $A = A_1 + A_2 + A_3 + \dots + A_8$ .

i	$x_i$	$f(x_i)$	$f(x_i) \cdot h = A_i$
0	2	4	1,5
1	2,375	5,640625	2,115234375
2	2,75	7,5625	2,8359375
3	3,125	9,765625	3,662109375
4	3,5	12,25	4,59375
5	3,875	15,015625	5,630859375
6	4,25	18,0625	6,7734375
7	4,625	21,390625	8,021484375
		Soma = A	35,1328125

O valor exato é 39 e foi encontrado 35,1328125. O erro parece grotesco, não é? Mesmo assim, segue uma ideia de um algoritmo. Um ensaio de um algoritmo pode ser como segue.

Erros acumulados em x e Área

Dados  $f$ ,  $a$ ,  $b$ ,  $n$

$$h = (b-a)/n$$

$$x = a$$

$$\text{Área} = 0$$

Para  $i = 0, \dots, n-1$  faça  
 Área = Área +  $f(x) \cdot h$

$$x = x + h$$

$$\text{Integral} = \text{Área}$$

Sem arredondamentos sucessivos

Dados  $f$ ,  $a$ ,  $b$ ,  $n$

$$h = (b-a)/n$$

{ x calculado no laço }

$$\text{Soma} = 0$$

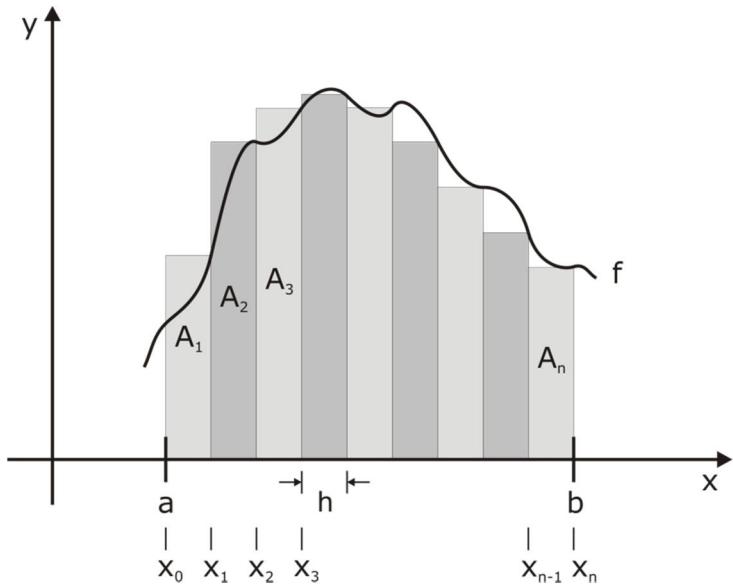
Para  $i = 0, \dots, n-1$  faça  
 x = a + i \* h

$$\text{Soma} = \text{Soma} + f(x)$$

$$\text{Integral} = \text{Soma} \cdot h$$

## Regra dos Retângulos à Direita

Considere a figura que segue e pense no óbvio ou, mais ainda, pense no óbvio ululante. Déjà vu.



O valor da integral será a soma das áreas dos retângulos, ou seja,  $A = A_1 + A_2 + A_3 + \dots + A_n$ .

Neste caso,  $\int_a^b f(x).dx \approx \sum_{i=1}^n f(x_i).h$ , com  $h = (b-a)/n$ .

Nas regiões onde  $f(x)$  é convexa, a área de cada faixa é maior que a área verdadeira e, nas regiões onde  $f(x)$  é côncava, a área de cada faixa é menor que a área verdadeira.

Calculando numericamente, com a equação precedente, com  $n = 8$ , tem-se o que segue.

$$f(x) = x^2, a = 2, b = 5. \text{ Se } n = 8, \text{ então, } h = (5-2)/8 = 0,375.$$

Os pontos para cálculo dos retângulos à esquerda são  $x_1 = 2,375, \dots, x_8 = 5$ .

O valor da integral é a soma das áreas dos retângulos, ou seja,  $A = A_1 + A_2 + A_3 + \dots + A_8$ .

i	$x_i$	$f(x_i)$	$f(x_i).h = A_i$
1	2,375	5,640625	2,115234375
2	2,75	7,5625	2,8359375
3	3,125	9,765625	3,662109375
4	3,5	12,25	4,59375
5	3,875	15,015625	5,630859375
6	4,25	18,0625	6,7734375
7	4,625	21,390625	8,021484375
8	5	25	9,375
	Soma		43,0078125

O valor exato é 39 e foi encontrado 43,0078125. O erro parece grotesco, não é? Mesmo assim, segue uma ideia de um algoritmo. Um ensaio de um algoritmo pode ser como segue.

Erros acumulados em x e Área

Dados  $f$ ,  $a$ ,  $b$ ,  $n$

$$h = (b-a)/n$$

$$x = a$$

$$\text{Área} = 0$$

Para  $i = 1, \dots, n$  faça

$$\text{Área} = \text{Área} + f(x).h$$

$$x = x + h$$

$$\text{Integral} = \text{Área}$$

Sem arredondamentos sucessivos

Dados  $f$ ,  $a$ ,  $b$ ,  $n$

$$h = (b-a)/n$$

{ x calculado no laço }

$$\text{Soma} = 0$$

Para  $i = 1, \dots, n$  faça

$$x = a + i * h$$

$$\text{Soma} = \text{Soma} + f(x)$$

$$\text{Integral} = \text{Soma}.h$$

Nenhum desses métodos anteriores tem boa precisão, por motivos óbvios. O que fazer?

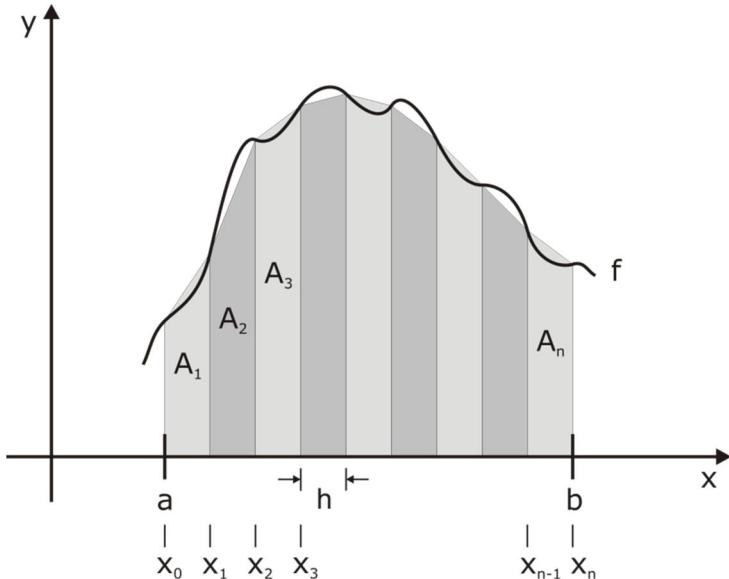
Que tal a média destes dois casos? Boa ideia. A média da área de cada retângulo (à esquerda e à direita) será a área de um trapézio. Super ideia. Alguém já pensou nisto lá pelos idos de 1700. E nem existia computador naquela época. Ou existia?

Naquela época não existia televisão, computador, celular, Facebook, Twitter e outros e, então, o passatempo era inventar moda, principalmente com a tal da matemática. Era um tempo muito bom. Ou não. Depende de sua necessidade de ficar cutucando as teclas de seu celular durante as aulas de MNC. Depois conversaremos sobre isto.

## Regra dos Trapézios

O esperado é que com a Regra dos Trapézios, encontre-se o valor médio entre a Regra do Retângulo à Esquerda e a Regra do Retângulo à Direita, ou seja,  $I = (35,1328125+43,0078125)/2 = 39,0703125$ .

Considere a figura que segue e pense no óbvio ou, mais ainda, pense no óbvio ululante. Déjà vu<sup>++</sup>.



O valor da integral será a soma das áreas dos retângulos, ou seja,  $A = A_1 + A_2 + A_3 + \dots + A_n$ .

Neste caso,  $\int_a^b f(x) dx \approx \sum_{i=0}^{n-1} \frac{f(x_i) + f(x_{i+1})}{2} h$ , com  $h = (b-a)/n$ .

Nas regiões onde  $f(x)$  é convexa, a área de cada faixa é maior que a área verdadeira e, nas regiões onde  $f(x)$  é côncava, a área de cada faixa é menor que a área verdadeira.

Contudo, esta diferença é bem menor do que nas Regras dos Retângulos.

Calculando numericamente, com a equação precedente, com  $n = 8$ , tem-se o que segue.

$$f(x) = x^2, a = 2, b = 5. \text{ Se } n = 8, \text{ então, } h = (5-2)/8 = 0,375.$$

Os pontos para cálculo dos trapézios são  $x_0 = 2, \dots, x_8 = 5$ .

O valor da integral é a soma das áreas dos trapézios, ou seja,  $A = A_1 + A_2 + A_3 + \dots + A_8$ .

i	$x_i$	$x_{i+1}$	$f(x_i) + f(x_{i+1})$	$(f(x_i) + f(x_{i+1}))/2 \cdot h = A_i$
0	2	2,375	9,640625	1,8076171875
1	2,375	2,75	13,203125	2,4755859375
2	2,75	3,125	17,328125	3,2490234375
3	3,125	3,5	22,015625	4,1279296875
4	3,5	3,875	27,265625	5,1123046875
5	3,875	4,25	33,078125	6,2021484375
6	4,25	4,625	39,453125	7,3974609375
7	4,625	5	46,390625	8,6982421875
		Soma	39,0703125	

O valor exato é 39 e foi encontrado 39,0703125. O erro é bem menor que o erro encontrado nas Regras dos Retângulos, não é?

Um ensaio de um algoritmo pode ser como segue.

Dados  $f$ ,  $a$ ,  $b$ ,  $n$   
 $h = (b-a)/n$   
Área = 0  
Para  $i = 0, \dots, n-1$  faça  
 $x = a + i \cdot h$   
Área = Área +  $(f(x) + f(x+h))/2 \cdot h$   
Integral = Área

Neste algoritmo, como nos anteriores, para evitar erros sucessivos de arredondamento, deve-se efetuar a divisão por 2 e o produto por  $h$ , fora do laço da soma das parcelas da área.

Além disto, verifica-se que para  $i = 0$ , calcula-se  $(f(x_0)+f(x_1))$ , para  $i = 1$ , calcula-se  $(f(x_1)+f(x_2))$ , ..., para  $i = n-1$ , calcula-se  $(f(x_{n-1})+f(x_n))$ . Portanto, a área é  $[(f(x_0)+f(x_1))/2 + (f(x_1)+f(x_2)+\dots+f(x_{n-1})) \cdot h]$ .

Assim, um algoritmo um pouco mais decente elaborado pode ser como segue.

Dados f, a, b, n  
 $h = (b-a)/n$   
Área =  $(f(a)+f(b))/2$   
Para i = 1, ..., n-1 faça  
 $x = a + i * h$   
Área = Área + f(x)  
Integral = Área.h

Agora chegou a hora de você, amado leitor, fazer os programas simples para estes três casos e reservá-los para utilizar no belíssimo programa que você fará ao final do capítulo.

E, é claro que você vai utilizar um interpretador de funções, para que seus programas executem qualquer função solicitada pelo usuário.

Eu gostei da ideia sobre os programas. E você? Já que todos gostaram, seguem as regras de Simpson.

## Regras de Simpson

Antes de iniciar com as Regras de Simpson, é necessária uma Seção Fofoca, pois há controvérsias sobre sua criação. Apenas um pouco de fofoca. Depois as coisas seguirão **quase** tão normais, como sempre.

## Seção Fofoca



Você sabia que **quase** todos os livros de Métodos Numéricos afirmam que Simpson inventou o que se chama de Simpson's Rule, mas que estas regras já existiam há mais de 100 anos, inventadas por Kepler e se chamavam Keplersche Fassregel?

É verdade?

É o que estão dizendo, mas não diga que fui eu.



**Thomas Simpson**

Leicestershire  
20 de agosto de 1710  
Leicestershire  
14 de maio de 1761

Matemático e inventor britânico.

**Johannes Kepler**

Weil der Stadt  
27 de dezembro de 1571  
Ratisbona  
15 de novembro de 1630  
Astrônomo e matemático alemão.

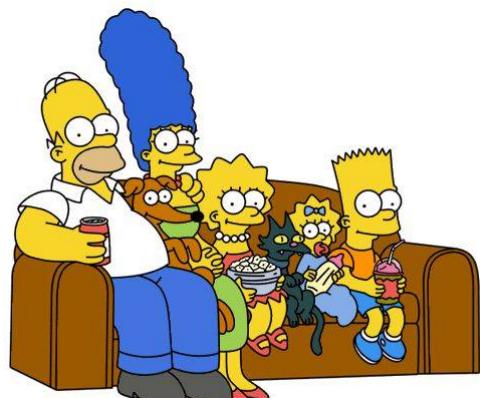


Pode ser uma grande polêmica, mas imagine se o Homer Simpson formulasse regras para calcular Integrais.

Acho que seriam integrais duplas ou, até triplas, para calcular o volume de latas de cerveja.

As regras que seguem não tem nada a ver com o povo de Springfield, mas são divertidas.

Então, divirta-se.



## Regras de Simpson - Após Seção Fofoca

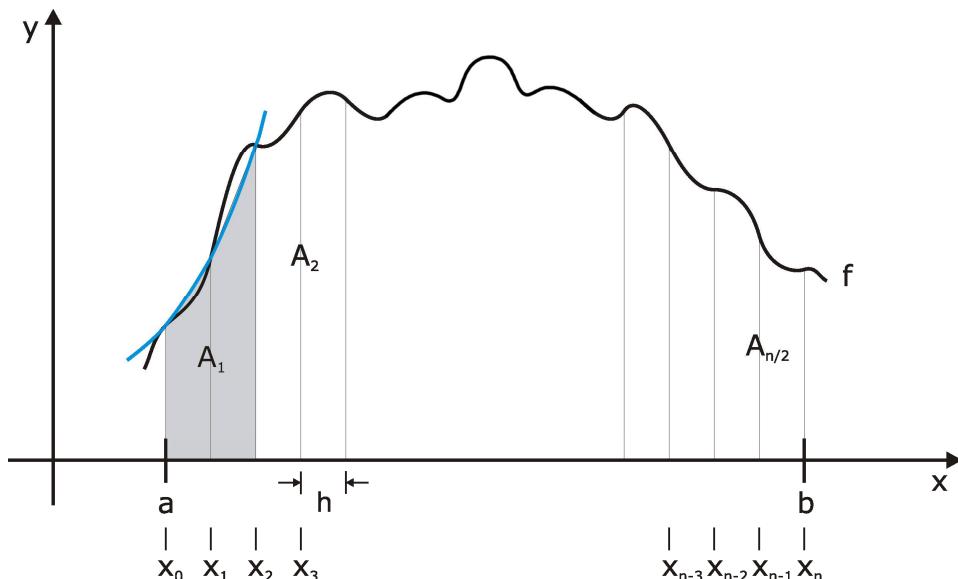
A ideia é interpolar uma função utilizando alguns pontos consecutivos da função e utilizar o polinômio interpolador para calcular a integral nas faixas contidas entre estes pontos.

Isto feito, estende-se a formulação para todas as faixas do intervalo  $[a, b]$ . Simples assim.

### Regra 1/3 de Simpson

Considere uma função contínua no intervalo  $[a, b]$ .

Determina-se uma função de segundo grau que passa pelos pontos  $(x_0, y_0)$ ,  $(x_1, y_1)$  e  $(x_2, y_2)$  com a formulação do polinômio de segundo grau de Newton-Gregory, que é uma parábola.



Para ler e pensar, antes de continuar.

As áreas são calculadas a cada três pontos, ou seja, para duas faixas.

Então, para  $n$  pontos, tem-se  $n/2$  áreas calculadas.

Para que o método funcione para todo o intervalo  $[a, b]$ , o número de faixas deve ser par, ou seja,  $n$  deve ser par.

$$\text{Então, com } n = 2, \text{ tem-se } \int_a^b f(x) dx \approx \int_a^b p_2(x) dx = \int_a^b \left( y_0 + s \Delta y_0 + \frac{s(s-1)}{2!} \Delta^2 y_0 \right) dx.$$

$$\text{Como } s = \frac{x-x_0}{h}, \quad ds = \frac{dx}{h} \quad \text{e, então, } dx = h \cdot ds.$$

Para efetuar a integração entre  $x_0$  e  $x_2$ ,  $a = x_0$  e  $b = x_2$ .

$$\text{Então, para } x = a, \quad s = \frac{a-a}{h} = 0 \quad \text{e, para } x = b, \quad s = \frac{b-a}{h} = \frac{2h}{h} = 2.$$

$$\begin{aligned} \text{Desta forma, } \int_a^b f(x) dx &\approx \int_a^b \left( y_0 + s \Delta y_0 + \frac{s(s-1)}{2!} \Delta^2 y_0 \right) dx = \int_0^2 \left( y_0 + s \Delta y_0 + \frac{s^2 - s}{2} \Delta^2 y_0 \right) h \cdot ds = \\ &= h \left[ y_0 s + \Delta y_0 \cdot \frac{s^2}{2} + \frac{\Delta^2 y_0}{2} \cdot \left( \frac{s^3}{3} - \frac{s^2}{2} \right) \right] \Big|_0^2 = h \left[ y_0 \cdot 2 + \Delta y_0 \cdot \frac{4}{2} + \frac{\Delta^2 y_0}{2} \left( \frac{8}{3} - \frac{4}{2} \right) \right] - h \cdot 0 = h \left[ 2y_0 + 2 \cdot \Delta y_0 + \frac{1}{3} \cdot \Delta^2 y_0 \right]. \end{aligned}$$

Como  $\Delta y_0 = y_1 - y_0$  e  $\Delta^2 y_0 = \Delta y_1 - \Delta y_0 = (y_2 - y_1) - (y_1 - y_0) = y_2 - 2 \cdot y_1 + y_0$ , tem-se

$$\int_a^b f(x) dx = h \left[ 2.y_0 + 2.\Delta y_0 + \frac{1}{3} \cdot \Delta^2 y_0 \right] = h \left[ 2.y_0 + 2 \cdot (y_1 - y_0) + \frac{1}{3} \cdot (y_2 - 2 \cdot y_1 + y_0) \right] = h \left[ \frac{1}{3} \cdot y_0 + \frac{4}{3} \cdot y_1 + \frac{1}{3} \cdot y_2 \right].$$

Portanto,  $\int_a^b f(x) dx = \frac{h}{3} \cdot [y_0 + 4 \cdot y_1 + y_2]$ , que vale para os três primeiros pontos, que equivale às duas primeiras faixas, ou seja, para a área  $A_1$ .

Estendendo para todas as faixas, tem-se  $\int_a^b f(x) dx \approx A_1 + A_2 + \dots + A_{n/2}$ .

Então,  $\int_a^b f(x) dx \approx \frac{h}{3} \cdot [y_0 + 4 \cdot y_1 + y_2] + \frac{h}{3} \cdot [y_2 + 4 \cdot y_3 + y_4] + \dots + \frac{h}{3} \cdot [y_{n-2} + 4 \cdot y_{n-1} + y_n]$ , que pode ser escrita como  $\int_a^b f(x) dx \approx \frac{h}{3} \cdot [y_0 + 4 \cdot (y_1 + y_3 + \dots + y_{n-1}) + 2 \cdot (y_2 + y_4 + \dots + y_{n-2}) + y_n]$ .

Esta regra vale para  $n$  par. Se  $n$  é ímpar, pode-se calcular a integral de  $x_0 = a$  até  $x_{n-1}$  com a regra e, para a última faixa, calcular a integral de  $x_{n-1}$  até  $x_n = b$  com a regra do trapézio. A integral será a soma das duas anteriores.

Calculando numericamente, com a equação precedente, com  $n = 8$ , tem-se o que segue.

$$f(x) = x^2, a = 2, b = 5. \text{ Se } n = 8, \text{ então, } h = (5-2)/8 = 0,375.$$

O valor da integral é a soma ponderada das parcelas, ou seja,  $A = h/3 \cdot [f_0 + 4 \cdot (f_1 + f_3 + f_5 + f_7) + 2 \cdot (f_2 + f_4 + f_6) + f_8]$ .

i	$x_i$	$f(x_i)$	m	$m \cdot f(x_i)$
0	2	4	1	4
1	2,375	5,640625	4	22,5625
2	2,75	7,5625	2	15,125
3	3,125	9,765625	4	39,0625
4	3,5	12,25	2	24,5
5	3,875	15,015625	4	60,0625
6	4,25	18,0625	2	36,125
7	4,625	21,390625	4	85,5625
8	5	25	1	25
		Soma		312
		$h/3 \cdot \text{Soma}$		39,0104

O valor exato é 39 e foi encontrado 39,0104. O erro é bem menor que o erro encontrado nas Regras dos Retângulos e dos Trapézios, não é? Um ensaio de um algoritmo pode ser como segue.

```
Dados f, a, b, n
h = (b-a)/n // n deve ser par
A = f(a)+f(b)
Para i = 1, ..., n/2 faça
  x = a + (2.i-1).h // índices ímpares
  A = A + 4.f(x)
Para i = 1, ..., n/2-1 faça
  x = a + (2.i).h // índices pares
  A = A + 2.f(x)
Integral = h/3.A
```

Reescrevendo o algoritmo de forma um pouco menos infantil.

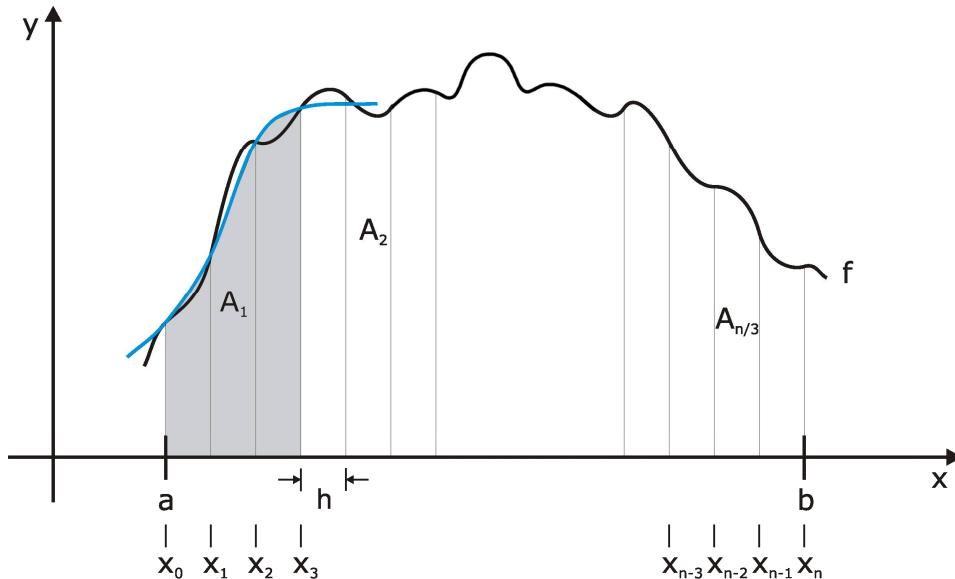
```
Dados f, a, b, n
h = (b-a)/n // n deve ser par
A4 = 0; A2 = 0
Para i = 1, ..., n/2 faça
  x = a + (2.i-1).h // índices ímpares
  A4 = A4 + f(x)
Para i = 1, ..., n/2-1 faça
  x = a + (2.i).h // índices pares
  A2 = A2 + f(x)
Integral = h. (f(a)+f(b)+4. A4+2. A2)/3
```

O velho lembrete: somas e subtrações, depois multiplicações e finalmente divisões. Reduz acumulo de erros.

## Regra 3/8 de Simpson

Considere uma função contínua no intervalo  $[a, b]$ .

Determina-se uma função de terceiro grau que passa pelos pontos  $(x_0, y_0), (x_1, y_1), (x_2, y_2)$  e  $(x_3, y_3)$  com a formulação do polinômio de terceiro grau de Newton-Gregory.



Para ler e pensar, antes de continuar.

As áreas são calculadas a cada quatro pontos, ou seja, para três faixas.

Então, para  $n$  pontos, tem-se  $n/3$  áreas calculadas.

Para que o método funcione para todo o intervalo  $[a, b]$ , o número de faixas deve ser múltiplo de três, ou seja,  $n$  deve ser múltiplo de três.

$$\text{Então, com } n = 3, \text{ tem-se } \int_a^b f(x) dx \approx \int_a^b p_3(x) dx = \int_a^b \left( y_0 + s \Delta y_0 + \frac{s(s-1)}{2!} \Delta^2 y_0 + \frac{s(s-1)(s-2)}{3!} \Delta^3 y_0 \right) dx.$$

$$\text{Como } s = \frac{x - x_0}{h}, \quad ds = \frac{dx}{h} \quad \text{e, então, } dx = h \cdot ds.$$

Para efetuar a integração entre  $x_0$  e  $x_3$ ,  $a = x_0$  e  $b = x_3$ .

$$\text{Então, para } x = a, s = \frac{a - a}{h} = 0 \quad \text{e, para } x = b, s = \frac{b - a}{h} = \frac{3 \cdot h}{h} = 3.$$

Desta forma, tem-se

$$\begin{aligned} \int_a^b f(x) dx &\approx \int_a^b \left( y_0 + s \Delta y_0 + \frac{s(s-1)}{2!} \Delta^2 y_0 + \frac{s(s-1)(s-2)}{3!} \Delta^3 y_0 \right) dx = \\ &= \int_0^3 \left( y_0 + s \Delta y_0 + \frac{s^2 - s}{2} \Delta^2 y_0 + \frac{s^3 - 3s^2 + 2s}{6} \Delta^3 y_0 \right) h \cdot ds = \\ &= h \cdot \left[ y_0 \cdot s + \Delta y_0 \cdot \frac{s^2}{2} + \frac{\Delta^2 y_0}{2} \cdot \left( \frac{s^3}{3} - \frac{s^2}{2} \right) + \frac{\Delta^3 y_0}{6} \cdot \left( \frac{s^4}{4} - s^3 + s^2 \right) \right]_0^3 = \\ &= h \cdot \left[ y_0 \cdot 3 + \Delta y_0 \cdot \frac{9}{2} + \frac{\Delta^2 y_0}{2} \left( \frac{27}{3} - \frac{9}{2} \right) + \frac{\Delta^3 y_0}{6} \left( \frac{81}{4} - 27 + 9 \right) \right] - h \cdot 0 = \\ &= h \cdot \left[ 3 \cdot y_0 + \frac{9}{2} \cdot \Delta y_0 + \frac{9}{4} \cdot \Delta^2 y_0 + \frac{3}{8} \cdot \Delta^3 y_0 \right]. \end{aligned}$$

Como  $\Delta y_0 = y_1 - y_0$ ,

$$\Delta^2 y_0 = \Delta y_1 - \Delta y_0 = (y_2 - y_1) - (y_1 - y_0) = y_2 - 2 \cdot y_1 + y_0 \quad \text{e}$$

$$\Delta^3 y_0 = \Delta^2 y_1 - \Delta^2 y_0 = (y_3 - 2 \cdot y_2 + y_1) - (y_2 - 2 \cdot y_1 + y_0) = y_3 - 3 \cdot y_2 + 3 \cdot y_1 - y_0, \quad \text{tem-se}$$

$$\int_a^b f(x) dx \approx 3h \left[ y_0 + \frac{3}{2} \cdot \Delta y_0 + \frac{3}{4} \cdot \Delta^2 y_0 + \frac{1}{8} \cdot \Delta^3 y_0 \right] =$$

$$3h \left[ y_0 + \frac{3}{2} \cdot (y_1 - y_0) + \frac{3}{4} \cdot (y_2 - 2y_1 + y_0) + \frac{1}{8} \cdot (y_3 - 3y_2 + 3y_1 - y_0) \right] =$$

$$3h \left[ \frac{1}{8} \cdot y_0 + \frac{3}{8} \cdot y_1 + \frac{3}{8} \cdot y_2 + \frac{1}{8} \cdot y_3 \right] =$$

$$\frac{3}{8} \cdot h \cdot [y_0 + 3y_1 + 3y_2 + y_3]$$

Portanto,  $\int_a^b f(x) dx \approx \frac{3}{8} h [y_0 + 3y_1 + 3y_2 + y_3]$ , que vale para os quatro primeiros pontos, que equivale às três primeiras faixas, ou seja, para a área  $A_1$ .

Estendendo para todas as faixas, tem-se  $\int_a^b f(x) dx \approx A_1 + A_2 + \dots + A_{n/3}$ .

Então,

$$\int_a^b f(x) dx \approx \frac{3}{8} h [y_0 + 3y_1 + 3y_2 + y_3] + \frac{3}{8} h [y_3 + 3y_4 + 3y_5 + y_6] + \dots + \frac{3}{8} h [y_{n-3} + 3y_{n-2} + 3y_{n-1} + y_n],$$

que pode ser escrita como

$$\int_a^b f(x) dx \approx \frac{3}{8} h \left[ y_0 + 3 \cdot \underbrace{(y_1 + y_2 + y_4 + y_5 + \dots + y_{n-2} + y_{n-1})}_{\text{índices } i \text{ não múltiplos de 3}} + 2 \cdot \underbrace{(y_3 + y_6 + y_9 + \dots + y_{n-3})}_{\text{índices } i \text{ múltiplos de 3}} + y_n \right].$$

Esta regra vale para  $n$  múltiplo de 3. Se  $n$  não é múltiplo de 3, pode-se calcular a integral de  $x_0 = a$  até um número máximo de faixas múltiplas de 3 e, o restante, com 1/3 de Simpson, se restante é 2 ou Trapézio, se o restante é 1. A integral será a soma das duas anteriores.

Calculando numericamente, com a equação precedente, com  $n = 9$ , tem-se o que segue.

$$f(x) = x^2, a = 2, b = 5. \text{ Se } n = 9, \text{ então, } h = (5-2)/9 = 0,333\dots \text{ ou, } 1/3.$$

O valor da integral é a soma ponderada das parcelas, ou seja,  $A = 3.h/8.[f_0 + 3.(f_1+f_2+f_4+f_5+f_7+f_8) + 2.(f_3+f_6) + f_9]$ .

i	$x_i$	$f(x_i)$	m	$m.f(x_i)$
0	2	4	1	4
1	2,333333	5,444444	3	16,33333
2	2,666667	7,111111	3	21,33333
3	3	9	2	18
4	3,333333	11,111111	3	33,33333
5	3,666667	13,444444	3	40,33333
6	4	16	2	32
7	4,333333	18,77778	3	56,33333
8	4,666667	21,77778	3	65,33333
9	5	25	1	25
		Soma		312
		3.h/8.Soma		39

O valor exato é 39 e foi encontrado 39. Ficou razoável. :-)

Um ensaio de um algoritmo pode ser como segue.

```
Dados f, a, b, n
h = (b-a)/n // n deve ser múltiplo de 3
A = f(a)+f(b)
// índices não múltiplos de 3
Para i = 1, ..., n/3 faça
  x = a + (3.i-2).h;   A = A + 3.f(x)
  x = a + (3.i-1).h;   A = A + 3.f(x)
// índices múltiplos de 3
Para i = 1, ..., n/3-1 faça
  x = a + (3.i).h
  A = A + 2.f(x)
Integral = 3.h/8.A
```

Reescrevendo o algoritmo de forma um pouco menos infantil.

Certamente você sabe fazer isto, mas Patricinha diz que fez e pediu para publicar o que segue.

```

Dados f, a, b, n
h = (b-a)/n // n deve ser múltiplo de 3
A3 = 0; A2 = 0
Para i = 1, ..., n/3 faça // índices não múltiplos de 3
  x = a + (3.i-2).h
  A3 = A3 + f(x)
  x = a + (3.i-1).h
  A3 = A3 + f(x)
Para i = 1, ..., n/3-1 faça // índices não múltiplos de 3
  x = a + (3.i).h
  A2 = A2 + f(x)
Integral = 3.h. (f(a)+f(b)+3.A3+2.A2)/8

```

O velho lembrete: somas e subtrações, depois multiplicações e finalmente divisões. Reduz acumulo de erros. A Patricinha aprendeu isto.

## Quadratura de Gauss

Lá vem o Gauß novamente, que também é conhecido, neste nosso país tupiniquim, como Gauss.

Quadratura? O Capítulo é sobre Integrais. Que bicho é este? No dicionário, encontra-se o que segue.

Qua·dra·tu·ra  
substantivo feminino

1. Redução de qualquer figura geométrica a um quadrado equivalente.
2. [Astronomia] Situação respectiva de dois astros afastados um do outro 90°.
3. [Belas-Artes] Pintura a fresco.
4. Pintura de ornatos arquitetônicos.

Quadratura de uma curva

- Expressão analítica da área limitada por essa curva.

Quadratura do círculo

- Redução de um círculo a um quadrado equivalente (problema até hoje sem solução).
- [Figurado] Coisa impossível ou irrealizável. = UTOPIA

Então, todas as regras que somam as áreas de pequenas faixas, sejam retângulos ou trapézios, são quadraturas?

Sim.

E não seria mais adequado chamar a Quadratura de Gauss por Integral de Gauss?

Não.

Integral de Gauss ou Integral Gaussiana ou Integral de Euler-Poisson são nomes reservados para o problema de cálculo de

$\int_{-\infty}^{\infty} e^{-x^2} dx = \sqrt{\pi}$ , que é muito utilizada em física e estatística, pois a distribuição normal descreve diversos fenômenos de interesse nestas áreas.



Agora entendi. E se for Utopia?

A Fórmula de Gauss, para calcular Integrais Numéricas, fornece um resultado mais preciso do que as Regras de Trapézios e Regras de Simpson, para a mesma quantidade de pontos no intervalo  $[a, b]$ . Contudo, os pontos não são escolhidos por quem utiliza o método, mas sim, pelo próprio método. Além disto, oferece valores exatos para a integral de polinômios de grau  $2n-1$ , onde  $n$  é o número de pontos.

Considere  $I = \int_a^b f(x) dx \approx \sum_{i=0}^n w_i f(x_i)$ , onde  $w_0, w_1, \dots, w_n$  são os coeficientes ou pesos<sup>18</sup> e  $x_0, x_1, \dots, x_n$  são os pontos ou nós no intervalo  $[a, b]$ , com  $x_0 = a$  e  $x_n = b$ .

Os pesos e os pontos são determinados utilizando-se polinômios ortogonais e os limites de integração é função de cada tipo de polinômio, como seguem.

<sup>18</sup> Os pesos, são usualmente chamados de w (weight). Mesmo que você fale deletar, por favor, não fale weightar.

Polinômio	w	a	b
Legendre	1	-1	1
Chebyshev	$1/\sqrt{1-x^2}$	-1	1
Laguerre	$e^{-x}$	0	$\infty$
Hermite	$e^{-x^2}$	$-\infty$	$\infty$

Para o polinômio de Legendre, o Método é conhecido como Gauss-Legendre, com  $w = 1$ ,  $a = -1$  e  $b = 1$ .

Como os limites de integração são  $a = -1$  e  $b = 1$ , é necessária uma transformação dos valores  $a$  e  $b$  de integração originais, para estes novos limites.

O procedimento inicia com a mudança do intervalo de integração de  $[a, b]$  para  $[-1, 1]$ . Para tanto, efetua-se uma mudança de variável.

Fazendo  $x = \frac{1}{2}[(b-a)t + (b+a)]$ , para  $x = a$ ,  $t = -1$ , para  $x = b$ ,  $t = 1$  e  $dx = \frac{1}{2}(b-a)dt$ .

$$\text{Assim, } I = \int_a^b f(x).dx = \int_{-1}^1 f\left(\frac{1}{2}[(b-a)t + (b+a)]\right) \cdot \frac{1}{2} \cdot (b-a).dt = \int_{-1}^1 \frac{b-a}{2} \cdot f\left(a \cdot \frac{1-t}{2} + b \cdot \frac{1+t}{2}\right) dt.$$

$$\text{Fazendo } F(t) = \frac{b-a}{2} \cdot f\left(a \cdot \frac{1-t}{2} + b \cdot \frac{1+t}{2}\right), \quad I = \int_{-1}^1 F(t).dt.$$

Para dois pontos, tem-se  $I = \int_{-1}^1 F(t).dt = w_0 \cdot F(t_0) + w_1 \cdot F(t_1)$ , onde  $w_0$ ,  $w_1$ ,  $t_0$  e  $t_1$  devem ser determinados e não dependem da função  $F$ . Para determinar as quatro variáveis, devem-se determinar quatro equações que serão definidas para  $F(t) = t^k$ , com  $k = 0, 1, 2, 3$ .

$$\int_{-1}^1 F(t).dt = \int_{-1}^1 t^k.dt = w_0 \cdot F(t_0) + w_1 \cdot F(t_1) = w_0 \cdot t_0^k + w_1 \cdot t_1^k, \quad k = 0, 1, 2, 3.$$

$$\text{Para } k = 0, \int_{-1}^1 t^0.dt = t \Big|_{-1}^1 = 2 \Rightarrow w_0 \cdot t_0^0 + w_1 \cdot t_1^0 = 2$$

$$\text{Para } k = 1, \int_{-1}^1 t^1.dt = \frac{t^2}{2} \Big|_{-1}^1 = 0 \Rightarrow w_0 \cdot t_0^1 + w_1 \cdot t_1^1 = 0$$

$$\text{Para } k = 2, \int_{-1}^1 t^2.dt = \frac{t^3}{3} \Big|_{-1}^1 = \frac{2}{3} \Rightarrow w_0 \cdot t_0^2 + w_1 \cdot t_1^2 = \frac{2}{3}$$

$$\text{Para } k = 3, \int_{-1}^1 t^3.dt = \frac{t^4}{4} \Big|_{-1}^1 = 0 \Rightarrow w_0 \cdot t_0^3 + w_1 \cdot t_1^3 = 0$$

O sistema resultante é

$$\begin{cases} w_0 + w_1 = 2 \\ w_0 \cdot t_0 + w_1 \cdot t_1 = 0 \\ w_0 \cdot t_0^2 + w_1 \cdot t_1^2 = \frac{2}{3} \\ w_0 \cdot t_0^3 + w_1 \cdot t_1^3 = 0 \end{cases}$$

A solução é

$$\begin{cases} w_0 = 1 \\ w_1 = 1 \\ t_0 = -\frac{\sqrt{3}}{3} \\ t_1 = \frac{\sqrt{3}}{3} \end{cases}$$

Então,  $\int_a^b f(x).dx = \int_{-1}^1 F(t).dt = w_0 \cdot F(t_0) + w_1 \cdot F(t_1) = 1 \cdot F\left(-\frac{\sqrt{3}}{3}\right) + 1 \cdot F\left(\frac{\sqrt{3}}{3}\right)$ , sendo que  $F(t)$  é definida como

$$F(t) = \frac{b-a}{2} \cdot f\left(a \cdot \frac{1-t}{2} + b \cdot \frac{1+t}{2}\right).$$

Como a fórmula de Gauss é exata para polinômios até grau  $2n-1$ , neste caso é exata até grau 3.

Calculando numericamente, com a equação precedente, com  $n = 2$ , tem-se o que segue.

$$f(x) = x^2, a = 2, b = 5. \text{ Se } n = 2, \text{ então, os pesos e pontos são } w_0 = 1, w_1 = 1, t_0 = -\sqrt{3}/3, t_1 = \sqrt{3}/3.$$

O valor da integral é a soma ponderada das parcelas, ou seja,  $A = w_0 \cdot F(t_0) + w_1 \cdot F(t_1)$ .

$$\int_a^b f(x) dx = \int_{-1}^1 F(t) dt = w_0 F(t_0) + w_1 F(t_1) = 1 \cdot F\left(-\frac{\sqrt{3}}{3}\right) + 1 \cdot F\left(\frac{\sqrt{3}}{3}\right), \text{ com } F(t) = \frac{b-a}{2} \cdot f\left(a \frac{1-t}{2} + b \frac{1+t}{2}\right).$$

$$w_0 = 1, F(t_0) = \frac{5-2}{2} \cdot f\left(2 \cdot \frac{1+\sqrt{3}/3}{2} + 5 \cdot \frac{1-\sqrt{3}/3}{2}\right) = 1,5 \cdot f(2,633975) = 1,5 \cdot 2,633975^2 = 10,406733,$$

$$w_1 = 1, F(t_1) = \frac{5-2}{2} \cdot f\left(2 \cdot \frac{1-\sqrt{3}/3}{2} + 5 \cdot \frac{1+\sqrt{3}/3}{2}\right) = 1,5 \cdot f(4,366025) = 1,5 \cdot 4,366025^2 = 28,593267.$$

$$I = w_0 \cdot F(t_0) + w_1 \cdot F(t_1) = 1 \cdot 10,406733 + 1 \cdot 28,593267 = 39. \text{ O valor exato é } 39 \text{ e o valor encontrado é } 39.$$

Como a fórmula com 2 pontos oferece valores exatos para polinômios até grau 3, isto era esperado.

Segue um ensaio de algoritmo.

```
Dados f, a, b, n          // n deve ser 2, conforme dedução anterior
w0 = 1                   // valor tabelado
w1 = 1                   // valor tabelado
t0 = -0,5773502691      // valor tabelado
t1 = 0,5773502691       // valor tabelado
A = 0
Para i = 0, ..., n-1 faça
  x = (a.(1-ti) + b.(1+ti))/2
  A = A + wi.f(x)
Integral = (b-a).A/2
```

Para  $n > 2$ , deve-se consultar os valores dos pesos e pontos na tabela correspondente e, evidentemente, deve-se alterar o trecho do algoritmo que atribui os valores de  $w_i$  e  $t_i$ , em um laço adequado.

E a tabela? Continue lendo.

Para  $n$  pontos, a Fórmula de Gauss-Legendre é  $I = \int_{-1}^1 F(t) dt = \sum_{i=0}^{n-1} w_i F(t_i)$ , onde  $w_i$  e  $t_i$  estão tabelados a seguir, para  $1 \leq n \leq 8$ .

n	i	w <sub>i</sub>	t <sub>i</sub>
1	0	2	0
2	0	1,0000000000	-0,5773502691
	1	1,0000000000	+0,5773502691
3	0	0,5555555555	-0,7745966692
	1	0,8888888888	0,0000000000
	2	0,5555555555	+0,7745966692
4	0	0,3478548451	-0,8611363115
	1	0,6521451548	-0,3399810435
	2	0,6521451548	+0,3399810435
	3	0,3478548451	+0,8611363115
5	0	0,2369268850	-0,9061798459
	1	0,4786286704	-0,5384693101
	2	0,5688888888	0,0000000000
	3	0,4786286704	+0,5384693101
	4	0,2369268850	+0,9061798459

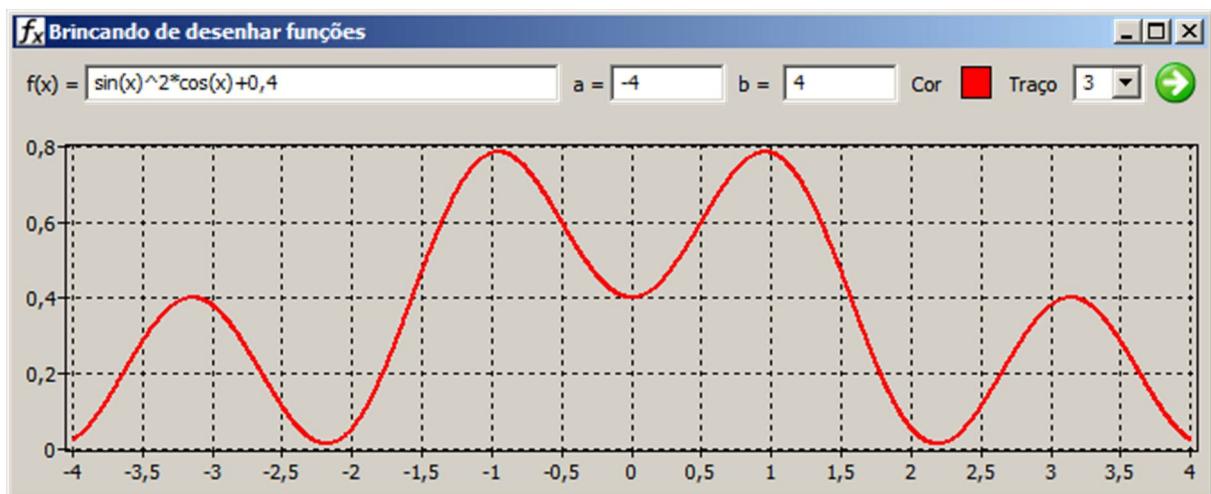
6	0	0,1713244923	-0,9324695142
	1	0,3607615730	-0,6612093864
	2	0,4679139345	-0,2386191860
	3	0,4679139345	+0,2386191860
	4	0,3607615730	+0,6612093864
	5	0,1713244923	+0,9324695142
7	0	0,1294849661	-0,9491079123
	1	0,2797053914	-0,7415311855
	2	0,3818300505	-0,4058451513
	3	0,4179591836	0,0000000000
	4	0,3818300505	+0,4058451513
	5	0,2797053914	+0,7415311855
	6	0,1294849661	+0,9491079123
8	0	0,1012285362	-0,9602898564
	1	0,2223810344	-0,7966664774
	2	0,3137066458	-0,5255324099
	3	0,3626837833	-0,1834346424
	4	0,3626837833	+0,1834346424
	5	0,3137066458	+0,5255324099
	6	0,2223810344	+0,7966664774
	7	0,1012285362	+0,9602898564

## Patricinha volta à cena

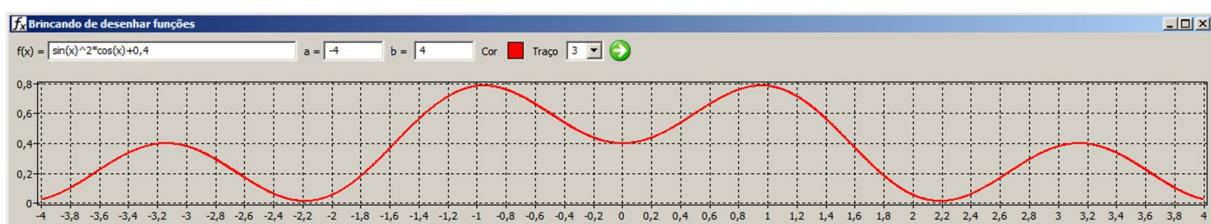
Para testar seus programas, você pode utilizar o Programa da Patricinha, *inventar suas funções*, ver bem as suas caras, estimar graficamente o valor da integral e verificar se seu programa está efetuando os cálculos corretamente.

Já que você reproduziu o Programa da Patricinha (ou não reproduziu?), poderá usar o que aprendeu reproduzindo o programa, incluindo o desenho da função em seu programa de Integrais.

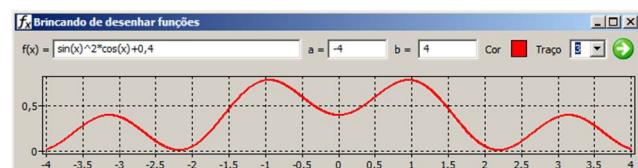
E para exercitar manualmente, utilize esta função vista no Programa da Patricinha para calcular a integral com todos os métodos que você aprendeu (ou não aprendeu?).



Se você já está contando os quadradinhos para saber o valor da Integral, lembre-se que o eixo horizontal e o eixo vertical não tem a mesma escala. Para ver em escala, estique ou encolha na horizontal ou na vertical, como segue. Contando grosseiramente, somando os quadrados inteiros, os quadrados quase inteiros e somando duas ou três pequenas partes como sendo um quadrado, encontrei 74. Assim, a área seria  $74 \cdot 0,2 \cdot 0,2 = 2,96$ . Conte os seus.



Esticando a fôrma de outra forma<sup>19</sup>, obtém-se, em minha contagem, 12 quadradinhos. Assim, a área seria  $12 \cdot 0,5 \cdot 0,5 = 3$ . Interessante, não é? Quanto menor o quadradinho, mais precisa é a contagem. Exatamente o que se pode ver em todos os métodos.



Se cansar de contar os quadradinhos, os resultados analítico e numérico desta quadradura são

$$\int_{-4}^4 (\sin^2 x \cdot \cos x + 0,4) dx = \frac{\sin^3 x}{3} + \frac{2x}{5} \Big|_{-4}^4 = 2 \cdot \frac{\sin^3 4}{3} + 2 \cdot \frac{2 \cdot 4}{5} = 2,9110275720127750362579094907845.$$

<sup>19</sup> Quem foi que disse que o acento diferencial não é necessário? Que raio de acordo ortográfico é este? Imagine se eu escrevesse *esticando a fôrma de outra forma*. O que seria? *Esticando a fôrma de outra fôrma, esticando a fôrma de outra fôrma, esticando a fôrma de outra fôrma ou esticando a fôrma de outra fôrma?* Mais um favor do Senhor *não sei de nada*.

Resolvendo o simpático problema com dois métodos.

1/3 de Simpson, com  $n = 16$ .

$f(x) = \sin^2 x + \cos x + 0,4$  com  $a = -4$ ,  $b = 4$ .

Se  $n = 16$ , então,  $h = (4+4)/16 = 0,5$ .

i	$x_i$	$f(x_i)$	m	$m.f(x_i)$
0	-4	0,025625605	1	0,025625605
1	-3,5	0,28477006	4	1,139080241
2	-3	0,380284441	2	0,760568883
3	-2,5	0,113055267	4	0,452221067
4	-2	0,055920719	2	0,111841438
5	-1,5	0,47038325	4	1,881533001
6	-1	0,782573701	2	1,565147401
7	-0,5	0,601711134	4	2,40684536
8	0	0,4	2	0,8
9	0,5	0,601711134	4	2,40684536
10	1	0,782573701	2	1,565147401
11	1,5	0,47038325	4	1,881533001
12	2	0,055920719	2	0,111841438
13	2,5	0,113055267	4	0,452221067
14	3	0,380284441	2	0,760568883
15	3,5	0,28477006	4	1,139080241
16	4	0,025625605	1	0,025625605
		Soma		17,48572599
		h/3.Soma		2,914287665

Refazendo os cálculos com a Regra 3/8 de Simpson, com  $n = 18$ , tem-se  $h = (4+4)/18 = 0,44444444444444$ . Neste caso, a Soma será 17,50127089 e o valor da integral será 2,916878482.

Gauss, com  $n = 8$ .

$f(x) = \sin^2 x + \cos x + 0,4$  com  $a = -4$ ,  $b = 4$ .

Se  $n = 8$ , então,  $w_i$  e  $t_i$  são os da tabela que segue.

i	$w_i$	$t_i$
0	0,1012285362	-0,9602898564
1	0,2223810344	-0,7966664774
2	0,3137066458	-0,5255324099
3	0,3626837833	-0,1834346424
4	0,3626837833	+0,1834346424
5	0,3137066458	+0,5255324099
6	0,2223810344	+0,7966664774
7	0,1012285362	+0,9602898564

$$I = \int_{-4}^4 F(t) dt = \sum_{i=0}^7 w_i F(t_i) = \sum_{i=0}^7 w_i f(x_i)$$

$$\text{com } x_i = (a.(1-t_i) + b.(1+t_i))/2$$

i	$x_i$	$f(x_i)$	$w_i.f(x_i)$
0	-3,841159426	0,082788753	0,008380584
1	-3,18666591	0,397971839	0,088501389
2	-2,10212964	0,023396477	0,00733963
3	-0,73373857	0,73303991	0,265861688
4	0,73373857	0,73303991	0,265861688
5	2,10212964	0,023396477	0,00733963
6	3,18666591	0,397971839	0,088501389
7	3,841159426	0,082788753	0,008380584
		Soma	0,740166584
		(b-a).Soma/2	2,960666334

A integral analítica, para o intervalo dado é  $I = 2,911027572$ .

Com a Regra 1/3 de Simpson,  $I = 2,914287665$  e com a Regra 3/8 de Simpson,  $I = 2,916878482$ . A interpretação é que as aproximações quadráticas são melhores que as aproximações cúbicas para a função dada.

Com a Regra 1/3 de Simpson,  $I = 2,914287665$  e com a Regra de Gauss,  $I = 2,960666334$ .

E agora? Qual a explicação para isto? Dê mais uma olhada nas definições dos Métodos e lembre-se que um deles soma faixas a partir de função polinomial interpolada nos pontos e o outro, de forma semelhante, para pontos e pesos do próprio método, tem solução exata para funções polinomiais de grau  $2n-1$ , que não é o caso desta função.

Mas, utilizando Gauss com  $n = 8$ , no intervalo  $[-4, 0]$  obtém-se  $I = 1,455515604$ . Por simetria, no intervalo  $[-4, 4]$  o valor da integral é o dobro, ou seja,  $I = 2,911031208$ . Se alguém (algum simpático aluno) criasse tabelas para pesos e pontos de Gauss para  $n = 9, \dots, 32$ , seria super legauss.

E agora, chegou a hora de ...

*Fui interrompido pela Patrincinha, que vem correndo e falando de modo esbaforido.*

*Fessor, fessor, eu fiz a tabela para  $n = 16$ . Vire a página e veja.*

i	w <sub>i</sub>	t <sub>i</sub>
0	0, 0271524594117541	-0, 9894009349916499
1	0, 0622535239386479	-0, 9445750230732326
2	0, 0951585116824928	-0, 8656312023878318
3	0, 1246289712555339	-0, 7554044083550030
4	0, 1495959888165767	-0, 6178762444026438
5	0, 1691565193950025	-0, 4580167776572274
6	0, 1826034150449236	-0, 2816035507792589
7	0, 1894506104550685	-0, 0950125098376374
8	0, 1894506104550685	0, 0950125098376374
9	0, 1826034150449236	0, 2816035507792589
10	0, 1691565193950025	0, 4580167776572274
11	0, 1495959888165767	0, 6178762444026438
12	0, 1246289712555339	0, 7554044083550030
13	0, 0951585116824928	0, 8656312023878318
14	0, 0622535239386479	0, 9445750230732326
15	0, 0271524594117541	0, 9894009349916499

E agora, chegou a hora de ...

*Fui interrompido pelo Joãozinho, que vem correndo e falando de modo esbaforido.*

*Fessor, fessor, eu fiz a tabela para n = 32. Nem vire a página e veja.*

i	w <sub>i</sub>	t <sub>i</sub>
0	0, 0070186100094701	-0, 9972638618494816
1	0, 0162743947309057	-0, 9856115115452684
2	0, 0253920653092621	-0, 9647622555875064
3	0, 0342738629130214	-0, 9349060759377397
4	0, 0428358980222267	-0, 8963211557660521
5	0, 0509980592623762	-0, 8493676137325700
6	0, 0586840934785355	-0, 7944837959679424
7	0, 0658222227763618	-0, 7321821187402897
8	0, 0723457941088485	-0, 6630442669302152
9	0, 0781938957870703	-0, 5877157572407623
10	0, 0833119242269467	-0, 5068999089322294
11	0, 0876520930044038	-0, 4213512761306353
12	0, 0911738786957639	-0, 3318686022821277
13	0, 0938443990808046	-0, 2392873622521371
14	0, 0956387200792749	-0, 1444719615827965
15	0, 0965400885147278	-0, 0483076656877383
16	0, 0965400885147278	0, 0483076656877383
17	0, 0956387200792749	0, 1444719615827965
18	0, 0938443990808046	0, 2392873622521371
19	0, 0911738786957639	0, 3318686022821277
20	0, 0876520930044038	0, 4213512761306353
21	0, 0833119242269467	0, 5068999089322294
22	0, 0781938957870703	0, 5877157572407623
23	0, 0723457941088485	0, 6630442669302152
24	0, 0658222227763618	0, 7321821187402897
25	0, 0586840934785355	0, 7944837959679424
26	0, 0509980592623762	0, 8493676137325700
27	0, 0428358980222267	0, 8963211557660521
28	0, 0342738629130214	0, 9349060759377397
29	0, 0253920653092621	0, 9647622555875064
30	0, 0162743947309057	0, 9856115115452684
31	0, 0070186100094701	0, 9972638618494816

Resolvendo o exemplo anterior, ou seja,  
 $\int_{-4}^4 (\sin^2 x \cdot \cos x + 0,4) dx$ , com a tabela  
da Patricinha, obtém-se o resultado  
I = 2,911027572.

O resultado obtido através da integral  
calculada analiticamente é idêntico,  
com 10 algarismos significativos.

Resolvendo o exemplo anterior, ou seja,  
 $\int_{-4}^4 (\sin^2 x \cdot \cos x + 0,4) dx$ , com a tabela  
do Joãozinho, obtém-se o resultado  
I = 2,911027572.

Não contem para o Joãozinho, mas com  
10 algarismos significativos, o resultado  
com 32 pontos é o mesmo que o resul-  
tado com 16 pontos.

Mas é necessário observar que se as  
tabelas da Patricinha e do Joãozinho  
fossem elaboradas com mais algarismos  
significativos, a do Joãozinho seria me-  
lhore, ou seja, provocaria resultados me-  
lhores para as integrais calculadas com  
suas tabelas.

Então, algum aluno poderá criar tabelas  
com mais algarismos significativos.

Agora, para ser legauss, algum simpático aluno deverá criar tabelas para n = 33, 34, 35, ..., 64, ou  
mais pontos. Além disto, seria adequado que as tabelas tivessem pesos e pontos com 32 algarismos  
significativos, ou mais.

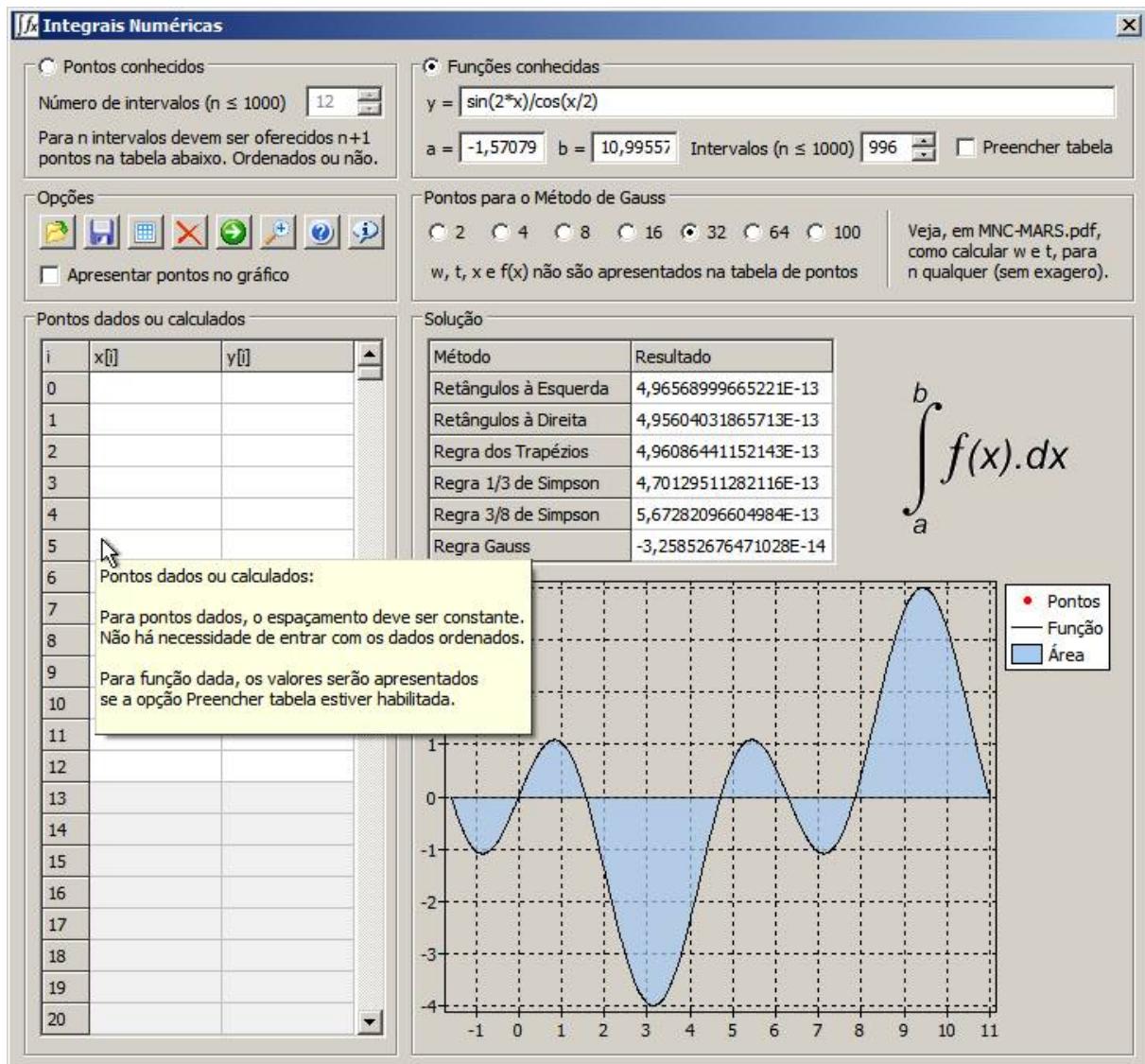
E agora, chegou a hora. Faça um simpático programa que resolva integrais por todos esses métodos.  
Só isso.

## Sugestões para seus programas

Um simpático programa, para resolver integrais numéricas deverá permitir entrada de dados por pontos ou por função. No caso de pontos dados, deve permitir a definição da quantidade de pontos e habilitar uma grade para entrada dos pontos. No caso de funções dadas, deve permitir a entrada da equação de função, dos limites de integração e do número de intervalos ou pontos. Em qualquer um dos casos, deve apresentar os resultados e o gráfico da área calculada.

Além disto, o programa deve permitir que o usuário salve os dados digitados para uso posterior, pois a quantidade de dados pode ser muito grande e se o usuário quiser utilizar os mesmos dados, com algumas alterações, para um novo cálculo, em outra ocasião, poderá carregar os dados que salvou anteriormente.

Um programa com as características descritas poderia ter a cara que segue.



Com este programa é possível:

- entrada de dados por pontos ou por função;
- calcular a solução numérica da integral dada por pontos ou por função;
- desenhar o gráfico com a área da integral numérica.

## Instruções claras e precisas sobre trabalhos computacionais para avaliação

Todas as instruções necessárias estão descritas no mesmo item, do Capítulo 7. Copiar e colar é muito fácil, mas não é nada elegante, agradável, simpático ou conveniente repetir tudo aquilo que já está escrito em capítulo anterior. Além disto, se você já entregou algum trabalho, já sabe como deve proceder. Caso contrário, veja as instruções no Capítulo 7. E antes de entregar, leia as instruções em <http://sacoman.dco.fc.unesp.br/mnc/pet/>.

Se você optar em fazer um programa que tenha a aparência da figura apresentada ou similar, poderá continuar lendo para ter algumas explicações que poderão ser úteis para você.

### Um exemplo de programa

Minha ajuda será *apenas na fabricação da interface*. O resto é com você. Depois de pronto, limpe todo lixo gerado durante a elaboração do projeto, compacte com Zip, Rar ou qualquer outro compactador, com o sugestivo nome T6-RA-V1.ext, onde RA é o número de seu Registro Acadêmico, ext é a extensão fornecida pelo seu compactador, V1 é a primeira versão e entregue via endereço eletrônico de recepção que você já sabe qual é. Não sabe qual é? Então leia tudo sobre o assunto no endereço <http://sacoman.dco.fc.unesp.br/mnc/pet/>.

Então, mãos a obra.

Crie um diretório apropriado para desenvolver seu projeto (e.g.: X:\MNC-Trabalhos\T6). Organize seu ambiente de trabalho e crie subdiretórios necessários para o projeto, como, por exemplo, um diretório chamado Figuras, para as figuras dos botões e ícone e outro chamado Exemplo, para o exemplo executável que está à sua disposição.

Copie de <http://sacoman.dco.fc.unesp.br/mnc/material/> o arquivo M6-Integrais Numéricas.rar e descompacte. Mova IntegraisNumericas.exe, Interpretador.dll e IntegraisNumericas.chm para o diretório exemplo, para executar quando quiser. Mova as figuras para o diretório Figuras.

Inicie seu ambiente de projeto, no meu caso, Lazarus, ajeite as janelas da forma como achar conveniente e salve o projeto com o nome IntegraisNumericas.lpr e a unidade principal com o nome Principal. Pressione o botão Nova Fórmula e salve dando o nome Sobre para esta nova unidade.

No Editor de Código, pressione a aba Sobre, pressione F12 para ver a fórmula, atribua Caption=Sobre o programa Integrais Numéricas, BorderStyle=bsDialog e o resto, faça como considerar conveniente, pois você já leu este tipo de instrução em todos os outros exemplos anteriores. Você pode colocar um ou mais componentes TLabel com os textos que quiser e um componente TImage com sua foto, afirmando que você fez o programa sozinho. Mas não deixe de ver o do programa exemplo que está a sua disposição.

No Editor de Código, pressione a aba Principal, pressione F12 para ver a fórmula, atribua Caption=Integrais Numéricas, BorderStyle=bsSingle, para BorderIcons, biMaximize=False e biMinimize=False, para Font, Name=Tahoma e Size=8, Height=649, Width=720, Position=poDesktopCenter e ShowHint=True.

Lance, na fórmula, alguns componentes como segue.

Componente	Caption	Left	Top	Height	Width
GroupBox1		8	8	80	238
GroupBox2		254	8	80	460
GroupBox3	Opções	8	91	75	238
GroupBox4	Pontos para o Método de Gauss	254	91	75	460
GroupBox5	Pontos dados ou calculados	8	168	476	238
GroupBox6	Solução	254	168	476	460

Lance, na fórmula, bem no topo, dois componentes TRadioButton. Atribua, para o RadioButton1, Left=16, Top=8, Caption=Pontos conhecidos. Atribua, para o RadioButton2, Left=260, Top=8, Caption=Funções conhecidas.

Desta forma, os botões de rádio ocuparão a região dos títulos do GroupBox1 e do GroupBox2, que foram deixados vazios. Além disto, servirão para habilitar e desabilitar o GroupBox1 e o GroupBox2.

No GroupBox1, lance alguns componentes como segue.

Componente	Caption	Left	Top	Width	MinValue	MaxValue	Value
Label1	Número de intervalos ( $n \leq 1000$ )	8	8				
Label2	Para $n$ intervalos devem ser oferecidos $n+1$ pontos na tabela abaixo. Ordenados ou não.	8	30				
SpinEdit1		177	4	50	2	1000	20

Utilize o Editor de Seqüência de Caracteres, da aba Propriedades, para editar o Caption do Label2 em duas linhas.

No GroupBox2, lance alguns componentes como segue.

Componente	Caption	Left	Top	Width	Text	Width	MinValue	MaxValue	Value
Label3	$y =$	8	8						
Label4	$a =$	8	37						
Label5	$b =$	92	37						
Label6	Intervalos ( $n \leq 1000$ )	176	37						
Edit1		28	4	418	(remova)				
Edit2		28	33	56	(remova)				
Edit3		112	33	56	(remova)				
SpinEdit2		283	33	50		50	2	1000	20
CheckBox1	Preencher tabela	344	35						

No GroupBox3, lance alguns componentes como segue.

Componente	Left	Top	Glyph
SpeedButton1	8	3	open.gif
SpeedButton2	36	3	save.gif
SpeedButton3	64	3	grid.gif
SpeedButton4	92	3	delete.gif

Componente	Left	Top	Glyph	Caption
SpeedButton5	120	3	forward.gif	
SpeedButton6	148	3	zoomin.gif	
SpeedButton7	176	3	help2.gif	
SpeedButton8	204	3	inform1.gif	

Lance, também, um TCheckBox (CheckBox2), com Left=8, Top=33 e Caption=Apresentar pontos no gráfico.

No GroupBox4, lance alguns componentes como segue.

Componente	Left	Top	Caption	Height	Shape
RadioButton1	8	6	2		
RadioButton2	48	6	4		
RadioButton3	88	6	8		
RadioButton4	128	6	16		
RadioButton5	168	6	32		
RadioButton6	208	6	64		
RadioButton7	248	6	100		
Label1	11	32	w, t, x e f(x) não são apresentados na tabela de pontos		
Label2	314	6	Veja, em MNC-MARS.pdf, como calcular w e t, para n qualquer (sem exagero).		
Bevel1	300	0		50	bsLeftLine
SpeedButton8	204	3			

No GroupBox5, lance um TStringGrid (StringGrid1) com Left=8, Top=4, ColCount=3, RowCount=1002 e, em Options, acrescente goEditing=True.

No GroupBox6, lance um TStringGrid (StringGrid2) com Left=8, Top=4, ColCount=2, RowCount=7 e DefaultColWidth=128.

Lance um TImage (Image1) com Left=294, Top=28, em Anchors, desmarque akLeft e marque akRight e carregue, em Picture, a figura Integral.png.

Lance um TChart (Chart1) com Left=0, Top=152, Width=450, Height=300, em Legend, atribua Visible=True.

Marque o TChart1 e, delicadamente, cutuque o botão direito do mouse e selecione, no menu suspenso, o item Edit series.

No editor de séries, pressione o botão Add e selecione, no menu suspenso, Line series, para adicionar Chart1LineSeries1. Pressione, novamente, o botão Add e selecione, no menu suspenso, Line series, para adicionar Chart1LineSeries2. Pressione, mais uma vez, o botão Add e selecione, no menu suspenso, Area series, para adicionar Chart1AreaSeries1. Mantenha o editor de séries aberto.

No editor de séries, marque Chart1LineSeries1. No Editor de Propriedades, atribua Title=Pontos, LineType=ltNone, Pointer.Brush.Color=clRed, Pointer.HorzSize=2, Pointer.Pen.Color=clRed, Pointer.Style=psCircle, Pointer.VertSize=2 e ShowPoints=True.

No editor de séries, marque Chart1LineSeries2. No Editor de Propriedades, atribua Title=Função.

No editor de séries, marque Chart1AreaSeries1. No Editor de Propriedades, atribua Title=Área, AreaBrush.Color=clSkyBlue, AreaLinesPen.Color=clSkyBlue, Transparency=75 e UseZeroLevel=True.

Feche o editor de séries e lance os dois últimos componentes, que não são visuais, que servirão como auxiliares para abrir e salvar arquivos.

Lance em qualquer lugar da forma um TOpenDialog (OpenDialog1) e atribua Title= Abrir arquivo existente, DefaultExt=.txt, Filter=Arquivo texto de dados para integral| \*.txt|Todos os arquivos| \*.\* e InitialDir=. e pode-se utilizar o Editor de filtro para editar o filtro. Deixe um espaço à esquerda da palavra Abrir, em Title.

Lance em qualquer lugar da forma um TSaveDialog (SaveDialog1) e atribua Title= Salvar arquivo como, DefaultExt=.txt, Filter=Arquivo texto de dados para integral| \*.txt|Todos os arquivos| \*.\* e InitialDir=. e pode-se utilizar o Editor de filtro para editar o filtro. Deixe um espaço à esquerda da palavra Salvar, em Title.

Todos os componentes estão lançados. Os textos para Hint são escritos no programa, pois fica mais fácil para editar textos longos ou com várias linhas. Além disto, fica mais fácil para efetuar alterações. Retorne algumas páginas e verifique se a fórmula se parece com a fórmula da figura apresentada como sugestão de programa. Então, agora, só falta a CPFL interromper o fornecimento de energia elétrica e você perder tudo. Que tal salvar? **SALVE**. Você já sabe que salvar é a salvação. Agora, pode até compilar e ver uma interface completa, mas que não faz nada. Então só falta fazer o programa.

Para criar as rotinas que serão executadas quando os eventos ocorrem, acompanhe a seção cutuque aqui e cutuque ali, que segue. Para cada componente cutucado, utilize a aba Eventos para dar um toque duplo no evento conveniente para criar a rotina. Para cada rotina criada, insira um comentário após o comando begin, para documentar as rotinas. A tabela que segue apresenta, para cada componente, o nome, o evento, a rotina associada e uma sugestão de comentário.

Componente	Evento	Rotina criada	Comentário
Form1	OnCreate	FormCreate	// Tempo de hint de 20 segundos = 20000 ms
RadioButton1	OnClick	RadioButton1Click	// Entrada de dados para Pontos conhecidos
RadioButton2	OnClick	RadioButton2Click	// Entrada de dados para Funções conhecidas
SpinEdit1	OnExit	SpinEdit1Exit	// Ao sair do campo SpinEdit1, executar automaticamente SpeedButton3Click para redimensionar a grade
SpeedButton1	OnClick	SpeedButton1Click	// Abrir arquivo, ler dados e preencher campos
SpeedButton2	OnClick	SpeedButton2Click	// Salvar arquivo com dados dos campos preenchidos
SpeedButton3	OnClick	SpeedButton3Click	// Habilitar grade para a quantidade de pontos estabelecidos
SpeedButton4	OnClick	SpeedButton4Click	// Limpar grades, dados e gráficos
SpeedButton5	OnClick	SpeedButton5Click	// Resolver os problemas para 1 Pontos dados e 2 Função dada
SpeedButton6	OnClick	SpeedButton6Click	// Zoom
SpeedButton7	OnClick	SpeedButton7Click	// Ajuda
SpeedButton8	OnClick	SpeedButton8Click	// Informa
StringGrid1	OnPrepareCanvas	StringGrid1PrepareCanvas	// Ao desenhar a grade, deixar células editáveis brancas e não editáveis cinzas
StringGrid1	OnSelectCell	StringGrid1SelectCell	// Não permitir edição nas células não editáveis cinzas

Além dessas rotinas existem outras que não são disparadas por eventos. São rotinas para atribuir pontos e pesos de Gauss, verificar se existem dados repetidos, ordenar dados da grade quando forem digitados fora de ordem crescente, verificar se os espaçamentos dos dados digitados são constantes e *calcular os seis casos de integrais apresentados neste texto*.

A sua parte é apenas *calcular os seis casos de integrais apresentados neste texto*.

Então, acompanhe atentamente a listagem do programa que segue, para entender, para recheiar as rotinas e para fazer a sua parte. As rotinas estão completamente documentadas. São longas, mas são simples. E enquanto estiver recheando as rotinas, **SALVE SEMPRE**.

```

unit Principal;
{$mode objfpc}{$H+}

interface

uses
  Classes, SysUtils, FileUtil, Forms, Controls, Graphics, Dialogs, StdCtrls,
  ExtCtrls, Spin, Buttons, Grids, TAGraph, TASeries, TAFuncSeries;

type
  { TForm1 }

TForm1 = class(TForm)
  Bevel1: TBevel;
  ...
  StringGrid2: TStringGrid;
  procedure FormCreate(Sender: TObject);
  ...
  procedure StringGrid1SelectCell(Sender: TObject; aCol, aRow: Integer;
    var CanSelect: Boolean);
private
  { private declarations }
public
  { public declarations }
end;

var
  Form1: TForm1;

implementation

{$R *.lfm}

{ TForm1 }

uses
  Sobre, LCLIntf; // LCLIntf para usar OpenDocument

var
  // Para o caso de Pontos conhecidos
  n: Integer; // Número de intervalos = n e número de pontos = n+1
  // Para o caso de Funções conhecidas
  a, b, h, Integral: Extended; // Limites a, b, passo ou largura das faixas h, va-
  lor da Integral
  f: string; // Função
  p, fp: Extended; // Ponto p e valor de f(p)
  Erro: Word; // Erro ao avaliar a função com Interpretador.dll (0 = sem erro)
  w, t: array[0..99] of Extended; // Pesos e pontos de Gauss
  nGauss: Integer; // Número de pontos de Gauss
  // Para os dois casos
  x, y: array[0..1000] of Extended; // Pontos x e y = f(x) para calcular Integral e desenhar Gráfico

function FxR1(f: String; x: Extended; var y: Extended): Word; stdcall; external 'Interpre-
tador.dll';

procedure PontosPesosGauss(nGauss: Integer);
(*$I twConst.pas *) // Inclui arquivo twConst.pas com tabelas de (w, t) para Gauss
var
  k: Integer;
begin
  FillChar(w, SizeOf(w), 0); // Limpa vetores w, t
  FillChar(t, SizeOf(t), 0);
  case nGauss of
    2: for k := 0 to 1 do
      begin
        w[k] := wConst2[k];
        t[k] := tConst2[k];
      end;
    4: for k := 0 to 3 do
      begin
        w[k] := wConst4[k];
        t[k] := tConst4[k];
      end;
  end;
end;

```

```

8: for k := 0 to 7 do
begin
  w[k] := wConst8[k];
  t[k] := tConst8[k];
end;
16: for k := 0 to 15 do
begin
  w[k] := wConst16[k];
  t[k] := tConst16[k];
end;
32: for k := 0 to 31 do
begin
  w[k] := wConst32[k];
  t[k] := tConst32[k];
end;
64: for k := 0 to 63 do
begin
  w[k] := wConst64[k];
  t[k] := tConst64[k];
end;
100: for k := 0 to 99 do
begin
  w[k] := wConst100[k];
  t[k] := tConst100[k];
end;
end;

function DadosRepetidos(var r, s: Integer): Boolean;
var
  i, j: Integer;
begin
  Result := False;
  for i := 0 to n-1 do
    for j := i+1 to n do
      if x[i] = x[j] then
        begin
          r := i;
          s := j;
          Result := True;
          Exit;
        end;
  end;
procedure OrdenarDados;
var
  Trocou: Boolean;
  Auxiliar: Extended;
  i, j: Integer;
begin
  // Ordena valores de forma crescente nos vetores x e y, mantendo os dados da grade como estão
  Trocou := True;
  j := n-1;
  while Trocou do
    begin
      Trocou := False;
      for i := 0 to j do
        if x[i] > x[i+1] then
          begin
            Auxiliar := x[i]; x[i] := x[i+1]; x[i+1] := Auxiliar;
            Auxiliar := y[i]; y[i] := y[i+1]; y[i+1] := Auxiliar;
            Trocou := True;
          end;
      j := j -1;
    end;
  end;

function EspacamentoConstante: Boolean;
var
  i: Integer;
begin
  // Verifica se os espaçamentos dos valores x digitados, são constantes, com erro máximo de 1E-5
  Result := True;

```

```

for i := 1 to n do
  if Abs( x[i] - x[i - 1] - h ) / h > 1E-5 then
    begin
      Result := False;
      Break;
    end;
  end;

// Funções para cálculo das Integrais
// São chamadas pela rotina SpeedButton5Click
// Só são chamadas se os dados de Pontos conhecidos ou Funções conhecidas foram testados

function RetangulosE: Extended;
var
  ...
begin
  ...
end;

function RetangulosD: Extended;
begin
  ...
end;

function Trapezios: Extended;
begin
  ...
end;

function Simpson13: Extended;
begin
  ...
end;

function Simpson38: Extended;
begin
  ...
end;

function Gauss: Extended;
begin
  ...
end;

procedure TForm1.FormCreate(Sender: TObject);
var
  i: Integer;
begin
  // Tempo de hint de 20 segundos = 20000 ms
  Application.HintHelpPause := 20000;

  // Textos para Informações (Hint) de alguns componentes
  // No GroupBox1
  RadioButtton1.Hint := 'Integral numérica para pontos conhecidos:' #10#10+
    'Utilize esta opção para entrar com os' #10+
    'pontos conhecidos na tabela abaixo.' #10;
  SpinEdit1.Hint := 'Número de intervalos (n):' #10#10+
    'Para a sequência de pontos (x[i], y[i]),' #10+
    'i = 0, ..., n, o total de pontos é n+1.' #10#10+
    'Entendeu, ou preciso fazer um desenho?' #10;
  // No GroupBox2
  RadioButtton2.Hint := 'Integral numérica para função conhecida:' #10#10+
    'Utilize esta opção para entrar com' #10+
    'f, a, b e n, nos campos abaixo.' #10;
  Edit1.Hint := 'Função a ser integrada:' #10#10+
    'Escreva a expressão de f(x).' #10+
    'Os pontos (x[i], y[i]) serão calculados' #10+
    'considerando os valores de a, b e n.' #10;
  Edit2.Hint := 'Limite inferior de integração:' #10#10+
    'O valor de x[0] é o limite inferior da integração.' #10;

```

Esta é a sua parte  
Confio em você

```

Edit3.Hint := 'Limite superior de integração: '#10#10+
  'O valor de x[n] é o limite superior da integração.'#10;
SpinEdit2.Hint := 'Número de intervalos entre os pontos: '#10#10+
  'O número de intervalos é n.'#10+
  'O tamanho de cada intervalo é h = (x[n]-x[0])/n.'#10;
CheckBox1.Hint := 'Preencher tabela:'#10#10+
  'Habilite para ver os pontos calculados'#10+
  'na tabela Pontos dados ou calculados.'#10;

// No GroupBox3
SpeedButton1.Hint := 'Abrir:'#10#10+
  'Permite selecionar arquivo de dados'#10+
  'para preencher a grade dos pontos ou' #10+
  'para preencher informações de função.'#10;
SpeedButton2.Hint := 'Salvar:'#10#10+
  'Permite salvar arquivo com dados'#10+
  'da grade dos pontos ou dados de' #10+
  'informação de função de um problema.'#10#10+
  'Os dados salvos são relativos aos'#10+
  'pontos ou função, conforme o tipo'#10+
  'de problema em uso.'#10;
SpeedButton3.Hint := 'Habilitar tabela de entrada de dados:'#10#10+
  'Deve-se habilitar a entrada de dados quando' #10+
  '- o número n+1 de pontos for alterado;'#10+
  '- mudar da opção Função conhecida para Pontos conhecidos.'#10;
SpeedButton4.Hint := 'Limpardados de cálculos anteriores:'#10#10+
  'Limpaa todos os dados entrados pelo usuário'#10+
  'e todos os valores e gráficos calculados.'#10;
SpeedButton5.Hint := 'Calcular integral e desenha gráficos:'#10#10+
  'Para Pontos conhecidos, desenha os pontos e trapézios para representar a área aproximada.'#10+
  'Para Função conhecida, desenha os pontos, os trapézio e a função dada.'#10;
SpeedButton6.Hint := 'Ampliar gráfico:'#10#10+
  'Apresenta apenas a região do gráfico ampliada.'#10+
  'Cutoque e veja a coisa funcionando.'#10+
  'E se não quiser ver, não cutoque.'#10#10+
  'Mas é interessante.'#10+
  'Acho que você deveria cutucar.'#10#10+
  'Patriinha e Joãozinho unissonamente:'#10+
  'Mas como fazer isto?'#10#10+
  'Isto não faz parte de MNC.'#10+
  'Mas poderão aprender a fazer isto em PAW. :)'#10#10+
  'Patriinha:'#10+
  'Vou me matricular na disciplina. :)'#10#10+
  'Joãozinho:'#10+
  'Vou aprender sozinho consultando o Gúgou. : -0' #10#10+
  'Patriinha:'#10+
  'Se é com o Gúgou, não é sozinho.'#10#10+
  'Joãozinho:'#10+
  'Tá bom, vou encontrar 15 bichões de informações, mas farei isto sozinho.'#10#10+
  'Patriinha:'#10+
  'Quer saber de uma coisa?'#10#10+
  'Joãozinho:'#10+
  'Quero. Fale se for mulher.'#10#10+
  'Patriinha:'#10+
  'Você é um ... '#10#10+
  'Cartão vermelho para a Patriinha, novamente.'#10+
  'Fim da aula.'#10+
  'O que aconteceu depois, não sei.'#10#10+
  'Mas curse PAW, é muito LEGAW.'#10+
  'Se você não sabe quem são MNC, Patriinha, Joãozinho ou PAW, então:'#10#10+
  '- você é um aluno que nunca leu MNC-MARS.pdf; '#10+
  '- você é um pirata que está utilizando o programa indevidamente.'#10;
SpeedButton7.Hint := 'Ajuda:'#10#10+
  'Apresenta o arquivo de ajuda, caso exista.'#10#10+
  'Pois é, se não existir, não apresenta. :)'#10;
SpeedButton8.Hint := 'Informação:'#10#10+
  'Apresenta informações sobre autoria e direitos.'#10#10+
  'Cuidado: Pirataria é crime. :-( '#10;
CheckBox2.Hint := 'Apresentar os pontos:'#10#10+
  'Habilite para ver os pontos no gráfico.'#10#10+
  'Se a quantidade de pontos for muito grande será,'#10+
  'visto um traço grosso vermelho dos pontos justapostos.'#10#10+
  'Portanto, use esta opção de forma adequada.'#10;

```

```

// No GroupBox4
Label8.Hint := 'Certamente é o seu texto preferido.'#10#10+
  'E se não for, não me deixe saber disto. :-)'#10;
// No GroupBox5
StringGrid1.Hint := 'Pontos dados ou calculados:'#10#10+
  'Para pontos dados, o espaçamento deve ser constante.'#10#10+
  'Não há necessidade de entrar com os dados ordenados.'#10#10+
  'Para função dada, os valores serão apresentados'#10#10+
  'se a opção Preencher tabela estiver habilitada.'#10;
// No GroupBox6
StringGrid2.Hint := 'Solução numérica:'#10#10+
  'Apresenta solução para cada método.'#10#10+
  'Se houver erro, a mensagem será' #10#10+
  'apresentada na coluna Resultado.'#10;
Image1.Hint := 'Homenagem ao Toninho:'#10#10+
  'Vi va o Toninho!'#10#10+
  'Vi va!'#10#10+
  'Se você não entendeu, deve ser um desses' #10#10+
  'piratas que usam programas indevidamente.'#10;

// Botão Zoom out escondido
SpeedButton6B.Visible := False;

// Diferentes versões do Windows têm diferentes tamanhos para elementos de grade
// Este programa foi desenvolvido e compilado com Windows XP
// No Windows XP DefaultRowHeight = 20, mas no Windows 10, por exemplo, é maior
// Para ficar correto, seguem as linhas que forçam DefaultRowHeight = 20 para todas as versões
// Se o programa for executado apenas em Windows XP estas linhas são absolutamente desnecessárias
StringGrid1.DefaultRowHeight := 20;
StringGrid2.DefaultRowHeight := 20;

// Tamanho e texto da Grade 1
with StringGrid1 do
begin
  ColWidths[0] := 30;
  ColWidths[1] := 84;
  ColWidths[2] := 84;
  Width := 220;           // Projetoado com Windows XP, mas deve funcionar em outras versões
  Height := 22*20+4;
  Cells[0, 0] := 'i';
  Cells[1, 0] := 'x[i]';
  Cells[2, 0] := 'y[i]';
  for i := 1 to 1001 do
    Cells[0, i] := IntToStr(i-1);
end;
RadioButton1.Checked := True;

// Texto da Grade 2
with StringGrid2 do
begin
  Cells[0, 0] := 'Método';
  Cells[1, 0] := 'Resultado';
  Cells[0, 1] := 'Retângulos à Esquerda';
  Cells[0, 2] := 'Retângulos à Direita';
  Cells[0, 3] := 'Regra dos Trapézios';
  Cells[0, 4] := 'Regra 1/3 de Simpson';
  Cells[0, 5] := 'Regra 3/8 de Simpson';
  Cells[0, 6] := 'Regra Gauss';
end;

// A forma é criada com RadioButton1 marcado e, então, inicial é o valor de SpinEdit1
n := SpinEdit1.Value;

end;

procedure TForm1.RadioButton1Click(Sender: TObject);
begin
  // Entrada de dados para Pontos conhecidos
  SpinEdit1.Enabled := True;
  Edit1.Enabled := False;
  Edit2.Enabled := False;
  Edit3.Enabled := False;
  SpinEdit2.Enabled := False;
  CheckBox1.Enabled := False;

```

```

StringGrid1.Options := StringGrid1.Options+[goEditing];
if StringGrid1.CanSetFocus then
  SpinEdit1EditText(SelF);
end;

procedure TForm1.RadioButton2Click(Sender: TObject);
begin
  // Entrada de dados para Funções conhecidas
  SpinEdit1.Enabled := False;
  Edit1.Enabled := True;
  Edit2.Enabled := True;
  Edit3.Enabled := True;
  SpinEdit2.Enabled := True;
  CheckBox1.Enabled := True;
  StringGrid1.Options := StringGrid1.Options-[goEditing];
end;

procedure TForm1.SpeedButton1Click(Sender: TObject);
var
  Lista: TStringList;           // Lista de cadeias de caracteres dos dados, para ler em arquivos
  i: Integer;
begin
  // Abrir arquivo, ler dados e preencher campos
  if not OpenDialog1.Execute then // Se não informou arquivo, cai fora
    Exit;
  Lista := TStringList.Create;    // Cria Lista
  Lista.LoadFromFile(OpenDialog1.FileName);
  if Lista[0] = 'Pontos' then    // Pontos dados
  begin
    RadioButton1.Checked := True;
    SpinEdit1.Value := StrToInt(Lista[1]);
    StringGrid1.Clear([gzNormal]); // Limpar grade de Pontos dados ou calculados
    StringGrid1.TopRow := 1;       // Rolar grade para ficar com a linha 1 no topo
    for i := 0 to SpinEdit1.Value do
      StringGrid1.Cells[1, i+1] := Lista[i+2];
    for i := 0 to SpinEdit1.Value do
      StringGrid1.Cells[2, i+1] := Lista[SpinEdit1.Value+i+3];
    SpeedButton3.Click;           // Ajustar grade (cores e células editáveis)
    Lista.Free;                  // Libera e destroi Lista para reduzir uso de memória
    Exit;
  end;
  if Lista[0] = 'Função' then    // Função dada
  begin
    RadioButton2.Checked := True;
    Edit1.Text := Lista[1];
    Edit2.Text := Lista[2];
    Edit3.Text := Lista[3];
    SpinEdit2.Value := StrToInt(Lista[4]);
    CheckBox1.Checked := Lista[5] = '1';
    RadioButton3.Checked := Lista[6] = '2';
    RadioButton4.Checked := Lista[6] = '4';
    RadioButton5.Checked := Lista[6] = '8';
    RadioButton6.Checked := Lista[6] = '16';
    RadioButton7.Checked := Lista[6] = '32';
    RadioButton8.Checked := Lista[6] = '64';
    RadioButton9.Checked := Lista[6] = '100';
    Lista.Free;                  // Libera e destroi Lista para reduzir uso de memória
    Exit;
  end;
end;

procedure TForm1.SpeedButton2Click(Sender: TObject);
var
  Lista: TStringList;           // Lista de cadeias de caracteres dos dados, para salvar em arquivos
  i: Integer;
begin
  // Salvar arquivo com dados dos campos preenchidos
  Lista := TStringList.Create;    // Cria Lista
  if RadioButton1.Checked then    // Pontos dados
  begin
    Lista.Add('Pontos');
    Lista.Add(IntToStr(SpinEdit1.Value));
  end;

```

```

for i := 1 to SpinEdit1.Value+1 do
  Lista.Add(StringGrid1.Cells[1, i]);
for i := 1 to SpinEdit1.Value+1 do
  Lista.Add(StringGrid1.Cells[2, i]);
if SaveDialog1.Execute then
  Lista.SaveToFile(SaveDialog1.FileName);
Lista.Free; // Libera e destroi Lista para reduzir uso de memória
Exit;
end;
if RadioButton2.Checked then // Função dada
begin
  Lista.Add('Função');
  Lista.Add(Edit1.Text);
  Lista.Add(Edit2.Text);
  Lista.Add(Edit3.Text);
  Lista.Add(IntToStr(SpinEdit2.Value));
  if CheckBox1.Checked then Lista.Add('1') else Lista.Add('0');
  if RadioButon3.Checked then Lista.Add('2');
  if RadioButon4.Checked then Lista.Add('4');
  if RadioButon5.Checked then Lista.Add('8');
  if RadioButon6.Checked then Lista.Add('16');
  if RadioButon7.Checked then Lista.Add('32');
  if RadioButon8.Checked then Lista.Add('64');
  if RadioButon9.Checked then Lista.Add('100');
  if SaveDialog1.Execute then
    Lista.SaveToFile(SaveDialog1.FileName);
  Lista.Free; // Libera e destroi Lista para reduzir uso de memória
  Exit;
end;
end;

procedure TForm1.SpeedButton3Click(Sender: TObject);
begin
  // Habilitar grade para a quantidade de pontos estabelecidos
  if RadioButon2.Checked then // Se Função conhecida, não habilita
  begin
    ShowMessage('Habilitar Tabela de entrada de dados é uma opção para Pontos conhecidos.'#10+
      'Para Função conhecida, os pontos serão calculados e poderão ser apresentados.'#10+
      'Para apresentá-los, marque a opção Preencher tabela.');
    Exit;
  end; // Se Pontos conhecidos, habilita
  n := SpinEdit1.Value;
  StringGrid1.SetFocus;
  StringGrid1.Refresh;
end;

procedure TForm1.SpeedButton4Click(Sender: TObject);
begin
  // Limpar grades, dados e gráficos
  with StringGrid1 do
  begin
    SpinEdit1.Value := 20;
    Edit1.Text := '';
    Edit2.Text := '';
    Edit3.Text := '';
    SpinEdit2.Value := 20;
    CheckBox1.Checked := False;
    CheckBox2.Checked := False;
    RadioButon5.Checked := True;
    StringGrid1.Clear([gzNormal]);
    StringGrid2.Clear([gzNormal]);
    Chart1LineSeries1.Clear;
    Chart1LineSeries2.Clear;
    Chart1AreaSeries1.Clear;
    n := 20;
  end;
end;

```

```

procedure TForm1.SpeedButton5Click(Sender: TObject);
var
  i: Integer;
  r, s: Integer;
begin
  // Resolver os problemas para |1 Pontos dados| e |2 Função dada|
  // |1 Pontos dados|
  if RadioGroup1.Checked then
  begin
    n := SpinEdit1.Value;
    with StringGrid1 do
    begin
      // Ler coluna x da grade e armazenar no vetor x
      for i := 0 to n do
      begin
        try
          x[i] := StrToFloat(Cells[1, i+1]);
        except
          ShowMessage('Verifique o valor de x[' + IntToStr(i) +'].');
          Col := 1;
          Row := i+1;
          SetFocus;
          Exit;
        end
      end;
      // Ler coluna y da grade e armazenar no vetor y
      for i := 0 to n do
      begin
        try
          y[i] := StrToFloat(Cells[2, i+1]);
        except
          ShowMessage('Verifique o valor de y[' + IntToStr(i) +'].');
          Col := 2;
          Row := i+1;
          SetFocus;
          Exit;
        end
      end;
    end;
    // Verificar se há valores de x repetidos
    if DadosRepetidos(r, s) then
    begin
      ShowMessage('Os valores de x[' + IntToStr(r) +'] e de x[' + IntToStr(s) +'] são iguais.'#10+
                  'Remova ou altere os dados repetidos para valores de x.');
      Exit;
    end;
    // Ordenar os dados com x crescente
    OrdenarDados;
    // Calcular espaçamento constante h para comparações dos espaços de x e para utilizar nos métodos
    h := (x[n]-x[0])/n;
    // Verificar se espaçamento é constante
    if not EspacamentoConstante then
    begin
      ShowMessage('Os valores de x, já ordenados, não apresentam espaçamento constante.'#10+
                  'A tolerância utilizada foi 1E-5 para calcular o erro relativo.'#10+
                  'Este método requer espaçamento constante para valores de x.');
      Exit;
    end;
    // Executar cada Método (RetangulosE, RetangulosD, Trapezi os, Simpson13, Simpson38)
    try
      StringGrid2.Cells[1, 1] := FloatToStr(RetangulosE);
    except
      StringGrid2.Cells[1, 1] := 'Erro ao calcular a integral';
    end;
    try
      StringGrid2.Cells[1, 2] := FloatToStr(RetangulosD);
    except
      StringGrid2.Cells[1, 2] := 'Erro ao calcular a integral';
    end;
    try
      StringGrid2.Cells[1, 3] := FloatToStr(Trapezios);
    except
      StringGrid2.Cells[1, 3] := 'Erro ao calcular a integral';
    end;
  end;
end;

```

```

try
  StringGrid2.Cells[1, 4] := FloatToStr(Simpson13);
except
  StringGrid2.Cells[1, 4] := 'Erro ao calcular a integral';
end;
try
  StringGrid2.Cells[1, 5] := FloatToStr(Simpson38);
except
  StringGrid2.Cells[1, 5] := 'Erro ao calcular a integral';
end;
// Não se calcula Integral de Gaus com Pontos Dados
StringGrid2.Cells[1, 6] := 'Não se aplica a pontos';
// Desenhar pontos dados e trapézios representativos de áreas aproximadas
Chart1LineSeries1.Clear;
Chart1LineSeries2.Clear;
Chart1AreaSeries1.Clear;
for i := 0 to n do
begin
  if CheckBox2.Checked then
    Chart1LineSeries1.AddXY(x[i], y[i]);
  Chart1LineSeries2.AddXY(x[i], y[i]);
  Chart1AreaSeries1.AddXY(x[i], y[i]);
end;
end;

// || 2 Função dada ||
if RadioButon2.Checked then
begin
  f := Trim(Edit1.Text);
  if f = '' then
  begin
    ShowMessage('Informe a função.');
    Edit1.SetFocus;
    Exit;
  end;
  try
    a := StrToFloat(Edit2.Text);
  except
    ShowMessage('Valor de a incorreto.');
    Edit2.SetFocus;
    Exit;
  end;
  try
    b := StrToFloat(Edit3.Text);
  except
    ShowMessage('Valor de b incorreto.');
    Edit3.SetFocus;
    Exit;
  end;
  if b <= a then
  begin
    ShowMessage('b deve ser maior que a.');
    Edit3.SetFocus;
    Exit;
  end;
  n := SpinEdit2.Value;
  // Calcular espaçamento constante h para calcular os pontos e utilizar nos métodos
  h := (b-a)/n;
  for i := 0 to n do
  begin
    p := a+i*h;
    Erro := FxR1(f, p, fp);
    if Erro <> 0 then
    begin
      ShowMessage('Erro ao avaliar f(x) no ponto '+FloatToStr(p)+'.');
      Exit;
    end;
    x[i] := p;
    y[i] := fp;
  end;

```

```

if CheckBox1.Checked then // Apresenta pontos calculados da função dada
begin
  StringGrid1.Clean([gzNormal]); // Limpar grade de Pontos dados ou calculados
  StringGrid1.TopRow := 1; // Rolar grade para ficar com a linha 1 no topo
  // Habilitar grade para a quantidade de pontos estabelecidos
  SpinEdit1.Value := SpinEdit2.Value;
  StringGrid1.SetFocus;
  StringGrid1.Refresh;
  for i := 0 to n do
  begin
    StringGrid1.Cells[1, i+1] := FloatToStr(x[i]);
    StringGrid1.Cells[2, i+1] := FloatToStr(y[i]);
  end;
end;
// Definir número de pontos para Método de Gauss (nGauss)
if RadioButon3.Checked then
  nGauss := 2;
if RadioButon4.Checked then
  nGauss := 4;
if RadioButon5.Checked then
  nGauss := 8;
if RadioButon6.Checked then
  nGauss := 16;
if RadioButon7.Checked then
  nGauss := 32;
if RadioButon8.Checked then
  nGauss := 64;
if RadioButon9.Checked then
  nGauss := 100;
// Carregar vetores w, t para número de pontos definido
PontosPesosGauss(nGauss);
// Executar cada Método (RetangulosE, RetangulosD, Trapezios, Simpson13, Simpson38, Gauss)
try
  StringGrid2.Cells[1, 1] := FloatToStr(RetangulosE);
except
  StringGrid2.Cells[1, 1] := 'Erro ao calcular a integral';
end;
try
  StringGrid2.Cells[1, 2] := FloatToStr(RetangulosD);
except
  StringGrid2.Cells[1, 2] := 'Erro ao calcular a integral';
end;
try
  StringGrid2.Cells[1, 3] := FloatToStr(Trapezios);
except
  StringGrid2.Cells[1, 3] := 'Erro ao calcular a integral';
end;
try
  StringGrid2.Cells[1, 4] := FloatToStr(Simpson13);
except
  StringGrid2.Cells[1, 4] := 'Erro ao calcular a integral';
end;
try
  StringGrid2.Cells[1, 5] := FloatToStr(Simpson38);
except
  StringGrid2.Cells[1, 5] := 'Erro ao calcular a integral';
end;
try
  StringGrid2.Cells[1, 6] := FloatToStr(Gauss);
except
  StringGrid2.Cells[1, 6] := 'Erro ao calcular a integral';
end;
// Desenhar pontos dados e trapézios representativos de áreas aproximadas
Chart1LineSeries1.Clear;
Chart1LineSeries2.Clear;
Chart1AreaSeries1.Clear;
for i := 0 to n do
begin
  if CheckBox2.Checked then
    Chart1LineSeries1.AddXY(x[i], y[i]);
    Chart1LineSeries2.AddXY(x[i], y[i]);
    Chart1AreaSeries1.AddXY(x[i], y[i]);
  end;
end;
end;

```

```
procedure TForm1.SpeedButton6Click(Sender: TObject);
begin
  // Zoom
```



Aqui foi feita a rotina que destaca o gráfico, colocando o componente em uma forma redimensionável.

```
end;
```

Assunto para PAW.

```
procedure TForm1.SpeedButton6BClick(Sender: TObject);
begin
  // Zoom out
```



Aqui foi feita a rotina que recoloca o gráfico na fôrma principal não redimensionável.

```
end;
```

Assunto para PAW.

```
procedure TForm1.SpeedButton7Click(Sender: TObject);
begin
```

```
  // Ajuda
```

```
  // Você pode fazer um arquivo Integrals.chm ou um arquivo Integrals.pdf ou um simples arquivo Integrals.txt
  // Escreva o nome do seu arquivo de ajuda na função OpenDocument
```

```
  if not OpenDocument('IntegralsNumericas.chm') then
```

```
    MessageDlg('Informação', 'Arquivo de Ajuda' +#10+ 'IntegralsNumericas.chm' +#10+'não foi encontrado.', mtInformation, [mbOk], 0);
```

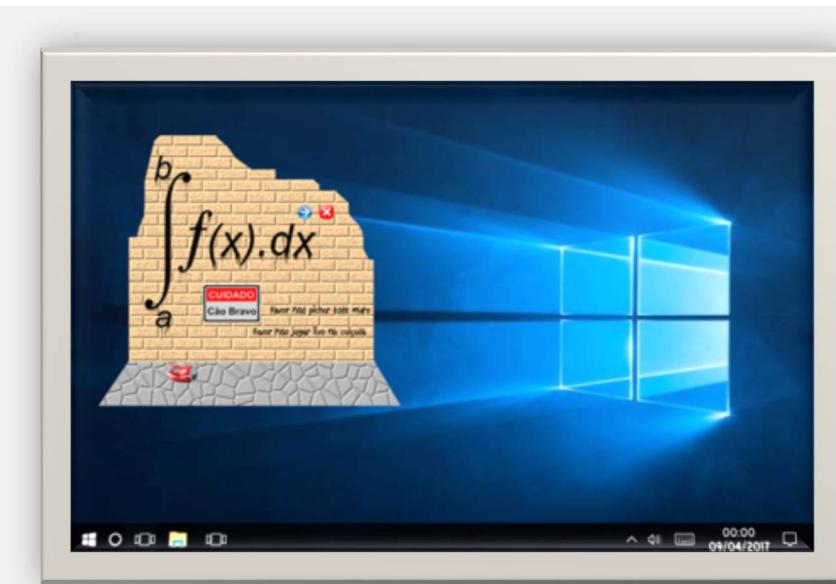
```
end;
```

```
procedure TForm1.SpeedButton8Click(Sender: TObject);
begin
```

```
  // Informa
```

```
  Form2.ShowModal();
```

```
end;
```



No programa exemplo, há um *About box* com uma fôrma irregular arrastável com som e outras coisa bregas. Use e descubra as várias habilidades disponíveis.

Assunto para PAW.

```
procedure TForm1.SpinEdit1Exit(Sender: TObject);
begin
```

```
  // Ao sair do campo SpinEdit1, executar automaticamente SpeedButton3Click para redimensionar a grade
  SpeedButton3.Click;
```

```
end;
```

```

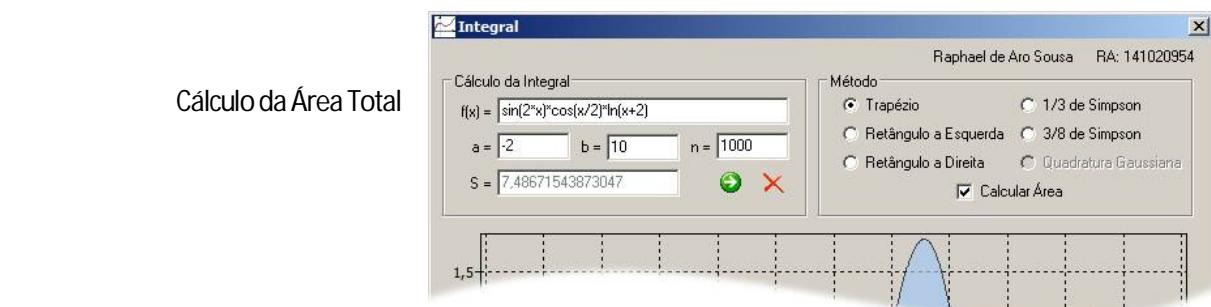
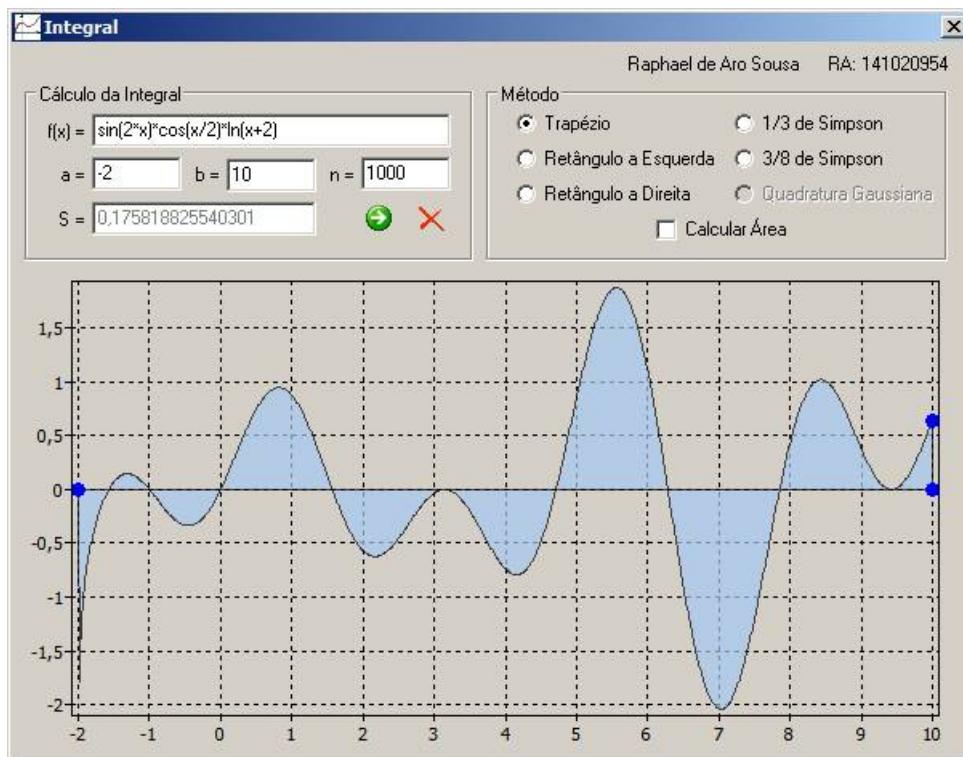
procedure TForm1.StringGrid1PrepareCanvas(sender: TObject; aCol, aRow: Integer;
  aState: TGridDrawState);
begin
  // Ao desenhar a grade, deixar células editáveis brancas e não editáveis cinzas
  with StringGrid1 do
    if (aCol > 0) and (aRow > n+1) then
      Canvas.Brush.Color := $00F0F0F0;
end;

procedure TForm1.StringGrid1SelRectCell1(Sender: TObject; aCol, aRow: Integer;
  var CanSelect: Boolean);
begin
  // Não permitir edição nas células não editáveis cinzas
  if RadioButon2.Checked then
    Exit;
  if ARow > n+1 then
    CanSelect := False;
end;

```

E assim, você terminou de desenvolver o seu programa para cálculo de Integrais numéricas. Parabéns.

Caso queira um programa que calcule Integrais Numéricas somente para entrada de dados através de funções, poderá elaborar um baseado nas figuras que seguem. Este exemplo de programa foi elaborado por *Raphael de Aro Sousa* no primeiro semestre de 2015.



O programa encontra-se à disposição no endereço já conhecido, na seção Programas Exemplo, no arquivo M6-Integrais Numéricas-B.rar.

## Cálculo de Pesos e Pontos de Gauss

Caso seja de seu interesse, segue um programa para calcular pesos e pontos de Gauss para diferentes números de pontos.

Crie um projeto Lazarus, salve com o nome wtGauss.lpi, dimensione a fôrma com Height=240, Width=376, BorderStyle=bsDialog, Caption=Cálculo de (w,t) de Gauss, Font.Name=Tahoma, Font.Size=8 e Position=poDesktopCenter. Utilize o menu Projeto, Opções de Projeto e preencha Título com Cálculo de (w,t) de Gauss e carregue o ícone MNC-Integrals.ico.

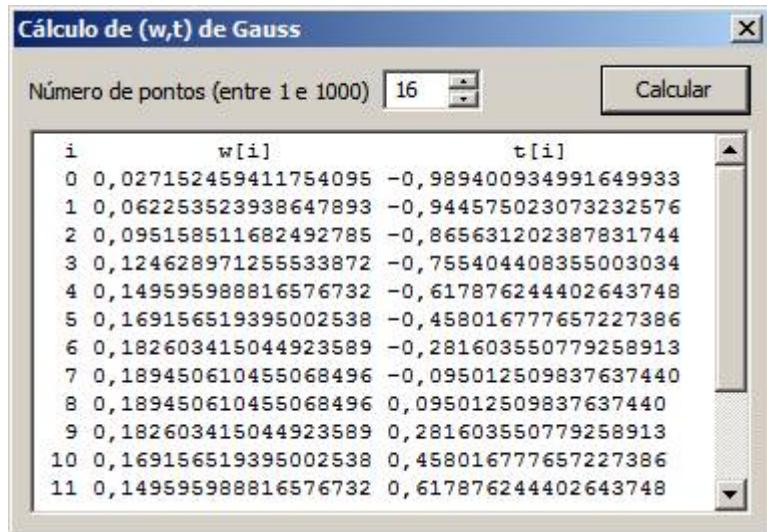
Lance um TLabel (Label1) com Left=8, Top=14 e Caption=Número de pontos (entre 1 e 1000).

Lance um TSpinEdit (SpinEdit1) com Left=184, Top=10, MaxValue=1000, MinValue=1 e Value=16.

Lance um TButton (Button1) com Left=293, Top=8, Caption=Calcular e dê um toque duplo no botão para criar a rotina Button1Click, que será executada quando ocorrer o evento OnClick no botão.

Lance um TMemo (Memo1) com Left=8, Top=40, Height=192, Width=360, Font.Name=Courier New, Font.size=8 e ScrollBars=ssAutoVertical.

Vai ficar com esta cara.



A seguir, introduza algumas constantes, tipos e variáveis, crie uma rotina GaussLegendre para calcular os pesos e pontos de Gauss-Legendre e preencha a rotina Button1Click que lerá o valor do SpinEdit1, executará a rotina GaussLegendre e preencherá o Memo1. Só isto.

E como fazer? Basta copiar da listagem que segue. E depois é só calcular os seus pesos e pontos para tabelas com 1, 2, 3, ..., 1000 pontos. Divirta-se.

O programa encontra-se à disposição, na seção Programas Exemplo, no arquivo M6-Integrals Numéricas-C.rar.

```

unit Uni t1;

{$mode obj fpc}{$H+}

interface

uses
  Classes, SysUtils, FileUtil, Forms, Controls, Graphics, Dialogs, StdCtrls,
  Spin;

```

```

type
  { TForm1 }
  TForm1 = class(TForm)
    Button1: TButton;
    Label1: TLabel;
    Memo1: TMemo;
    SpinEdit1: TSpinEdit;
    procedure Button1Click(Sender: TObject);
  private
    { private declarations }
  public
    { public declarations }
  end;

var
  Form1: TForm1;

implementation
{$R *.lfm}

{ TForm1 }

const
  nMaximo = 1000; // Compatível com SpinEdit1.MaxValue
  x1 = -1.0; // Pode ser modificado para diferentes conversões de
  x2 = 1.0; // intervalos de integração (e.g: x1 = 0, x2 = 1)

type
  VetorExtended = array [1 .. nMaximo] of Extended;

var
  i, n: Integer;
  w, t: VetorExtended;

procedure GaussLegendre(x1, x2: Extended; var w, t: VetorExtended; n: Integer);
const
  Eps = 1.0E-18; // Precisão ajustável conforme máquina, compilador, ...
var
  m, j, i: integer;
  z1, z, xm, xl, pp, p3, p2, p1: Extended;
begin
  m := (n+1) div 2;
  xm := 0.5*(x2+x1);
  xl := 0.5*(x2-x1);
  for i := 1 to m do
  begin
    z := Cos(Pi * (i - 0.25) / (n+0.5));
    repeat
      p1 := 1.0;
      p2 := 0.0;
      for j := 1 to n do
      begin
        p3 := p2;
        p2 := p1;
        p1 := ((2.0*j - 1.0)*z*p2 - (j - 1.0)*p3)/j;
      end;
      pp := n*(z*p1 - p2)/(z*z - 1.0);
      z1 := z;
      z := z1 - p1/pp;
    until (Abs(z-z1) <= Eps);
    t[i] := xm - xl * z;
    t[n+1-i] := xm + xl * z;
    w[i] := 2.0*xl / ((1.0 - z*z)*pp*pp);
    w[n+1-i] := w[i];
  end;
end;

procedure TForm1.Button1Click(Sender: TObject);
begin
  n := SpinEdit1.Value;
  Memo1.Clear;
  GaussLegendre(x1, x2, w, t, n);
  Memo1.Lines.Add(' i w[i] t[i]');
  for i := 1 to n do
    Memo1.Lines.Add(Format('%3d',[i-1])+' '+FloatToStrF(w[i],ffFixed,24,22)+' '+FloatToStrF(t[i],ffFixed,24,22));
  end;
end.

```

## 13- Sistemas de equações não-lineares

Em um país, cujo nome me nego a informar, em uma região geográfica, que pode ser qualquer uma delas, em uma cidade qualquer, o esgoto é coletado em um grande reservatório e, uma vez por dia, o conteúdo do reservatório é lançado em uma lagoa. A concentração de poluente, que depende do tempo, é decrescente e dada pela equação empírica internacionalmente conhecida, criada pela OMS (Organização Mundial da Sujeira), como sendo  $C(t) = 70.e^{\beta \cdot t} + 20.e^{\omega \cdot t}$ . A ONG (Organização Nacional Gastronômica), preocupada com o odor da lagoa interferindo na ocupação dos restaurantes da cidade, resolveu calcular os valores de  $\beta$  e  $\omega$  para a lagoa em questão para verificar quanto tempo leva desde o lançamento do esgoto até que seja suportável frequentar os restaurantes associados à entidade que deve ser após a concentração de poluente reduzir para 8 ou menos. Para isto, efetuou medições em dois momentos e os resultados estão na tabela abaixo da fotografia da lagoa.



t	1	2
C(t)	27,5702	17,6567

Após determinar os valores de  $\beta$  e  $\omega$  para a descarga de esgoto, a ONG solicitará que o lançamento de esgoto seja em horário tal que a partir das 19h a concentração de poluente seja aceitável.

Inventado este belo problema, só resta resolvê-lo. Não, não é nada disto. Antes da solução é necessário o modelo matemático. Depois disto, é só resolvê-lo.

As variáveis  $\beta$  e  $\omega$  são letras bonitinhos, mas é melhor voltar ao velho costume de chamar os elementos de um vetor com nomes iguais, variando os índices. Então,  $\beta = x_1$  e  $\omega = x_2$ .

Para  $t = 1$ , tem-se  $C(1) = 70.e^{1.x_1} + 20.e^{1.x_2} = 27,5702$ .

Para  $t = 2$ , tem-se  $C(2) = 70.e^{2.x_1} + 20.e^{2.x_2} = 17,6567$ .

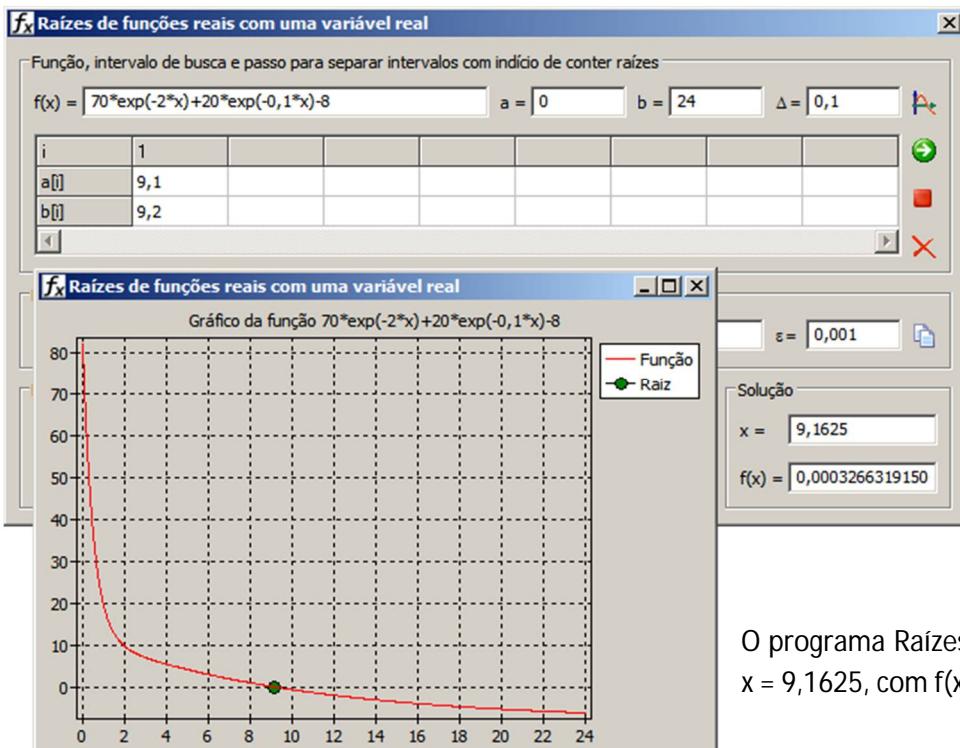
Então, tem-se o sistema de equações não-lineares  $\begin{cases} 70.e^{1.x_1} + 20.e^{1.x_2} = 27,5702 \\ 70.e^{2.x_1} + 20.e^{2.x_2} = 17,6567 \end{cases}$ .

A solução é  $x_1 = -2$  e  $x_2 = -0,1$ , com  $C(1) = 27,570218$  e  $C(2) = 17,656710$  e erro, medido em C, menor que  $2.10^{-5}$ .

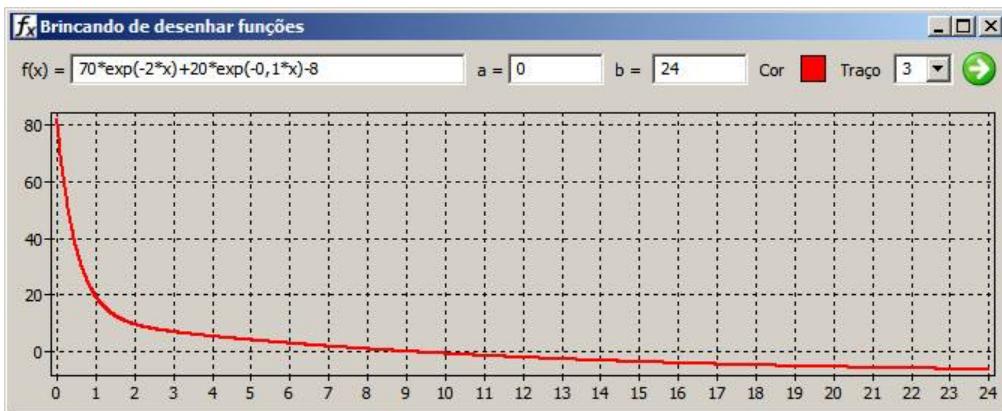
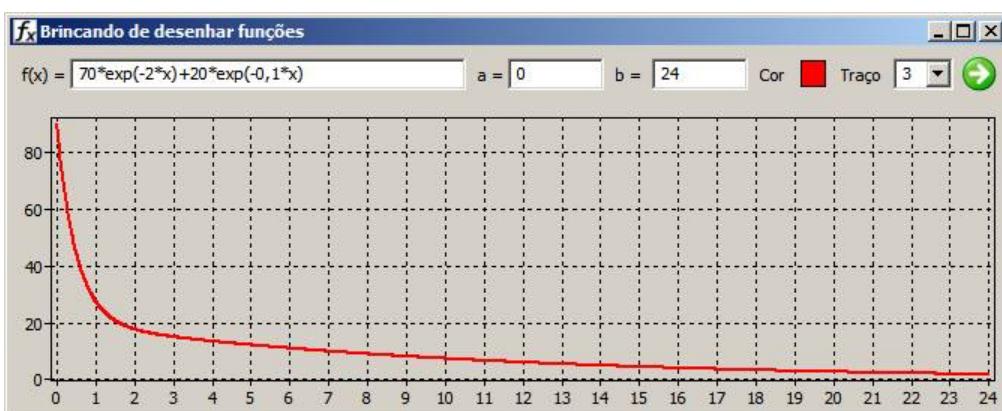
E assim foi resolvido o problema de sistemas de equações não-lineares. Mas o problema da ONG vai além disto. Se às 19h a concentração de poluição deve ser menor ou igual a 8, deseja-se saber qual o horário limite para o lançamento do esgoto. Eis um problema de calcular a raiz de apenas uma equação não-linear. E a equação para esta lagoa é  $C(t) = 70.e^{-2.t} + 20.e^{-0,1.t} = 8$ . Ou, com a notação usual para cálculo de raízes,  $f(x) = 70.e^{-2.x} + 20.e^{-0,1.x} - 8 = 0$ . Resolvendo por qualquer método conhecido, obtém-se  $x = 9,1625$ .

A ONG estimou que com  $t = 9,1625$ ,  $C(t)$  será 8 e que, portanto, o horário limite para descarga do fetido fluido na lagoa deverá ser até às 9h50. Assim, até às 19h, o aroma inconfundível do esgoto estará tal que não será perceptível nos restaurantes da região do lago e os turistas que frequentam estes restaurantes poderão se deliciar com os fabulosos pratos oferecidos, sentindo o aroma da comida e não, o odor do esgoto. Bom isto, não é? E, por fim, deve-se lembrar de que sujar é muito fácil e limpar, nem tanto. Mas isto é outra história.

Para ver a curva gerada e para conferir os resultados, a ONG utilizou os programas que seguem. Que bom que você também fez programas como estes.



O programa Raízes oferece o resultado  
 $x = 9,1625$ , com  $f(x) = 0,000327$ .



O Programa da Patricinha mostra que a equação  $f(x) = 8$ , para o ponto 9,1625 ou mostra que a equação  $f(x)-8 = 0$ , para o ponto 9,1625. Bastar usar o zoom do programa ou, até, o zoom do texto.

Mas como calcular as raízes do sistema não-linear? Voltando aos MNC, isto é apresentado a seguir.

## Equações lineares x equações não-lineares

As equações lineares são do tipo  $a_1 \cdot x_1 + a_2 \cdot x_2 + \dots + a_n \cdot x_n$ , ou seja, cada variável é multiplicada apenas por uma constante.

As equações não-lineares têm suas variáveis envolvidas em qualquer tipo de relação funcional, como

$x_1^2, x_2^{-2}, x_1 \cdot x_2, \frac{\sqrt{x_2}}{x_3}, \operatorname{sen}(x_1), e^{x_1+x_3}$  ou qualquer outra que não tenha a forma  $a_1 \cdot x_1 + a_2 \cdot x_2 + \dots + a_n \cdot x_n$ .

Um sistema de  $n$  equações e  $n$  variáveis é chamado sistema de equações não-lineares se, ao menos uma das equações for não-linear.

Assim, um sistema de equações não-lineares, com as constantes do membro direito levadas ao membro esquerdo, é escrito como

$$\begin{cases} f_1(x_1, x_2, x_3, \dots, x_n) = 0 \\ f_2(x_1, x_2, x_3, \dots, x_n) = 0 \\ \vdots \\ f_n(x_1, x_2, x_3, \dots, x_n) = 0 \end{cases} \text{ ou, como } \begin{cases} f_1(X) = 0 \\ f_2(X) = 0 \\ \vdots \\ f_n(X) = 0 \end{cases}, \text{ ou seja, com termo independente nulo, onde } X = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}.$$

$$\text{Utilizando notação vetorial, tem-se } F(X) = 0, \text{ com } X = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} \text{ e } F(X) = \begin{pmatrix} f_1(X) \\ f_2(X) \\ \vdots \\ f_n(X) \end{pmatrix}.$$

O vetor  $X^*$  que satisfaz  $F(X^*) = 0$  é raiz do sistema de equações não-lineares.

Como exemplos, tem-se:

$$1 \quad \begin{cases} x_1^2 + x_2^2 = 4 \\ x_1^2 - x_2^2 = 1 \end{cases}$$

Escrevendo na forma  $F(X) = 0$ , tem-se

$$\begin{cases} f_1(X) = x_1^2 + x_2^2 - 4 = 0 \\ f_2(X) = x_1^2 - x_2^2 - 1 = 0 \end{cases}$$

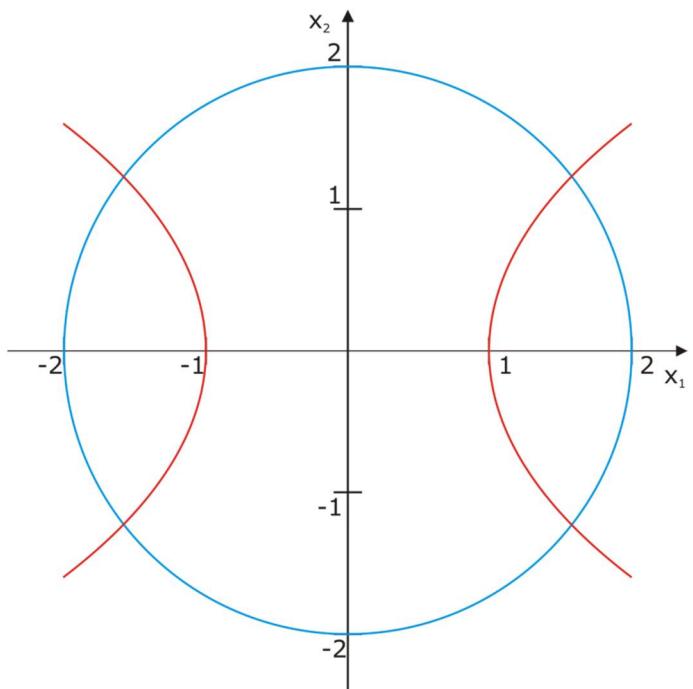
Este sistema não-linear admite quatro soluções que são os pontos onde  $f_1(X)$  e  $f_2(X)$  se interceptam.

As soluções são:

$$X = \begin{pmatrix} a \\ b \end{pmatrix}, X = \begin{pmatrix} a \\ -b \end{pmatrix}, X = \begin{pmatrix} -a \\ b \end{pmatrix}, X = \begin{pmatrix} -a \\ -b \end{pmatrix}$$

$$a = 1,58113883022206$$

$$b = 1,22474487139159$$

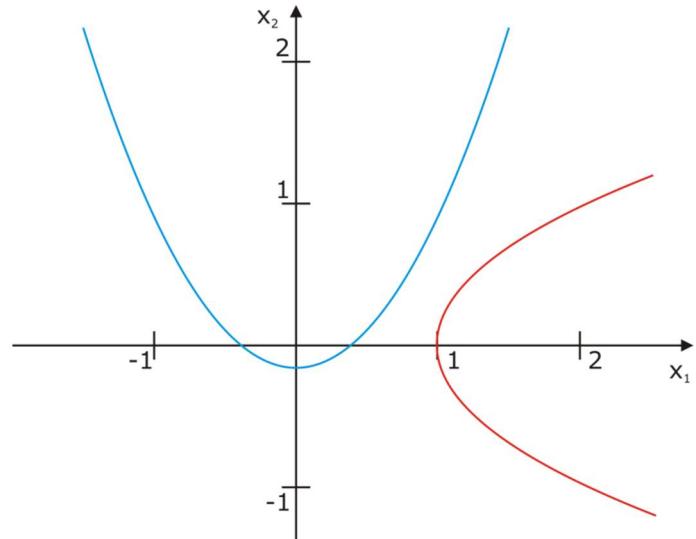


$$2 \quad \begin{cases} x_1^2 - x_2 = 0,2 \\ x_1 - x_2^2 = 1 \end{cases}$$

Escrevendo na forma  $F(X) = 0$ , tem-se

$$\begin{cases} f_1(X) = x_1^2 - x_2 - 0,2 = 0 \\ f_2(X) = x_1 - x_2^2 - 1 = 0 \end{cases}$$

Este sistema não-linear não tem soluções, pois não existem pontos onde  $f_1(X)$  e  $f_2(X)$  se interceptam.

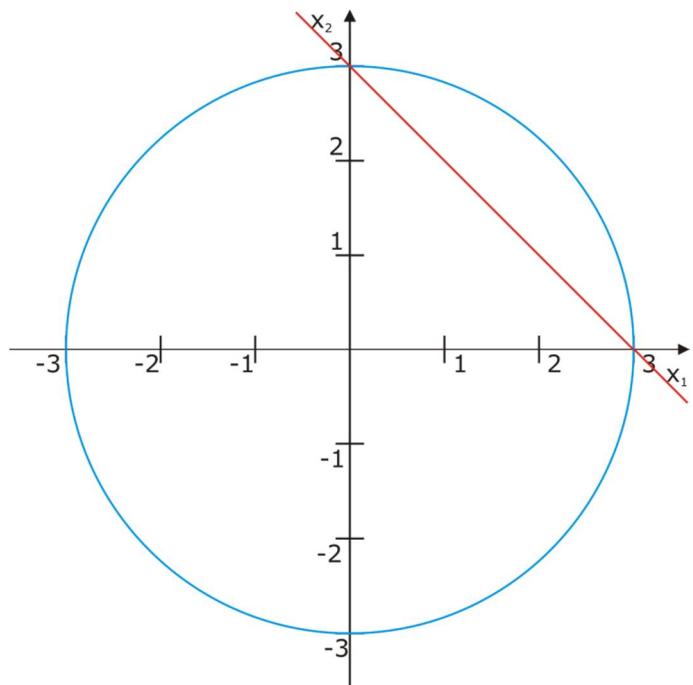


$$3 \quad \begin{cases} x_1^2 + x_2^2 = 9 \\ x_1 + x_2 = 3 \end{cases}$$

Escrevendo na forma  $F(X) = 0$ , tem-se

$$\begin{cases} f_1(X) = x_1^2 + x_2^2 - 9 = 0 \\ f_2(X) = x_1 + x_2 - 3 = 0 \end{cases}$$

Este sistema não-linear admite duas soluções que são os pontos onde  $f_1(X)$  e  $f_2(X)$  se interceptam.



As soluções são:

$$X = \begin{pmatrix} 3 \\ 0 \end{pmatrix} \text{ e } X = \begin{pmatrix} 0 \\ 3 \end{pmatrix}$$

O sistema é não-linear, já que ao menos uma das equações é não-linear.

## Método de Newton

Lá vem o Newton novamente. E vem com alguma novidade? Sim e não. O método é o mesmo, mas estendido para o  $\mathbb{R}^n$ .

Para encontrar a raiz de uma função  $f: \mathbb{R} \rightarrow \mathbb{R}$ , é apresentado, no Capítulo Raízes de funções de uma variável, o método que segue.

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}, \text{ que deve ser recalculado até que } |x_{k+1} - x_k| \leq \varepsilon \text{ ou até que } |x_{k+1} - x_k| / |x_{k+1}| \leq \varepsilon.$$

Não deixe nenhum matemático ortodoxo ver isto, mas generalizando para funções  $f: \mathbb{R}^n \rightarrow \mathbb{R}$ , tem-se:

$$X^{k+1} = X^k - \frac{F(X^k)}{F'(X^k)}, \text{ onde } F'(X^k) \text{ é o Jacobiano de } F(X), J(X), \text{ na iteração } k.$$

Espero que nenhum matemático cartesiano tenha visto.

Como  $J(X)$  é uma matriz e como dizem que é pecado dividir por matriz, vou começar tudo novamente.

Para  $f: R \rightarrow R$ , tem-se  $x_{k+1} = x_k - f'(x_k)^{-1} \cdot f(x_k)$

Agora os cartesianos podem ver.

Generalizando para funções  $f: R^n \rightarrow R$ , tem-se  $X^{k+1} = X^k - F'(X^k)^{-1} \cdot F(X^k)$ .

Então, o método é como o já conhecido método para  $f: R \rightarrow R$ , ou seja,  $X^{k+1} = X^k - J(X^k)^{-1} \cdot F(X^k)$ , que deve ser recalculado até que o critério de parada  $\|X^{k+1} - X^k\| \leq \varepsilon$  ou  $\|X^{k+1} - X^k\| / \|X^{k+1}\| \leq \varepsilon$  seja atendido.

Assim, dada uma aproximação inicial  $X^0 = (x_1, x_2, \dots, x_n)^T$ , as aproximações seguintes são obtidas com  $X^{k+1} = X^k - J(X^k)^{-1} \cdot F(X^k)$ , para  $k = 0, 1, 2, \dots$

A matriz Jacobiana de  $F(X^k)$ , já conhecida desde o Capítulo Derivadas, para  $n$  funções com  $X \in R^n$ , é

$$Jf(X^k) = \begin{bmatrix} \nabla^T f_1(X^k) \\ \vdots \\ \nabla^T f_n(X^k) \end{bmatrix} = \begin{bmatrix} \frac{\partial}{\partial x_1} f_1(X^k) & \frac{\partial}{\partial x_2} f_1(X^k) & \dots & \frac{\partial}{\partial x_n} f_1(X^k) \\ \frac{\partial}{\partial x_1} f_2(X^k) & \frac{\partial}{\partial x_2} f_2(X^k) & \dots & \frac{\partial}{\partial x_n} f_2(X^k) \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial}{\partial x_1} f_n(X^k) & \frac{\partial}{\partial x_2} f_n(X^k) & \dots & \frac{\partial}{\partial x_n} f_n(X^k) \end{bmatrix}, \text{ para } F(X^k) = \begin{pmatrix} f_1(X^k) \\ f_2(X^k) \\ \vdots \\ f_n(X^k) \end{pmatrix}.$$

Para resolver iterativamente  $X^{k+1} = X^k - J(X^k)^{-1} \cdot F(X^k)$  é necessário inverter  $J(X^k)$ , que um processo moroso ou, de alto custo computacional, como se diz por aí. Mas pode-se contornar a situação.

Para  $X^{k+1} = X^k - J(X^k)^{-1} \cdot F(X^k)$ , pode-se fazer  $h^k = -J(X^k)^{-1} \cdot F(X^k)$ . Então,  $X^{k+1} = X^k + h^k$  e  $J(X^k) \cdot h^k = -F(X^k)$ .

Desta forma, pode-se repetir o procedimento  $J(X^k) \cdot h^k = -F(X^k)$ ,  $X^{k+1} = X^k + h^k$ , para  $k = 1, 2, 3, \dots$  até que o critério de parada seja atendido. O critério pode ser  $\|X^{k+1} - X^k\| \leq \varepsilon$  ou  $\|X^{k+1} - X^k\| / \|X^{k+1}\| \leq \varepsilon$  ou  $\|X^{k+1} - X^k\| / \max\{\|X^{k+1}\|, 1\} \leq \varepsilon$ .

Além do critério de parada, é conveniente que se interrompa o procedimento após um número pré-determinado de iterações, para não deixar o seu computador calcular tresloucadamente por horas, dias, meses ou anos, algo que não seja atingível.

O método deve convergir se obedecidas as seguintes condições:

- 1- As funções  $f_i(X)$ , com  $i = 1, 2, \dots, n$  e as suas derivadas, até segunda ordem, devem ser contínuas na vizinhança que contém a raiz do sistema.
- 2- O determinante de  $J(X)$  deve ser não nulo.
- 3 A estimativa inicial  $X^0$  deve ser suficientemente próxima da solução do sistema.

Tal como apresentado no Método de Newton para solução de raízes de funções com uma variável, este método para  $n$  funções com  $n$  variáveis, a dedução pode ser obtida através de expansão de  $F$  por Série de Taylor. Isto é apresentado para  $f: \mathbb{R} \rightarrow \mathbb{R}$ , no Capítulo Raízes de funções de uma variável.

Um exemplo para treinar as munhecas. Pegue sua calculadora e prossiga.

$$\begin{cases} x_1^2 + x_2^2 = 4 \\ x_1^2 - x_2^2 = 1 \\ f_1(X) = x_1^2 + x_2^2 - 4 = 0 \\ f_2(X) = x_1^2 - x_2^2 - 1 = 0 \end{cases}$$

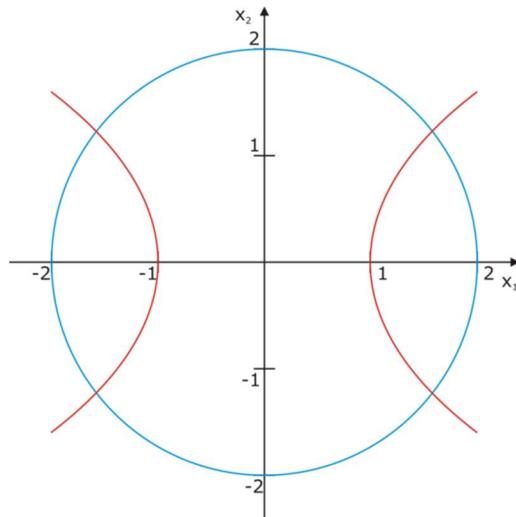
O sistema admite quatro soluções.

As soluções são:

$$X = \begin{pmatrix} a \\ b \end{pmatrix}, X = \begin{pmatrix} a \\ -b \end{pmatrix}, X = \begin{pmatrix} -a \\ b \end{pmatrix}, X = \begin{pmatrix} -a \\ -b \end{pmatrix}$$

$$a = 1,58113883022206$$

$$b = 1,22474487139159$$



Como o método é iterativo e convergente, para diferentes pontos de partida, obtêm-se diferentes soluções. Considere a solução inicial  $X = (2 \ 2)^T$ ,  $\epsilon = 0,01$  e, o esperado, é obter a solução  $X = (a \ b)^T$ .

O Jacobiano é dado por  $J(X) = \begin{bmatrix} 2 \cdot x_1 & 2 \cdot x_2 \\ 2 \cdot x_1 & -2 \cdot x_2 \end{bmatrix}$  e  $h^k$  é calculado através do sistema  $J(X^k) \cdot h^k = -F(X^k)$ .

Cuidado com o sinal

$k$	$X^k$	$F(X^k)$	$J(X^k)$	$-F(X^k)$	$h^k$	$X^{k+1}$	$\  X^{k+1} - X^k \ $
1	$\begin{pmatrix} 2 \\ 2 \end{pmatrix}$	$\begin{pmatrix} 4 \\ -1 \end{pmatrix}$	$\begin{bmatrix} 4 & 4 \\ 4 & -4 \end{bmatrix}$	$\begin{pmatrix} -4 \\ 1 \end{pmatrix}$	$\begin{pmatrix} -0,375 \\ -0,625 \end{pmatrix}$	$\begin{pmatrix} 1,625 \\ 1,375 \end{pmatrix}$	0,728869
2	$\begin{pmatrix} 1,625 \\ 1,375 \end{pmatrix}$	$\begin{pmatrix} 0,53125 \\ -0,25 \end{pmatrix}$	$\begin{bmatrix} 3,25 & 2,75 \\ 3,25 & -2,75 \end{bmatrix}$	$\begin{pmatrix} -0,53125 \\ 0,25 \end{pmatrix}$	$\begin{pmatrix} 0,043269 \\ 0,142045 \end{pmatrix}$	$\begin{pmatrix} 1,581731 \\ 1,232955 \end{pmatrix}$	0,148490
3	$\begin{pmatrix} 1,581731 \\ 1,232955 \end{pmatrix}$	$\begin{pmatrix} 0,022049 \\ -0,018305 \end{pmatrix}$	$\begin{bmatrix} 3,163462 & 2,465909 \\ 3,163462 & -2,465909 \end{bmatrix}$	$\begin{pmatrix} -0,022049 \\ 0,018305 \end{pmatrix}$	$\begin{pmatrix} -0,000592 \\ -0,008182 \end{pmatrix}$	$\begin{pmatrix} 1,581139 \\ 1,224772 \end{pmatrix}$	0,008204

Considere a solução inicial  $X = (-2 \ -2)^T$ ,  $\epsilon = 0,01$  e, o esperado, é obter a solução  $X = (-a \ -b)^T$ .

$k$	$X^k$	$F(X^k)$	$J(X^k)$	$-F(X^k)$	$h^k$	$X^{k+1}$	$\  X^{k+1} - X^k \ $
1	$\begin{pmatrix} -2 \\ -2 \end{pmatrix}$	$\begin{pmatrix} -4 \\ 1 \end{pmatrix}$	$\begin{bmatrix} -4 & -4 \\ -4 & 4 \end{bmatrix}$	$\begin{pmatrix} 4 \\ -1 \end{pmatrix}$	$\begin{pmatrix} 0,625 \\ 0,375 \end{pmatrix}$	$\begin{pmatrix} -1,375 \\ -1,625 \end{pmatrix}$	0,728869
2	$\begin{pmatrix} -1,375 \\ -1,625 \end{pmatrix}$	$\begin{pmatrix} 0,53125 \\ -1,75 \end{pmatrix}$	$\begin{bmatrix} -2,75 & -3,25 \\ -2,75 & 3,25 \end{bmatrix}$	$\begin{pmatrix} -0,53125 \\ 1,75 \end{pmatrix}$	$\begin{pmatrix} -0,221591 \\ 0,350962 \end{pmatrix}$	$\begin{pmatrix} -1,596591 \\ -1,274038 \end{pmatrix}$	0,415062
3	$\begin{pmatrix} -1,596591 \\ -1,274038 \end{pmatrix}$	$\begin{pmatrix} 0,172277 \\ -0,074071 \end{pmatrix}$	$\begin{bmatrix} -3,193182 & -2,548077 \\ -3,193182 & 2,548077 \end{bmatrix}$	$\begin{pmatrix} -0,172277 \\ 0,074071 \end{pmatrix}$	$\begin{pmatrix} 0,015377 \\ 0,048340 \end{pmatrix}$	$\begin{pmatrix} -1,581214 \\ -1,225698 \end{pmatrix}$	0,050727
4	$\begin{pmatrix} -1,581214 \\ -1,225698 \end{pmatrix}$	$\begin{pmatrix} 0,002573 \\ -0,002100 \end{pmatrix}$	$\begin{bmatrix} -3,162427 & -2,451397 \\ -3,162427 & 2,451397 \end{bmatrix}$	$\begin{pmatrix} -0,002573 \\ 0,002100 \end{pmatrix}$	$\begin{pmatrix} 0,000074 \\ 0,000074 \end{pmatrix}$	$\begin{pmatrix} -1,581139 \\ -1,225624 \end{pmatrix}$	0,000106

Para treinar as munhecas um pouco mais, pode-se partir de  $(2 -2)^T$  e de  $(-2 2)^T$  e as soluções esperadas são  $(a -b)^T$  e  $(-a b)^T$ .

Para treinar, mais ainda, pode-se tentar resolver o segundo exemplo, partindo de  $(1 1)^T$  e, depois de 5437 iterações, decidir parar, pois não tem solução.

$$\text{O segundo exemplo é } \begin{cases} x_1^2 - x_2 = 0,2 \\ x_1 - x_2^2 = 1 \end{cases}.$$

E treinando mais um pouco, pode-se resolver o terceiro exemplo, partindo de  $(2 1)^T$  para encontrar a solução  $(3 0)^T$  e partindo de  $(1 2)^T$ , para encontrar a solução  $(0 3)^T$ .

$$\text{O terceiro exemplo é } \begin{cases} x_1^2 + x_2^2 = 9 \\ x_1 + x_2 = 3 \end{cases}.$$

## Método de Newton Modificado

É claro que você já sabe que isto é para economizar nos cálculos das derivadas necessárias para calcular o Jacobiano.

A forma mais simples e, também grosseira, é calcular o Jacobiano apenas uma vez, para  $X^0$ , ou seja,  $J(X^0)$  e utilizar ao longo de todas as iterações.

Uma forma menos rude é recalcular o Jacobiano a cada t iterações utilizá-lo nas iterações seguintes, até uma nova iteração múltipla de t.

Uma boa ideia é resolver o sistema  $J(X^k).h^k = -F(X^k)$  com decomposição L.U. No caso do cálculo do Jacobiano apenas para  $X^0$ , após a decomposição, calculam-se os sistemas das várias iterações, com os procedimentos  $L.y = -F(X^0)$  e  $U.h^k = y$ .

No caso de se calcular o Jacobiano a cada t iterações o procedimento é o mesmo, para as iterações seguintes até uma iteração múltipla de t.

Segue exemplo, calculando o Jacobiano apenas para  $X^0$  e sem utilizar decomposição L.U. Pura preguiça.

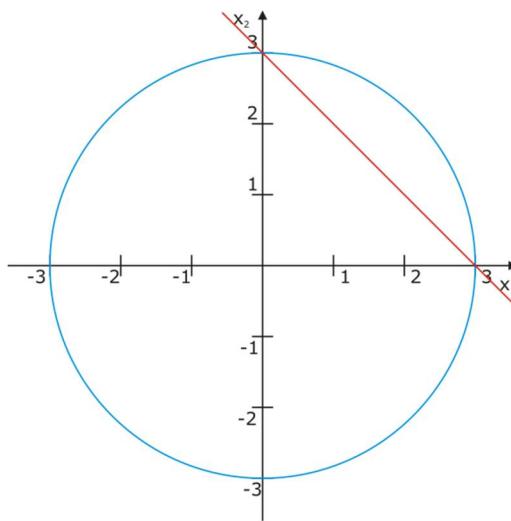
$$\begin{cases} x_1^2 + x_2^2 = 9 \\ x_1 + x_2 = 3 \\ f_1(X) = x_1^2 + x_2^2 - 9 = 0 \\ f_2(X) = x_1 + x_2 - 3 = 0 \end{cases}$$

O sistema admite duas soluções.

As soluções são:

$$X = \begin{pmatrix} 3 \\ 0 \end{pmatrix}, X = \begin{pmatrix} 0 \\ 3 \end{pmatrix}$$

O ponto de partida é  $(1 5)^T$ .



O Jacobiano é dado por  $J(X) = \begin{bmatrix} 2x_1 & 2x_2 \\ 1 & 1 \end{bmatrix}$  e é constante para todas as iterações, aplicado no ponto  $(1 5)^T$ .

Então, tem-se  $J(X) = J(X^0)$  para todas as iterações.

$$J(X) = \begin{bmatrix} 2 & 10 \\ 1 & 1 \end{bmatrix}$$

Considere  $\epsilon = 0,01$ .

k	$X^k$	$F(X^k)$	$J(X^k)$	$-F(X^k)$	$h^k$	$X^{k+1}$	$\ X^{k+1} - X^k\ $
1	$\begin{pmatrix} 1 \\ 5 \end{pmatrix}$	$\begin{pmatrix} 17 \\ 3 \end{pmatrix}$	$\begin{bmatrix} 2 & 10 \\ 1 & 1 \end{bmatrix}$	$\begin{pmatrix} -17 \\ -3 \end{pmatrix}$	$\begin{pmatrix} -1,625 \\ -1,375 \end{pmatrix}$	$\begin{pmatrix} -0,625 \\ 3,625 \end{pmatrix}$	2,128673
2	$\begin{pmatrix} 0,625 \\ 3,625 \end{pmatrix}$	$\begin{pmatrix} 4,53125 \\ 0 \end{pmatrix}$	$\begin{bmatrix} 2 & 10 \\ 1 & 1 \end{bmatrix}$	$\begin{pmatrix} -4,53125 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 0,566406 \\ -0,566406 \end{pmatrix}$	$\begin{pmatrix} -0,058594 \\ 3,058594 \end{pmatrix}$	0,801019
3	$\begin{pmatrix} -0,058594 \\ 3,058594 \end{pmatrix}$	$\begin{pmatrix} 0,358429 \\ 0 \end{pmatrix}$	$\begin{bmatrix} 2 & 10 \\ 1 & 1 \end{bmatrix}$	$\begin{pmatrix} -0,358429 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 0,044804 \\ -0,044804 \end{pmatrix}$	$\begin{pmatrix} -0,013790 \\ 3,013790 \end{pmatrix}$	0,063362
4	$\begin{pmatrix} -0,013790 \\ 3,013790 \end{pmatrix}$	$\begin{pmatrix} 0,083121 \\ 0 \end{pmatrix}$	$\begin{bmatrix} 2 & 10 \\ 1 & 1 \end{bmatrix}$	$\begin{pmatrix} -0,083121 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 0,010390 \\ -0,010390 \end{pmatrix}$	$\begin{pmatrix} -0,003400 \\ 3,003400 \end{pmatrix}$	0,014693
5	$\begin{pmatrix} -0,003400 \\ 3,003400 \end{pmatrix}$	$\begin{pmatrix} 0,020423 \\ 0 \end{pmatrix}$	$\begin{bmatrix} 2 & 10 \\ 1 & 1 \end{bmatrix}$	$\begin{pmatrix} -0,020423 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 0,002553 \\ -0,002553 \end{pmatrix}$	$\begin{pmatrix} -0,000847 \\ 3,000847 \end{pmatrix}$	0,003610

Com o Jacobiano constante, a convergência ocorreu com 5 iterações.

Utilizando o Método de Newton, que atualiza o Jacobiano a cada iteração, a convergência seria mais rápida? Boa pergunta. E a resposta é fácil. Basta você repetir os cálculos atualizando o Jacobiano.

Segue a solução, mas seria bom se você fizesse todos os cálculos, para treinar.

k	$X^k$	$F(X^k)$	$J(X^k)$	$-F(X^k)$	$h^k$	$X^{k+1}$	$\ X^{k+1} - X^k\ $
1	$\begin{pmatrix} 1 \\ 5 \end{pmatrix}$	$\begin{pmatrix} 17 \\ 3 \end{pmatrix}$	$\begin{bmatrix} 2 & 10 \\ 1 & 1 \end{bmatrix}$	$\begin{pmatrix} -17 \\ -3 \end{pmatrix}$	$\begin{pmatrix} -1,625 \\ -1,375 \end{pmatrix}$	$\begin{pmatrix} -0,625 \\ 3,625 \end{pmatrix}$	2,128673
2	$\begin{pmatrix} 0,625 \\ 3,625 \end{pmatrix}$	$\begin{pmatrix} 4,53125 \\ 0 \end{pmatrix}$	$\begin{bmatrix} -1,25 & 7,25 \\ 1 & 1 \end{bmatrix}$	$\begin{pmatrix} -4,53125 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 0,533088 \\ -0,533088 \end{pmatrix}$	$\begin{pmatrix} -0,091912 \\ 3,091912 \end{pmatrix}$	0,753901
3	$\begin{pmatrix} -0,091912 \\ 3,091912 \end{pmatrix}$	$\begin{pmatrix} 0,568366 \\ 0 \end{pmatrix}$	$\begin{bmatrix} -0,183824 & 6,183824 \\ 1 & 1 \end{bmatrix}$	$\begin{pmatrix} -0,568366 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 0,089258 \\ -0,089258 \end{pmatrix}$	$\begin{pmatrix} -0,002653 \\ 3,002653 \end{pmatrix}$	0,126230
4	$\begin{pmatrix} -0,002653 \\ 3,002653 \end{pmatrix}$	$\begin{pmatrix} 0,015934 \\ 0 \end{pmatrix}$	$\begin{bmatrix} -0,005307 & 6,005307 \\ 1 & 1 \end{bmatrix}$	$\begin{pmatrix} -0,015934 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 0,002651 \\ -0,002651 \end{pmatrix}$	$\begin{pmatrix} -0,000002 \\ 3,000002 \end{pmatrix}$	0,003749

Como era esperado, mas nem sempre ocorre, o Método de Newton, que calcula o Jacobiano a cada iteração, convergiu mais rapidamente que o Método de Newton Modificado, que reutiliza o Jacobiano de iterações anteriores.

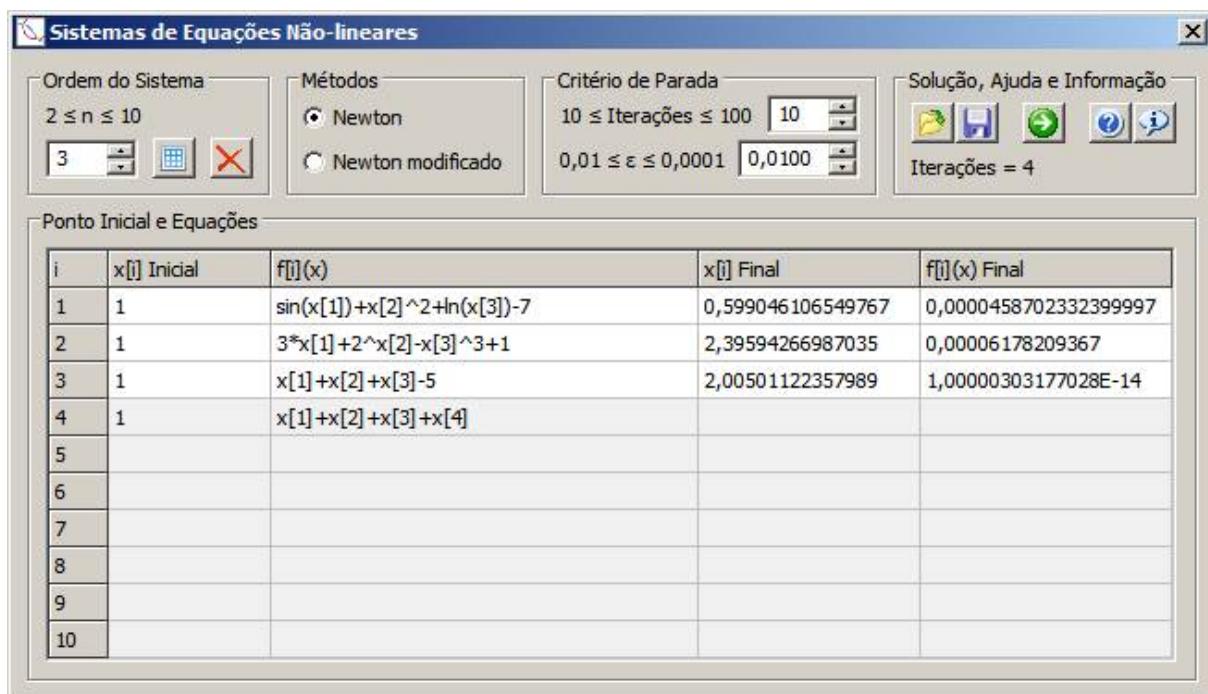
E agora, chegou a hora. Faça um simpático programa que resolva sistemas de equações não-lineares com esses dois métodos. Só isso.

### Sugestões para seus programas

Um simpático programa, para resolver sistemas de equações não-lineares deverá permitir a definição da ordem do sistema, a seleção do método, definição do critério de parada, entrada das equações e ponto inicial.

Além disto, o programa deve permitir que o usuário salve os dados digitados para uso posterior, pois a quantidade equações pode ser grande e se o usuário quiser utilizar as mesmas equações, com algumas alterações, para um novo cálculo, em outra ocasião, poderá carregar os dados que salvou anteriormente.

Um programa com as características descritas poderia ter a cara que segue.



Com este programa é possível:

- definir a ordem do sistema, ajustar a grade de entrada de dados e limpar todos os dados;
- escolher o método de cálculo;
- definir o critério de parada por iterações e por precisão;
- entrar com funções e ponto inicial;
- salvar e carregar problemas salvos;
- calcular a solução numérica do sistema.

### Instruções claras e precisas sobre trabalhos computacionais para avaliação

Todas as instruções necessárias estão descritas no mesmo item, do Capítulo 7. Copiar e colar é muito fácil, mas não é nada elegante, agradável, simpático ou conveniente repetir tudo aquilo que já está escrito em capítulo anterior. Além disto, se você já entregou algum trabalho, já sabe como deve proceder. Caso contrário, veja as instruções no Capítulo 7. E antes de entregar, leia as instruções em <http://sacoman.dco.fc.unesp.br/mnc/pet/>.

Se você optar em fazer um programa que tenha a aparência da figura apresentada ou similar, poderá continuar lendo para ter algumas explicações que poderão ser úteis para você.

### Um exemplo de programa

Minha ajuda será *apenas na fabricação da interface*. O resto é com você. Depois de pronto, limpe todo lixo gerado durante a elaboração do projeto, compacte com Zip, Rar ou qualquer outro compactador, com o sugestivo nome T7-RA-V1.ext, onde RA é o número de seu Registro Acadêmico, ext é a extensão fornecida pelo seu compactador, V1 é a primeira versão e entregue via endereço eletrônico de recepção que você já sabe qual é. Não sabe qual é? Então leia tudo sobre o assunto no endereço <http://sacoman.dco.fc.unesp.br/mnc/pet/>.

Então, mãos a obra.

Crie um diretório apropriado para desenvolver seu projeto (e.g.: X:\MNC-Trabalhos\T7). Organize seu ambiente de trabalho e crie subdiretórios necessários para o projeto, como, por exemplo, um diretório chamado Figuras, para as figuras dos botões e ícone e outro chamado Exemplo, para o exemplo executável que está à sua disposição.

Copie de <http://sacoman.dco.fc.unesp.br/mnc/material/> o arquivo M7-Sistemas de Equações Não-lineares.rar e descompacte. Mova SistemasNaoLineares.exe, Interpretador.dll e SistemasNaoLineares.pdf para o diretório exemplo, para executar quando quiser. Mova as figuras para o diretório Figuras.

Inicie seu ambiente de projeto, no meu caso, Lazarus, ajeite as janelas da forma como achar conveniente e salve o projeto com o nome SistemasNaoLineares.lpr e a unidade principal com o nome Principal. Pressione o botão Nova Fórmula e salve dando o nome Sobre para esta nova unidade.

## ÓPA. PERAÍ.

Esse programa você vai fazer sozinho. Com a cara deste programa da sugestão ou com outra cara qualquer. Mas fica para você mostrar suas habilidades de programador, sem copiar centenas de linhas de código e preenchendo apenas as rotinas referentes aos algoritmos. Este você fará tudo e sozinho. E lembre-se que eu confio em você.

Contudo, você pode ver o programa exemplo funcionando.

O programa encontra-se à disposição no endereço já conhecido, na seção Programas Exemplo, no arquivo M7-Sistemas de Equações Não-lineares.rar.

Execute o programa, leia o arquivo de ajuda e execute os exemplos anexos.

O arquivo de ajuda deste programa é um simples arquivo pdf com apenas uma página.

Acompanha um diretório chamado Exemplos, com 10 arquivos de texto que podem ser carregados no programa.

Divirta-se.

**Sistemas de Equações Não-lineares**

**Sistemas de Equações Não-lineares**

Ordem do Sistema		Métodos	Critério de Parada		Solução, Ajuda e Informação	
2 ≤ n ≤ 10	<input checked="" type="radio"/> Newton	10 ≤ Iterações ≤ 100	10	0,01 ≤ ε ≤ 0,0001	0,0100	
<input type="button" value="3"/>	<input type="radio"/> Newton modificado					Iterações = 4
<b>Ponto Inicial e Equações</b>						
i	x[i] Inicial	f[i](x)	x[i] Final	f[i](x) Final		
1	1	s[x[1]]+x[2]^2+ln(x[3])-7	0,599046106549767	0,000045870232399997		
2	1	3*x[1]+2*x[2]-x[3]^3+1	2,39594266987035	0,00006178209367		
3	1	x[1]+x[2]+x[3]-5	2,00501122357989	1,00000303177028E-14		
4	1	x[1]+x[2]+x[3]+x[4]				
5						
6						
7						
8						
9						
10						

Passe o apontador do mouse sobre os botões e sobre botões de rádio para ver suas funções.

Se você for fazer um programa semelhante, cuidado com o valor de  $\epsilon$ . O Epsilon apresentado é para teste de parada dos métodos de solução do sistema. O  $\epsilon$  para calcular as derivadas parciais é 0,0001 e o passo h inicia com 1000. $\epsilon$ .

Ao utilizar este programa, a grade é habilitada conforme o valor da ordem do sistema, mas ao redimensionar a ordem para um valor menor, os dados não são apagados. Isto é útil, caso o usuário queira testar um sistema com uma quantidade de equações e depois com uma quantidade menor e, a seguir, voltar ao valor inicial.

Para apagar, existe botão específico.

Junto com o programa há uma série de arquivos de teste, em formato texto, com o formato que segue.

- 1- Ordem do sistema (uma linha com o valor n)
- 2- Linhas com funções (n linhas)
- 3- Linhas com o ponto inicial (n linhas com n componentes do vetor inicial)

Os exemplos são variados e com ordem 2 até 10, que é o limite deste programa exemplo.

Se houver solução, é apresentada nas duas últimas colunas, apresentando x e f(x). A solução será tanto melhor quanto mais próxima de zero f(x) estiver.

Em geral, o método não modificado deverá apresentar valores melhores, pois calcula o Jacobiano a cada iteração e o método modificado utiliza o Jacobiano do ponto inicial para todas as iterações.

Não deixe de clicar no botão de informação sobre o programa. É realmente brega.

## 14- Equações Diferenciais Ordinárias



## 15- Dicas, truques e quebra-galhos para desenvolver programas

Os **DTQ-Dicas, truques e quebra-galhos** aparecem neste Capítulo na ordem em que são citados nos programas. Então, é conveniente ler os programas e, sempre que surgir um comentário no programa, informando que se trata de um DTQ, salte para este Capítulo, entenda o DTQ e volte para a leitura do programa. Mas nada e ninguém impedem a leitura dos DTQ sem ler os programas.

### **DTQ 1 - Escolhendo a Fonte para a fórmula, textos e controles em geral**

#### *Problema*

Quando se elabora um programa para ser utilizado em diferentes versões do Windows, os textos serão apresentados de forma diferente, para cada versão, se for utilizada a fonte padrão. Cada versão do Windows tem uma fonte padrão diferente e seus tamanhos são diferentes. Um programa executado no Windows XP poderá apresentar um texto com fonte MS Sans e, no Windows 10 com fonte Tahoma. No primeiro caso, o texto poderá ser totalmente visível e, no segundo, poderá extravasar o tamanho da fórmula.

#### *Solução*

Pode-se prevenir este problema de versões, projetando o programa, em qualquer versão do Windows, utilizando uma fonte diferente da fonte padrão e o programa apresentará os textos com o mesmo tamanho em diferentes versões do Windows. Uma sugestão agradável é projetar os programas utilizando fonte Tahoma, com tamanho 8.

### **DTQ 2 - Prevenindo cintilação quando a fórmula é atualizada constantemente**

#### *Problema*

Quando há uma atualização constante de textos ou outros componentes na fórmula, é comum que estes elementos fiquem cintilando, tremulando ou oscilando. Isto é comum quando, por exemplo, um Label se move pela fórmula, dentro de um laço, com determinados intervalos de tempo, controlados por um Timer ou não.

#### *Solução*

Pode-se prevenir ou diminuir a cintilação, utilizando-se um dos comandos que seguem, ou os dois.

```
procedure TForm1.FormShow(Sender: TObject);
begin
  ControlStyle := ControlStyle + [csOpaque];
  DoubleBuffered := True;
  ...
  ...
end;
```

### **DTQ 3 - Prevenindo que o programa congele por um longo tempo**

#### *Problema*

Quando existe um laço com comando **for**, **repeat** ou **while**, cujo tempo de processamento seja longo, é comum que o programa não responda até que o laço termine. Isto impede que se arraste a fórmula através do vídeo, que se utilize um componente do programa (e.g.: um botão), que se interrompa o programa ou, em casos extremos, que se utilizem outros programas simultaneamente.

### Solução

Pode-se prevenir este congelamento, processando as mensagens enviadas pelo Windows, dentro destes laços, como segue.

```
// Exemplo exagerado de laço infinito
while True do
begin
  ...
  ...
  Application.ProcessMessages; // Programa recebe e processa mensagens do Windows
  ...
  ...
end;

procedure TForm1.Button1Click(Sender: TObject);
begin
  // Durante ação do while, Close só será atendido se houver ProcessMessages dentro do laço
  Close;
end;
```

### DTQ 4 - Break não é pecado

**Pecar ou não pecar? GoTo nunca foi pecado. Break não é pecado. Halt é meio exagerado, mas não é pecado, mas o assunto agora é o Break**

#### Problema

Quando existe um laço com comando **for**, **repeat** ou **while**, pode parecer que a saída do laço ocorrerá apenas quando as condições de contorno do laço forem atendidas. Contudo, pode-se interromper o laço, em determinada situação, com o comando **Break**. Isto é muito mais claro e eficiente do que longas estruturas de programa que alguns consideram elegantes, mas não são. Pascal é elegante. As estruturas de programação em Pascal são elegantes. Mas Break existe, faz parte da linguagem e, consequentemente, pode e deve ser utilizado.

### Solução

Pode-se prevenir este congelamento, processando as mensagens enviadas pelo Windows, dentro destes laços, como segue.

```
// Exemplo exagerado de laço infinito
while True do
begin
  ...
  ...
  if AconteceAlgunaCoisaDeterminada then
    Break;
  ...
end;
```

### Comparação entre elegância pouco clara e eficiência muito clara

```
var
  a, b: Real;
  PodeSair := Boolean;

...
PodeSair := False;
repeat
  ...
  ... // Vários cálculos com a
  ... // Vários cálculos com b
  if AlgumaCoisaAcontece then
    PodeSair := True;
  ...
until (a > b) or PodeSair;
```

```
var
  a, b: Real;

repeat
  ...
  ... // Vários cálculos com a
  ... // Vários cálculos com b
  if AlgumaCoisaAcontece then
    Break;
  ...
until (a > b);
```

## DTQ 5 - Definindo as espessuras das barras de rolagem

### *Problema*

Quando se elabora um programa para ser utilizado em diferentes versões do Windows, as barras de rolagem serão apresentadas com espessuras diferentes. Com isto, um projeto bem elaborado em uma versão do Windows, poderá apresentar os componentes com folga após as barras de rolagem, ou não apresentar completamente o componente que contém a barra de rolagem.

### *Solução*

Pode-se prevenir este problema de versões, projetando o programa, em qualquer versão do Windows, utilizando ... ....

Foi amplamente explicado em aula. Mesmo assim, ainda estou escrevendo isto, para constar no texto. Fique frio.

## DTQ 6 - Executando arquivos externos

### *Problema*

Quando se deseja executar um arquivo externo ao programa, pode-se utilizar uma função já existente para esta tarefa. Isto é útil para abrir arquivos de Ajuda (chm, pdf, txt) ou executar programas (Calculadora, Bloco de notas).

### *Solução*

Pode-se utilizar a função OpenDocument, definida como segue:

```
function OpenDocument(APath: string): Boolean;
```

Para tanto, é necessário incluir LCLIntf na cláusula **uses** de **interface** ou de **implementation**.

APath é nome do arquivo que se deseja executar, incluindo o caminho. Para executar, por exemplo, o Arquivo de Ajuda do programa, chamado Ajuda.chm pode-se escrever a seguinte linha de código:

```
OpenDocument('Ajuda.chm');
```

Isto funcionará se o arquivo Ajuda.chm estiver no mesmo diretório do programa que contém o comando.

Caso se queira executar o mesmo arquivo, mas que esteja no diretório Ajuda\_do\_Programa, deve-se utilizar o comando que segue:

```
OpenDocument('Ajuda_do_Programa\Ajuda.chm');
```

Para verificar possíveis erros, o trecho que segue é o ideal:

```
if not OpenDocument('Interpretador.chm') then
```

```
  ShowMessage('Não foi possível abrir o arquivo de ajuda Interpretador.chm.');
```

Note que foi solicitado executar um arquivo e a função OpenDocument solicitará que o arquivo seja aberto e isto ocorrerá conforme a associação existente entre o tipo do arquivo e o programa padrão.

Por exemplo, txt com Bloco de notas, html com o navegador padrão e, assim por diante.

Para executar um programa, pode-se fazer como segue:

```
OpenDocument('Notepad.exe'); // Executará o Bloco de notas
```

```
OpenDocument('Calc.exe'); // Executará a Calculadora
```

Os comandos para executar Notepad.exe e Calc.exe funcionarão, pois estes programas encontram-se no diretório do sistema, que está definido na variável Path do sistema.

Para executar, por exemplo, o Word, é muito mais fácil solicitar que seja aberto um arquivo docx. Contudo se houver um desejo incontido de executar o Word, pode-se utilizar o código que segue:

```
if not OpenDocument('C:\Arquivos de programas\Microsoft Office\Office14\Word.exe') then
```

```
  ShowMessage('Não foi possível executar o Word.');
```

Meio insano fazer isto. Há formas melhores que são apresentadas em PAW. Curse PAW, é legaw.

Estou pensando recursivamente em escrever vários DTQ neste Capítulo. Se tiver sugestões, envie.

Veja meu pensamento recursivo na próxima página.



Pensamento Recursivo

Marco Antônio Rahal Sacoman - 1980, 1981, 1982, 1983, ...

Pensamento Recursivo  
Marco Antônio Rahal Sacoman  
1980

## DTQ Extra - Menu Irritantemente Piscante da versão 1.6.4 do Lazarus

### Problema

Quando não há projeto aberto, o menu do IDE do Lazarus fica piscando tresloucadamente. Inicie o Lazarus, utilize o Menu Arquivo, Fechar tudo e veja o menu piscando. Pode ser que você até goste disto, mas, no meu caso isto irrita e irrita muito.

### Solução

No diretório C:\Lazarus\ide faça uma cópia do arquivo SourceFileManager.pas, chamando, por exemplo, de SourceFileManager-Original-Menu-Piscante.pas. Abra o arquivo SourceFileManager.pas para edição. Pode ser usando o próprio Lazarus. Pressione F3 e procure o texto TSourceEditorTabCommandsStamp. Altere as linhas que seguem.

#### Como é

```
{ TSourceEditorTabCommandsStamp }

function TSourceEditorTabCommandsStamp.Changed(ASrcEdit: TSourceEditor): Boolean;
begin
  Result := not (
    (FSrcEdit := ASrcEdit) and
    (ASrcEdit^.IsLocked = ASrcEdit^.IsLocked) and
    (FSourceNotebook = ASrcEdit^.SourceNotebook) and
    (FPageIndex = ASrcEdit^.SourceNotebook^.PageIndex) and
    (FPageCount = ASrcEdit^.SourceNotebook^.PageCount));
  if not Result then Exit;

  FSrcEdit := ASrcEdit;
  if ASrcEdit^.IsLocked then
begin
  FSrcEdit^.IsLocked := ASrcEdit^.IsLocked;
  FSourceNotebook := ASrcEdit^.SourceNotebook;
  FPageIndex := ASrcEdit^.SourceNotebook^.PageIndex;
  FPageCount := ASrcEdit^.SourceNotebook^.PageCount;
end;
end;
```

#### Como deve ficar

```
{ TSourceEditorTabCommandsStamp }

function TSourceEditorTabCommandsStamp.Changed(ASrcEdit: TSourceEditor): Boolean;
begin
  Result := not (
    (FSrcEdit := ASrcEdit) and
    ((ASrcEdit^.IsLocked = True) or (
      (FSrcEdit^.IsLocked = ASrcEdit^.IsLocked))) and
    (FSourceNotebook = ASrcEdit^.SourceNotebook) and
    (FPageIndex = ASrcEdit^.SourceNotebook^.PageIndex) and
    (FPageCount = ASrcEdit^.SourceNotebook^.PageCount));
  if not Result then Exit;

  FSrcEdit := ASrcEdit;
  if ASrcEdit^.IsLocked then
begin
  FSrcEdit^.IsLocked := ASrcEdit^.IsLocked;
  FSourceNotebook := ASrcEdit^.SourceNotebook;
  FPageIndex := ASrcEdit^.SourceNotebook^.PageIndex;
  FPageCount := ASrcEdit^.SourceNotebook^.PageCount;
end;
end;
```

Logo a seguir, encontre TSourceEditorCommandsStamp. Altere as linhas que seguem.

**Como é**

```
{ TSourceEditorCommandsStamp }
```

```
function TSourceEditorCommandsStamp.Changed(ASrcEditor: TSourceEditor;
  ADisplayState: TDisplayState): Boolean;
begin
  Result := not (
    (FSrcEditor = ASrcEditor) and
    (ASrcEditor <> nil) and
    (FDisplayState = ADisplayState) and
    (FEditorComponentStamp = ASrcEditor.EditorComponent.ChangeStamp) and
    (FEditorCaretStamp = ASrcEditor.EditorComponent.CaretStamp));
end;

if not Result then Exit;

FSrcEditor := ASrcEditor;
FDisplayState := ADisplayState;
if ASrcEditor <> nil then
begin
  FEditorComponentStamp := ASrcEditor.EditorComponent.ChangeStamp;
  FEditorCaretStamp := ASrcEditor.EditorComponent.CaretStamp;
end;
end;
```

**Como deve ficar**

```
{ TSourceEditorCommandsStamp }
```

```
function TSourceEditorCommandsStamp.Changed(ASrcEditor: TSourceEditor;
  ADisplayState: TDisplayState): Boolean;
begin
  Result := not (
    (FSrcEditor = ASrcEditor) and
    ((ASrcEditor = nil) or (
      (FDisplayState = ADisplayState) and
      (FEditorComponentStamp = ASrcEditor.EditorComponent.ChangeStamp) and
      (FEditorCaretStamp = ASrcEditor.EditorComponent.CaretStamp)));
  );
end;

if not Result then Exit;

FSrcEditor := ASrcEditor;
FDisplayState := ADisplayState;
if ASrcEditor <> nil then
begin
  FEditorComponentStamp := ASrcEditor.EditorComponent.ChangeStamp;
  FEditorCaretStamp := ASrcEditor.EditorComponent.CaretStamp;
end;
end;
```

Salve e feche o arquivo SourceFileManager.pas. Se estiver editando no Lazarus, Salve e utilize o menu Arquivo, Fechar tudo. No menu do Lazarus, selecione Ferramentas, Construir o Lazarus com perfil: IDE Normal. Uma janela será apresentada solicitando confirmação. Responda SIM. O Lazarus vai recompilar o IDE, encerrará e iniciará novamente. O menu não estará mais piscando quando não houver arquivos carregados para edição no IDE.

Você acabou de recompilar o Lazarus. Parabéns. Você vai deixar qualquer professor orgulhoso de ter um aluno como você.

## 16- Seja curioso e leia isto agora

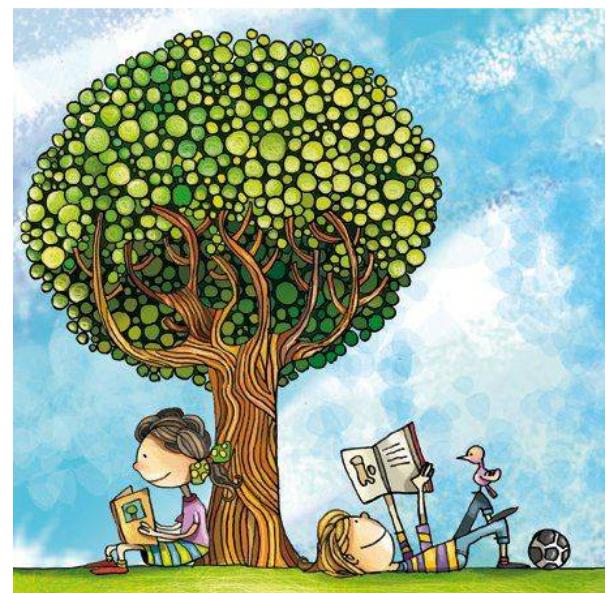
Este texto foi elaborado para utilização exclusiva dos alunos já citados exaustivamente no início deste texto.

Se você é um dos alunos para os quais o texto foi elaborado e encontrar erros, faça a sua parte e informe, para que seja corrigido e reeditado.

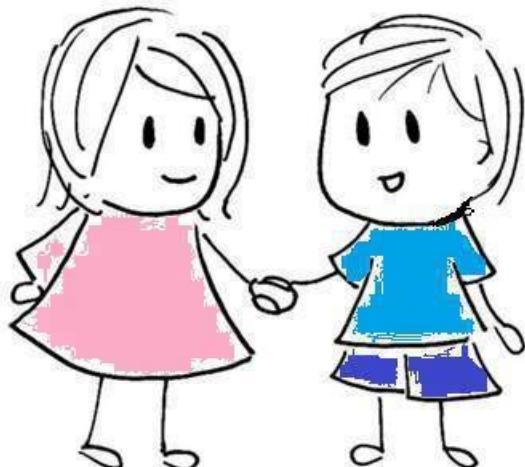
Se você **não** é um dos alunos para os quais o texto foi elaborado e encontrar erros, faça a sua parte, também, afinal nem deveria estar lendo o texto e informe, para que seja corrigido e reeditado.

Se você, sendo ou não, um dos alunos para os quais o texto foi elaborado e tiver o incontido desejo de colaborar, fale comigo para combinarmos o que poderá ser feito.

Lembre-se que, em geral, a árvore que lhe dá sombra foi plantada por outra pessoa que a deixou para você. Eu me esforço para praticar isto e espero que você, também.



EU ACREDITO  
QUE O SENTIDO  
DA VIDA  
SEJA FAZER SENTIDO  
A OUTRAS VIDAS!



Quando um homem planta árvores sob cuja sombra sabe que nunca haverá de sentar-se, começou então a entender o sentido da vida.



Estava indo tudo bem até que o Homer chegou.

FIM



Ligue os pontos

- Leonard Hofstadter ①
- Sheldon Cooper ②
- Penny ③
- Howard Wolowitz ④
- Rajesh Koothrappali ⑤
- Bernadette Rostenkowski ⑥
- Amy Farrah Fowler ⑦
- Stuart Bloom ⑧



- ① Interessante, mas preciso arranjar uma namorada.
- ② Quantas letrinhas σ e δ estranhas que esses nerds usam.
- ③ Muito legal. Acho que vou ajudar a encontrar erros no texto.
- ④ Ajudar? Minha existência já é uma ajuda para a humanidade.
- ⑤ Apropriado. Exercita os neurônios. Mas prefiro me exercitar com o Shelly.
- ⑥ Até eu que não tenho PhD consigo encontrar dezenas de erros neste texto.
- ⑦ Aposto que o meu Howard vai encontrar mais erros do que todos os outros.
- ⑧ Ficam brincando de MNC e nunca se lembram de mim.

# Fim Final

E para terminar com o Fim Final, Final Mesmo, segue a seção Odeio Corrupção. Você não precisa ler.

Um colóquio entre Patricinha e Joãozinho.

Patricinha

Joãozinho, na frase *Eu sou Petista*, qual o tipo de sujeito?

Joãozinho

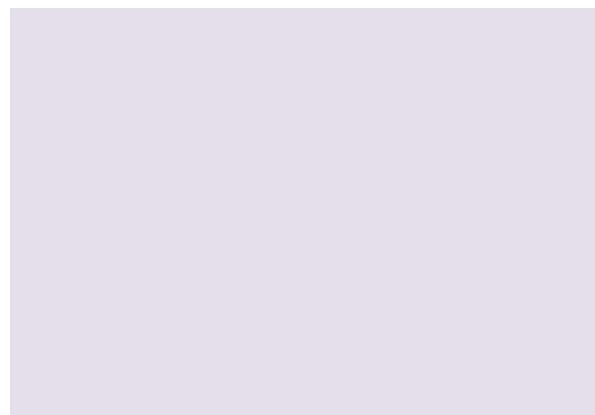
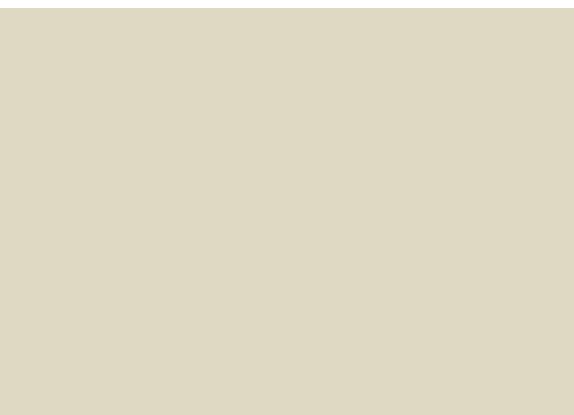
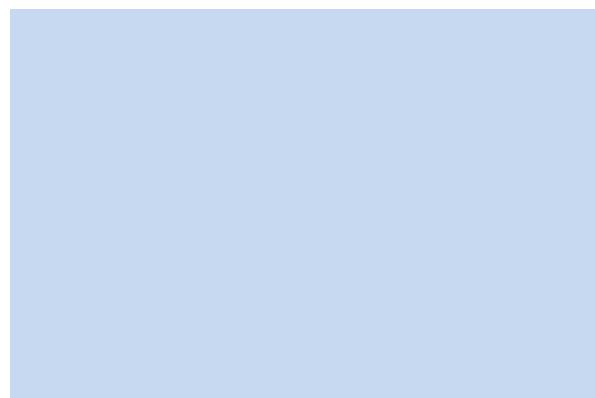
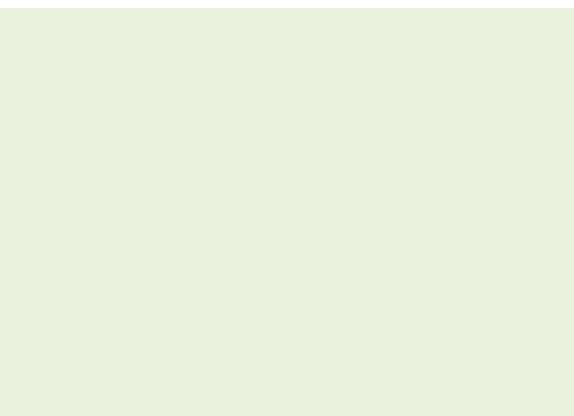
Depende. Se for sujeito simples, é LADRÃO. Se for sujeito composto, é QUADRILHA. Se for sujeito oculto, é LARANJA. Mas se for sujeito indeterminado, é ONG BENEFICIADA.

Patricinha

E se for sujeito inexistente?

Joãozinho

Aí é PATRIMÔNIO DO LULA, né, Patricinha?



## Fim Final, Final Mesmo

Se você é PeTista e, fiel que é, não gostou, lembre-se que você não precisa utilizar este texto para cursar a disciplina.

Além disto, se tiver um incontido desejo de ver um gracejo sobre outros partidos envolvidos em falso-catas, envie seu gracejo e tenha a certeza de que publicarei. Tem quadradinhos coloridos para todas as cores de partidos. Envie seu texto e a cor preferida.

Entenda que este é um espaço absolutamente democrático para denunciar as mazelas que este e outros ParTidos proporcionaram à nossa amada pátria.

