Universidade de São Paulo
Instituto de Matemática e Estatística
Laboratório de Ensino de Matemática

Uma introdução ao MuPAD

Leônidas O. Brandão
DCC - IME - USP
E-mail: `leo@ime.usp.br`
WWW: `http://www.ime.usp.br/~leo`

Carlos Juiti Watanabe
Doutorando - IME - USP
E-mail: `juiti@ime.usp.br`
WWW: `http://www.ime.usp.br/~juiti`

Apostila de apoio didático aos alunos do Instituto de Matemática e Estatística da Universidade de São Paulo.

São Paulo – Outubro de 1997

1 Introdução

O objetivo deste texto é fazer uma apresentação rápida do programa MuPAD, introduzindo alguns comandos "básicos" e com isto permitindo ao leitor começar a usar MuPAD. Para maiores detalhes recomendamos a leitura do manual "on line" do MuPAD (está no `HELP | MuPAD MANUAL`).

O MuPAD é um programa desenvolvido principalmente pela Universidade de Paderborn (Alemanha), sendo distribuido gratuitamente para propósitos educativos. O MuPAD é análogo aos famosos: Mathematica^R da Wolfram Institute of Research e ao Maple^R da Universidade de Waterloo. Todos esses pacotes, dentre outros, dedicam-se principalmente a tornar a vida do matemático (ou do usuário de Matemática) mais prática.

O MuPAD é um sistema de "álgebra computacional" e por isto podemos trabalhar, por exemplo, com frações e não com suas aproximações em ponto flutuante. Assim, $1/2 + 1/3$ resulta $5/6$, quando em ponto flutuante teríamos algo como $0.5 + 0.333 = 0.833$ (sic).

O MuPAD é um sistema que pode fazer desde cálculos simples até complexos, além de dispor de comandos para visualização de funções. Tais características fazem dele um "software genérico", que pode ser útil para especialistas de todas as áreas que necessitem efetuar "contas".

2 Convenções

O MuPAD trabalha de modo interativo, isto é, ele contém uma área de trabalho ("work space") na qual você deve digitar seu comandos e visualizar as respostas obtidas. A seguir resumimos as principais características do pacote e que devem ser seguidas sob pena de nada funcionar!

- Para que um cálculo seja efetuado, deve ser digitado um terminador: "ponto e vírgula" (;) ou "dois pontos" (:), após o término de TODA expressão ou comando. O ; fará com que imediatamente após o ENTER, apareça o resultado desta expressão ou comando, enquanto que o : não permite isto (por isto é recomendável usar o ; só quando estiver interessado em visualizar o comando).
- Sempre que terminar de digitar uma expressão ou comando deve ser teclar a ENTER ou um <SHIFT>ENTER. (Sempre que você tiver uma série de comandos relacionados - com dependências - dê preferência ao

<SHIFT>ENTER, pois isto fará o MuPAD não registrar imediatamente seu comando e para futuras alterações destes comandos, bastará um ENTER ao final para todos os comandos sejam novamente interpretados.)

- Quando o MuPAD está pronto para executar alguma instrução, surge um sinal •, após o qual o usuário pode digitar o nome da instrução ou do comando. Caso haja alguma alteração, o usuário pode utilizar as setas e o "mouse" para mover o cursor até o erro e corrigi-lo.
- Há diferenciação entre maiúsculas e minúsculas. Isso quer dizer que, por exemplo, **sin(x)** é diferente de **Sin(x)** ou **sin(X)**. (Todos os comandos MuPAD são em letras minúsculas.)
- Existe o sinal de igual (=) e um símbolo **:=** (como no Turbo Pascal). Este último serve para atribuir valores ou expressões a variáveis/funções como, por exemplo: **a:=10:** e **b:=a+100:**.
- O MuPAD tem constantes irracionais, como π e e. Todas essas constantes devem ter seus nomes digitados em letras maiúsculas, por exemplo o π deve ser digitado **PI** e para se obter um valor aproximado (expandido em forma decimal), basta digitar **float(PI)**;
- Se for necessário, por qualquer motivo, alterar a quantidade de casas decimais para a aproximação, basta digitar: **DIGITS:=**n; onde n é o número de casas decimais desejadas.
- Existe uma "entidade" chamada **infinity**, que serve para representar o "infinito". Um exemplo será apresentado quando falarmos sobre somatórios.
- Os nomes de funções devem ser digitados em letras minúsculas, portanto, para calcular o seno de π, deve ser digitado **sin(PI)**.
- Os valores obtidos pelo MuPAD, são (ou tentam ser) os mais "precisos " possíveis, isto é, se algum resultado for um irracional como o $\sqrt{2}$, o MuPAD irá apresentar $2^{1/2}$, como valor de $\sqrt{2}$.
- Para elevar um número x a outro y, devemos digitar x ^ y.

3 Funções e Comandos Úteis

Existem algumas funções que fazem do MuPAD, um "software" interessante (sob o ponto de vista de " facilitar a vida" do usuário). A seguir listamos alguns deste comandos, com uma rápida explicação e exemplos de uso.

<u>Muito cuidado com a sintaxe!</u> Esquecer de colocar um terminador (**:** ou **;**) ao final de um comando ou expressão significa ERRO. Qualquer letra trocada de maiúscula para minúscula ou vice versa é motivo de ERRO. Por isto, preste atenção.

1. **array**($i_1..f_1$, $i_2..f_2$,$\cdots$, $i_n..f_n$);
Define um "agregado" de variáveis, como se fosse um vetor. Por exemplo, para definir um vetor de nome `A` com índices entre 3 e 19, preenchendo a elemento A_i com o valor i^3, faça o seguinte:

• A := **array**(3..19): <SHIFT>ENTER	`define um vetor com elementos 3 até 19;`
• **for** n **from** 3 **to** 19 **do** <SHIFT>ENTER	`''laço'', n indo de 3 até 19 (item 7 abaixo);`
• A[n] := n^3**:** <SHIFT>ENTER	`define a posição` n `do vetor A;`
• **print**("A[", n, "] = ", A[n])**:** <SHIFT>ENTER	`''imprime'' na tela (vide item 11);`
• **end_for:** ENTER	`finaliza o comando` **for**;

(para ver como funciona o comando **for** veja o item **7.**.)
Para definir um agregado com duas dimensões faça: A := **array**(1..2,1..4):. Os elementos deste agregado são: $A[1,1]$, $A[1,2]$,$\cdots$, $A[1,4]$, $A[2,1]$, $A[2,2]$, $\cdots$.

2. binomial(*expressão*$_1$, *expressão*$_2$);

Calcula o número binomial $\binom{\text{expressão}_1}{\text{expressão}_2}$. Por exemplo: **binomial**(4, 2) calcula o número $\binom{4}{2} = \frac{4!}{2!2!} = 6$.

3. exp(*expressão*);

Calcula $e^{\mathit{expressão}}$, sendo que *expressão* deve ser um número. Exemplo:

- **float**(**exp**(**PI*sqrt**(163)));

calcula uma aproximação para $e^{\pi\sqrt{163}}$.

4. fact(*número*);

Essa função retorna o fatorial de um número natural. Por exemplo, se quisermos calcular o fatorial de 5 que é $5{\times}4{\times}3{\times}2{\times}1{\times}1$ (lembramos que o fatorial de 0 é 1 e que o fatorial de $n \geq 1$ é $n{\times}(n-1)!$). devemos digitar:

- **fact**(5);

e obtemos como resultado 120, que é 5!

5. factor(*expressão*);

Essa função simplesmente fatora a expressão, ou seja, faz uma decomposição da expressão, resultado um "vetor", onde a primeira coordenada é o sinal, a segunda coordenada é um fator, cujo expoente é a terceira coordenada, a quarta coordenada é um fator cujo expoente é a quinta coordenada e assim por diante. Por exemplo: se quisermos calcular a fatoração de $(x^2 - y^2)^2$, devemos digitar:

- **factor**($(x \,\hat{}\, 2 - y \,\hat{}\, 2) \,\hat{}\, 2$);

obtendo como resultado o vetor $[1,\ x-y,\ 2,\ x+y,\ 2]$: o 1 é o sinal (no caso, positivo), $x+y$ aparece ao quadrado, e portanto está explicado o porque do 2 logo após o $x+y$ e o $x-y$ também aparece ao quadrado (o que explica o 2 após o $x-y$), isto é, **factor**($(x \,\hat{}\, 2 - y \,\hat{}\, 2) \,\hat{}\, 2$) $= 1{\times}(x+y) \,\hat{}\, 2{\times}(x-y) \,\hat{}\, 2$.
Observação: Para a fatoração de inteiros, podemos utilizar o ifactor (que é o "factor" para inteiros). A sintaxe é a mesma que a do **factor**, portanto fica a cargo do leitor fazer um teste do "**ifactor**".
Agora será listada um pequeno subconjunto de funções "pré-definidas" do MuPAD.

6. float(*expressão*);

Fornece uma aproximação para a expressão. Atenção, esta *expressão* deve ser um número, como por exemplo:

- x:=1/3: **float**(x); **float**(1/5); ENTER

calcula um valor aproximado para 1/3 e de 1/5 (vide sexto e sétimo itens da seção anterior).

7. for *variável* **from** *início_inteiro* **to** *fim_inteiro* **do** *comandos:* **end_for:**

Este é um comando de "programação", com ele é possível fazer coisas mais sofisticadas. O **for** é como um somatório, os *comandos:* (entre o **do** e o **end_for:**) são para: *variável=início_inteiro*; *variável=início_inteiro*+1; *variável=início_inteiro*+2 ;· · ·; *variável=fim_inteiro-1*; *variável=fim_inteiro.*
Por exemplo, para imprimirmos na tela a lista

`1 -> 1^3=1; 2 -> 2^3=8; 3 -> 3^3=27; 4 -> 4^3=64; 5 -> 5^3=125;` · · ·

digite os seguintes comandos:

- **for** n **from** 1 **to** 13 **do** <SHIFT>ENTER
- **print**(n," -> ", n,"^3 = ", n^3): <SHIFT>ENTER
- **end_for:** ENTER

8. if *condição_lógica* **then** *comandos_1*: [**else** *comandos_2*:] **end_if:**
Este é um comando especial para programação de "rotinas" mais sofisticadas, veja por exemplo uma utilização do **if** na seção **4**. Os colchetes indicam que o "**else** *comandos_2*:" é opcional. Se você desejar fazer um comando que atribui o valor 1 se uma variável n é maior que 10 e 0 se n é menor, faça o seguinte:

- x := **if** n>10 **then** 1: <SHIFT>ENTER
- **else** 0: <SHIFT>ENTER
- **end_if:** <SHIFT>ENTER
- **print**(x): #para testar o resultado - não se esqueça de definir n# ENTER

(qualquer coisa entre os símbolos # são comentários, o que significa que o MuPAD vai ignorar os comentários.)

9. ln(*expressão*);
Calcula o logaritmo natural (ou neperiano) da (expressão). Por exemplo,
• x := 3: **ln**(**exp**(x)); ENTER
calcula $\ell n(\exp(3)) = 3$.

10. _plus(*lista_de_expressões*);
Realiza a soma de todas as variáveis e expressões em *lista_de_expressões*. Exemplo: **_plus**(1/i, 0.2, 1/3, PI); devolve o valor da expressão $1/i + 0.2 + 1/3 + \pi$.

11. print(*lista_de_expressões*);
Lista a sequência *lista_de_expressões* na tela, onde *lista_de_expressões* é uma lista de expressões e palavras (entre aspas dupla), separadas por vírgulas (vide exemplos acima, em 1 e 7).

12. sqrt(*expressão*);
Calcula a raiz quadrada da *expressão* que deve ser obrigatóriamente um número (uma variável com valor previamente definido pode ser vista como um número). Por exemplo, **sqrt**(2); calcula $\sqrt{2}$.

Observação: Se for necessário obter uma aproximação, basta digitar **float**(**sqrt**(2));

13. simplify(*expressão*);
Simplifica a expressão, quer seja numérica ou não, porém seu algoritmo não é tão sofisticado quanto o **factor**. Exemplo:

- n := 2: **simplify**(((1+**sqrt**(5))/2)^(n+1) / **sqrt**(5) - ((1-**sqrt**(5))/2)^(n+1) / **sqrt**(5))

calcula um modo simples de apresentar $(\frac{1+\sqrt{5}}{2})^{n+1}\frac{1}{\sqrt{5}} - (\frac{1-\sqrt{5}}{2})^{n+1}\frac{1}{\sqrt{5}} = 2$
Observação: Para algumas simplificações envolvendo variáveis (identificadores) são necessárias algumas "suposições". Para isso podemos utilizar o comando "**assume**" como ilustrado a seguir,

- **assume** (x,**Type::RealNum**): **simplify**(**ln**(**exp**(x)));

Isso faz com que o valor de x seja real e portanto vale a identidade de $ln(e^x) = x$. Os dois pontos servem para "especificar o domínio" da variável x na expressão

14. solve({ eq_1, eq_2, ..., eq_n}, {var_1,var_2,...,var_m});
Resolve um sistema de equações, nas variáveis var_1, var_2, ..., var_m. Isto pode ser exemplificado da seguinte maneira:

- **solve**({$x \hat{} 2 + y \hat{} 2 = 1,\ a * x + b * y = 0$},{$x,\ y$});

resolve o sistema de equações:

$$\begin{cases} x^2 + y^2 = 1 \\ ax + by = 0 \end{cases}$$

15. **sum**(*expressão, variável*=$v_i..v_f$);
que calcula a soma dos valores da expressão quando a variável assume cada valor inteiro, desde v_i até v_f. Ou seja, serve para cálculos de somatórios e é matematicamente equivalente a $\sum\limits_{k=v_i}^{v_f}$ *expressão.*
<u>Observação:</u> v_i e v_f devem ser inteiros.

Vejamos um exemplo: Para calcularmos $\sum\limits_{i=1}^{10} 1/2^i$, basta digitar

- **sum**$(1/2 \,\hat{}\, i, i = 1..10)$;

Para calcular uma aproximação do somatório acima, basta digitar a seguinte instrução logo após o cálculo acima (veja mais detalhes sobre o **float** no item 5 a seguir):

- **float**(%);

Neste caso, o sinal de % significa o último resultado obtido pelo MuPAD.
Para calcular calcular $\sum\limits_{i=1}^{\infty} 1/2^i$, devemos digitar:

- **sum**$(1/2 \,\hat{}\, i, i = 1..infinity)$;

e obtemos como resultado (depois de muito esperar), o valor 1, que é o valor de $\sum\limits_{i=1}^{\infty} \dfrac{1}{2^i}$.

4 Funções definidas pelo usuário

Existem duas maneiras de se definir uma função no MuPAD.

– A primeira, é direta utilizada para funções "simples". Por exemplo, queremos definir uma função chamada **f**, através de algumas funções já conhecidas, como por exemplo **sin**(x) e **cos**(x). Desse modo, se **f** for o produto de **sin**(x) por **cos**(2x), digitamos:

- f **:=** **sin**(x)***cos**(2*x)**:**

Para calcular o valor de **f** no ponto x=8, por exemplo, devemos digitar:

- **eval**(**subs(** f, x=8**))** ;

calcula o falor de **f**, quando **x** vale 8. O **eval** é uma abreviação de *evaluation,* ou *evaluating*, enquanto o **subs** é uma abreviação de **substitution**.

– A segunda maneira é mais completa: pode-se definir funções sofisticadas utilizando comandos de programação, como ocorre em linguagens como o Pascal ou C. Por exemplo, para definir a função módulo: $\begin{cases} x & \text{se } x \geq 0 \\ -x & \text{se } x < 0 \end{cases}$, digitamos

• a:=**proc**(x) <SHIFT>ENTER	`a é nome da função e x parâmetro`
• **begin** <SHIFT>ENTER	`marca início dos comandos que definirão a função`
• **if** x<0 **then** -x <SHIFT>ENTER	`se x<0 retorna -x`
• **else** x	`caso contrário, retorna x` <SHIFT>ENTER
• **end_if:** ENTER	`marca fim do comando if`
• **end_proc:** ENTER	`marca fim dos comandos da função`

Apresentamos a seguir um exemplo (um pouco mais sofisticado), onde definimos uma *função recorrente*, que descreve o somatório de 2^k, començando com $k = 1$, isto é, na verdade a função definida a seguir será uma formalização da função

$$\sum_{k=1}^{n} 2^k.$$

Porém, para que o código desta função seja perfeitamente entendido, é necessário algum conhecimento de programação. Se não dispõe deste conhecimento, tente abstrair os detalhes de definição da função (quinta até oitava linhas).

Exemplo: Vejamos um exemplo mais sofisticado, no qual definimos a função **g** recorrente[1] como

$$g(n) = \begin{cases} 1, & \text{se } n = 1 \\ g(n-1) + 2^{n-1}, & \text{se } n \geq 2. \end{cases}$$

• g := **proc**(n) <SHIFT>ENTER	`Define a função` g `com parâmetro` n
• **local** r; <SHIFT>ENTER	`Define uma variável local de` g `(opcional)`
• **begin** <SHIFT>ENTER	`Obrigatório: inicia corpo de` g
• r:=1: <SHIFT>ENTER	`''Inicializa'' variável` r
• **if** n>1 then <SHIFT>ENTER	`Se` n≤2 `então` g(n) `será 1`
• **for** k **from** 1 **to** n-1 **do** <SHIFT>ENTER	`Laço:` k `varia desde de` 1 `até o valor de` n-1
• r:=r+2^(k) <SHIFT>ENTER	`Acumula` 2^k `na variável auxiliar` r
• **end_for:** <SHIFT>ENTER	`Obrigatório: finaliza comando` **if**
• **end_if:** <SHIFT>ENTER	`Obrigatório: finaliza laço` **for**
• r: <SHIFT>ENTER	`É o valor que ''retornado'' por` g [g(n)]
• **end_proc:** <SHIFT>ENTER	`Obrigatório: finaliza comando` **proc**
• **for** k **from** 1 **to** 10 **do** <SHIFT>ENTER	
• **print**("g(",k,")=", g(k)) <SHIFT>ENTER	
• **end_for:**	

5 Gráficos

Um outro tipo de recurso muito útil do MuPAD (e assemelhados) é permitir visualização de funções de $I\!R$ em $I\!R^2$ (**plot2d**) e em $I\!R^3$ (**plot3d**).

Vamos examinar as duas funções disponíveis para gráficos (muito cuidado com as letras maiúsculas e minúsculas - o que está em negrito deve ser digitado exatamente como indicado),

- **plot2d**([**Mode**=**Curve**, [u, u * **sin**(u)], u = [0,2*PI], **Grid** = [50]]);

– O **plot2d** serve para fazer gráfico de funções ou curvas parametrizadas, note que aparece [u, u***sin**(u)]: dentro do colchete, o primeiro parâmetro (u) significa variação na abscissa e o segundo, variação na ordenada (u***cos**(u)). O intervalo da abscissa onde o gráfico será definido é dado pelo comando obrigatório u=[0, 2*PI], isto é, u começa valendo 0 e "vai" até 2π, segundo o passo que é definido pelo comprimento do intervalo dividido pelo valor que está no **Grid** (o terceiro parâmetro no primeiro conjunto de colchetes).

– O que está dentro do primeiro conjunto de colchetes, indica o modo em que o gráfico será descrito (no caso será através de uma curva). A variação da coordenada "x" (abscissa) está no terceiro par de colchetes. O último "parâmetro" do primeiro par de colchetes (**Grid**) é o "passo" com que o MuPAD irá "exibir o gráfico".

- **plot3d**(**Axes**=**Box**, **Ticks**=0, **TitlePosition** =**Below**, [**Mode** = **Curve**, [u, u, **sin**(u*PI)], u = [−3.0, 3.0], **Grid** = [50], **Style** = [**Impulses**]]);

[1] Um exemplo clássico de função recorrente é o fatorial, que é conhecida pela simbologia $f(n) = n!$: $f(0) = 1$ e $f(n) = n \times f(n-1)$, para $n \geq 1$. Veja o item **4.** da seção **3**.

- **plot3d**([**Mode** = **Surface**, $[u, v, 1/2 * \sin(u * u + v * v)]$, u = [0, PI], v = [0, PI],
  Grid = [30, 30], **Style** = [**HiddenLine**, **Mesh**]]);

Os mesmos comentários servem para o plot3d, com alguns adendos:

– O **plot3d** tem dois tipos de gráficos o de curvas e o de superfícies. O parâmetro **Style**, serve para "dizer ao MuPAD, que tipo de gráfico será exibido, isto é, com que tipo de linhas, se é preenchido ou não.

6 Matrizes

Quando for necessário utilizar matrizes, devemos "avisar" o MuPAD. Para isso usamos a seguinte instrução:

- **export**(**Dom**): **export**(**linalg**): ENTER

isto deve ser feito apenas uma vez na área de trabalho, por que uma vez "exportado" o pacote de Álgebra Linear (**linalg**) ele estará sempre ativo (faça o seguinte teste: digite a linha acima de dê um ENTER, volte com o cursor até essa mesma linha e dê novo ENTER).

Agora é necessário criar um "definidor" de matrizes:

- Mat:=**Matrix**(**ExpressionField**(**normal**)) : ENTER

o que define uma matriz genérica cujas entradas serão expressões algébricas. Vamos examinar alguns exemplos (digite antes as duas linhas acima):

- m:=Mat([[a, b],[$1, a * b$]]): ENTER

Aqui foi definida a matriz m = $\begin{bmatrix} a & b \\ 1 & a * b \end{bmatrix}$.

- 1/m; m^(-1); ENTER

Aqui foi pedido para calcular a inversa da matriz **m**, de duas formas equivalentes.

Vamos agora definir duas matrizes A e B, após o que calcularemos A×B×B×A^(-1),

- A:=Mat([[1, 2], [4, -10]]): <SHIFT>ENTER
- B:=Mat([[2, 0], [0, 4]]): <SHIFT>ENTER
- C:=A×B×B×A^(-1): <SHIFT>ENTER
- **print**(A, B, C): ENTER

Por enquanto é só, isto foi só uma leve "pincelada" sobre o que o MuPAD pode fazer. Descubra novas e úteis maneira de usar o MuPAD e divirta-se!!