Maxima no Cálculo

(Interface gráfica wxMaxima)

Ulysses Sodré
Andrielber da Silva Oliveira
Sônia Ferreira Lopes Toffoli
Giovana Alves
Isabela Belansson

Londrina-PR, 20 de Maio de 2009 Arq: maxcalc.tex

Conteúdo

Introdução

O Maxima ([1]) é um CAS (Computer Algebra System) que realiza muitas operações matemá-ticas simbólicas da lista apresentada abaixo. O Maxima (freeware) equivale ao Maple. Neste trabalho, usamos a interface gráfica wxMaxima para Windows, abordando as principais funções utilizadas nas operações matemáticas que estão dispo-níveis no Maxima. Não tivemos a menor intenção de abordar as milhares de funções existentes no Maxima. Existem alternativas para todos os outros grandes sistemas operacionais.

Na Seção I, nós apresentamos uma mini história do Maxima e citamos alguns detalhes sobre a instalação do Maxima. Mostramos todos os Menus do wxMaxima, bem como todas as suas alternativas. Destacamos atenção especial à linha de comando.

Na Seção II, realizamos cálculos numéricos no Maxima, usando a matemática: dos números racionais, dos números decimais (float) e dos números decimais com muitos dígitos (bigfloat). Tratamos também de cálculos de somatórios (somas) e produtos, tanto finitos como infinitos.

Na Seção III, estudamos cálculos algébricos com expansão e fatoração de expressões reais ou complexas, substituição de valores em expressões, resolução de sistemas lineares e não-lineares. Trabalhamos com precisão de muitos dígitos nas respostas dos problemas. Calculamos zeros de funções polinomiais de uma variável.

Na Seção IV, abordamos elementos de Álgebra Linear com listas, vetores, matrizes e suas principais operações e inversa de uma matriz por escalonamento.

Na Seção V, estudamos o Cálculo diferencial e Integral. Tratamos sobre limites, derivadas, diferenciais, integrais, frações parciais, integração por substituição, expansão de Taylor, comprimento de arco de uma curva e resolvemos equações diferenciais ordinárias.

Na seção VI, tratamos sobre transformações Trigonométricas, elementos de geometria diferencial no plano, equações recursivas, interpolação, aproximação e ajuste linear.

Na seção VII, realizamos aplicações no Cálculo de funções de várias variáveis, incluindo Derivadas parciais, Máximos e Mínimos com multiplicadores de Lagrange e com o Teorema de Sylvester, Sequências e Séries de números reais e Séries de potências.

Nas outras seções, estudamos Elementos de Geometria diferencial, Sequências recursivas e alguns elementos de Interpolação e Aproximação.

O núcleo de um CAS manipula simbolicamente expressões matemáticas, incluindo:

1. Simplificação de expressões, simplificação automática com hipóteses e restrições. Substituição e operações simbólicas ou numéricas com precisão arbitrária.
2. Mudança da forma de expressões: expansão, produtos e somas, fatoração, frações parciais, reescrever funções trigonométricas como exponenciais.
3. Diferenciação parcial e total. Estudos de otimização com e sem restrição.
4. Solução exata ou aproximada de sistemas de equações lineares ou não-lineares.
5. Solução de algumas equações diferenciais ordinárias e equações de diferenças.

[1] Maxima deve ser lida como Mæk-sima.

6. Cálculos de: limites, derivadas, integrais definidas e indefinidas, numéricas, inclusive em várias variáveis, transformadas integrais de Laplace e Fourier.
7. Operações com séries numéricas e de potências como expansão, soma e produto.
8. Operações com matrizes incluindo produto direto e cálculo de inversas.
9. Mostra de expressões matemáticas em uma forma matemática como em LaTeX.
10. Pacotes adicionais de Matemática aplicada, Física e Estatística.
11. Plotagem de gráficos 2D, 3D e paramétricos de funções bem como animação.
12. Inclusão de alguma linguagem de programação.

1 Detalhes gerais sobre o Maxima

1.1 Mini história do Maxima

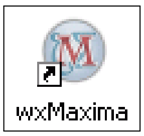
Maxima é um sistema de álgebra computacional, implementado em Lisp. O Maxima é derivado do sistema Macsyma, desenvolvido no MIT nos anos de 1968 a 1982 como parte do Projeto MAC. MIT remanejou uma cópia do código fonte do Macsyma para o Departamento de Energia em 1982; aquela versão é agora conhecida como Macsyma DOE. Uma cópia do Macsyma DOE foi mantida pelo Professor William F. Schelter da Universidade do Texas de 1982 até sua morte em 2001. Em 1998, Schelter obteve permissão do Departamento de Energia para liberar o código fonte do Macsyma DOE sob a Licença Pública GNU, e em 2000 ele iniciou o projeto Maxima no SourceForge para manter e desenvolver o Macsyma DOE, agora chamado Maxima.

1.2 Sobre a instalação do Maxima

1. Para executar o Maxima no Windows ou no Linux, digite

   Maxima: Linhas de comando
   ```
   maxima [ENTER]
   ```

2. Se você usa o `Windows`, eu sugiro que instale o `Maxima` na pasta `C:\Maxima` do seu computador.
3. Para este material, utilizamos a versão `5.14.0`, com a interface gráfica wxMaxima versão 0.7.4 tendo os comandos escritos em Português usado no Brasil.
4. O processo de instalação do `Maxima` no Windows, cria um ícone com a forma:

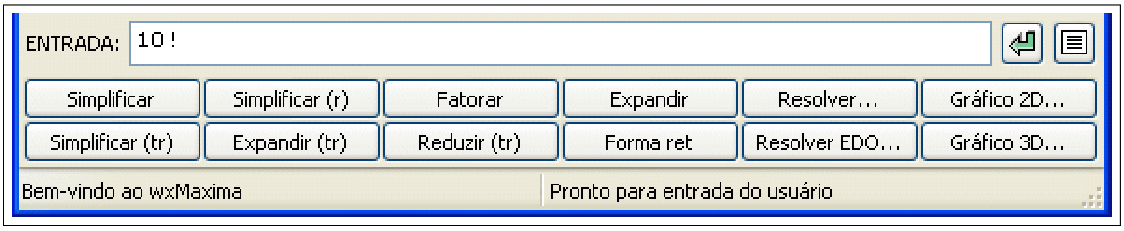


5. Podemos usar a interface gráfica `wxMaxima`, pressionando este ícone.
6. Após inicializar o programa `wxmaxima.exe`, você verá uma janela contendo algo como:

Maxima: Linhas de comando

```
/*
wxMaxima 0.7.4 http://wxmaxima.sourceforge.net
Maxima 5.14.0 http://maxima.sourceforge.net
Using Lisp GNU Common Lisp (GCL) GCL 2.6.8 (aka GCL)
Distributed under the GNU Public License. See the file COPYING.
Dedicated to the memory of William Schelter.
The function bug_report() provides bug reporting information.
(%i1)
```

7. Na última linha da tela mostrada acima, você verá o rótulo `(%i1)` (input1) que significa a primeira entrada da linha de comando do Maxima.
8. Na parte inferior da janela aberta há uma área com uma caixa de entrada:



9. Nesta figura, estão doze botões com as principais funções usadas no Maxima.
10. A caixa denominada `ENTRADA:` é muito importante. Através dela, podemos inserir as nossas expressões matemáticas. Observe que eu já digitei `10!` (fatorial de 10):

Maxima: Linhas de comando

```
10!
```

11. Tomando o cuidado de não alterar o código, (a menos que saiba o que está fazendo), digite `10!` e depois pressione a tecla `Enter` para obter:

Maxima: Linhas de comando

```
(%o1) 3628800;
```

12. A resposta `(%o1)` (output1) que aparece no notebook é o resultado da operação solicitada na linha de comando que tem a etiqueta `(%i1)`.
13. Muitas vezes indicaremos o código a ser digitado e a resposta por:

Maxima: Linhas de comando

```
10!;
>>> 3628800
```

14. Outras vezes, a resposta ficará centralizada após o código. Fatoramos a saída da última operação (%) com:

Maxima: Linhas de comando

```
factor(%);
```

$$2^8\,3^4\,5^2\,7$$

15. Para Encerrar a sessão do Maxima, pressione as teclas Ctrl-Q, isto é, pressione a tecla Ctrl com um dedo e `ao mesmo tempo` pressione a tecla Q com outro dedo.

1.3 Menus do wxMaxima e todas as suas alternativas

As entradas mais comuns do Maxima possuem teclas de atalho na interface `wxMaxima`, como por exemplo, as teclas Ctrl-O que serve para `Abrir` um `Arquivo`.

Arquivo Aqui ficam informações sobre arquivos de entrada e arquivos saída do Maxima.

- **Abrir** Ctrl-O Abrir um arquivo com a extensão `wxm`.
- **Read File** Ler um arquivo com a extensão `wxm`.
- **Salvar** Ctrl-S Salvar um arquivo com a extensão `wxm`.
- **Salvar como** Salvar um arquivo com um nome e extensão desejados pelo usuário.
- **Carregar pacote** Ctrl-L Carregar pacote com a extensão `wxm`.
- **Carregar arquivo batch** Ctrl-B Carrega um pacote com a extensão `bat`.
- **Exportar para HTML** Exporta o notebook aberto como uma página HTML.
- **Selecionar arquivo**
- **Monitorar arquivo**
- **Imprimir** Ctrl-P Imprimir o notebook aberto.
- **Sair** Ctrl-Q Encerrar a sessão do Maxima.

Editar Detalhes sobre o processo de edição do notebook.

- **Copiar** Ctrl-C Copia a parte selecionada para o clipboard.
- **Copiar texto** Ctrl-C Copia a parte selecionada como texto para o clipboard.
- **Copiar TeX** Copia a parte selecionada como LaTeX para o clipboard.
- **Apagar seleção** Elimina a entrada selecionada.
- **Copiar como imagem** Copia a parte selecionada como uma imagem.
- **Seleção para imagem** Seleciona uma imagem.
- **Seleção para entrada** F5
- **Recortar** Ctrl-X Copia a parte selecionada para o clipboard e elimina o mesmo do local onde a parte está.
- **Colar** Ctrl-V Cola o material que está no clipboard.
- **Desdobrar** Abrir a área etiquetada.
- **Entrada longa** Ctrl-I Abre janela de edição para entrada com várias linhas.
- **Editar entrada** Ctrl-E Edita a parte selecionada.
- **Reavaliar entrada** Ctrl-R Executa de novo a entrada selecionada.
- **Inserir** Inserir entrada, texto, nome e título da sessão.
  - **Entrada** F7 Insere uma expressão matemática.
  - **Texto** F6 Inserir texto.
  - **Sessão** Ctrl-F6 Inserir nome da sessão.
  - **Título** Ctrl-Shift-P Inserir título do capítulo desenvolvido.
- **Limpar a tela** Limpa a tela completamente.
- **Aproximar** Alt-I
- **Afastar** Alt-O
- **Ir à entrada** F4

Ir à janela de saída F3

Selecionar última entrada F2

Configurar Configurações gerais da interface wxMaxima.

Maxima Detalhes sobre o funcionamento do Maxima.

Interromper Ctrl-G Interrompe uma operação que está sendo realizada pelo Maxima.

Reiniciar Maxima Reinicia o Maxima, limpando as funções, limpando as variáveis e limpando as expressões que estavam na memória.

Limpar memória Limpa a memória, isto é, limpa todas as funções, variáveis e expressões que estavam na memória.

Adicionar ao caminho

Mostrar funções Mostra as funções usadas na sessão atual.

Mostrar definições Mostra as definições usadas na sessão atual.

Mostrar variáveis Mostra todas as variáveis usadas na sessão atual.

Apagar função Elimina uma função.

Apagar variável Elimina uma variável.

Alternar exibição do tempo Indica o tempo gasto em uma tarefa.

Alterar exibição 2D Alterna as expressões como `xml`, `ascii` ou `none`.

Mostrar forma TeX Mostra a expressão como uma expressão em TeX.

Equações Detalhes sobre equações.

Resolver ... Abre janela de diálogo para resolver uma equação exatamente.

Resolver numericamente ... Abre janela de diálogo para resolver uma equação numericamente.

Raízes de polinômio Obtém todas as raízes do polinômio indicado na passagem anterior com a função `allroots`.

Raízes do polinômio (real) Obtém todas as raízes reais do polinômio indicado na passagem anterior com a função `realroots`.

Resolver sistema linear ... Janela de diálogo para obter o número de equações do sistema e na seqüência abre outra janela para receber as equações e também as variáveis que serão obtidas na solução em função das equações.

Resolver sistema algébrico ... Janela de diálogo para obter o número de equações do sistema e na seqüência abre outra janela para receber as equações e também as variáveis que serão obtidas na solução em função das equações.

Eliminar variável ... Janela de diálogo para eliminar uma variável de uma expressão matemática.

Resolver EDO ... Janela de diálogo para resolver EDO.

Problema de valor inicial (1) ... Janela de diálogo para resolver um PVI1.

Problema de valor inicial (2) ... Janela de diálogo para resolver um PVI2.

Problema de valor de fronteira ... Janela de diálogo para resolver um PVF Problema de valor de fronteira.

Resolver EDO com Laplace ... Janela de diálogo para resolver uma EDO através do método de Laplace.

Valor no ponto ... Janela de diálogo para associar um valor a um ponto.

Álgebra Operações comuns da Álgebra e da Álgebra Linear.

Gerar matriz ... Janela de diálogo para gerar uma matriz com as entradas postas uma a uma, de uma forma lenta.

Introduzir matriz ... Janela de diálogo para inserir uma matriz com todas as entradas em uma caixa.

Inverter matriz Inverte a matriz indicada na última passagem.

Polinômio característico ... Janela de diálogo para gerar o polinômio característico de uma matriz.

Determinante Calcula o determinante de uma matriz.

Autovalores Calcula os autovalores (e suas multiplicidades) de uma matriz.

Autovetores Calcula os autovetores, autovalores (e suas multiplicidades).

Matriz adjunta Calcula a matriz adjunta de uma matriz dada.

Transpor matriz Calcula a transposta de uma matriz dada.

Criar lista ... Janela de diálogo para gerar uma lista de uma certa expressão, quando a variável percorre um certo domínio.

Aplicar lista ... Janela de diálogo para aplicar uma operação a uma lista existente.

Mapear lista ... Janela de diálogo para aplicar uma função a uma lista.

Mapear a uma matriz ... Janela de diálogo para aplicar uma função a uma matriz.

Cálculo Operações de Cálculo Diferencial e Integral.

Integrar ... Janela de diálogo para realizar a integral de uma função.

Integração Risch ... Janela de diálogo para a integral de Risch de uma função.

Mudar variável ... Janela de diálogo para realizar uma mudança de variável em uma expressão matemática.

Diferenciar ... Janela de diálogo para a derivada ou diferencial de uma função.

Encontrar limite ... Janela de diálogo para calcular o limite de uma função.

Obter série ... Janela de diálogo para gerar a série de Taylor de uma função.

Aproximação de Padé ... Janela de diálogo para gerar a aproximação de Padé.

Calcular soma ... Janela de diálogo para gerar a soma (o somatório) de uma expressão indexada sobre um certo domínio.

Calcular produto ... Janela de diálogo para gerar o produto de uma expressão indexada sobre um certo domínio.

Transformada de Laplace ... Janela de diálogo para calcular a transformada de Laplace de uma função.

Transformada inversa de Laplace ... Janela de diálogo para calcular a transformada inversa de Laplace de uma função.

Máximo divisor comum ... Janela de diálogo para calcular o máximo divisor comum entre polinômios.

Mínimo múltiplo comum ... Janela de diálogo para calcular o mínimo múltiplo comum entre polinômios.

Dividir polinômios ... Janela de diálogo para dividir polinômios e indicar o resto em uma lista.

Frações parciais ... Janela de diálogo para realizar a decomposição de uma expressão racional em frações parciais.

Fração contínua Gera uma fração contínua a partir de uma lista de números inteiros.

Simplificar Processos de simplificação de expressões.

- **Simplificar expressão** Simplificação racional da expressão com a função `ratsimp`. Ver botão Simplificar na área de baixo do notebook.
- **Simplificar radicais** Simplifica uma expressão que pode conter logaritmos, exponenciais e radicais, converte-a em uma expressão comum, usando a função `radcan`. Ver botão Simplificar (r) na área de baixo do notebook.
- **Fatorar expressão** Fatora uma expressão real com a função `factor`. Ver botão Fatorar na área de baixo do notebook.
- **Fatorar complexo** Fatorar expressão complexa com a função `gfactor`.
- **Expandir expressão** Expandir expressão com a função `expand`. Ver botão Expandir na área de baixo do notebook.
- **Expandir logaritmos** Expandir expressão com logaritmos.
- **Contrair logaritmos** Contrair expressão com logaritmos.
- **Fatoriais e gama** Realiza cálculos de fatoriais e com a função `gamma`.
  - **Converter para fatoriais** Converter binômios e as funções Beta e Gama para fatoriais.
  - **Converter para gama** Converter binômios, fatoriais e a função `beta` para função `gama`.
  - **Simplificar fatoriais** Simplificar uma expressão envolvendo fatoriais.
  - **Combinar fatoriais** Combinar uma expressão envolvendo fatoriais.
- **Simplificação trigonométrica** Simplificação de expressões trigonométricas.
  - **Simplificar trigonométricas** Simplifica expressão trigonométrica utilizando a função `trigsimp`. Ver botão Simplificar (tr) na área de baixo do notebook.
  - **Reduzir trigonométricas** Reduz expressão trigonométrica com a função `trigreduce`. Ver botão Reduzir (tr) na área de baixo do notebook.
  - **Expandir trigonométricas** Expande expressão trigonométrica utilizando a função `trigexpand`. Ver botão Expandir (tr) na área de baixo do notebook.
  - **Forma canônica** Simplifica expressão trigonométrica a uma forma canônica com a função `trigrat`.
- **Simplificação complexa** Realiza simplificações em expressões complexas.
  - **Converter para forma retangular** Converte uma expressão complexa para a forma retangular com a função `rectform`. Ver botão Forma ret na área de baixo do notebook.
  - **Converter para forma polar** Converte expressão complexa para a forma polar.
  - **Obter parte real** Gera a parte real de expressão complexa.
  - **Obter parte imaginária** Gera a parte imaginária de expressão complexa.
  - **Demoivre** Converte função exponencial de argumento imaginário para a forma trigonométrica.
  - **Exponencializar** Converte funções trigonométricas para a forma exponencial.
- **Substituir ...** Substituir variável em uma expressão.
- **Avaliar formas substantivas** Avaliar todas as formas substantivas na expressão.
- **Alternar flag algébrico**
- **Adicionar igualdade algébrica ...** Adicionar igualdade ao simplificador racional.
- **Cálculo com módulo ...** Configurar calculos com módulo.

Gráficos Realizar a construção de gráficos.

Gráficos 2D ... Janela de diálogo para a construção de gráficos 2D.
Gráficos 3D ... Janela de diálogo para a construção de gráficos 3D.
Formato do gráfico ... Janela de diálogo para ajustar o formato do gráfico.

Numérico Alternativas para saídas numéricas.

Alternar saída numérica Alternar saída numérica.
Para `float` Gera a saída `float` de uma expressão.
Para `bfloat` Gera a saída `bfloat` de uma expressão.
Ajustar precisão ... Janela de diálogo para ajustar precisão de ponto flutuante.

Ajuda Sistema de Ajuda do Maxima.

Ajuda do Maxima F1 Mostrar help do Máxima.
Descrever Ctrl-H Mostrar a descrição de comando.
Exemplo Mostra exemplos de uso de comando indicado.
Apropos Mostra comandos semelhantes ao comando indicado.
Mostrar dica Mostra dica do Maxima.
Informações da versão Informações sobre a Versão do Maxima.
Relatar bug Relatar bug no programa.
Sobre Informações sobre o Maxima

1.4 A linha de comando na interface wxMaxima

1. O Maxima pode ser usado em linha de comando com a interface `wxMaxima`. Você informa um comando, obtém uma resposta e pode inserir o comando seguinte.
2. O objeto `(%i1)` é uma *etiqueta*. Cada linha de entrada ou linha de saída possui uma etiqueta numerada.
3. As etiquetas numeradas podem ser usadas como referências em outras expressões até o final da sessão. *Nunca use nomes de variáveis* como `%i1` ou `%o5`, pois isto causará confusão com as linhas etiquetadas.
4. As etiquetas com `i` (input) denotam comandos e etiquetas com `o` (output) denotam respostas geradas pelo Maxima.
5. Todas as funções embutidas possuem nomes que estão em minúsculas como por exemplo: `sin`, `cos`, `save`, `load`, etc.
6. As principais constantes embutidas possuem nomes em minúsculas como por exemplo: `%e`, `%pi`, `inf`, `minf`, `infinity`, etc.
7. O Maxima é sensível ao contexto, isto é, letras maiúsculas e minúsculas são diferentes.
8. Se você digitar `SIN(x)` ou `Sin(x)`, o Maxima assume que você quer escrever alguma outra função embutida que é diferente da função `sin`.
9. Nomes de funções e variáveis definidas pelo usuário podem ter maiúsculas `func(XY)`, minúsculas `FUNC(xy)` ou ambas, como `Func(Xy)`, `Func(xY)`, que são diferentes.
10. Encerramos uma linha de comando com ponto e vírgula (;) seguido da tecla `Enter`.

11. Um comando simples é uma expressão com um ponto e vírgula ; no final, e neste caso, a resposta é mostrada na tela.

Maxima: Linhas de comando
```
2*5;
>>>  10
```

12. A tecla `Enter` sozinha não indica que você realizou a sua entrada, mas se você esquecer do ponto e vírgula no final da linha de comando da interface `wxMaxima`, a interface colocará este ponto para você.

13. Você pode escrever vários comandos na mesma linha de comando, desde que use o sinal de ponto e vírgula ; entre estes comandos:

Maxima: Linhas de comando
```
10^1; 10^3; 10^5;
>>> 10
    1000
    100000
```

14. Você pode realizar a mesma operação que antes usando a função `display`:

Maxima: Linhas de comando
```
display(10^1); display(10^3); display(10^5);
```

$$\begin{aligned} 10^1 &= 10 \\ 10^3 &= 1000 \\ 10^5 &= 100000 \end{aligned}$$

15. Você pode obter o mesmo resultado que antes usando o função `do`. Assim, `para cada i=1 até 5 com passo 2, mostra a potência` 10^i:

Maxima: Linhas de comando
```
for i:1 thru 5 step 2 do display(10^i);
```

16. Realizamos a mesma operação que antes usando o função `ldisplay`:

Maxima: Linhas de comando
```
ldisplay(a[1]:10^1); ldisplay(a[3]:10^3); ldisplay(a[5]:10^5);
```

$$\begin{aligned} a_1 &= 10 \\ a_3 &= 1000 \\ a_5 &= 100000 \end{aligned}$$

17. Podemos realizar a mesma operação que antes usando o função `do`:

Maxima: Linhas de comando
```
for i:1 thru 5 step 2 do ldisplay(a[i]:10^i);
```

$$\begin{aligned} a_1 &= 10 \\ a_3 &= 1000 \\ a_5 &= 100000 \end{aligned}$$

18. Podemos escrever vários comandos na mesma linha de comando, com `vírgulas` entre os comandos, envolvendo toda a linha com um par de colchetes `[ ]`:

Maxima: Linhas de comando
```
[10^2, 10^3, 10^4];
>>> [100, 1000, 10000]
```

19. Uma entrada com um dollar (`$`) no final, faz com que a resposta não apareça na tela. Isto é útil para resultados intermediários longos que não desejamos mostrar na tela.

Maxima: Linhas de comando
```
2^50$
```

20. Para criar uma `linha` de comando com várias linhas de texto, a interface `wxMaxima` possui a tecla de atalho `Ctrl+I` que abre uma janela e permite editar a expressão.
21. Um comando complexo pode exigir mais do que uma linha de comando e as linhas podem ser encerradas com um ponto e vírgula ou com um `dollar`.

Maxima: Linhas de comando
```
load("diag")$
A: matrix([2,0],[3,5])$
jordan(A)$
minimalPoly(%);
>>> (x-5)(x-2)
```

Assim, o polinômio minimal da matriz `A` é dado por $m(x) = (x-5)(x-2)$.

22. No Maxima, o sinal de multiplicação de expressões numéricas ou literais, é um asterisco $*$. Na notação comum da matemática, a resposta aparece sem o sinal.

Maxima: Linhas de comando
```
(x+2)*(x-3);
>>> (x-3)(x+2)
```

23. O SINAL `%` REPRESENTA A ÚLTIMA EXPRESSÃO CALCULADA PELO MAXIMA.
24. Para expandir o produto obtido no último resultado, basta digitar:

Maxima: Linhas de comando
```
expand(%);
```

$$x^2 - x - 6$$

25. O Maxima interpreta os expoentes por três formas distintas, sendo duas coincidentes:

Maxima: Linhas de comando
```
[x^3^4, x^(3^4), (x^3)^4];
```

$$[x^{81}, \quad x^{81}, \quad x^{12}]$$

26. Para associar um valor a uma variável, use dois pontos (:), que não é um sinal de igualdade (Observação: O sinal de igualdade é usado em equações).

27. No Maxima, podemos associar uma constante a um valor e depois utilizar esta constante em algum cálculo.

Maxima: Linhas de comando
```
(%i1) c:10; c^2;
(%o1)    10
(%o2)    100
```

28. Pondo <u>uma</u> aspa simples `'` (apóstrofe) antes da expressão, o Maxima `evita` o cálculo da expressão, gerando uma expressão denominada `substantiva` no Maxima.

Maxima: Linhas de comando
```
'c^2;
```

$$c^2$$

29. Pondo <u>duas</u> aspas simples `''` (apóstrofes) antes da expressão, o Maxima realizará o cálculo da expressão substantiva.

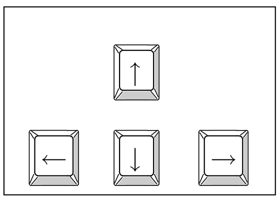
Maxima: Linhas de comando
```
''c^2;
>>> 100
```

30. Para repetir uma linha `(%i2)` que já foi digitada anteriormente, basta usar a etiqueta `''%i2` com <u>duas</u> aspas simples (`''`) antes dela. Não use aspas duplas!

Maxima: Linhas de comando
```
''%i2;
```

Esta forma repete a linha `(%i2)` sem digitar de novo, os comandos, funções e variáveis da linha toda.

31. Todos os comandos digitados em uma sessão do Maxima ficam armazenados na memória do computador, sob o nome de history da sessão. Para acessar tais comandos, basta pressionar as quatro setas de movimento do cursor:



32. Para repetir um comando que você já foi digitado, basta acessar o referido comando com as setas de movimento do cursor.
33. Se existe uma linha com a expressão `c^2;` basta você procurar tal linha com as setas de movimento (para cima e para baixo) e ao encontrar a mesma, pressionar `Enter` sem realizar todo o trabalho de digitação.
34. Para usar o último resultado calculado, use a respectiva etiqueta de saída `o` ou o símbolo especial de porcentagem (%).
35. Para interromper um cálculo sem sair do Maxima, pressione `Ctrl+G`.

36. Se, após digitar um código, você observa que o cálculo demora para realizar a operação:

Maxima: Linhas de comando
```
sum(1/x^2, x, 1, 1000000);
```
utilize a tecla de atalho `Ctrl+G` para interromper a operação.

37. Você verá uma tela com uma mensagem semelhante a esta:

Maxima: Linhas de comando
```
Maxima encountered a Lisp error:
 Console interrupt.
 Automatically continuing.
 To reenable the Lisp debugger set *debugger-hook* to nil.
```

38. As constantes e (base do logaritmo natural), i (raiz quadrada de -1) e pi (3.14159) são referidas como `%e`, `%i` e `%pi`, como:

Maxima: Linhas de comando
```
[ %pi , %e , %i]
```

39. O uso de % nas constantes nada tem a ver com o uso de `%` na referência ao cálculo.
40. Você pode obter ajuda sobre uma palavra como `zeromatrix` no Maxima, com:

Maxima: Linhas de comando
```
?? zeromatrix;
```

41. Você obterá uma resposta semelhante a:

Maxima: Linhas de comando
```
0: zeromatrix (Funções e Variáveis Def. Matrizes e Álgebra Linear)
1: zeromatrixp (Funções e Variáveis Definidas para linearalgebra)
Enter space-separated numbers, ‘all’ or ‘none’:
```

42. Para ver informações sobre `1: zeromatrixp`, digite `1` e pressione a tecla Enter. Você deverá ver uma tela semelhante a

Maxima: Linhas de comando
```
-- Função: zeromatrixp (<M>)
   Se <M> não for uma matriz de bloco, retorna ‘true’ se
   ‘is(equal(<e>,0))’ for verdadeiro para cada elemento <e> da
   matriz <M>. Se <M> for uma matriz de bloco, retorna ‘true’ se
   ‘zeromatrixp’ avaliar para ‘true’ para cada elemento de <e>.
   (%o6) true
```

43. Você pode obter um exemplo de como usar a palavra `laplace` no Maxima, com:

Maxima: Linhas de comando
```
example(laplace);
```

```
laplace(%e(a+2*t)*sin(t)*t, t, s)
```

$$\frac{e^{a}\,(2s-4)}{(s^2-4s+5)^2}$$

1.5 Uma pequena lista de funções do Maxima

Ver o manual de referência do Maxima `doc/html/maxima_toc.html` (na pasta principal de instalação do Maxima). No próprio Maxima, você pode usar `describe`(NomeFuncao).

`allroots`**(a)** Obtém todas as (em geral complexas) raízes da equação polinomial `A` e lista as mesmas em formato `numerico` (isto é, com 16 dígitos significativos).

`append`**(a,b)** Anexa a lista `b` à list `a`, resultando uma única lista.

`batch`**(a)** Carrega e roda um programa denominado `a`.

`coeff`**(a,b,c)** Toma a expressão `a` e fornece os coeficientes de `b` elevado à potência `c`.

`concat`**(a,b)** Concatena o símbolo `a` com o símbolo `b`, criando o símbolo `ab`.

`cons`**(a,b)** Acrescenta o objeto `a` à lista `b` como o seu primeiro elemento.

`demoivre`**(a)** Transforma as exponenciais complexas `a` para suas equivalentes trigonométricas.

`denom`**(a)** Fornece o denominador de `a`. Por exemplo:

Maxima: Linhas de comando
```
a:= (1+x)/(1-x^2);
denom(a)
```

`depends`**(a,b)** Declara `a` como uma função de `b`. Isto é útil para escrever derivadas não avaliadas, usadas em Equações Diferenciais.

`desolve`**(a,b)** Tenta resolver um sistema linear `a` de Equações Diferenciais Ordinárias para as incógnitas `b` usando Transformadas de Laplace.

`determinant`**(a)** Retorna o determinante de uma matriz quadrada `a`.

`diff`**(a,b1,c1,b2,c2,...,bn,cn)** Gera a derivada parcial mixta de `a` com respeito a cada `bi`, `ci` vezes. `diff(a,b,1)` pode ser escrita como `diff(a,b)`. `'diff(...)` gera uma derivada não avaliada, para ser usada em uma Equação Diferencial.

`eigenvalues`**(a)** Retorna duas listas, a primeira com os autovalores de uma matriz quadrada `a` e a segunda com as suas respectivas multiplicidades.

`eigenvectors`**(a)** Faz tudo que `eigenvalues` faz, e, além disso, anexa uma lista dos autovetores de `a`.

`entermatrix`**(a,b)** Solicita que o usuário insira uma matriz $a \times b$, elemento-a-elemento.

`ev`**(a,b1,b2,...,bn)** Avalia `a` sob as condições `bi`. Os termos `bi` podem ser equações, listas de equações (como as retornadas por `solve`), ou associações, nesses casos `ev` ”plugs”os `bi` dentro de `a`. Os `Bi` podem ser palavras como `numer` (e o resultado retorna em formato numérico), `detout` (as matrizes inversas em `a` são realizadas com o determinante fatorado fora), ou `diff` (as derivadas em `a` são avaliadas, isto é, `'diff` em `a` é trocada por `diff`). Por simplicidade, em um comando manual (isto é, não dentro de uma função definida pelo usuário), o `ev` pode ser quebrado, reduzindo a sintaxe para `a,b1,b2,...,bn`.

`expand`**(a)** Expande algebricamente `a`. Em particular, a multiplicação é distribuída sobre a adição.

`exponentialize`**(a)** Transforma todas as funções trigonométricas em `a` para as suas equivalentes exponenciais complexas.

`factor`**(a)** Fatora `a`.

`freeof`**(a,b)** Retorna `true` se a variável `a` não é parte da expressão `b`.

`grind`**(a)** Mostra uma variável ou função `a` em um formato compacto. Quando usado com `writefile` e um editor fora do Maxima, oferece um esquema para produzir arquivos `batch` que incluem expressões geradas pelo Maxima.

`ident`**(n)** Returna uma matriz identidade $n \times n$.

`imagpart`**(a)** Retorna a parte imaginária de `a`.

`integrate`**(a,b)** Tenta obter a integral indefinida de `a` com respeito à variável `b`.

`integrate(a,x,c,d)` Tenta calcular a integral indefinida de `a` na variável `x`, de `x=c` a `x=d`. Os limitantes de integração `c` e `d` podem ser infinito: `inf` (positivo) ou `minf` (negativo).

`invert`**(a)** Calcula a inversa da matriz quadrada `a`.

`kill`**(a)** Remove a variável `a` e as suas associações e propriedades do ambiente do Maxima.

`limit`**(a,x,c)** Fornece o limite da expressão `a` quando a variável `x` se aproxima de `c`. O valor de `c` pode ser tomado como `inf` ou `minf` como em `integrate`.

`lhs`**(a)** Fornece o lado esquerdo da equação `a`.

`loadfile`**(a)** Carrega um arquivo de disco com o nome `a` a partir do diretório padrão. O arquivo do disco deve estar no formato próprio (isto é, criado por um comando `save`.

`makelist`**(a,x,c,d)** Cria uma lista de `a`'s (cada um dos quais talvez dependa de `x`), concatenado de `x=c` até `x=d`.

`map`**(a,b)** Aplica a função `a` sobre as subexpressões de `b`.

`matrix`**(a1,a2,...,an)** Cria uma matriz consistindo de linhas `ai`, onde cada linha `ai` é uma lista de `m` elementos `[b1,b2,...,bm]`.

`num`**(a)** Fornece o numerador de `a`.

`ode2`**(Eq,y,x)** Tenta resolver equações diferenciais ordinárias `Eq` de primeira e segunda ordem na variável `y` como função da variável `x`.

`part`**(a,b1,b2,...,bn)** Primeiro toma a parte de ordem `b1` de `a`, depois a parte de ordem `B2` daquela e assim por diante.

`playback`**(m)** Mostra as últimas `m` etiquetas e suas expressões associadas. Se `m` é omitido, todas as linhas são mostradas. Ver o Manual para outras opções.

`ratsimp`**(a)** Simplifica `a` e retorna um quociente de dois polinômios.

`realpart`**(a)** Retorna a parte real de `a`.

`rhs`**(a)** Fornece o lado direito da equação `a`.

`save`**(a,b1,b2,..., bn)** Cria um arquivo com nome `a` no atual diretório padrão, com variáveis, funções ou arrays `bi`. O formato do arquivo permite que ele seja carregado no Maxima com o comando `loadfile`. Tudo (mesmo as etiquetas) pode ser `salvo` se `b1` é `all`.

`solve`**(Eq,x)** Tenta resolver a equação algébrica `Eq` para a incógnita `x`. Uma lista de soluções da equação é retornada. Mais simplesmente, se `a` é uma equação da forma `Eq=0`, ela pode ser abreviada simplesmente pela expressão `Eq`.

`string`**(a)** Converte a para a notação linear do Maxima (similar à notação de Fortran) exatamente como se ela tivesse sido digitada e coloca a no *buffer* para possível edição. A expressão na forma de `string` não deve ser usada nos cálculos.

`stringout`**(arq,b1,b2,...,bn)** Cria um arquivo denominado `arq` no atual diretório padrão, com variáveis (por exemplo, etiquetas) `b1,b2,...,bn`. O arquivo fica texto puro e não pode ser recarregado no Maxima, mas as expressões 'stringout' podem ser incorporadas em um programa Fortran, Basic ou C com pouca edição.

`subst`**(a,b,f)** Substitui `a` por `b` em `f`.

`taylor`**(f,x,a,n)** Expande `f` em uma série de Taylor na variável `x` em torno do ponto `x=a`, até incluir o termo $(x-a)^n$. Maxima também suporta expansões de Taylor em mais do que uma variável independente. Ver o Manual para detalhes.

`transpose`**(M)** Fornece a transposta da matriz `M`.

`trigexpand`**(f)** Expande uma função trigonométrica simplificada `f` que usa fórmulas da soma de ângulos para simplificar os argumentos individuais de `sin` ou `cos`. Por exemplo, `trigexpand(sin(x+y))` fornece `cos(x)sin(y)+sin(x)cos(y)`.

`trigreduce`**(f)** Função para simplificação trigonométrica de `f` que faz uso de identidades para converter produtos e potências de `sin` e `cos` em uma soma de termos, cada uma delas contendo somente um simples `sin` ou `cos`.
Por exemplo: `trigreduce(sin(x)^2)` fornece `(1-cos(2x))/2`.

`trigsimp`**(a)** Função de simplificação trigonométrica que troca `tan`, `sec`, etc., por seus equivalentes em `sin` e `cos`. Também reconhece a identidade $\sin^2(x)+\cos^2(x)=1$.

2 Cálculos numéricos

2.1 Cálculos com expressões reais

1. Vá ao menu `Maxima` e pressione `Reiniciar Maxima`, para limpar todas as funções, variáveis e expressões que estavam na memória do computador.
2. As operações aritméticas comuns são: `+` adição, `-` subtração, `*` multiplicação, `/` divisão, `^` exponencial, `^^` potência de matrizes, `.` produto de matrizes e `sqrt` raiz quadrada.
3. Saídas do Maxima são mostradas pela aritmética exata (racional).

Maxima: Linhas de comando
```
1/100 + 1/101;
```

$$\frac{201}{10100}$$

4. Vamos associar um valor irracional a uma letra a. Na tela vemos a sua forma simbólica:

Maxima: Linhas de comando
```
a: (1 + sqrt(2))^5;
```

$$(\sqrt{2}+1)^5$$

5. Expandimos o cálculo anterior e simplificamos com:

Maxima: Linhas de comando
```
expand(a);
```

$$29\sqrt{2}+41$$

6. Para que um resultado seja mostrado em notação decimal, basta acrescentar a expressão $\boxed{\text{,numer;}}$ no final da linha de comando.

Maxima: Linhas de comando
```
a, numer;
>>> 82.01219330881976
```

7. O mesmo resultado pode ser visto em notação decimal, com a linha de comando.

Maxima: Linhas de comando
```
float(a);
>>> 82.01219330881976
```

8. Nesta versão do Maxima, o acréscimo numer fornece 16 digitos significativos, sendo que o último pode ser aproximado, e este número de dígitos é controlado pela variável do Maxima fpprec, cujo valor padrão é `16`.
9. Para saber o número de dígitos que o Maxima está operando, digite:

Maxima: Linhas de comando
```
fpprec;
>>> 16
```

10. O Maxima oferece *alta precisão* de modo arbitrário com a função `bfloat`:

Maxima: Linhas de comando
```
bfloat(a);
>>> 8.201219330881975b1
```

11. Modificaremos o número de dígitos com fpprec, para que a saída tenha `50` dígitos:

Maxima: Linhas de comando
```
fpprec: 50;
>>> 50
```

12. Agora, solicitaremos o valor de `a` com o comando `bfloat`:

Maxima: Linhas de comando
```
bfloat(a);
>>> 8.2012193308819746562221553176641464233398437 5b1
```

13. Com estas alterações, obteremos números grandes exatos:

Maxima: Linhas de comando
```
40!;
>>> 815915283247897734345611269596115894272000000000
```

2.2 Cálculos de somas e produtos finitos e infinitos

1. Vá ao menu `Maxima` e pressione `Reiniciar Maxima`, para limpar todas as funções, variáveis e expressões que estavam na memória do computador.

2. Um somatório finito dos 100 primeiros números naturais:

   Maxima: Linhas de comando
   ```
   sum(n, n, 1, 100);
   >>> 5050
   ```

3. Somatórios finitos dos quadrados e cubos dos 100 primeiros números naturais:

   Maxima: Linhas de comando
   ```
   sum(n^2, n, 1, 100);
   >>> 338350
   sum(n^3, n, 1, 100);
   >>> 25502500
   ```

4. Um produto finito dos 20 primeiros números naturais:

   Maxima: Linhas de comando
   ```
   product(n, n, 1, 20);
   >>> 2432902008176640000
   ```

5. Produtos finitos dos quadrados e cubos dos 20 primeiros números naturais:

   Maxima: Linhas de comando
   ```
   product(n^2, n, 1, 20);
   >>> 5919012181389927685417441689600000000
   product(n^3, n, 1, 20);
   >>> 14400376622525549608547603031202889616850944000000000000
   ```

6. Fórmulas das somas finitas das potências unitárias, quadráticas e cúbicas dos `n` primeiros números naturais:

   Maxima: Linhas de comando
   ```
   [sum(i,i,1,n), sum(i^2,i,1,n), sum(i^3,i,1,n)], simpsum=true;
   ```

   $$[\frac{n^2+n}{2}, \frac{2n^3+3n^2+n}{6}, \frac{n^4+2n^3+n^2}{4}]$$

7. Um produto finito com fatores polinomiais tendo coeficientes simples:

   Maxima: Linhas de comando
   ```
   % Comentário: simplify_products:true;
   product(x+i, i, 1, 4);
   >>> (x+1)(x+2)(x+3)(x+4)
   ```

8. Um produto finito com fatores polinomiais tendo coeficientes funcionais:

   Maxima: Linhas de comando
   ```
   product(x+i*(i+1)/2, i, 1, 4);
   >>> (x+1)(x+3)(x+6)(x+10)
   ```

9. Um produto finito sem simplificação:

Maxima: Linhas de comando
```
product(k, k, 1, n);
```
$$\prod_{k=1}^{n} k$$

10. O mesmo produto finito do item anterior com simplificação:

Maxima: Linhas de comando
```
product(k, k, 1, n), simpproduct;
>>> n!
```

11. Um produto finito mais complicado:

Maxima: Linhas de comando
```
product(integrate(x^k,x,0,1), k, 1, n);
```
$$\prod_{k=1}^{n} \frac{1}{k+1}$$

12. Um produto finito de potências condicionais:

Maxima: Linhas de comando
```
product(if k<=5 then a^k else b^k, k, 1, 10);
```
$$a^1 \times a^2 \times a^3 \times a^4 \times a^5 \times b^6 \times b^7 \times b^8 \times b^9 \times b^{10} \quad = \quad a^{15} b^{40}$$

13. Um somatório finito de 10 frações é realizado com:

Maxima: Linhas de comando
```
sum(1/(i*(i+1)), i, 1, 100);
```

14. Três somas infinitas muito importantes:

Maxima: Linhas de comando
```
[A:sum(1/n^2,n,1,inf), B:sum(1/n^3,n,1,inf), C:sum(1/n^4,n,1,inf)];
```
$$[\sum_{n=1}^{\infty} \frac{1}{n^2}, \quad \sum_{n=1}^{\infty} \frac{1}{n^3}, \quad \sum_{n=1}^{\infty} \frac{1}{n^4}]$$

15. Tais séries convergem, e os valores das mesmas, são obtidas com a opção `simpsum=true`:

Maxima: Linhas de comando
```
[A, B, C], simpsum=true;
```
$$[\frac{\pi^2}{6}, \qquad \zeta(3), \qquad \frac{\pi^4}{90}]$$

16. Também podemos adicionar somatórios não avaliados:

Maxima: Linhas de comando
```
A + B + C;
```
$$\sum_{n=1}^{\infty} \frac{1}{n^2} + \sum_{n=1}^{\infty} \frac{1}{n^3} + \sum_{n=1}^{\infty} \frac{1}{n^4}$$

17. Podemos colocar todos os termos sob um mesmo sinal de soma com `sumcontract`:

Maxima: Linhas de comando
```
sumcontract(%);
```

$$\sum_{n=1}^{\infty} \frac{1}{n^2} + \frac{1}{n^3} + \frac{1}{n^4}$$

18. Agora vamos avaliar com a função `ev` (evaluation) esta soma

Maxima: Linhas de comando
```
ev(%,simpsum=true);
```

$$\zeta(3) + \frac{\pi^4}{90} + \frac{\pi^2}{6}$$

3 Cálculos algébricos

3.1 Expandindo, fatorando e substituindo

1. Vá ao menu `Maxima` e pressione `Reiniciar Maxima`, para limpar todas as funções, variáveis e expressões que estavam na memória do computador.
2. O Maxima é uma importante ferramenta computacional para realizar cálculos algébricos. Vemos isto quando o Maxima manipula expressões algébricas.
3. Aqui está uma expressão polinomial:

Maxima: Linhas de comando
```
(x + 3*y + x^2*y)^2;
```

$$(x^2 y + 3y + x)^2$$

4. Vamos expandir esta última expressão com:

Maxima: Linhas de comando
```
expand(%);
```

$$x^4 y^2 + 6x^2 y^2 + 9y^2 + 2x^3 y + 6xy + x^2$$

5. Vamos fatorar esta última expressão com:

Maxima: Linhas de comando
```
factor(%);
```

$$(x^2 y + 3y + x)^2$$

6. Vamos expandir a última expressão e substituir a variável `x` pela fração `5/z`:

Maxima: Linhas de comando
```
expand(%), x=5/z;
```

$$\frac{30y}{z} + \frac{150y^2}{z^2} + \frac{25}{z^2} + \frac{250y}{z^3} + \frac{625y^2}{z^4} + 9y^2$$

7. Utilizamos a função `ratsimp` (Simplificação racional) para colocar todos os termos sobre um mesmo denominador:

Maxima: Linhas de comando
```
ratsimp(%);
```

$$\frac{9y^2z^4+30yz^3+(150y^2+25)z^2+250yz+625y^2}{z^4}$$

8. Vamos fatorar esta última expressão com a função `factor`:

Maxima: Linhas de comando
```
factor(%);
```

$$\frac{(3yz^2+5z+25y)^2}{z^4}$$

3.2 Expressões com números complexos

1. Vá ao menu `Maxima` e pressione `Reiniciar Maxima`, para limpar todas as funções, variáveis e expressões que estavam na memória do computador.
2. Definimos uma expressão com números complexos como:

Maxima: Linhas de comando
```
w: 3 + k * %i;
>>> ik+3
```

3. A função `realpart` retorna a parte real de um complexo:

Maxima: Linhas de comando
```
realpart(w);
>>> 3
```

4. A função `imagpart` retorna a parte imaginária de um complexo:

Maxima: Linhas de comando
```
imagpart(w);
>>> k
```

5. Podemos obter uma potência de um complexo com expoente real:

Maxima: Linhas de comando
```
w^4;
```

$$(i\,k+3)^4$$

6. Podemos expandir a última expressão com:

Maxima: Linhas de comando
```
expand(%);
```

$$k^4-12ik^3-54k^2+108ik+81$$

7. Podemos multiplicar um complexo por uma exponencial do tipo `e^u`:

Maxima: Linhas de comando
```
w^2 * e^{%i * k + 3};
```

$$e^{ik+3}\,(ik+3)^2$$

8. Podemos multiplicar um complexo por uma exponencial do tipo `exp(u)`:

Maxima: Linhas de comando
```
w^2 * exp(%i * k + 3);
```

$$(ik+3)^2\,e^{ik+3}$$

3.3 Resolução de um sistema linear

1. Vá ao menu `Maxima` e pressione `Reiniciar Maxima`, para limpar todas as funções, variáveis e expressões que estavam na memória do computador.
2. O Maxima pode obter a solução exata de um sistema linear. Criaremos um sistema de equações lineares, com as equações indicados pelos nomes `eq1`, `eq2` e `eq3`.

Maxima: Linhas de comando
```
eq1: 2*x + 3*y - 4*z = 1;
eq2: 7*x - 1*y - 3*z = 3;
eq3: 1*x + 1*y + 2*z = 5;
```

3. A função `linsolve` resolve o sistema linear, recebendo duas listas, a primeira com as equações `eq1`, `eq2` `eq3` e a segunda com as variáveis `x,y,z`.

Maxima: Linhas de comando
```
linsolve([eq1,eq2,eq3], [x,y,z]);
```

4. A resposta é uma `lista` em um par de colchetes `[...]` com a solução do sistema:

$$[x=\frac{94}{81},\quad y=\frac{103}{81},\quad z=\frac{104}{81}]$$

5. Vamos estudar um outro exemplo.

Maxima: Linhas de comando
```
eq1: x + z = y;
eq2: 2*a*x - y = 2*a^2;
eq3: y - 2*z = 2;
```

6. Para gerar uma saída apropriada, usaremos uma variável de controle sobre o processo de solução e o modo de programação:

Maxima: Linhas de comando
```
[globalsolve:false, programmode:true];
```

7. Agora podemos obter a resposta com:

Maxima: Linhas de comando
```
linsolve([eq1,eq2,eq3], [x,y,z]);
>>> [x=a+1, y=2a, z=a-1]
```

3.4 Resolução de um sistema algébrico não linear

1. Vá ao menu Maxima e pressione `Reiniciar Maxima`, para limpar todas as funções, variáveis e expressões que estavam na memória do computador.

2. O Maxima pode resolver um sistema não linear de equações algébricas, obtendo soluções exatas. Criaremos um sistema de equações, em que as equações são indicadas pelas variáveis `eq1`, `eq2` e `eq3`.

Maxima: Linhas de comando

```
eq1: x + y*z = 1;
eq2: y - x*z = 0;
eq3: x + y = 5;
```

3. A função `solve` que resolve o sistema, recebe uma lista com duas listas, a primeira com as equações `eq1`, `eq2` `eq3` e a segunda com as variáveis `x,y,z`.

Maxima: Linhas de comando

```
solve([eq1,eq2,eq3], [x,y,z]);
```

$$\begin{array}{ll} \text{Solução1 : } [& [x = \frac{25\sqrt{79}i+25}{6\sqrt{79}i-34}, y = \frac{5\sqrt{79}i+5}{\sqrt{79}i+11}, z = \frac{\sqrt{79}i+1}{10}], \\ \text{Solução2 : } & [x = \frac{25\sqrt{79}i-25}{6\sqrt{79}i+34}, y = \frac{5\sqrt{79}i-5}{\sqrt{79}i-11}, z = -\frac{\sqrt{79}i-1}{10}] \quad] \end{array}$$

4. A resposta é uma `lista` em dois colchetes `[...]` e cada colchete contém uma solução distinta para as equações simultâneas. Para simplificar números complexos com racionalização, usamos:

Maxima: Linhas de comando

```
G: rectform(%);
```

$$\begin{array}{ll} [& [x = \frac{11}{4} - \frac{\sqrt{79}i}{4}, y = \frac{\sqrt{79}i}{4} + \frac{9}{4}, z = \frac{\sqrt{79}i}{10} + \frac{1}{10}], \\ & [x = \frac{\sqrt{79}i}{4} + \frac{11}{4}, y = \frac{9}{4} - \frac{\sqrt{79}i}{4}, z = \frac{1}{10} - \frac{\sqrt{79}i}{10}] \quad] \end{array}$$

3.5 Trabalhando com precisão nas respostas

1. Para reduzir a precisão nos cálculos para 3 dígitos, digite:

Maxima: Linhas de comando

```
fpprec:3; bfloat(G);
>>> [[x=2.75b0 -2.22b0 i, y=2.22b0 i +2.25b0, z=8.89b-1 i +1.0b-1],
     [x=2.22b0 i +2.75b0, y=2.25b0 -2.22b0 i, z=1.0b-1-8.89b-1 i] ]
```

2. Simplificamos a última expressão, com:

Maxima: Linhas de comando

```
ratsimp(%);
```

$$\left[\quad [x=-\frac{364\,i-451}{164},y=\frac{364\,i+369}{164},z=\frac{80\,i+9}{90}],\ [x=\frac{364\,i+451}{164},y=-\frac{364\,i-369}{164},z=-\frac{80\,i-9}{90}]\quad\right]$$

3.6 Zeros de polinomiais de uma variável

1. Vá ao menu `Maxima` e pressione `Reiniciar Maxima`, para limpar todas as funções, variáveis e expressões que estavam na memória do computador.
2. O Cálculo de zeros (raízes) exatos de polinomiais de uma variável é um problema `fechado` que aprendemos na escola, mas pode ser usado para dar exemplos de artifícios úteis.
3. Seja uma função polinomial `f=f(x)` de grau 2, que atribuímos à variável `poli`:

   Maxima: Linhas de comando
   ```
   poli: x^2-x-12;
   ```
   $$x^2-x-12$$
4. Para calcular as raízes da equação polinomial `f(x)=0`, usamos a função `solve`:

   Maxima: Linhas de comando
   ```
   sol: solve(poli=0,x);
   >>> [x=-3,x=4]
   ```
5. Agora vamos reconstruir a polinomial original, mostrando as duas partes da resposta `sol`:

   Maxima: Linhas de comando
   ```
   p1: part(sol,1);
   >>> x=-3
   p2: part(sol,2);
   >>> x=4
   ```
6. Na parte `p1` há uma expressão à esquerda que pode ser obtida com a função `lhs` e uma expressão à direita que pode ser obtida com a função `rhs`. Na parte `p2` há uma expressão à esquerda que pode ser obtida com a função `lhs` e outra expressão à direita que pode ser obtida com a função `rhs`.

   Maxima: Linhas de comando
   ```
   lhs(p1);
   >>> x
   rhs(p1);
   >>> -3
   lhs(p2);
   >>> x
   rhs(p2);
   >>> 4
   ```
7. Os fatores `fat1` e `fat2` podem ser construídos como

Maxima: Linhas de comando
```
fat1: lhs(p1)-rhs(p1);
>>> x+3
fat2: lhs(p2)-rhs(p2);
>>> x-4
```

8. Agora podemos obter a expressão polinomial dada inicialmente:

Maxima: Linhas de comando
```
p: expand(fat1 * fat2);
```

$$x^2 - x - 12$$

9. Uma outra forma de recuperar a expressão original, realizando em diversos passos.
10. Primeiro reescrevemos as equações como termos, usando a função `map` do Maxima que aplica uma certa função a todos os elementos de uma lista.

Maxima: Linhas de comando
```
Lista: map( lambda([eq],lhs(eq)-rhs(eq)), sol );
>>> [x+3,x-4]
```

11. Observação: A função usada para aplicar os elementos da lista é uma função especial do Maxima, denominada `lambda`, e no nosso caso tem a forma:

Maxima: Linhas de comando
```
lambda( [eq], lhs(eq)-rhs(eq) )
```

12. Uma expressão `lambda` inicia com o símbolo `lambda` seguido por parênteses. Dentro dos parênteses fica a lista de parâmetros formais escrita em colchetes e uma expressão algébrica. Sabemos que uma expressão será atribuída ao parâmetro formal quando a função for chamada. Subtraímos o lado direito do lado esquerdo para obter um `fator`.
13. Para verificar que tais termos são os fatores do polinômio `poli`, multiplicamos estes fatores, usando agora a função `apply` que aplica uma operação aritmética aos elementos de uma lista. Para transformar uma lista de termos em um produto, aplicamos multiplicação aos elementos daquela lista:

Maxima: Linhas de comando
```
apply("*", Lista);
>>> (x-4)(x+3)
```

14. Isso é um polinômio na forma fatorada. Expansão nos fornece o polinômio inicial:

Maxima: Linhas de comando
```
expand(%);
```

$$x^2 - x - 12$$

15. A função `solve` obtém zeros de polinomial de uma variável, pondo os zeros em uma lista de equações. A função `map` mapeia uma função para todos os elementos da lista e responde com uma lista que serve para reescrever uma lista de equações como uma lista de termos. A função `apply` aplica uma operação a todos os elementos de uma lista e responde com um resultado simples e serve para multiplicar os elementos de uma lista.

4 Elementos de Álgebra Linear

4.1 Listas no Maxima

1. `Reiniciar o Maxima`, limpando as funções, variáveis e expressões da memória do PC.
2. Criando duas listas com comprimentos diferentes:

Maxima: Linhas de comando
```
A: [10,20,30];
>>> [10,20,30]
B: [40,50,60,70,80,90];
>>> [40,50,60,70,80,90]
```

3. Exibindo o primeiro elemento da lista B

Maxima: Linhas de comando
```
B[1]
>>> 40
```

4. Anexando duas listas para formar uma terceira lista:

Maxima: Linhas de comando
```
C: append(A,B);
>>> [10,20,30,40,50,60,70,80,90]
```

5. Obtendo o comprimento de uma lista (número de elementos da lista):

Maxima: Linhas de comando
```
length(C);
>>> 9
```

6. Modificando o primeiro elemento da lista B

Maxima: Linhas de comando
```
B[1]: 15;
>>> 15
```

7. Mostrando a lista B já modificada:

Maxima: Linhas de comando
```
B;
>>> [15,50,60,70,80,90]
```

4.2 Operações com vetores

1. `Reiniciar o Maxima`, limpando as funções, variáveis e expressões da memória do PC.
2. Para obter o oposto do vetor-linha u=(1,2,3), basta digitar

Maxima: Linhas de comando
```
u: [1,2,3]; -u
>>> [-1,-2,-3]
```

3. Para somar o vetor u como o seu oposto -u, basta digitar

Maxima: Linhas de comando
```
u + (-u)
>>> [0,0,0]
```

4. Para somar os vetores-linha (listas) u=(1,2,3) e v=(4,5,6), basta digitar:

Maxima: Linhas de comando
```
u: [1,2,3]; v: [4,5,6]; u+v;
>>> [1,2,3]
    [4,5,6]
    [5,7,9]
```

5. Multiplicamos o escalar k=5 pelo vetor-linha v=(1,2,3) com a operação $*$:

Maxima: Linhas de comando
```
k:5; v:[2,1,3]; k*v;
>>> 5
    [2, 1, 3]
    [10, 5, 15]
```

6. Para construir o conjunto `{u,v,w}` formado pelos vetores u=(1,2), v=(3,4) e w=(5,6), do espaço R2, podemos digitar

Maxima: Linhas de comando
```
u:[1,2]; v:[3,4]; w:[5,6];
>>> [1, 2]
    [3, 4]
    [5, 6]
```

7. Criaremos uma combinação linear `C` de `{u,v,w}`, com os escalares `x`, y e `z`:

Maxima: Linhas de comando
```
C: x*u + y*v + z*w
>>> [5 z + 3 y + x, 6 z + 4 y + 2 x]
```

8. O sistema linear `C` é formado por 2 equações (combinações lineares) obtidas por:

Maxima: Linhas de comando
```
Eq1: C[1];            /* Primeira linha de C */
>>>  5 z + 3 y + x
Eq2: C[2];            /* Segunda linha de C */
>>>  6 z + 4 y + 2 x
```

9. Para resolver um sistema linear `C` nas três variáveis `x,y,z`, digitamos

Maxima: Linhas de comando
```
linsolve([Eq1,Eq2], [x,y,z]);
>>> [x = %r1, y = -2 %r1, z = %r1]
```

10. Pela resposta, as variáveis `x`, `y` e `z` dependem de uma variável genérica `%r1`, o que significa que o conjunto `{u,v,w}` é linearmente dependente.

11. Verificamos a dependência linear do conjunto {u,v,w} resolvendo o sistema linear nas variáveis x,y, (Em R2, existem no máximo dois vetores LI) tomando:

Maxima: Linhas de comando
```
linsolve([Eq1,Eq2], [x,y]);
>>>  [x = z, y = -2z]
```

12. Para construir um conjunto {u,v,w} formado pelos vetores u=(1,2,3), v=(4,5,6) e w=(7,8,1), do espaço R3, digitamos

Maxima: Linhas de comando
```
u: [1,2,3]; v: [4,5,6]; w: [7,8,1];
```

13. Para verificar se o conjunto de vetores {u,v,w} é linearmente dependente, construímos o sistema linear

Maxima: Linhas de comando
```
S: x*u + y*v + z*w
```

14. Resolvemos o sistema linear com a função linsolve, digitando

Maxima: Linhas de comando
```
linsolve([S[1],S[2],S[3]], [x,y,z]);
```

Pela resposta: $[x = 0, \ y = 0, \ z = 0]$, o conjunto {u,v,w} é linearmente independente.

15. Para escrever o vetor m=(11,12,13) como combinação dos vetores do conjunto {u,v,w} do item anterior, construímos um processo quase igual ao anterior. Como o conjunto {u,v,w} já está na memória do computador, basta inserir o vetor m:

Maxima: Linhas de comando
```
m: [11,12,13];
```

16. Para gerar z como combinação de {u,v,w}, construímos o sistema linear:

Maxima: Linhas de comando
```
S: x*u + y*v + z*w - m
```

17. Resolvemos o sistema linear, digitando

Maxima: Linhas de comando
```
linsolve([S[1],S[2],S[3]], [x,y,z]);
```

$$[x = -\frac{7}{3}, y = \frac{10}{3}, z = 0]$$

Pela resposta acima, segue que

$$m = -\frac{7}{3}u + \frac{10}{3}v + (0)w$$

18. Para escrever um vetor m=(a,b,c) como combinação de {u,v,w} do item anterior, repetimos o processo anterior, trocando o vetor (11,12,13) pelo vetor (a,b,c). Como {u,v,w} já está na memória do computador, basta inserir apenas o vetor:

Maxima: Linhas de comando
```
m: [a,b,c];
```

19. Para escrever o vetor `m` como combinação linear dos vetores do conjunto `{u,v,w}`, construímos o sistema linear `S`:

Maxima: Linhas de comando
```
S: x*u + y*v + z*w - m
```

20. Resolvemos o sistema linear, digitando:

Maxima: Linhas de comando
```
linsolve([S[1],S[2],S[3]], [x,y,z])
```

A resposta fornece

$$[x = -\frac{3c-38b+43a}{24}, y = \frac{3c-10b+11a}{12}, z = -\frac{c-2b+a}{8}]$$

significando que

$$m = -\frac{3c-38b+43a}{24}u + \frac{3c-10b+11a}{12}v + \frac{c-2b+a}{8}w$$

Concluímos que o vetor `m=(a,b,c)` de R3 é combinação linear dos vetores do conjunto `{u,v,w}`, que gera o espaço R3, logo, `{u,v,w}` é uma base para o espaço vetorial R3.

4.3 Principais operações com matrizes

1. Vá ao menu `Maxima` e pressione `Reiniciar Maxima`, para limpar todas as funções, variáveis e expressões que estavam na memória do computador.
2. O Teorema Fundamental de Álgebra Linear garante que toda transformação linear `T` está associada a uma matriz `M` e reciprocamente, toda matriz `M` define uma transformação linear. Com este teorema, podemos obter propriedades de transformações lineares pelas propriedades das respectivas matrizes associadas.
3. O Maxima pode calcular: determinante, inversa, autovalores e autovetores de matrizes com elementos simbólicos (isto é, elementos envolvendo variáveis algébricas.)
4. Iniciaremos inserindo uma matriz com o nome `m`, elemento-a-elemento:

Maxima: Linhas de comando
```
m: entermatrix(2,2);
```

5. Após pressionar `Enter` vemos na tela a seguinte pergunta:

Maxima: Linhas de comando
```
Is the matrix  1.Diagonal 2.Symmetric 3.Antisymmetric 4.General
Answer 1, 2, 3 or 4 :
```

6. Digite:

Maxima: Linhas de comando
```
4. General (Geral)
>>> 4
```

7. O Maxima começa a mostrar as posições das entradas para que você insira cada uma separadamente. Digite de acordo com o que está indicado abaixo:

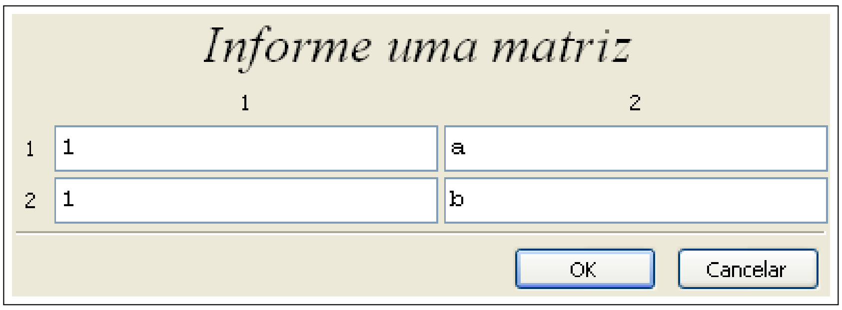
Maxima: Linhas de comando
```
Row 1 Column 1:  (Linha 1 Coluna 1)
1
Row 1 Column 2:  (Linha 1 Coluna 2)
a
Row 2 Column 1:  (Linha 2 Coluna 1)
1
Row 2 Column 2:  (Linha 2 Coluna 2)
b
```

Finalmente, você vê a mensagem `Matrix entered` e aparece a matriz completa:

$$\begin{bmatrix} 1 & a \\ 1 & b \end{bmatrix}$$

8. Outro modo de inserir esta matriz é acionar o menu `Álgebra`, `Introduzir matriz...`, para ver uma janela de diálogo semelhante a

9. Pressione $\boxed{\text{OK}}$ para ver uma janela, que pode ser preenchida da forma seguinte:



10. Pressionando $\boxed{\text{OK}}$, veremos algo como

$$\begin{bmatrix} 1 & a \\ 1 & b \end{bmatrix}$$

11. A forma mais rápida de criar uma matriz A com 2 linhas e 2 colunas, é digitar:

Maxima: Linhas de comando
```
A: matrix([1,3],[7,5]);
```

$$\begin{bmatrix} 1 & 3 \\ 7 & 5 \end{bmatrix}$$

12. Podemos somar a matriz A com ela própria, digitando

Maxima: Linhas de comando
```
A+A;
```

$$\begin{bmatrix} 2 & 6 \\ 14 & 10 \end{bmatrix}$$

13. Podemos multiplicar o escalar 2 pela matriz A, com:

Maxima: Linhas de comando
```
2*A;
```

para obter o mesmo resultado que antes:

$$\begin{bmatrix} 2 & 6 \\ 14 & 10 \end{bmatrix}$$

14. Digitando `A^2`, obtemos uma matriz 2x2 que é uma potência não usual, sendo que cada entrada de `A` está ao quadrado da respectiva entrada de A.

Maxima: Linhas de comando
```
A^2;
```

$$\begin{bmatrix} 1 & 9 \\ 49 & 25 \end{bmatrix}$$

15. Digitando `A*A`, obtemos um produto elemento-a-elemento da matriz `A`, que é um produto não usual em Álgebra Linear:

Maxima: Linhas de comando
```
A*A;
```

$$\begin{bmatrix} 1 & 9 \\ 49 & 25 \end{bmatrix}$$

16. O produto usual de matrizes, por exemplo, $A \times A$ é obtido, digitando:

Maxima: Linhas de comando
```
A.A;
```

$$\begin{bmatrix} 22 & 18 \\ 42 & 46 \end{bmatrix}$$

17. A mesma resposta que `A.A;` é obtida com a potência usual de matrizes, com o código:

Maxima: Linhas de comando
```
A^^2;
```

18. Podemos obter a transposta de uma matriz A, digitando:

Maxima: Linhas de comando
```
transpose(A);
```

$$\begin{bmatrix} 1 & 7 \\ 3 & 5 \end{bmatrix}$$

19. Obtemos agora o traço de uma matriz A, digitando:

Maxima: Linhas de comando
```
mat_trace(A);
>>> 6
```

20. Obtemos agora a adjunta de uma matriz A, digitando:

Maxima: Linhas de comando
```
adjoint(A);
```

$$\begin{bmatrix} 5 & -3 \\ -7 & 1 \end{bmatrix}$$

21. Obtemos agora o determinante de uma matriz A, digitando:

Maxima: Linhas de comando
```
determinant(A);
>>> -16
```

22. Realizamos a divisão da adjunta da matriz A pelo determinante da matriz A:

Maxima: Linhas de comando
```
adjoint(A)/determinant(A);
```

$$\begin{bmatrix} -\frac{5}{16} & \frac{3}{16} \\ \frac{7}{16} & -\frac{1}{16} \end{bmatrix}$$

23. Obtemos a inversa de uma matriz A, digitando:

Maxima: Linhas de comando
```
invert(A);
```

$$\begin{bmatrix} -\frac{5}{16} & \frac{3}{16} \\ \frac{7}{16} & -\frac{1}{16} \end{bmatrix}$$

o que significa que

$$A^{-1} = \frac{\mathrm{adj}(A)}{\det(A)}$$

24. Vamos calcular de novo a adjunta de uma matriz A:

Maxima: Linhas de comando
```
adjoint(A);
```

$$\begin{bmatrix} 5 & -3 \\ -7 & 1 \end{bmatrix}$$

25. Agora obteremos a inversa da matriz a, com o modificador `detout`:

Maxima: Linhas de comando

```
invert(A), detout;
```

$$-\frac{\begin{bmatrix} 5 & -3 \\ -7 & 1 \end{bmatrix}}{16}$$

26. A inversa parece uma fração, e o modificador `detout` (`determinant out`) informa que o determinante no denominador (fora da inversa) e a adjunta no numerador.

27. Uma matriz algébrica pode ser construída como um conjunto de listas nas linhas, como:

Maxima: Linhas de comando

```
M: matrix([a,b],[c,d]);
```

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

28. Vamos construir uma matriz numérica:

Maxima: Linhas de comando

```
N: matrix([1],[2]);
```

$$\begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

29. Realizamos o produto ponto (`.`) da matriz M pela matriz N. O número de linhas da matriz `M` deve ser igual ao número de colunas da matriz `N`.

Maxima: Linhas de comando

```
M.N;
```

$$\begin{bmatrix} 2b+a \\ 2d+c \end{bmatrix}$$

30. Selecionando a coluna 1 da matriz M

Maxima: Linhas de comando

```
col(M,1);
```

$$\begin{bmatrix} a \\ c \end{bmatrix}$$

31. Selecionando a coluna 2 da matriz M

Maxima: Linhas de comando

```
col(M,2);
```

$$\begin{bmatrix} b \\ d \end{bmatrix}$$

32. Selecionando a linha 2 da matriz M

Maxima: Linhas de comando

```
row(M,2);
>>> [ c d ]
```

33. Obtendo a linha 1 da matriz M:

Maxima: Linhas de comando
```
M[1];
>>> [a, b]
```

34. Obtendo apenas a linha 2 da matriz M:

Maxima: Linhas de comando
```
M[2]
>>> [c, d]
```

35. Obtendo um elemento da matriz situado na linha 1 e na coluna 2 de M:

Maxima: Linhas de comando
```
M[1,2]
>>> b
```

36. Obtendo a matriz M^2, usando o operador `.` para o produto usual de matrizes.

Maxima: Linhas de comando
```
M.M;
```

$$\begin{bmatrix} bc+a^2 & bd+ab \\ cd+ac & d^2+bc \end{bmatrix}$$

37. A mesma resposta para M^2, pode ser obtida com

Maxima: Linhas de comando
```
M^^2;
```

38. Verificação que `M.M = M^^2`:

Maxima: Linhas de comando
```
M.M - M^^2;
```

$$\begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

39. Agora verificamos a igualdade com a função `compare`. Após executar a linha de comando, vemos o sinal de igualdade =, indicando que as matrizes comparadas são iguais.

Maxima: Linhas de comando
```
compare(M.M,M^^2);
>>> =
```

40. A inversa de uma matriz pode ser obtida por:

Maxima: Linhas de comando
```
M^^(-1);
```

$$\begin{bmatrix} \frac{d}{ad-bc} & -\frac{b}{ad-bc} \\ -\frac{c}{ad-bc} & \frac{a}{ad-bc} \end{bmatrix}$$

41. Multiplicando `M` por `M^^(-1)` e fazendo a simplificando racional:

Maxima: Linhas de comando
```
M.M^^(-1); ratsimp(%);
```

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

42. Podemos obter os Autovalores e Autovetores de uma matriz pelo processo usualmente tratado em uma aula de Álgebra Linear. Seja a matriz

Maxima: Linhas de comando
```
M: matrix([1,2,3],[4,5,6],[7,8,9])
```

43. Podemos construir uma matriz da forma

Maxima: Linhas de comando
```
M - z * ident(3);
```

$$\begin{bmatrix} 1-z & 2 & 3 \\ 4 & 5-z & 6 \\ 7 & 8 & 9-z \end{bmatrix}$$

44. O polinômio característico da matriz M na variável z, é dado por:

Maxima: Linhas de comando
```
determinant(%);
>>>  ((5-z)(9-z)-48)(1-z)-2(4(9-z)-42)+3(32-7(5-z))
```

45. Obtemos a equação característica com o comando

Maxima: Linhas de comando
```
ratsimp(%);
```

$$-z^3+15z^2+18z$$

46. Resolvemos esta última equação na variável `z`, digitando:

Maxima: Linhas de comando
```
solve(%, [z]);
```

47. Obtemos assim os autovalores de M:

$$[z=-\frac{3\sqrt{33}-15}{2},\ z=\frac{3\sqrt{33}+15}{2},\ z=0]$$

48. O Maxima possui funções para obter autovalores, autovetores, equações e polinômios característicos. A função `charpoly(M,x)` gera o polinômio característico da matriz M na variável x, definido por `p(x)=determinant(M-diagmatrix(length(M),x))`.

49. Vamos considerar uma matriz M como:

Maxima: Linhas de comando
```
M: matrix([3,1],[2,4]);
```

$$\begin{bmatrix} 3 & 1 \\ 2 & 4 \end{bmatrix}$$

50. O comando abaixo desenvolve a expressão polinomial e simplifica a mesma.

Maxima: Linhas de comando
```
expand(charpoly(M,x));
```

$$x^2 - 7x + 10$$

51. Usamos agora o modo de programação para resolver a última equação:

Maxima: Linhas de comando
```
(programmode:true, solve(%));
>>> [x=5, x=2]
```

52. Criamos então um vetor genérico `Y` que servirá para gerar os autovetores:

Maxima: Linhas de comando
```
Y: matrix([a],[b]);
```

$$\begin{bmatrix} a \\ b \end{bmatrix}$$

53. Avaliamos a equação característica para obter as duas equações do sistema linear

Maxima: Linhas de comando
```
ev(M.Y-x*Y, %th(2)[1]);
```

$$\begin{bmatrix} b-2a \\ 2a-b \end{bmatrix}$$

54. Tomamos agora a expressão localizada na primeira linha e na primeira coluna da última da última expressão obtida anteriormente:

Maxima: Linhas de comando
```
%[1,1] = 0;
>>>  b-2a = 0
```

55. Na seqüência, usaremos a restrição $a^2 + b^2 = 1$ para garantir que os autovetores $v = (a,b)$ (vetores próprios de M) sejam unitários, isto é, $|v| = |(a,b)| = 1$.

Maxima: Linhas de comando
```
a^2 + b^2 = 1;
solve([%th(2),%], [a,b]);
```

$$[\,[a = -\frac{1}{\sqrt{5}},\ b = -\frac{2}{\sqrt{5}}]\,,[\,a = \frac{1}{\sqrt{5}},\ b = \frac{2}{\sqrt{5}}\,]\,]$$

56. O modo fácil de obter os autovalores da matriz A é usar a função `eigenvalues`:

Maxima: Linhas de comando
```
eigenvalues(A);
>>> [ [-2,8], [1,1] ]
```

Lista1: `[-2,8]`

(a) O primeiro valor da Lista1 é `-2` que é primeiro autovalor

(b) O segundo valor da Lista1 é `8` que é o segundo autovalor.

Lista2: `[1,1]`

(a) O primeiro valor da Lista2 é `1` = multiplicidade do autovalor `-2`
(b) O segundo valor da Lista2 é `1` = multiplicidade do autovalor `8`

57. A expressão `P: copymatrix(M)` gera uma cópia da matriz M com o nome P. Este é o único modo para gerar uma copia separada da matriz M elemento a elemento.
58. Ao realizar atribuições de uma matriz para outra, como em `m2:m1`, não copiamos `m1`. Atribuições como `m2[i,j]:x` ou `setelmx(x,i,j,m2)` modificam `m1[i,j]`.
59. Obtemos os autovetores da matriz A, com a função `eigenvectors`:

Maxima: Linhas de comando
```
eigenvectors(A);
>>> [ [ [-2,8], [1,1] ], [1,-1], [1,7/3] ]
```

Lista1: `[-2,8]`

(a) O primeiro valor da Lista1: `-2` é primeiro autovalor
(b) O segundo valor da Lista1: `8` é o segundo autovalor

Lista2: `[1,1]`

(a) O primeiro valor da Lista2: `1` é a multiplicidade do autovalor `-2`
(b) O segundo valor da Lista2: `1` é a multiplicidade do autovalor `8`

Lista3: `[1,-1]` é o autovetor associado ao autovalor `-2`.
Lista4: `[1,7/3]` é o autovetor associado ao autovalor `8`.

60. Vamos inserir as matrizes A e B de mesma ordem, digitando:

Maxima: Linhas de comando
```
A: matrix([1,3],[ 7,5]);
```

$$\begin{bmatrix} 1 & 3 \\ 7 & 5 \end{bmatrix}$$

Maxima: Linhas de comando
```
B: matrix([8,2],[-4,3]);
```

$$\begin{bmatrix} 8 & 2 \\ -4 & 3 \end{bmatrix}$$

61. Podemos somar as matrizes A e B, digitando:

Maxima: Linhas de comando
```
A+B;
```

$$\begin{bmatrix} 9 & 5 \\ 3 & 8 \end{bmatrix}$$

62. Podemos multiplicar as matrizes A e B, digitando:

Maxima: Linhas de comando
```
A.B;
```

$$\begin{bmatrix} -4 & 11 \\ 36 & 29 \end{bmatrix}$$

63. Vamos construir a matriz M, digitando:

Maxima: Linhas de comando
```
M: matrix([1,2,3],[4,5,6],[7,8,1]);
```

64. O posto da matriz M pode ser obtido com a função `rank`:

Maxima: Linhas de comando
```
rank(M)
>>> 3
```

A matriz M tem posto máximo em R3, isto é, o conjunto das linhas de M é LI, ou seja, os vetores-linhas da matriz M formam uma base para o espaço vetorial R3.

65. Vamos mudar agora a matriz M, para algo como:

Maxima: Linhas de comando
```
M: matrix([1,2,3],[4,5,6],[7,8,9]);
```

66. O posto desta matriz M pode ser obtido com:

Maxima: Linhas de comando
```
rank(M)
>>> 2
```

Esta matriz não tem posto máximo em R3, isto é, o conjunto formado pelas linhas da matriz M é LD (linearmente dependente), isto é, os vetores-linhas `não` formam uma base para o espaço vetorial R3.

67. Obtemos o subespaço Imagem da transformação linear associada à matriz M, com:

Maxima: Linhas de comando
```
M: matrix([1,2,3],[4,5,6],[7,8,9]);
columnspace(M)
```

Obtemos o subespaço gerado pelos vetores-coluna do conjunto `{v1,v2,v3}` que é uma base para o espaço coluna de M.

$$\text{span}\left(\begin{bmatrix}1\\4\\7\end{bmatrix},\begin{bmatrix}2\\5\\8\end{bmatrix}\right)$$

O subespaço Imagem da transformação T é gerado pelos vetores-coluna `(1,4,7)` e `(2,5,8)`.

68. Para obter o subespaço gerado pelo núcleo de uma matriz M, digite:

Maxima: Linhas de comando
```
M: matrix([1,2,3],[4,5,6],[7,8,9]);
nullspace(M);
```

$$\text{span}\left(\begin{bmatrix}1\\-2\\1\end{bmatrix}\right)$$

O Núcleo da transformação linear associada a M é a reta gerada pelo vetor `(1,-2,1)`.

69. Exercício: Para cada item, digite o código Maxima que gera o resultado indicado:

(a) Definir uma matriz M com 3 linhas e 3 colunas, para obter

$$M = \begin{bmatrix} 5 & 4 & 5 \\ 0 & 5 & 6 \\ 0 & 0 & 3 \end{bmatrix}$$

(b) Realizar a soma da matriz M com a própria matriz M, para obter

$$\begin{bmatrix} 10 & 8 & 10 \\ 0 & 10 & 12 \\ 0 & 0 & 6 \end{bmatrix}$$

(c) Multiplicar o escalar 2 pela matriz M para obter:

$$\begin{bmatrix} 10 & 8 & 10 \\ 0 & 10 & 12 \\ 0 & 0 & 6 \end{bmatrix}$$

(d) Obter uma matriz com as entradas elevadas ao quadrado para as correspondentes entradas de M.

$$\begin{bmatrix} 25 & 16 & 25 \\ 0 & 25 & 36 \\ 0 & 0 & 9 \end{bmatrix}$$

(e) Multiplique a matriz M pela matriz M, para obter:

$$\begin{bmatrix} 25 & 16 & 25 \\ 0 & 25 & 36 \\ 0 & 0 & 9 \end{bmatrix}$$

(f) Mostrar que a matriz M elevada ao quadrado é dada por:

$$\begin{bmatrix} 25 & 40 & 64 \\ 0 & 25 & 48 \\ 0 & 0 & 9 \end{bmatrix}$$

(g) Mostrar que a transposta de M é dada por:

$$\begin{bmatrix} 5 & 0 & 0 \\ 4 & 5 & 0 \\ 5 & 6 & 3 \end{bmatrix}$$

(h) Mostrar que o traço da matriz M é igual a 13.

(i) Mostrar que a adjunta da matriz M é dada por:

$$\begin{bmatrix} 15 & -12 & -1 \\ 0 & 15 & -30 \\ 0 & 0 & 25 \end{bmatrix}$$

(j) Mostrar que o determinante da matriz M é igual a 75.

(k) Dividir a adjunta de M pelo determinante de M, para obter:

$$\begin{bmatrix} \frac{1}{5} & -\frac{4}{25} & -\frac{1}{75} \\ 0 & \frac{1}{5} & -\frac{2}{5} \\ 0 & 0 & \frac{1}{3} \end{bmatrix}$$

(l) Mostrar que a inversa de M é:

$$\begin{bmatrix} \frac{1}{5} & -\frac{4}{25} & -\frac{1}{75} \\ 0 & \frac{1}{5} & -\frac{2}{5} \\ 0 & 0 & \frac{1}{3} \end{bmatrix}$$

(m) Mostrar que os autovalores da matriz M, bem como as suas respectivas multiplicidades, são dados por `[[3,5], [1,2]]`.

(n) Mostrar que os autovalores, suas multiplicidades e os respectivos autovetores da matriz M, podem ser obtidos por uma linha de comando, para gerar:

$$[[[3,5],[1,2]],[1,-\frac{6}{7},\frac{2}{7}],[1,0,0]]$$

70. Para obter o subespaço gerado pelo Núcleo da matriz, digite:

Maxima: Linhas de comando
```
M: matrix( [1,2,3], [4,5,6], [7,8,1] )
nullspace(M)
>>> span()
```

significando que o núcleo da matriz M é o subespaço nulo de R3.

71. Para escalonar a matriz

Maxima: Linhas de comando
```
M: matrix([1,2,3],[4,5,6],[7,8,9]);
```

através do processo de Gauss, basta digitar

Maxima: Linhas de comando
```
echelon(M);
```

$$\begin{bmatrix} \mathbf{1} & 2 & 3 \\ 0 & \mathbf{1} & 2 \\ 0 & 0 & 0 \end{bmatrix}$$

72. A função `triangularize` escalona a matriz pelo método de Gauss, mas o *pivot* (primeiro elemento não nulo de uma linha não nula) em cada linha não é normalizado.

Maxima: Linhas de comando
```
triangularize(M);
```

$$\begin{bmatrix} \mathbf{1} & 2 & 3 \\ 0 & \mathbf{-3} & -6 \\ 0 & 0 & \mathbf{0} \end{bmatrix}$$

4.4 Inversa de uma matriz através do escalonamento

1. Vá ao menu `Maxima` e pressione `Reiniciar Maxima`, para limpar todas as funções, variáveis e expressões que estavam na memória do computador.
2. Obteremos a inversa de uma matriz quadrada A de ordem 2 pelo método do escalonamento. Cada operação elementar realizada sobre a matriz A é obtida pelo produto de uma matriz elementar E por A. Assim, obteremos o escalonamento passo a passo.
3. Criamos uma matriz ampliada `M=(A,Id)`, onde A é uma matriz e `Id` é a identidade de mesma ordem posta à direita de A. Por operações elementares sobre `M` obteremos a matriz `M1=(Id,A1)` em que `Id` é a identidade à esquerda de `A1` e `A1` é a inversa de A.
4. Vamos construir uma matriz A quadrada de ordem 2, digitando:

Maxima: Linhas de comando
```
A: matrix([1,2],[3,4]);
```

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

5. Com a função `addcol` criamos uma matriz ampliada M, pondo a matriz A e à sua direita a matriz identidade de ordem 2:

Maxima: Linhas de comando
```
M: addcol(A,ident(2))
```

$$\begin{bmatrix} 1 & 2 & \mathbf{1} & \mathbf{0} \\ 3 & 4 & \mathbf{0} & \mathbf{1} \end{bmatrix}$$

6. Para anular o número 3 da posição M(2,1) da matriz M, repetimos a Linha1 e na Linha2 escrevemos a soma da Linha2 com o produto de -3 pela Linha1. Assim, criamos a matriz `E1` que corresponde a esta operação elementar por linhas.

Maxima: Linhas de comando
```
E1: matrix([1,0],[0,1]-3*[1,0]);
```

$$\begin{bmatrix} 1 & 0 \\ -3 & 1 \end{bmatrix}$$

7. Realizamos agora o produto das matrizes `E1` e M:

Maxima: Linhas de comando
```
E1.M
```

$$\begin{bmatrix} 1 & 2 & \mathbf{1} & \mathbf{0} \\ 0 & -2 & \mathbf{-3} & \mathbf{1} \end{bmatrix}$$

8. Na Linha2, colocamos a soma da Linha1 com a Linha2 do produto `E1.M`. Para tal, construiremos uma operação elementar `E2` que realiza esta operação. Definimos

Maxima: Linhas de comando
```
E2: matrix([1,0]+[0,1],[0,1]);
```

$$\begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$$

9. Realizamos o produto das matrizes `E2`, `E1` e `M`:

Maxima: Linhas de comando
```
E2.E1.M
```

$$\begin{pmatrix} 1 & 0 & \mathbf{-2} & \mathbf{1} \\ 0 & -2 & \mathbf{-3} & \mathbf{1} \end{pmatrix}$$

10. Mantemos intacta a Linha1 e dividimos a Linha2 por `-2` e colocamos o resultado na Linha2. A operação elementar `E3` que realiza esta operação é definida pela matriz:

Maxima: Linhas de comando
```
E3: matrix([1,0],[0,1]/(-2));
```

$$\begin{bmatrix} 1 & 0 \\ 0 & -1/2 \end{bmatrix}$$

11. Obtemos então a inversa da matriz `M`, à direita da identidade, com:

Maxima: Linhas de comando
```
E3.E2.E1.M
```

$$\begin{bmatrix} 1 & 0 & \mathbf{-2} & \mathbf{1} \\ 0 & 1 & \mathbf{3/2} & \mathbf{-1/2} \end{bmatrix}$$

12. O produto das três operações elementares fornece a inversa da matriz A, isto é:

Maxima: Linhas de comando
```
E3.E2.E1
```

$$\begin{bmatrix} -2 & 1 \\ 3/2 & -1/2 \end{bmatrix}$$

4.5 Inversa de matriz por escalonamento (forma alternativa)

1. Vá ao menu `Maxima` e pressione `Reiniciar Maxima`, para limpar todas as funções, variáveis e expressões que estavam na memória do computador.
2. Obteremos agora a inversa de uma matriz quadrada A de ordem 2 pelo método do escalonamento, porém trataremos as operações de uma forma mais rápida que a anterior.
3. Seja a matriz quadrada A de ordem 2, definida por:

Maxima: Linhas de comando
```
A: matrix([4,3],[2,1]);
```

$$\begin{bmatrix} 4 & 3 \\ 2 & 1 \end{bmatrix}$$

4. Criamos a matriz ampliada `M=(A,I)`:

Maxima: Linhas de comando
```
M: addcol(A,ident(2));
```

$$\begin{bmatrix} 4 & 3 & 1 & 0 \\ 2 & 1 & 0 & 1 \end{bmatrix}$$

5. A primeira linha de M é `M[1]` e a segunda linha de M é `M[2]`.
6. Vamos realizar a operação elementar:

Maxima: Linhas de comando
```
M[2]: M[1]-2*M[2];
M;
```

Não mostraremos a resposta da primeira linha, mas a matriz M é dada por:

$$\begin{bmatrix} 4 & 3 & 1 & 0 \\ 0 & 1 & 1 & -2 \end{bmatrix}$$

7. Vamos realizar a operação elementar:

Maxima: Linhas de comando
```
M[1]: M[1]-3*M[2];
M;
```

A primeira resposta não será mostrada, mas a matriz M é dada por:

$$\begin{bmatrix} 4 & 0 & -2 & 6 \\ 0 & 1 & 1 & -2 \end{bmatrix}$$

8. Vamos realizar a operação elementar:

Maxima: Linhas de comando
```
M[1]: (1/4)*M[1];
M;
```

Não mostraremos a primeira resposta mas a nova matriz M tem a forma:

$$\begin{bmatrix} 1 & 0 & -\frac{1}{2} & \frac{3}{2} \\ 0 & 1 & 1 & -2 \end{bmatrix}$$

4.6 Outra forma para obter a inversa por escalonamento

1. Vá ao menu `Maxima` e pressione `Reiniciar Maxima`, para limpar todas as funções, variáveis e expressões que estavam na memória do computador.
2. O Maxima possui a função `rowop(M,i,j,k)` que realiza uma operação elementar, transformando a linha `M[i]` da matriz `M` na linha `M[i]-k*M[j]`, onde `i` e `j` são indicadores de linhas de `M`. Por exemplo, `rowop(M,2,1,1/2)` significa que a nova linha `M[2]` passará a ter a forma `M[2]-(1/2)*M[1]`.
3. Seja a matriz quadrada A de ordem 2, definida por:

Maxima: Linhas de comando
```
A: matrix([4,3],[2,1]);
```

$$\begin{bmatrix} 4 & 3 \\ 2 & 1 \end{bmatrix}$$

4. Criamos a matriz ampliada `M=(A,I)`:

Maxima: Linhas de comando
```
M: addcol(A,ident(2));
```

$$\begin{bmatrix} 4 & 3 & 1 & 0 \\ 2 & 1 & 0 & 1 \end{bmatrix}$$

5. A primeira linha de M é `M[1]` e a segunda linha de M é `M[2]`.
6. Seja a operação elementar que cria uma linha `M[2]` como sendo `M[2]-(1/2)*M[1]`

Maxima: Linhas de comando
```
M: rowop(M,2,1,1/2);
```

$$\begin{bmatrix} 4 & 3 & 1 & 0 \\ 0 & -\frac{1}{2} & -\frac{1}{2} & 1 \end{bmatrix}$$

7. Seja a operação elementar que altera a linha `M[1]` com:

Maxima: Linhas de comando
```
M: rowop(M,1,2,-6);
```

$$\begin{bmatrix} 4 & 0 & -2 & 6 \\ 0 & -\frac{1}{2} & -\frac{1}{2} & 1 \end{bmatrix}$$

8. O Maxima pode escrever uma matriz diagonal, com os elementos `p` e `q` na diagonal principal e o código `diag(p,q)`, mas precisamos primeiro carregar o pacote `diag` com:

Maxima: Linhas de comando
```
load("diag");
```

9. Vamos construir uma matriz diagonal com `1/4` e `-2` na diagonal principal.

Maxima: Linhas de comando
```
L: [1/4, -2];
diag(L)
```

$$\begin{bmatrix} \frac{1}{4} & 0 \\ 0 & -2 \end{bmatrix}$$

10. Agora multiplicamos a última matriz (diagonal) pela matriz M:

Maxima: Linhas de comando
```
M: % . M;
```

$$\begin{bmatrix} 1 & 0 & -\dfrac{1}{2} & \dfrac{3}{2} \\ 0 & 1 & 1 & -2 \end{bmatrix}$$

11. O Maxima dispõe da função `submatrix` que extrai uma submatriz de uma matriz dada. Por exemplo, `submatrix(M,1,2)` retira da matriz M as colunas 1 e 2, assim

Maxima: Linhas de comando
```
submatrix(M,1,2);
```

produz a inversa da matriz A, isto é,

$$\begin{bmatrix} -\dfrac{1}{2} & \dfrac{3}{2} \\ 1 & -2 \end{bmatrix}$$

12. Exemplos de uso da função `submatrix`. Construiremos uma matriz `Z` na forma abaixo, com as teclas de atalho ctrl-I:

Maxima: Linhas de comando
```
Z: matrix([11,12,13,14],[21,22,23,24],[31,32,33,34],[41,42,43,44]);
```

$$\begin{bmatrix} 11 & 12 & 13 & 14 \\ 21 & 22 & 23 & 24 \\ 31 & 32 & 33 & 34 \\ 41 & 42 & 43 & 44 \end{bmatrix}$$

(a) Obtendo uma submatriz sem a linha 1 da matriz Z:

Maxima: Linhas de comando

```
submatrix(1,Z);
```

$$\begin{bmatrix} 21 & 22 & 23 & 24 \\ 31 & 32 & 33 & 34 \\ 41 & 42 & 43 & 44 \end{bmatrix}$$

(b) Obtendo uma submatriz sem a linha 1 e sem a coluna 2 de Z:

Maxima: Linhas de comando

```
submatrix(1,Z,2);
```

$$\begin{bmatrix} 21 & 23 & 24 \\ 31 & 33 & 34 \\ 41 & 43 & 44 \end{bmatrix}$$

(c) Obtendo uma submatriz sem a coluna 2 da matriz Z:

Maxima: Linhas de comando

```
submatrix(Z,2);
```

$$\begin{bmatrix} 11 & 13 & 14 \\ 21 & 23 & 24 \\ 31 & 33 & 34 \\ 41 & 43 & 44 \end{bmatrix}$$

(d) Obtendo uma submatriz sem as linhas 1 e 2 da matriz Z:

Maxima: Linhas de comando

```
submatrix(1,2,Z);
```

$$\begin{bmatrix} 31 & 32 & 33 & 34 \\ 41 & 42 & 43 & 44 \end{bmatrix}$$

(e) Obtendo uma submatriz sem as colunas 1 e 2 da matriz Z:

Maxima: Linhas de comando

```
submatrix(Z,1,2);
```

$$\begin{bmatrix} 13 & 14 \\ 23 & 24 \\ 33 & 34 \\ 43 & 44 \end{bmatrix}$$

5 Cálculo diferencial e Integral

5.1 Limites, Derivadas e diferenciais

1. Vá ao menu `Maxima` e pressione `Reiniciar Maxima`, para limpar todas as funções, variáveis e expressões que estavam na memória do computador.
2. No Maxima podemos calcular limites de muitas funções racionais e transcendentes.

3. Para funções racionais, é muitas vezes importante saber o valor no infinito:

Maxima: Linhas de comando
```
limit((x^2-1)/(x^2+1),x,inf);
>>> 1
```

4. O valor de uma expressão indeterminada da forma `0/0` pode ser calculado:

Maxima: Linhas de comando
```
limit((x^2+x-6)/(x^4+x^3-19*x^2+11*x+30),x,2);
>>> -5/21
```

5. Limites laterais nem sempre coincidem em um determinado ponto.
6. Calculamos o limite lateral à direita, para valores de `%pi/2 + epsilon` para valores pequenos de incremento de epsilon. Maxima obtém o valor limite `minus infinity`.

Maxima: Linhas de comando
```
limit(tan(x),x,%pi/2,plus);
```

$$-\infty$$

7. Quando tomamos o limite lateral à esquerda no mesmo ponto

Maxima: Linhas de comando
```
limit(tan(x),x,%pi/2,minus);
```

$$\infty$$

8. Podemos calcular este limite sem informar ao Maxima o lado onde queremos calcular:

Maxima: Linhas de comando
```
limit(tan(x),x,%pi/2);
>>> und
```

9. Essa resposta significa und de undefined. Esta é uma dica que podemos calcular os limites laterais de ambos os lados. Aqui está um exemplo diferente:

Maxima: Linhas de comando
```
limit(tan(3*x)/tan(x),x,%pi/2);
>>> 1/3
```

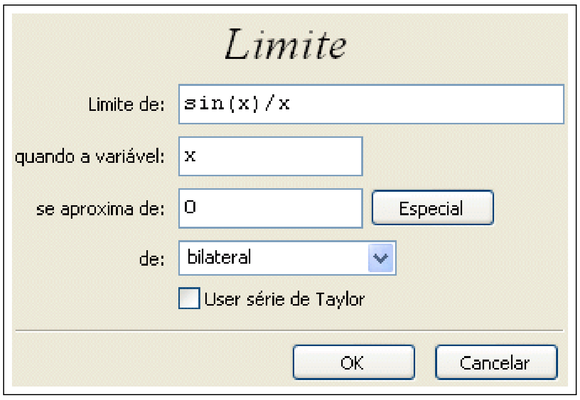
10. Calculando um limite fundamental

Maxima: Linhas de comando
```
limit(sin(x)/x,x,0)
>>> 1
```

11. Calculando outro limite fundamental

Maxima: Linhas de comando
```
limit((%e^x-1)/x,x,0)
>>> 1
```

12. Obtemos o limite anterior com o menu `Cálculo`, pressionando `Encontrar limite...` e preenchendo a janela de diálogo que aparece, de acordo com o que está na figura:



13. Calculando outro limite fundamental

Maxima: Linhas de comando
```
limit((sin(x)-x)/x^2,x,0)
>>> 0
```

14. Calculando o limite de `f(x)=x*log(x)` quando `x` tende a 0 pela `direita`.

Maxima: Linhas de comando
```
limit(x*log(x),x,0,plus)
>>> 0
```

15. Calculando o limite de `f(x)=x*log(x)` quando `x` tende a 0 pela `esquerda`.

Maxima: Linhas de comando
```
limit(x*log(x),x,0,minus)
>>> 0
```

16. Calculando o limite de `f(x)=(1+x)^(1/x)` quando `x` tende a 0.

Maxima: Linhas de comando
```
limit((x+1)^(1/x),x,0)
>>> e
```

17. Calculando o limite de `f(x)=e^x/x` quando `x` tende a `infinito`.

Maxima: Linhas de comando
```
limit(%e^x/x,x,inf)
```

$$\infty$$

18. No cálculo do limite, o Maxima usa símbolos e palavras especiais como inf infinito positivo, minf infinito negativo, und (undefined) indefinido, ind indefinido mas associado e infinity infinito complexo.

19. O limite de `f(x)=sin(1/x)` quando `x` tende a `0` resulta `indefinido`.

Maxima: Linhas de comando
```
limit(sin(1/x),x,0)
>>> ind
```

20. O limite de `f(x)=log(x)/x` quando `x` tende a `zero` resulta `indefinido`.

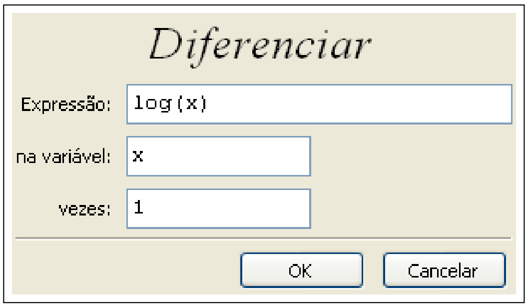
Maxima: Linhas de comando
```
assume(x>0);
limit(log(x)/x,x,0)
>>> und
```

21. Vá ao menu `Maxima` e pressione `Reiniciar Maxima`, para limpar todas as funções, variáveis e expressões que estavam na memória do computador.

22. Calculando a derivada de `f(x)=log(x)` na variável `x`.

Maxima: Linhas de comando
```
diff(log(x),x);
>>> 1/x
```

23. Para calcular a mesma derivada do item anterior, você pode acessar o menu `Cálculo`, depois `Diferenciar` e preencher de acordo com a figura:



24. Calculando a derivada de ordem 3 da função `f(x)=log(x)` na variável `x`.

Maxima: Linhas de comando
```
diff(log(x),x,3);
```

$$\frac{2}{x^3}$$

25. Calculando a diferencial de `f(x)=log(x)`

Maxima: Linhas de comando
```
diff(log(x));
```

$$\frac{\mathrm{del}(x)}{x}$$

26. Calculando a derivada de `f(x,y)=exp(2x+3y)` com relação à variável `x`.

Maxima: Linhas de comando
```
diff(exp(2*x+3*y),x);
```

$$2e^{3y+2x}$$

27. Calculando a derivada de `f(x,y)=exp(2x+3y)` com relação à variável `y`.

Maxima: Linhas de comando
```
diff(exp(2*x+3*y),y);
```

$$3e^{3y+2x}$$

28. Calculando a diferencial (total) de `f(x,y)=exp(2x+3y)`. No resultado abaixo, del(x) é a diferencial dx e del(y) é a diferencial dy.

Maxima: Linhas de comando
```
diff(exp(2*x+3*y));
```

$$3e^{3y+2x}\mathrm{del}(y)+2e^{3y+2x}\mathrm{del}(x)$$

29. Calculando a diferencial total de `f(x,y,z)=xyz`.

Maxima: Linhas de comando
```
diff(x*y*z);
>>> xy del(z) + xz del(y) + yz del(x)
```

30. Sabemos que a função `f(x)=|x|` (valor absoluto) não possui derivada no ponto `x=0`, pois as derivadas laterais `L1` e `L2` são diferentes. Neste caso, vamos usar a função `print` para a resposta.

Maxima: Linhas de comando
```
f(x):= abs(x)$
qf0: (f(x)-f(0))/(x-0)$
L1: limit(qf0, x, 0, plus)$
L2: limit(qf0, x, 0, minus)$
print("L1=",L1, " é diferente de L2=",L2);
>>> L1=1  é diferente de L2=-1
```

31. Mas, podemos calcular a derivada de `f(x)=abs(x)` para os valores de `x` não nulos.

Maxima: Linhas de comando
```
diff(abs(x),x);
```

$$\frac{x}{|x|}$$

32. Vamos definir o símbolo `f` para a função complicada `f=f(x)`:

Maxima: Linhas de comando
```
f: x^2 * %e^(k*x) * sin(w*x);
```

$$x^2\,e^{kx}\sin(wx)$$

33. Calculamos a derivada de `f` com respeito à variável `x`:

Maxima: Linhas de comando
```
diff(f,x);
```

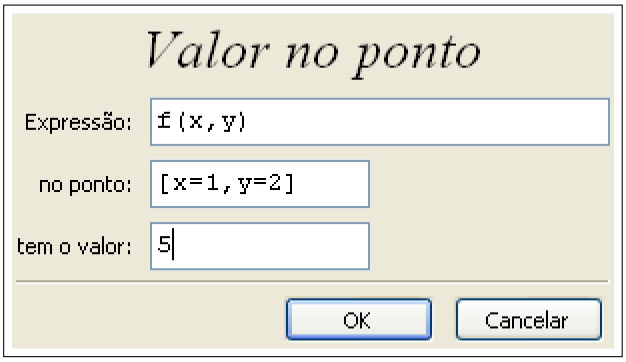
$$kx^2\,e^{kx}\sin(wx)+2xe^{kx}\sin(wx)+wx^2\,e^{kx}\cos(wx)$$

34. Vá ao menu `Maxima` e pressione `Reiniciar Maxima`, para limpar todas as funções, variáveis e expressões que estavam na memória do computador.

5.2 Atribuindo valor a um ponto do domínio de uma função

1. A função `atvalue` atribui valor a uma certa expressão `Expr` no ponto indicado. Em geral, os valores extremos de uma função são obtidos por este mecanismo.
2. A função `atvalue(f(x),[x=a],c)` atribui o valor `c` a função `f(x)` no ponto `x=a`.

   Maxima: Linhas de comando
   ```
   atvalue(f(x),[x=0],12);
   >>> 12
   ```
3. Podemos atribuir o valor 5 à função `f=f(x,y)` no ponto `(x,y)=(1,2)`, acessando o menu `Equações`, depois `Valor no ponto ...` e preencher de acordo com a figura



4. Vejamos qual é o valor de `f(0)`:

   Maxima: Linhas de comando
   ```
   f(0)
   >>> 12
   ```
5. A função `atvalue(f(x,y),[x=a,y=b],c)` atribui o valor `c` para a expressão `f(x,y)` no ponto em que `x=a` e `y=b`.

   Maxima: Linhas de comando
   ```
   atvalue(f(x,y),[x=1,y=2],5);
   >>> 5
   ```
6. Vejamos qual é o valor de `f(1,2)`:

   Maxima: Linhas de comando
   ```
   f(1,2)
   >>> 5
   ```
7. Vá ao menu `Maxima` e pressione `Reiniciar Maxima`, para limpar todas as funções, variáveis e expressões que estavam na memória do computador.

5.3 Integrais

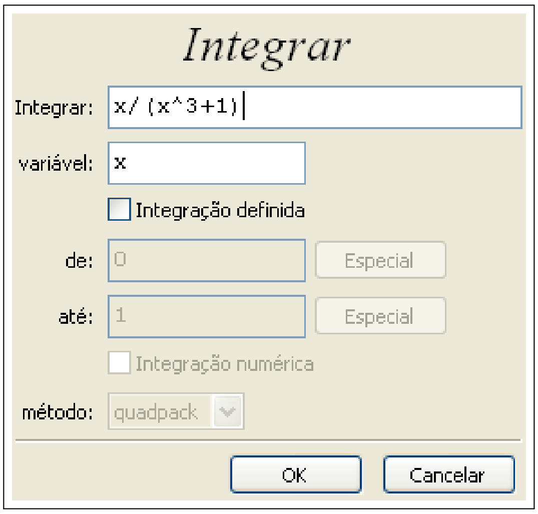
1. O Maxima pode calcular integrais indefinidas de muitas funções. O processo de integração simbólica descreve o algoritmo para calcular integrais indefinidas.

2. Calcularemos a integral de uma função de variável real:

Maxima: Linhas de comando
```
integrate(x/(x^3+1),x);
```

$$\frac{\log(x^2-x+1)}{6}+\frac{\operatorname{atan}(\frac{2x-1}{\sqrt{3}})}{\sqrt{3}}-\frac{\log(x+1)}{3}$$

3. Outro modo de integrar a função do item anterior é acessar o menu `Cálculo`, pressionar `Integrar` e preencher a janela de diálogo como está na figura:



4. Para verificar esse resultado, vamos calcular a derivada da integral obtida:

Maxima: Linhas de comando
```
diff(%,x);
```

$$\frac{2}{3(\frac{(2x-1)^2}{3}+1)}+\frac{2x-1}{6(x^2-x+1)}-\frac{1}{3(x+1)}$$

5. A última expressão é uma soma de três expressões racionais. Cada termo da integral contribui com uma expressão racional.

6. Usamos a função `ratsimp` para simplificar esta somas de expressões racionais e trazer todas as três expressões a um denominador comum:

Maxima: Linhas de comando
```
ratsimp(%);
```

$$\frac{x}{x^3+1}$$

7. Vamos definir o símbolo g para a função da variável `x`:

Maxima: Linhas de comando
```
g: %e^(k*x) * sin(w*x);
```

$$e^{kx}\sin(wx)$$

8. Agora nós obtemos a integral indefinida de `f` com respeito à variável `x`:

Maxima: Linhas de comando
```
integrate(g,x);
```

$$\frac{e^{kx}\left(k\sin(wx)-w\cos(wx)\right)}{w^2+k^2}$$

9. Podemos obter uma integral indefinida como:

Maxima: Linhas de comando
```
integrate (1/x^2, x);
>>> -1/x
```

10. Uma leve mudança na sintaxe fornece a integral definida no intervalo `[a,b]`:

Maxima: Linhas de comando
```
integrate (1/x^2, x, a, b);
```

11. Para obter a resposta, são feitas três perguntas. Sugiro que responda com a palavra positive a todas as três perguntas, para obter a resposta

$$\frac{1}{a}-\frac{1}{b}$$

12. Para não ter que responder às perguntas, você poderia usar a função `assume` e assumir que `a>0`, `b>0` e que `a<b`. A forma de usar tais hipóteses é:

Maxima: Linhas de comando
```
assume(a>0); assume(b>0); assume(a<b);
integrate (1/x^2, x, a, b);
```

$$\frac{1}{a}-\frac{1}{b}$$

13. Calcularemos uma integral imprópria convergente. `inf` significa `infinito`:

Maxima: Linhas de comando
```
integrate(1/x^2, x, 1, inf);
>>> 1
```

14. Uma integral imprópria divergente.

Maxima: Linhas de comando
```
integrate(1/x, x, 1, inf);
Integral is divergent
 -- an error.  Quitting.  To debug this try debugmode(true);
```

15. Vá ao menu `Maxima` e pressione `Reiniciar Maxima`, para limpar todas as funções, variáveis e expressões que estavam na memória do computador.

5.4 Decomposição em frações parciais

1. Todas as funções do Maxima mostram algum resultado, mas não mostram os passos dos cálculos executados para obter o resultado. Vamos decompor a expressão racional $p(x) = \frac{1}{x^2(x^2+1)}$ em frações parciais, usando a função `partfrac`:

Maxima: Linhas de comando
```
partfrac(1/(x^2*(x^2+1)),x);
```

$$\frac{1}{x^2} - \frac{1}{x^2+1}$$

5.5 Integração por substituição

1. Algumas vezes, o Maxima fornece uma resposta denominada forma substantiva de uma integral, informando que não foi capaz de realizar a integral. Por exemplo:

Maxima: Linhas de comando
```
integrate(1/((x-3)^4+1), x);
```

$$\int \frac{1}{(x-3)^4+1} dx$$

2. A função `changevar(Expr,f(x,y),y,x)` troca a variável `x` da expressão `Expr` sujeita à relação `f(x,y)=0` e a nova expressão ficará em função da variável `y`.
3. Neste caso, tomaremos `x=3+y` como a substituição que ajuda a resolver a integral:

Maxima: Linhas de comando
```
changevar(%, x=3+y,y,x);
```

$$\int \frac{1}{y^4+1} dy$$

4. Ao realizar a integral com

Maxima: Linhas de comando
```
ev(%,integrate);
```

$$\frac{\log(y^2+\sqrt{2}y+1)}{4\sqrt{2}} - \frac{\log(y^2-\sqrt{2}y+1)}{4\sqrt{2}} + \frac{\operatorname{atan}(\frac{2y+\sqrt{2}}{\sqrt{2}})}{2\sqrt{2}} + \frac{\operatorname{atan}(\frac{2y-\sqrt{2}}{\sqrt{2}})}{2\sqrt{2}}$$

5. Realizando a substituição reversa obtemos a integral desejada.

Maxima: Linhas de comando
```
sfx: %, y=x-3;
```

$$\frac{\log((x-3)^2+\sqrt{2}(x-3)+1)}{4\sqrt{2}} - \frac{\log((x-3)^2-\sqrt{2}(x-3)+1)}{4\sqrt{2}} + \frac{\operatorname{atan}(\frac{2(x-3)+\sqrt{2}}{\sqrt{2}})}{2\sqrt{2}} + \frac{\operatorname{atan}(\frac{2(x-3)-\sqrt{2}}{\sqrt{2}})}{2\sqrt{2}}$$

6. O integrando é uma função positiva para todo x. Para tais funções podemos avaliar a integral definida para obter seu valor, digitando

Maxima: Linhas de comando
```
'integrate(1/((x-3)^4+1), x, 0, 1);
```

podemos escrever:

Maxima: Linhas de comando
```
ratsimp(subst(1,x,sfx) - subst(0,x,sfx));
```

7. Tal expressão usa a substituição para a integral indefinida nos limites da integral definida.
8. O valor numérico é:

Maxima: Linhas de comando
```
float(%);
>>> 0.028312178860342
```

9. A integral numérica de Romberg confirma o resultado:

Maxima: Linhas de comando
```
romberg(1/((x-3)^4+1), x, 0, 1);
>>> 0.028312178867484
```

10. Vejamos como o Maxima realiza uma integral da forma $\int f(x)f'(x)\,dx = \int u.du$:

Maxima: Linhas de comando
```
integrate((x^3+x^2)*(3*x^2+2*x), x);
```

$$\frac{(x^3+x^2)^2}{2}$$

11. Caso em que a potência de f é um número inteiro positivo:

Maxima: Linhas de comando
```
integrate( (x^3+x^2)^10 * (3*x^2+2*x), x);
```

$$\frac{(x^3+x^2)^{11}}{11}$$

12. Caso em que a potência f é um número racional:

Maxima: Linhas de comando
```
integrate((x^4+1)^(2/3) * (4*x^3), x);
```

$$\frac{3(x^4+1)^{\frac{5}{3}}}{5}$$

13. Vá ao menu Maxima e pressione `Reiniciar Maxima`, para limpar todas as funções, variáveis e expressões que estavam na memória do computador.

5.6 Expansão de Taylor

1. Obteremos a expansão em série de Taylor de algumas funções elementares na variável x em torno do ponto x=0 contendo termos com potências até a ordem 10:

Maxima: Linhas de comando
```
taylor(sin(x), x, 0, 10);
```

$$x-\frac{x^3}{6}+\frac{x^5}{120}-\frac{x^7}{5040}+\frac{x^9}{362880}+\ldots$$

Maxima: Linhas de comando
```
taylor(cos(x), x, 0, 10);
```

$$1-\frac{x^2}{2}+\frac{x^4}{24}-\frac{x^6}{720}+\frac{x^8}{40320}-\frac{x^{10}}{3628800}+\ldots$$

Maxima: Linhas de comando
```
taylor(exp(x), x, 0, 10);
```

$$1+x+\frac{x^2}{2}+\frac{x^3}{6}+\frac{x^4}{24}+\frac{x^5}{120}+\frac{x^6}{720}+\frac{x^7}{5040}+\frac{x^8}{40320}+\frac{x^9}{362880}+\frac{x^{10}}{3628800}+\ldots$$

Maxima: Linhas de comando
```
taylor(log(1+x), x, 0, 10);
```

$$x-\frac{x^2}{2}+\frac{x^3}{3}-\frac{x^4}{4}+\frac{x^5}{5}-\frac{x^6}{6}+\frac{x^7}{7}-\frac{x^8}{8}+\frac{x^9}{9}-\frac{x^{10}}{10}+\ldots$$

Maxima: Linhas de comando
```
taylor(tan(x), x, 0, 10);
```

$$x+\frac{x^3}{3}+\frac{2x^5}{15}+\frac{17x^7}{315}+\frac{62x^9}{2835}+\ldots$$

Maxima: Linhas de comando
```
taylor(log(1+x)-log(1-x), x, 0, 10);
```

$$2x+\frac{2x^3}{3}+\frac{2x^5}{5}+\frac{2x^7}{7}+\frac{2x^9}{9}+\ldots$$

2. Agora, vamos complicar com uma função f e definir outra função h como a divisão de f por uma função seno hiperbólico elevado ao quadrado:

Maxima: Linhas de comando
```
f: x^2 * exp(k*x) * sin(w*x)$
h: f / sinh(k*x)^2;
```

$$\frac{e^{kx}x^2\sin(wx)}{\sinh(kx)^2}$$

3. Obteremos a expansão Et em série de Taylor da função h na variável x em torno do ponto x=0 contendo termos com potências até a ordem 2:

Maxima: Linhas de comando
```
Et: taylor(h,x,0,2);
```

$$\frac{wx}{k^2}+\frac{wx^2}{k}+\ldots$$

4. Como o desenvolvimento tem problemas com a avaliação de algumas expressões em `x=0`, o Maxima pode realizar limites de funções reais expandidas em série de Taylor. A função `tlimit` de `h` quando `x` tende a `0` é calculada como:

Maxima: Linhas de comando
```
tlimit(h,x,0);
>>> 0
```

5.7 Comprimento de arco de uma curva

1. Vá ao menu `Maxima` e pressione `Reiniciar Maxima`, para limpar todas as funções, variáveis e expressões que estavam na memória do computador.
2. Para calcular o comprimento do arco de uma curva diferenciável z=z(x), aplicaremos os conceitos de derivada e integral definida, com o limitante inferior a menor que o limitante superior `b`.

Maxima: Linhas de comando
```
assume(a<b);
```

3. Usaremos uma função arbitrária `z=z(x)`, que vai depender da curva a ser definida no momento oportuno.

Maxima: Linhas de comando
```
z: lambda([x]);
```

4. A fórmula para calcular o comprimento da curva z=z(x) de x=a até x=b é:

Maxima: Linhas de comando
```
L(z,a,b):= block(z1:diff(z(x),x),
           L:integrate(sqrt(1+z1^2),x,a,b), L);
```

$$L(z,a,b) := \mathrm{block}(\mathrm{z1} : \mathrm{diff}(\mathrm{z}(\mathrm{x}),\mathrm{x}),\mathrm{L} : \mathrm{integrate}(\sqrt{1+\mathrm{z1}^2},\mathrm{x},\mathrm{a},\mathrm{b}),\mathrm{L})$$

5. Para $f(x)=1+x+x^2$, calcularemos o comprimento do arco de `a=0` até `b=1`:

Maxima: Linhas de comando
```
f: lambda([x], 1+x+x^2);
a:0; b:1;
L(f,a,b);
```

$$\frac{\mathrm{asinh}(3)+3\sqrt{10}}{4}-\frac{\mathrm{asinh}(1)+\sqrt{2}}{4}$$

6. Obtemos o mesmo resultado na forma decimal, pressionando Ctrl-F ou com:

Maxima: Linhas de comando
```
float(%), numer;
>>> 2.252423072586142
```

7. Vamos considerar agora uma outra função mais complicada, para calcular o comprimento do arco da curva de a=0.2 até b=1:

Maxima: Linhas de comando
```
g(x):= 1/x;
a:0.2; b:1;
L(g,a,b), numer;
```

8. Após algum tempo, vemos que o Maxima não calculou, pois mostra apenas a resposta:

$$\int_{0.2}^{1} \left(\frac{1}{x^4}+1\right)^{0.5} dx$$

9. Utilizaremos o pacote romberg que contém a função romberg (mesmo nome) para realizar a integral numérica. Inicialmente devemos carregar o pacote com:

Maxima: Linhas de comando
```
load("romberg");
```

10. Cálculo da integral sobre o intervalo (0.2,1], com a mudança do valor de a:

Maxima: Linhas de comando
```
a:0.2$
b:1 $
Lr: romberg(sqrt(1+(diff(g(x),x))^2),x,a,b);
>>> 4.151453857194474
```

11. Cálculo da integral sobre o intervalo (0.1,1], com a mudança do valor de a:

Maxima: Linhas de comando
```
a:0.1$
b:1.0$
Lr: romberg(sqrt(1+(diff(g(x),x))^2),x,a,b);
>>> 9.152620479728629
```

12. Cálculo da integral sobre o intervalo (0.05,1], com a mudança do valor de a:

Maxima: Linhas de comando
```
a:0.05$
b:1.0$
Lr: romberg(sqrt(1+(diff(g(x),x))^2),x,a,b);
>>> 19.15276677402481
```

13. Cálculo da integral sobre o intervalo (0.03,1], com a mudança do valor de a:

Maxima: Linhas de comando
```
a:0.03$
b:1.0$
Lr: romberg(sqrt(1+(diff(g(x),x))^2),x,a,b);
>>> 32.48611607134156
```

14. Cálculo da integral sobre o intervalo `(0.02,1]`, com a mudança do valor de `a`:

Maxima: Linhas de comando
```
a:0.02$
b:1.0$
Lr: romberg(sqrt(1+(diff(g(x),x))^2),x,a,b);
>>> 49.15279700927701
```

15. Cálculo da integral sobre o intervalo `(0.01,1]`, com a mudança do valor de `a`:

Maxima: Linhas de comando
```
a:0.01$
b:1.0$
Lr: romberg(sqrt(1+(diff(g(x),x))^2),x,a,b);
>>> 99.15281131713053
```

16. Cálculo da integral sobre o intervalo `(0.005,1]`, com a mudança do valor de `a`:

Maxima: Linhas de comando
```
a:0.005$
b:1.0$
Lr: romberg(sqrt(1+(diff(g(x),x))^2),x,a,b);
>>> 199.152837826789
```

17. Conclusão: O que acontece quando `a` tende a `0`?

5.8 Resolvendo equações diferenciais ordinárias

1. Vá ao menu `Maxima` e pressione `Reiniciar Maxima`, para limpar todas as funções, variáveis e expressões que estavam na memória do computador.
2. O Maxima pode mostrar Equações Diferenciais Ordinárias (EDO), com as derivadas em uma forma `substantiva` ou seja, que não é avaliada. Observe com cuidado a aspa simples antes da expressão::

Maxima: Linhas de comando
```
'diff(y,x);
```

$$\frac{d}{dx}y$$

3. O operador `'` na expressão `'diff(y,x);` significa `não realizar a avaliação`.
4. Se você digitar `diff(y,x)` sem usar este operador `'` o Maxima irá gerar a resposta `0`.
5. Primeiro estudaremos Equações Diferenciais Ordinárias de 1a. ordem. Muita atenção com o apóstrofe `'` antes de `diff`, pois devemos evitar o cálculo imediato da derivada.

Maxima: Linhas de comando
```
eq1: 'diff(y,x) = -y;
```

6. Nós obtemos a equação na forma matemática convencional que ela é escrita:

$$\frac{d}{dx}y = -y$$

7. Nós usaremos a função `ode2` para obter a solução através de:

Maxima: Linhas de comando
```
ode2(eq1, y, x);
```

8. A solução contém a constante de integração `%c`:

$$y = \%c\, e^{-x}$$

9. Para obter uma solução particular, que satisfaz um determinado valor inicial, digamos `y(1)=8`, nós utilizamos a função `ic1` (initial condition) para definir a condição inicial. Posteriormente, associaremos a solução com a variável `sol`:

Maxima: Linhas de comando
```
sol: ic1(%, x=1, y=8);
```

10. Tal condição inicial escolhe a solução que passa pelo ponto `(1,8)` e obtemos:

$$y = 8\, e^{1-x}$$

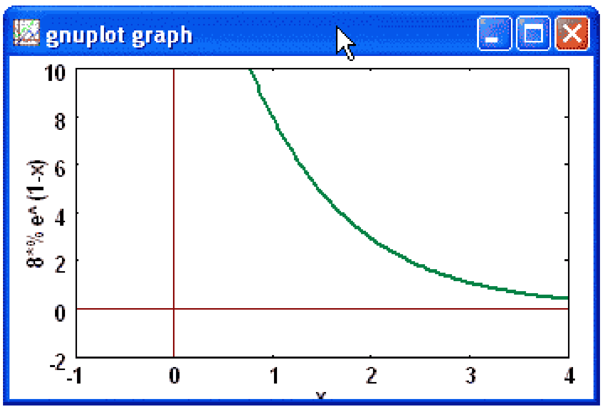
11. Podemos obter o lado direito de `sol`, com

Maxima: Linhas de comando
```
ld: rhs(sol);
```

$$8\, e^{1-x}$$

12. Podemos desenhar o gráfico da solução `y=y(x)` com:

Maxima: Linhas de comando
```
plot2d(ld, [x,-1,4], [y,-2,10]);
```



13. Agora, estudaremos Equações Diferenciais Ordinárias de segunda ordem. Da mesma forma que antes, vamos construir a EDO:

Maxima: Linhas de comando
```
eq2: 'diff(y,x,2) + y = 0;
```

$$\frac{d^2}{dx^2}\, y + y = 0$$

14. Obteremos a solução geral desta equação com a função `ode2`:

Maxima: Linhas de comando
```
sol2: ode2(eq2, y, x);
>>> y=%k1 sin(x) + %k2 cos(x)
```

15. Esta solução possui dois parâmetros. Há dois modos para calcular os valores destes parâmetros.
    (PVI) Problema de Valor Inicial: Nós obtemos os parâmetros de uma particular solução e de sua derivada no ponto dado.
    (PVF) Problema de Valor de Fronteira: Nós necessitamos saber os valores da solução pesquisada em dois pontos diferentes.
16. Para indicar uma condição inicial, usamos a função `ic2`, que indica um ponto da solução e a tangente à solução (derivada) naquele ponto.
17. Vamos considerar a condição inicial tal que `y(0)=2` e `y'(0)=1`

Maxima: Linhas de comando
```
cond2: ic2(sol2, x=0, y=2, 'diff(y,x)=1);
>>> y = sin(x) + 2 cos(x)
```

18. Podemos obter o lado direito `ld` de `cond2`, com

Maxima: Linhas de comando
```
ladodireito: rhs(cond2);
>>> sin(x) + 2 cos(x)
```

19. Verificamos que o valor inicial `y(0)=2` é satisfeito por esta solução:

Maxima: Linhas de comando
```
ev(ladodireito,x=0);
>>> 2
```

20. A derivada do `ld` da solução é dada por:

Maxima: Linhas de comando
```
diff(ladodireito,x);
>>> cos(x) - 2 sin(x)
```

21. Verificamos que a derivada da solução satisfaz a condição `'diff(y,x)=1`:

Maxima: Linhas de comando
```
ev(%,x=0);
>>> 1
```

22. Observação: Não consegui combinar estas duas expressões em uma simples expressão como `ev(diff(rhs(ps),x),x=0);` pois o resultado não combina com o desejado.
23. Para um Problema de Valor de Fronteira (PVF), nós usamos a função `bc2`:

Maxima: Linhas de comando
```
bc2(sol2, x=0, y=2, x=2, y=-1);
```

$$y = 2\cos(x) - \frac{(2\cos(2)+1)\sin(x)}{\sin(2)}$$

24. Exercício: Verifique que as duas condições de fronteira: `y(0)=2, y'(2)=-1` são satisfeitas pela solução do item anterior.

25. Vamos estudar outra equação diferencial ordinária:

Maxima: Linhas de comando
```
ed2: 'diff(y,x,2) + 'diff(y,x) + y;
```

$$\frac{d^2}{dx^2}y+\frac{d}{dx}y+y$$

26. A solução desta equação é obtida com a função `ode2`:

Maxima: Linhas de comando
```
s2: ode2(ed2,y,x);
```

$$y=e^{-\frac{x}{2}}(\%k1\ \sin(\frac{\sqrt{3}x}{2})+\%k2\ \cos(\frac{\sqrt{3}x}{2}))$$

27. Se temos um Problema de Valor de Fronteira tal que `y(0)=2` e `y(2)=-1`, obtemos:

Maxima: Linhas de comando
```
bc2(s2, x=0, y=2, x=2, y=-1);
```

$$y=e^{-\frac{x}{2}}\left(2\cos\left(\frac{\sqrt{3}\,x}{2}\right)-\frac{\left(2\cos(\sqrt{3})+e\right)\sin(\frac{\sqrt{3}\,x}{2})}{\sin(\sqrt{3})}\right)$$

28. Exercícios para casa:

    (a) Se $g(x)=\exp(ax)\sin(x)\cos(x)$ é uma função real, mostrar que a integral indefinida de `g=g(x)` é dada por
    $$\frac{e^{ax}(a\sin(2x)-2\cos(2x))}{2(a^2+4)}$$

    (b) Verifique o resultado, calculando a derivada do último resultado com relação à variável `x`, para obter:
    $$\frac{ae^{ax}(a\sin(2x)-2\cos(2x))}{2(a^2+4)}+\frac{e^{ax}(4\sin(2x)+2a\cos(2x))}{2(a^2+4)}$$

    (c) Utilize a simplificação com `ratsimp` para tornar a resposta mais simples na forma
    $$\frac{e^{ax}\sin(2x)}{2}$$

    (d) Para ficar livre do seno do arco duplo, aplique `trigexpand` para obter:
    $$e^{ax}\cos(x)\sin(x)$$

    que é a mesma função `g=g(x)` apresentada inicialmente.

29. Exercícios de integrais com extensões de radical simples:

    (a) Calcular a integral da função indicada por:

    Maxima: Linhas de comando
    ```
    (-4*x^3*(x^4+1)^(2/3)-16*x^3*(x^4+1)^(1/3))/(3*x^8+6*x^4+3)
    ```

    (b) Realizar simplificações com `ratsimp` para obter:

    Maxima: Linhas de comando
    ```
    ratsimp(%);
    ```

    (c) Calcular a integral abaixo em termos de exponencial e logaritmos, com

    Maxima: Linhas de comando
    ```
    integrate(((6*x^5+7*x^4-36*x^3+18*x-21)*(x^4+1)^(2/3)+
    (2*x^6-20*x^4-40*x^3+18*x^2+12)*(x^4+1)^(1/3))/(3*x^8+6*x^4+3),x);
    ```

    (d) Usar `radcan` para obter uma representação canônica dos radicais:

    Maxima: Linhas de comando
    ```
    radcan(%);
    ```

    (e) Calcular a integral indicada por

    Maxima: Linhas de comando
    ```
    integrate(7*tan(x)^8 +7*tan(x)^6,x);
    ```

    (f) Realizar simplificações trigonométricas

    Maxima: Linhas de comando
    ```
    trigsimp(%);
    ```

    para obter $\tan(x)^7 = \tan^7(x)$.

5.9 Integrais de Linha

1. Vá ao menu `Maxima` e pressione `Reiniciar Maxima`, para limpar todas as funções, variáveis e expressões que estavam na memória do computador.
2. Uma integral de linha em duas dimensões é: $\oint_C P(x,y)dx + Q(x,y)dy$ onde x e y são as coordenadas ao longo da curva de integração C. A curva de integração pode ser especificada através das coordenadas de seus pontos em termos de um parâmetro t: $x = f(t), y = g(t)$.
3. Para três dimensões, temos a forma $\oint_C P(x,y,z)dx + Q(x,y,z)dy + R(x,y,z)dz$ onde x, y e z são as coordenadas ao longo da curva de integração C. A curva de integração pode ser especificada através das coordenadas de seus pontos em termos de um parâmetro t: $x = f(t), y = g(t), z = h(t)$.
4. O cálculo de uma integral de linha. Consideremos o integrando:

   Maxima: Linhas de comando
   ```
   integrando: x^2*y*diff(x) + y*z*diff(y) + z*x*diff(z);
   ```

   $$xz\,\mathrm{del}(z) + yz\,\mathrm{del}(y) + x^2y\,\mathrm{del}(x)$$

5. Seja a curva de integração:

   Maxima: Linhas de comando
   ```
   curva: [x=cos(t), y=sin(t), z=sin(t)];
   >>> [x = cos(t) , y = sin(t), z = sin(t)]
   ```

6. As coordenadas desse caminho devem ser substituídas no integrando:

Maxima: Linhas de comando
```
sublis(%, integrando);
```

$$\sin(t)^2 \mathrm{del}(\sin(t)) + \cos(t)\sin(t)\mathrm{del}(\sin(t)) + \cos(t)^2 \sin(t)\mathrm{del}(\cos(t))$$

7. Agora vamos avaliar as derivadas:

Maxima: Linhas de comando
```
ev(%, diff);
```

$$-\cos(t)^2 \sin(t)^2 \mathrm{del}(t) + \cos(t)\sin(t)^2 \mathrm{del}(t) + \cos(t)^2 \sin(t)\mathrm{del}(t)$$

8. Para integrar essa expressão, removemos o `del(t)` com:

Maxima: Linhas de comando
```
%, del(t)=1;
```

$$-\cos(t)^2 \sin(t)^2 + \cos(t)\sin(t)^2 + \cos(t)^2 \sin(t)$$

9. Agora podemos integrar a função de uma variável real:

Maxima: Linhas de comando
```
integrate(%, t, 0, 2*%pi);
```

$$-\frac{\pi}{4}$$

10. O cálculo de integrais de linha exige quatro passos: (P1) Substituir as equações da curva no integrando; (P2) Avaliar as derivadas; (P3) Remover a diferencial do parâmetro; (P4) Realizar o cálculo de uma integral definida.

11. Para simplificar esse cálculo e reduzir o risco de erros de digitação, é bom colocar esses quatro passos em uma definição funcional (usando $\boxed{\text{Ctrl-I}}$):

Maxima: Linhas de comando
```
IntLin(fn,curva,par,p0,p1):= block(
   [substFn, x, xx],
   substFn: sublis(curva, fn),
   x: ev(substFn,diff),
   xx: subst(1,diff(par),x),
   integrate(xx,par,p0,p1));
```

12. Aqui está o exemplo anterior, agora com esta nova função:

Maxima: Linhas de comando
```
IntLin(x^2*y*diff(x) + y*z*diff(y) + z*x*diff(z),
       [x=cos(t),y=sin(t),z=sin(t)], t, 0, 2*%pi);
```

$$-\frac{\pi}{4}$$

13. Um exemplo do livro *Teoria e Problemas de Transformadas de Laplace* de M. Spiegel:

Maxima: Linhas de comando
```
IntLin((x^2-y)*diff(x)+(y^2 + x)*diff(y),[x=t,y=t^2+1],t,0,1);
>>> 2
```

6 Transformações Trigonométricas

1. Vá ao menu Maxima e pressione `Reiniciar Maxima`, para limpar todas as funções, variáveis e expressões que estavam na memória do computador.
2. Identidades trigonométricas podem ser manipuladas algebricamente com a função `trigexpand` que usa as fórmulas de somas de ângulos para criar o argumento dentro de cada função trigonométrica tão simples quanto possível:

   Maxima: Linhas de comando
   ```
   sin(u+v) * cos(u)^3;
   ```
   $$\cos(u)^3 \sin(v+u)$$
3. A expansão trigonométrica é criada com:

   Maxima: Linhas de comando
   ```
   trigexpand(%);
   ```
   $$\cos(u)^3\ (\cos(u)\sin(v)+\sin(u)\cos(v))$$
4. A função `trigreduce` converte uma expressão em uma combinação linear de `senos` e `cossenos`, em que cada termo contém um simples `sin` ou um simples `cos`, mas sem potências de funções trigonométricas:

   Maxima: Linhas de comando
   ```
   trigreduce(%);
   ```
   $$\frac{\sin(v+4u)+\sin(v-2u)}{8}+\frac{3\sin(v+2u)+3\sin(v)}{8}$$
5. Para reescrever um produto de senos e cossenos como uma soma finita de Fourier, podemos usar a função `trigreduce`:

   Maxima: Linhas de comando
   ```
   trigreduce(sin(x)^3*cos(x));
   ```
   $$\frac{2\sin(2x)-\sin(4x)}{8}$$
6. Para obter uma forma que não tem funções trigonométricas de arcos múltiplos, com:

   Maxima: Linhas de comando
   ```
   trigexpand(%);
   ```
   $$\frac{4\cos(x)\sin(x)^3-4\cos(x)^3\sin(x)+4\cos(x)\sin(x)}{8}$$
7. Parece que ficou complicado, mas podemos agora usar a função `trigsimp` para simplificar expressões que usam a identidade $\cos^2(x)+\sin^2(x)=1$.
8. Usando `trigsimp` obtemos exatamente a expressão inicial.

   Maxima: Linhas de comando
   ```
   trigsimp(%);
   ```
   $$\cos(x)\sin(x)^3$$

   mas, isto nem sempre é possível de ser obtido.

9. Substituir uma variável é um modo forte para realizar simplificações com `trigexpand`.
10. Vimos que: (a) `trigreduce` reescreve uma expressão trigonométrica como uma soma de termos, cada um deles tendo somente `sin` ou `cos`, (b) `trigexpand` simplifica os argumentos de função trigonométrica, (c) as fórmulas de soma de arcos são usadas para aquela simplificação, (d) `trigsimp` executa várias simplificações de expressões trigonométricas, (e) a função `trigsimp` usa a identidade $\cos^2(x) + \sin^2(x) = 1$ para a simplificação, (f) `trigrat` simplifica uma expressão racional em funções trigonométricas.

7 Maxima no Cálculo de funções de Várias variáveis

7.1 Derivadas parciais e aplicações

1. Vá ao menu `Maxima` e pressione `Reiniciar Maxima`, para limpar todas as funções, variáveis e expressões que estavam na memória do computador.
2. A linha de comando genérica utilizada para calcular derivadas parciais é

   Maxima: Linhas de comando
   ```
   diff(f(x1,x2,...,xm), x1, n1, x2, n2, ..., xm, nm);
   ```

3. Calcular as derivadas parciais de $w = f(x,y,z) = x^3y^2z - 5xy + 3yz - 2xz + 10$ com relação às variáveis x, y e z.

   Maxima: Linhas de comando
   ```
   w : x^3*y^2*z - 5*x*y +3*y*z - 2*x*z + 10;
   diff(w,x,1,y,1,z,1);
   ```

4. Se $f(x,y) = \dfrac{xy}{x^2+y^2}$ se $(x,y) \neq (0,0)$ e $f(0,0) = 0$, determinar $f_x(0,0)$ e $f_y(0,0)$

   (a) definindo a função $f = f(x,y)$:

   Maxima: Linhas de comando
   ```
   f(x,y):=(if abs(x)+abs(y)>0 then x*y/(x^2+y^2) else 0);
   ```

   (b) Calculando a derivada de f em relação a x no ponto (0,0):

   Maxima: Linhas de comando
   ```
   assume(h>0); fx0: limit((f(0+h,0)-f(0,0))/h,h,0);
   ```

   (c) Calculando a derivada de f em relação a y no ponto (0,0):

   Maxima: Linhas de comando
   ```
   assume(k>0); fk0: limit((f(0,0+k)-f(0,0))/k,k,0);
   ```

   (d) Mostraremos agora que $f = f(x,y)$ é descontínua em $(0,0)$.

   (e) Fazendo $y = mx$. Calculando o limite de f quando $(x,y) \to (0,0)$:

   Maxima: Linhas de comando
   ```
   limit(m*x^2/(x^2+(m*x)^2),x,0);
   ```

   (f) O limite depende do modo de tender a zero, logo, não existe. Por isso, $f(x,y)$ não é continua em $(0,0)$.

(g) Vamos calcular $g_x(0,0)$, definindo $g = g(x,y)$:

Maxima: Linhas de comando
```
g(x,y):=(if abs(x)+abs(y)>0 then x*y*(x^2-y^2)/(x^2+y^2) else 0);
```

(h) Derivando g em relação a x no ponto $(0,0)$

Maxima: Linhas de comando
```
gx00: limit((g(h,0)-g(0,0))/h,h,0);
```

(i) Calcularemos $g_y(0,0)$, a derivada de g com relação a y em $(0,0)$:

Maxima: Linhas de comando
```
gy00: limit((g(0,k)-g(0,0))/k,k,0);
```

(j) Se $(x,y) \neq (0,0)$, derivando g em relação a x:

Maxima: Linhas de comando
```
diff(x*y*(x^2-y^2)/(x^2+y^2),x);
```

(k) Simplificando o resultado obtido:

Maxima: Linhas de comando
```
ratsimp(%);
```

(l) Definindo g_x, derivada de g em relação a x

Maxima: Linhas de comando
```
gx(x,y):=(if abs(x)+abs(y)>0 then
        (-(y^5-4*x^2*y^3-x^4*y)/(y^4+2*x^2*y^2+x^4)) else 0);
```

(m) Derivando g em relação a y:

Maxima: Linhas de comando
```
diff(x*y*(x^2-y^2)/(x^2+y^2),y);
```

(n) Simplificando o resultado obtido:

Maxima: Linhas de comando
```
ratsimp(%);
```

(o) Definindo g_y, derivada de g em relação a y

Maxima: Linhas de comando
```
gy(x,y):=(if abs(x)+abs(y)>0 then
        (-(x*y^4+4*x^3*y^2-x^5)/(y^4+2*x^2*y^2+x^4)) else 0);
```

(p) Calcular a segunda derivada de g com relação a x, isto é, $g_{xx}(0,0)$.

Maxima: Linhas de comando
```
gxx00: limit((gx(h,0)-gx(0,0))/h,h,0);
```

(q) Calcular a segunda derivada de g com relação a y, isto é, $g_{yy}(0,0)$.

Maxima: Linhas de comando
```
gyy00: limit((gy(0,k)-gy(0,0))/k,k,0);
```

(r) Calcular $g_{xy}(0,0)$

Maxima: Linhas de comando
```
gxy00: limit((gx(0,k)-gx(0,0))/k,k,0);
```

(s) Calcular $g_{yx}(0,0)$

Maxima: Linhas de comando
```
gyx00: limit((gy(h,0)-gy(0,0))/h,h,0);
```

(t) Observar que $g_{xy}(0,0)$ é diferente de $g_{yx}(0,0)$.

7.2 Máximos e Mínimos com multiplicadores de Lagrange

1. Vá ao menu `Maxima` e pressione `Reiniciar Maxima`, para limpar todas as funções, variáveis e expressões que estavam na memória do computador.
2. Determinar a menor distância da origem $(0,0)$ à hiperbole $x^2+8xy+7y^2=225$.
   (a) Devemos obter o mínimo do `quadrado` de $d(x,y)=\sqrt{x^2+y^2}$ (distância da origem ao ponto (x,y)), sujeito ao vínculo $v(x,y)=x^2+8xy+7y^2-225=0$.
   (b) Pelo método dos multiplicadores de Lagrange, devemos construir uma função $f=f(x,y)$ que é a soma do vínculo $v=v(x,y)$ com um múltiplo m do quadrado da distância $d^2=d^2(x,y)$, isto é,
   Maxima: Linhas de comando
   ```
   f: x^2 + 8*x*y +7*y^2 - 225 + m*(x^2 + y^2);
   ```
   (c) Agora, calculamos a derivada da função $f=f(x,y)$ em relação a x:
   Maxima: Linhas de comando
   ```
   fx: diff(f,x);
   ```
   (d) Calculamos a derivada da função $f=f(x,y)$ em relação a y:
   Maxima: Linhas de comando
   ```
   fy: diff(f,y);
   ```
   (e) Das derivadas anteriores, como (x,y) é não nulo, devemos obter a matriz
   Maxima: Linhas de comando
   ```
   M: matrix([2*m+2,8],[8,2*m+14]);
   ```
   (f) Tomando o determinante de M igual a zero, segue que:
   Maxima: Linhas de comando
   ```
   D: determinant(M)=0;
   ```
   (g) Agora, resolvemos esta última equação para obter o valor de m:
   Maxima: Linhas de comando
   ```
   solve([D], [m]);
   ```
   (h) Caso 1: Se $m=1$ em f_x ou f_y, segue que $x=-2y$.
   (i) Construamos a função $v=v(x,y)$.
   Maxima: Linhas de comando
   ```
   v: x^2 + 8*x*y +7*y^2 - 225 = 0;
   ```
   (j) Se $x=-2y$, substituímos o valor de x em $x^2+8xy+7y^2=225$ com:
   Maxima: Linhas de comando
   ```
   a: subst (-2*y, x, v);
   ```
   (k) Resolvendo a equação a e obtendo os valores de y:
   Maxima: Linhas de comando
   ```
   solve([a], [y]);
   ```
   (l) Observamos que neste caso, não há solução real.
   (m) Caso 2: Se $m=-9$ em f_x ou f_y, segue que $y=2x$.
   (n) Substituímos o valor de y em v:
   Maxima: Linhas de comando
   ```
   b: subst (2*x, y, v);
   ```

(o) Resolvemos a equação b para obter o valor de x

Maxima: Linhas de comando
```
solve([b], [x]);
```

(p) Logo $x^2 = 5$, $y^2 = 4 * x^2 = 20$ e $x^2 + y^2 = 25$, e a menor distância é $\sqrt{25} = 5$.

3. Determinar os valores máximo e mínimo de $f(x,y,z) = x^2 + y^2 + z^2$ sujeito aos vínculos $x^2/4 + y^2/5 + z^2/25 = 1$ e $z = x + y$.

(a) Devemos calcular os extremos da função $f(x,y,z) = x^2 + y^2 + z^2$.

(b) Definindo a função:

Maxima: Linhas de comando
```
f(x,y,z):= x^2 + y^2 + z^2;
```

(c) Definindo os vínculos

Maxima: Linhas de comando
```
u(x,y,z):= x^2/4 + y^2/5 + z^2/25 -1 ;
v(x,y,z):= x + y - z ;
```

(d) Usaremos 2 multiplicadores de Lagrange m, n e consideraremos a função:

Maxima: Linhas de comando
```
g: f(x,y,z) + m*u(x,y,z) + n*v(x,y,z);
```

(e) Calcularemos as derivadas parciais de g, e anularemos as mesmas para obter as equações:

Maxima: Linhas de comando
```
eq1: diff(g,x)=0;
eq2: diff(g,y)=0;
eq3: diff(g,z)=0;
```

(f) Obtendo os valores de x, y e z:

Maxima: Linhas de comando
```
c: solve([eq1,eq2,eq3],[x,y,z]);
```

(g) Da segunda condição de vínculo $x + y - z = 0$, obtemos com a divisão por n, suposto diferente de zero (o que está justificado, pois se $n = 0$, ter-se-ia $x = 0$, $y = 0$, $z = 0$ que não satisfaz a primeira condição)

Maxima: Linhas de comando
```
d: (v(-(2*n)/(m+4),-(5*n)/(2*m+10),(25*n)/(2*m+50)))/n=0;
```

(h) Simplificando a expressão

Maxima: Linhas de comando
```
e: ratsimp(d);
```

(i) Obtendo o valor de m:

Maxima: Linhas de comando
```
solve([e], [m]);
```

(j) Caso 1: Se $m = -10$, substituímos o valor de m em c

Maxima: Linhas de comando
```
subst(-10,m,c);
```

(k) Tomando tal valor e aplicando em u, obtemos:

Maxima: Linhas de comando
```
h: u(n/3,n/2,(5*n)/6);
```

(l) Resolvendo a equação h para obter n:

Maxima: Linhas de comando
```
solve([h], [n]);
```

(m) Definindo x, y e z em função de n:

Maxima: Linhas de comando
```
[x: n/3, y:n/2, z:(5*n)/6];
```

(n) Deste modo obtemos os pontos:

Maxima: Linhas de comando
```
P1: [ subst(-6*sqrt(5))/sqrt(19),n,x),
      subst(-6*sqrt(5))/sqrt(19),n,y),
      subst(-6*sqrt(5))/sqrt(19),n,z) ];
P2: [ subst(6*sqrt(5))/sqrt(19),n,x),
      subst(6*sqrt(5))/sqrt(19),n,y),
      subst(6*sqrt(5))/sqrt(19),n,z) ];
```

(o) Caso 2: Se $m = -75/17$, substituímos o valor de m em c:

Maxima: Linhas de comando
```
subst(-75/17,m,c);
```

(p) Levando em u:

Maxima: Linhas de comando
```
i: u((34*n)/7,-(17*n)/4,(17*n)/28);
```

(q) Calculando o valor de n em i:

Maxima: Linhas de comando
```
solve([i], [n]);
```

(r) Definindo x, y e z em função de n:

Maxima: Linhas de comando
```
[x: (34*n)/7, y: -(17*n)/4, z: (17*n)/28];
```

(s) Assim obtemos os pontos críticos:

Maxima: Linhas de comando
```
P1: [ subst(-70*sqrt(2)/(17*sqrt(323)),n,x),
      subst(-70*sqrt(2)/(17*sqrt(323)),n,y),
      subst(-70*sqrt(2)/(17*sqrt(323)),n,z) ];
P2: [ subst(70*sqrt(2)/(17*sqrt(323)),n,x),
      subst(70*sqrt(2)/(17*sqrt(323)),n,y),
      subst(70*sqrt(2)/(17*sqrt(323)),n,z) ];
```

(t) O valor de $f(x,y,z) = x^2 + y^2 + z^2$ correspondente a estes pontos é:

Maxima: Linhas de comando
```
f( (20*sqrt(2))/sqrt(323),
  -(35*sqrt(2))/(2*sqrt(323)),
   (5*sqrt(2))/(2*sqrt(323)) );
```

(u) Então, o máximo procurado é 10 e o mínimo é 75/17.

7.3 Máximos e Mínimos com o Teorema de Sylvester

Estudaremos Máximos e Mínimos de funções de várias variáveis utilizando o Teorema de Sylvester. Apresentamos dois exemplos: um com duas variáveis e outro com três variáveis.

1. Vá ao menu `Maxima` e pressione `Reiniciar Maxima`, para limpar todas as funções, variáveis e expressões que estavam na memória do computador.
2. Inserindo a função $g(x,y) = 2x^4 + y^4 - 2x^2y^2 - 2y$:

Maxima: Linhas de comando
```
g(x,y):= 2*x^4+y^4-2*x^2*y^2-2*y;
```

3. Definindo `eq1` derivada de g em relação a x e `eq2` derivada de g em relação a y:

Maxima: Linhas de comando
```
eq1: diff(g(x,y),x);
eq2: diff(g(x,y),y);
```

4. $Gx(x,y)$ derivada de g em relação a x e $Gy(x,y)$ derivada de g em relação a y:

Maxima: Linhas de comando
```
Gx(x,y):= 8*x^3-4*x*y^2;
Gy(x,y):= 4*y^3-4*x^2*y-2;
```

5. Resolvendo o sistema de equações:

Maxima: Linhas de comando
```
solve([eq1,eq2],[x,y]);
```

6. As únicas soluções reais são $P1 = (-1/\sqrt{2}, 1)$ e $P2 = (1/\sqrt{2}, 1)$:
7. Calculando a derivada segunda de g em relação a x e definindo a função `Gxx(x,y)`:

Maxima: Linhas de comando
```
diff(g(x,y),x,2);
Gxx(x,y):= 24*x^2-4*y^2;
```

8. Calculando a derivada segunda de g em relação a y e definindo a função `Gyy(x,y)`:

Maxima: Linhas de comando
```
diff(g(x,y),y,2);
Gyy(x,y):=12*y^2-4*x^2;
```

9. Calculando as derivadas parciais de segunda ordem da função g em relação a x e y, e definindo a função `Gxy(x,y)`:

Maxima: Linhas de comando
```
diff(g(x,y),x,1,y,1);
Gxy(x,y):=-8*x*y;
```

10. Calculando as derivadas parciais de segunda ordem da função g em relação a y e x, e definindo a função `Gyx(x,y)`:

Maxima: Linhas de comando
```
diff(g(x,y),y,1,x,1);
Gyx(x,y):=-8*x*y;
```

11. Construindo a matriz hessiana H no ponto $(-1/\sqrt(2),1)$:

Maxima: Linhas de comando
```
H: matrix([Gxx(-1/sqrt(2),1),Gxy(-1/sqrt(2),1)],
          [Gyx(-1/sqrt(2),1),Gyy(-1/sqrt(2),1)]);
```

12. Determinando a ordem da matriz H:

Maxima: Linhas de comando
```
n1: part(matrix_size(H),1);
```

13. Definindo Z1[n]:

Maxima: Linhas de comando
```
Z1[n]: H;
```

14. Construindo as matrizes de ordem $n-1$ a partir de Z1[n]:

Maxima: Linhas de comando
```
for k:1 thru n1-1 do (Z1[n-k]: submatrix(n1-k+1,Z1[n-k+1],n1-k+1),
    ldisplay(Z1[n-k]));
```

15. Calculando o determinante de Z1[n]:

Maxima: Linhas de comando
```
determinant(Z1[n]);
```

16. Como todos os determinantes são positivos, $P1 = (-1/\sqrt{2},1)$ é ponto de mínimo.
17. Construindo a matriz hessiana H2 no ponto $(1/\sqrt{2},1)$:

Maxima: Linhas de comando
```
H2: matrix([Gxx(1/sqrt(2),1),Gxy(1/sqrt(2),1)],
           [Gyx(1/sqrt(2),1),Gyy(1/sqrt(2),1)]);
```

18. Determinando a ordem da matriz H2:

Maxima: Linhas de comando
```
n2: part(matrix_size(H),1);
```

19. Definindo Z2[n]:

Maxima: Linhas de comando
```
Z2[n]: H2;
```

20. Construindo as matrizes de ordem $n-1$ a partir de Z2[n]:

Maxima: Linhas de comando
```
for k:1 thru n2-1 do (Z2[n-k]: submatrix(n2-k+1,Z2[n-k+1],n2-k+1),
    ldisplay(Z2[n-k]));
```

21. Calculando o determinante de Z2[n]:

Maxima: Linhas de comando
```
determinant(Z2[n]);
```

22. Como todos os determinantes são positivos, $P2 = (1/\sqrt{2},1)$ é ponto de mínimo.

23. O código abaixo mostra a forma de plotar a função dada:

Maxima: Linhas de comando
```
plot3d(g(x,y), [x,-5,5], [y,-5,5], [plot_format,gnuplot])$
```

Agora vamos apresentar uma função de três variáveis.

1. Vamos assumir que $x > 0$, $y > 0$ e $z > 0$:

   Maxima: Linhas de comando
   ```
   assume(x>0,y>0,z>0);
   ```

2. Inserindo a função $h(x,y,z) = x + \dfrac{y^2}{4x} + \dfrac{z^2}{y} + \dfrac{2}{z}$:

   Maxima: Linhas de comando
   ```
   h(x,y,z):= x + y^2/(4*x) + z^2/y + 2/z;
   ```

3. Definindo as equações `eq1` (derivada de h em relação a x, `eq2` (derivada de h em relação a y) e `eq3` (derivada de h em relação a z):

   Maxima: Linhas de comando
   ```
   eq1: diff(h(x,y,z),x);
   eq2: diff(h(x,y,z),y);
   eq3: diff(h(x,y,z),z);
   ```

4. Resolvendo o sistema de equações:

   Maxima: Linhas de comando
   ```
   solve([eq1,eq2,eq3],[x,y,z]);
   ```

5. Calculando as derivadas parciais de segunda ordem da função `h` em relação às variáveis x, y, z, x e y, x e z, y e x, y e z, z e x e z e y:

   Maxima: Linhas de comando
   ```
   mxx: diff(h(x,y,z),x,2);
   mxy: diff(h(x,y,z),x,1,y,1);
   mxz: diff(h(x,y,z),x,1,z,1);
   myx: diff(h(x,y,z),y,1,x,1);
   myy: diff(h(x,y,z),y,2);
   myz: diff(h(x,y,z),y,1,z,1);
   mzx: diff(h(x,y,z),z,1,x,1);
   mzy: diff(h(x,y,z),z,1,y,1);
   mzz: diff(h(x,y,z),z,2);
   ```

6. Construindo a matriz hessiana `M`

   Maxima: Linhas de comando
   ```
   M: matrix([mxx,mxy,mxz], [myx,myy,myz], [mzx,mzy,mzz]);
   ```

7. Substituindo o ponto crítico $(1/2,1,1)$ na matriz `M`:

8. Não substituiremos $(0,0,0)$ e $(-1/2,-1,-1)$, pois assumimos que x, y e z são positivos.

   Maxima: Linhas de comando
   ```
   C: subst(1/2,x,M)$ D: subst(1,y,C)$ E: subst(1,z,D);
   ```

9. Determinando a ordem da matriz E:

Maxima: Linhas de comando
```
n: part(matrix_size(E),1);
```

10. Definindo Z[n]:

Maxima: Linhas de comando
```
Z[n]: E;
```

11. Construindo as matrizes de ordem n-1 a partir de Z[n]:

Maxima: Linhas de comando
```
for k:1 thru n-1 do (Z[n-k]: submatrix(n-k+1,Z[n-k+1],n-k+1),
    ldisplay(Z[n-k]));
```

12. Calculando o determinante das matrizes de ordem n-1

Maxima: Linhas de comando
```
for k:1 thru n-1 do (d[n-k]: determinant(Z[n-k]), ldisplay(d[n-k]));
```

13. Calculando o determinante de Z[n]:

Maxima: Linhas de comando
```
determinant(Z[n]);
```

14. Como todos os determinantes são positivos, $(1/2, 1, 1)$ é ponto de mínimo.

7.4 Sequências e Séries numéricas reais

1. Vá ao menu Maxima e pressione Reiniciar Maxima, para limpar todas as funções, variáveis e expressões que estavam na memória do computador.
2. Dada uma seqüência real f=f(n), podemos criar uma lista de pares ordenados da forma [n,f(n)] onde a variável n pertence ao intervalo [n1,n2], sendo n1<n2. A forma geral da linha de comando para gerar esta lista no Maxima é:

Maxima: Linhas de comando
```
Lista: makelist( [n,f(n)], n, n1, n2 );
```

3. Exemplo: Seja a sequência definida por f(n)=1/n.
   (a) Construção da lista com n variando de 1 a 20:

   Maxima: Linhas de comando
   ```
   x1: makelist([n,1/n], n, 1, 20);
   ```

   (b) Após criar esta lista, podemos plotar a seqüência no intervalo [n1,n2] com o Gnuplot, usando a forma geral plot2d([discrete,Lista], [style, XYZ]); onde XYZ é o estilo que queremos plotar, pode ser: dots, points, lines, ...
   (c) Gráfico da seqüência com o estilo points:

   Maxima: Linhas de comando
   ```
   plot2d( [discrete,x1], [style,points] );
   ```

4. Exemplo: Seja a sequência definida por `f(n)=n/(2n+1)`.

   (a) Construção da lista com `n` variando de 1 a 50:

   Maxima: Linhas de comando
   ```
   x2: makelist([n, n/(2*n+1)], n, 1, 50)$
   ```

   (b) Com a colocação de $ no final da linha de comando, os dados não apareceram na tela. Caso queira visualizar os dados, troque o $ pelo ponto e vírgula.

   (c) Gráfico da seqüência com o estilo `lines`:

   Maxima: Linhas de comando
   ```
   plot2d( [discrete,x2], [style,lines] );
   ```

5. Exemplo: Seja a seqüência definida por `f(n)=(-1)^(n+1)/n`.

   (a) Construção da lista com `n` variando de 1 a 50:

   Maxima: Linhas de comando
   ```
   x3: makelist( [n,(-1)^(n+1)/n], n, 1, 50 )$
   ```

   (b) Gráfico da seqüência com o estilo `impulses`:

   Maxima: Linhas de comando
   ```
   plot2d([discrete,x3], [style,impulses]);
   ```

6. Exemplo: Seja a seqüência definida por `f(n)=n/(1+n^2)` com variáveis no conjunto dos números inteiros.

   (a) Construção da lista com `n` variando de -10 a 10:

   Maxima: Linhas de comando
   ```
   x4: makelist( [n,n/(1+n^2)], n, -10, 10 );
   ```

   (b) Gráfico da seqüência com o estilo `dots`:

   Maxima: Linhas de comando
   ```
   plot2d([discrete,x4],[style,dots]);
   ```

7. Exemplo: Seja a função $f : R \to R$ definida por $f(x) = \frac{x}{1+x^2}$.

   (a) A lista com `x` variando de -10 a 10 é construída com:

   Maxima: Linhas de comando
   ```
   x5: makelist( [x,x/(1+x^2)], x, 0, 50 )$
   ```

   (b) Gráfico da seqüência:

   Maxima: Linhas de comando
   ```
   plot2d(x/(1+x^2), [x, 0, 50]);
   ```

8. Para plotar as seqüências `x2` e `x3` no mesmo gráfico, devemos digitar:

   Maxima: Linhas de comando
   ```
   plot2d([[discrete,x2],[discrete,x3]],[style,points]);
   ```

9. Plotaremos as seqüências `x2`, `x3` e `x5`, no mesmo gráfico, com estilo `points`, com:

   Maxima: Linhas de comando
   ```
   plot2d([[discrete,x2],[discrete,x3],[discrete,x5]],[style,points]);
   ```

10. Para obter o limite de uma função `f=f(x)` no ponto `x=a`, devemos digitar o código: `limit(f(x),x,a);`.

11. Exemplo: Obter o limite da seqüência x5.

Maxima: Linhas de comando
```
limit(x/(1+x^2),x,inf);
```

12. Para plotar a seqüência indicada por x1 juntamente com a função f(x)=1/x no mesmo gráfico, digitamos

Maxima: Linhas de comando
```
plot2d([[discrete,x1],1/x],[x,1,20],[style,[points,3,5],[lines,1,5]]);
```

onde

   (a) f(x)= 1/x é a função a ser plotada;
   (b) [x,1,50] indica que x pertence a [1,50],
   (c) [points,3,5] indica que a seqüência é plotada com o estilo de pontos com a espessura 3 e a cor do ponto igual a 5;
   (d) [lines,1,5] indica que a função é plotada com o estilo de linhas de espessura 1 e a cor da linha igual a 5.

13. Plotamos uma seqüência limitada x3, junto com o seu limite no mesmo gráfico, com:

Maxima: Linhas de comando
```
plot2d([[discrete,x3],0],[x,1,50],[style,[points,3,5],[lines,1,5]]);
```

14. Usando o comando if, analisamos o crescimento ou decrescimento de uma sequência.
15. O Maxima compara u(n) e u(n+1) e retorna um sinal de menor ou maior, respectivamente, se a seqüência é crescente ou decrescente.
16. Exemplo: Determinar se a seqüencia a(n)=exp(n) é crescente ou decrescente.
   (a) Definindo u(n):

   Maxima: Linhas de comando
   ```
   u(n) := exp(n);
   ```

   (b) Comparando os termos u(n) e u(n+1):

   Maxima: Linhas de comando
   ```
   if(u(n) < u(n+1)) then disp(crescente) else disp(decrescente);
   ```

   (c) Concluímos que a seqüência é crescente.
17. Exemplo: Determinar se a seqüencia a(n)=1/exp(n) é crescente ou decrescente.
   (a) Definindo u(n):

   Maxima: Linhas de comando
   ```
   u(n) := 1/exp(n);
   ```

   (b) Comparando u(n) e u(n+1):

   Maxima: Linhas de comando
   ```
   if(u(n) < u(n+1)) then disp(crescente) else disp(decrescente);
   ```

   (c) Concluímos que a seqüência é decrescente.
18. A forma geral de somar elementos em uma série de números no Maxima, é sum(f(n),n,i1,i2); que é a soma de todos os termos da forma f(n), com n variando de i1 a i2.

19. Exemplo: Soma dos recíprocos dos quadrados dos 50 primeiros números naturais.

Maxima: Linhas de comando
```
sum(1/n^2, n, 1, 50);
```

20. Calculando o valor numérico da soma obtida:

Maxima: Linhas de comando
```
bfloat(%);
```

21. Exemplo: Soma das potências de `3^(-n)` sobre todos os números naturais:

Maxima: Linhas de comando
```
sum(1/3^n, n, 1, inf);
```

22. Pondo `simpsum` (simplify sum) no final da linha de comando, simplificamos a soma:

Maxima: Linhas de comando
```
sum (1/3^n, n, 1, inf), simpsum;
```

23. O teste da comparação de séries é utilizado para compar a soma de uma série com a soma de uma série cuja convergência é conhecida.

24. Exemplo: Para determinar se a série $\sum_{n=1}^{\infty} 4/(3^n+1)$ é convergente ou divergente:

    (a) Tomamos $u(n) = 4/(3^n+1)$ e $v(n) = 4/(3^n)$, que são termos positivos.

    (b) Definindo `u(n)` e `v(n)` no Maxima:

    Maxima: Linhas de comando
    ```
    u(n) := 4/(3^n+1); v(n) := 4/(3^n);
    ```

    (c) Utilizaremos a função `compare` para comparar as seqüências `u=u(n)` e `v=v(n)`.

    Maxima: Linhas de comando
    ```
    compare(u(n),v(n));
    ```

    (d) O Maxima responde com o símbolo < o que significa que `u(n) < v(n)`.

    (e) Assim, podemos garantir que a série dos `u(n)` é convergente pois a série dos `v(n)` é uma série geométrica cuja soma é finita, pois a sua razão é $r = 1/3 < 1$.

25. Exemplo: Para saber se a série $\sum_{n=1}^{\infty} 1/(\sqrt{n}-1)$ converge ou diverge:

    (a) Tomamos `u(n)=sqrt(n)-1` e `v(n)=sqrt(n)`, que são positivos.

    (b) Definimos `u(n)` e `v(n)` no Maxima:

    Maxima: Linhas de comando
    ```
    u(n) := sqrt(n)-1; v(n) := sqrt(n);
    ```

    (c) Comparamos `u(n)` e `v(n)`

    Maxima: Linhas de comando
    ```
    compare(u(n),v(n));
    ```

    (d) O Maxima responde com o símbolo < o que significa que `u(n) < v(n)`. Sendo assim, $1/u(n) > 1/v(n)$

    (e) Afirmamos que a série com termo geral $1/u(n)$ é divergente pois a série com termo geral $1/v(n)$ é uma série p-série, com $n = 1/2 < 1$, logo a série dada diverge.

26. O teste da comparação com limite utiliza a razão entre os termos gerais das séries.
27. Exemplo: Determine se é convergente ou divergente a série infinita cujo termo geral é da forma $f(n) = 4/(1+3^n)$, (com n variando de 1 a ∞).
    (a) Definindo `u(n)` e `v(n)` no Maxima:

    Maxima: Linhas de comando
    ```
    u(n) := 4/(3^n+1); v(n) := 4/(3^n);
    ```
    (b) A serie infinita `sum(4/(3^n),n,1,inf)` converge pois é uma série geométrica, cuja razão é `r=1/3<1`.
    (c) Calculando o limite da razão entre os termos gerais das duas séries no Maxima:

    Maxima: Linhas de comando
    ```
    limit(u(n)/v(n),n,inf);
    >>> 0
    ```
    (d) Concluímos que a série dada é convergente.
28. Exemplo: Analisar se a série $S = \sum_{n=1}^{\infty} 1/(n^n)^n$ converge ou diverge.
    (a) Definindo `u(n)` e `v(n)` no Maxima:

    Maxima: Linhas de comando
    ```
    u(n) := 1/(n^n)^n; v(n) := 1/n^2;
    ```
    (b) A serie infinita $\sum_{n=1}^{\infty} 1/n^2$ converge pois é uma p-série, com $p = 2 > 1$.
    (c) Calculando o limite da razão entre os termos gerais das séries no Maxima:

    Maxima: Linhas de comando
    ```
    limit(u(n)/v(n),n,inf);
    ```
    (d) Como S converge, podemos concluir que a série dada é convergente.
29. Exemplo: Determinar se a série $\sum_{n=1}^{\infty} 1/\sqrt{n}$ converge ou diverge.
    (a) Definindo `u(n)` e `v(n)`

    Maxima: Linhas de comando
    ```
    u(n) := 1/sqrt(n); v(n) := 1/n;
    ```
    (b) A série infinita $\sum_{n=1}^{\infty} 1/n$ diverge, pois é a série p-harmônica com `p=1`.
    (c) Concluímos que a série é divergente, pois o limite é ∞, isto é:

    Maxima: Linhas de comando
    ```
    limit(u(n)/v(n),n,inf);
    ```
30. Existe um teste de convergência que utiliza a integral de uma função real que coincide com o termo geral da série.
31. Exemplo: Verificar que a série $\sum_{n=1}^{\infty} 4/(3^n+1)$ converge.
    (a) Definimos a função real:

    Maxima: Linhas de comando
    ```
    f(x):= 4/(3^x+1);
    ```
    (b) Verificamos se a função é decrescente:

    Maxima: Linhas de comando
    ```
    compare(f(x),f(x+1));
    ```

(c) Observe que `f` é uma função continua sobre o conjunto `[1,infinito)`.

(d) Calculando a integral de `f=f(x)` de `x=1` a `infinito`:

Maxima: Linhas de comando
```
integrate(f(x), x, 1, inf);
```

(e) Usamos o comando abaixo para gerar respostas com grande precisão.

Maxima: Linhas de comando
```
bfloat(%);
```

32. Exemplo: Provar que a série harmônica diverge.

(a) Definindo a função e também o seu domínio:

Maxima: Linhas de comando
```
assume(x>0);
f(x) := 1/x;
```

(b) Calculando a derivada da função `f=f(x)` para analisar o decrescimento:

Maxima: Linhas de comando
```
diff(f(x),x);
```

(c) Como $-1/x^2$ é negativo para todo $x > 1$, segue que `f=f(x)` é decrescente.

(d) Observamos que `f` é continua, para $x > 1$.

(e) Assumiremos que todos os índices `n` são números inteiros positivos:

Maxima: Linhas de comando
```
assume(n>0);
```

(f) Calculando a integral de `f=f(x)` de 1 até `n+1`:

Maxima: Linhas de comando
```
Nome: integrate(f(x), x, 1, n+1);
```

(g) Concluímos que a série harmônica é divergente, pois o limite da integral `Nome` quando `n` tende a `+infinito`, isto é,

Maxima: Linhas de comando
```
limit(Nome, n, inf);
```

33. Teste de convergência para séries alternadas (Teste de Leibniz)

34. Exemplo: Provaremos que a série alternada $S1 = \sum_{n=1}^{\infty} (-1)^n \, 1/\sqrt{n}$ converge.

(a) Definindo `a(n)`

Maxima: Linhas de comando
```
a(n) := 1/sqrt(n);
```

(b) Verificando se a série é decrescente:

Maxima: Linhas de comando
```
compare(a(n+1),a(n));
```

(c) Calculando limite do valor absoluto do termo geral:

Maxima: Linhas de comando
```
limit(a(n),n,inf);
```

(d) Pelo teste de séries alternadas, segue que S1 converge.

35. Exemplo: Analisaremos a convergência da série alternada $S2 = \sum_{n=1}^{\infty} (-1)^{n+1}/2^n$.

    (a) Definindo o termo geral `a(n)` no Maxima:

    Maxima: Linhas de comando
    ```
    a(n) := 1/2^n;
    ```

    (b) Verificando se a sequência dos termos gerais é decrescente:

    Maxima: Linhas de comando
    ```
    compare(a(n+1),a(n));
    ```

    (c) Calculando limite do termo geral da série, obtemos:

    Maxima: Linhas de comando
    ```
    limit(a(n), n, inf);
    ```

    (d) Pelo teste de séries alternadas segue que `S2` é convergente.

36. Agora, calcularemos o erro em uma série alternada obtida com $f(x) = \log(1+x)$. Se $x \in (-1,1)$, então $\log(1+x) = \sum_{n=1}^{\infty} (-1)^{n+1} x^n/n$.

    (a) Para obter um limitante superior para o erro cometido em qualquer cálculo com $a \in (-1,1)$ basta obter o maior índice n para o qual $|a^n/n| < Erro$. A soma dos termos da série com índices menores do que `n` fornecem o resultado com o erro informado.

    (b) Para calcular o valor de $\log(1.01)$ pela soma dos três primeiros termos da série, vamos usar a série dada com `x=0.01` e escrever:

    Maxima: Linhas de comando
    ```
    Erro4 : (0.01)^4/4;
    ```

    (c) Para calcular o valor de $\log(1.01)$ pela soma dos 22 primeiros termos da série, vamos usar a série dada com `x=0.01` e escrever:

    Maxima: Linhas de comando
    ```
    Erro22 : (0.01)^23/23;
    ```

37. Utilizaremos agora o Teste da Razão (Critério de d'Alembert) para analisar se a série $S1 = \sum_{n=1}^{\infty} (-1)^{n+1} \frac{n}{2^n}$ converge ou diverge.

    (a) Definindo `u(n)`:

    Maxima: Linhas de comando
    ```
    u(n) := (-1)^(n+1)*(n/2^n);
    ```

    (b) Calculando `u(n+1)` no Maxima:

    Maxima: Linhas de comando
    ```
    u(n+1);
    ```

    (c) Calculando o limite da razão entre o termo seguinte e o anterior da série dada:

    Maxima: Linhas de comando
    ```
    limit(abs(u(n+1)/u(n)), n, inf);
    ```

    (d) Pelo teste da razão, segue que a série `S1` é convergente.

38. Analisar se a série $S2 = \sum_{n=1}^{\infty} \frac{2^n}{n^2}$ converge ou diverge.

    (a) Definindo `u(n)`:

    Maxima: Linhas de comando
    ```
    u(n):= 2^n/n^2;
    ```

    (b) Calculando o limite:

    Maxima: Linhas de comando
    ```
    limit(abs(u(n+1)/u(n)), n, inf);
    ```

    (c) Pelo teste da razão, segue que `S2` é divergente.

39. Exemplo: Determinar se a série $S3 = \sum_{n=1}^{\infty} \frac{1}{n+1}$ converge ou diverge.

    (a) Definindo `u(n)` no Maxima:

    Maxima: Linhas de comando
    ```
    u(n):=1/(n+1);
    ```

    (b) b) Calculando o limite:

    Maxima: Linhas de comando
    ```
    limit(abs(u(n+1)/u(n)), n, inf);
    ```

    (c) Sendo assim, nenhuma conclusão quanto à convergência de `S3` pode ser tirada com este teste, mas como `S3` difere da séria harmônica em apenas um termo, podemos concluir que a série dada é divergente.

40. Usando o Teste da Raiz (de Cauchy), mostraremos que a série $\sum_{n=1}^{\infty} \frac{2^{3n+1}}{n^n}$ converge.

    (a) Definindo `u(n)`:

    Maxima: Linhas de comando
    ```
    u(n) := 2^(3*n+1)/n^n;
    ```

    (b) Calculando $\sqrt[n]{|u(n)|}$:

    Maxima: Linhas de comando
    ```
    a1 : abs(u(n))^(1/n);
    ```

    (c) Concluímos que a série é convergente, pois o limite de `a1` é:

    Maxima: Linhas de comando
    ```
    limit(a1,n,inf);
    ```

41. Exemplo: Mostraremos que a série $\sum_{n=1}^{\infty} \exp(n^2)/n^n$ diverge.

    (a) Definindo `u(n)`:

    Maxima: Linhas de comando
    ```
    u(n) := exp(n^2)/n^n;
    ```

    (b) Calculamos $\sqrt[n]{|u(n)|}$, digitando:

    Maxima: Linhas de comando
    ```
    a2: abs(u(n))^(1/n);
    ```

    (c) Concluímos que a série é divergente, pois o limite de `a2` é:

    Maxima: Linhas de comando
    ```
    limit(a2,n,inf);
    ```

42. Exemplo: Analisaremos se a série $\sum_{n=1}^{\infty}(\frac{n}{n+1})^n$ converge ou diverge.

    (a) Tomamos `u(n)`:

    Maxima: Linhas de comando
    ```
    u(n):=(n/(n+1))^n;
    ```

    (b) Calculamos $\sqrt[n]{|u(n)|}$, com:

    Maxima: Linhas de comando
    ```
    a3: abs(u(n))^(1/n);
    ```

    (c) Calculamos o limite de `a3`:

    Maxima: Linhas de comando
    ```
    limit(a3,n,inf);
    ```

    (d) Não podemos concluir quanto à convergência desta série com este teste, mas, como $\lim_{n\to\infty} u(n) = 1/e$, concluímos que esta série diverge.

7.5 Séries de potências

1. Vá ao menu `Maxima` e pressione `Reiniciar Maxima`, para limpar todas as funções, variáveis e expressões que estavam na memória do computador.
2. Obtemos uma série de potências usando a linha de comando `powerseries(expr,x,a);`.
3. Exemplo: Para obter uma série de potência para `log(1-x)`, definimos a expressão:

   Maxima: Linhas de comando
   ```
   f(x) := log(1-x);
   ```

4. Geramos a série:

   Maxima: Linhas de comando
   ```
   powerseries(f(x),x,0);
   ```

5. Exemplo: Obter uma série de potência para `log(sin(x)/x)`, com:

   Maxima: Linhas de comando
   ```
   powerseries(log(sin(x)/x),x,0);
   ```

6. O raio de convergência da série de potências é obtido por

   Maxima: Linhas de comando
   ```
   r: limit(abs(a(n)/a(n+1)),n,inf);
   ```

7. Exemplo: Obter o raio de convergência da série de potências $\sum_{n=1}^{\infty} n!x^n$.

   Maxima: Linhas de comando
   ```
   a(n) := n!*x^n;
   ```

   Maxima: Linhas de comando
   ```
   r: limit(abs(a(n)/a(n+1)),n,inf);
   ```

8. Como o raio de convergência é 0, a série dada converge apenas em `x=0`.

9. Podemos derivar ou integrar uma série de potências e a série obtida possui o mesmo raio de convergência da série inicial

10. Consideremos a série

Maxima: Linhas de comando
```
a: powerseries(log(1-x),x,0);
```

11. Calculando a derivada de a com relação a x:

Maxima: Linhas de comando
```
diff(a,x);
```

12. Calculando a integral de a em relação a x

Maxima: Linhas de comando
```
integrate(a,x);
```

13. Para obter uma série de Maclaurin, até o grau n, digitamos o seguinte comando:

Maxima: Linhas de comando
```
taylor(expr, x, 0, n);
```

14. Exemplo: Obter a série de MacLaurin para `log(x+1)` até `n=10`:

Maxima: Linhas de comando
```
taylor(log(x+1), x, 0, 10);
```

15. Exemplo 2: Obter a série de MacLaurin para $f(x) = 1/(\cos(x) - \sec(x))^3$ até `n=4`.

Maxima: Linhas de comando
```
taylor(1/(cos(x) - sec(x))^3, x, 0, 4);
```

16. Exemplo: Obter a série de MacLaurin para `sin(y+x)`, até `n=3`:

Maxima: Linhas de comando
```
taylor(sin(y+x), x, 0, 3, y, 0, 3);
```

17. A forma geral para obter uma série de Taylor em torno do ponto a é:

Maxima: Linhas de comando
```
taylor(expr, x, a, n);
```

18. Exemplo: Obter a série de Taylor para `exp(x)` em torno do ponto 2 até `n=6`:

Maxima: Linhas de comando
```
taylor(exp(x), x, 2, 6);
```

19. Exemplo: Obter a série de Taylor para $\sqrt{\sin(x)+x+1}$, em torno do ponto `pi/2` até `n=3`:

Maxima: Linhas de comando
```
taylor(sqrt(sin(x)+x+1), x, %pi/2, 3);
```

20. Exemplo: Obter a série de Taylor para `1/log(x+1)` em torno do ponto 3 até `n=4`:

Maxima: Linhas de comando
```
taylor(1/log(x+1), x, 3, 4);
```

8 Elementos de geometria diferencial no plano

1. Vá ao menu `Maxima` e pressione `Reiniciar Maxima`, para limpar todas as funções, variáveis e expressões que estavam na memória do computador.

2. A representação paramétrica de uma curva plana pode ser escrita como um vetor em duas dimensões. Os elementos desse vetor são funções do parâmetro `t`:

   Maxima: Linhas de comando
   ```
   g(t):= [sin(t),cos(t)];
   ```

3. Podemos derivar a função `g=g(t)` com relação à variável `t`:

   Maxima: Linhas de comando
   ```
   diff(g(t),t);
   >>> [ cos(t) , -sin(t)]
   ```

4. Vamos definir uma função (`estrutura complexa`):

   Maxima: Linhas de comando
   ```
   R(a):= [-a[2], a[1]];
   ```

$$R(a) := [-a_2, a_1]$$

5. Na definição de `R`, assumimos que `a` é um vetor com exatamente dois elementos. Isso não foi declarado nem foi verificado. Vejamos como funciona este operador:

   Maxima: Linhas de comando
   ```
   R(g(y));
   >>> [ -cos(y), sin(y) ]
   ```

6. A ação do operador `R` sobre a curva `g=g(t)` é uma rotação de 90 graus. O produto escalar entre os vetores `g(t)` e `R(g(t))` é zero, pois:

   Maxima: Linhas de comando
   ```
   g(t) . R(g(t));
   >>> 0
   ```

7. Agora vamos definir a curvatura `k` de uma curva plana `z=z(t)`:

   Maxima: Linhas de comando
   ```
   df1: diff(z(t),t);
   df2: diff(z(t),t,2);
   k(z,t):= (df2.R(df1))/(df1.df1)^(3/2);
   ```

$$k(z,t) := \frac{\mathrm{df2}.R(\mathrm{df1})}{(\mathrm{df1}.\mathrm{df1})^{3/2}}$$

8. Agora vamos calcular a curvatura da curva `g=g(t)`:

   Maxima: Linhas de comando
   ```
   k(g,t);
   ```

$$\frac{-\sin(t)^2-\cos(t)^2}{(\sin(t)^2+\cos(t)^2)^{\frac{3}{2}}}$$

9. O resultado não *parece* simples, mas `g=g(t)` é uma circunferência e sabemos que esta curva tem raio e curvatura constante. Simplificamos este resultado com:

Maxima: Linhas de comando
```
trigsimp(%);
>>> -1
```

Observamos que o Maxima confirma que a circunferência tem curvatura constante.

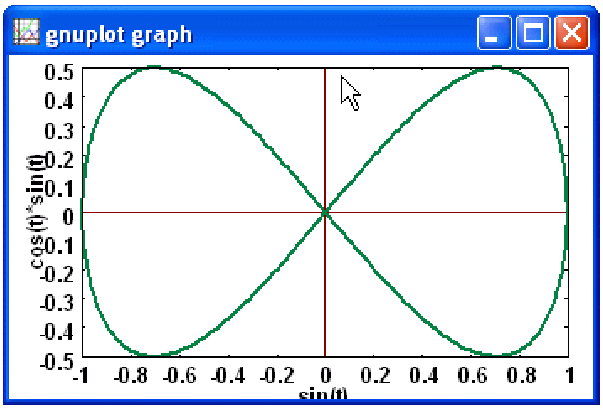
10. Agora, vamos examinar uma curva em forma de `oito` indicada por `m=m(t)`:

Maxima: Linhas de comando
```
m(t):= [sin(t), sin(t)*cos(t)];
>>> m(t) := [ sin(t), sin(t) cos(t) ]
```

11. Desenhamos esta curva com a linha de comando:

Maxima: Linhas de comando
```
plot2d(append('[parametric], m(t),[[t,-%pi,%pi]],[[nticks,360]]));
```

12. No Maxima, a figura sempre é mostrada em uma janela separada:



13. Agora calcularemos a curvatura de `m=m(t)`

Maxima: Linhas de comando
```
k(m,t);
```

$$-1$$

14. Vamos montar a seqüência anterior em uma única etapa com várias linhas de código:

Maxima: Linhas de comando
```
m(t):= [sin(t),sin(t)*cos(t)];
df1: diff(m(t),t);
df2: diff(m(t),t,2);
R(a):= [-a[2],a[1]];
k(m,t):= df2.R(df1)/(df1.df1)^(3/2);
trigsimp(k(m,t));
```

15. Aprendemos que os elementos de um vetor são escritos entre colchetes e que o ponto `.` é usado para o produto escalar entre dois vetores.

9 Equações Recursivas

1. Vá ao menu Maxima e pressione `Reiniciar Maxima`, para limpar todas as funções, variáveis e expressões que estavam na memória do computador.

2. Para resolver equações de diferenças, devemos carregar o pacote `solve_rec`, o que pode ser feito com o seguinte comando:

   Maxima: Linhas de comando
   ```
   load(solve_rec);
   ```

3. Vamos definir uma equação recursiva homogênea (de Fibonacci):

   Maxima: Linhas de comando
   ```
   eq1: f[n]=f[n-1]+f[n-2];
   ```

$$f_n = f_{n-1} + f_{n-2}$$

4. Podemos resolver esta equação recursiva com condições iniciais, com:

   Maxima: Linhas de comando
   ```
   solve_rec(eq1, f[n], f[1]=1, f[2]=1);
   ```

$$f_n = \frac{\sqrt{5}\,(\sqrt{5}+1)^n}{5\,2^n} - \frac{(\sqrt{5}-1)^n\,\sqrt{5}\,(-1)^n}{5\,2^n}$$

5. Vamos definir uma equação recursiva não-homogênea (de Fibonacci):

   Maxima: Linhas de comando
   ```
   eq2: f[n]=f[n-1]+f[n-2] + n/2^n;
   ```

$$f_n = f_{n-1} + f_{n-2} + \frac{n}{2^n}$$

6. Resolvemos equação de diferenças com condições iniciais, com:

   Maxima: Linhas de comando
   ```
   solve_rec(eq2, f[n], f[1]=1, f[2]=1);
   ```

$$f_n = -\frac{(\sqrt{5}-1)^n\,(27\sqrt{5}+5)\,(-1)^n}{100\,2^n} - \frac{n}{5\,2^n} + \frac{(\sqrt{5}+1)^n\,(27\sqrt{5}-5)}{100\,2^n} - \frac{2}{5\,2^n}$$

7. Tomemos uma equação de diferenças diferente (não coloquei =0):

   Maxima: Linhas de comando
   ```
   2*x*(x+1)*y[x] - (x^2+3*x-2)*y[x+1] + (x-1)*y[x+2];
   ```

$$(x-1)\,y_{x+2} - (x^2+3x-2)\,y_{x+1} + 2x\,(x+1)\,y_x$$

8. A solução desta equação de diferenças com condições iniciais é obtida por:

   Maxima: Linhas de comando
   ```
   solve_rec(%, y[x], y[1]=1, y[3]=3);
   ```

$$y_x = 3\,2^{x-2} - \frac{x!}{2}$$

9. Tomemos a equação recursiva de 2a. ordem:

Maxima: Linhas de comando
```
eq3: f[n+2]=7*f[n+1]-10*f[n];
```

$$f_{n+2} = 7\,f_{n+1} - 10\,f_n$$

10. A solução geral desta equação é obtida por:

Maxima: Linhas de comando
```
solve_rec(eq3, f[n]);
```

$$f_n = \%k_1\,5^n + \%k_2\,2^n$$

11. Uma solução particular desta equação com condições iniciais, é dada por:

Maxima: Linhas de comando
```
solve_rec(eq3, f[n], f[1]=20, f[2]=70);
```

$$f_n = 2\,5^n + 5\,2^n$$

12. Resolvendo uma equação recursiva inserida na linha de comando:

Maxima: Linhas de comando
```
solve_rec((n+4)*s[n+2]+s[n+1]-(n+1)*s[n],s[n]);
```

$$s_n = \frac{\%k_2}{(n+1)\,(n+2)} - \frac{\%k_1\,(2n+3)\,(-1)^n}{(n+1)(n+2)}$$

13. Uma equação recursiva de 1a. ordem conhecida como pg:

Maxima: Linhas de comando
```
pg: a[n+1]-q*a[n]$
solve_rec(pg, a[n], a[1]=a1);
```

$$a_n = a1\,q^{n-1}$$

14. Uma equação recursiva de 2a. ordem conhecida como pa:

Maxima: Linhas de comando
```
pa: a[n+2]-2*a[n+1]+a[n]$
solve_rec(pa, a[n], a[1]=a1, a[2]=a1+r);
```

$$a_n = n\,r - r + a1$$

15. Usamos a função `radcan` para colocar a resposta na forma usual:

Maxima: Linhas de comando
```
radcan(%);
```

$$a_n = (n-1)\,r + a1$$

16. Vamos tomar uma uma equação recursiva e resolver

Maxima: Linhas de comando
```
eq5: g[n+1] = g[n]*3/2$
solve_rec(eq5,g[n]);
```

$$g_n = \frac{\%k_1\, 3^n}{2^n}$$

17. Tomemos outra equação recursiva e vamos obter a sua solução:

Maxima: Linhas de comando
```
eq6: h[n+2]=h[n+1]*2 + 3*h[n]$
sol: solve_rec(eq6, h[n]);
```

$$h_n = \%k_1\, 3^n + \%k_2\, (-1)^n$$

18. Para obter uma expressão para um dado valor de `n`, como por exemplo, `n=5`, pode-se avaliar:

Maxima: Linhas de comando
```
ev(sol, n=5);
```

$$h_5 = 243\,\%k_1 - \%k_2$$

19. A mesma função `ev` pode também ser usada para fornecer valores para os valores iniciais `h[0]` e `h[1]`.

Maxima: Linhas de comando
```
ev(sol,n=0);
ev(sol,n=1)];
```

$$h_0 = \%k_2 + \%k_1$$
$$h_1 = 3\,\%k_1 - \%k_2$$

10 Interpolação de Lagrange e Interpolação por spline

1. Vá ao menu `Maxima` e pressione `Reiniciar Maxima`, para limpar todas as funções, variáveis e expressões que estavam na memória do computador.
2. Faremos um breve estudo de interpolação Lagrange (polinomial). Carregaremos o pacote: `interpol` que contém a função `lagrange`, necessária aos nossos propósitos.

Maxima: Linhas de comando
```
load("interpol");
```

3. Vamos inserir os dados, observando que as abscissas estão fora de ordem:

Maxima: Linhas de comando
```
p: [[7,2],[8,2],[1,5],[3,2],[6,7]];
>>> [ [7,2], [8,2], [1,5], [3,2], [6,7] ]
```

4. Chamamos a função `lagrange` sobre a lista `p` dos dados:

Maxima: Linhas de comando
```
lagrange(p);
```

$$\frac{73\,x^4}{420} - \frac{701\,x^3}{210} + \frac{8957\,x^2}{420} - \frac{5288\,x}{105} + \frac{186}{5}$$

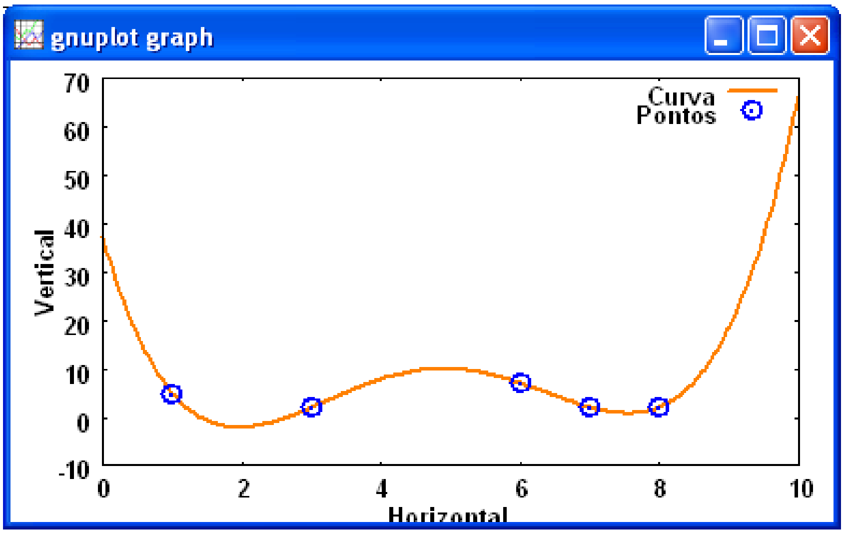
5. Criamos uma função polinomial `f=f(x)` com o último resultado:

Maxima: Linhas de comando
```
f(x):=''%;
```

$$f(x) := \frac{73\,x^4}{420} - \frac{701\,x^3}{210} + \frac{8957\,x^2}{420} - \frac{5288\,x}{105} + \frac{186}{5}$$

6. Plotamos a função polinomial `f=f(x)` junto com os pontos:

Maxima: Linhas de comando
```
plot2d([f(x),[discrete,p]],[x,0,10],[style,[lines,1,3],[points,9,6]],
  [legend,"Curva","Pontos"],[xlabel,"Horizontal"],[ylabel,"Vertical"]);
```



7. Aplicamos (mapeamos) a função polinomial `f=f(x)` em alguns pontos:

Maxima: Linhas de comando
```
map(f, [2.3,5/7,%pi]);
```

$$[-1.567534999999992, \frac{919062}{84035}, \frac{73\,\pi^4}{420} - \frac{701\,\pi^3}{210} + \frac{8957\,\pi^2}{420} - \frac{5288\,\pi}{105} + \frac{186}{5}]$$

8. Para ver o último resultado em números decimais, pressione `Ctrl-F` ou digite:

Maxima: Linhas de comando
```
float(%), numer;
>>> [-1.567534999999992,10.9366573451538,2.893196551256935]
```

9. Mudamos o nome da variável da função `f=f(x)`:

Maxima: Linhas de comando
```
lagrange(p, varname=w);
```

$$\frac{73\,w^4}{420} - \frac{701\,w^3}{210} + \frac{8957\,w^2}{420} - \frac{5288\,w}{105} + \frac{186}{5}$$

10. Vá ao menu `Maxima` e pressione `Reiniciar Maxima`, para limpar todas as funções, variáveis e expressões que estavam na memória do computador.

11. Realizaremos um pequeno estudo de splines cúbicos. Primeiro, carregaremos o pacote: `interpol` que contém a função `cspline`, necessária aos nossos propósitos.

Maxima: Linhas de comando
```
load("interpol");
```

12. Vamos inserir os dados, observando que as abscissas estão fora de ordem:

Maxima: Linhas de comando
```
p: [[7,2],[8,4],[1,5],[3,2],[6,7]];
>>> [ [7,2], [8,4], [1,5], [3,2], [6,7] ]
```

13. Chamamos a função `cspline` SEM as derivadas nas extremidades, com

Maxima: Linhas de comando
```
cspline(p);
```

Aparece uma expressão matemática muito grande que não colocaremos aqui.

14. Criamos o primeiro spline `f=f(x)` com a última saída do Maxima:

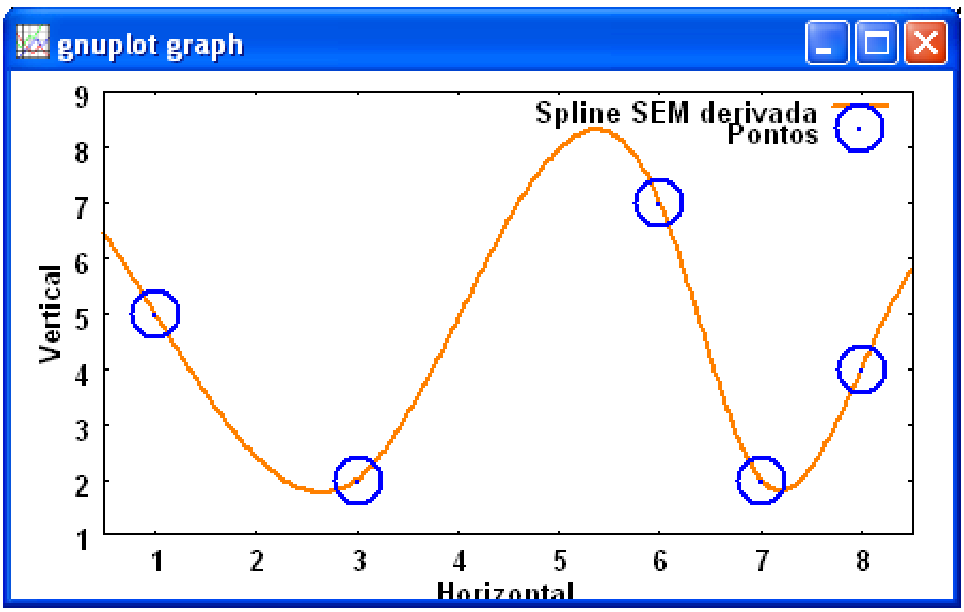
Maxima: Linhas de comando
```
f(x):=''%$
```

15. Aplicamos a função `map` aplica o spline f=f(x) a alguns valores de x:

Maxima: Linhas de comando
```
map(f,[2.3,5/7,%pi]), numer;
>>> [1.958581204379562,5.835457853631547,2.241034852942001]
```

16. Plotamos o spline `f=f(x)` bem como os pontos interpolados:

Maxima: Linhas de comando
```
plot2d(['(f(x)),[discrete,p]],[x,.5,8.5],[style,[lines,1,3],[points,7,6]],
  [legend,"SEM derivada","Pontos"],[xlabel,"Horizontal"],[ylabel,"Vertical"])
```



17. Vamos construir uma outra função `cspline` COM as derivadas nas extremidades:

Maxima: Linhas de comando
```
cspline(p, d1=0, dn=0);
```

Aparece uma expressão matemática muito grande que não colocaremos aqui.

18. Criamos o segundo spline g=`g(x)` com a última saída do Maxima:

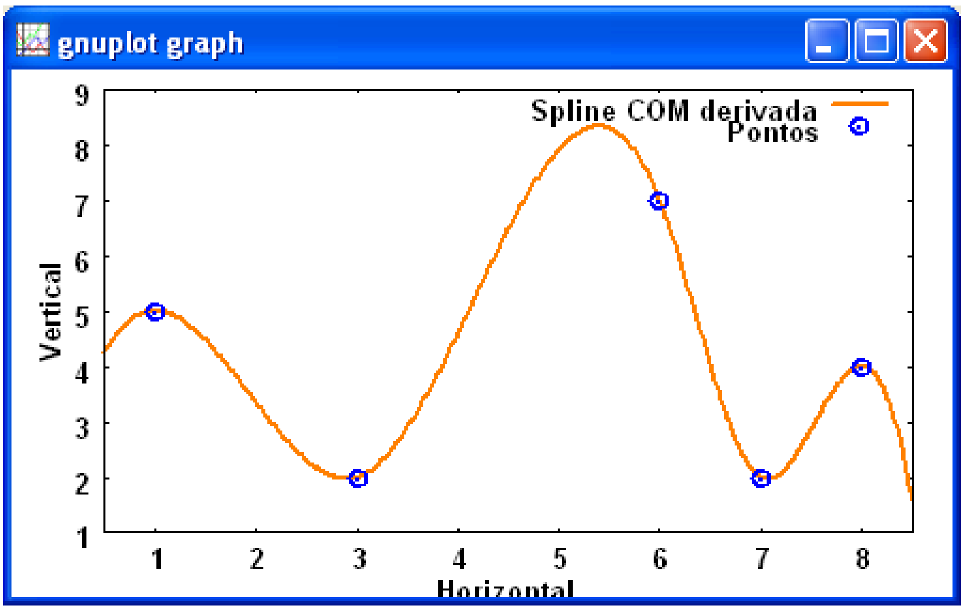
Maxima: Linhas de comando
```
g(x):=''%$
```

19. Aplicamos a função `map` aplica o spline g=g(x) a alguns valores de x:

Maxima: Linhas de comando
```
map(g,[2.3,5/7,%pi]), numer;
>>> [2.672729166666665,4.771622934888241,2.137060426557962]
```

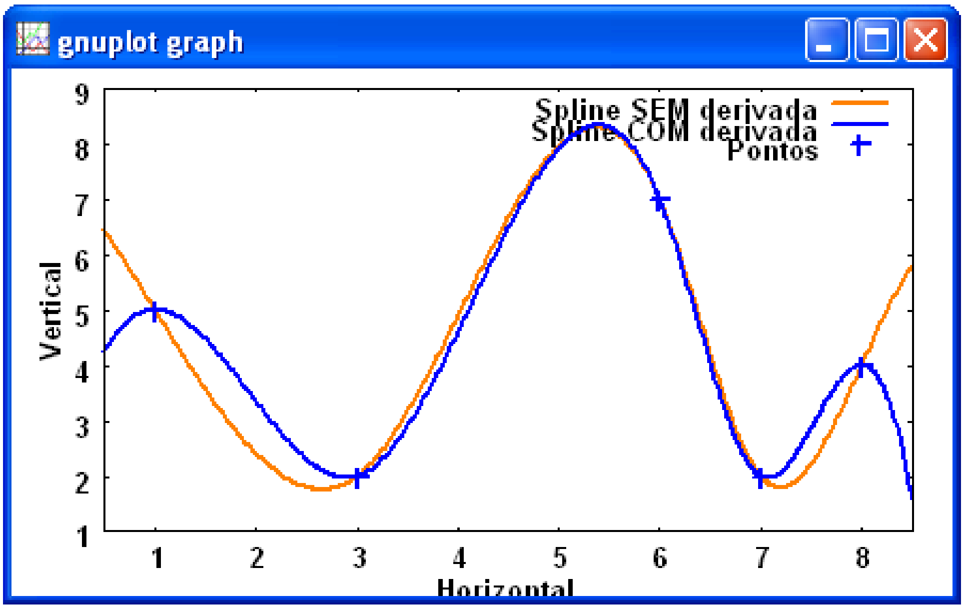
20. Plotamos o spline g=`g(x)` bem como os pontos interpolados:

Maxima: Linhas de comando
```
plot2d(['(g(x)),[discrete,p]],[x,.5,8.5],[style,[lines,1,3],[points,7,6]],
  [legend,"COM derivada","Pontos"],[xlabel,"Horizontal"],[ylabel,"Vertical"])
```



21. Agora plotamos os dois splines `f=f(x)` e g=`g(x)` e os pontos interpolados:

Maxima: Linhas de comando
```
plot2d(['(f(x)),'(g(x)),[discrete,p]],[x,.5,8.5],
       [style,[lines,1,3],[lines,1,6],[points,7,6]],
       [legend,"Spline SEM der","Spline COM der","Pontos"],
       [xlabel,"Horizontal"],[ylabel,"Vertical"]);
```



11 Ajuste linear

1. Vamos usar o Maxima para obter a reta `y=a+bx` de regressão linear, pelo Método dos Mínimos Quadrados. Para isto, deveremos resolver um sistema linear da forma:

$$\begin{array}{rcrcl} a\,n & + & b\,Sx & = & Sy \\ a\,Sx & + & b\,Sx2 & = & Sxy \end{array}$$

onde `n` é o número de elementos tanto da lista `x` como da lista `y` e

$$Sx = \sum_{i=1}^{n} x_i, \qquad Sy = \sum_{i=1}^{n} y_i, \qquad Sx2 = \sum_{i=1}^{n} (x_i)^2, \qquad Sxy = \sum_{i=1}^{n} (x_i)(y_i)$$

2. Vamos realizar a entrada de dados através de três listas:

Maxima: Linhas de comando

```
x: [-1, 0,1,2]$  y: [-2,-1,0,1]$  xy: [[-1,-2],[0,-1],[1,0],[2,1]]$
```

3. Vamos definir alguns elementos necessários à resolução do sistema:

Maxima: Linhas de comando

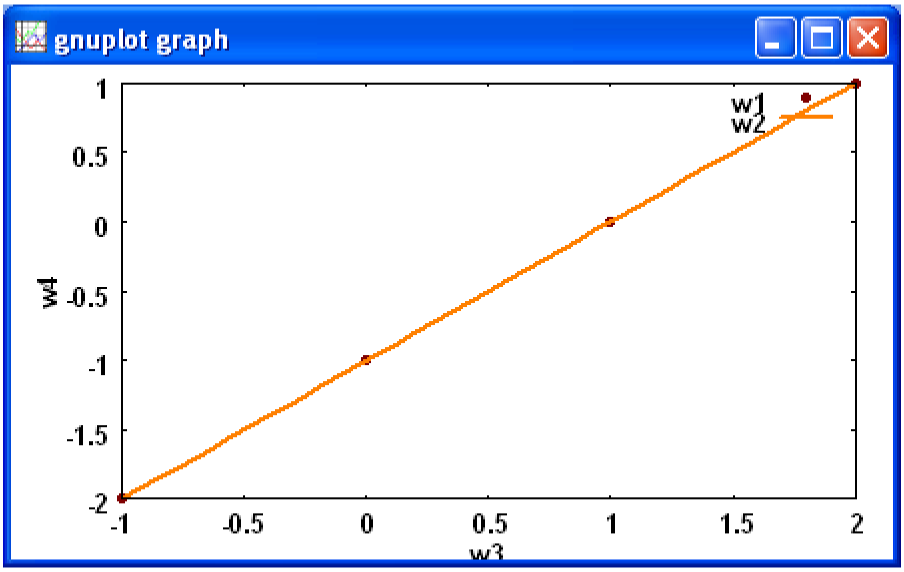
```
n:length(x)$
Sx: sum(x[i],i,1,n), simpsum$
Sy: sum(y[i],i,1,n), simpsum$
Sx2: sum(x[i]^2,i,1,n), simpsum$
Sxy: sum(x[i]*y[i],i,1,n), simpsum$
[n, Sx, Sy, Sx2, Sxy];
>>> [4,2,-2,6,4]
```

4. Para resolver o sistema, vamos usar uma variável de controle e pedir ao Maxima que resolva o sistema linear usando a função `linsolve`:

Maxima: Linhas de comando
```
globalsolve:true$
linsolve([a*n + b*Sx = Sy, a*Sx + b*Sx2 = Sxy], [a,b]);
>>> [a: -1, b: 1]
```

5. Agora vamos plotar a reta de regressão bem como os pontos ajustados:

Maxima: Linhas de comando
```
plot2d([[discrete,xy],a+b*z],[z,x[1],x[n]],
        [style,[points,3,5],[lines,1,3]],[legend,"w1","w2"],
        [xlabel,"w3"],[ylabel,"w4"]);
```



6. Salmo 91

1 Aquele que habita no esconderijo do Altíssimo, à sombra do Todo-Poderoso descansará.

2 Direi do Senhor: Ele é o meu refúgio e a minha fortaleza, o meu Deus, em quem confio.

3 Porque ele te livra do laço do passarinho, e da peste perniciosa.

4 Ele te cobre com as suas penas, e debaixo das suas asas encontras refúgio; a sua verdade é escudo e broquel.

5 Não temerás os terrores da noite, nem a seta que voa de dia,

6 nem peste que anda na escuridão, nem mortandade que assola ao meio-dia.

7 Mil poderão cair ao teu lado, e dez mil à tua direita; mas tu não serás atingido.

8 Somente com os teus olhos contemplarás, e verás a recompensa dos ímpios.

9 Porquanto fizeste do Senhor o teu refúgio, e do Altíssimo a tua habitação,

10 nenhum mal te sucederá, nem praga alguma chegará à tua tenda.

11 Porque aos seus anjos dará ordem a teu respeito, para te guardarem em todos os teus caminhos.

12 Eles te sustentarão nas suas mãos, para que não tropeces em alguma pedra.

13 Pisarás o leão e a áspide; calcarás aos pés o filho do leão e a serpente.

14 Pois que tanto me amou, eu o livrarei; pô-lo-ei num alto retiro, porque ele conhece o meu nome.

15 Quando ele me invocar, eu lhe responderei; estarei com ele na angústia, livrá-lo-ei, e o honrarei.

16 Com longura de dias fartá-lo-ei, e lhe mostrarei a minha salvação.

Índice