

Universidade Federal de Santa Catarina
Centro Tecnológico
Departamento de Informática e Estatística
***LIVITEC – Laboratório de Informática
para Vigilância Tecnológica***

**CÁLCULO NUMÉRICO
EM COMPUTADORES
*-Provas e Projetos-***

Volume 2

*Bernardo Gonçalves Riso
Mirela Sechi Moretti Annoni Notare*

Florianópolis, SC, 2011

SUMÁRIO

VOLUME 1

Apresentação	3
--------------	---

PARTE I – PROVAS

PROVA 1 - Sistemas de Numeração	6
PROVA 2 - Representação em Ponto Flutuante	12
PROVA 3 – Erros	20
PROVA 4 - Introdução às Equações Algébricas e Transcendentes	24
PROVA 5 - Método da Bissecção	32
PROVA 6 - Método da Falsa-Posição	39
PROVA 7 - Método de Newton-Raphson	46
PROVA 8 - Introdução às Equações Polinomiais	53
PROVA 9 - Método de Birge-Vieta	62
PROVA 10 - Método de Müller	74
PROVA 11 - Introdução aos Sistemas de Equações Lineares	84
PROVA 12 - Método de Eliminação Gaussiana	89
PROVA 13 - Método de Fatoração LU	97
PROVA 14 - Método Iterativo de Gauss-Seidel	109
PROVA 15 - Introdução aos Sistemas Não-Lineares	116
PROVA 16 - Método de Newton	121

PARTE II – PROJETOS

PROJETO 1 - Programa Mudança de Base	131
PROJETO 2 – Programa Leitura de Registro	133
PROJETO 3 – Programa Cálculo de Erros	137
PROJETO 4 – Programa Valores Funcionais	139
PROJETO 5 - Programa Bissecção	142
PROJETO 6 – Programa Falsa_Posição	144
PROJETO 7 – Programa Newton_Raphson	146
PROJETO 8 – Programa Descartes	148
PROJETO 9 – Programa Birge_Vieta	150
PROJETO 10 – Programa Müller	153

VOLUME 2

Sumário	158
Agradecimentos	160
Apresentação	161

PARTE I – PROVAS	174
------------------	-----

PROVA 17 - Método da Secante	175
PROVA 18 - Método Quase Newton	184
PROVA 19 - Introdução ao Ajustamento de Curvas	191

PROVA 20 - Ajuste Polinomial	200
PROVA 21 - Ajuste Não Polinomial	209
PROVA 22 - Interpolação Lagrangeana	217
PROVA 23 - Interpolação de Gregory-Newton	225
PROVA 24 - Integração de Newton-Côtes	232
PROVA 25 - Integração de Simpson	241
 PARTE II – PROJETOS	 250
PROJETO 11 - Programa Sistema Linear	251
PROJETO 12 - Programa Gauss	255
PROJETO 13 - Programa LU	257
PROJETO 14 - Programa Gauss-Seidel	260
PROJETO 15 - Programa Não Linear	263
PROJETO 16 - Programa Newton	266
PROJETO 17 - Programa Quase_Newton	270
PROJETO 18 - Programa Tabela	273
PROJETO 19 - Programa Ajuste_de_Parábola	277
PROJETO 20 - Programa Ajuste_Exponencial	281
PROJETO 21 - Programa Interpolação_Linear	285
PROJETO 22 - Programa Lagrange	287
PROJETO 23 - Programa Trapézios	290
PROJETO 24 - Programa Simpson_1/3	293
PROJETO 25 - Programa Gauss_Dois_Pontos	296
PROJETO 26 - Programa Runge_Kutta_4	297
PROJETO 27 - Programa Relaxação	298

AGRADECIMENTOS

Os autores desejam agradecer ao senhor *Itamar Annoni Notare* pelo empenho na organização dos capítulos deste segundo volume de Cálculo Numérico em Computadores. Sem sua contribuição o texto ora apresentado não poderia ter o mesmo bom gosto na distribuição das matérias que compõem esta monografia.

APRESENTAÇÃO

Nesta apresentação do segundo volume do livro “*Cálculo Numérico em Computadores – Provas e Projetos*” discorreremos sobre um conjunto de resultados de uma pesquisa de natureza científica e educacional sobre aspectos relacionados à investigação de *metodologias* para:

- (a) o estabelecimento de um modelo de *provas* de avaliação de conhecimento; e
- (b) a elaboração de *projetos* computacionais na área de cálculo numérico.

Em tal contexto, estudamos as propostas nesse sentido já apresentadas na literatura, fazemos a crítica dessas propostas e desenvolvemos o que acreditamos ser uma *nova concepção metodológica* para a abordagem dessa área de conhecimento. Essa nova concepção determina uma *estrutura original para as provas de domínio de conteúdo* de cálculo numérico computacional (que são submetidas a alunos de graduação das áreas técnicas e científicas), além de um *procedimento sistemático para a criação de algoritmos estruturados*. As provas, os algoritmos e os correspondentes programas computacionais devem constituir um livro que apresente características originais se comparado aos textos disponíveis atualmente. Para melhor estruturar esta apresentação, dividimos os assuntos em seções numeradas e tituladas.

Na *seção 1* desta apresentação preparamos uma introdução. Na *seção 2*, fazemos uma rápida revisão de cinco livros didáticos da área de cálculo numérico, em busca de indicações de modelos de provas e projetos. Na *seção seguinte*, a *seção 3*, desenvolvemos uma metodologia para a definição da estrutura e confecção de questões de provas para disciplinas de Cálculo Numérico em Computadores, assim como para a forma de apresentação das respostas a essas questões. Na última seção, a *seção 4*, definimos uma metodologia para a realização de projetos que resultem em programas computacionais didáticos do cálculo numérico. Seguem conclusões preliminares desse esforço, na *seção 5*, e algumas referências bibliográficas, na *seção 6*.

1. Introdução. A disciplina *Cálculo Numérico em Computadores* reúne, em seu conteúdo programático, uma série de métodos numéricos adequados à resolução de problemas oriundos da modelagem matemática de fenômenos que são de interesse em várias áreas da ciência e da técnica. Em sala de aula, uma vez estudado um método numérico, e discutidas as principais características desse método, realizam-se exemplos de sua utilização em casos simples e ilustrativos. O passo seguinte do procedimento didático, ainda em sala de aula, consiste no desenvolvimento de algoritmos que generalizam e automatizam o emprego de tal método.

Após uma série de aulas com o estudo de vários métodos numéricos, estudo esse realizado de acordo com a sistemática apresentada no parágrafo anterior, aplica-se uma *prova parcial* para avaliar o conhecimento dos alunos. Essa prova tem as seguintes características gerais:

- (a) trata-se de uma prova com *consulta livre* ao material que o aluno traz de casa; e

(b) apresenta *quatro questões de igual valor*, sendo a primeira de natureza dissertativa, a segunda e a terceira de natureza numérica, e, na última, pede-se para o aluno construir um algoritmo.

A investigação de natureza científica sobre procedimentos sistemáticos convenientes à elaboração e resolução de provas como as descritas acima, e ao desenvolvimento de projetos computacionais que programem algoritmos, envolve a definição de *metodologias* que dêem a essa investigação um suporte teórico. No que se refere às provas, é preciso, portanto, que haja uma metodologia para a sua elaboração que atenda às necessidades de clareza na redação das questões, de objetividade e de varredura dos temas submetidos à avaliação. Já, no que se refere aos algoritmos, é indispensável que se tenha uma metodologia para a sua construção, de modo que atenda a, pelo menos, três requisitos: simplicidade, eficiência e sistematicidade.

No livro “*Provas e Projetos de Cálculo Numérico em Computadores*”, em fase de desenvolvimento e que, uma vez pronto, se pretende submeter à Editora Universitária, apresenta-se uma proposta resultante de um trabalho de pesquisa científica sobre a melhor maneira de:

- (a) confeccionar e resolver provas de cálculo numérico computacional; e
- (b) de desenvolver projetos para a construção e execução de programas numéricos de modo que se obtenha, no final do processo de desenvolvimento, um produto de natureza didática, legível e facilmente analisável.

Os livros didáticos, normalmente encontráveis no contexto universitário do ensino de disciplinas da área de cálculo numérico, não costumam trazer *modelos de provas nem projetos de implementação de programas computacionais que sigam uma metodologia de refinamentos sucessivos*. Não há, de nosso conhecimento, nenhum livro dessa área que proponha modelos de provas. Alguns livros apresentam algoritmos, mas somente uns poucos apresentam programas computacionais correspondentes aos métodos numéricos.

2. Revisão Bibliográfica. Nesta seção fazemos uma análise preliminar de cinco livros didáticos, devotados à área de cálculo numérico, em *busca de sugestões de modelos de provas e de modos de confecção de projetos de programas computacionais*. Os livros considerados são os seguintes:

- [1] “*Cálculo Numérico – Características Matemáticas e Computacionais dos Métodos Numéricos*”, cujos autores são: Décio Sperandio, João Teixeira Mendes e Luiz Henry Monken e Silva;
- [2] “*Programação e Métodos Computacionais*”, em dois volumes, de autoria de Tércio Pacitti e Cyril P. Atkinson;
- [3] “*Cálculo Numérico com Estudos de Casos em Fortran IV*”, de William S. Dorn e Daniel D. McCracken, traduzido do inglês por José Abel Royo dos Santos e Ana Lúcia Serio dos Santos;
- [4] “*Algoritmos Numéricos – Sequenciais e Paralelos*”, que tem a autoria de Bernardo Gonçalves Riso, Christianne Marie Schweitzer e Gastón Pedro Alauzet Heerdt; e

[5] “*Cálculo Numérico em Computadores – Provas e Projetos*”, de autoria de Bernardo Gonçalves Riso e Mirela Sechi Moretti Annoni Notare.

No primeiro livro relacionado, o livro [1], temos um excelente texto didático, moderno, bem organizado e que apresenta interessantíssimas discussões sobre a aplicabilidade dos métodos numéricos. É um livro muitíssimo bem confeccionado. São aspectos positivos desse livro a qualidade do texto, dos exemplos, das figuras, das tabelas, e a preocupação com o rigor matemático das expressões e das deduções, além da apresentação de teoremas e suas demonstrações. Ao final de cada capítulo, sugere ao leitor uma grande quantidade de exercícios. Na página 3, esse livro apresenta a seguinte definição de algoritmo:

É a descrição seqüencial dos passos que caracterizam um método numérico. O algoritmo fornece uma descrição completa de operações bem definidas por meio das quais o conjunto de dados de entrada é transformado em dados de saída. Por operações bem definidas entendem-se as aritméticas e lógicas que um computador pode realizar. Dessa forma, um algoritmo consiste de uma seqüência de n passos, o algoritmo deve fornecer valores ao menos ‘próximos’ daqueles que são procurados. O número n pode não ser conhecido *a priori*. É o caso de algoritmos iterativos cuja idéia será enfocada a seguir. Nesse caso, em geral tem-se para n apenas uma cota superior.

Não há, contudo, nesse extraordinário livro, modelos de provas para a verificação do conhecimento adquirido pelos estudantes. Também não há projetos de algoritmos ou de programas computacionais.

Já, no livro [2] encontramos, no primeiro volume, uma bem desenvolvida apresentação das características da linguagem FORTRAN, e dos modos de utilizá-la. Trata-se, portanto, podemos dizer, de um excelente manual de programação FORTRAN de aplicação geral, não especificamente de métodos numéricos. O segundo volume, entretanto, volta-se para a apresentação de métodos numéricos. Nesse volume apresentam-se numerosos algoritmos e os programas computacionais em FORTRAN que lhes são correspondentes. Os algoritmos têm a forma de fluxogramas, e seguem um “sumário computacional” que indica as principais etapas da resolução de problemas. Em seguida à apresentação de um algoritmo, exibe-se o programa FORTRAN associado ao algoritmo, assim como o relatório de uma execução desse programa para um particular conjunto de dados. A nosso ver, os programas são excessivamente complexos para um iniciante, pois utilizam recursos avançados de programação, além de sub-rotinas, e elaborados formatos para a apresentação de resultados.

Pelo que podemos observar, não há nesse livro (pioneiro da computação numérica no Brasil), apesar de todos os méritos de que é merecedor, nenhuma sugestão sobre a elaboração de provas para avaliar o conhecimento dos leitores. Faz-se, contudo, a realização de projetos computacionais, mas com o emprego de um estilo criticável sob o ponto de vista da evolução do processo de desenvolvimento. Pois não se tem, propriamente, um procedimento gradual, estruturado e progressivo, de refinamentos sucessivos de uma idéia inicial, que permita acompanhar o longo e difícil raciocínio que culmina com o detalhamento de um algoritmo e a realização do seu programa.

O livro [3] é, provavelmente, um caso único de texto de cálculo numérico em que se mostra a aplicação de métodos numéricos em situações práticas do mundo real, ou muito próximo dele. Os métodos numéricos são apresentados em detalhe e suas características são bem discutidas. Além disso, apresenta “grafos de processos” para a aplicação desses métodos e algoritmos na forma de fluxogramas. As listagens dos programas FORTRAN são então exibidas, assim como os relatórios de suas execuções. Infelizmente, também aqui não há sugestões sobre a elaboração de provas para averiguar o aprendizado dos estudantes. Mas há, podemos dizer, o desenvolvimento de numerosos projetos computacionais com a realização de algoritmos e de programas numéricos, embora esses algoritmos e programas sejam apresentados sem um desejável processo de desenvolvimento progressivo que venha facilitar o entendimento por parte de iniciantes no assunto.

Diferentemente dos livros mencionados anteriormente, o livro [4] preocupa-se em estabelecer uma metodologia de refinamentos sucessivos para a criação de algoritmos numéricos didáticos. Nesse livro propõe-se conceber um algoritmo em basicamente três etapas de refinamentos sucessivos. Cada etapa apresenta duas versões para o algoritmo: uma versão gráfica; e uma versão literal. Na *primeira etapa*, faz-se uma representação gráfica do algoritmo na forma de uma caixa preta com aberturas para o recebimento de dados e para a emissão de resultados, e uma representação literal, em pseudo-linguagem de programação, na qual se definem os tipos de algumas variáveis e já se esboça o corpo do algoritmo. Na *segunda etapa* consideram-se aspectos internos da caixa preta definida na etapa anterior e faz-se a representação literal correspondente. Finalmente, na *última etapa* a metodologia faz estabelecer a representação gráfica e a correspondente representação literal do algoritmo completamente detalhado. Nesse livro não se propõem modelos para a realização de provas de conhecimento, mas sugere-se, de algum modo, um procedimento sistemático para o projeto computacional de métodos numéricos.

O livro [5], por sua vez, é o resultado de uma investigação que visa o estabelecimento de um modelo de *provas* e de *projetos* computacionais na área de cálculo numérico, apresentando nível de complexidade supostamente compatível com um primeiro estudo universitário nessa área. Sugere a submissão de provas com consulta ao material selecionado livremente pelo aluno, aproximando esse aluno de situações que poderá encontrar durante sua vida profissional. Além disso, propõe que as provas devam ter quatro questões de igual valor, sendo uma dissertativa, duas numéricas e um algoritmo. Essas questões precisam ser respondidas de modo detalhado e a adequação dos resultados deve ser brevemente comentada para que o aluno demonstre consciência do que se passa. Os algoritmos necessitam ser construídos paulatinamente, seguindo uma sistemática baseada em refinamentos sucessivos.

Um aspecto importante a levar em conta é que os textos tradicionais, isto é, os livros didáticos de cálculo numérico são elaborados por professores - de altíssimo gabarito técnico, que escrevem para seus leitores, estudantes universitários. Os livros [5] e aquele que estamos agora escrevendo têm uma característica diferente. Eles se colocam na *posição do estudante que responde* a um conjunto de questões de prova (estas, sim, elaboradas por um professor) e desenvolve projetos computacionais (também propostos pelo mestre). Mas a perspectiva é diferente daquela dos textos didáticos clássicos: enquanto lá se escreve do professor para o estudante, em [5] e no livro que agora se

escreve em continuação a [5], supostamente é o estudante quem escreve textos para ser lidos e avaliados pelo professor.

Em resumo, podemos dizer que o livro que estamos atualmente escrevendo procura dar continuidade ao livro [5], respeitando todas as suas consideradas boas características. A contribuição que esses livros trazem - o livro [5] e o livro que ora escrevemos - é a de apresentar uma *nova abordagem* para o estudo, para a avaliação e para a criação de projetos que envolvem o desenvolvimento de programas do cálculo numérico.

3. Metodologia para a confecção de provas. O sistema de avaliação prevê a realização de *quatro provas* distribuídas ao longo de um período letivo. Cada prova vale dez pontos e a nota final é a média aritmética das notas dessas quatro provas. O cabeçalho de cada prova anuncia o tema específico que é avaliado. Oferecem-se quatro questões ao estudante.

A *primeira* questão é de natureza dissertativa, pois pede a descrição resumida de um determinado assunto ligado ao tema da prova; a *segunda* e a *terceira* pedem a aplicação de algum método numérico (com o auxílio, possivelmente, de uma calculadora científica) para a resolução de problemas, e são, por esse motivo, qualificadas como questões numéricas; e, finalmente, a *quarta* questão, caracterizada como algoritmo, procura verificar a desenvoltura do estudante no que diz respeito à elaboração de algoritmos numéricos simples e de caráter didático, descritos com utilização de uma pseudolinguagem de programação, e referente a um método numérico incluído no âmbito da prova. Como exemplo de uma prova com as características descritas acima, tem-se:

PROVA 17 – Método Quase-Newton

FOLHA DE QUESTÕES:

Questão 1 (descritiva) – Por favor, considere o problema da resolução numérica de *Sistemas de Equações Não-Lineares*. Apresente um Método Quase Newton adequado à resolução desse tipo de sistema. Compare as características desse método com as do Método de Newton. Não deixe de escrever, pelo menos, uma página. Se achar conveniente, use fórmulas, gráficos e tabelas para ilustrar a sua exposição.

Questão 2 (numérica) – Use um *Método Quase-Newton* para tentar resolver o sistema dado a seguir:

$$\begin{cases} x + 2y - 2,1 = 0 \\ 3x^2 + y^2 - 6,9 = 0 \end{cases}$$

Adote $x = y = 0,5$ como estimativa inicial da solução do sistema. Faça duas iterações, indicando sempre todas as operações efetuadas. Para apresentar os resultados organizadamente, preencha uma tabela com o cabeçalho:

i	x_i	y_i	$\text{desvio}_i = \text{abs}(x_{i+1} - x_i) + \text{abs}(y_{i+1} - y_i)$
---	-------	-------	---

Por favor, no final, escreva um comentário que avalie a evolução do processo iterativo.

Questão 3 (numérica) – Tente resolver o sistema de equações não lineares dado abaixo com a aplicação de um *Método Quase Newton*:

$$\begin{cases} F(x, y) = 2x - 2y - 2,1 = 0 \\ G(x, y) = x^3 + 3y^2 - 6,9 = 0 \end{cases}$$

Adote $x = y = 0,5$ como estimativa inicial da solução do sistema. Faça três iterações sempre indicando as operações efetuadas. Preencha uma tabela que apresente os resultados organizadamente. Por favor, não deixe de escrever um comentário que avalie o processo iterativo.

Questão 4 (algoritmo) – Por favor, construa um algoritmo estruturado em três blocos funcionais de instruções (Entrada dos Dados, Processamento dos Cálculos e Saída dos Resultados) que permita a aplicação de um *Método Quase-Newton* no caso da resolução do sistema de equações não lineares dado a seguir:

$$\begin{cases} F(x, y) = x + 2y - 1,9 = 0 \\ G(x, y) = 3x^2 + y^2 - 7,11 = 0 \end{cases}$$

Boa Prova!

4. Metodologia para a criação de algoritmos. No desenvolvimento de algoritmos didáticos do Cálculo Numérico em Computadores, a *simplicidade* é importante porque se pretendem obter, antes de tudo, algoritmos simples. Eles apenas fixam, de modo claro e fácil de assimilar, a lógica de cada método numérico programado. Esses algoritmos não precisam ter recursos muito sofisticados para a representação dos dados ou dos resultados. Por isso, a formatação das informações não é uma preocupação típica desses algoritmos. Além disso, aspectos de robustez (para que dados mal escolhidos sejam denunciados e rejeitados logo, isto é, antes que se perca muito tempo com o seu processamento) e eficiência computacional (de modo a evitar desperdício de memória e de tempo de processamento) não precisam ser considerados com muito empenho.

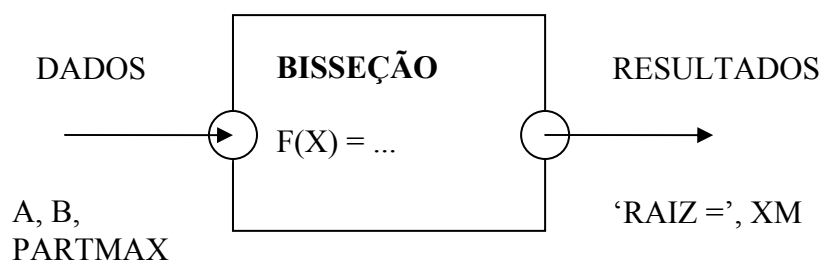
A *eficiência* da metodologia permite que ela seja empregada com sucesso na maioria dos casos, resultando em um produto com boas características: estruturação, legibilidade, facilidade de análise e de implementação, além de facilidade para extensão das funcionalidades.

A *estruturação* contribui para a boa e rápida compreensão do algoritmo. Para obtê-la, considera-se a divisão do algoritmo em três grandes blocos funcionais, a saber: *Entrada dos Dados* (caracterizado, principalmente, pelo emprego de comandos de

leitura, do tipo LEIA...); *Processamento dos Cálculos* (este bloco, que é caracterizado pelas instruções que constituem o método implementado, contém, sobretudo, comandos de atribuição, expressões matemáticas e estruturas lógicas de repetição e seleção; ele é, quase sempre, o bloco mais complexo de todo o algoritmo); e *Saída dos Resultados* (bloco caracterizado por instruções que mandam escrever resultados numéricos tanto intermediários como finais, assim como imprimir análises e comentários que avaliam a qualidade dos resultados, o sucesso ou o insucesso da aplicação do método; a maioria de seus comandos é do tipo ESCREVA...).

A *sistematicidade* do processo de desenvolvimento, garantida pelo uso da metodologia para a construção de algoritmos, estabelece certo número de etapas que devem ser observadas pelos criadores de algoritmos. De acordo com uma concepção de desenvolvimento passo-a-passo, partindo dos aspectos mais abstratos e de alto nível, pode-se incluir refinamentos progressivamente (abordagem top-down). Desse modo, a versão final é obtida com o mínimo de esforço e, talvez, o que é mais importante: apresenta um formato padronizado.

4.1. Etapas de refinamento sucessivo. Na *primeira etapa*, utiliza-se um *recurso gráfico* (caro aos engenheiros e técnicos) na forma de uma caixa preta com duas aberturas: uma, para a aquisição dos dados; a outra, para a emissão dos resultados. Esses fluxos de entrada e saída de informações são representados por setas. Sob a seta de entrada dos dados, escrevem-se os nomes das variáveis portadoras dos dados. Sob a seta de saída dos resultados, escrevem-se os identificadores das variáveis portadores dos resultados esperados com a execução do algoritmo.



A caixa preta deve ser identificada com o nome escolhido para o algoritmo, por exemplo, BISSEÇÃO. Ainda na primeira etapa do desenvolvimento do algoritmo, a sua representação gráfica é traduzida para uma *concepção literal* em que aparece o nome do algoritmo, e também os tipos (REAL, INTEIRO, etc.) de algumas das variáveis. Desse modo, produz-se um texto todo escrito com letras maiúsculas cujo emprego visa à padronização e ao melhor entendimento do algoritmo. A equação algébrica ou transcendente $f(x) = 0$, que se deseja resolver, fornece a função $F(X) = \dots$ que é definida inicialmente no interior da caixa preta e, depois, na versão literal (logo após a declaração dos tipos das variáveis).

Além disso, já que a concepção inicial é um mero esboço do algoritmo pretendido, destina-se um espaço (indicado, preliminarmente, por reticências e que será

mais tarde preenchido com blocos de instruções) para o corpo do algoritmo, o qual está situado entre as palavras reservadas INÍCIO e FIM DO ALGORITMO. Veja a figura a seguir. Nela se considera, como exemplo de aplicação da metodologia proposta, o desenvolvimento de um algoritmo para o *Método da Bissecção*. Observe que as palavras reservadas da pseudolinguagem de programação são realçadas com tipos em negrito:

```

ALGORITMO BISSEÇÃO
    REAL A, B, XM
    INTEIRO PART, PARTMAX
    F(X) = ...
INÍCIO
    .....

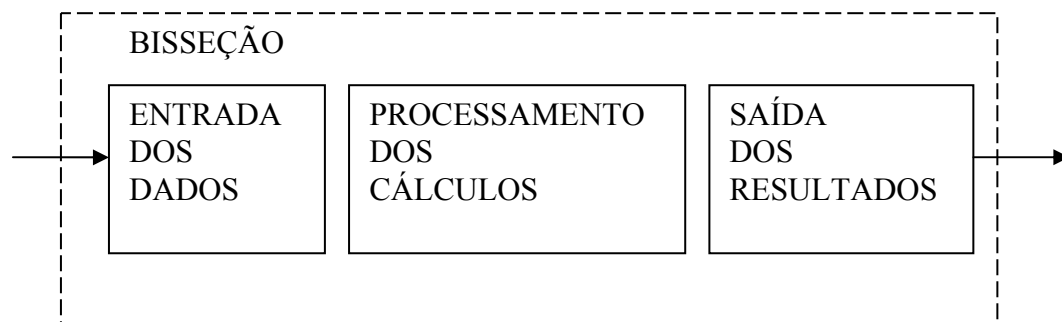
FIM DO ALGORITMO BISSEÇÃO.

```

Na *segunda etapa* do desenvolvimento do algoritmo, abre-se a caixa preta inicial, isto é, faz-se o primeiro conjunto de refinamentos. Tal se consegue considerando a caixa preta constituída de três outras caixas pretas:

- (1) ENTRADA DOS DADOS;
- (2) PROCESSAMENTO DOS CÁLCULOS; e
- (3) SAÍDA DOS RESULTADOS.

Elas representam os principais blocos de instruções de um algoritmo numérico. Cada bloco correspondente a um tipo de funcionalidade, sugerido pelo nome do bloco. Esses blocos de instruções são executados em seqüência. Veja a figura a seguir:



A essa representação gráfica com três caixas pretas corresponde uma outra, com *três blocos de instruções*, que são as suas versões literais. No caso do Método da Bissecção (concebido de modo extremamente simples), pode-se ter:

```

(* ENTRADA DOS DADOS: *)
    LEIA A, B, PARTMAX
(* FIM DA ENTRADA DOS DADOS. *)

```

```

(* PROCESSAMENTO DOS CÁLCULOS: *)
    PART = 1
    ENQUANTO (PART MENOR PARTMAX) FAÇA
        XM = (A + B)/2
        SE (F(A)*F(XM) MENOR 0) ENTÃO B = XM FIM SE
        SE (F(XM)*F(B) MENOR 0) ENTÃO A = XM FIM SE
        SE (F(XM) IGUAL 0) ENTÃO PART = PARTMAX FIM SE
        PART = PART + 1
    FIM ENQUANTO
(* FIM DO PROCESSAMENTO DOS CÁLCULOS. *)

(* SAÍDA DOS RESULTADOS: *)
    ESCREVA 'RAIZ = ', XM
(* FIM DA SAÍDA DOS RESULTADOS. *)

```

No presente caso, por se adotar uma concepção extremamente simples, o refinamento realizado na segunda etapa já permite escrever o *algoritmo completo* (exclusivamente com letras maiúsculas, para facilitar a inspeção visual). Para obter essa versão completa, basta introduzir os três blocos no corpo do algoritmo, indicando a sua execução em seqüência, e declarar os tipos de todas as variáveis empregadas no algoritmo. Em outras circunstâncias, contudo, pode ser necessário considerar que uma ou mais das caixas pretas são susceptíveis de mais refinamento progressivo. Nesse caso, cada caixa preta pode ser aberta e compreendida como a composição de uma ou mais dessas caixas. E assim por diante, conforme seja necessário para a construção passo-a-passo do algoritmo, até a sua versão final. A seguir apresenta-se uma visão do *algoritmo completamente desenvolvido*.

ALGORITMO BISSEÇÃO

```

    REAL A, B, XM; INTEIRO PART, PARTMAX
    F(X) = ...
INÍCIO
    (* ENTRADA DOS DADOS: *)
        LEIA A, B, PARTMAX
    (* FIM DA ENTRADA DOS DADOS. *)
    (* PROCESSAMENTO DOS CÁLCULOS: *)
        PART = 1
        ENQUANTO (PART MENOR PARTMAX) FAÇA
            XM = (A + B)/2
            SE (F(A)*F(XM) MENOR 0) ENTÃO B = XM FIM SE
            SE (F(XM)*F(B) MENOR 0) ENTÃO A = XM FIM SE
            SE (F(XM) IGUAL 0) ENTÃO PART = PARTMAX FIM SE
            PART = PART + 1
        FIM ENQUANTO
    (* FIM DO PROCESSAMENTO DOS CÁLCULOS. *)
    (* SAÍDA DOS RESULTADOS: *)
        ESCREVA 'RAIZ = ', XM
    (* FIM DA SAÍDA DOS RESULTADOS. *)
FIM DO ALGORITMO BISSEÇÃO.

```

4.2. Ampliação da funcionalidade de algoritmos. A funcionalidade do algoritmo BISSEÇÃO pode ser *estendida* introduzindo-se, por exemplo, um controle mais apurado do particionamento do intervalo $[A; B]$ que contém a raiz XM da equação $f(x) = 0$. Assim, a cada particionamento pode-se calcular:

(1) a *amplitude do intervalo*: $DIF = B - A$ que deve ser comparado com um valor de referência $DIFMAX$, lido na entrada dos dados; e

(2) o *valor da função* calculado no ponto médio, XM , e tomado em valor absoluto: $FM = ABS(F(XM))$, a ser comparado com um valor de referência lido na entrada de dados: $FMAX$.

Introduzindo essas *duas extensões de funcionalidade*, o algoritmo pode assumir agora o seguinte aspecto:

ALGORITMO BISSEÇÃO

REAL A, B, XM, DIF, DIFMAX, FM, FMAX

INTEIRO PART, PARTMAX

$F(X) = \dots$

INÍCIO

(* ENTRADA DOS DADOS: *)

LEIA A, B, PARTMAX, DIFMAX, FMAX

(* FIM DA ENTRADA DOS DADOS. *)

(* PROCESSAMENTO DOS CÁLCULOS: *)

PART = 1; **DIFMAX** = 0.0001; **FM** = 100

ENQUANTO ((**PART** **MENOR** **PARTMAX**) **E**

(**DIF** **MAIOR** **DIFMAX**) **E** (**FM** **MAIOR** **FMAX**)) **FAÇA**

XM = (**A** + **B**)/2

SE (**F(A)*F(XM)** **MENOR** 0) **ENTÃO** **B** = **XM** **FIM SE**

SE (**F(XM)*F(B)** **MENOR** 0) **ENTÃO** **A** = **XM** **FIM SE**

SE (**F(XM)** **IGUAL** 0) **ENTÃO** **PART** = **PARTMAX** **FIM SE**

DIF = **B** - **A**; **FM** = **ABS(F(XM))**; **PART** = **PART** + 1

FIM ENQUANTO

(* FIM DO PROCESSAMENTO DOS CÁLCULOS. *)

(* SAÍDA DOS RESULTADOS: *)

ESCREVA 'RAIZ = ', **XM**

ESCREVA 'VALOR FUNCIONAL CORRESPONDENTE = ', **FM**

(* FIM DA SAÍDA DOS RESULTADOS. *)

FIM DO ALGORITMO BISSEÇÃO.

4.3. Implementação de algoritmos. A versão final do algoritmo deve estar pronta para ser *implementada*. Assim, o algoritmo BISSEÇÃO, por exemplo, que já incorpora a funcionalidade estendida, pode ser implementado imediatamente em linguagem Pascal ou outra linguagem de programação. Para obter essa implementação

em Pascal, as estruturas lógicas do algoritmo podem ser mapeadas em estruturas Pascal correspondentes.

O quadro dado a seguir constitui uma proposta inicial a ser considerada em um projeto de pesquisa que visa definir possibilidades de mapeamento de estruturas algorítmicas (construídas em pseudolinguagem de programação) para construções de programas (em Pascal e, possivelmente, em outras linguagens de programação):

Mapeamento de Estruturas Lógicas		
Nº.	Estruturas algorítmicas	Estruturas PASCAL
1	ALGORITMO BISSEÇÃO	program bissecao; uses crt; var
2	REAL A,B,XM,DIF,DIFMAX,FM, FMAX	a,b,xm,dif,difmax,fm,fmax: real;
3	INTEIRO PART, PARTMAX	part,partmax: integer;
4	$F(X) = ((9 * X + 0) * X) - 3$	function f(x:real):real; begin f:=((9*x + 0)*x) - 3;end; {function}
5	INÍCIO	begin
6	(* ENTRADA DOS DADOS: *)	{ Entrada dos Dados: } clr;
7	LEIA A, B, PARTMAX, DIFMAX, FMAX	readln(a,b,partmax,difmax,fmax);
8	(*FIM DA ENTRADA DOS DADOS.*)	{ Fim da Entrada dos Dados. }
9	(* PROCESSAMENTO DOS CÁLCULOS: *)	{ Processamento dos Cálculos: }
10	PART = 1; DIFMAX = 0.0001; FM = 100	part:=1; difmax:=0.0001; fm:=100;
11	ENQUANTO (PART MENOR PARTMAX) E	while((part<partmax)and
12	(DIF MAIOR DIFMAX) E	(dif>difmax)and
13	(FM MAIOR FMAX)) FAÇA	(fm>fmax))do begin
14	XM = (A + B)/2	xm:=(a + b)/2;
15	SE (F(A)*F(XM) MENOR 0) ENTÃO B = XM FIM SE SE (F(XM)*F(B) MENOR 0) ENTÃO A = XM FIM SE SE (F(XM) IGUAL 0) ENTÃO PART = PARTMAX FIM SE	if(f(a)*f(xm)<0) then begin b:=xm end; {if} if(f(xm)*f(b)<0)then begin a:=xm end; {if} if(f(xm)=0)then begin part:=partmax end; {if}
16	DIF=B – A; FM=F(XM); PART=PART + 1	dif:=b-a; fm:=f(xm); part:=partmax;
17	FIM ENQUANTO	end; {while}
18	(* FIM DO PROCESSAMENTO DOS CÁLCULOS.*)	{ Fim do Processamento dos Cálculos. }
19	(* SAÍDA DOS RESULTADOS: *)	{ Saída dos Resultados: }
20	ESCREVA ‘RAIZ = ‘, XM	writeln(‘raiz = ‘,xm);
21	ESCREVA ‘VALOR FUNCIONAL CORRESPONDENTE = ‘, FM	writeln(‘valor funcional correspondente = ‘,fm); readkey;

22	(* FIM DA SAÍDA DOS RESULTADOS. *)	{ Fim da Saída dos Resultados. }
23	FIM DO ALGORITMO BISSEÇÃO.	{ Fim do Algoritmo Bissecção. } end.

O programa obtido pode agora ser digitado em um arquivo da ferramenta *Borland Pascal*, ou de outra ferramenta, para ser compilado e executado para fins de teste. Caso se deseje, a versão final do algoritmo (muito simples, no exemplo apresentado) pode ser ainda mais estendida com a inclusão de recursos mais sofisticados (incluindo, possivelmente, a formatação de dados e resultados).

5. Conclusões Preliminares. Inicialmente, podemos fazer uma discussão sobre as metodologias expostas. Nesta seção empreendemos uma *investigação preliminar* sobre as características principais da metodologia para a elaboração de provas (exposta na seção 3) e para o desenvolvimento de projetos computacionais (exposta na seção 4).

No que se refere à metodologia para a confecção de provas, temos vantagens e desvantagens.

Vantagens - trata-se de uma metodologia simples que leva, sistematicamente, a resultados padronizados e bem estruturados na maioria dos casos. Considerando essas qualidades, podemos dizer que ela é uma *metodologia adequada* para os fins aos quais se destina.

Desvantagens: ela não é uma metodologia conveniente para a confecção de provas de altíssimo nível de complexidade. O desenvolvimento de algoritmos muito complexos pode exigir uma quantidade de tempo bem superior aos cem minutos previstos para aquelas provas que temos em vista, além, evidentemente, de técnicas sofisticadas de engenharia de software.

Já, no que se refere à realização de projetos computacionais, neste trabalho apresentamos uma *metodologia* para a construção de algoritmos do Cálculo Numérico em Computadores e, além disso, para a implementação desses algoritmos na forma de programas didáticos. Essa metodologia é ilustrada com um exemplo de implementação simples do *Método da Bissecção*. A conclusão a que podemos chegar é a de que, no caso de sistemas muito simples, a metodologia leva a um bom resultado. Contudo, se os sistemas são mais complexos, a metodologia proposta exige complementação de recursos, notadamente no caso de haver necessidade de formatação de dados e de resultados.

6. Referências Bibliográficas Relacionadas com esta Apresentação:

[1] Sperandio, D.; Mendes, J. D.; Monken e Silva, L. H.: “*Cálculo Numérico – Características Matemáticas e Computacionais dos Métodos Numéricos*”. Editora Pearson Prentice Hall. São Paulo. 2003.

[2] Pacitti, T.; Atkinson, C. P.: “*Programação e Métodos Computacionais*”, em dois volumes. Livros Técnicos e Científicos Editora S. A. Rio de Janeiro. 1976.

[3] Dorn, W. S.; McCracken, D. D.: “*Cálculo Numérico com Estudos de Casos em Fortran IV*”, traduzido do inglês por José Abel Royo dos Santos e Ana Lúcia Serio dos Santos. Editora da Universidade de São Paulo e Editora Campus. São Paulo. 1978.

[4] Riso, B. G.; Schweitzer, C. M.; Heerdt, G. P. A.: “*Algoritmos Numéricos – Seqüenciais e Paralelos*”. Florianópolis, SC, Editora da UFSC, 1996.

[5] Riso, B. G.; Notare, M. S. M. A.: “*Cálculo Numérico em Computadores – Provas e Projetos, Volume I*”. Florianópolis, SC, UFSC, 2010.

[6] Riso, B. G.; Notare, M. S. M. A.: “*Cálculo Numérico em Computadores – Provas e Projetos*”. Florianópolis, SC, UFSC, 2011. (Livro em preparação).

[7] Riso, B. G.; Oliveira, C. A. de: “*Ensino de Cálculo Numérico em Computadores*”. Painel em forma de artigo apresentado na SEPEX 2009, Florianópolis, SC, UFSC.

Bernardo Gonçalves Riso
Mirela Sechi Moretti Annoni Notare

Parte I - PROVAS

PROVA 17 – Método da Secante

FOLHA DE QUESTÕES:

Questão 1 (dissertativa) – Apresente o *Método da Secante*. Por favor, ao apresentá-lo, diga em que situações esse método pode ser aplicado e o que se deve esperar como resultado de sua utilização. Sua resposta precisa ocupar, pelo menos, uma página. Se desejar, ilustre a descrição desse método com figuras, fórmulas e esquemas gráficos que complementem e enriqueçam o texto.

Questão 2 (numérica) – Por favor, considere a equação

$$f(x) = 4x^3 + 3x - 1 = 0.$$

Tente obter uma raiz real dessa equação com o emprego do *Método da Secante*. Adote $x_1 = 0$ e $x_2 = 1$ como estimativas iniciais da raiz procurada. Faça cinco iterações, sempre indicando as operações que vier a efetuar. Por favor, escreva um comentário sobre a evolução dos cálculos, e armazene os principais resultados em uma tabela com o seguinte cabeçalho:

i	x_i	x_{i+1}	x_{i+2}	y_i	y_{i+1}	y_{i+2}	$ x_{i+1} - x_i $
---	-------	-----------	-----------	-------	-----------	-----------	-------------------

Questão 3 (numérica) – Por favor, use o *Método da Secante* para tentar obter uma raiz real da equação dada a seguir:

$$f(x) = 2x^4 + 3x^2 - 10 = 0.$$

Faça três iterações a partir das estimativas $x_1 = 0,5$ e $x_2 = 1,2$. Analise a evolução do processo iterativo, buscando indícios de convergência. Indique todas as operações efetuadas e não deixe de armazenar os resultados dos cálculos em uma tabela com o cabeçalho:

i	x_i	x_{i+1}	$f(x_i)$	$f(x_{i+1})$	$x_{i+1} - x_i$
---	-------	-----------	----------	--------------	-----------------

No final, escreva um comentário sobre a evolução dos cálculos.

Questão 4 (algoritmo) – Por favor, escreva um algoritmo estruturado em três blocos funcionais (Entrada dos Dados, Processamento dos Cálculos e Saída dos Resultados) para automatizar a aplicação do *Método da Secante* no caso da determinação de uma raiz real da equação polinomial $p(x) = 4x^3 + 3x - 1 = 0$.

Boa Prova!

RESPOSTAS:

Resposta à Questão 1 – O Método da Secante é um método iterativo que se aplica tanto às equações algébricas (entre estas se situam as equações polinomiais) quanto às equações transcendentais. Ele permite determinar raízes reais, e também raízes complexas, desses dois tipos de equações. Assim, dada a equação $f(x) = 0$ e duas estimativas iniciais x_1 e x_2 de uma raiz α dessa equação, podem-se obter novas aproximações de α com o emprego da expressão

$$x_{i+1} = [x_i p(x_{i-1}) - x_{i-1} p(x_i)] / [p(x_{i-1}) - p(x_i)]$$

em que i pode assumir os valores 1, 2, 3, ...

Sendo que a sequência de valores x_1, x_2, x_3, \dots pode, ou não, convergir para o valor da raiz procurada, α .

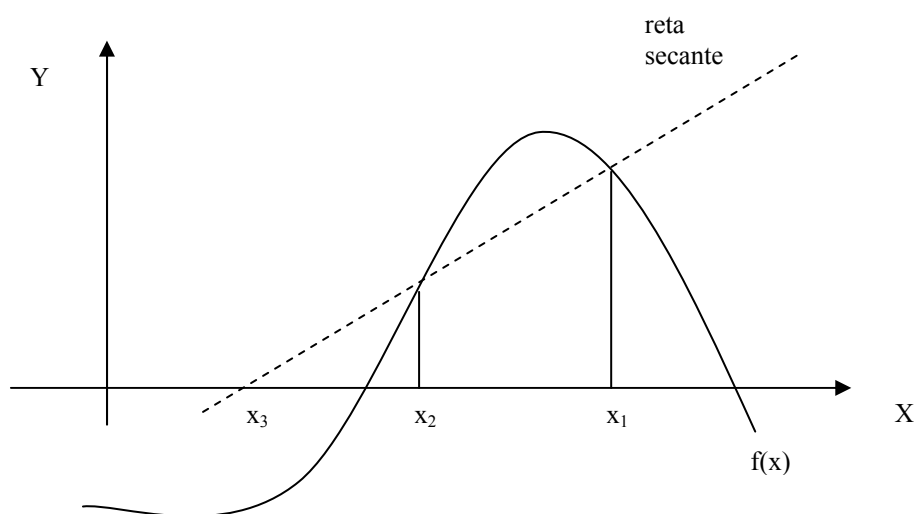
O processo iterativo não deve continuar indefinidamente. Ele precisa ser interrompido quando se obtiver:

(a) uma boa precisão para representar do valor da raiz α , de acordo com algum critério de precisão previamente estabelecido, tal como, por exemplo, o valor funcional $f(x_k) < f_{\min}$, e em que f_{\min} é um valor arbitrariamente pequeno, estabelecido de acordo com o interesse do usuário do método e x_k é um valor da sequência obtida com a aplicação da fórmula que resume o Método da Secante;

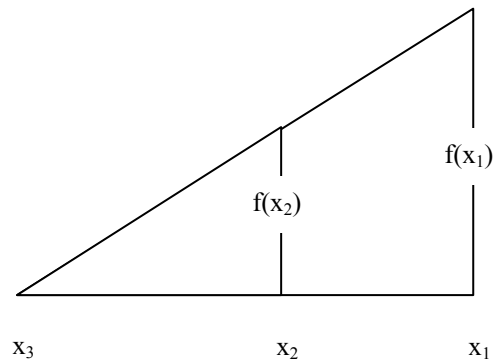
(b) um estreitamento suficiente do intervalo que contém a raiz, isto é, de modo que $|x_{i+1} - x_i| < d_{\min}$, em que d_{\min} é um valor arbitrário, suficientemente pequeno, e que relaciona-se ao objetivo da utilização prática da raiz procurada; e

(c) uma quantidade de iterações que iguale o número máximo permitido e previamente fixado. Por exemplo, definindo-se $it_{\max} = 100$ iterações. Assim, impede-se que, no caso de divergência, não se tenha um prolongamento inútil do esforço de busca de uma representação precisa da raiz α .

Pode-se fazer uma interpretação gráfica do Método da Secante com uma figura como segue:



A dedução da fórmula empregada no Método da Secante para calcularem-se as novas estimativas da raiz α pode partir da consideração de triângulos semelhantes:



Base do triângulo maior / Base do menor = Altura do maior / Altura do menor.
Isto é

$$(x_1 - x_3) / (x_2 - x_3) = f(x_1) / f(x_2)$$

Como, em uma proporção, o produto dos extremos é igual ao produto dos meios, tem-se

$$(x_1 - x_3) f(x_2) = (x_2 - x_3) f(x_1)$$

Efetuando os produtos indicados,

$$[x_1 f(x_2) - x_3 f(x_2)] = [x_2 f(x_1) - x_3 f(x_1)]$$

Isolando os termos que contêm x_3 no primeiro membro,

$$x_3 f(x_1) - x_3 f(x_2) = x_2 f(x_1) - x_1 f(x_2)$$

Deixando x_3 em evidência,

$$x_3 = [x_2 f(x_1) - x_1 f(x_2)] / [f(x_1) - f(x_2)]$$

Que é a fórmula à qual desejávamos chegar.

Resposta à Questão 2 – Em cada iteração usaremos a fórmula:

$$x_{i+2} = (x_i y_{i+1} - x_{i+1} y_i) / (y_{i+1} - y_i), \quad i = 1, 2, 3, \dots$$

sendo: $y_i = f(x_i) = 4x_i^3 + 3x_i - 1$, com $i = 1, 2, 3, 4, 5$.

i	x_i	x_{i+1}	x_{i+2}	y_i	y_{i+1}	y_{i+2}	$\text{abs}(x_{i+1} - x_i)$
1	0	1	0,1428	- 1	+ 6	- 0,560	1
2	0	0,1428	0,3245	- 1	- 0,560	+ 0,110	0,1428
3	0,1428	0,3245	0,2947	- 0,560	+ 0,110	- 0,0135	0,1817
4	0,3245	0,2947	0,2980	+ 0,110	- 0,0135	- 0,0001	0,0298
5	0,2947	0,2980	0,2970	- 0,0135	- 0,0001	- 0,0042	0,0033

Linha i = 1:

Tem-se $x_1 = 0$,

$$\text{com } y_1 = f(x_1) = f(0) = 4(0)^3 + 3(0) - 1 = 1; \text{ e}$$

$$x_2 = 1,$$

$$\text{com } y_2 = f(x_2) = f(1) = 4(1)^3 + 3(1) - 1 = 6.$$

Cálculo de x_3 :

$$x_3 = (x_2 y_1 - x_1 y_2) / (y_1 - y_2)$$

$$x_3 = [(1)(-1) - (0)(6)] / [(-1) - (6)] = -1 / -7$$

$$x_3 = 0,1428$$

Cálculo de y_3 :

$$y_3 = f(x_3) = f(0,1428) = 4(0,1428)^3 + 3(0,1428) - 1$$

$$y_3 = -0,560$$

Cálculo do intervalo que contém a raiz:

$$|x_2 - x_1| = |1 - 0| = 1$$

Linha i = 2:

Tem-se $x_2 = 0$, com $y_2 = 1$; e

$$x_3 = 0,1428, \text{ com } y_3 = -0,560$$

Cálculo de x_4 :

$$x_4 = (x_3 y_2 - x_2 y_3) / (y_2 - y_3)$$

$$x_4 = [(0,1428)(-1) - (0)(0,560)] / [(-1) - (0,560)]$$

$$x_4 = 0,3245$$

Cálculo de y_4 :

$$y_4 = f(x_4) = f(0,3245) = 4(0,3245)^3 + 3(0,3245) - 1$$

$$y_4 = 0,1102$$

Cálculo do intervalo que contém a raiz:

$$|x_3 - x_2| = |0,1428 - 0| = 0,1428$$

Linha i = 3:

Tem-se $x_3 = 0,1428$, com $y_3 = -0,560$; e

$$x_4 = 0,3245, \text{ com } y_4 = 0,1102.$$

Cálculo de x_5 :

$$x_5 = (x_4 y_3 - x_3 y_4) / (y_3 - y_4)$$

$$x_5 = [(0,3245)(-0,5601) - (0,1428)(0,1102)] / [(-0,560) - (0,1102)]$$

$$x_5 = -0,1974 / -0,67$$

$$x_5 = 0,2947$$

Cálculo de y_5 :

$$y_5 = f(x_5) = f(0,2947) = 4(0,2947)^3 + 3(0,2947) - 1$$

$$y_5 = -0,01352$$

Cálculo do intervalo que contém a raiz:

$$|x_4 - x_3| = |0,3245 - 0,1428| = 0,1817$$

Linha i = 4:

Tem-se $x_4 = 0,3245$, com $y_4 = 0,1102$; e

$x_5 = 0,2947$, com $y_5 = -0,01352$.

Cálculo de x_6 :

$$x_6 = (x_5 y_4 - x_4 y_5) / (y_4 - y_5)$$

$$x_6 = [(0,2947)(0,1102) - (0,3245)(-0,01352)] / [(0,1102) - (-0,01352)]$$

$$x_6 = 0,03680 / 0,1235$$

$$x_6 = 0,2980$$

Cálculo de y_6 :

$$y_6 = f(x_6) = f(0,2980) = 4(0,2980)^3 + 3(0,2980) - 1$$

$$y_6 = -0,0001456$$

Cálculo do intervalo que contém a raiz:

$$|x_5 - x_4| = |0,2947 - 0,3245| = 0,0298$$

Linha i = 5:

Tem-se $x_5 = 0,2947$, com $y_5 = -0,0135$; e

$x_6 = 0,2980$, com $y_6 = -0,000146$.

Cálculo de x_7 :

$$x_7 = (x_6 y_5 - x_5 y_6) / (y_5 - y_6)$$

$$x_7 = [(0,2980)(-0,0135) - (0,2947)(-0,000146)] / [(-0,0135) - (-0,000146)]$$

$$x_7 = -0,00398 / -0,0134$$

$$x_7 = 0,2970$$

Cálculo de y_7 :

$$y_7 = f(x_7) = f(0,2970) = 4(0,2970)^3 + 3(0,2970) - 1$$

$$y_7 = -0,00421$$

Cálculo do intervalo que contém a raiz:

$$|x_6 - x_5| = |0,2980 - 0,2947| = 0,0033$$

Comentário: o processo iterativo resultante da aplicação do Método da Secante apresenta uma sequência de valores que se aproxima cada vez mais da raiz procurada. A melhor representação obtida para essa raiz é 0,2980.

Resposta à Questão 3 – Nesta questão, tem-se $f(x) = 2x^4 + 3x^2 - 10$. As estimativas iniciais da raiz procurada, α , são: $x_1 = 0,5$ e $x_2 = 1,2$.

i = 1:

$$x_1 = 0,5$$

$$f(x_1) = 2(0,5)^4 + 3(0,5)^2 - 10 = -9,125$$

$$x_2 = 1,2$$

$$f(x_2) = 2(1,2)^4 + 3(1,2)^2 - 10 = -1,5328$$

$$x_2 - x_1 = 1,2 - 0,5 = 0,7.$$

i = 2:

$$x_2 = 1,2; f(x_2) = -1,5328$$

$$x_3 = [x_2 f(x_1) - x_1 f(x_2)] / [f(x_1) - f(x_2)]$$

$$x_3 = [(1,2)(-9,125) - (0,5)(-1,5328)] / [(-9,125) - (-1,5328)]$$

$$x_3 = 1,34$$

$$f(x_3) = 2(1,34)^4 + 3(1,34)^2 - 10 = 1,87$$

$$x_3 - x_2 = 1,34 - 1,2 = 0,14.$$

i = 3:

$$x_3 = 1,34; f(x_3) = 1,87$$

$$x_4 = [x_3 f(x_2) - x_2 f(x_3)] / [f(x_2) - f(x_3)]$$

$$x_4 = [(1,34)(-1,5328) - (1,2)(1,87)] / [(-1,5328) - (1,87)]$$

$$x_4 = 1,26$$

$$f(x_4) = 2(1,26)^4 + 3(1,26)^2 - 10 = -0,123$$

$$x_4 - x_3 = 1,26 - 1,34 = -0,08.$$

i = 4:

$$x_4 = 1,26; f(x_4) = -0,123$$

$$x_5 = [x_4 f(x_3) - x_3 f(x_4)] / [f(x_3) - f(x_4)]$$

$$x_5 = [(1,26)(1,87) - (1,34)(-0,123)] / [(1,87) - (-0,123)]$$

$$x_5 = 1,2679$$

$$f(x_5) = 2(1,2679)^4 + 3(1,2679)^2 - 10 = -8,72 \cdot 10^{-3}$$

$$x_5 - x_4 = 1,2679 - 1,26 = 0,0079.$$

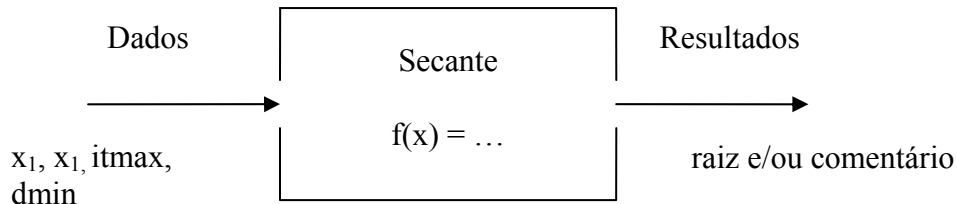
Resumindo, em uma tabela, os cálculos efetuados:

i	x_i	x_{i+1}	$f(x_i)$	$f(x_{i+1})$	$x_{i+1} - x_i$
1	0,5	1,2	-9,125	-1,5328	+0,7
2	1,2	1,34	-1,5328	1,87	+0,14
3	1,34	1,26	1,87	-0,123	-0,08
4	1,26	1,2679	-0,123	$-8,72 \cdot 10^{-3}$	+0,0079

Comentário: neste caso de aplicação do Método da Secante, observa-se que o processo iterativo produz uma sequência de valores que é convergente para a raiz α da equação dada. Essa convergência fica evidente quando se considera que a diferença $x_{i+1} - x_i$ diminui progressivamente, em valor absoluto, a cada iteração. Após realizar-se o processo pode-se apresentar:

$$\alpha \approx 1,2679 \text{ com } f(1,2679) = -8,72 \cdot 10^{-3}.$$

Resposta à Questão 4 – O algoritmo pode ser visualizado, em altíssimo nível de abstração, como uma caixa preta identificada com a denominação Secante. Tal caixa tem duas aberturas: na primeira, a da esquerda, dá-se a entrada dos dados; e na segunda, a da direita, a saída dos resultados.



Neste caso, os dados são: as estimativas iniciais da raiz (variáveis x_1 e x_2), a quantidade máxima de iterações permitidas (variável $itmax$) e o desvio máximo permitido (variável $dmin$) que deve ser um valor pequeno, para assegurar a apresentação de um valor preciso da raiz ao final das iterações.

A execução do algoritmo precisa fornecer um valor que represente a raiz com precisão (variável $raiz$) no caso em que o processo iterativo produza uma sequência de valores convergente. De qualquer modo, mesmo que não se verifique a convergência, um comentário sobre a qualidade do resultado pode ser impresso.

Deve-se prever a definição da função $f(x)$.

O esquema gráfico da caixa preta corresponde a um trecho escrito em pseudolinguagem de programação que já considera os tipos das variáveis mencionadas e de outras variáveis: IT para a contagem de iterações; $X1$ e $X2$ para guardar as

aproximações obtidas em duas iterações sucessivas; e D, para guardar o valor da diferença entre duas aproximações, calculado a cada iteração.

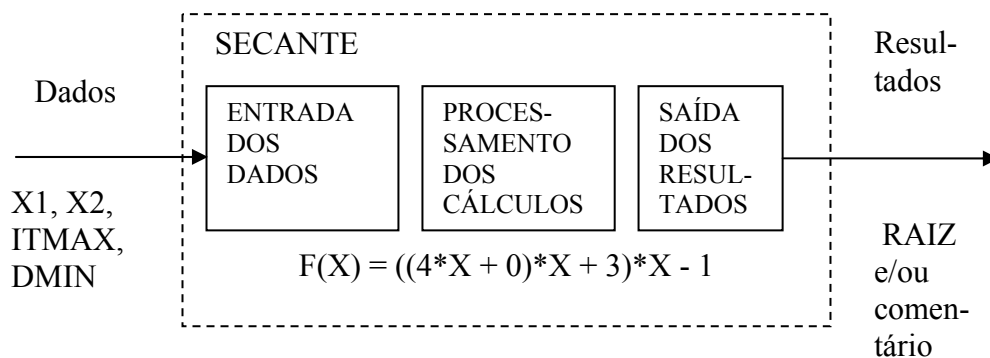
```

ALGORITMO SECANTE
  INTEIRO IT, ITMAX
  REAL X1, X2, D, DMIN
  F(X) = ...

INÍCIO
  .
  .
  .
FIM DO ALGORITMO SECANTE

```

Segunda etapa: abre-se a caixa preta para visualizar sua composição interna. As caixas pretas *Entrada dos Dados*, *Processamento dos Cálculos* e *Saída dos Resultados*, agora visíveis, definem as principais partes (cada parte com sua funcionalidade específica) do algoritmo que se está construindo.



A *Entrada dos Dados* pode, então, ser detalhada, assim como o *Processamento dos Cálculos* e a *Saída dos Resultados*. Os tipos das novas variáveis são declarados. O algoritmo toma a seguinte feição:

```

ALGORITMO SECANTE

```

```

(* CALCULA UMA RAIZ REAL DE UMA EQUAÇÃO ALGÉBRICA
OU TRANSCENDENTE PELO MÉTODO DA SECANTE. *)

```

```

  REAL X1, X2, X3, Y1, Y2, Y3

```

REAL DIF, DIFMIN
INTEIRO I, IMAX
 $F(X) = ((4 \cdot X + 0) \cdot X + 3) \cdot X - 1$

INÍCIO

(* ENTRADA DOS DADOS: *)

LEIA X1, X2, IMAX, DIFMIN

(* FIM DA ENTRADA DOS DADOS. *)

(* PROCESSAMENTO DOS CÁLCULOS: *)

I = 1; DIF = 100

ENQUANTO ((I MENOR IMAX) E (DIF MAIOR DIFMIN)) FAÇA

Y1 = F(X1); Y2 = F(X2)

$X3 = (X1 \cdot Y2 - X2 \cdot Y1) / (Y2 - Y1)$

(* CÁLCULO DO DESVIO: *)

DIF = ABS(X2 - X1)

(* FIM DO CÁLCULO DO DESVIO: *)

SE (ABS(Y2) MENOR ABS(Y1)) ENTÃO

X1 = X2

FIM SE

X2 = X3

(* INCREMENTO DO CONTADOR DE ITERAÇÕES: *)

I = I + 1

(* FIM DO INCREMENTO DO CONTADOR DE ITERAÇÕES: *)

FIM ENQUANTO

(* FIM DO PROCESSAMENTO DOS CÁLCULOS. *)

(* SAÍDA DOS RESULTADOS: *)

SE (DIF MENOR OU IGUAL DIFMIN) ENTÃO

ESCREVA 'PRECISÃO OBTIDA'

ESCREVA 'RAIZ = ', X3

FIM SE

SE (DIF MAIOR DIFMIN) ENTÃO

ESCREVA 'PRECISÃO NÃO OBTIDA'

FIM SE

(* FIM DA SAÍDA DOS RESULTADOS. *)

FIM DO ALGORITMO SECANTE.

PROVA 18 – Método Quase-Newton

FOLHA DE QUESTÕES:

Questão 1 (dissertativa) – Por favor, considere a resolução numérica de *Sistemas de Equações Não-Lineares*. Apresente um Método Quase-Newton adequado à resolução desse tipo de sistema. Compare as características desse método com as do Método de Newton. Não deixe de escrever, pelo menos, uma página. Se achar conveniente, use fórmulas, gráficos e tabelas para ilustrar a sua exposição.

Questão 2 (numérica) – Use um *Método Quase-Newton* para tentar resolver o sistema dado a seguir:

$$\begin{cases} x + 2y - 2,1 = 0 \\ 3x^2 + y^2 - 6,9 = 0 \end{cases}$$

Adote $x = y = 0,5$ como estimativa inicial da solução do sistema. Faça duas iterações, indicando sempre todas as operações efetuadas. Para apresentar os resultados organizadamente, preencha uma tabela com o cabeçalho:

i	x_i	y_i	desvio _i = $\text{abs}(x_{i+1} - x_i) + \text{abs}(y_{i+1} - y_i)$
---	-------	-------	---

Por favor, no final, escreva um comentário que avalie a evolução do processo iterativo.

Questão 3 (numérica) – Tente resolver o sistema de equações não lineares dado abaixo com a aplicação de um *Método Quase Newton*:

$$\begin{cases} F(x, y) = 2x - 2y - 2,1 = 0 \\ G(x, y) = x^3 + 3y^2 - 6,9 = 0 \end{cases}$$

Adote $x = y = 0,5$ como estimativa inicial da solução do sistema. Faça três iterações. Preencha uma tabela que apresente os resultados organizadamente. Por favor, não deixe de escrever um comentário que avalie o processo iterativo.

Questão 4 (algoritmo) – Por favor, construa um algoritmo estruturado em três blocos funcionais de instruções (Entrada dos Dados, Processamento dos Cálculos e Saída dos Resultados) que permita a aplicação de um *Método Quase-Newton* no caso da resolução do sistema de equações não lineares dado a seguir:

$$\begin{cases} F(x, y) = x + 2y - 1,9 = 0 \\ G(x, y) = 3x^2 + y^2 - 7,11 = 0 \end{cases}$$

Boa Prova!

RESPOSTAS:

Resposta à Questão 1 – Para os sistemas de equações não lineares, não existem métodos analíticos gerais de solução, isto é, métodos analíticos que se apliquem a todo e qualquer desses sistemas. Por esse motivo, em muitos casos, a resolução de tais sistemas precisa ser efetuada com o emprego de um método numérico. Mesmo que se trate de um tipo particular de sistema não linear para o qual tenha sido desenvolvido um método analítico, ainda assim pode ser conveniente a utilização de um método numérico. Esse é o caso, por exemplo, em que se dispõe de uma eficiente implementação computacional de tal método.

Esses fatos ilustram bem a importância da aplicação de métodos numéricos no contexto de sistemas de equações não lineares e justificam o seu estudo. Ainda mais que sabemos que esse tipo de sistema ocorre com muita frequência na modelagem matemática de muitos fenômenos estudados nas diversas ciências exatas e na tecnologia.

Para resolver numericamente um sistema de equações não lineares pode-se lançar mão, por exemplo, do Método de Newton e de métodos denominados Quase Newton.

No caso do Método de Newton, as fórmulas a utilizar iterativamente podem ser obtidas realizando-se uma analogia com o Método de Newton-Raphson, que é empregado na resolução de equações algébricas e transcendentais isoladas. Agora, trabalha-se com matrizes e, em vez de uma derivada total, tem-se uma matriz de derivadas parciais: a jacobiana do sistema.

Assim, dado o sistema de equações não lineares, $F(X) = 0$, em uma notação vetorial; e uma estimativa preliminar da solução, X_1 ; em cada iteração resolve-se o sistema linear

$$J D = -F$$

em que J é a matriz jacobiana calculada para essa iteração, e F é o vetor dos valores funcionais calculados com a última estimativa. O vetor D tem como seus elementos as incógnitas do sistema linear. Ele representa a correção que se deve aplicar à estimativa obtida na iteração anterior para se obter a nova estimativa:

$$X_{i+1} = X_i + D_i$$

O processo iterativo resultante da aplicação do Método de Newton pode apresentar uma sequência de valores que se aproxima mais ou menos rapidamente da solução exata. Nesses casos, diz-se que há convergência. Em alguns casos, entretanto, a sequência não é convergente. Convém, então, reescrever o sistema ou adotar uma estimativa inicial mais próxima da solução exata. Ou empregar outro método.

Os métodos numéricos ditos Quase Newton assemelham-se ao Método de Newton. Um desses métodos calcula a jacobiana uma única vez e a emprega em todas as iterações. Outra variante adapta a fórmula do Método de Newton-Raphson de modo

que, em lugar da derivada total, emprega a derivada parcial. Por exemplo, dado o sistema de equações não lineares:

$$\begin{cases} F_1(x, y) = x + 2y - 1,9 = 0 \\ F_2(x, y) = 3x^2 + y^2 - 7,11 = 0 \end{cases}$$

em cada iteração ($i = 1, 2, 3, \dots$) faz-se o cálculo de nova estimativa:

$$\begin{aligned} x_{i+1} &= x_i - F_1(x_i, y_i)/DXF_1(x_i, y_i) \\ y_{i+1} &= y_i - F_2(x_i, y_i)/DYF_2(x_i, y_i) \end{aligned}$$

Nessas expressões, $DXF_1(x_i, y_i)$ representa a derivada parcial de $F_1(x, y)$ em relação a x , calculada para $x = x_i$ e $y = y_i$; e, de modo semelhante, $DYF_2(x_i, y_i)$ corresponde à derivada parcial de $F_2(x, y)$ em relação a y , calculada para $x = x_i$ e $y = y_i$.

Resposta à Questão 2 – Nessa questão, tem-se

$$\begin{cases} f(x, y) = x + 2y - 2,1 = 0 \\ g(x, y) = 3x^2 + y^2 - 6,9 = 0 \end{cases}$$

Assim, $\partial f/\partial x = 1$; e $\partial g/\partial y = 2y$

Primeira iteração ($i = 1$).

Tem-se a estimativa inicial: $x_1 = y_1 = 0,5$; deseja-se obter x_2 e y_2 .

$$\begin{aligned} x_2 &= x_1 - f(x_1, y_1) / \partial f/\partial x \text{ calculada para } x = x_1 \text{ e } y = y_1 \\ y_2 &= y_1 - g(x_1, y_1) / \partial g/\partial y \text{ calculada para } x = x_1 \text{ e } y = y_1 \end{aligned}$$

$$\begin{aligned} x_2 &= 0,5 - [0,5 + 2(0,5) - 2,1] / [1] = 0,5 - 0,5 - 1 + 2,1 = 1,1 \\ y_2 &= 0,5 - [3(0,5)^2 + (0,5)^2 - 6,9] / [2(0,5)] = 0,5 - [-5,9] / [1] = 6,4 \end{aligned}$$

$$\text{desvio}_1 = |1,1 - 0,5| + |6,4 - 0,5| = 6,5$$

Segunda iteração ($i = 2$).

Tem-se a estimativa: $x_2 = 1,1$ e $y_2 = 6,4$; deseja-se obter x_3 e y_3 .

$$\begin{aligned} x_3 &= x_2 - f(x_2, y_2) / \partial f/\partial x \text{ calculada para } x = x_2 \text{ e } y = y_2 \\ y_3 &= y_2 - g(x_2, y_2) / \partial g/\partial y \text{ calculada para } x = x_2 \text{ e } y = y_2 \end{aligned}$$

$$\begin{aligned} x_3 &= 1,1 - [1,1 + 2(6,4) - 2,1] / [1] = 1,1 - 1,1 - 12,8 + 2,1 = -10,7 \\ y_3 &= 6,4 - [3(1,1)^2 + (6,4)^2 - 6,9] / [2(6,4)] = 6,4 - [37,69] / [12,8] = 3,455 \end{aligned}$$

$$\text{desvio}_2 = |-10,7 - 1,1| + |3,455 - 6,4| = 14,745$$

i	x _i	y _i	desvio _i = (x _{i+1} - x _i) + (y _{i+1} - y _i)
1	0,5	0,5	6,5
2	1,1	6,4	14,745
3	-10,7	3,455	-

Comentário: o aumento do desvio da primeira iteração para a segunda (de 6,5 para 14,745) indica divergência. Nesse caso, torna-se conveniente adotar outra estimativa inicial, ou reescrever as fórmulas empregadas ou, ainda, buscar outro método para resolver o sistema dado.

Resposta à Questão 3 – Tem-se o sistema não linear:

$$\begin{cases} F(x, y) = 2x - 2y - 2,1 = 0 \\ G(x, y) = x^3 + 3y^2 - 6,9 = 0 \end{cases}$$

e $x = y = 0,5$ como estimativa inicial da solução. As derivadas parciais das funções envolvidas que interessam ao Método Quase Newton são: $\partial F / \partial x = 2$; e $\partial G / \partial y = 6y$.

Primeira iteração (i = 1).

Tem-se a estimativa inicial: $x_1 = y_1 = 0,5$; deseja-se obter x_2 e y_2 .
 $x_2 = x_1 - \{F(x_1, y_1) / \partial F / \partial x \text{ calculada para } x = x_1 \text{ e } y = y_1\}$
 $y_2 = y_1 - \{G(x_1, y_1) / \partial G / \partial y \text{ calculada para } x = x_1 \text{ e } y = y_1\}$
 $x_2 = 0,5 - \{[2(0,5) - 2(0,5) - 2,1] / [2]\} = 0,5 - \{[- 2,1] / [2]\} = 1,55$
 $y_2 = 0,5 - \{[(0,5)^3 + 3(0,5)^2 - 6,9] / [6(0,5)]\} = 0,5 - \{[- 6,025] / [3]\} = 2,508$
 $\text{desvio}_1 = |1,55 - 0,5| + |2,508 - 0,5| = 1,05 + 2,008 = 3,058$

Segunda iteração (i = 2).

Tem-se a estimativa: $x_2 = 1,55$ e $y_2 = 2,508$; deseja-se obter x_3 e y_3 .
 $x_3 = x_2 - \{F(x_2, y_2) / \partial F / \partial x \text{ calculada para } x = x_2 \text{ e } y = y_2\}$
 $y_3 = y_2 - \{G(x_2, y_2) / \partial G / \partial y \text{ calculada para } x = x_2 \text{ e } y = y_2\}$
 $x_3 = 1,55 - \{[2(1,55) - 2(2,508) - 2,1] / [2]\} = 1,55 - \{[- 4,016] / [2]\} = 3,558$
 $y_3 = 2,508 - \{[(1,55)^3 + 3(2,508)^2 - 6,9] / [6(2,508)]\} =$
 $y_3 = 2,508 - \{[15,69] / [15,048]\} = 1,465$
 $\text{desvio}_2 = |3,558 - 1,55| + |2,508 - 1,465| = 2,008 + 1,043 = 3,051$

Terceira iteração (i = 3).

Tem-se a estimativa: $x_3 = 3,558$ e $y_3 = 1,465$; deseja-se obter x_4 e y_4 .
 $x_4 = x_3 - \{F(x_3, y_3) / \partial F / \partial x \text{ calculada para } x = x_3 \text{ e } y = y_3\}$
 $y_4 = y_3 - \{G(x_3, y_3) / \partial G / \partial y \text{ calculada para } x = x_3 \text{ e } y = y_3\}$
 $x_4 = 3,558 - \{[2(3,558) - 2(1,465) - 2,1] / [2]\}$
 $x_4 = 3,558 - \{[- 4,016] / [2]\} = 5,566$
 $y_4 = 1,465 - \{[(3,558)^3 + 3(1,465)^2 - 6,9] / [6(1,465)]\}$
 $y_4 = 1,465 - \{[44,58] / [8,79]\} = - 3,607$
 $\text{desvio}_3 = |5,566 - 3,558| + |- 3,607 - 1,465| = 2,008 + 5,072 = 7,080$

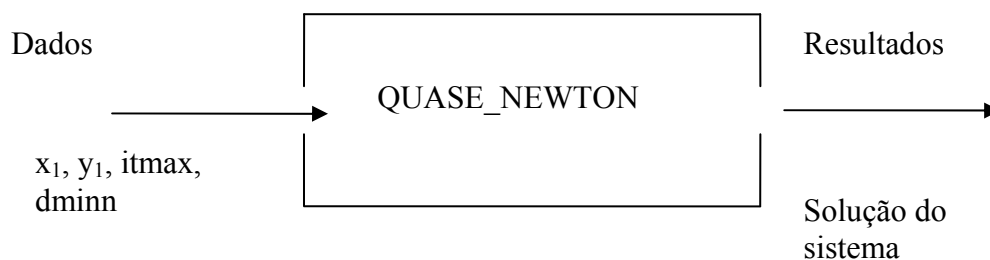
i	x_i	y_i	$\text{desvio}_i = x_{i+1} - x_i + y_{i+1} - y_i $
1	0,5	0,5	3,058
2	1,55	2,508	3,051
3	3,558	1,465	7,080
4	5,566	- 3,607	-

Comentário: como se pode observar na tabela, embora tenha havido uma pequena diminuição do desvio da primeira para a segunda iteração, de 3,058 para 3,051, em seguida o desvio aumentou consideravelmente, de 3,051 para 7,080. Pode-se interpretar esse fato como uma oscilação que não garante que haja convergência na sequência dos valores que constituem as estimativas da solução do sistema. Nesse caso, pode ser conveniente realizarem-se mais iterações para verificar como o processo iterativo passa se comporta.

Resposta à Questão 4 – Constrói-se um algoritmo estruturado para aplicar o Método Quase Newton ao sistema dado no enunciado da questão:

$$\begin{cases} F(x, y) = x + 2y - 1,9 = 0 \\ G(x, y) = 3x^2 + y^2 - 7,11 = 0 \end{cases}$$

O algoritmo pode, inicialmente, ser representado por uma caixa preta identificada como QUASE_NEWTON. Ela dispõe de duas aberturas de cada lado. Na da esquerda, dá-se a entrada dos dados. E na da direita, a saída dos resultados:



Esse esquema gráfico inicial corresponde a um esboço do algoritmo escrito em pseudolinguagem de programação. Tal esquema contempla já os tipos das variáveis mencionadas:

ALGORITMO QUASE_NEWTON

```

REAL    X(100), Y(100), D(100), DMIN
INTEIRO IT, ITMAX
F(X, Y) = X+2*Y-1.9; G(X, Y) = 3*X*X+Y*Y-7.11
DFX(X, Y) = 1; DGY(X, Y) = 2*Y

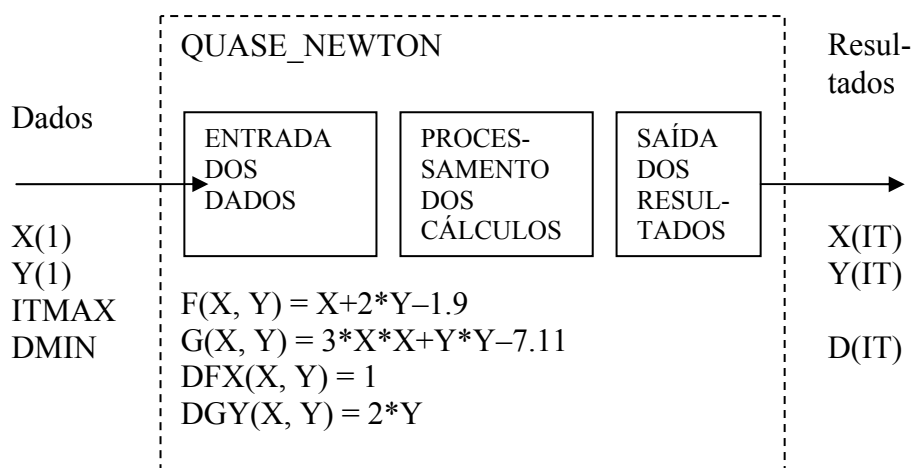
```

INÍCIO

·
·
·

FIM DO ALGORITMO QUASE_NEWTON.

Abrindo-se a caixa preta para visualização de sua composição interna, distinguem-se as três caixas que correspondem aos blocos de instruções com as principais funcionalidades de um algoritmo numérico. As caixas pretas são: *Entrada dos Dados*, *Processamento dos Cálculos* e *Saída dos Resultados*.



Em seguida, detalhando cada um dos três blocos de instruções:

ALGORITMO QUASE_NEWTON

(* ESTE ALGORITMO RESOLVE UM SISTEMA NÃO LINEAR DE DUAS EQUAÇÕES PELO MÉTODO QUASE-NEWTON. *)

```

REAL X(100), Y(100), D(100), DMIN
INTEIRO IT, ITMAX
 $F(X, Y) = X + 2 * Y - 1.9$ 
 $G(X, Y) = 3 * X * X + Y * Y - 7.11$ 
 $DFX(X, Y) = 1$ 
 $DGY(X, Y) = 2 * Y$ 

```

INÍCIO

(* ENTRADA DOS DADOS: *)

```
LEIA X(1), Y(1), ITMAX, DMIN
```

(* FIM DA ENTRADA DOS DADOS. *)

(* PROCESSAMENTO DOS CÁLCULOS: *)

```
IT = 1; D(1) = 100
```

```
ENQUANTO ((IT MENOR ITMAX) E (D(IT) MAIOR DMIN)) FAÇA
```

```
  X(IT+1) = X(IT) - (F(X(IT), Y(IT)) / DFX(X(IT), Y(IT)))
```

```
  Y(IT+1) = Y(IT) - (G(X(IT), Y(IT)) / DGY(X(IT), Y(IT)))
```

```
  D(IT+1) = ABS(X(IT+1) - X(IT)) + ABS(Y(IT+1) - Y(IT))
```

```
  IT = IT + 1
```

```
FIM ENQUANTO
```

(* FIM DO PROCESSAMENTO DOS CÁLCULOS. *)

(* SAÍDA DOS RESULTADOS: *)

```
SE (D(IT) MENOR OU IGUAL DMIN) ENTÃO
```

```
  ESCREVA 'O PROCESSO RESULTOU EM CONVERGÊNCIA'
```

```
  ESCREVA 'VALORES APROXIMADOS:'
```

```
  ESCREVA 'X = ', X(IT), 'Y = ', Y(IT), 'DESVIO = ', D(IT)
```

```
FIM SE
```

```
SE (D(IT) MAIOR DMIN) ENTÃO
```

```
  ESCREVA 'NÃO SE OBTVEVE RESULTADO PRECISO'
```

```
FIM SE
```

(* FIM DA SAÍDA DOS RESULTADOS.*)

FIM DO ALGORITMO QUASE_NEWTON.

PROVA 19 – Introdução ao Ajustamento de Curvas

FOLHA DE QUESTÕES:

Questão 1 (dissertativa) – Apresente o problema do ajustamento de curvas. Diga como esse problema pode ser resolvido com o *Método dos Mínimos Quadrados*. Por favor, ocupe pelo menos uma página.

Questão 2 (numérica) – Use o *Método dos Mínimos Quadrados* para ajustar uma *reta* aos dados tabelados apresentados a seguir. Por favor, estenda a tabela dada para o cálculo dos somatórios. Indique todas as operações efetuadas.

x	0,0	1,0	0,5	0,3	0,3	0,5	0,7	0,8
y	5,1	5,5	5,1	5,3	5,2	5,0	5,4	5,5

Questão 3 (numérica) – Empregue o *Método dos Mínimos Quadrados* para ajustar uma *parábola* aos dados da tabela apresentada a seguir. Estenda essa tabela para o cálculo dos somatórios. Por favor, indique todas as operações efetuadas.

x	0,0	1,0	0,5	0,3	0,3	0,5	0,7	0,8
y	5,1	5,5	5,1	5,3	5,2	5,0	5,4	5,5

Questão 4 (algoritmo) – Construa um algoritmo para ler e imprimir uma *tabela* de valores (x_i, y_i) , $i = 1, n..$

Boa Prova!

RESPOSTAS:

Resposta à Questão 1 – O problema do ajustamento de uma curva $f(x)$ a um conjunto de m pares de dados (x_i, y_i) , $i = 1, 2, 3, \dots, m$, consiste em determinar a função $f(x)$, de um tipo previamente escolhido (uma função polinomial, por exemplo) que traduz, de modo suficientemente bom, a tendência revelada pela disposição dos pontos (x_i, y_i) , $i = 1, 2, 3, \dots, m$, que representam graficamente tais pares de dados.

Essa tendência pode ser observada convenientemente ao se marcar os pontos em uma folha de papel. Ela pode ser crescente, decrescente, estável, pode crescer aceleradamente ou decrescer aceleradamente, apresentar sinuosidade e assim por diante. Cabe ao observador do comportamento dos dados escolher um tipo de curva (polinomial, exponencial, logarítmica,...) para ajustar aos pontos.

O ajuste de curvas pode ser realizado à mão, com o emprego de recursos gráficos. Entretanto, é mais adequado, muitas vezes, obter uma solução numérica. Tal solução deve resultar em uma expressão algébrica associada à curva de ajustamento.

Assim, no caso de se desejar ajustar uma reta, então $p(x) = a_1 + a_2x$ e tem-se que determinar o valor de a_1 e de a_2 , que são os dois coeficientes do polinômio do primeiro grau que correspondem à reta. No caso de se ajustar uma parábola, é preciso determinar os três coeficientes a_1, a_2 e a_3 de $p(x) = a_1 + a_2 x + a_3 x^2$.

Uma teoria muito empregada em situações práticas, e que orienta o procedimento para a definição de tais expressões algébricas no ajustamento de curvas, é a do Método dos Mínimos Quadrados. A idéia básica dessa teoria é a de que, uma vez escolhido o tipo que essa curva deve ter (por exemplo, uma reta), a melhor curva de ajustamento, $p(x)$, deverá resultar no mínimo da soma dos quadrados dos desvios. Isso quer dizer que qualquer outra reta que viesse a ser escolhida resultaria em um valor maior para a soma dos quadrados dos desvios.

De acordo com essa concepção, e considerando como desvio a diferença

$$\text{desvio}_i = [y_i - p(x_i)]$$

em que y_i é uma ordenada de um par de dados (x_i, y_i) , $i = 1, 2, 3, \dots, m$, e $p(x_i)$ é o valor do polinômio de ajustamento

$$p(x) = a_1 + a_2 x + \dots + a_n x^{n-1} + a_{n+1} x^n$$

calculado para o valor da abscissa x_i correspondente àquela ordenada y_i .

$$[\text{desvio}_i]^2 = [y_i - p(x_i)]^2$$

A consideração do ponto de mínimo da função

$$Q(a_1, a_2, a_3, \dots, a_{n+1}) = \sum [y_i - p(x_i)]^2, \quad i = 1, 2, 3, \dots, m$$

denominada soma dos quadrados dos desvios, cujos argumentos são os $n+1$ coeficientes $a_1, a_2, a_3, \dots, a_{n+1}$ de um polinômio de ajustamento $p(x)$ de grau n , conduz às derivadas parciais dessa função (função Q) em relação a cada um dos argumentos.

No ponto de mínimo de Q , todas essas derivadas parciais são nulas.

$$\partial Q / \partial a_i = 0, \quad i = 1, 2, 3, \dots, n+1$$

Tais equações, consideradas simultaneamente, constituem um sistema de $n+1$ equações lineares com $n+1$ incógnitas. A resolução desse sistema permite obter os valores dos $n+1$ coeficientes: $a_1, a_2, a_3, \dots, a_{n+1}$ do polinômio de ajustamento $p(x)$.

Como se vê, os coeficientes são obtidos pela resolução de um sistema de equações lineares. Tem-se que resolver um sistema de duas equações e duas incógnitas para o caso da reta, um sistema de três equações e três incógnitas para o caso da parábola, e assim por diante.

O Método dos Mínimos Quadrados pode ser estendido de maneira a permitir que outros tipos de curvas, além de polinômios, possam vir a ser utilizados em casos de ajustamento. Vários livros de Cálculo Numérico, assim como livros de tabelas e fórmulas matemáticas trazem listas de funções que podem ser utilizadas para o ajustamento com o Método dos Mínimos Quadrados.

Resposta à Questão 2 – São dados oito pares (x_i, y_i) , $i = 1, 2, 3, \dots, 8$ que representam oito pontos aos quais precisamos ajustar uma reta: $p(x) = a_1 + a_2 x$. Os coeficientes a_1 e a_2 da reta de ajustamento podem ser obtidos ao se resolver o sistema de equações lineares:

$$\begin{pmatrix} n & \sum x_i \\ \sum x_i & \sum x_i^2 \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \end{pmatrix} = \begin{pmatrix} \sum y_i \\ \sum y_i x_i \end{pmatrix} \quad i = 1, 2, 3, \dots, 8$$

Cálculo dos somatórios:

i	x_i	y_i	x_i^2	$x_i y_i$
1	0,0	5,1	0,0	0,0
2	1,0	5,5	1,0	5,5
3	0,5	5,1	0,25	2,55
4	0,3	5,3	0,09	1,59
5	0,3	5,2	0,09	1,56
6	0,5	5,0	0,25	2,5
7	0,7	5,4	0,49	3,78
8	0,8	5,5	0,64	4,4
Σ	4,1	42,1	2,81	21,88

De modo que o sistema linear pode ser escrito como:

$$\begin{pmatrix} 8 & 4,1 \\ 4,1 & 2,81 \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \end{pmatrix} = \begin{pmatrix} 42,1 \\ 21,88 \end{pmatrix}$$

Resolução do sistema linear pela Regra de Cramer:

Determinante principal da matriz dos coeficientes:

$$\det = (8)(2,81) - (4,1)(4,1) = 5,67$$

Primeiro determinante característico:

$$\det 1 = (42,1)(2,81) - (4,1)(21,88) = 28,593$$

Segundo determinante característico:

$$\det 2 = (8)(21,88) - (42,1)(4,1) = 2,43$$

$$a_1 = \det 1 / \det = 28,593 / 5,67 = 5,043$$

$$a_2 = \det 2 / \det = 2,43 / 5,67 = 0,429$$

Donde a reta de ajustamento: $p(x) = 5,043 + 0,429 x$, cuja utilização conduz à tabela:

k	x_k	$p(x_k)$
1	0,0	0,429
2	0,3	1,942
3	0,5	2,951
4	0,7	3,959
5	0,8	4,463
6	1,0	5,472

Resposta à Questão 3 – Nesta outra questão, pede-se para ajustar uma parábola:

$$p(x) = a_1 + a_2 x + a_3 x^2$$

a um conjunto de oito pontos. A determinação dos três coeficientes a_1 , a_2 e a_3 da parábola envolve a resolução de um sistema de três equações lineares com três incógnitas.

$$\begin{cases} na_1 + a_2 \sum x + a_3 \sum x^2 = \sum y \\ a_1 \sum x + a_2 \sum x^2 + a_3 \sum x^3 = \sum xy \\ a_1 \sum x^2 + a_2 \sum x^3 + a_3 \sum x^4 = \sum x^2 y \end{cases}$$

em que n representa a quantidade de pontos (no presente caso, oito) e o índice i, que faz variar os valores de x_i e y_i nos somatórios é tal que $i = 1, 2, 3, \dots, n$. Cálculo dos somatórios:

i	x_i	y_i	x_i^2	$x_i y_i$	x_i^3	x_i^4	$x_i^2 y_i$
1	0,0	5,1	0,0	0,0	0,0	0,0	0,0
2	1,0	5,5	1,0	5,5	1,0	1,0	5,5
3	0,5	5,1	0,25	2,55	0,125	0,0625	1,275
4	0,3	5,3	0,09	1,59	0,027	0,0081	0,477
5	0,3	5,2	0,09	1,56	0,027	0,0081	0,468
6	0,5	5,0	0,25	2,5	0,125	0,0625	0,125
7	0,7	5,4	0,49	3,78	0,343	0,2401	2,6446
8	0,8	5,5	0,64	4,4	0,512	0,4096	3,52
Σ	4,1	42,1	2,81	21,88	2,159	1,7909	14,0096

De modo que se tem o seguinte sistema de equações lineares:

$$\left\{ \begin{array}{l} n a_1 + \Sigma x a_2 + \Sigma x^2 a_3 = \Sigma y \rightarrow 8a_1 + 4,1 a_2 + 2,81 a_3 = 42,1 \\ \Sigma x a_1 + \Sigma x^2 a_2 + \Sigma x^3 a_3 = \Sigma xy \rightarrow 4,1 a_1 + 2,81 a_2 + 2,159 a_3 = 21,88 \\ \Sigma x^2 a_1 + \Sigma x^3 a_2 + \Sigma x^4 a_3 = \Sigma x^2 y \rightarrow 2,81 a_1 + 2,159 a_2 + 1,7909 a_3 = 14,0096 \end{array} \right.$$

Em que $n = 8$ é a quantidade de pontos, e $i = 1, 2, 3, \dots, 8$.

Resolvendo o sistema com o auxílio da Regra de Cramer, calculamos o determinante principal, \det , da matriz dos coeficientes e os determinantes característicos: $\det 1$; $\det 2$; e $\det 3$, correspondentes a cada incógnita:

Determinante principal: $\det =$

8	4,1	2,81
4,1	2,81	2,159
2,81	2,159	1,7909

$$\det = (8)(2,81)(1,7909) + (4,1)(2,159)(2,81) + (2,81)(2,159)(4,1) - (2,81)(2,81)(2,81) - (2,159)(2,159)(8) - (1,7909)(4,1)(4,1)$$

$$\det = 90,00711 - 89,583318$$

$$\det = 0,423792$$

Primeiro determinante característico: $\det 1 =$

42,1	4,1	2,81
21,88	2,81	2,159
14,0096	2,159	1,7909

$$\det 1 = (\mathbf{42,1})(2,81)(1,7909) + (4,1)(2,159)(\mathbf{14,0096}) + (2,81)(2,159)(\mathbf{21,88}) \\ - (2,81)(2,81)(\mathbf{14,0096}) - (2,159)(2,159)(\mathbf{42,1}) - (1,7909)(\mathbf{21,88})(4,1)$$

$$\det 1 = (211,865) + (124,012) + (132,741) - (110,621) - (196,240) - (160,658)$$

$$\det 1 = 468,618 - 467,519$$

$$\det 1 = 1,0099$$

Segundo determinante característico: $\det 2 =$

8	42,1	2,81
4,1	21,88	2,159
2,81	14,0096	1,7909

$$\det 2 = (8)(\mathbf{21,88})(1,7909) + (\mathbf{42,1})(2,159)(2,81) + (2,81)(\mathbf{14,0096})(4,1) \\ - (2,81)(\mathbf{21,88})(2,81) - (2,159)(\mathbf{14,0096})(8) - (1,7909)(4,1)(\mathbf{42,1})$$

$$\det 2 = (313,479) + (118,362) + (161,405) - (172,767) - (241,974) - (309,127)$$

$$\det 2 = (593,245) - (481,894)$$

$$\det 2 = 111,351$$

Terceiro determinante característico: $\det 3 =$

8	4,1	42,1
4,1	2,81	21,88
2,81	2,159	14,0096

$$\det 3 = (8)(2,81)(\mathbf{14,0096}) + (4,1)(\mathbf{21,88})(2,81) + (\mathbf{42,1})(2,159)(4,1) \\ - (\mathbf{42,1})(2,81)(2,81) - (\mathbf{21,88})(2,159)(8) - (\mathbf{14,0096})(4,1)(4,1)$$

$$\det 3 = (314,935) + (252,079) + (372,664) - (332,426) - (377,911) - (235,501)$$

$$\det 3 = 939,678 - 945,836$$

$$\det 3 = -6,158$$

Cálculo dos coeficientes da parábola:

$$a_1 = \det 1 / \det = 1,0099 / 0,423792 = 2,3830$$

$$a_2 = \det 2 / \det = 111,351 / 0,423792 = 262,75$$

$$a_3 = \det 3 / \det = -6,158 / 0,423792 = -14,531$$

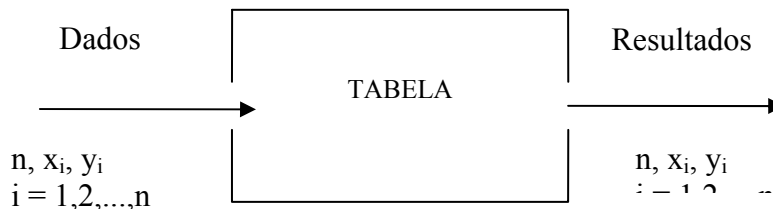
Conclusão: a parábola de ajustamento é

$$p(x) = 2,3830 + 262,75 x - 14,531 x^2$$

Resposta à Questão 4 – Construiremos um algoritmo para:

- (1) na Entrada de Dados ler n , a quantidade de pontos, e as coordenadas $[x_i, y_i]$ desses pontos;
- (2) no Processamento dos Cálculos, não há cálculos a realizar; e
- (3) na Saída dos Resultados, imprimir os valores das coordenadas lidas.

O algoritmo pode, inicialmente, e no nível mais alto de abstração, ser representado por uma caixa preta identificada como TABELA. A caixa tem duas aberturas de cada lado: na da esquerda, dá-se a entrada dos dados; e na da direita, a saída dos resultados:

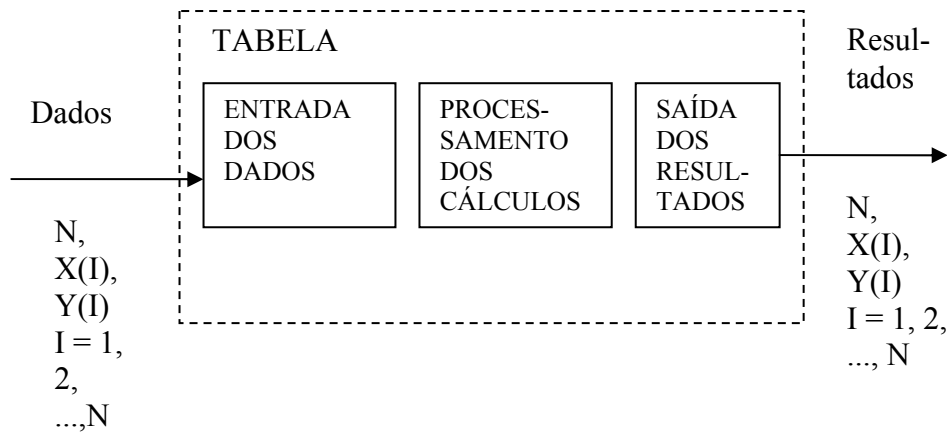


Esse esquema gráfico inicial corresponde a um trecho escrito em pseudolinguagem de programação. Tal esquema contempla já os tipos das variáveis mencionadas:

```

ALGORITMO TABELA
  INTEIRO I, N
  REAL X(50), Y(50)
INÍCIO
  .
  .
  .
FIM DO ALGORITMO TABELA.
  
```

Em uma segunda etapa, agora abrindo-se a caixa preta para visualização de sua composição interna, distinguem-se as caixas que correspondem às principais funcionalidades de um algoritmo numérico. As caixas pretas *Entrada dos Dados*, *Processamento dos Cálculos* e *Saída dos Resultados*.



A *Entrada dos Dados* pode ser detalhada como segue:

```
(* ENTRADA DOS DADOS: *)
  LEIA N
  PARA (I DE 1 ATÉ N) FAÇA
    LEIA X(I), Y(I)
  FIM PARA
(* FIM DA ENTRADA DOS DADOS. *)
```

No *Processamento dos Cálculos*, não há contas a fazer:

```
(* PROCESSAMENTO DOS CÁLCULOS: *)
  (* NÃO HÁ CONTAS A FAZER *)
(* FIM DO PROCESSAMENTO DOS CÁLCULOS. *)
```

NA *Saída dos Resultados*, manda-se escrever o valor de cada coordenada, na ordem em que foram lidas:

```
(* SAÍDA DOS RESULTADOS: *)
  PARA (I DE 1 ATÉ N) FAÇA
    ESCREVA 'X('I,') = ',X(I)
    ESCREVA 'Y('I,') = ',Y(I)
  FIM PARA
(* FIM DA SAÍDA DOS RESULTADOS. *)
```

Agora, reunindo os três blocos de detalhamentos, apresenta-se o algoritmo completo:

ALGORITMO TABELA

```
(* ESTE ALGORITMO LÊ E IMPRIME
UM CONJUNTO DE PARES DE DADOS. *)
```

```
REAL X(50), Y(50)
```

INTEIRO N, I
INÍCIO

(* ENTRADA DOS DADOS: *)
 LEIA N
 PARA (I DE 1 ATÉ N) FAÇA
 LEIA X(I), Y(I)
 FIM PARA
(* FIM DA ENTRADA DOS DADOS. *)

(* PROCESSAMENTO DOS CÁLCULOS: *)
 (* NÃO HÁ CÁLCULOS A FAZER *)
(* FIM DO PROCESSAMENTO DOS CÁLCULOS. *)

(* SAÍDA DOS RESULTADOS: *)
 PARA (I DE 1 ATÉ N) FAÇA
 ESCREVA 'X('I,') = ',X(I)
 ESCREVA 'Y('I,') = ',Y(I)
 FIM PARA
(* FIM DA SAÍDA DOS RESULTADOS. *)

FIM DO ALGORITMO TABELA.

PROVA 20 – Ajuste Polinomial

FOLHA DE QUESTÕES:

Questão 1 (dissertativa) – Estabeleça o problema do ajuste polinomial com o emprego do Método dos Mínimos Quadrados. Descreva brevemente o ajuste linear, o ajuste parabólico e o ajuste cúbico. Por favor, ocupe, com a sua descrição, pelo menos, uma página.

Questão 2 (numérica) – Considere os dados apresentados na tabela dada abaixo.

x	0,5	0,6	0,6	0,7	0,8	0,9	1,0	1,1
y	-4	-3	-2	0	-1	-3	-5	-7

Por favor, ajuste uma reta

$$p(x) = a + b x$$

a esses dados com a utilização do Método dos Mínimos Quadrados. Agora, para cada valor de x , calcule o correspondente valor $p(x)$. Finalmente, responda: qual é o valor de $p(0,4)$?

Questão 3 (numérica) – Ajuste uma parábola $p(x) = a_1 + a_2 x + a_3 x^2$ ao conjunto de pares de valores apresentados na tabela dada a seguir. Use o Método dos Mínimos Quadrados. Por favor, não deixe de indicar todas as operações efetuadas. No final, apresente a expressão da parábola com destaque.

x	0,1	0,3	0,5	0,7	0,9	0,9	1,0	1,3
y	4	8	10	0	-1	-2	-2	-3

Questão 4 (algoritmo) – Construa um algoritmo estruturado que programe o ajuste parabólico com o auxílio do Método dos Mínimos Quadrados. Por favor, faça-o em três etapas de refinamento sucessivo. A estrutura do algoritmo deve apresentar três blocos principais de instruções para; (a) entrada dos dados; (b) processamento dos cálculos; e (c) saída dos resultados.

Boa Prova!

RESPOSTAS:

Resposta à Questão 1 – Pode-se dizer que o problema do ajustamento polinomial consiste, basicamente, em determinar um polinômio $p(x)$ do grau m ,

$$p(x) = a_1 + a_2 x + a_3 x^2 + a_4 x^3 + \dots + a_{m+1} x^m$$

que represente, de modo formal, a tendência de um fenômeno descrito por um conjunto de n pares de valores $[x_i, y_i]$, $i = 1, 2, 3, \dots, n$.

Os pares de valores podem ter sido obtidos, por exemplo, em uma experiência de laboratório com uso de instrumentos de medição, seguindo uma metodologia específica de coleta de valores numéricos, ou em um levantamento de dados em campo.

O resultado obtido nesses casos constitui uma tabela de pares de valores $[x_i, y_i]$, $i = 1, 2, 3, \dots, n$ que podem ser interpretados como coordenadas de n pontos marcados em um plano cartesiano.

Conforme a disposição desses pontos no plano, pode-se escolher um ajustamento linear; nesse caso, $m = 1$, e

$$p(x) = a_1 + a_2 x$$

ou um ajuste parabólico, caso em que $m = 2$ e

$$p(x) = a_1 + a_2 x + a_3 x^2$$

ou, ainda, $m = 3$, para se ajustar uma cúbica:

$$p(x) = a_1 + a_2 x + a_3 x^2 + a_4 x^3$$

e assim por diante.

Usando-se a teoria que dá embasamento ao Método dos Mínimos Quadrados, obtém-se o polinômio que fornece o mínimo da soma dos quadrados dos desvios.

Esses desvios podem ser definidos como

$$d_i = p(x_i) - y_i, \\ i = 1, 2, 3, \dots, n.$$

A soma dos quadrados dos desvios pode ser representada como uma função Q de várias variáveis:

$$Q(a_1, a_2, a_3, \dots, a_{m+1}) = \sum (d_i)^2 = \sum [p(x_i) - y_i]^2 \\ i = 1, 2, 3, \dots, n.$$

No ponto de mínimo, todas as derivadas parciais de Q , em relação às variáveis, são nulas:

$$\left\{ \begin{array}{l} \partial Q / \partial a_1 = 0 \\ \partial Q / \partial a_2 = 0 \\ \partial Q / \partial a_3 = 0 \\ \dots \\ \partial Q / \partial a_{m+1} = 0 \end{array} \right.$$

Assim, substituindo Q pela expressão que a define e derivando-se em relação a cada uma das variáveis, obtém-se um sistema de $m+1$ equações lineares com $m+1$ incógnitas.

$$\left\{ \begin{array}{l} n a_1 + a_2 \sum x_i + \dots + a_{m+1} \sum x_i^m = \sum y_i \\ a_1 \sum x_i + a_2 \sum x_i^2 + \dots + a_{m+1} \sum x_i^{m+1} = \sum x_i y_i \\ \vdots \\ a_1 \sum x_i^m + a_2 \sum x_i^{m+1} + \dots + a_{m+1} \sum x_i^{2m} = \sum x_i^m y_i \end{array} \right.$$

$$i = 1, 2, 3, \dots, n.$$

A resolução desse sistema linear fornece os $m+1$ valores dos coeficientes do polinômio $p(x)$, do grau m , escolhido para o ajustamento.

Resposta à Questão 2 – Nesta questão, o valor de n , que representa a quantidade de pontos da tabela, é 8. Como o ajuste é linear, o valor de m , que representa o grau do polinômio de ajustamento, é 1. Para se obter a reta de ajustamento $p(x) = a + bx$, resolve-se o sistema de duas equações lineares com duas incógnitas:

$$\left\{ \begin{array}{l} n a + b \sum x_i = \sum y_i \\ a \sum x_i + b \sum x_i^2 = \sum x_i y_i \\ i = 1, 2, 3, \dots n. \end{array} \right.$$

A obtenção dos somatórios pode ser realizada com o auxílio de uma tabela:

i	x_i	y_i	x_i^2	$x_i y_i$
1	0,5	-4	0,25	- 2,0
2	0,6	-3	0,36	- 1,8
3	0,6	-2	0,36	- 1,2
4	0,7	0	0,49	0,0
5	0,8	-1	0,64	- 0,8
6	0,9	-3	0,81	- 2,7
7	1,0	-5	1,00	-5,0
8	1,1	-7	1,21	- 7,7
-	$\sum x_i = 6,2$	$\sum y_i = - 25$	$\sum x_i^2 = 5,12$	$\sum x_i y_i = - 21,2$

Introduzindo os valores dos somatórios tem-se, então, o sistema:

$$\begin{cases} (8) a + (6,2) b = - 25 \\ (6,2) a + (5,12) b = - 21,2 \end{cases}$$

Resolvendo o sistema com a Regra de Cramer,

$$\text{Det} = (8)(5,12) - (6,2)(6,2) = 40,96 - 38,44 = 2,52$$

$$\text{DetA} = (-25)(5,12) - (6,2)(- 21,2) = -128 - (-131,44) = 3,44$$

$$\text{DetB} = (8)(- 21,2) - (-25)(6,2) = - 169,6 - (- 155) = - 14,6$$

$$a = \text{DetA}/\text{Det} = 3,44 / 2,52 = 1,365$$

$$b = \text{DetB} / \text{Det} = - 14,6 / 2,52 = - 5,794$$

O polinômio de ajustamento nesse caso é

$$p(x) = 1,365 - 5,794 x$$

Cálculo dos valores de $p(x_k)$, considerando que, na tabela originalmente dada, havia repetição de abscissas, agora eliminada.

$$k = 1, 2, 3, \dots 7$$

k	1	2	3	4	5	6	7
x_k	0,5	0,6	0,7	0,8	0,9	1,0	1,1
$p(x_k)$	-1,532	-2,111	-2,691	-3,270	-3,850	-4,429	-5,008

Para completar a respsta, $p(0,4) = 1,365 - 5,794(0,4) = -0,9526$

Resposta à Questão 3 – Pede-se, nesta questão, para utilizar o Método dos Mínimos Quadrados no ajuste de uma parábola $p(x) = a_1 + a_2 x + a_3 x^2$ a um conjunto de oito pares de valores apresentados em uma tabela. Para determinar os coeficientes a_1 , a_2 e a_3 da parábola, vamos resolver o sistema de três equações lineares com três incógnitas:

$$\begin{cases} n a_1 + a_2 \sum x_i + a_3 \sum x_i^2 = \sum y_i \\ a_1 \sum x_i + a_2 \sum x_i^2 + a_3 \sum x_i^3 = \sum x_i y_i \\ a_1 \sum x_i^2 + a_2 \sum x_i^3 + a_3 \sum x_i^4 = \sum x_i^2 y_i \end{cases}$$

Para obter os valores dos somatórios, construiremos uma tabela.

i	x_i	y_i	x_i^2	x_i^3	x_i^4	$x_i y_i$	$x_i^2 y_i$
1	0,1	4	0,01	0,001	0,0001	0,4	0,04
2	0,3	8	0,09	0,027	0,0081	2,4	0,72
3	0,5	10	0,25	0,125	0,0625	5,0	2,50
4	0,7	0	0,49	0,343	0,2401	0,0	0,00
5	0,9	-1	0,81	0,729	0,6561	-0,9	-0,81
6	0,9	-2	0,81	0,729	0,6561	-1,8	-1,62
7	1,0	-2	1,00	1,000	1,0000	-2,0	-2,00
8	1,3	-3	1,69	2,197	2,8561	-3,9	-5,07
$\Sigma \rightarrow$	5,7	14	5,15	5,151	5,4791	-0,8	-6,24

Agora, construindo o sistema de equações lineares que permite obter os valores dos coeficientes do polinômio de segundo grau que corresponde à parábola de ajustamento:

$$\left\{ \begin{array}{l} 8 a_1 + 5,7 a_2 + 5,15 a_3 = 14 \\ 5,7 a_1 + 5,15 a_2 + 5,151 a_3 = -0,8 \\ 5,15 a_1 + 5,151 a_2 + 5,4791 a_3 = -6,24 \end{array} \right.$$

Resolução do sistema com o emprego da Regra de Cramer:

$$\begin{aligned} \text{Det} &= (8)(5,15)(5,4791) + (5,7)(5,151)(5,15) + (5,15)(5,151)(5,7) \\ &\quad - (5,15)(5,15)(5,15) - (5,151)(5,151)(8) - (5,4791)(5,7)(5,7) \\ \text{Det} &= 225,73892 + 151,2076 + 151,2076 - 136,59087 - 212,2624 - 178,01595 \\ \text{Det} &= 528,15413 - 526,86924 = 1,284888 \end{aligned}$$

$$\begin{aligned} \text{Det1} &= (14)(5,15)(5,4791) + (5,7)(5,151)(-6,24) + (5,15)(5,151)(-0,8) \\ &\quad - (5,15)(5,15)(-6,24) - (5,151)(5,151)(14) - (5,4791)(-0,8)(5,7) \\ \text{Det1} &= 190,61022 - 180,97411 = 9,636102 \end{aligned}$$

$$\begin{aligned} \text{Det2} &= (8)(-0,8)(5,4791) + (14)(5,151)(5,15) + (5,15)(-6,24)(5,7) \\ &\quad - (5,15)(-0,8)(5,15) - (5,151)(-6,24)(8) - (5,4791)(5,7)(14) \\ \text{Det2} &= 153,14566 - 158,87626 = -5,7306 \end{aligned}$$

$$\begin{aligned} \text{Det3} &= (8)(5,15)(-6,24) + (5,7)(-0,8)(5,15) + (14)(5,151)(5,7) \\ &\quad - (14)(5,15)(5,15) - (-0,8)(5,151)(8) - (-6,24)(5,7)(5,7) \\ \text{Det3} &= 130,4778 - 135,611 = -5,1332 \end{aligned}$$

Então:

$$a_1 = \text{Det1}/\text{Det} = 9,636102/ 1,284888 = 7,4995657 \approx 7,5$$

$$a_2 = \text{Det2}/\text{Det} = - 5,7306/ 1,284888 = - 4,4599996 \approx - 4,5$$

$$a_3 = \text{Det3}/\text{Det} = - 5,1332/ 1,284888 = - 3,9950563 \approx - 4,0$$

E a parábola de ajustamento é $p(x) = 7,5 - 4,5 x - 4,0 x^2$

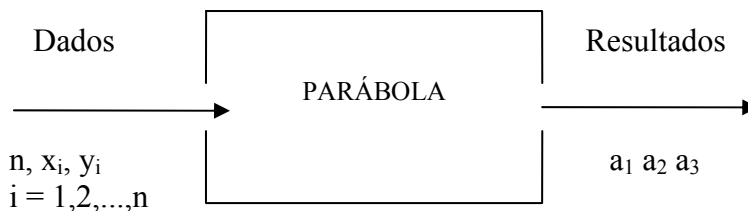
Resposta à Questão 4 – Vamos construir um algoritmo para:

(1) na Entrada de Dados ler n (a quantidade de pontos aos quais ajustaremos uma parábola), e as coordenadas $[x_i, y_i]$ desses pontos;

(2) no Processamento dos Cálculos, obtém-se os somatórios, depois os determinantes e, finalmente, os coeficientes da parábola; e

(3) na Saída dos Resultados, mandam-se imprimir os valores dos coeficientes calculados.

Inicialmente, representa-se o algoritmo, graficamente, e no nível mais alto de abstração, como uma caixa preta identificada como PARÁBOLA. A caixa tem duas aberturas: na da esquerda, dá-se a entrada dos dados; e, na da direita, a saída dos resultados:

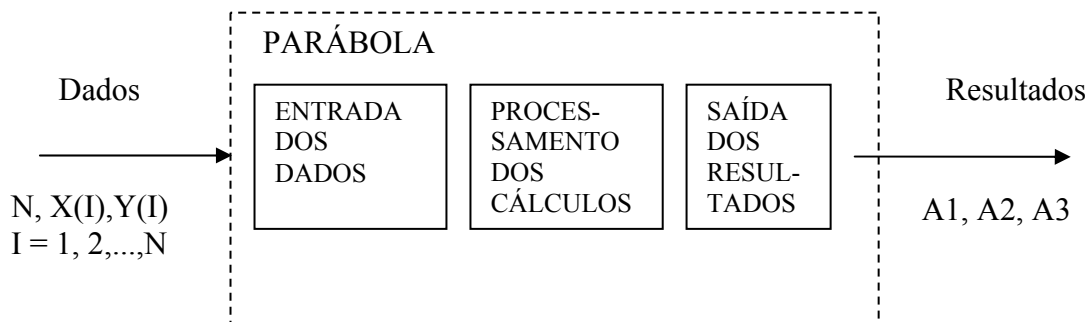


O esquema gráfico inicial corresponde a um trecho escrito em pseudolinguagem de programação. Tal esquema já define os tipos das variáveis relacionadas na entrada dos dados e na saída dos resultados:

```
ALGORITMO PARÁBOLA
    INTEIRO I, N
    REAL X(50), Y(50)
    REAL A1, A2, A3
INÍCIO
    ...
FIM DO ALGORITMO PARÁBOLA.
```

Em uma segunda etapa, abre-se a caixa preta para visualização de sua composição interna, ainda usando um esquema gráfico. Distinguem-se então três caixas pretas. Elas correspondem aos blocos de instruções das principais funcionalidades de

um algoritmo numérico. As caixas pretas são: *Entrada dos Dados*, *Processamento dos Cálculos* e *Saída dos Resultados*.



A *Entrada dos Dados* pode ser detalhada como segue:

```

(* ENTRADA DOS DADOS: *)
  LEIA N
  PARA (I DE 1 ATÉ N) FAÇA
    LEIA X(I), Y(I)
  FIM PARA
(* FIM DA ENTRADA DOS DADOS. *)
  
```

No *Processamento dos Cálculos*,

```

(* PROCESSAMENTO DOS CÁLCULOS: *)
  (* CÁLCULO DOS SOMATÓRIOS: *)
    SX = 0; SX2 = 0; SX3 = 0; SX4 = 0
    SY = 0; SXY = 0; SX2Y = 0
    PARA (I DE 1 ATÉ N) FAÇA
      SX = SX + X(I)
      SX2 = SX2 + X(I)*X(I)
      SX3 = SX3 + X(I)*X(I)*X(I)
      SX4 = SX4 + X(I)*X(I)*X(I)*X(I)
      SY = SY + Y(I)
      SXY = SXY + X(I)*Y(I)
      SX2Y = SX2Y + X(I)*X(I)*Y(I)
    FIM PARA
  (* FIM DO CÁLCULO DOS SOMATÓRIOS. *)
  (* CÁLCULO DOS DETERMINANTES: *)
    DET = N*SX2*SX4 + SX*SX3*SX2 + SX2*SX3*SX
      - SX2*SX2*SX2 - SX3*SX3*N - SX4*SX*SX
    DET1 = SY*SX2*SX4 + SX*SX3*SX2Y + SX2*SX3*SXY
      - SX2*SX2*SX2Y - SX3*SX3*SY - SX4*SXY*SX
    DET2 = N*SXY*SX4 + SY*SX3*SX2 + SX2*SX2Y*SX
      - SX2*SXY*SX2 - SX3*SX2Y*N - SX4*SX*SY
    DET3 = N*SX2*SX2Y + SX*SXY*SX2 + SY*SX3*SX
      - SY*SX2*SX2 - SXY*SX3*N - SX2Y*SX*SX
  (* FIM DO CÁLCULO DOS DETERMINANTES. *)
  (* USO DA REGRA DE CRAMER: *)
  
```

A1 = DET1/DET; A2 = DET2/DET; A3 = DET3/DET
(* FIM DO USO DA REGRA DE CRAMER. *)

(* FIM DO PROCESSAMENTO DOS CÁLCULOS. *)

Na *Saída dos Resultados*, manda-se escrever o valor de cada coordenada, na ordem em que foram lidas:

(* SAÍDA DOS RESULTADOS: *)
ESCREVA 'P(X) = A1 + A2 *X + A3 *X*X'
ESCREVA 'A1 = ', A1
ESCREVA 'A2 = ', A2
ESCREVA 'A3 = ', A3
(* FIM DA SAÍDA DOS RESULTADOS. *)

Reunindo os três blocos de detalhamentos, e definindo-se os tipos das variáveis utilizadas para os somatórios e determinantes, tem-se o algoritmo completo:

ALGORITMO PARÁBOLA

(* AJUSTA UMA PARÁBOLA A UM CONJUNTO
DE PARES DE DADOS COM O EMPREGO DO
MÉTODO DOS MÍNIMOS QUADRADOS. *)

REAL X(50), Y(50)
REAL A1, A2, A3
REAL SX, SX2, SX3, SX4
REAL SY, SXY, SX2Y
INTEIRO I, N

INÍCIO

(* ENTRADA DOS DADOS: *)

LEIA N
PARA (I DE 1 ATÉ N) FAÇA
LEIA X(I), Y(I)
FIM PARA

(* FIM DA DENTRADA DOS DADOS. *)

(* PROCESSAMENTO DOS CÁLCULOS: *)

(* CÁLCULO DOS SOMATÓRIOS: *)
SX = 0; SX2 = 0; SX3 = 0; SX4 = 0
SY = 0; SXY = 0; SX2Y = 0
PARA (I DE 1 ATÉ N) FAÇA
SX = SX + X(I)
SX2 = SX2 + X(I)*X(I)
SX3 = SX3 + X(I)*X(I)*X(I)
SX4 = SX4 + X(I)*X(I)*X(I)*X(I)
SY = SY + Y(I)
SXY = SXY + X(I)*Y(I)
SX2Y = SX2Y + X(I)*X(I)*Y(I)

```

FIM PARA
(* FIM DO CÁLCULO DOS SOMATÓRIOS. *)
(* CÁLCULO DOS DETERMINANTES: *)
DET = N*SX2*SX4 + SX*SX3*SX2 + SX2*SX3*SX
      - SX2*SX2*SX2 - SX3*SX3*N - SX4*SX*SX
DET1 = SY*SX2*SX4 + SX*SX3*SX2Y + SX2*SX3*SXY
      - SX2*SX2*SX2Y - SX3*SX3*SY - SX4*SXY*SX
DET2 = N*SXY*SX4 + SY*SX3*SX2 + SX2*SX2Y*SX
      - SX2*SXY*SX2 - SX3*SX2Y*N - SX4*SX*SY
DET3 = N*SX2*SX2Y + SX*SXY*SX2 + SY*SX3*SX
      - SY*SX2*SX2 - SXY*SX3*N - SX2Y*SX*SX
(* FIM DO CÁLCULO DOS DETERMINANTES. *)
(* USO DA REGRA DE CRAMER: *)
A1 = DET1/DET; A2 = DET2/DET; A3 = DET3/DET
(* FIM DO USO DA REGRA DE CRAMER. *)
(* FIM DO PROCESSAMENTO DOS CÁLCULOS. *)

(* SAÍDA DOS RESULTADOS: *)
ESCREVA 'P(X) = A1 + A2 *X + A3 *X*X'
ESCREVA 'A1 = ', A1
ESCREVA 'A2 = ', A2
ESCREVA 'A3 = ', A3
(* FIM DA SAÍDA DOS RESULTADOS. *)

FIM DO ALGORITMO PARÁBOLA.

```

PROVA 21 – Ajuste Não-Polinomial

FOLHA DE QUESTÕES:

Questão 1 (dissertativa) – Apresente o problema do ajuste não-polinomial. Como esse problema pode ser resolvido com a utilização do Método dos Mínimos Quadrados? Por favor, não escreva menos do que uma página.

Questão 2 (numérica) – Ajuste a curva exponencial $Y = AB^x$ aos dados apresentados na tabela dada a seguir. Após obter os valores de A e B, use esses valores para calcular as ordenadas correspondentes aos valores das abscissas. Por favor, não deixe de indicar todas as operações efetuadas.

x	1,3	2,8	5,9	7,8
y	1,2	5,9	15,9	30,4

Questão 3 (numérica) – Faça o ajustamento da curva $Y = Ae^{Bx}$ aos dados apresentados na tabela abaixo. Use a curva de ajustamento para obter a ordenada correspondente a $x = 4,5$. Por favor, indique todas as operações efetuadas.

x	2,2	3,9	4,6	5,8	6,6	7,8	8,1	9,0	10,0
y	4,0	9,8	11,5	32,7	55,0	70,3	90,8	100,0	99,0

Questão 4 (algoritmo) – Construa um algoritmo para ajustar a curva $Y = AB^x$ a um conjunto de dados. Desenvolva o seu algoritmo em três fases de refinamento sucessivo. A versão final do algoritmo deve reunir todas as partes que tenham sido elaboradas em separado.

Boa Prova!

RESPOSTAS:

Resposta à Questão 1 – O problema do ajustamento não polinomial pode ser apresentado como aquele de se definir uma determinada curva de natureza algébrica ou transcendente, por exemplo, exponencial, ou logarítmica ou outra, mas não polinomial, que se pode ajustar a um conjunto de n pontos dados pelas suas coordenadas cartesianas: $[x_i, y_i]$, com i assumindo os valores $1, 2, 3, \dots, n$.

Várias curvas podem ser utilizadas para o fim do ajustamento não polinomial com o auxílio do Método dos Mínimos Quadrados. Muitos livros de Cálculo Numérico, de Estatística e de fórmulas e tabelas matemáticas oferecem uma coleção bastante extensa de tais funções.

Algumas dessas funções são, por exemplo:

$Y = AB^x$ (neste caso, tem-se uma função exponencial); e

$Y = Ae^{Bx}$ (neste outro caso, e é a base dos logaritmos naturais: $e = 2,718281828\dots$).

Para ajustar a primeira dessas funções podem-se aplicar logaritmos decimais no seguinte sentido:

$$\log Y = \log (AB^x)$$

$$\log Y = \log A + (\log B)x$$

Agora, $p(x)$ pode ser associado a $\log Y$; já, $\log A$ (uma constante) é associado a a_1 e $\log B$ (outra constante), a a_2 .

Tem-se, então, $p(x) = a_1 + a_2 x$, recaindo-se no caso de ajustamento polinomial, neste caso, de uma reta. Usa-se, então, o Método dos Mínimos Quadrados para o cálculo de a_1 e a_2 .

Para ajustar a segunda função, podem-se aplicar logaritmos naturais (ou neperianos):

$$\ln Y = \ln (Ae^{Bx})$$

$$\ln Y = \ln A + (\ln e) Bx, \text{ como } \ln e = 1,$$

$$\ln Y = \ln A + Bx$$

De modo semelhante ao que foi feito para o primeiro exemplo, novamente, $p(x)$ pode ser associado a $\ln Y$; $\ln A$ é associado a a_1 e B , a a_2 .

Tem-se, então, $p(x) = a_1 + a_2 x$, recaindo-se, também neste caso, no ajustamento polinomial de uma reta em que o Método dos Mínimos Quadrados é empregado para a obtenção dos valores de a_1 e a_2 .

Resposta à Questão 2 – Deseja-se ajustar a curva exponencial $Y = AB^x$ aos dados da tabela

x	1,3	2,8	5,9	7,8
y	1,2	5,9	15,9	30,4

Aplicando logaritmos decimais à expressão $Y = AB^x$, tem-se

$$\log Y = \log A + x \log B$$

ou

$$p(x) = a_1 + a_2 x$$

$$\text{Então } A = 10^{a_1} \text{ e } B = 10^{a_2}$$

Para obter a_1 e a_2 , resolvemos o sistema de duas equações lineares com duas incógnitas:

$$\begin{cases} n a_1 + a_2 \sum x = \sum \log y \\ a_1 \sum x + a_2 \sum x^2 = \sum x \log y \end{cases}$$

em que $n = 4$, corresponde à quantidade de pontos definidos na tabela. Para calcular os somatórios, construímos uma tabela.

i	x_i	y_i	log y_i	x log y_i	x_i²
1	1,3	1,20	0,079181	0,102935	1,690
2	2,8	5,90	0,770852	2,158386	7,840
3	5,9	15,9	1,201397	7,088242	34,81
4	7,8	30,4	1,482874	11,56642	60,84
Σ →	17,8	-	3,534304	20,91598	105,18

Temos então o sistema:

$$\begin{cases} 4 a_1 + 17,8 a_2 = 3,534304 \\ 17,8 a_1 + 105,18 a_2 = 20,91598 \end{cases}$$

Cuja resolução pela regra de Cramer é apresentada em seguida:

$$\text{Det} = (4)(105,18) - (17,8)(17,8) = 420,72 - 316,84 = 103,88$$

$$\text{Det1} = (3,534304)(105,18) - (17,8)(20,91598) = 371,738 - 372,304 = -0,5663$$

$$\text{Det2} = (4)(20,91598) - (3,534304)(17,8) = 83,6639 - 62,9106 = 20,753$$

De modo que

$$a_1 = \text{Det1} / \text{Det} = -0,5663/103,88 = -0,0054515$$

$$a_2 = \text{Det2} / \text{Det} = 20,753/103,88 = 0,19978$$

$$A = 10^{a_1} = 10^{-0,0054515} = 0,9875 \text{ e}$$

$$B = 10^{a_2} = 10^{0,19978} = 1,5841$$

$$\text{Conclusão: } Y = (0,9875)(1,5841)^x$$

Resposta à Questão 3 – Nesta questão, pede-se para fazer o ajustamento da curva

$$Y = Ae^{Bx}$$

aos dados da tabela abaixo. Além disso, pede-se para usar a curva de ajustamento e obter a ordenada correspondente a $x = 4,5$.

x	2,2	3,9	4,6	5,8	6,6	7,8	8,1	9,0	10,0
y	4,0	9,8	11,5	32,7	55,0	70,3	90,8	100,0	99,0

Empregando logaritmos naturais à expressão $Y = Ae^{Bx}$, obtemos

$$\ln Y = \ln A + Bx,$$

isto é, podemos proceder como se fizéssemos o ajustamento da reta $p(x) = a_1 + a_2x$, em que $a_1 = \ln A$ e $a_2 = B$. Resolvemos o sistema de duas equações lineares com duas incógnitas descrito abaixo:

$$\begin{cases} n a_1 + a_2 \sum x = \sum \ln y \\ a_1 \sum x + a_2 \sum x^2 = \sum x \ln y \end{cases}$$

em que $n = 9$ é a quantidade de pontos a considerar. Para obter os somatórios, construímos uma tabela.

i	x_i	y_i	$\ln y_i$	$x \ln y_i$	x_i^2
1	2,20	4,0	1,386	3,0492	4,84
2	3,90	9,8	2,282	8,8998	15,21
3	4,60	11,5	2,442	11,2332	21,16
4	5,80	32,7	3,487	20,2246	33,64
5	6,60	55,0	4,007	26,4462	43,56
6	7,80	70,3	4,253	33,1734	60,84
7	8,10	90,8	4,509	36,5229	65,61
8	9,00	100,0	4,605	40,5810	81,00
9	10,0	99,0	4,595	45,9500	100,0
$\Sigma \rightarrow$	58,00	473,1	31,567	226,080	344,86

Substituindo os valores dos somatórios no sistema:

$$\begin{cases} 9 a_1 + 58 a_2 = 31,567 \\ 58 a_1 + 344,86 a_2 = 226,08 \end{cases}$$

Agora, podemos resolver o sistema linear com o auxílio da Regra de Cramer. Para isso, precisamos calcular três determinantes:

Determinante principal:

$$\text{Det} = (9)(344,86) - (58)(58) = 3103,74 - 3364 = -260,26$$

Determinantes característicos:

$$\text{Det1} = (31,567)(344,86) - (58)(226,08) = 10886,2 - 13112,64 = -2226,44$$

$$\text{Det2} = (9)(226,08) - (31,567)(58) = 2034,72 - 1830,886 = 203,834$$

Cálculo dos coeficientes:

$$a_1 = \text{Det1}/\text{Det} = -2226,44/-260,26 = 8,554676$$

$$a_2 = \text{Det2}/\text{Det} = 203,834/-260,26 = -0,7831937$$

Então

$$A = e^{a_1} = e^{8,554676} = 5190,9707;$$

$$B = a_2 = -0,7831937$$

$$\text{Conclusão: } Y = Ae^{Bx} = 5190,9707e^{-0,7831937x}$$

Para $x = 4,5$, temos:

$$Y_{4,5} = Ae^{B(4,5)} = 5190,9707[e^{-0,7831937(4,5)}]$$

$$Y_{4,5} = 5190,9707[e^{-3,52437165}] = 5190,9707[0,029470319] = 152,97956$$

Resposta à Questão 4 – Nesta resposta vamos construir um algoritmo para ajustar a curva exponencial do tipo $Y = AB^x$ a um conjunto de n pares de dados. Desenvolveremos o algoritmo em três fases de refinamento sucessivo. A versão final do algoritmo reunirá todas as partes que tenham sido elaboradas em separado.

Fase 1: Inicialmente, representa-se o algoritmo, no nível mais alto de abstração, como uma caixa preta identificada como AJUST_EXP. A caixa tem duas aberturas: na da esquerda, dá-se a entrada dos dados; e, na da direita, a saída dos resultados:

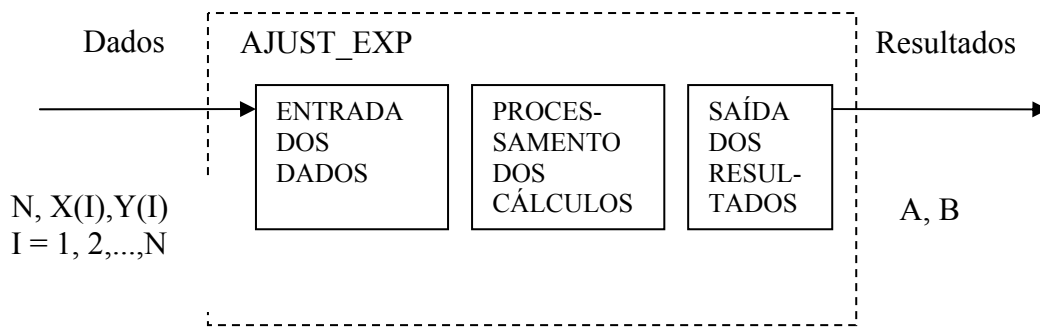


O esquema gráfico inicial corresponde a um trecho escrito em pseudolinguagem de programação. Tal esquema já define os tipos das variáveis relacionadas na entrada dos dados e na saída dos resultados:

```

ALGORITMO AJUST_EXP
  INTEIRO I, N
  REAL X(20), Y(20)
  REAL A, B
INÍCIO
  .
  .
  .
FIM DO ALGORITMO AJUST_EXP
  
```

Fase 2: abre-se a caixa preta para visualização de sua composição interna. Distinguem-se então três novas caixas pretas. Elas correspondem aos blocos de instruções das principais funcionalidades de um algoritmo numérico. As caixas pretas são: *Entrada dos Dados*, *Processamento dos Cálculos* e *Saída dos Resultados*.



A Entrada dos Dados pode ser detalhada como segue:

(* ENTRADA DOS DADOS: *)

```

LEIA N
PARA (I DE 1 ATÉ N) FAÇA
    LEIA X(I), Y(I)
FIM PARA

```

(* FIM DA ENTRADA DOS DADOS. *)

Já, o Processamento dos Cálculos pode ser detalhado como:

(* PROCESSAMENTO DOS CÁLCULOS: *)

```

SX = 0; SX2 = 0; SLNY = 0; SXLNY = 0
PARA (I DE 1 ATÉ N) FAÇA
    SX = SX + X(I)
    SX2 = SX2 + X(I)*X(I)
    SLNY = SLNY + LN(Y(I))
    SXLNY = SXLNY + X(I)*LN(Y(I))
FIM PARA
DET = N*SX2 - SX*SX
DET1 = SLNY*SX2 - SX*SXLNY
DET2 = N*SXLNY - SLNY*SX
A1 = DET1/DET
A2 = DET2/DET
A = e^A1
B = A2

```

(* FIM DO PROCESSAMENTO DOS CÁLCULOS. *)

Em que e é a base dos logaritmos naturais.

Agora, detalhamos a Saída dos Resultados:

(* SAÍDA DOS RESULTADOS: *)

```

ESCREVA 'Y = Ae^BX'
ESCREVA 'A = ', A
ESCREVA 'B = ', B

```

(* FIM DA SAÍDA DOS RESULTADOS. *)

Fase 3: Finalmente, reunindo todas as partes desenvolvidas em separado e incluindo a definição dos tipos de todas as variáveis utilizadas:

ALGORITMO AJUST_EXP

(AJUSTA FUNÇÃO EXPONENCIAL A
UM CONJUNTO DE PARES DE DADOS. *)*

REAL X(20), Y(20)
REAL A, B, A1, A2, DET, DET1, DET2
REAL SX, SX2, SLNY, SXLNY
INTEIRO I,N

INÍCIO

(* ENTRADA DOS DADOS: *)

LEIA N
PARA (I DE 1 ATÉ N) FAÇA
 LEIA X(I), Y(I)
FIM PARA

(* FIM DA ENTRADA DOS DADOS. *)

(* PROCESSAMENTO DOS CÁLCULOS: *)

SX = 0; SX2 = 0; SLNY = 0; SXLNY = 0
PARA (I DE 1 ATÉ N) FAÇA
 SX = SX + X(I)
 SX2 = SX2 + X(I)*X(I)
 SLNY = SLNY + LN(Y(I))
 SXLNY = SXLNY + X(I)*LN(Y(I))
FIM PARA

DET = N*SX2 – SX*SX
DET1 = SLNY*SX2 – SX*SXLNY
DET2 = N*SXLNY – SLNY*SX
A1 = DET1/DET
A2 = DET2/DET
A = e^A1
B = A2

(* FIM DO PROCESSAMENTO DOS CÁLCULOS. *)

(* SAÍDA DOS RESULTADOS: *)

ESCREVA 'Y = Ae^BX'
ESCREVA 'A = ', A
ESCREVA 'B = ', B

(* FIM DA SAÍDA DOS RESULTADOS. *)

FIM DO ALGORITMO AJUST_EXP.

PROVA 22 – Interpolação Lagrangeana

FOLHA DE QUESTÕES:

Questão 1 (dissertativa) – Diga como se dá o emprego dos *Polinômios de Lagrange* na resolução do problema da interpolação. Se achar adequado, construa gráficos e dê exemplos que ilustrem a sua apresentação. Por favor, ocupe pelo menos uma página com a sua descrição.

Questão 2 (numérica) - Use a *interpolação lagrangeana* para obter $f(2,1)$ a partir dos dados apresentados a seguir. Por favor, indique claramente todas as operações efetuadas. Não deixe de avaliar a adequação do resultado obtido.

x	2,00	2,30	2,50
y = f(x)	-1,00	0,00	-2,00

Questão 3 (numérica) – Considere a tabela dada abaixo. Obtenha $f(1,35)$ com os *Polinômios de Lagrange*. No final, responda à seguinte pergunta: o valor obtido é razoável no contexto dos dados?

x	1,00	1,50	2,00
y = f(x)	4,00	3,00	2,00

Questão 4 (algoritmo) – Construa um algoritmo para implementar computacionalmente um caso de interpolação com o emprego dos *Polinômios de Lagrange*. Faça-o em sucessivas etapas de refinamento, introduzindo novos detalhes a partir de uma caixa preta inicial.

Boa Prova!

RESPOSTAS:

Resposta à Questão 1 – As práticas que envolvem o emprego da interpolação polinomial surgem freqüentemente em todas as áreas científicas e técnicas. É assim na Física, por exemplo, e em todas as Engenharias.

Basicamente, a interpolação polinomial consiste no uso de uma técnica para se obter o valor de uma função $y = f(x)$ correspondente a um ou mais valores particulares de x . O valor de y é obtido com polinômios.

A necessidade de se realizar a interpolação ocorre porque, às vezes, não se conhece a expressão algébrica de $f(x)$. Em outras ocasiões, a necessidade se dá porque não se deseja utilizar a expressão algébrica de $f(x)$: esse é o caso, pode-se dizer, quando $f(x)$ tem um aspecto muito complicado, seu cálculo envolve muitas operações ou leva a pouca eficiência computacional.

Para se realizar a interpolação foram desenvolvidas várias técnicas que possuem características adequadas para o tratamento automático em programas de computador. Esse é o caso do emprego dos Polinômios de Lagrange, dos Polinômios de Gregory-Newton e da Teoria de Splines.

Consideremos uma tabela como a que segue:

i	x_i	$y_i = f(x_i)$
0	x_0	y_0
1	x_1	y_1
2	x_2	y_2
.	.	.
.	.	.
.	.	.
n	x_n	y_n

Nessa tabela tem-se um conjunto de $n+1$ pontos dados pelas suas coordenadas cartesianas $[x_i, y_i = f(x_i)]$, $i = 0, 1, 2, \dots, n$. Nessas circunstâncias, um caso de interpolação consiste em se determinar $y_{\text{barra}} = f(x_{\text{barra}})$ correspondente a um valor x_{barra} conhecido mas não pertencente à tabela. Calcula-se o valor de y_{barra} a partir de um polinômio $p(x)$ cujo gráfico passe necessariamente por todos os pontos da tabela. Como resultado, obtém-se $y_{\text{barra}} \approx p(x_{\text{barra}})$.

Esse problema pode ser resolvido com qualquer uma das técnicas já mencionadas. E, independentemente da forma utilizada, o polinômio interpolante é único. Cabe, entretanto, escolher preliminarmente quantos pontos da tabela serão considerados para a definição do polinômio interpolante. Se apenas dois pontos forem escolhidos, então a interpolação será linear; se três pontos não alinhados, então a interpolação será parabólica, e assim por diante.

No caso de utilização dos Polinômios de Lagrange, o polinômio interpolante (ou interpolador) pode ser escrito, para o caso de $n+1$ pontos, como:

$$p(x) = y_0L_0(x) + y_1L_1(x) + y_2L_2(x) + \dots + y_nL_n(x)$$

em que $L_0, L_1, L_2, \dots, L_n$ representam os Polinômios de Lagrange. Esses polinômios têm a seguinte expressão algébrica:

$$L_k(x) = \frac{\{(x-x_0)(x-x_1)\dots(x-x_{k-1})(x-x_{k+1})\dots(x-x_n)\}}{\{(x_k-x_0)(x_k-x_1)\dots(x_k-x_{k-1})(x_k-x_{k+1})\dots(x_k-x_n)\}}$$

Em que $k = 0, 1, 2, \dots, n$.

Resposta à Questão 2 – Considerando a tabela dada a seguir, vamos obter o valor de $f(2,1)$ com o emprego dos Polinômios de Lagrange.

x	2,00	2,30	2,50
y = f(x)	-1,00	0,00	-2,00

Para tal, vamos admitir a aproximação $f(2,1) \approx p(2,1)$, em que $p(x)$ é o polinômio interpolador dado na forma:

$$p(x) = y_0L_0(x) + y_1L_1(x) + y_2L_2(x) + \dots + y_nL_n(x)$$

De modo que

$$f(2,1) \approx p(2,1) = y_0L_0(2,1) + y_1L_1(2,1) + y_2L_2(2,1) + \dots + y_nL_n(2,1)$$

Os símbolos $L_0, L_1, L_2, \dots, L_n$ representam Polinômios de Lagrange.

Ora, tendo-se um conjunto de pontos (x_i, y_i) e em que $i = 0, 1, 2, \dots, n$, a expressão geral de um Polinômio de Lagrange pode ser escrita como:

$$L_k(x) = \frac{\{(x-x_0)(x-x_1)\dots(x-x_{k-1})(x-x_{k+1})\dots(x-x_n)\}}{\{(x_k-x_0)(x_k-x_1)\dots(x_k-x_{k-1})(x_k-x_{k+1})\dots(x_k-x_n)\}}$$

Em que os valores de k podem ser dados por:

$$k = 0, 1, 2, \dots, n.$$

Para tornar os dados coerentes com a notação que estamos usando, reescrevemos a tabela já apresentada antes, introduzindo uma linha com os valores do índice i :

i	0	1	2
x_i	2,00	2,30	2,50
$y_i = f(x_i)$	-1,00	0,00	-2,00

No presente caso, $i = 0, 1, 2$. Então, temos três Polinômios de Lagrange: L_0 , L_1 e L_2 :

$$L_0(x) = \{(x-x_1)(x-x_2)\} / \{(x_0-x_1)(x_0-x_2)\}$$

$$L_0(2,1) = \{(2,1-2,30)(2,1-2,50)\} / \{(2,00-2,30)(2,00-2,50)\}$$

$$L_0(2,1) = \{(-0,2)(-0,4)\} / \{(-0,3)(-0,5)\} = \{0,08\} / \{0,15\} \approx 0,53$$

$$L_1(x) = \{(x-x_0)(x-x_2)\} / \{(x_1-x_0)(x_1-x_2)\}$$

$$L_1(2,1) = \{(2,1-2,00)(2,1-2,50)\} / \{(2,30-2,00)(2,30-2,50)\}$$

$$L_1(2,1) = \{(0,1)(-0,4)\} / \{(0,3)(-0,2)\} = \{-0,04\} / \{-0,06\} \approx 0,7$$

$$L_2(x) = \{(x-x_0)(x-x_1)\} / \{(x_2-x_0)(x_2-x_1)\}$$

$$L_2(2,1) = \{(2,1-2,00)(2,1-2,30)\} / \{(2,50-2,00)(2,50-2,30)\}$$

$$L_2(2,1) = \{(0,1)(-0,2)\} / \{(0,5)(0,2)\} = \{-0,02\} / \{0,1\} = -0,2$$

Agora, pode-se calcular o valor de $p(2,1)$:

$$p(2,1) = y_0L_0(2,1) + y_1L_1(2,1) + y_2L_2(2,1)$$

Substituindo os valores dados para as ordenadas (y_0 , y_1 e y_2) e os valores calculados para os Polinômios de Lagrange (L_0 , L_1 e L_2):

$$p(2,1) = (-1,00)(0,53) + (0,00)(0,7) + (-2,0)(-0,2)$$

$$p(2,1) = (-0,53) + (0,00) + (0,4)$$

$$p(2,1) = -0,13.$$

Então

$f(2,1) \approx -0,13$

Considerações. O valor -0,53, obtido para $f(2,1)$ está, aparentemente, de acordo com os dados do enunciado da questão. Pois é razoável supor que o ponto $(2,1; -0,53)$ se posiciona sobre a parábola definida pelos três pontos dados, a menos dos arredondamentos dos valores calculados.

Ele se situa no trecho situado entre os valores -1,00 de $f(2,00)$ e 0,00 de $f(2,30)$.

Resposta à Questão 3 – Nesta questão vamos calcular $f(1,35)$ com os polinômios de Lagrange a partir dos dados tabelados abaixo. Relativamente à tabela dada no enunciado da questão, já se incluiu uma linha com os valores do índice i :

i	0	1	2
x_i	1,00	1,50	2,00
y_i = f(x_i)	4,00	3,00	2,00

Vamos admitir a aproximação $f(1,35) \approx p(1,35)$, em que $p(x)$ é o polinômio interpolador:

$$p(x) = y_0L_0(x) + y_1L_1(x) + y_2L_2(x)$$

Cálculo dos Polinômios de Lagrange:

$$L_0(x) = \{(x-x_1)(x-x_2)\} / \{(x_0-x_1)(x_0-x_2)\}$$

$$L_0(1,35) = \{(1,35-1,50)(1,35-2,00)\} / \{(1,00-1,50)(1,00-2,00)\}$$

$$L_0(1,35) = \{(-0,15)(-0,65)\} / \{(-0,50)(-1,00)\} = \{0,0975\} / \{0,50\} = 0,195$$

$$L_1(x) = \{(x-x_0)(x-x_2)\} / \{(x_1-x_0)(x_1-x_2)\}$$

$$L_1(1,35) = \{(1,35-1,00)(1,35-2,00)\} / \{(1,50-1,00)(1,50-2,00)\}$$

$$L_1(1,35) = \{(0,35)(-0,65)\} / \{(0,50)(-0,50)\} = \{-0,2275\} / \{-0,25\} = 0,91$$

$$L_2(x) = \{(x-x_0)(x-x_1)\} / \{(x_2-x_0)(x_2-x_1)\}$$

$$L_2(1,35) = \{(1,35-1,00)(1,35-1,50)\} / \{(2,00-1,00)(2,00-1,50)\}$$

$$L_2(1,35) = \{(0,35)(-0,15)\} / \{(1,00)(0,50)\} = \{-0,0525\} / \{0,5\} = -0,105$$

Então,

$$p(1,35) = y_0L_0(1,35) + y_1L_1(1,35) + y_2L_2(1,35)$$

$$p(1,35) = (4,00)(0,195) + (3,00)(0,91) + (2,00)(-0,105)$$

$$p(1,35) = (0,78) + (2,73) + (-0,21) = 3,3$$

Concluindo, $f(1,35) \approx 3,3$. Esse é um resultado compatível com os dados, já que os pontos lidos no enunciado situam-se sobre uma reta $r(x) = ax + b$, tal que:

$$\begin{cases} r(1,00) = a(1,00) + b = 4,00 \\ r(1,50) = a(1,50) + b = 3,00 \end{cases}$$

Alternativamente à abordagem lagrangeana, pode-se realizar a resolução desse sistema de equações lineares que fornece $a = -2$ e $b = 6$. De modo que

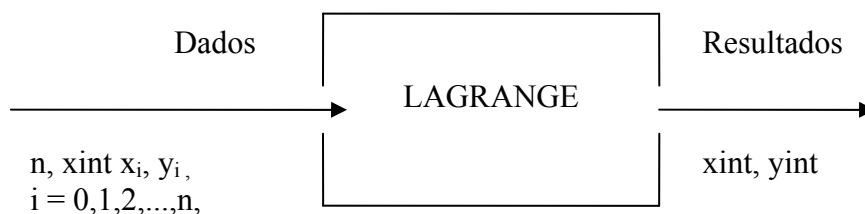
$r(1,00) = 4,00$; $r(1,35) = 3,3$; $r(1,50) = 3,00$; e $r(2,00) = 2,00$.

Pondo o resultado em destaque:

$f(1,35) \approx 3,3.$

Resposta à Questão 4 – Construção de um algoritmo que implementa computacionalmente um caso de interpolação com o emprego dos *Polinômios de Lagrange*.

Inicialmente, desenha-se a caixa preta que representa o algoritmo no mais alto nível de abstração:



Na *Entrada dos Dados* deve ser lido o valor de n e de x_{int} . E os pares de valores $[x_i; y_i = f(x_i)]$ com $i = 0, 1, 2, 3, \dots, n$, em que n representa o valor máximo que o grau do polinômio de interpolação pode assumir, e o valor de x_{int} (ou x_{barra}) é aquele para o qual se deseja obter o correspondente valor de $y_{int} = f(x_{int})$ ou $(y_{barra} = f(x_{barra}))$.

Na *Saída dos Resultados* espera-se apresentar o valor de x_{int} e de seu correspondente y_{int} .

Esboço preliminar do algoritmo, desta vez em versão literal:

ALGORITMO LAGRANGE

REAL X(5), Y(5)

REAL XINT, YINT

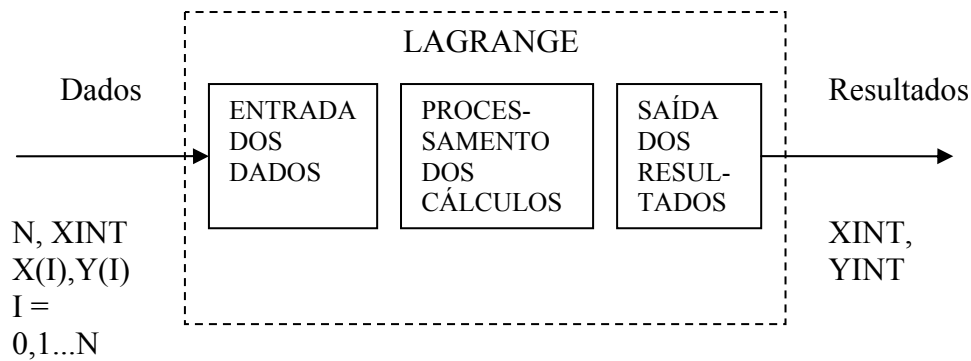
INTEIRO I, N

INÍCIO

·
·
·

FIM DO ALGORITMO LAGRANGE.

Abre-se a caixa preta para visualização de seu interior composto de três novas caixas pretas: *Entrada dos Dados*, *Processamento dos Cálculos* e *Saída dos Resultados*:



Agora, detalhando os três blocos internos e montando o algoritmo completo:

ALGORITMO LAGRANGE

(IMPLEMENTA UMA INTERPOLAÇÃO
COM POLINÔMIOS DE LAGRANGE. *)*

REAL X(5), Y(5), L(5), XINT, YINT
INTEIRO I, K, N

INÍCIO

(* ENTRADA DOS DADOS: *)

LEIA N, XINT
PARA (I DE 0 ATÉ N) FAÇA
 LEIA X(I), Y(I)
FIM PARA

(* FIM DA ENTRADA DOS DADOS. *)

(* PROCESSAMENTO DOS CÁLCULOS: *)

(CÁLCULO DOS POLINÔMIOS DE LAGRANGE: *)*

PARA (K DE 0 ATÉ N) FAÇA
 L(K) = 1
 PARA (I DE 0 ATÉ N) FAÇA
 SE (I NÃO IGUAL K) ENTÃO
 L(K) = L(K)*((XINT-X(I))/(X(K)-X(I)))
 FIM SE
FIM PARA

(FIM DO CÁLCULO DOS POLINÔMIOS DE LAGRANGE. *)*

(CÁLCULO DO VALOR DO POLINÔMIO INTERPOLANTE: *)*

YINT = 0
PARA (I DE 0 ATÉ N) FAÇA
 YINT = YINT + Y(I)*L(I)
FIM PARA

(FIM DO CÁLCULO DO VALOR DO POLINÔMIO INTERPOLANTE. *)*

(* FIM DO PROCESSAMENTO DOS CÁLCULOS. *)

(* SAÍDA DOS RESULTADOS: *)

(* SAÍDA DOS VALORES DOS POLINÔMIOS DE LAGRANGE: *)

PARA (I DE 0 ATÉ N) FAÇA

ESCREVA 'L(' , I, ') = ' , L(I)

FIM PARA

(* FIM DA SAÍDA DOS VALORES DOS POLINÔMIOS DE LAGRANGE. *)

(* SAÍDA DO VALOR DO POLINÔMIO DE INTERPOLAÇÃO: *)

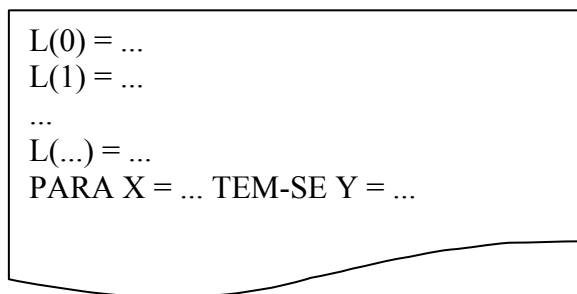
ESCREVA 'PARA X = ' , XINT, 'TEM-SE Y = ' , YINT

(* FIM DA SAÍDA DO VALOR DO POLINÔMIO DE INTERPOLAÇÃO. *)

(* FIM DA SAÍDA DOS RESULTADOS. *)

FIM DO ALGORITMO LAGRANGE.

Visualização do relatório que será produzido pela execução do algoritmo:



L(0) = ...
L(1) = ...
...
L(...) = ...
PARA X = ... TEM-SE Y = ...

PROVA 23 – Interpolação de Gregory-Newton

FOLHA DE QUESTÕES:

Questão 1 (dissertativa) – Escreva sobre a utilização dos polinômios de *Gregory-Newton* em casos de interpolação. Se achar adequado, apresente fórmulas, faça gráficos e dê exemplos que possam enriquecer a sua apresentação. Por favor, não deixe de ocupar pelo menos uma página com a sua dissertação.

Questão 2 (numérica) – Considere a tabela apresentada abaixo. Por favor, calcule as diferenças divididas correspondentes, indicando claramente todas as operações efetuadas. Organize as diferenças divididas em uma tabela.

i	0	1	2	3
x_i	3,0	3,5	4,0	4,5
$y_i = f(x_i)$	0,0	1,0	3,0	8,0

Questão 3 (numérica) – Considere a seguinte tabela de diferenças divididas. Use-a para obter $f(3)$ com a utilização do *Polinômio de Gregory-Newton com Diferenças Divididas*. Por favor, indique todas as operações efetuadas.

i	x_i	$y_i = f(x_i)$	$\Delta^1 y_i$	$\Delta^2 y_i$	$\Delta^3 y_i$
0	1	0	1	0,167	-0,114625
1	5	4	2	-0,75	-
2	7	8	-1	-	-
3	9	6	-	-	-

Questão 4 (algoritmo) – Faça um algoritmo estruturado para implementar computacionalmente um caso de *Interpolação de Gregory-Newton com Diferenças Divididas*. Desenvolva o algoritmo em sucessivas etapas de refinamento, introduzindo novos detalhes a partir de uma caixa preta inicial.

Boa Prova!

RESPOSTAS:

Resposta à Questão 1 – Os Polinômios de Gregory-Newton podem ser escritos de diversas maneiras. Podem ser escritos, por exemplo, em função de diferenças divididas, centradas, ascendentes e descendentes.

No caso de diferenças centradas, ascendentes e descendentes, exige-se que os pontos da tabela de dados estejam igualmente espaçados: $x_{i+1} - x_i = \text{constante}$, $i = 0, 1, 2, \dots, n$. Já, no caso de diferenças divididas, o espaçamento constante não é exigido, por esse motivo, ela é a técnica de emprego mais geral.

Vamos considerar a tabela de dados apresentada a seguir. Nessa tabela, os valores de x não precisam estar igualmente espaçados, mas devem estar ordenados de forma crescente: $x_{i+1} > x_i$, $i = 0, 1, 2, \dots, n$.

i	x_i	$y_i = f(x_i)$
0	x_0	y_0
1	x_1	y_1
2	x_2	y_2
.	.	.
.	.	.
.	.	.
n	x_n	y_n

Podemos então definir diferença dividida de ordem k aplicada a y_i como:

$$\Delta^k y_i = (\Delta^{k-1} y_{i+1} - \Delta^{k-1} y_i) / (x_{i+k} - x_i)$$

Em que $k = 0, 1, 2, \dots, n$ e $i = 0, 1, 2, \dots, n-k$.

Além disso, $\Delta^0 y_i = y_i$

O polinômio de Gregory-Newton definido em termos de diferenças divididas pode tomar o seguinte aspecto:

$$\begin{aligned} p(x) = & \Delta^0 y_0 \\ & + (x - x_0) \Delta^1 y_0 \\ & + (x - x_0) (x - x_1) \Delta^2 y_0 \\ & + (x - x_0) (x - x_1) (x - x_2) \Delta^3 y_0 \\ & + \dots \\ & + (x - x_0) (x - x_1) (x - x_2) \dots (x - x_{n-1}) \Delta^n y_0 \end{aligned}$$

Ou de modo mais condensado:

$$p(x) = \Delta^0 y_0 + \sum \left\{ \Delta^k y_0 \left[\prod (x - x_i) \right] \right\}$$

Em que $k = 1, 2, \dots, n$ e $i = 0, 1, 2, \dots, k-1$

Neste caso, os operadores diferenças divididas são aplicados em y_0 , mas não é preciso ser sempre assim. Em alguns casos pode-se fazer a aplicação desses operadores de modo diferente, por exemplo, em y_1 ou y_2 .

Resposta à Questão 2 – Vamos calcular as diferenças divididas. Os valores dessas diferenças serão armazenados em uma tabela.

Fórmula que será empregada para o cálculo das diferenças divididas:

$$\Delta^k y_i = (\Delta^{k-1} y_{i+1} - \Delta^{k-1} y_i) / (x_{i+k} - x_i)$$

Em que

$$k = 0, 1, 2, \dots, n$$

e

$$i = 0, 1, 2, \dots, n-k.$$

$$\text{Além disso, } \Delta^0 y_i = y_i$$

k = 0. As diferenças divididas de primeira ordem são os próprios valores das ordenadas.

k = 1. Cálculo das diferenças divididas de primeira ordem:

$$\begin{aligned} \text{Para } i = 0, \text{ tem-se } \Delta^1 y_0 &= (\Delta^0 y_1 - \Delta^0 y_0) / (x_1 - x_0) \\ \Delta^1 y_0 &= (y_1 - y_0) / (x_1 - x_0) = (1,0 - 0,0) / (3,5 - 3,0) = 2,0 \end{aligned}$$

$$\begin{aligned} \text{Para } i = 1, \text{ tem-se } \Delta^1 y_1 &= (\Delta^0 y_2 - \Delta^0 y_1) / (x_2 - x_1) \\ \Delta^1 y_1 &= (y_2 - y_1) / (x_2 - x_1) = (3,0 - 1,0) / (4,0 - 3,5) = 4,0 \end{aligned}$$

$$\begin{aligned} \text{Para } i = 2, \text{ tem-se } \Delta^1 y_2 &= (\Delta^0 y_3 - \Delta^0 y_2) / (x_3 - x_2) \\ \Delta^1 y_2 &= (y_3 - y_2) / (x_3 - x_2) = (8,0 - 3,0) / (4,5 - 4,0) = 10 \end{aligned}$$

k = 2. Cálculo das diferenças divididas de segunda ordem:

$$\begin{aligned} \text{Para } i = 0, \text{ tem-se } \Delta^2 y_0 &= (\Delta^1 y_1 - \Delta^1 y_0) / (x_2 - x_0) \\ \Delta^2 y_0 &= (4,0 - 2,0) / (4,0 - 3,0) = 2,0 \end{aligned}$$

Para $i = 1$, tem-se $\Delta^2 y_1 = (\Delta^1 y_2 - \Delta^1 y_1) / (x_3 - x_1)$
 $\Delta^2 y_1 = (10,0 - 4,0) / (4,5 - 3,5) = 6,0$

k = 3. Cálculo das diferenças divididas de terceira ordem:

Para $i = 0$, tem-se $\Delta^3 y_0 = (\Delta^2 y_1 - \Delta^2 y_0) / (x_3 - x_0)$
 $\Delta^3 y_0 = (6,0 - 2,0) / (4,5 - 3,0) = 2,67$

i	x_i	$\Delta^0 y_i = y_i = f(x_i)$	$\Delta^1 y_i$	$\Delta^2 y_i$	$\Delta^3 y_i$
0	3,0	0,0	2,0	2,0	2,67
1	3,5	1,0	4,0	6,0	-
2	4,0	3,0	10,0	-	-
3	4,5	8,0	-	-	-

Resposta à Questão 3 – Nesta questão vamos calcular $f(3)$ com o Polinômio de Gregory-Newton escrito em função de diferenças divididas. Consideramos a versão desse polinômio em que todas as diferenças são aplicadas em y_0 .

$$\begin{aligned}
 p(x) = & \Delta^0 y_0 \\
 & + (x - x_0) \Delta^1 y_0 \\
 & + (x - x_0) (x - x_1) \Delta^2 y_0 \\
 & + (x - x_0) (x - x_1) (x - x_2) \Delta^3 y_0 \\
 & + \dots \\
 & + (x - x_0) (x - x_1) (x - x_2) \dots (x - x_{n-1}) \Delta^n y_0
 \end{aligned}$$

No presente caso, $n = 3$.

Foram fornecidas as seguintes diferenças divididas:

$$\Delta^0 y_0 = 0; \quad \Delta^1 y_0 = 1; \quad \Delta^2 y_0 = 0,167; \quad \Delta^3 y_0 = -0,114625$$

Substituindo os valores dessas diferenças e os valores das abscissas na expressão do polinômio:

$$p(x) = 0 + (x - 1) (1) + (x - 1) (x - 5) (0,167) + (x - 1) (x - 5) (x - 7) (-0,114625)$$

Fazendo x tomar o valor 3,

$$p(3) = 0 + (3 - 1) (1) + (3 - 1) (3 - 5) (0,167) + (3 - 1) (3 - 5) (3 - 7) (-0,114625)$$

$$p(3) = 0 + (2) (1) + (2) (-2) (0,167) + (2) (-2) (-4) (-0,114625)$$

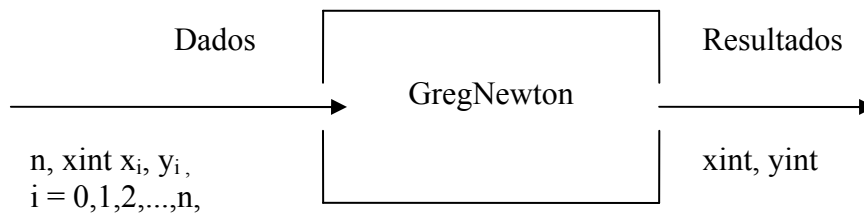
$$p(3) = 0 + 2 + (-0,668) + (-1,834) = -0,502$$

$$p(3) = -0,502$$

Conclusão: $f(3) \approx -0,502$
--

Resposta à Questão 4 – Construiremos um algoritmo que implementa computacionalmente um caso de interpolação com o emprego do *Polinômio de Gregory-Newton com Diferenças Divididas*.

Desenhamos a caixa preta que representa o algoritmo no mais alto nível de abstração:



Na *Entrada dos Dados* manda-se ler o valor de n e de x_{int} . Além disso, são lidos os pares de valores $[x_i; y_i = f(x_i)]$ com $i = 0, 1, 2, 3, \dots, n$. Aqui, n representa o valor máximo que o grau do polinômio de interpolação pode assumir, e o valor de x_{int} (ou x_{barra}) é aquele para o qual se deseja obter o correspondente valor de $y_{int} = f(x_{int})$ ou ($y_{barra} = f(x_{barra})$). Esse valor será fornecido pelo Polinômio de Gregory-Newton.

Na *Saída dos Resultados* espera-se apresentar o valor de x_{int} e de seu correspondente y_{int} .

Esboço preliminar do algoritmo, desta vez em versão literal:

ALGORITMO GREGNEWTON

REAL X(5), Y(5)

REAL XINT, YINT

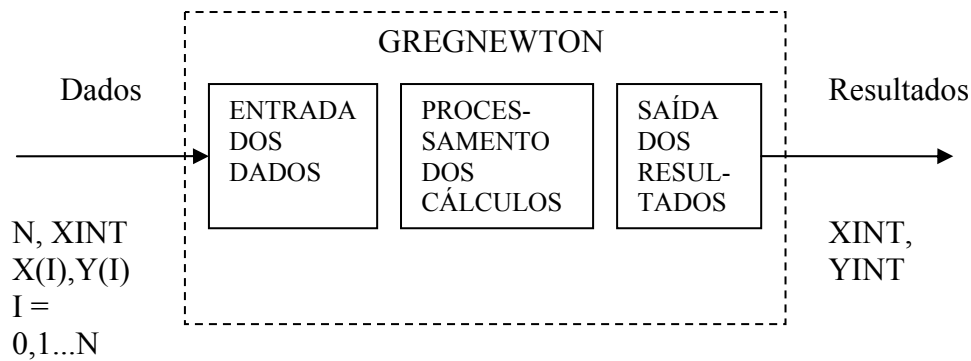
INTEIRO I, N

INÍCIO

·
·
·

FIM DO ALGORITMO GREGNEWTON.

A abertura da caixa preta permite a visualização de seu interior: três novas caixas pretas, com as funcionalidades *Entrada dos Dados*, *Processamento dos Cálculos* e *Saída dos Resultados*:



Agora, podemos detalhar os três blocos internos. Após reunir os três blocos, temos o algoritmo completo:

ALGORITMO GREGNEWTON

(REALIZA UMA INTERPOLAÇÃO
COM O POLINÔMIO DE GREGORY-NEWTON
E DIFERENÇAS DIVIDIDAS. *)*

REAL XINT, YINT
REAL X(5), Y(5), PROD(10)
REAL D(10, 10)
INTEIRO I, K, N

INÍCIO

(* ENTRADA DOS DADOS: *)

LEIA N, XINT
PARA (I DE 0 ATÉ N) FAÇA
 LEIA X(I), Y(I)
FIM PARA

(* FIM DA ENTRADA DOS DADOS. *)

(* PROCESSAMENTO DOS CÁLCULOS: *)

(* CÁLCULO DAS DIFERENÇAS DIVIDIDAS: *)

PARA (I DE 0 ATÉ N) FAÇA
 D(0, I) = Y(I)
FIM PARA

PARA (K DE 1 ATÉ N) FAÇA
 PARA (I DE 0 ATÉ N-K) FAÇA
 D(K, I) = (D(K-1, I+1) - D(K-1, I)) / (X(I+K) - X(I))
 FIM PARA
FIM PARA

(* FIM DO CÁLCULO DAS DIFERENÇAS DIVIDIDAS. *)

(* CÁLCULO DOS PRODUTÓRIOS: *)

PROD(0) = 1

PARA (K DE 1 ATÉ N) FAÇA

PROD(K) = PROD(K-1)*(XINT - X(K-1))

FIM PARA

(* FIM DO CÁLCULO DOS PRODUTÓRIOS. *)

(* CÁLCULO DO VALOR DO POLINÔMIO INTERPOLANTE: *)

YINT = 0

PARA (K DE 0 ATÉ N) FAÇA

YINT = YINT + D(K,0)*PROD(K)

FIM PARA

(* FIM DO CÁLCULO DO VALOR DO POLINÔMIO INTERPOLANTE. *)

(* FIM DO PROCESSAMENTO DOS CÁLCULOS. *)

(* SAÍDA DOS RESULTADOS: *)

(* SAÍDA DO VALOR DO POLINÔMIO DE INTERPOLAÇÃO: *)

ESCREVA 'PARA X = ', XINT

ESCREVA 'TEM-SE Y = ', YINT

(* FIM DA SAÍDA DO VALOR DO POLINÔMIO DE INTERPOLAÇÃO. *)

(* FIM DA SAÍDA DOS RESULTADOS. *)

FIM DO ALGORITMO GREGNEWTON.

Aspecto que assumirá o relatório produzido pela execução do algoritmo GregNewton:

PARA X = ...

TEM-SE Y = ...

PROVA 24 – Newton-Côtes

FOLHA DE QUESTÕES:

Questão 1 (descritiva) – Estabeleça considerações recolhidas de seu estudo sobre a *Integração Numérica*. Refira-se aos métodos abordados em aula. Se desejar, apresente figuras ilustrativas e fórmulas matemáticas referentes ao assunto. Por favor, não deixe de preencher, pelo menos, uma página.

Questão 2 (numérica) – Use a *Regra dos Trapézios*, com espaçamento constante $h = 0,25$, para calcular a integral definida apresentada a seguir. Por favor, indique todas as operações efetuadas.

$$\int_{2,0}^{3,0} (3x^2 - 5x - 16) dx$$

Questão 3 (numérica) – Obtenha o valor da integral definida apresentada abaixo. Para tal, use o *Método de Newton-Côtes* (Trapézios), adotando espaçamento constante $h = 0,15$. Por favor, indique todas as operações efetuadas ao longo dos cálculos.

$$\int_{1,45}^{1,75} (5e^x) dx$$

Questão 4 (algoritmo) – Construa um algoritmo para formalizar a aplicação do *Método de Newton-Côtes* (Regra dos Trapézios) no caso da integração da função:

$$f(x) = 5e^x$$

Por favor, observe que, na Entrada dos Dados, devem ser lidos os limites de integração (a e b) e a quantidade de passos, n ; no Processamento dos Cálculos, mande calcular o comprimento do passo, h , e faça a aplicação do método numérico mencionado; e, finalmente, na Saída dos Resultados, mande imprimir um relatório com o valor da integral.

Boa Prova!

RESPOSTAS:

Resposta à Questão 1 – No Cálculo Numérico em Computadores estuda-se a resolução de integrais definidas, isto é, integrais que possuem, como limites de integração, valores finitos. Veja o exemplo,

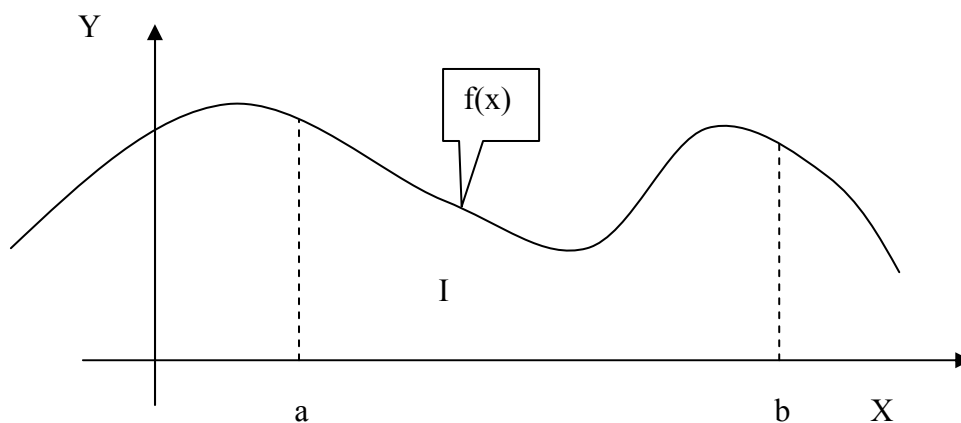
$$\int_2^3 5e^x dx$$

Nesse exemplo, o limite inferior de integração é 2 e o limite superior é 3, ambos, portanto, finitos. Somente em alguns casos especiais a integração numérica trata de problemas em que um dos limites de integração, ou ambos, são indefinidos.

Em uma integração tal como

$$I = \int_a^b f(x) dx$$

associa-se o valor da integral I à área sob a curva da função $f(x)$, considerada entre os limites a e b.



Para avaliar a área sob o gráfico de $f(x)$ estudam-se vários métodos numéricos. Assim, pode-se realizar o cálculo de I com o auxílio das Fórmulas de Newton-Côtes.

A mais simples das Fórmulas de Newton-Côtes é denominada Regra dos Trapézios. Para empregar a Regra dos Trapézios, considera-se a tabela preenchida com os valores de x e seus correspondentes $f(x)$:

i	x_i	$y_i = f(x_i)$
0	x_0	$y_0 = f(x_0)$
1	x_1	$y_1 = f(x_1)$
...
n	x_n	$y_n = f(x_n)$

Nessa tabela, o intervalo de integração (a ; b) é dividido em n subintervalos:

$$h_i = x_{i+1} - x_i, \text{ em que } i = 0, 1, 2, \dots, n-1$$

Esses subintervalos podem, ou não, ter igual comprimento. No caso de serem iguais, diz-se que os pontos $[x_i, y_i = f(x_i)]$, $i = 0, 1, 2, \dots, n$ são igualmente espaçados e, então, tem-se o passo constante

$$h = x_{i+1} - x_i, \text{ em que } i = 0, 1, 2, \dots, n-1$$

Calculam-se as áreas dos n trapézios (semi-somas dos comprimentos das bases multiplicadas pelas alturas):

$$(1/2)(y_{i+1} + y_i)h_i$$

$$\text{em que } i = 0, 1, 2, \dots, n-1$$

A soma dessas áreas fornece I_T , que é um valor próximo de I :

$$I_T \approx I = \int_a^b f(x)dx$$

$$I_T = \sum (1/2)(y_{i+1} + y_i)h_i$$

$$\text{em que } i = 0, 1, 2, \dots, n-1$$

No caso de passo constante,

$$I \approx I_T = \sum (1/2)(y_{i+1} + y_i)h, i = 0, 1, 2, \dots, n-1$$

Outras fórmulas de Newton-Côtes fazem passar não uma reta a cada dois pontos, como no caso da Fórmula dos Trapézios, mas uma parábola a cada três pontos, ou uma cúbica a cada quatro pontos, e assim por diante. O uso de fórmulas que subentendem curvas de grau mais alto permite obter, em muitos casos, resultados mais precisos.

Outra técnica é a integração gaussiana, mas esta requer a utilização da expressão algébrica da função integranda $f(x)$.

Resposta à Questão 2 – Nesta resposta, usa-se a *Regra dos Trapézios*, com espaçamento constante $h = 0,25$, para calcular o valor da integral definida:

$$I = \int_{2,0}^{3,0} (3x^2 - 5x - 16) dx$$

Inicialmente, constrói-se a tabela de dados. Neste caso, $i = 0, 1, 2, 3, 4$. Têm-se cinco pontos e espaçamento constante:

i	x_i	$y_i = f(x_i)$
0	2,00	-14
1	2,25	-12,0625
2	2,50	-9,75
3	2,75	-7,0625
4	3,00	-4

$$i = 0: y_0 = f(x_0) = 3(2,00)^2 - 5(2,00) - 16 = 12 - 10 - 16 = -14$$

$$i = 1: y_1 = f(x_1) = 3(2,25)^2 - 5(2,25) - 16 = 12 - 10 - 16 = -12,0625$$

$$i = 2: y_2 = f(x_2) = 3(2,50)^2 - 5(2,50) - 16 = 12 - 10 - 16 = -9,75$$

$$i = 3: y_3 = f(x_3) = 3(2,75)^2 - 5(2,75) - 16 = 12 - 10 - 16 = -7,0625$$

$$i = 4: y_4 = f(x_4) = 3(3,00)^2 - 5(3,00) - 16 = 12 - 10 - 16 = -4$$

Cálculo do valor aproximado da integral I:

$$I \approx \sum (1/2)(y_{i+1} + y_i)h, i = 0, 1, 2, \dots, 4$$

$$I \approx (1/2)(y_1 + y_0)(h) + (1/2)(y_2 + y_1)(h) + (1/2)(y_3 + y_2)(h) + (1/2)(y_4 + y_3)(h)$$

$$I \approx (1/2)[-12,0625 + (-14)](0,25) + (1/2)[-9,75 + (-12,0625)](0,25) + (1/2)[-7,0625 + (-9,75)](0,25) + (1/2)[-4 + (-7,0625)](0,25)$$

$$I \approx -3,2578125 - 2,7265625 - 2,1015625 - 1,3828125$$

$$I = -9,46875 \approx -9,5$$

Comparação do valor numérico obtido acima para a integral, com o valor exato, obtido analiticamente:

$$I = \int_{2,0}^{3,0} (3x^2 - 5x - 16)dx$$

$$I = [x^3 - 2,5x^2 - 16x]_{2,0}^{3,0}$$

$$I = [3,0^3 - 2,5(3,0)^2 - 16(3,0)] - [2,0^3 - 2,5(2,0)^2 - 16(2,0)]$$

$$I = [27 - 22,5 - 48] - [8 - 10 - 32]$$

$$I = -43,5 + 34 = -9,5$$

Conclusão: pode-se dizer que, nesse caso, a Regra dos Trapézios permitiu calcular com boa precisão o valor da integral proposta.

Resposta à Questão 3 – Nesta resposta obtém-se o valor da integral definida apresentada abaixo com o Método de Newton-Côtes (Fórmula dos Trapézios). Adota-se espaçamento constante $h = 0,15$.

$$I = \int_{1,45}^{1,75} (5e^x)dx$$

Desenvolve-se o processo de cálculo a partir da montagem da tabela com os valores de x e $f(x)$. Nessa tabela, i varia de zero até dois.

i	x_i	$y_i = f(x_i)$
0	1,45	21,315572
1	1,60	24,765162
2	1,75	28,773013

Agora, pode-se usar a Regra dos Trapézios:

$$I \approx (1/2)(y_1 + y_0)(h) + (1/2)(y_2 + y_1)(h)$$

$$I \approx (1/2)(24,765162 + 21,315572)(0,15) + (1/2)(28,773013 + 24,765162)(0,15)$$

$$I \approx (1/2)(46,080734)(0,15) + (1/2)(53,538175)(0,15)$$

$$I \approx 3,456055 + 4,0153631$$

$$I \approx 7,4714181$$

Comparação desse valor com o valor obtido analiticamente:

$$I = \int_{1,45}^{1,75} (5e^x) dx$$

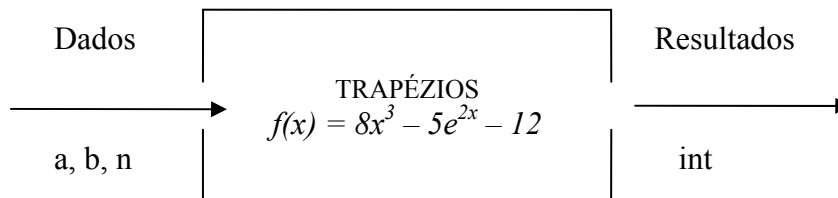
$$I = 5(e^{1,75} - e^{1,45}) = 5(5,7546 - 4,2631) = 5(1,4915) = 7,4574 \approx 7,46.$$

Observa-se que o valor obtido numericamente tem boa precisão. Uma precisão melhor pode ser alcançada com a adoção de um passo h menor.

Resposta à Questão 4 – Construção de um algoritmo para fazer a aplicação do *Método de Newton-Côtes* (Fórmula dos Trapézios) no caso da integração da função:

$$f(x) = 8x^3 - 5e^{2x} - 12.$$

Caixa preta inicial:



A tal caixa preta corresponde a um esboço do algoritmo escrito em pseudolinguagem de programação. Nesse esboço já se podem definir os tipos das variáveis: A (limite inferior de integração); B (limite superior); N (quantidade de passos); e INT (valor da integral). A função $F(X)$ também já pode ser definida:

ALGORITMO TRAPÉZIOS

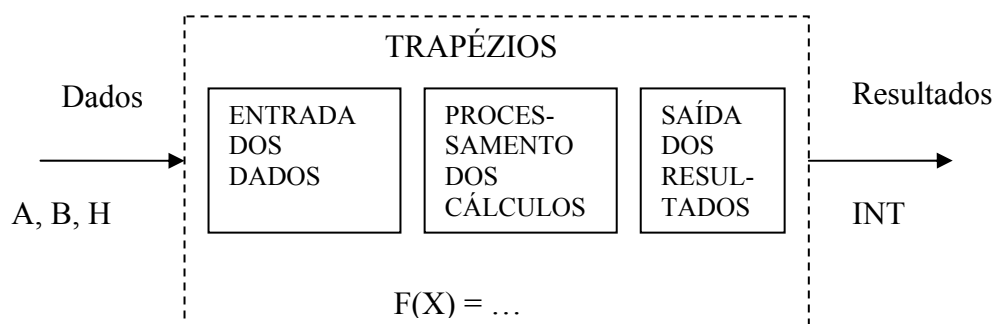
REAL A, B, H, INT
 INTEIRO N
 $F(X) = 8 * X * X * X - 5 * \text{EXP}(2 * X) - 12$

INÍCIO

.

FIM DO ALGORITMO TRAPÉZIOS.

Abre-se a caixa preta. Visualiza-se a sua composição interna. As caixas pretas *Entrada dos Dados*, *Processamento dos Cálculos* e *Saída dos Resultados* representam as principais funcionalidades do algoritmo.



A *Entrada dos Dados* pode ser detalhada como segue:

```
(* ENTRADA DOS DADOS: *)
  LEIA A, B, H
(* FIM DA ENTRADA DOS DADOS. *)
```

No detalhamento do *Processamento dos Cálculos*, calcula-se o comprimento do passo, que fica armazenado na variável H, e o valor da integral, que é guardado na variável INT.

```
(* PROCESSAMENTO DOS CÁLCULOS: *)
  H = (B - A)/N
(* CÁLCULO DAS COORDENADAS DOS PONTOS: *)
  X(0) = A; X(N) = B
  PARA (I DE 1 ATÉ N-1) FAÇA
    X(I) = X(I-1) + H
```

```

        Y(I) = F(X(I))
    FIM PARA
    (* FIM DO CÁLCULO DAS COORDENADAS DOS PONTOS. *)

    (* CÁLCULO DA INTEGRAL: *)
    INT = 0
    PARA (I DE 0 ATÉ N-1) FAÇA
        INT = INT + (1/2)*(Y(I) + Y(I+1))*H
    FIM PARA
    (* FIM DO CÁLCULO DA INTEGRAL. *)

    (* FIM DO PROCESSAMENTO DOS CÁLCULOS. *)

```

No detalhamento da *Saída dos Resultados*, manda-se escrever o valor armazenado na variável INT:

```

    (* SAÍDA DOS RESULTADOS: *)
    ESCRIVA 'INTEGRAL = ', INT
    (* FIM DA SAÍDA DOS RESULTADOS. *)

```

Agora, reunindo os detalhamentos, apresenta-se o algoritmo completo para o caso de se desejar usar a Regra dos Trapézios para tentar obter o valor de uma integral definida, considerando que são dados os valores A e B dos limites de integração, assim como a quantidade de passos N, e é conhecida a expressão algébrica da função F(X):

ALGORITMO TRAPÉZIOS

```

    (* USA A REGRA DOS TRAPÉZIOS PARA
    CALCULAR UMA INTEGRAL DEFINIDA. *)

    REAL X(20), Y(20)
    REAL A, B, H, INT
    INTEIRO I, N
    F(X) = 8*X*X*X - 5*EXP(2*X) - 12

INÍCIO

    (* ENTRADA DOS DADOS: *)
    LEIA A, B, N
    (* FIM DA ENTRADA DOS DADOS. *)

    (* PROCESSAMENTO DOS CÁLCULOS: *)
    H = (B - A)/N

    (* CÁLCULO DAS COORDENADAS DOS PONTOS: *)
    X(0) = A; X(N) = B
    PARA (I DE 1 ATÉ N-1) FAÇA
        X(I) = X(I-1) + H
        Y(I) = F(X(I))
    FIM PARA

```

(* FIM DO CÁLCULO DAS COORDENADAS DOS PONTOS. *)

(* CÁLCULO DA INTEGRAL: *)

INT = 0

PARA (I DE 0 ATÉ N-1) FAÇA

INT = INT + (1/2)*(Y(I) + Y(I+1))*H

FIM PARA

(* FIM DO CÁLCULO DA INTEGRAL. *)

(* FIM DO PROCESSAMENTO DOS CÁLCULOS. *)

(* SAÍDA DOS RESULTADOS: *)

ESCREVA 'INTEGRAL = ', INT

(* FIM DA SAÍDA DOS RESULTADOS. *)

FIM DO ALGORITMO TRAPÉZIOS.

PROVA 25 – Integração de Simpson

FOLHA DE QUESTÕES:

Questão 1 (dissertativa) – Por favor, escreva uma pequena dissertação de, pelo menos, uma página, sobre o modo de integrar funções com a *Fórmula de Simpson 1/3*. Apresente fórmulas, figuras e exemplos que possam enriquecer sua apresentação.

Questão 2 (numérica) – Use a *Fórmula de Simpson 1/3* (ou das Parábolas) para obter o valor da integral definida:

$$I = \int_{2,00}^{3,00} f(x)dx$$

em que $f(x) = x^3 + e^{-2x} - 5$, considerando a tabela apresentada a seguir:

i	0	1	2	3	4
x_i	2,00	2,25	2,50	2,75	3,00
$y_i = f(x_i)$	3,0183156	6,4017339	10,631737	15,800961	22,002478

Por favor, não deixe de indicar claramente todas as operações efetuadas. Além disso, compare o resultado calculado numericamente com o resultado obtido analiticamente.

Questão 3 (numérica) – Calcule o valor da integral definida

$$I = \int_{1,0}^{1,4} (3x^2 + 2)dx$$

com o emprego da *Fórmula de Simpson 1/3*. Por favor, use espaçamento constante $h = 0,1$. Indique todas as operações efetuadas durante o processo de cálculo.

Questão 4 (algoritmo) – Por favor, construa um algoritmo estruturado em três blocos de instruções (Entrada dos Dados, Processamento dos Cálculos e Saída dos Resultados) para formalizar a aplicação da integração numérica de funções com a *Fórmula de Simpson 1/3*.

Boa Prova!

RESPOSTAS:

Resposta à Questão 1 – Na integração de funções em que os limites de integração, inferior e superior, são finitos, pode-se empregar a Fórmula de Simpson 1/3, também denominada de Fórmula Parabólica (ou das Parábolas). Seu emprego precisa de uma tabela em que se tenham pontos definidos pelos valores de suas abscissas e ordenadas: $[x, f(x)]$, tendo em mente que $f(x)$ é a função que se deseja integrar.

A quantidade de pontos assim definidos deve ser ímpar, igual a três ou maior do que três. Essa exigência se compreende uma vez que, no uso da Fórmula de Simpson 1/3 passa-se uma parábola a cada três pontos e calcula-se a área sob cada uma dessas parábolas. No caso de haver uma quantidade par de pontos - por exemplo, quatro pontos - a integração pode ser realizada, mas não com uso exclusivo da Fórmula de Simpson 1/3: pode-se empregar esta fórmula enquanto for possível e, depois, uma vez a Fórmula dos Trapézios. A área sob as parábolas é somada àquela sob a reta para se ter o valor da integral.

A dedução da Fórmula das Parábolas pode tomar o seguinte aspecto: deseja-se calcular

$$I = \int_{x_0}^{x_{2n}} f(x) dx$$

A tabela que representa a função $f(x)$ possui $2n+1$ pontos: $[x_i, y = f(x_i)]$, $i = 0, 1, 2, 3, \dots, 2n+1$.

A quantidade de parábolas é n ; e $2n$ é a quantidade de subintervalos.

Pelos três primeiros pontos, (x_0, y_0) , (x_1, y_1) e (x_2, y_2) passa a parábola p_1 ;

A parábola p_2 passa pelos pontos (x_2, y_2) , (x_3, y_3) e (x_4, y_4) ;

A parábola p_3 passa por (x_4, y_4) , (x_5, y_5) e (x_6, y_6) ;

E assim por diante.

Ora, a parábola qualquer, p_i , pode ser definida por $p_i(x) = ax^2 + bx + c$.

Para $x = x_{i-1}$:

$$p_i(x_{i-1}) = a(x_{i-1})^2 + b(x_{i-1}) + c = y_{i-1}$$

Para $x = x_i$:

$$p_i(x_i) = a(x_i)^2 + b(x_i) + c = y_i$$

Para $x = x_{i+1}$:

$$p_i(x_{i+1}) = a(x_{i+1})^2 + b(x_{i+1}) + c = y_{i+1}$$

O sistema de três equações com três incógnitas pode ser resolvido, obtendo-se, a, b e c). Com esses valores calcula-se a integral I_i :

$$I_i = \int_{x_{i-1}}^{x_{i+1}} p_i(x) dx = \int_{x_{i-1}}^{x_{i+1}} (ax^2 + bx + c) dx = \left[(a/3)x^3 + (b/2)x^2 + cx \right]_{x_{i-1}}^{x_{i+1}}$$

Se o espaçamento é constante, então, $x_{i+1} - x_i = h$, $i = 0, 1, 2, \dots, 2n$. Nesse caso,

$$I_i = (h/3)[y_{i-1} + 4y_i + y_{i+1}] \text{ para uma parábola.}$$

Considerando as n parábolas,

$$I \approx I_1 + I_2 + \dots + I_n$$

E então,

$$I \approx (h/3)[y_0 + 4S_1 + 2S_2 + y_{2n}]$$

Em que:

$S_1 = y_1 + y_3 + \dots + y_{2n-1}$ (é a soma das ordenadas de índice ímpar); e

$S_2 = y_2 + y_4 + \dots + y_{2n-2}$ (é a soma das ordenadas internas de índice par).

Resposta à Questão 2 – Neste caso, deve-se realizar uma integração com a *Fórmula de Simpson 1/3* (ou das Parábolas) para obter o valor de:

$$I = \int_{2,00}^{3,00} f(x) dx$$

em que $f(x) = x^3 + e^{-2x} - 5$.

Considera-se a tabela:

i	0	1	2	3	4
x_i	2,00	2,25	2,50	2,75	3,00
$y_i = f(x_i)$	3,0183156	6,4017339	10,631737	15,800961	22,002478

Tem-se $2n+1 = 5$ (quantidade de pontos); $n = 2$ (quantidade de parábolas); $2n = 4$ (quantidade de subintervalos); e $h = 0,25$ (espaçamento constante). A fórmula a ser usada é:

$$I \approx (h/3)[y_0 + 4S_1 + 2S_2 + y_{2n}]$$

Em que:

$S_1 = y_1 + y_3 + \dots + y_{2n-1}$ (é a soma das ordenadas de índice ímpar); e

$S_2 = y_2 + y_4 + \dots + y_{2n-2}$ (é a soma das ordenadas internas de índice par).

Neste caso,

$$S_1 = y_1 + y_3 = 6,4017339 + 15,800961 = 22,202694$$

$$S_2 = y_2 = 10,631737$$

Então,

$$I \approx (0,25/3)[3,0183156 + 4(22,202694) + 2(10,631737) + 22,002478]$$

$$I \approx (0,083333333)[3,0183156 + 88,810776 + 21,263474 + 22,002478]$$

$$I \approx (0,083333333)[135,09504]$$

$$I \approx 11,257915$$

A seguir, desenvolve-se o procedimento analítico para o cálculo do valor da integral. O objetivo é compará-lo com o valor obtido numericamente.

Integrando por partes:

$$\int (x^3 + e^{-2x} - 5)dx = \int (x^3)dx + \int (e^{-2x})dx + \int (-5)dx$$

$$\int (x^3 + e^{-2x} - 5)dx = (1/4)x^4 + (-1/2)(e^{-2x}) + (-5x) + C$$

Em que C é uma constante de integração.

Considerando os limites de integração:

$$I = \int_{2,00}^{3,00} (x^3 + e^{-2x} - 5)dx = \left[(1/4)x^4 + (-1/2)(e^{-2x}) + (-5x) \right]_{2,00}^{3,00}$$

$$I = [(1/4)(3^4) - (1/2)(e^{-2(3)}) - 5(3)] - [(1/4)(2^4) - (1/2)(e^{-2(2)}) - 5(2)]$$

$$I = [20,25 - 0,0012393 - 15] - [4 - 0,00911578 - 10]$$

$$I = [5,2487606] - [-6,0091578]$$

$$I = 11,257918.$$

Conclusão: Observa-se que o procedimento analítico e o método numérico conduziram a resultados muito próximos.

Resposta à Questão 3 – Para responder a esta questão, calcula-se o valor da integral definida:

$$I = \int_{1,0}^{1,4} (3x^2 + 2)dx$$

com o emprego da *Fórmula de Simpson 1/3*. Neste problema, usa-se espaçamento constante $h = 0,1$. Faz-se indicação de todas as operações efetuadas.

Inicialmente, constrói-se a tabela com as coordenadas dos pontos que representa a função $f(x) = 3x^2 + 2$ a ser integrada:

i	0	1	2	3	4
x_i	1,0	1,1	1,2	1,3	1,4
$y_i = f(x_i)$	5,00	5,63	6,32	7,07	7,88

Agora, tem-se:

$2n+1 = 5$ (quantidade de pontos);
 $n = 2$ (quantidade de parábolas);
 $2n = 4$ (quantidade de subintervalos); e
 $h = 0,1$ (espaçamento constante).

A fórmula a ser usada é:

$$I \approx (h/3)[y_0 + 4S_1 + 2S_2 + y_{2n}]$$

Em que:

$S_1 = y_1 + y_3 + \dots + y_{2n-1}$ é a soma das ordenadas de índice ímpar; e

$S_2 = y_2 + y_4 + \dots + y_{2n-2}$ é a soma das ordenadas internas de índice par.

Neste caso,

$$S_1 = y_1 + y_3 = 5,63 + 7,07 = 12,7$$

$$S_2 = y_2 = 6,32$$

Então,

$$I \approx (0,1/3)[5,00 + 4(12,7) + 2(6,32) + 7,88]$$

$$I \approx (0,03333333)[5,00 + 50,8 + 12,64 + 7,88]$$

$$I \approx (0,03333333)[76,32]$$

$$I \approx 2,5439974$$

Cálculo analítico da integral:

$$I = \int_{1,0}^{1,4} (3x^2 + 2)dx = \left[x^3 + 2x \right]_{1,0}^{1,4}$$

$$I \approx [(1,4)^3 + 2(1,4)] - [(1,0)^3 + 2(1,0)]$$

$$I \approx [2,744 + 2,8] - [1,0 + 2,0]$$

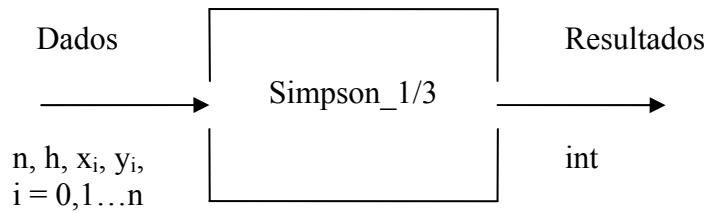
$$I \approx [5,544] - [3,0]$$

$$I \approx 2,544$$

Conclusão: o valor obtido de modo analítico e o valor obtido numericamente estão muito próximos.

Resposta à Questão 4 – No algoritmo que se pretende construir a função a integrar é representada por $n+1$ pontos dados por suas coordenadas: $x_i, y_i, i = 0, 1, \dots, n$. O espaçamento dos pontos é constante: h (que, por simplicidade, é um valor lido, embora, a rigor, pudesse ser calculado sem aumentar demais a complexidade do algoritmo).

Inicia-se o processo de elaboração do algoritmo com o desenho de uma caixa preta que representa, graficamente, o algoritmo no nível mais alto de abstração:



Agora, faz-se um esboço do algoritmo (este, escrito em pseudolinguagem de programação). Nessa fase de desenvolvimento do algoritmo já se podem definir os tipos de algumas variáveis:

N (quantidade de subintervalos);
 H (passo);
 X(I) e Y(I) (coordenadas dos pontos que representam a função a integrar); e
 INT (valor da integral):

ALGORITMO SIMPSON_1/3

REAL X(49), Y(49), H, INT
 INTEIRO N

INÍCIO

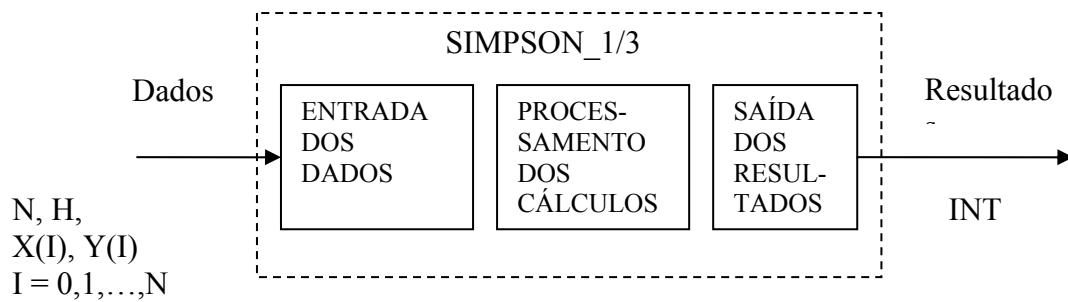
.

.

.

FIM DO ALGORITMO SIMPSON_1/3.

No detalhamento, visualiza-se a estrutura interna do algoritmo. As caixas pretas *Entrada dos Dados*, *Processamento dos Cálculos* e *Saída dos Resultados* correspondem às funcionalidades que os nomes já sugerem.



A seguir, detalha-se cada uma das funcionalidades. Reúnem-se os detalhes realizados:

ALGORITMO SIMPSON_1/3

(INTEGRA UMA FUNÇÃO DADA POR UM CONJUNTO DE PONTOS. USA A FÓRMULA DAS PARÁBOLAS (SIMPSON 1/3). *)*

REAL X(49), Y(49), S1, S2, H, INT
INTEIRO I, N

INÍCIO

(* ENTRADA DOS DADOS: *)

LEIA N, H
PARA (I DE 0 ATÉ N) FAÇA
 LEIA X(I), Y(I)
FIM PARA

(* FIM DA ENTRADA DOS DADOS. *)

(* PROCESSAMENTO DOS CÁLCULOS: *)

(* CÁLCULO DE S1: *)

S1 = 0
PARA (I DE 1 ATÉ 2*N-1 DE 2 EM 2) FAÇA
 S1 = S1 + Y(I)
FIM PARA

(* FIM DO CÁLCULO DE S1. *)

(* CÁLCULO DE S2: *)

S2 = 0
PARA (I DE 2 ATÉ 2*N-2 DE 2 EM 2) FAÇA
 S2 = S2 + Y(I)
FIM PARA

(* FIM DO CÁLCULO DE S2. *)

(* CÁLCULO DA INTEGRAL: *)
$$\text{INT} = (H/3) * (Y(0) + 4 * S1 + 2 * S2 + Y(2 * N))$$
(* FIM DO CÁLCULO DA INTEGRAL. *)

(* FIM DO PROCESSAMENTO DOS CÁLCULOS. *)

(* SAÍDA DOS RESULTADOS: *)
ESCREVA ' INTEGRAL = ', INT
(* FIM DA SAÍDA DOS RESULTADOS. *)

FIM DO ALGORITMO SIMPSON_1/3.

Parte II - PROJETOS

PROJETO 11 (Programa Sistema_Linear) – Construa um programa estruturado em linguagem Pascal para resolver um sistema de três equações lineares pela Regra de Cramer.

Solução

Inicialmente, consideremos um algoritmo para resolver o sistema $AX = B$:

ALGORITMO CRAMER

```
REAL A(3,3), B(3)
REAL DET, DET1, DET2, DET3
REAL X1, X2, X3
INTEIRO I, J
```

INÍCIO

(* ENTRADA DOS DADOS: *)

```
PARA (I DE 1 ATÉ 3) FAÇA
    LEIA A(I, J) PARA J DE 1 ATÉ 3, B(I)
FIM PARA
```

(* FIM DA ENTRADA DOS DADOS. *)

(* PROCESSAMENTO DOS CÁLCULOS: *)

(* UTILIZAÇÃO DA REGRA DE SARRUS: *)

(* DETERMINANTE PRINCIPAL: *)

```
DET = A(1,1)*A(2,2)*A(3,3)+A(1,2)*A(2,3)*A(3,1)+A(1,3)*A(3,2)*A(2,1)
      -A(1,3)*A(2,2)*A(3,1)-A(2,3)*A(3,2)*A(1,1)-A(3,3)*A(2,1)*A(1,2)
```

(* PRIMEIRO DETERMINANTE CARACTERÍSTICO: *)

```
DET1 = B(1)*A(2,2)*A(3,3)+A(1,2)*A(2,3)*B(3)+A(1,3)*A(3,2)*B(2)
      -A(1,3)*A(2,2)*B(3)-A(2,3)*A(3,2)*B(1)-A(3,3)*B(2)*A(1,2)
```

(* SEGUNDO DETERMINANTE CARACTERÍSTICO: *)

```
DET2 = A(1,1)*B(2)*A(3,3)+B(1)*A(2,3)*A(3,1)+A(1,3)*B(3)*A(2,1)
      -A(1,3)*B(2)*A(3,1)-A(2,3)*B(3)*A(1,1)-A(3,3)*A(2,1)*B(1)
```

(* TERCEIRO DETERMINANTE CARACTERÍSTICO: *)

```
DET3 = A(1,1)*A(2,2)*B(3)+A(1,2)*B(2)*A(3,1)+B(1)*A(3,2)*A(2,1)
      -B(1)*A(2,2)*A(3,1)-B(2)*A(3,2)*A(1,1)-B(3)*A(2,1)*A(1,2)
```

(* SOLUÇÃO DO SISTEMA LINEAR: *)

```
X1 = DET1/DET; X2 = DET2/DET; X3 = DET3/DET
```

(* FIM DO PROCESSAMENTO DOS CÁLCULOS. *)

(* SAÍDA DOS RESULTADOS: *)

```
ESCREVA 'X1 = ', X1, 'X2 = ', X2, 'X3 = ', X3
```

(* FIM DA SAÍDA DOS RESULTADOS. *)

FIM DO ALGORITMO CRAMER.

Agora, temos uma implementação Pascal desse algoritmo:

```
Program Cramer;
uses crt; var
  a:array[1..3,1..3] of real;
  b:array[1..3] of real;
  det, det1, det2, det3: real;
  x1, x2, x3: real;
  i,j: integer;
begin

  { * ENTRADA DOS DADOS: *}
  clrscr;
  writeln('Este programa resolve um sistema ax=b de');
  writeln('três equações lineares pela Regra de Cramer. ');
  writeln('Por favor, digite os elementos da matriz a');
  writeln('e do vetor b por linhas. ');
  for i:=1 to 3 do begin
    for j:=1 to 3 do begin
      writeln('Digite a['i','j,','j,']:'); readln(a[i,j]);
    end; {for}
    writeln('Digite b['i,']:'); readln(b[i]);
  end; {for}
  writeln('*****');
  { * FIM DA ENTRADA DOS DADOS. *}

  { * PROCESSAMENTO DOS CÁLCULOS: *}
  det:=
    a[1,1]*a[2,2]*a[3,3]+a[1,2]*a[2,3]*a[3,1]+a[1,3]*a[3,2]*a[2,1]
    -a[1,3]*a[2,2]*a[3,1]-a[2,3]*a[3,2]*a[1,1]-a[3,3]*a[2,1]*a[1,2];

  det1:=
    b[1]*a[2,2]*a[3,3]+a[1,2]*a[2,3]*b[3]+a[1,3]*a[3,2]*b[2]
    -a[1,3]*a[2,2]*b[3]-a[2,3]*a[3,2]*b[1]-a[3,3]*b[2]*a[1,2];

  det2:=
    a[1,1]*b[2]*a[3,3]+b[1]*a[2,3]*a[3,1]+a[1,3]*b[3]*a[2,1]
    -a[1,3]*b[2]*a[3,1]-a[2,3]*b[3]*a[1,1]-b[3]*a[2,1]*a[1,2];

  det3:=
    a[1,1]*a[2,2]*b[3]+a[1,2]*b[2]*a[3,1]+b[1]*a[3,2]*a[2,1]
    -b[1]*a[2,2]*a[3,1]-b[2]*a[3,2]*a[1,1]-b[3]*a[2,1]*a[1,2];

  x1:= det1/det;
  x2:= det2/det;
  x3:= det3/det;

  { * FIM DO PROCESSAMENTO DOS CÁLCULOS. *}
end;
```



```

{* SAÍDA DOS RESULTADOS: *}
writeln('*****');
readkey;
writeln('Determinante Principal:',det);
readkey;
writeln('Primeiro Determinante Característico:',det1);
readkey;
writeln('Segundo Determinante Característico:',det2);
readkey;
writeln('Terceiro Determinante Característico:',det3);
writeln('
');
readkey;
writeln('  SOLUÇÃO DO SISTEMA:  ');
writeln('
');
writeln('  x1 = ',x1);
writeln('  x2 = ',x2);
writeln('  x3 = ',x3);
readkey;
{* FIM DA SAÍDA DOS RESULTADOS. *}
end. {* FIM DO PROGRAMA CRAMER. *}

```

E uma execução do programa Cramer:

Este programa resolve um sistema $ax=b$ de três equações lineares pela Regra de Cramer.

Por favor, digite os elementos da matriz a e do vetor b por linhas.

Digite a[1,1]:

4

Digite a[1,2]:

-3

Digite a[1,3]:

2

Digite b[1]:

-1

Digite a[2,1]:

1

Digite a[2,2]:

4

Digite a[2,3]:

-4

Digite b[2]:

-1

Digite a[3,1]:

-2

Digite a[3,2]:

3

Digite a[3,3]:

1

Digite b[3]:

-4

Determinante Principal: 6.5000000000E+01

Primeiro Determinante Característico:-4.1000000000E+01

Segundo Determinante Característico:-1.0000000000E+02

Terceiro Determinante Característico:-8.1000000000E+01

SOLUÇÃO DO SISTEMA:

x1 = -6.3076923077E-01

x2 = -1.5384615385E+00

x3 = -1.2461538462E+00

PROJETO 12 (Programa Gauss) – Construa um programa computacional estruturado em linguagem Pascal para resolver um sistema de n equações lineares e n incógnitas pelo Método da Eliminação Gaussiana.

<pre> program EliminacaoGaussiana; uses crt; var i,j,k,kk,n: integer; a:array[1..10,1..10] of real; c:array[1..10] of real; x:array[1..10] of real; m,soma: real; begin {ENTRADA DOS DADOS;} clrscr; writeln('Este programa resolve um sistema de equacoes lineares'); writeln('pelo metodo de Eliminacao Gaussiana.');</pre> <pre> writeln('Digite a quantidade de equacoes:'); readln(n); writeln('Digite os elementos da matriz aumentada A C por linhas:'); for i:=1 to n do begin for j:=1 to n do begin writeln('Digite a['i','j']:'); readln(a[i,j]); end; {for} writeln('Digite c['i']:'); readln(c[i]); end; {for} {TESTE RELATIVO AOS DADOS;} for i:=1 to n do begin if(abs(a[i,i])<0.1) then begin writeln('Elemento da diagonal principal da matriz eh nulo'); writeln('ou muito proximo de zero'); writeln('A EXECUCAO DEVE SER INTERROMPIDA'); readkey; end; {if} end; {for} writeln('***** *****'); {FIM DO TESTE RELATIVO AOS DADOS.} {PROCESSAMENTO DOS CALCULOS;} {Primeira Etapa - Triangularizacao de A;} for k:=1 to n-1 do begin</pre>	<pre> Este programa resolve um sistema de equacoes lineares pelo metodo de Eliminacao Gaussiana. Digite a quantidade de equacoes: 3 Digite os elementos da matriz aumentada A C por linhas: Digite a[1,1]: 3 Digite a[1,2]: 2 Digite a[1,3]: 4 Digite c[1]: 1 Digite a[2,1]: 1 Digite a[2,2]: 1 Digite a[2,3]: 2 Digite c[2]: 2 Digite a[3,1]: 4 Digite a[3,2]: 3 Digite a[3,3]: -2 Digite c[3]: 3 ***** ***** Resultados obtidos: x[1] = -2.3333333333E+00 x[2] = 3.7500000000E+00 x[3] = 1.2500000000E-01</pre>
---	---

<pre> for i:=k+1 to n do begin m:=a[i,k]/a[k,k]; {pivo} a[i,k]:=0; for j:=k+1 to n do begin a[i,j]:=a[i,j]-m*a[k,j]; c[i]:=c[i]-m*c[k]; end; {for} end; {for} end; {for} readkey; {Segunda Etapa - Retrossubstituicao:} x[n]:=c[n]/a[n,n]; for kk:=1 to n-1 do begin k:=n-kk; soma:=0; for j:=(k+1) to n do begin soma:=soma+a[k,j]*x[j]; x[k]:=(c[k]-soma)/a[k,k]; end; {for} end; {for} readkey; {FIM DO PROCESSAMENTO DOS CALCULOS.} {SAIDA DOS RESULTADOS:} writeln('Resultados obtidos:'); for i:=1 to n do begin writeln('x['i,'] = ',x[i]); end; {for} readkey; end. </pre>	
--	--

PROJETO 13 (Programa LU) – Construa um programa estruturado em Pascal para realizar a resolução de um sistema linear com o Método da Fatoração LU.

SOLUÇÃO: Implementação do Algoritmo Fatoração LU em Pascal.

Segue abaixo o programa implantado em Pascal com o uso do compilador Borland Turbo Pascal. O programa é executado em um caso simples de um sistema de duas equações lineares com duas incógnitas.

```
PROGRAM FATORACAO_LU;
USES CRT;
VAR
  A,L,U,M: ARRAY[1..50,1..50] OF REAL;
  C,Y,X: ARRAY[1..50] OF REAL;
  SOMA: REAL;
  I,J,K,N: INTEGER;
BEGIN
  CLRSCR;
  { * ENTRADA DE DADOS: * }
  WRITELN('Este programa resolve sistemas');
  WRITELN('de equações lineares pelo método da');
  WRITELN('Fatoração LU. O sistema a ser resolvido');
  WRITELN('deve estar na forma Ax=C, em que A é a');
  WRITELN('matriz dos coeficientes, x é o vetor');
  WRITELN('das incógnitas e C é a matriz dos termos');
  WRITELN('independentes. ');
  WRITELN('Escreva a ordem do sistema linear (limite=50) : ');
  WRITE('n='); READ(n);
  WRITELN('Entre agora com o valor dos elementos da matriz A:');
  FOR I:=1 TO N DO BEGIN
    FOR J:=1 TO N DO BEGIN
      WRITE('A[' ,I ,',' ,J ,']=');
      READ(A[I,J]); END;
    WRITELN('Entre com o valor do elemento C[' ,I ,'] da matriz C:');
    WRITE('C[' ,I ,']=');
    READ(C[I]);
    WRITELN(' '); END;
  { * FIM DA ENTRADA DE DADOS. * }
  { * PROCESSAMENTO DOS CÁLCULOS: * }
  { * CONSTRUÇÃO DA MATRIZ U: * }
  FOR I:=1 TO N DO BEGIN
    FOR J:=1 TO N DO
      U[I,J]:=A[I,J] END;
    FOR J:=1 TO N-1 DO BEGIN
      FOR I:=J+1 TO N DO BEGIN
        M[I,J]:=U[I,J]/U[J,J];
        FOR K:=J TO N DO U[I,K]:=U[I,K] - M[I,J]*U[J,K];
      END; END;
  { * FIM DA CONSTRUÇÃO DA MATRIZ U. * }
  { * CONSTRUÇÃO DA MATRIZ L: * }
```

```

FOR I:=1 TO N DO BEGIN
FOR J:=1 TO N DO L[I,I]:=0; END;
FOR I:=1 TO N DO BEGIN
FOR J:=1 TO N DO BEGIN
L[I,I]:=1;
IF I>J THEN
L[I,J]:=M[I,J]; END; END;
{* FIM DA CONSTRUÇÃO DA MATRIZ L. *}
{* RESOLUÇÃO DO SISTEMA LY=C: *}
Y[1]:=C[1]/L[1,1];
FOR I:=2 TO N DO BEGIN
SOMA:=C[I];
FOR J:=1 TO I-1 DO SOMA:= SOMA-L[I,J]*Y[J];
Y[I]:=SOMA/L[I,I]; END;
{* FIM DA RESOLUÇÃO DO SISTEMA LY=C. *}
{* RESOLUÇÃO DO SISTEMA UX=Y: *}
X[N]:=Y[N]/U[N,N];

{* FIM DA RESOLUÇÃO DO SISTEMA LY=C. *}
{* RESOLUÇÃO DO SISTEMA UX=Y: *}
X[N]:=Y[N]/U[N,N];
FOR K:=1 TO N-1 DO BEGIN
I:=N-K;
SOMA:=Y[I];
FOR J:=(I+1) TO N DO SOMA:=SOMA-U[I,J]*X[J];
X[I]:=SOMA/U[I,I]; END;
{* FIM DA RESOLUÇÃO DO SISTEMA UX=Y. *}
{* FIM DO PROCESSAMENTO DOS CÁLCULOS . *}
READKEY;
WRITELN(' ');
{* SAÍDA DOS RESULTADOS: *}
WRITELN('Matriz dos valores de X:');
FOR I:=1 TO N DO WRITELN('X['I,']=',X[I]:5:4);
WRITELN('FIM. ');
{* FIM DA SAÍDA DOS RESULTADOS. *}
READKEY;
END.

```

RESULTADO DE UMA EXECUÇÃO:

Este programa resolve sistemas de equações lineares pelo método da Fatoração LU. O sistema a ser resolvido deve estar na forma $Ax=C$, em que A é a matriz dos coeficientes, x é o vetor das incógnitas e C é a matriz dos termos independentes.

Escreva a ordem do sistema linear (limite=50) :

n=2

Entre agora com o valor dos elementos da matriz A:

A[1,1]=3

A[1,2]=-2

Entre com o valor do elemento C[1] da matriz C:

C[1]=5

A[2,1]=-1

A[2,2]=5

Entre com o valor do elemento C[2] da matriz C:

C[2]=6

Matriz dos valores de X:

X[1]=2.8462

X[2]=1.7692

FIM.

PROJETO 14 (Programa Gauss_Seidel) – Construa um programa Pascal de modo estruturado para resolver um sistema de n equações lineares simultâneas com o uso do Método Iterativo de Gauss-Seidel.

<pre> program GaussSeidel; {INE 5202 Calculo Numerico em Computadores} uses crt; var i,j,n,it,itmax,liminf,limsup:integer; prec,precmin,somapre,somapos:real; a:array[1..10,1..10] of real; c:array[1..10] of real; x:array[1..10,1..10] of real; begin {DADOS;} clrscr; writeln('Este programa resolve um sistema de equacoes lineares'); writeln('pelo metodo iterativo de Gauss- Seidel'); readkey; writeln('Digite a quantidade de equacoes'); readln(n); writeln('Digite os elementos da matriz aumentada A/C por linhas'); for i:=1 to n do begin for j:=1 to n do begin writeln('Digite a[' ,i ,',' ,j ,']'); readln(a[i,j]); end; {for} writeln('Digite c[' ,i ,']'); readln(c[i]); end; {for} writeln('Digite a quantidade maxima de iteracoes:'); readln(itmax); writeln('Digite a precisao minima aceitavel:'); readln(precmin); {TESTE RELATIVO AOS DADOS;} for i:=1 to n do begin if(abs(a[i,i])<0.1) then begin writeln('Elemento da diagonal principal da matriz eh nulo'); writeln('ou muito proximo de zero'); writeln('A EXECUCAO DEVE SER INTERROMPIDA'); readkey; </pre>	<pre> {ITERACOES} while((it<=itmax)and(prec>=precmin))do begin writeln('Iteracao Numero ',it); prec:=0.0; for i:=1 to n do begin somapre:=0.0; if(i>1)then begin for j:=1 to i-1 do begin somapre:=somapre+a[i,j]*x[j,it+1]; end; {for} end; {if} somapos:=0.0; if(i<n)then begin for j:=i+1 to n do begin somapos:=somapos+a[i,j]*x[j,it]; end; {for} end; {if} {CALCULO DE NOVA ESTIMATIVA DA SOLUCAO DO SISTEMA:} x[i,it+1]:=(c[i]-somapre-somapos)/a[i,i]; prec:=prec+abs(x[i,it+1]-x[i,it]); writeln('x[' ,i ,'] = ',x[i,it+1]); end; {for} writeln('prec := ',prec); it:=it+1; readkey; end; {while} {SAIDA DOS RESULTADOS} if(prec<precmin)then begin writeln('Houve convergencia do processo iterativo'); writeln('para a solucao do sistema'); writeln('em ',it,' iteracoes, com prec = ',prec); end; {if} if(prec>precmin)then begin writeln('A precisao desejada nao foi obtida'); writeln('Foram realizadas ',it,' iteracoes'); writeln('com prec = ',prec); end; {if} writeln('ultimos valores obtidos:'); for i:=1 to n do begin writeln('x[' ,i ,'] = ',x[i,it]); </pre>
---	---

<pre> end; {if} end; {for} writeln('*****'); {PROCESSAMENTO} {INICIALIZACOES;} it:=1;prec:=1000.0; for i:=1 to n do begin x[i,it]:=c[i]/a[i,i]; end; {for} </pre>	<pre> end; {for} writeln('prec = ',prec); readkey; end. </pre>
---	--

<p>Este programa resolve um sistema de equacoes lineares pelo metodo iterativo de Gauss-Seidel. Digite a quantidade de equacoes: 3 Digite os elementos da matriz aumentada A C por linhas: Digite a[1,1]: 5 Digite a[1,2]: 1 Digite a[1,3]: 1 Digite c[1]: 5 Digite a[2,1]: 3 Digite a[2,2]: 4 Digite a[2,3]: 1 Digite c[2]: 6 Digite a[3,1]: 3 Digite a[3,2]: 3 Digite a[3,3]: 6 Digite c[3]: 0 Digite a quantidade maxima de iteracoes: 20 Digite a precisao minima aceitavel: 0.00001 ***** ***** Iteracao Numero 1: x[1] = 7.0000000000E-01</p>	<p>Iteracao Numero 4: x[1] = 1.0004625000E+00 x[2] = 9.9888750000E-01 x[3] = -9.9967500000E-01 prec := 8.2124999999E-03 ***** ***** Iteracao Numero 5: x[1] = 1.0001575000E+00 x[2] = 9.9980062500E-01 x[3] = -9.9997906250E-01 prec := 1.5221874992E-03 ***** ***** Iteracao Numero 6: x[1] = 1.0000356875E+00 x[2] = 9.9996800000E-01 x[3] = -1.0000018438E+00 prec := 3.1196875079E-04 ***** ***** Iteracao Numero 7: x[1] = 1.0000067688E+00 x[2] = 9.9999538438E-01 x[3] = -1.0000010766E+00 prec := 5.7070314142E-05 ***** ***** Iteracao Numero 8: x[1] = 1.0000011384E+00 x[2] = 9.9999941531E-01 x[3] = -1.0000002769E+00 prec := 1.0460940757E-05 ***** ***** Iteracao Numero 9: x[1] = 1.0000001723E+00 x[2] = 9.9999993998E-01 x[3] = -1.0000000561E+00</p>
--	---

<pre> x[2] = 9.7500000000E-01 x[3] = -8.3750000000E-01 prec := 1.6625000000E+00 ***** ***** Iteracao Numero 2: x[1] = 9.7250000000E-01 x[2] = 9.8000000000E-01 x[3] = -9.7625000000E-01 prec := 4.1625000000E-01 ***** ***** Iteracao Numero 3: x[1] = 9.9925000000E-01 x[2] = 9.9462500000E-01 x[3] = -9.9693750000E-01 prec := 6.2062499999E-02 ***** ***** Iteracao Numero 5: x[1] = 1.0001575000E+00 </pre>	<pre> prec := 1.7115207811E-06 ***** ***** Houve convergencia do processo iterativo para a solucao do sistema em 10 iteracoes, com prec = 1.7115207811E- 06 Ultimos valores obtidos: x[1] = 1.0000001723E+00 x[2] = 9.9999993998E-01 x[3] = -1.0000000561E+00 prec = 1.7115207811E-06 </pre>
--	--

PROJETO 15 (Programa Não_Linear) – Faça um programa estruturado em Pascal para verificar se um dado conjunto de valores satisfaz a um sistema de equações não-lineares.

Solução – Vamos considerar o sistema não linear dado a seguir.

$$\begin{cases} 3\text{sen}(x) - 4y - 12z - 1 = 0 \\ 4x^2 - 8y - 10z + 5 = 0 \\ 2e^x + 2y + 3z - 8 = 0 \end{cases}$$

Para verificar se um conjunto de valores satisfaz esse sistema, podemos construir um algoritmo como segue:

ALGORITMO VERIFVALORES

```
REAL A, B, C
REAL VALF1, VALF2, VALF3
F1(X, Y, Z) = 3*SEN(X) - 4*Y - 12*Z - 1
F2(X, Y, Z) = 4*X*X - 8*Y - 10*Z + 5
F3(X, Y, Z) = 2*EXP(X) + 2*Y + 3*Z - 8
```

INÍCIO

(* ENTRADA DOS DADOS: *)

```
LEIA A, B, C
```

(* FIM DA ENTRADA DOS DADOS: *)

(* PROCESSAMENTO DOS CÁLCULOS: *)

```
VALF1 = F1(A, B, C)
VALF2 = F2(A, B, C)
VALF3 = F3(A, B, C)
VAL = ABS(VALF1) + ABS(VALF2) + ABS(VALF3)
```

(* FIM DO PROCESSAMENTO DOS CÁLCULOS: *)

(* SAÍDA DOS RESULTADOS: *)

```
ESCREVA 'DADOS DE ENTRADA: '
ESCREVA 'VALOR DE F1 = ', VALF1
ESCREVA 'VALOR DE F2 = ', VALF2
ESCREVA 'VALOR DE F3 = ', VALF3
SE (VAL) IGUAL 0.0) ENTÃO
    ESCREVA 'VALORES SATISFAZEM SISTEMA'
FIM SE
SE (VAL) NÃO IGUAL 0.0) ENTÃO
    ESCREVA 'VALORES NÃO SATISFAZEM SISTEMA'
FIM SE
```

(* FIM DA SAÍDA DOS RESULTADOS: *)

FIM DO ALGORITMO VERIFVALORES.

Tal algoritmo pode ser codificado em linguagem Pascal:

```
Program VerifValores;
uses crt; var
  x, y, z, val : real;
  valf1, valf2, valf3 :real;

  function f1(x,y,z:real):real;
  begin f1:=3*sin(x)-4*y-12*z-1; end;

  function f2(x,y,z:real):real;
  begin f2:=4*x*x-8*y-10*z+5; end;

  function f3(x,y,z:real):real;
  begin f3:=2*exp(x)+2*y+3*z-8; end;

begin
readkey;
{* ENTRADA DOS DADOS: *}
  writeln('Digite o valor de x:');readln(x);
  writeln('Digite o valor de y:');readln(y);
  writeln('Digite o valor de z:');readln(z);
{* FIM DA ENTRADA DOS DADOS. *}

{* PROCESSAMENTO DOS CÁLCULOS: *}
  valf1:=f1(x,y,z);
  valf2:=f2(x,y,z);
  valf3:=f3(x,y,z);
  val:=abs(valf1)+abs(valf2)+abs(valf3);
{* FIM DO PROCESSAMENTO DOS CÁLCULOS. *}

{* SAÍDA DOS RESULTADOS: *}
  writeln('Dados de Entrada:');
  writeln('Valor de f1 = ',valf1);readkey;
  writeln('Valor de f2 = ',valf2);readkey;
  writeln('Valor de f3 = ',valf3);readkey;
  if(val=0.0)then
    writeln('Valores satisfazem sistema');
  if(val<>0.0)then
    writeln('Valores NÃO satisfazem sistema');
{* FIM DA SAÍDA DOS RESULTADOS. *}
readkey;
end. {* FIM DO PROGRAMA VERIFVALORES. *}

```

Relatório de uma execução do programa:

Digite o valor de x:

1

Digite o valor de y:

0

Digite o valor de z:

2

Dados de Entrada:

Valor de f1 = -2.2475587046E+01

Valor de f2 = -1.1000000000E+01

Valor de f3 = 3.4365636569E+00

Valores NÃO satisfazem sistema

PROJETO 16 (Programa Newton) – Implemente um programa estruturado em linguagem Pascal para empregar o Método de Newton na resolução de um sistema de equações não-lineares.

Solução – Vamos considerar o sistema apresentado abaixo:

$$\begin{cases} f(x,y) = x + 2y - 1,9 = 0 \\ g(x,y) = 3x^2 + y^2 - 7,1 = 0 \end{cases}$$

Para resolver tal sistema pelo Método de Newton, construímos o algoritmo dado a seguir:

ALGORITMO NEWTONSIST

```
REAL X(100), Y(100), DET, DET1, DET2
REAL D, DMIN, B1, B2
REAL J11, J12, J21, J22, DELTA1, DELTA2
INTEIRO IT, ITMAX
F(X, Y) = X + 2*Y - 1,9
G(X, Y) = 3*X*X + Y*Y - 7,1
DFX(X, Y) = 1; DFY(X, Y) = 2
DGX(X, Y) = 6*X; DGY(X, Y) = 2*Y
```

INÍCIO

(* ENTRADA DOS DADOS: *)

```
LEIA X(1), Y(1), ITMAX, DMIN
```

(* FIM DA ENTRADA DOS DADOS. *)

(* PROCESSAMENTO DOS CÁLCULOS: *)

```
IT = 1; D = 100
```

```
ENQUANTO ((IT MENOR ITMAX) E (D MAIOR DMIN)) FAÇA
```

(* JACOBIANA: *)

```
J11 = DFX(X(IT), Y(IT)); J12 = DFY(X(IT), Y(IT))
```

```
J21 = DGX(X(IT), Y(IT)); J22 = DGY(X(IT), Y(IT))
```

(* FIM DA JACOBIANA. *)

(* TERMOS INDEPENDENTES: *)

```
B1 = - F(X(IT), Y(IT)); B2 = - G(X(IT), Y(IT))
```

(* FIM DOS TERMOS INDEPENDENTES. *)

(* RESOLUÇÃO POR CRAMER: *)

```
DET = J11*J22 - J12*J21
```

```
DET1 = B1*J22 - J12*B2; DET2 = J11*B2 - B1*J21
```

```
DELTA1 = DET1/DET; DELTA2 = DET2/DET
```

```
X(IT+1) = X(IT)+DELTA1; Y(IT+1) = Y(IT)+DELTA2
```

(* FIM DA RESOLUÇÃO POR CRAMER. *)

```
D = ABS(B1) + ABS(B2); IT = IT+1
```

```
FIM ENQUANTO
```

(* FIM DO PROCESSAMENTO DOS CÁLCULOS. *)

(* SAÍDA DOS RESULTADOS: *)

```
SE (D MENOR OU IGUAL DMIN) ENTÃO
    ESCRIVA 'PRECISÃO ALCANÇADA'
    ESCRIVA 'X = ',X(IT), 'Y = ',Y(IT)
FIM SE
SE (D MAIOR DMIN) ENTÃO
    ESCRIVA 'PRECISÃO NÃO ALCANÇADA'
FIM SE
```

(* FIM DA SAÍDA DOS RESULTADOS. *)

FIM DO ALGORITMO NEWTONSIST.

Implementação Pascal desse Algoritmo:

```
Program Newtsis;
uses crt; var
    x: array[1..100] of real;
    y: array[1..100] of real;
    det, det1, det2, d, dmin, b1, b2 : real;
    delta1, delta2, j11, j12, j21, j22 : real;
    it, itmax : integer;

    function f(x, y: real): real;
        begin f:= x+2*y-1.9; end;

    function g(x, y: real): real;
        begin g:=3*x*x+y*y-7.1; end;

    function dfx(x, y: real): real;
        begin dfx:= 1; end;

    function dfy(x, y: real): real;
        begin dfy:= 2; end;

    function dgx(x, y: real): real;
        begin dgx:= 6*x; end;

    function dgy(x, y: real): real;
        begin dgy:= 2*y; end;

begin
    clrscr;

    { * ENTRADA DOS DADOS: *}
    writeln('Digite os Dados de Entrada:');
```

```

writeln('Digite x[1] ='); readln(x[1]);
writeln('Digite y[1] ='); readln(y[1]);
writeln('Digite itmax ='); readln(itmax);
writeln('Digite dmin ='); readln(dmin);
{* FIM DA ENTRADA DOS DADOS. *}

{* PROCESSAMENTO DOS CÁLCULOS: *}
it:=1; d:=100;
while ((it < itmax) and (d > dmin)) do begin

    {* JACOBIANA: *}
    j11:=dfx(x[it],y[it]);
    j12:=dfy(x[it],y[it]);
    j21:=dgx(x[it],y[it]);
    j22:=dgy(x[it],y[it]);
    {* FIM DA JACOBIANA. *}
    {* TERMOS INDEPENDENTES: *}
    b1:=-f(x[it],y[it]); b2:=-g(x[it],y[it]);
    {* FIM DOS TERMOS INDEPENDENTES. *}
    {* RESOLUÇÃO POR CRAMER: *}
    det:=j11*j22-j12*j21;
    det1:=b1*j22-j12*b2; det2:=j11*b2-b1*j21;
    delta1:=det1/det; delta2:=det2/det;
    x[it+1]:=x[it]+delta1; y[it+1]:=y[it]+delta2;
    {* FIM DA RESOLUÇÃO POR CRAMER. *}
    d:=abs(b1)+abs(b2); it:=it+1;
end;{while}
readkey;
{* FIM DO PROCESSAMENTO DOS CÁLCULOS. *}

{* SAÍDA DOS RESULTADOS: *}
writeln('*****');
writeln('Saída dos Resultados:');
writeln(' ');
readkey;
if(d <= dmin) then begin
    writeln('Precisão Alcançada!');
    writeln(' ');
    writeln('x = ',x[it]);
    writeln('y = ',y[it]);
end;{if}
if(d > dmin) then
    writeln('Precisão NÃO alcançada!');
writeln(' ');
writeln('FIM. ');
writeln('*****');
readkey;
{* FIM DA SAÍDA DOS RESULTADOS. *}
end.{FIM DO PROGRAMA NEWTSIS.}

```


Uma Execução para Dados Particulares:

Digite os Dados de Entrada:

Digite x[1] =

-5

Digite y[1] =

-7

Digite itmax =

50

Digite dmin =

0.000001

Saída dos Resultados:

Precisão Alcançada!

x = -1.2424728221E+00

y = 1.5712364110E+00

FIM.

PROJETO 17 (Programa Quase_Newton) – Faça um programa Pascal estruturado para resolver um sistema não-linear por um método Quase-Newton.

Solução:

Implementaremos o algoritmo:

ALGORITMO QUASE_NEWTON

```
REAL    X(100), Y(100), D(100), DMIN
INTEIRO IT, ITMAX
F(X, Y) = X+2*Y-1.9
G(X, Y) = 3*X*X+Y*Y-7.11
DFX(X, Y) = 1
DGY(X, Y) = 2*Y
```

INÍCIO

(* ENTRADA DOS DADOS: *)

```
LEIA X(1), Y(1), ITMAX, DMIN
```

(* FIM DA ENTRADA DOS DADOS. *)

(* PROCESSAMENTO DOS CÁLCULOS: *)

```
IT = 1; D(1) = 100
```

```
ENQUANTO ((IT MENOR ITMAX) E (D(IT) MAIOR DMIN)) FAÇA
```

```
    X(IT+1) = X(IT) - (F(X(IT), Y(IT)) / DFX(X(IT), Y(IT)))
```

```
    Y(IT+1) = Y(IT) - (G(X(IT), Y(IT)) / DGY(X(IT), Y(IT)))
```

```
    D(IT+1) = ABS(X(IT+1) - X(IT)) + ABS(Y(IT+1) - Y(IT))
```

```
    IT = IT + 1
```

```
FIM ENQUANTO
```

(* FIM DO PROCESSAMENTO DOS CÁLCULOS. *)

(* SAÍDA DOS RESULTADOS: *)

```
SE (D(IT) MENOR OU IGUAL DMIN) ENTÃO
```

```
    ESCRIVA 'VALORES APROXIMADOS: '
```

```
    ESCRIVA 'X = ', X(IT), 'Y = ', Y(IT), 'DESVIO = ', D(IT)
```

```
FIM SE
```

```
SE (D(IT) MAIOR DMIN) ENTÃO
```

```
    ESCRIVA 'NÃO SE OBTVEVE RESULTADO PRECISO'
```

(* FIM DA SAÍDA DOS RESULTADOS. *)

FIM DO ALGORITMO QUASE_NEWTON.

Implementação Pascal Correspondente a Esse Algoritmo:

```
Program QuaseNewton;
uses crt;
var
  X:array[1..100] of real;
  Y:array[1..100] of real;
  D:array[1..100] of real;
  DMIN: real;
  IT, ITMAX: integer;
  function F(X,Y:real): real; begin F:= X+2*Y-1.9; end;
  function G(X,Y:real): real; begin G:= 3*X*X+Y*Y-7.11; end;
  function FX(X,Y:real): real; begin DFX:= 1; end;
  function DGY(X,Y:real): real; begin DGY:= 2*Y; end;

begin

  { * ENTRADA DOS DADOS: *}
  clrscr;
  writeln('Digite X1');    readln(X[1]);
  writeln('Digite Y1');    readln(Y[1]);
  writeln('Digite ITMAX'); readln(ITMAX);
  writeln('Digite DMIN');  readln(DMIN);
  { * FIM DA ENTRADA DOS DADOS. *}

  { * PROCESSAMENTO DOS CÁLCULOS: *}
  IT:= 1; D[1]:= 100;
  while((IT<ITMAX) and (D[IT]>DMIN))do begin
    X[IT+1]:=X[IT]-F(X[IT],Y[IT])/DFX(X[IT],Y[IT]);
    Y[IT+1]:=Y[IT]-G(X[IT],Y[IT])/DGY(X[IT],Y[IT]);
    D[IT+1]:= ABS(X[IT+1]-X[IT]) + ABS(Y[IT+1]-Y[IT]);
    IT:= IT+1;
  end; {while}
  { * FIM DO PROCESSAMENTO DOS CÁLCULOS. *}

  { * SAÍDA DOS RESULTADOS: *}
  writeln('*****');
  if D[IT] <= DMIN then begin
    writeln('Valores Aproximados:');
    writeln(' X = ',X[IT]);
    writeln(' Y = ',Y[IT]);
    writeln(' DESVIO = ',D[IT-1]); end; {if}
  if D[IT] > DMIN then
    writeln('NÃO SE OBTVEVE RESULTADO PRECISO');
  readkey;
  { * FIM DA SAÍDA DOS RESULTADOS. *}

end. {FIM DO PROGRAMA QuaseNewton}
```

Resultado de uma execução desse programa:

```
Digite X1
-1
Digite Y1
-1
Digite ITMAX
5
Digite DMIN
0.01
*****
NÃO SE OBTVE RESULTADO PRECISO
```

PROJETO 18 (Programa Tabela) – Construa um programa em linguagem Pascal para calcular e imprimir uma tabela com os valores correspondente às necessidades de um ajuste parabólico.

Solução – Inicialmente, construímos um algoritmo:

ALGORITMO TABELA

```
REAL X(50), Y(50), SX, SY
REAL YX(50), YX2(50), SYX, SYX2
REAL X2(50), X3(50), X4(50)
REAL SX2, SX3, SX4
IIINTEIRO I, N
```

INÍCIO

(* ENTRADA DOS DADOS: *)

```
LEIA N
PARA (I DE 1 ATÉ N) FAÇA
    LEIA X(I), Y(I)
FIM PARA
```

(* FIM DA ENTRADA DOS DADOS. *)

(* PROCESSAMENTO DOS CÁLCULOS: *)

(* CÁLCULO DAS PARCELAS: *)

```
PARA (I DE 1 ATÉ N) FAÇA
    YX(I) = Y(I)*X(I)
    YX2(I) = Y(I)*X(I)*X(I)
    X2(I) = X(I)*X(I)
    X3(I) = X(I)*X(I)*X(I)
    X4(I) = X(I)*X(I)*X(I)*X(I)
FIM PARA
```

(* FIM DO CÁLCULO DAS PARCELAS. *)

(* CÁLCULO DOS SOMATÓRIOS: *)

```
SX=0; SY=0; SYX=0; SYX2=0; SX2=0; SX3=0; SX4=0
```

```
PARA (I DE 1 ATÉ N) FAÇA
    SX = SX + X(I)
    SY = SY + Y(I)
    SYX = SYX + Y(I)*X(I)
    SYX2 = SYX2 + Y(I)*X(I)*X(I)
    SX2 = SX2 + X(I)*X(I)
    SX3 = SX3 + X(I)*X(I)*X(I)
    SX4 = SX4 + X(I)*X(I)*X(I)*X(I)
FIM PARA
```

(* FIM DO CÁLCULO DOS SOMATÓRIOS. *)

(* FIM DO PROCESSAMENTO DOS CÁLCULOS. *)

(* SAÍDA DOS RESULTADOS: *)

```
PARA (I DE 1 ATÉ N) FAÇA
    ESCREVA 'I = ', I
    ESCREVA 'X = ', X(I), 'Y = ', Y(I),
```

```

        ESCREVA 'X2 = ', X2(I), 'YX = ', YX(I), 'YX2 = ', YX2(I)
        ESCREVA 'X3 = ', X3(I), 'X4 = ', X(4)
    FIM PARA
    ESCREVA 'SOMATÓRIOS: '
    ESCREVA 'N=',N,'SX=',SX,'SY=',SY,'SYX=',SYX,'SYX2=',SYX2
    ESCREVA 'SX2=',SX2,'SX3=',SX3,'SX4=',SX4
(* FIM DA SAÍDA DOS RESULTADOS *)

```

FIM DO ALGORITMO TABELA.

Em seguida, realizamos uma codificação do algoritmo:

```

Program tabela;
uses crt; var
    x: array[1..50] of real;
    y: array[1..50] of real;
    x2: array[1..50] of real;
    yx: array[1..50] of real;
    yx2: array[1..50] of real;
    x3: array[1..50] of real;
    x4: array[1..50] of real;
    sx,sy,sx2,syx,syx2,sx3,sx4: real;
    i,n: integer;
begin
    clrscr;

    { * ENTRADA DOS DADOS: *}
    writeln('Digite a quantidade de pontos');
    readln(n);
    for i:=1 to n do begin
        writeln('Digite x'); readln(x[i]);
        writeln('Digite y'); readln(y[i]);
    end; {for}
    { * FIM DA ENTRADA DOS DADOS. *}

    { * PROCESSAMENTO DOS CÁLCULOS: *}
    { * CÁLCULO DAS PARCELAS: *}
    for i:=1 to n do begin
        x2[i]:= x[i]*x[i];
        x3[i]:= x[i]*x[i]*x[i];
        x4[i]:= x[i]*x[i]*x[i]*x[i];
        yx[i]:= y[i]*x[i];
        yx2[i]:= y[i]*x[i]*x[i];
    end; {for}
    { * FIM DO CÁLCULO DAS PARCELAS. *}
    { * CÁLCULO DOS SOMATÓRIOS: *}
    sx:=0;sy:=0;syx:=0;syx2:=0;sx2:=0;sx3:=0;sx4:=0;
    for i:=1 to n do begin

```

```

    sx:=sx+x[i];sy:=sy+y[i];
    syx:=syx+y[i]*x[i];
    syx2:=syx2+y[i]*x[i]*x[i];
    sx2:=sx2+x[i]*x[i];
    sx3:=sx3+x[i]*x[i]*x[i];
    sx4:=sx4+x[i]*x[i]*x[i]*x[i];
end; {for}
{* FIM DO CÁLCULO DOS SOMATÓRIOS. *}
{* FIM DO PROCESSAMENTO DOS CÁLCULOS. *}

{* SAÍDA DOS RESULTADOS: *}
readkey;
for i:=1 to n do begin
    writeln('...Parcelas:.....');
    writeln('i = ',i);
    writeln('x = ',x[i],' y = ',y[i]);
    writeln('x2 = ',x2[i],' yx = ',yx[i],' yx2 = ',yx2[i]);
    writeln('x3 = ',x3[i],' x4 = ',x4[i]);
    readkey;
end; {for}
writeln('.....');
writeln('...Somatórios:.....');
writeln('n = ',n,' sx = ',sx,' sy = ',sy);
writeln('syx = ',syx,' syx2 = ',syx2);
writeln('sx2 = ',sx2,' sx3 = ',sx3,' sx4 = ',sx4);
writeln('FIM'); readkey;
{* FIM DA SAÍDA DOS RESULTADOS. *}

end. {FIM DO PROGRAMA TABELA.}

```

Finalmente, executamos o programa para um conjunto de três pontos:

Digite a quantidade de pontos

3

Digite x

2

Digite y

-3

Digite x

1

Digite y

-2

Digite x

0

Digite y

-1

....Parcelas:.....

i = 1

x = 2.0000000000E+00 y = -3.0000000000E+00

x2 = 4.0000000000E+00 yx = -6.0000000000E+00 yx2 = -1.2000000000E+01

x3 = 8.0000000000E+00 x4 = 1.6000000000E+01

....Parcelas:.....

i = 2

x = 1.0000000000E+00 y = -2.0000000000E+00

x2 = 1.0000000000E+00 yx = -2.0000000000E+00 yx2 = -2.0000000000E+00

x3 = 1.0000000000E+00 x4 = 1.0000000000E+00

....Parcelas:.....

i = 3

x = 0.0000000000E+00 y = -1.0000000000E+00

x2 = 0.0000000000E+00 yx = 0.0000000000E+00 yx2 = 0.0000000000E+00

x3 = 0.0000000000E+00 x4 = 0.0000000000E+00

.....

....Somatórios:.....

n = 3 sx = 3.0000000000E+00 sy = -6.0000000000E+00

syx = -8.0000000000E+00 syx2 = -1.4000000000E+01

sx2 = 5.0000000000E+00 sx3 = 9.0000000000E+00 sx4 = 1.7000000000E+01

FIM

PROJETO 19 (Programa Ajuste_de_Parábola) – Faça um programa Pascal estruturado (Entrada dos Dados, Processamento dos Cálculos e Saída dos Resultados) para implementar um caso de ajuste parabólico pelo Método dos Mínimos Quadrados.

Solução – construímos, primeiramente, um algoritmo:

ALGORITMO PARABOLA

```
REAL X(50), Y(50)
REAL SX, SX2, SX3, SX4
REAL SY, SYX, SYX2
REAL D, D1, D2, D3, A1, A2, A3
INTEIRO I, N
```

INÍCIO

(* ENTRADA DOS DADOS; *)

```
LEIA N
PARA (I DE 1 ATÉ N) FAÇA
    LEIA X(I), Y(I)
FIM PARA
```

(* FIM DA ENTRADA DOS DADOS. *)

(* PROCESSAMENTO DOS CÁLCULOS: *)

(* CÁLCULO DOS SOMATÓRIOS: *)

```
SX = 0; SY = 0; SYX = 0; SX2 = 0
SYX2 = 0; SX3 = 0; SX4 = 0
PARA (I DE 1 ATÉ N) FAÇA
    SX = SX + X(I); SY = SY + Y(I); SYX = SYX + Y(I)*X(I)
    SX2 = SX2 + X(I)*X(I); SYX2 = SYX2 + Y(I)*X(I)*X(I)
    SX3 = SX3 + X(I)*X(I)*X(I); SX4 = SX4 + X(I)*X(I)*X(I)*X(I)
FIM PARA
```

(* FIM DO CÁLCULO DOS SOMATÓRIOS. *)

(* CÁLCULO DOS DETERMINANTES: *)

```
D = N*SX2*SX4 + SX*SX3*SX2 + SX2*SX3*SX
  - SX2*SX2*SX2 - SX3*SX3*N - SX4*SX*SX
D1 = SY*SX2*SX4 + SX*SX3*SYX2 + SX2*SX3*SYX
  - SX2*SX2*SYX2 - SX3*SX3*SY - SX4*SYX*SX
D2 = N*SYX*SX4 + SY*SX3*SX2 + SX2*SYX2*SX
  - SX2*SYX*SX2 - SX3*SYX2*N - SX4*SX*SY
D3 = N*SX2*SYX2 + SX*SYX*SX2 + SY*SX3*SX
  - SY*SX2*SX2 - SYX*SX3*N - SYX2*SX*SX
```

(* FIM DO CÁLCULO DOS DETERMINANTES. *)

(* USO DA REGRA DE CRAMER: *)

```
A1 = D1/D; A2 = D2/D; A3 = D3/D
```

(* FIM DO USO DA REGRA DE CRAMER. *)

(* FIM DO PROCESSAMENTO DOS CÁLCULOS. *)

(* SAÍDA DOS RESULTADOS: *)

```
ESCREVA 'AJUSTE DE: A1*X*X + A2*X + A3'
ESCREVA 'COEFICIENTES: '
```

```

    ESCREVA 'A1 = ', A1
    ESCREVA 'A2 = ', A2
    ESCREVA 'A3 = ', A3
(* FIM DA SAÍDA DOS RESULTADOS. *)

```

FIM DO ALGORITMO PARÁBOLA.

Implementação do algoritmo em Pascal:

```

Program Parabola;
uses crt; var
    x:array[1..50]of real;
    y:array[1..50]of real;
    sx,sx2,sx3,sx4:real;
    sy,syx,syx2:real;
    d,d1,d2,d3,a1,a2,a3:real;
    i,n:integer;
begin
    clrscr;

    { * Entrada dos dados: * }
    writeln('Ajuste Parabólico');
    writeln('Digite a quantidade de pontos:');
    readln(n);
    for i:=1 to n do begin
        writeln('digite x:'); readln(x[i]);
        writeln('digite y:'); readln(y[i]);
    end; {for}
    { * Fim da entrada dos dados. * }

    { * Processamento dos cálculos: * }
    { * Cálculo dos somatórios: * }
    sx:=0;sy:=0;syx:=0;sx2:=0;
    syx2:=0;sx3:=0;sx4:=0;
    for i:=1 to n do begin
        sx:=sx+x[i];sy:=sy+y[i];
        syx:=syx+y[i]*x[i];
        sx2:=sx2+x[i]*x[i];
        syx:=syx+y[i]*x[i];
        sx3:=sx3+x[i]*x[i]*x[i];
        sx4:=sx4+x[i]*x[i]*x[i]*x[i];
    end; {for}
    { * Fim do cálculo dos somatórios. * }
    { * Cálculo dos determinantes: * }
    d:=n*sx2*sx4+sx*sx3*sx2+sx2*sx3*sx
    -sx2*sx2*sx2 -sx3*sx3*n-sx4*sx*sx;
    d1:=sy*sx2*sx4+sx*sx3*syx2+sx2*sx3*syx
    -sx2*sx2*syx2-sx3*sx3*sy-sx4*syx*sx;

```

```

d2:=n*syx*sx4+sy*sx3*sx2+sx2*syx2*sx
-sx2*syx*sx2-sx3*syx2*n-sx4*sx*sy;

d3:=n*sx2*syx2+sx*syx*sx2+sy*sx3*sx
-sy*sx2*sx2-syx*sx3*n-syx2*sx*sx;
{* Fim do cálculo dos determinantes. *}
{* Uso da Regra de cramer: *}
a1:=d1/d; a2:=d2/d; a3:=d3/d;
{* Fim do uso da Regra de Cramer. *}
{* Fim do processamento dos cálculos. *}

{* Saída dos resultados: *}
writeln('Ajuste de P(x) = A1*x*x + A2*x + A3');
writeln('Coeficientes:');
writeln('A1 = ',a1);
writeln('A2 = ',a2);
writeln('A3 = ',a3);
writeln('Determinantes:');
writeln('D = ',d);
writeln('D1 = ',d1);
writeln('D2 = ',d2);
writeln('D3 = ',d3);
writeln('FIM');
readkey;
{* Fim da saída dos resultados. *}
end. {* Fim do programa parábola. *}

```

Resultado de uma execução do programa Parábola:

```

Ajuste Parabólico
Digite a quantidade de pontos:
4
digite x:
-1
digite y:
5
digite x:
-3
digite y:
4
digite x:
-2
digite y:
6
digite x:
-1
digite y:
6
Ajuste de P(x) = A1*x*x + A2*x + A3

```

Coefficientes:

$$A1 = -9.0300000000E+02$$

$$A2 = -1.1340000000E+03$$

$$A3 = -2.8700000000E+02$$

Determinantes:

$$D = 8.0000000000E+00$$

$$D1 = -7.2240000000E+03$$

$$D2 = -9.0720000000E+03$$

$$D3 = -2.2960000000E+03$$

FIM

PROJETO 20 (Programa Ajuste_Exponencial) – Faça um programa Pascal para implementar um ajuste exponencial. Use o Método dos Mínimos Quadrados.

Solução – Consideremos um algoritmo para ajustar a função exponencial $Y = (A)B^X$. Aplicando logaritmos naturais, $\ln Y = \ln A + (\ln B) x$, que pode ser associado ao ajuste linear $p(x) = a_1 + a_2 x$, fazendo $a_1 = \ln A \rightarrow A = e^{a_1}$ e $a_2 = \ln B \rightarrow B = e^{a_2}$. Nesse caso, precisamos calcular os somatórios $\sum x_i$, $\sum x_i^2$, $\sum \ln y_i$, $\sum x_i \ln y_i$, com $i = 1, 2, \dots, n$, em que n é a quantidade de pontos aos quais se deseja ajustar a curva exponencial.

ALGORITMO AJUSTE_EXPONENCIAL

(Usa o Método dos Mínimos Quadrados e logaritmos naturais para ajustar a função exponencial $Y = (A)B^X$ a um conjunto de n pontos. O sistema de duas equações lineares é resolvido com a Regra de Cramer. *)*

```
REAL X(50), Y(50), A, B, A1, A2
REAL SX(50), SX2(50), SLY(50), SXLY(50)
REAL DET, DET1, DET2
INTEIRO N, I
```

INÍCIO

(* ENTRADA DOS DADOS: *)

```
LEIA N
PARA (I DE 1 ATÉ N) FAÇA
    LEIA X(I), Y(I)
FIM PARA
```

(* FIM DA ENTRADA DOS DADOS. *)

(* PROCESSAMENTO DOS CÁLCULOS: *)

```
(* CÁLCULO DOS SOMATÓRIOS: *)
SX = 0; SX2 = 0; SLY = 0; SXLY = 0
PARA (I DE 1 ATÉ N) FAÇA
    SX = SX + X(I)
    SX2 = SX2 + X(I)*X(I)
    SLY = SLY + LN(Y(I))
    SXLY = SXLY + X(I)*LN(Y(I))
FIM PARA
```

(* FIM DO CÁLCULO DOS SOMATÓRIOS. *)

(* USO DA REGRA DE CRAMER: *)

```
DET = N*SX2 - SX*SX
DET1 = SLY*SX2 - SX*SXLY
DET2 = N*SXLY - SLY*SX
A1 = DET1/DET; A2 = DET2/DET
```

(* FIM DO USO DA REGRA DE CRAMER. *)

```
A = EXP(A1); B = EXP(A2)
```

(* FIM DO PROCESSAMENTO DOS CÁLCULOS. *)

```

(* SAÍDA DOS RESULTADOS: *)
    ESCREVA 'AJUSTE EXPONENCIAL  $Y = (A)*(B^X)$ '
    ESCREVA 'A = ', A, 'B = ', B
(* FIM DA SAÍDA DOS RESULTADOS. *)

```

FIM DO ALGORITMO AJUSTE_EXPONENCIAL.

Programa Pascal Correspondente:

```

program exponencial;

{ * Usa o Método dos Mínimos Quadrados
  E logaritmos naturais para ajustar a função
  Exponencial  $Y = AB^x$  a um conjunto de n
  pontos. Resolve-se o sistema de duas equações
  lineares com a Regra de Cramer. * }

uses crt; var
  x, y: array [1..50] of real;
  sx, sx2, sly, sxly: real;
  a, b, a1, a2, det, det1, det2: real;
  n, i: integer;

begin

  { * ENTRADA DOS DADOS: * }
  writeln('Digite a quantidade de pontos:');
  readln(n);
  for i:=1 to n do begin
    writeln('digite x(',i,')');
    readln(x[i]);
    writeln('digite y(',i,')');
    readln(y[i]);
  end; {for}
  { * FIM DA ENTRADA DOS DADOS. * }

  { * PROCESSAMENTO DOS CÁLCULOS: * }
  { * CÁLCULO DOS SOMATÓRIOS: * }
  sx:=0; sx2:=0; sly:=0; sxly:=0;
  for i:=1 to n do begin
    sx:=sx+x[i];
    sx2:=sx2+x[i]*x[i];
    sly:=sly+ln(y[i]);
    sxly:=sxly+x[i]*ln(y[i]);
  end; {for}

```

```

{* FIM DO CÁLCULO DOS SOMATÓRIOS. *}
{* USO DA REGRA DE CRAMER: *}
    det:=n*sx2-sx*sx;
    det1:=sly*sx2-sx*sxly;
    det2:=n*sxly-sly*sx;
    a1:=det1/det; a2:=det2/det;
{* FIM DO USO DA REGRA DE CRAMER. *}
    a:=exp(a1); b:=exp(a2);
    readkey;
{* FIM DO PROCESSAMENTO DOS CÁLCULOS. *}

{* SAÍDA DOS RESULTADOS: *}
    writeln('.....');
    writeln('Ajuste exponencial Y = A*B^X');
    writeln('A = ',a);
    writeln('B = ',b);
    writeln('.....');
    readkey;
{* FIM DA SAÍDA DOS RESULTADOS. *}

end. { Fim do programa exponencial. }

```

Resultado de uma execução do programa:

```
Digite a quantidade de pontos:
3
Digite x(1)
1
digite y(1)
1
digite x(2)
2
digite y(2)
10
digite x(3)
3
digite y(3)
50
.....
Ajuste exponencial  $Y = A * B^X$ 
A = 1.5874010520E-01
B = 7.0710678119E+00
.....
```


PROJETO 21 (*Programa Interpolação Linear*) – Faça um programa que realize uma interpolação linear. Mande ler as coordenadas cartesianas de dois pontos e um valor x_{barra} para o qual o programa deve calcular e imprimir o valor correspondente y_{barra} .

Solução:

```
program interplinar;
(* Dadas as coordenadas [x1; y1], [x2; y2]
de dois pontos e um valor de xbarra,
calcula ybarra por interpolação linear. *)
uses crt; var
x1,y1,x2,y2,xbarra,ybarra:real;
begin
clrscr;

(* ENTRADA DOS DADOS: *)
writeln('Interpolação linear: ');
writeln('.....');
writeln('Digite a abscissa x1 =');
readln(x1);
writeln('Digite a ordenada y1 =');
readln(y1);
writeln('Digite a abscissa x2 =');
readln(x2);
writeln('Digite a ordenada y2 =');
readln(y2);
writeln('Digite a abscissa xbarra =');
readln(xbarra);
writeln('.....');
(* FIM DA ENTRADA DOS DADOS. *)

(* PROCESSAMENTO DOS CÁLCULOS: *)
ybarra:= (y1*(x2-xbarra)
+y2*(xbarra-x1))/(x2-x1);
(* FIM DO PROCESSAMENTO DOS CÁLCULOS. *)

(* SAÍDA DOS RESULTADOS: *)
writeln('x1 = ',x1,' y1 = ',y1);
writeln('x2 = ',x2,' y2 = ',y2);
writeln('.....');
writeln('xbarra = ',xbarra);
writeln('ybarra = ',ybarra);
writeln('FIM');
(* FIM DA SAÍDA DOS RESULTADOS. *)
readkey;
end.
```

Relatório Impresso pelo Programa:

Interpolação linear:

.....

Digite a abscissa x1 =

1

Digite a ordenada y1 =

1

Digite a abscissa x2 =

2

Digite a ordenada y2 =

2

Digite a abscissa xbarra =

1.5

.....

x1 = 1.0000000000E+00 y1 = 1.0000000000E+00

x2 = 2.0000000000E+00 y2 = 2.0000000000E+00

.....

xbarra = 1.5000000000E+00

ybarra = 1.5000000000E+00

FIM

PROJETO 22 (*Programa Lagrange*) – Implemente de maneira computacional a Interpolação Lagrangeana.

Solução:

Vamos considerar a implementação do seguinte algoritmo:

ALGORITMO LAGRANGE

REAL X(5), Y(5), L(5), XINT, YINT
INTEIRO I, K, N

INÍCIO

(* ENTRADA DOS DADOS: *)

LEIA N, XINT
PARA (I DE 0 ATÉ N) FAÇA
 LEIA X(I), Y(I)
FIM PARA

(* FIM DA ENTRADA DOS DADOS. *)

(* PROCESSAMENTO DOS CÁLCULOS: *)

(* CÁLCULO DOS POLINÔMIOS DE LAGRANGE: *)

PARA (K DE 0 ATÉ N) FAÇA
 L(K) = 1
 PARA (I DE 0 ATÉ N) FAÇA
 SE (I NÃO IGUAL K) ENTÃO
 $L(K) = L(K) * ((XINT - X(I)) / (X(K) - X(I)))$
 FIM SE
 FIM PARA
FIM PARA

(* FIM DO CÁLCULO DOS POLINÔMIOS DE LAGRANGE. *)

(* CÁLCULO DO VALOR DO POLINÔMIO INTERPOLANTE: *)

YINT = 0
PARA (I DE 0 ATÉ N) FAÇA
 YINT = YINT + Y(I)*L(I)
FIM PARA

(* FIM DO CÁLCULO DO VALOR DO POLINÔMIO INTERPOLANTE. *)

(* FIM DO PROCESSAMENTO DOS CÁLCULOS. *)

(* SAÍDA DOS RESULTADOS: *)

(* SAÍDA DOS VALORES DOS POLINÔMIOS DE LAGRANGE: *)

PARA (I DE 0 ATÉ N) FAÇA
 ESCREVA 'L(' , I, ') = ' , L(I)
FIM PARA

(* FIM DA SAÍDA DOS VALORES DOS POLINÔMIOS DE LAGRANGE. *)

(* SAÍDA DO VALOR DO POLINÔMIO DE INTERPOLAÇÃO: *)

ESCREVA 'PARA X = ', XINT, 'TEM-SE Y = ', YINT
 (* FIM DA SAÍDA DO VALOR DO POLINÔMIO DE INTERPOLAÇÃO. *)

(* FIM DA SAÍDA DOS RESULTADOS. *)

FIM DO ALGORITMO LAGRANGE.

Implementação Pascal:

```

program lagrange;
(* Implementa a interpolação lagrangeana. *)
uses crt; var
i, k, n: integer;
xint, yint: real;
x, y, l: array[1..5] of real;

begin clrscr;

(* ENTRADA DOS DADOS: *)
writeln('Digite n ='); readln(n);
for i:=0 to n do begin
  writeln('Digite x('i,') ='); readln(x[i]);
  writeln('Digite y('i,') ='); readln(y[i]);
end; {for}
writeln('Digite xint ='); readln(xint);
(* FIM DA ENTRADA DOS DADOS. *)

(* PROCESSAMENTO DOS CÁLCULOS: *)
(* CÁLCULO DOS VALORES
DOS POLINÔMIOS DE LAGRANGE: *)
for k:=0 to n do begin
  l[k]:= 1;
  for i:=0 to n do begin
    if(i <> k) then begin
      l[k]:= l[k]*((xint-x[i])
/(x[k]-x[i]));
    end; {if}
  end; {for}
end; {for}
(* FIM DO CÁLCULO DOS VALORES
DOS POLINÔMIOS DE LAGRANGE. *)
(* CÁLCULO DO VALOR DO POLINÔMIO DE INTERPOLAÇÃO: *)
yint:= 0;
for i:=0 to n do begin
  yint:= yint + y[i]*l[i];
end; {for}
(* FIM DO CÁLCULO DO POLINÔMIO DE INTERPOLAÇÃO. *)
(* FIM DO PROCESSAMENTO DOS CÁLCULOS. *)
  
```

```

(* SAÍDA DOS RESULTADOS: *)
writeln('Interpolação Lagrangeana. ');
writeln('Valores dos Polinômios de Lagrange: ');
for i:=0 to n do begin
    writeln('L(',i,') = ',l[i]);
end; {for}
writeln('Resultado da Interpolação: ');
writeln('para x = ',xint,' tem-se y = ',yint);
writeln('FIM');
readkey;
(* FIM DA SAÍDA DOS RESULTADOS. *)
end.

```

Resultado de uma execução desse programa:

```

Digite n =
2
Digite x(0) =
1
Digite y(0) =
4
Digite x(1) =
1.5
Digite y(1) =
3
Digite x(2) =
2
Digite y(2) =
2
Digite xint =
1.35
Interpolação Lagrangeana.
Valores dos Polinômios de Lagrange:
L(0) = 1.9500000000E-01
L(1) = 9.1000000000E-01
L(2) = -1.0500000000E-01
Resultado da Interpolação:
para x = 1.3500000000E+00 tem-se y = 3.3000000000E+00
FIM

```

PROJETO 23 (Programa Trapézios) – Faça um programa Pascal para integrar a função $5e^x$ pela Regra dos Trapézios. Por favor, adote os seguintes limites de integração: inferior: 1,45; superior: 1,75. E considere a divisão desse intervalo em seis subintervalos de igual comprimento.

Solução:

Neste programa implementa-se o seguinte algoritmo:

ALGORITMO TRAPÉZIOS

REAL X(20), Y(20)

REAL A, B, H, INT

INTEIRO I, N

$F(X) = 8 * X * X * X - 5 * EXP(2 * X) - 12$

INÍCIO

(* ENTRADA DOS DADOS: *)

LEIA A, B, H

(* FIM DA ENTRADA DOS DADOS. *)

(* PROCESSAMENTO DOS CÁLCULOS: *)

$N = (B - A) / H$

(* CÁLCULO DAS COORDENADAS DOS PONTOS: *)

$X(0) = A; X(N) = B$

PARA (I DE 1 ATÉ N-1) FAÇA

$X(I) = X(I-1) + H$

$Y(I) = F(X(I))$

FIM PARA

(* FIM DO CÁLCULO DAS COORDENADAS DOS PONTOS. *)

(* CÁLCULO DA INTEGRAL: *)

$INT = 0$

PARA (I DE 0 ATÉ N-1) FAÇA

$INT = INT + (1/2) * (Y(I) + Y(I+1)) * H$

FIM PARA

(* FIM DO CÁLCULO DA INTEGRAL. *)

(* FIM DO PROCESSAMENTO DOS CÁLCULOS. *)

(* SAÍDA DOS RESULTADOS: *)

ESCREVA 'INTEGRAL = ', INT

(* FIM DA SAÍDA DOS RESULTADOS. *)

FIM DO ALGORITMO TRAPÉZIOS.

Implementação Pascal correspondente:

```
program trapezios;

(* Resolve integral definida
de uma função f(x)
com o Método de Newton-Côtes
(Regra dos Trapézios). *)

uses crt; var

x: array[1..20] of real;
y: array[1..20] of real;
a, b, h, int: real;
i, n: integer;
function f(x:real):real;
begin
    f:= 5*exp(x);
end; {function}
begin
    clrscr;
    (* ENTRADA DOS DADOS: *)
    writeln('Este programa resolve uma integral');
    writeln('com a Regra dos Trapézios. ');
    writeln('Digite o limite inferior, a: '); readln(a);
    writeln('Digite o limite superior, b: '); readln(b);
    writeln('Digite a quantidade de subintervalos, n: ');
    readln(n);
    (* FIM DA ENTRADA DOS DADOS. *)
    (* PROCESSAMENTO DOS CÁLCULOS: *)
    h:=(b-a)/n;
    (* CÁLCULO DAS COORDENADAS DOS PONTOS: *)
    x[1]:= a; y[1]:= f(x[1]);
    x[n+1]:= b; y[n+1]:= f(x[n+1]);
    for i:= 2 to n do begin
        x[i]:= x[i-1] + h;
        y[i]:= f(x[i]);
    end; {for}
    (* FIM DO CÁLCULO DAS COORDENADAS DOS PONTOS. *)
    (* CÁLCULO DA INTEGRAL: *)
    int:= 0;
    for i:= 1 to n do begin
        int:= int + (1/2)*(y[i] + y[i+1])*h;
    end; {for}
```

Resultado da execução desse programa:

Este programa resolve uma integral
com a Regra dos Trapézios.

Digite o limite inferior, a:

1.45

Digite o limite superior, b:

1.75

Digite a quantidade de subintervalos, n:

6

Dados Lidos:

a = 1.4500000000E+00 b = 1.7500000000E+00 n = 6

Passo h = 5.0000000000E-02

Coordenadas dos pontos:

x(1) = 1.4500000000E+00 y(1) = 2.1315572576E+01

x(2) = 1.5000000000E+00 y(2) = 2.2408445352E+01

x(3) = 1.5500000000E+00 y(3) = 2.3557350913E+01

x(4) = 1.6000000000E+00 y(4) = 2.4765162122E+01

x(5) = 1.6500000000E+00 y(5) = 2.6034899136E+01

x(6) = 1.7000000000E+00 y(6) = 2.7369736959E+01

x(7) = 1.7500000000E+00 y(7) = 2.8773013380E+01

.....
Valor da Integral = 7.4589943729E+00

FIM

PROJETO 24 (*Programa Simpson_1/3*) – Implemente de modo computacional a integração numérica com a Regra de Simpson 1/3.

Solução

```
program simpson;
(* integra com Simpson 1/3 *)
uses crt; var
i, n: integer;
s1, s2, h, int: real;
x, y: array[1..49] of real;
begin clrscr;

(* ENTRADA DOS DADOS: *)
writeln('Integração com Simpson 1/3');
writeln('Entrada dos Dados:');
writeln('Digite a quantidade de pontos, n:');
readln(n);
writeln('Digite o passo constante, h:');
readln(h);
for i:= 1 to n do begin
    writeln('Digite x(,i,) =');
    readln(x[i]);
    writeln('Digite y(,i,) =');
    readln(y[i]);
end; {for}
(* FIM DA ENTRADA DOS DADOS. *)

(* PROCESSAMENTO DOS CÁLCULOS: *)

(* CÁLCULO DE S1: *)
s1:= 0; i:= 2;
repeat
    s1:= s1 + y[i];
    i:= i + 2;
until i > n-1;
(* FIM DO CÁLCULO DE S1. *)

(* CÁLCULO DE S2: *)
s2:= 0; i:= 3;
repeat
    s2:= s2 + y[i];
    i:= i + 2;
until i > n-2;
(* FIM DO CÁLCULO DE S2. *)

(* CÁLCULO DA INTEGRAL: *)
int:= (h/3)*(y[1] + 4*s1 + 2*s2 + y[n]);
(* FIM DO CÁLCULO DA INTEGRAL: *)
```

```

(* FIM DO PROCESSAMENTO DOS CÁLCULOS. *)

(* SAÍDA DOS RESULTADOS. *)
readkey;
writeln('Integração com Simpson 1/3. Saída dos dados:');
writeln('n = ',n,'; h = ',h);
for i:= 1 to n do begin
    writeln('x(',i,') = ',x[i], ' y(',i,') = ',y[i]);
end; {for}
readkey;
writeln('Valores calculados: s1 = ',s1,' s2 = ',s2);
writeln('Valor da integral: ',int);
writeln('FIM');
readkey;
(* FIM DA SAÍDA DOS RESULTADOS. *)
end. {FIM DO PROGRAMA SIMPSON.}

```

Resultado de uma execução:

Integração com Simpson 1/3

Entrada dos Dados:

Digite a quantidade de pontos, n:

5

Digite o passo constante, h:

0.25

Digite x(1) =

2.00

Digite y(1) =

3.0183156

Digite x(2) =

2.25

Digite y(2) =

6.4017339

Digite x(3) =

2.5

Digite y(3) =

10.631737

Digite x(4) =

2.75

Digite y(4) =

15.800961

Digite x(5) =

3.00

Digite y(5) =

22.002478

Integração com Simpson 1/3. Saída dos dados:

n = 5; h = 2.5000000000E-01

x(1) = 2.0000000000E+00 y(1) = 3.0183156000E+00

x(2) = 2.2500000000E+00 y(2) = 6.4017339000E+00

x(3) = 2.5000000000E+00 y(3) = 1.0631737000E+01

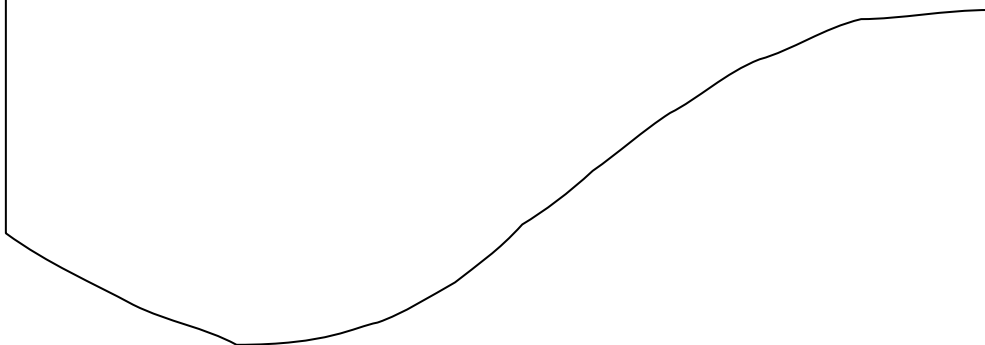
x(4) = 2.7500000000E+00 y(4) = 1.5800961000E+01

x(5) = 3.0000000000E+00 y(5) = 2.2002478000E+01

Valores calculados: s1 = 2.2202694900E+01 s2 = 1.0631737000E+01

Valor da integral: 1.1257920600E+01

FIM



PROJETO 25 (Programa Gauss_Dois_Pontos) – Faça um programa Pascal para calcular uma integral definida com o uso do Método de Gauss com dois pontos. Por favor, considere a função polinomial $f(x) = 4x^4 - 12$. Na Entrada dos Dados, dê diretamente os valores dos limites de integração: $a = 0$ e $b = 2$. No Processamento dos Cálculos, mande aplicar as fórmulas do método para calcular da integral. Na Saída dos Resultados, mande imprimir o resultado obtido.

SOLUÇÃO:

```

Program QuadGauss2pontos;
uses crt; var
x, a, b, a0,a1,t0,t1,x0,x1: Real;
Fmaiusct0, Fmaiusct1, Intgauss: Real;

function Fminusc(x:real): real;
begin
    Fminusc:= 4*x*x*x*x - 12;
end;
begin
    clrscr;
    (* ENTRADA DOS DADOS: *)
    a:= 0; b:= 2;

    (* PROCESSAMENTO DOS CÁLCULOS: *)
    a0:= 1; a1:= 1;
    t0:= -1/sqrt(3);
    t1:= +1/sqrt(3);
    x0:= ((b-a)/2)*t0 + (b+a)/2;
    x1:= ((b-a)/2)*t1 + (b+a)/2;
    Fmaiusct0 := ((b-a)/2)*Fminusc(x0);
    Fmaiusct1 := ((b-a)/2)*Fminusc(x1);
    Intgauss := a0*Fmaiusct0 + Fmaiusct1;

    (* SAÍDA DOS RESULTADOS: *)
    writeln('Integral = ',Intgauss);
    readkey;
end

```

Resultado da execução do programa:

Integral = 8.8888888888E-01.

PROJETO 26 (Programa Runge_Kutta_4) – Faça um programa Pascal para realizar uma aplicação do Método de Runge-Kutta de Quarta Ordem no caso da resolução de um Problema de Valor Inicial de Primeira Ordem.

SOLUÇÃO:

<pre> program rungekutta4; uses crt;var x,y:array[0..50]of real; h,k1,k2,k3,k4:real; i,qpassos:integer; function f(x,y:real):real; begin f:= - y + x + 2;end; {DADOS} begin clrscr; writeln('Digite o valor de x0:'); readln(x[0]); writeln('Digite o valor de y0:'); readln(y[0]); writeln('Digite o valor de h:'); readln(h); writeln('Digite o valor de qpassos:'); readln(qpassos); {PROCESSAMENTO} for i:=0 to qpassos-1 do begin K1:=f(x[i],y[i]); K2:=f(x[i]+h/2,y[i]+h*k1/2); K3:=f(x[i]+h/2,y[i]+h*k2/2); K4:= f(x[i]+h,y[i]+h*k3); x[i+1]:=x[i]+h; y[i+1]:=y[i]+(h/6)*(k1+2*k2+2*k3+k4); end; {RESULTADOS} for i:=0 to qpassos do begin writeln('i=',i,' x(,i,)=',x[i], y(,i,)=',y[i]); end; readkey; end. </pre>	<p>Digite o valor de x0:</p> <p>0.0</p> <p>Digite o valor de y0:</p> <p>2.0</p> <p>Digite o valor de h:</p> <p>0.1</p> <p>Digite o valor de qpassos:</p> <p>5</p> <p>i=0 x(0)= 0.0000000000E+00 y(0)= 2.0000000000E+00</p> <p>i=1 x(1)= 1.0000000000E-01 y(1)= 2.0048375000E+00</p> <p>i=2 x(2)= 2.0000000000E-01 y(2)= 2.0187309014E+00</p> <p>i=3 x(3)= 3.0000000000E-01 y(3)= 2.0408184220E+00</p> <p>i=4 x(4)= 4.0000000000E-01 y(4)= 2.0703202889E+00</p> <p>i=5 x(5)= 5.0000000000E-01 y(5)= 2.1065309344E+00</p>
---	---

PROJETO 27 (Programa Relaxação) – Realize um programa computacional codificado em linguagem Pascal para implementar o Método da Relaxação.

Solução:

```
program relaxacao;

    {Este programa resolve um sistema
    ax = b de n equações lineares pelo
    Método da Relaxação: prepara o sistema,
    calcula os resíduos, identifica o maior
    deles em valor absoluto e zera-o, agindo
    sobre a variável correspondente.}

    uses crt;var

        a:array[1..10,1..10] of real;
        b:array[1..10] of real;
        x,r:array[1..10,1..100] of real;
        n,i,j,it,itmax,max,iter:integer;
        prec,precmin,maxresid:real;
begin
    clrscr;

    { * ENTRADA DOS DADOS: *}

    writeln('Digite a quantidade n>0 de equações:');readln(n);
    writeln('Digite os coeficientes por linhas');
    writeln('Na diagonal, só valores não nulos!');
    for i:=1 to n do begin
        for j:=1 to n do begin
            writeln('Digite a['i','j,'] ='); readln(a[i,j]);
        end; {for}
        writeln('Digite b['i,'] ='); readln(b[i]);
    end; {for}
    { * LEITURA DAS ESTIMATIVAS INICIAIS: *}
    writeln('.....');
    writeln('Digite as estimativas iniciais:');
    for i:=1 to n do begin
        writeln('Digite x['i,1]=');readln(x[i,1]);
    end; {for}
    { * LEITURA DOS LIMITES PARA AS ITERAÇÕES: *}
    writeln('.....');
    writeln('Digite itmax =');readln(itmax);
    writeln('Digite precmin =');readln(precmin);
    { * FIM DA LEITURA DOS LIMITES PARA AS ITERAÇÕES. *}

    { * DADOS LIDOS: *}
end;
```

```

writeln('.....');
writeln('Dados Lidos:');
for i:=1 to n do begin
  for j:=1 to n do begin
    writeln('a['i','j']= ',a[i,j]);
  end; {for}
  writeln('b['i,']= ',b[i]);
end; {for}
writeln('itmax= ',itmax);
writeln('precmin= ',precmin);
{ * FIM DOS DADOS LIDOS. *}
{ * FIM DA ENTRADA DOS DADOS.*}
readkey;
{ * PROCESSAMENTO DOS CÁLCULOS: *}
{ * PREPARAÇÃO DO SISTEMA: *}
for i:=1 to n do begin
  for j:=1 to n do begin
    if(i<>j)then a[i,j]:=a[i,j]/-a[i,i];
  end; {for}
  b[i]:=b[i]/a[i,i]; a[i,i]:=-1;
end; {for}
writeln('.....');
writeln('Sistema Preparado:');
for i:=1 to n do begin
  for j:=1 to n do begin
    writeln('a['i','j']= ',a[i,j]);
  end; {for}
  writeln('b['i,']= ',b[i]);
end; {for}
writeln('.....');
{ * FIM DA PREPARAÇÃO DO SISTEMA. *}
{ * INICIALIZAÇÕES PARA CONTROLE: *}
it:=1; prec:=100.0;
{ * FIM DAS INICIALIZAÇÕES PARA CONTROLE. *}
{ * ITERAÇÕES: *}
while((it<itmax)and(prec>precmin))do begin
{ * CÁLCULO DOS RESÍDUOS: *}
  for i:=1 to n do begin
    r[i,it]:=b[i];
    for j:=1 to n do begin
      r[i,it]:=r[i,it]+a[i,j]*x[j,it];
    end; {for}
  end; {for}
{ * FIM DO CÁLCULO DOS RESÍDUOS. *}
{ * IDENTIFICAÇÃO DO MAIOR RESÍDUO: *}
  maxresid:= r[1,it]; max:=1;
  for i:= 2 to n do begin
    if(abs(r[i,it]) > abs(maxresid))then begin
      maxresid:=r[i,it]; max:=i;
    end; {if}

```

```

end; {for}
writeln('Maior Resíduo= ',maxresid);
writeln('Posição Do Maior Resíduo:',max);
{* FIM DA IDENTIFICAÇÃO DO MAIOR RESÍDUO. *}
{* NOVAS ESTIMATIVAS DA SOLUÇÃO: *}
writeln('.....');
writeln('Novas Estimativas Da Solução:');
for i:=1 to n do begin
  if(i = max)then
    x[i,it+1]:= x[i,it]+maxresid;
  {end if}
  if(i <> max)then
    x[i,it+1]:= x[i,it];
  {end if}
  writeln('x['i,',',it+1,']= ',x[i,it+1]);
readkey;
end; {for}
{* FIM DAS NOVAS ESTIMATIVAS DA SOLUÇÃO. *}
{* ATUALIZAÇÃO DAS VARIÁVEIS DE CONTROLE: *}
  it:= it+1; prec:= abs(maxresid);
{* FIM DA ATUALIZAÇÃO DAS VARIÁVEIS DE CONTROLE. *}
end; {while}
{* FIM DAS ITERAÇÕES. *}
{* FIM DO PROCESSAMENTO DOS CÁLCULOS. *}

readkey;

{* SAÍDA DOS RESULTADOS: *}
{* IMPRIMINDO A EVOLUÇÃO DOS CÁLCULOS: *}
writeln('*****');
writeln('Evolução dos Cálculos:');
readkey;
  for iter:= 2 to it do begin
    writeln(' ');
    writeln('Iteração N. ',iter,');
  readkey;
    for i:=1 to n do begin
      writeln('x['i,',',iter,']= ',x[i,iter]);
      writeln('  r['i,',',iter,']= ',r[i,iter]);
    readkey;
    end; {for}
  end; {for}
{* FIM DE IMPRIMINDO A EVOLUÇÃO DOS CÁLCULOS. *}
{* RESULTADO FINAL: *}
writeln('.....');
writeln('Resultado Final:');
readkey;
  if(prec <= precmin)then begin
    writeln('A Precisão Foi Alcançada!');
    for i:=1 to n do begin

```



```

readkey;
    writeln('x['i,',',it-1,']= ',x[i,it-1],
    '   r['i,',',it-1,']= ',r[i,it-1]);
end; {for}
end; {if}
if(prec > precmin)then
    writeln('A Precisão Não Foi Alcançada!');
{end if}
writeln('FIM.');
```

```

readkey;
{* FIM DO RESULTADO FINAL. *}
{* FIM DO ALGORITMO RELAXAÇÃO. *}
end.
```

Digite a quantidade $n > 0$ de equações:

3

Digite os coeficientes por linhas

Na diagonal, só valores não nulos!

Digite $a[1,1] =$

8

Digite $a[1,2] =$

2

Digite $a[1,3] =$

4

Digite $b[1] =$

2.1

Digite $a[2,1] =$

5

Digite $a[2,2] =$

5

Digite $a[2,3] =$

-3

Digite $b[2] =$

2.3

Digite $a[3,1] =$

3

Digite $a[3,2] =$

-6

Digite $a[3,3] =$

7

Digite $b[3] =$

4.4

.....

Digite as estimativas iniciais:

Digite $x[1,1] =$

1

Digite $x[2,1] =$

1

Digite $x[3,1] =$

3

.....

Digite $itmax =$

5

Digite $precmin =$

0.001

.....

Dados Lidos:

$a[1,1] = 8.0000000000E+00$

$a[1,2] = 2.0000000000E+00$

$a[1,3] = 4.0000000000E+00$

$b[1] = 2.1000000000E+00$

$a[2,1] = 5.0000000000E+00$

$a[2,2] = 5.0000000000E+00$

```

a[2,3]= -3.0000000000E+00
b[2]= 2.3000000000E+00
a[3,1]= 3.0000000000E+00
a[3,2]= -6.0000000000E+00
a[3,3]= 7.0000000000E+00
b[3]= 4.4000000000E+00
itmax= 5
precmin= 1.0000000000E-03
.....:
Sistema Preparado:
a[1,1]= -1.0000000000E+00
a[1,2]= -2.5000000000E-01
a[1,3]= -5.0000000000E-01
b[1]= 2.6250000000E-01
a[2,1]= -1.0000000000E+00
a[2,2]= -1.0000000000E+00
a[2,3]= 6.0000000000E-01
b[2]= 4.6000000000E-01
a[3,1]= -4.2857142857E-01
a[3,2]= 8.5714285714E-01
a[3,3]= -1.0000000000E+00
b[3]= 6.2857142857E-01
.....:
Maior Resíduo= -2.4875000000E+00
Posição Do Maior Resíduo:1
.....:
Novas Estimativas Da Solução:
x[1,2]= -1.4875000000E+00
x[2,2]= 1.0000000000E+00
x[3,2]= 3.0000000000E+00
Maior Resíduo= 2.7475000000E+00
Posição Do Maior Resíduo:2
.....:
Novas Estimativas Da Solução:
x[1,3]= -1.4875000000E+00
x[2,3]= 3.7475000000E+00
x[3,3]= 3.0000000000E+00
Maior Resíduo= 1.4782142857E+00
Posição Do Maior Resíduo:3
.....:
Novas Estimativas Da Solução:
x[1,4]= -1.4875000000E+00
x[2,4]= 3.7475000000E+00
x[3,4]= 4.4782142857E+00
Maior Resíduo= -1.4259821429E+00
Posição Do Maior Resíduo:1
.....:
Novas Estimativas Da Solução:
x[1,5]= -2.9134821429E+00
x[2,5]= 3.7475000000E+00

```

x[3,5]= 4.4782142857E+00

Evolução dos Cálculos:

Iteração N. 2:

x[1,2]= -1.4875000000E+00
r[1,2]= 0.0000000000E+00
x[2,2]= 1.0000000000E+00
r[2,2]= 2.7475000000E+00
x[3,2]= 3.0000000000E+00
r[3,2]= -8.7678571429E-01

Iteração N. 3:

x[1,3]= -1.4875000000E+00
r[1,3]= -6.8687500000E-01
x[2,3]= 3.7475000000E+00
r[2,3]= 0.0000000000E+00
x[3,3]= 3.0000000000E+00
r[3,3]= 1.4782142857E+00

Iteração N. 4:

x[1,4]= -1.4875000000E+00
r[1,4]= -1.4259821429E+00
x[2,4]= 3.7475000000E+00
r[2,4]= 8.8692857143E-01
x[3,4]= 4.4782142857E+00
r[3,4]= 0.0000000000E+00

Iteração N. 5:

x[1,5]= -2.9134821429E+00
r[1,5]= 0.0000000000E+00
x[2,5]= 3.7475000000E+00
r[2,5]= 0.0000000000E+00
x[3,5]= 4.4782142857E+00
r[3,5]= 0.0000000000E+00

x[2,3]= 3.7475000000E+00
r[2,3]= 0.0000000000E+00
x[3,3]= 3.0000000000E+00
r[3,3]= 1.4782142857E+00

Iteração N. 4:

x[1,4]= -1.4875000000E+00
r[1,4]= -1.4259821429E+00
x[2,4]= 3.7475000000E+00
r[2,4]= 8.8692857143E-01
x[3,4]= 4.4782142857E+00
r[3,4]= 0.0000000000E+00

Iteração N. 5:

```
x[1,5]= -2.9134821429E+00  
r[1,5]= 0.0000000000E+00  
x[2,5]= 3.7475000000E+00  
r[2,5]= 0.0000000000E+00  
x[3,5]= 4.4782142857E+00  
r[3,5]= 0.0000000000E+00
```

.....

Resultado Final:

A Precisão Não Foi Alcançada!

FIM.