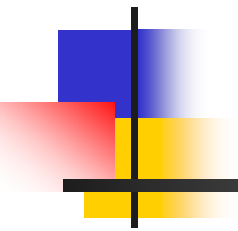
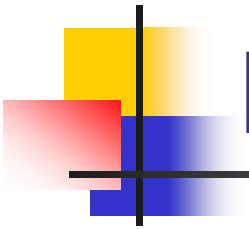


MATLAB

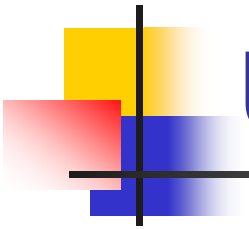
André Marques Saunite
Fabricio Aparecido Breve

Prof. Dr. Antonio Francisco do Prado



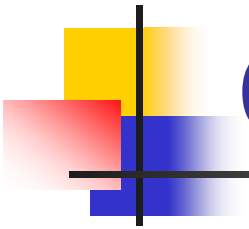
Histórico

- Ambiente interativo para computação envolvendo matrizes
- Desenvolvido no início da década de 80 por Cleve Moler, no Departamento de Ciência da Computação da Universidade do Novo México, EUA
- Versões mais recentes (4.0 em diante) foram desenvolvidas pela MathWorks Inc., que detêm os direitos autorais destas implementações
- Matlab = Matrix Laboratory
- Multiplataforma



Uso do MATLAB

- Matemática e Computação
- Desenvolvimento de Algoritmos
- Aquisição de Dados
- Modelagem, Simulação e Prototipação
- Análise, Exploração e Visualização de Dados
- Gráficos Científicos e de Engenharia
- Desenvolvimento de Aplicativos, incluindo desenvolvimento de interface gráfica



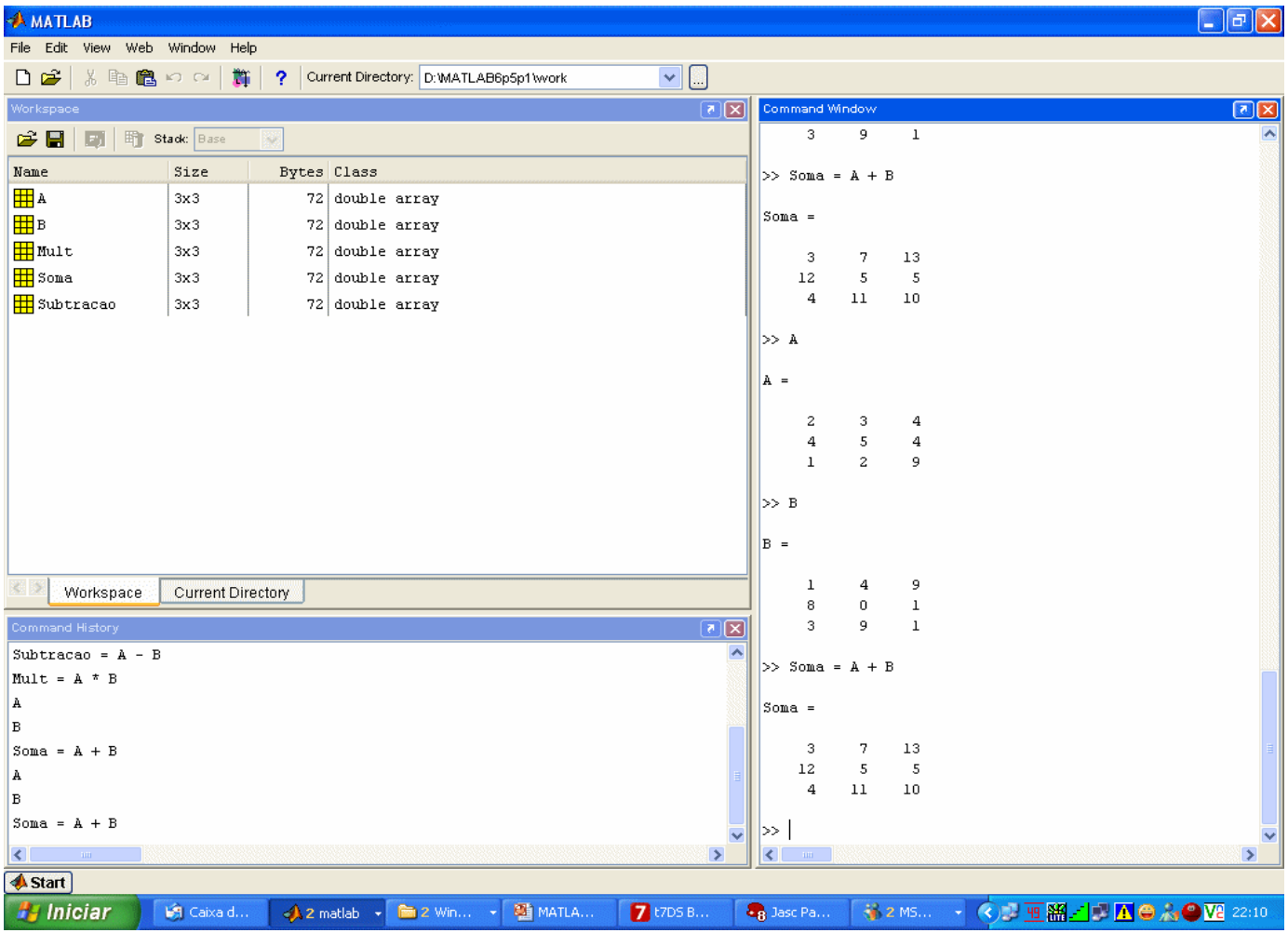
Características

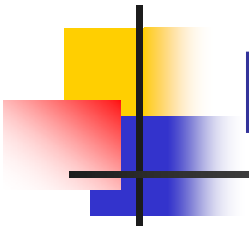
- É um sistema interativo cujo elemento de dado básico é um array que não requer dimensionamento. Isto permite resolver muitos problemas técnicos computacionais, especialmente aqueles com formulação de matrizes e vetores, em uma fração de tempo que levaria para escrever um programa em uma linguagem escalar não interativa (como C ou Fortran)
- Tem diversos toolboxes para aplicações específicas, incluindo:
  - Processamento de Sinais
  - Sistemas de Controle
  - Redes Neurais
  - Lógica Fuzzy
  - Wavelets
  - Simulação

Sistema Matlab

- **Ambiente de Desenvolvimento:** ferramentas que facilitam o uso do MATLAB, incluindo janela de comandos, histórico, área de trabalho, arquivos, etc.
- **Biblioteca de Funções Matemáticas:** coleção de algoritmos computacionais, variando de funções elementares como soma, seno, cosseno e aritmética complexa até funções mais sofisticadas como matriz inversa, funções de Bessel e Transformada Rápida de Fourier.
- **Linguagem Matlab:** linguagem de vetores/matriz de alto nível com controle de fluxo, funções, estrutura de dados, entrada/saída, e características de programação orientada a objetos.
- **Gráficos:** facilidades para mostrar vetores e matrizes como gráficos, e também anotações e impressões destes gráficos. Funções de alto nível para visualização de dados bidimensionais e tridimensionais, processamento de imagens, animação e apresentação de gráficos. Inclui também funções de baixo nível para personalizar a aparência de gráficos e para implementar interface gráfica para aplicações.
- **MATLAB API:** biblioteca que permite escrever programas em C e Fortran para interagir com o MATLAB. Inclui facilidades de chamar rotinas do MATLAB (elo dinâmico), chamada do MATLAB como motor computacional e leitura e escrita de arquivos MAT.

Ambiente de Desenvolvimento

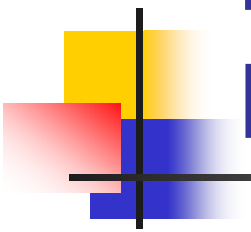




Linguagem Matlab

- Versões da Mathworks incluem facilidades gráficas, de visualização e impressão
- Entretanto existem interpretadores da “linguagem Matlab” em domínio público:
  - MATLAB 1.0
  - Octave
  - Rlab
- Existem ainda outros interpretadores comerciais de Matlab

Biblioteca de Funções Matemáticas



- O MATLAB incorpora diversas toolboxes com funções genéricas e específicas para diversas áreas
- Apesar de o MATLAB ser um software proprietário, os códigos-fonte das toolboxes podem ser livremente visualizados e editados.
- Existem diversas toolboxes de terceiros, tanto comerciais quanto livres.

Executando Funções e Iniciando Variáveis

- Declarando uma Matriz de 3X3 elementos:
  A = [1 2 3; 4 5 6; 7 8 10]

  Ao pressionar ENTER o MatLab responde com:

  A =

1	2	3
4	5	6
7	8	10

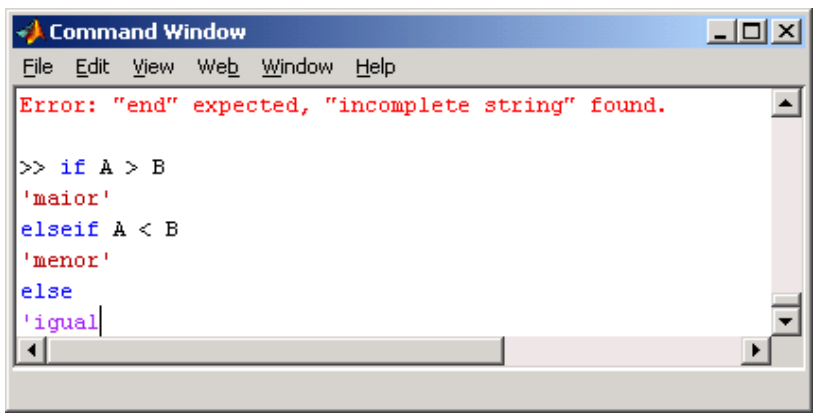
  Para executar uma função, digite a função incluindo seus argumentos e simplesmente pressione ENTER e o MATLAB exibe o resultado. Por exemplo, digite magic(3) e o MATLAB retorna

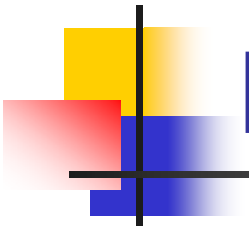
  ans =

8	1	6
3	5	7
4	9	2

Executando Funções e Iniciando Variáveis

- Para digitar múltiplas linhas antes de executá-las basta digitar a linha e pressionar SHIFT+ENTER.
- Para que o MATLAB execute um comando sem exibir resultado, basta colocar um ponto-e-vírgula no final do comando.
- As funções e variáveis são case-sensitive
- Syntax Highlighting e Parentheses Matching

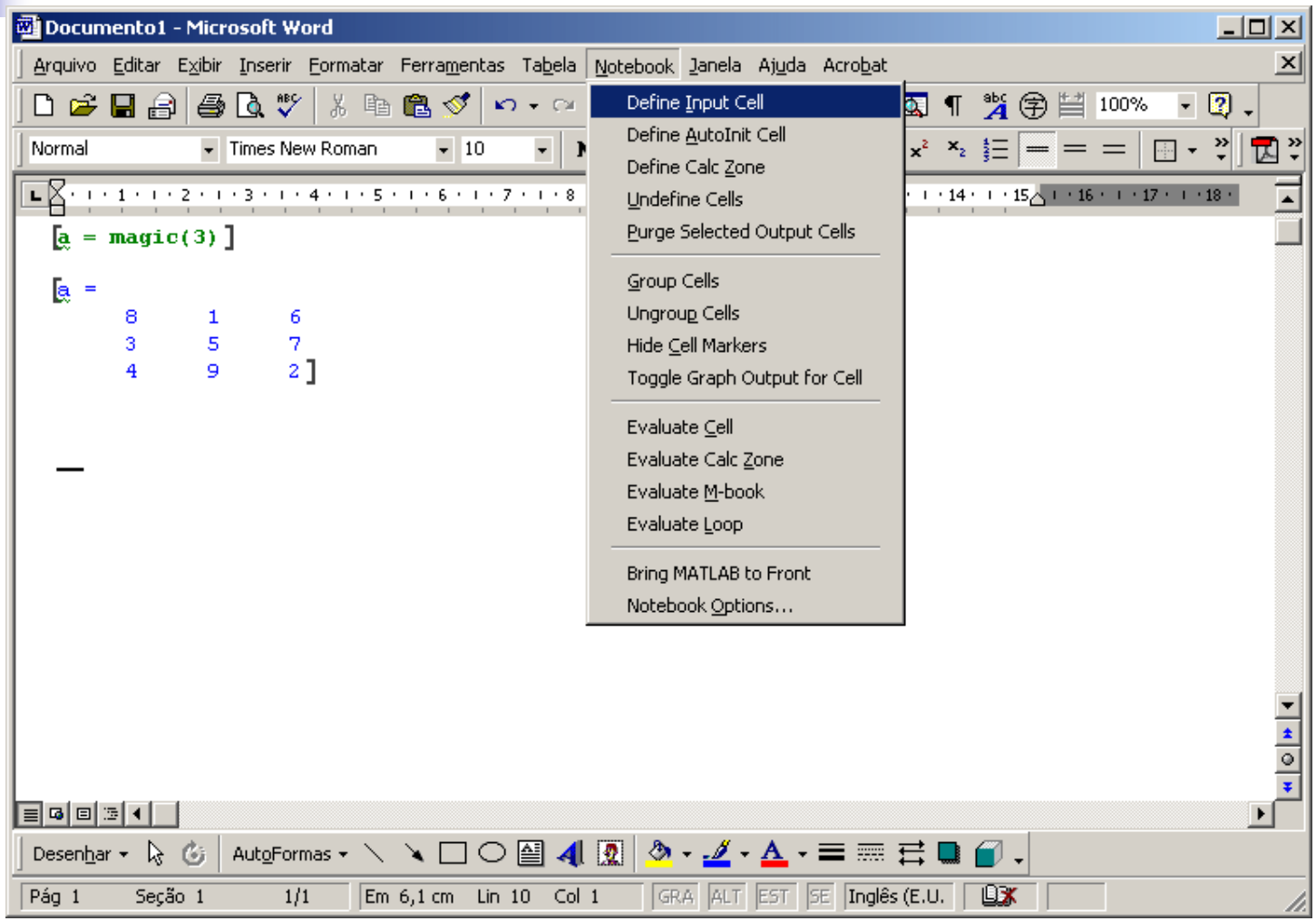




Funcionalidades

- Suporte para importar e exportar arquivos texto e binários (incluindo imagens, sons e vídeo)
- Funções de I/O de baixo nível análogas as funções da linguagem C
- Editor/Depurador incluído

Criando Documentação



Operações com Matrizes

```
A = [1 2 3; 4 5 6; 7 8 9];
B = [2 1 2; 4 1 7; 2 5 3];
X = A + B

X =
     3     3     5
     8     6    13
     9    13    12
```

Operadores:

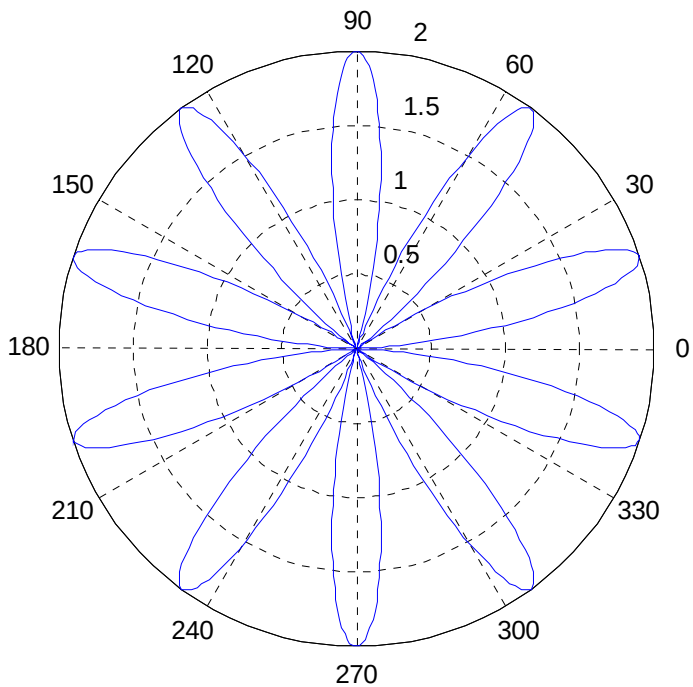
+	Adição
-	Subtração
*	Multiplicação
/	Divisão à esquerda
\	Divisão à direita
^	Potência
'	Transposição

Programação no Matlab

- Scripts
  - Não aceita argumentos de entrada ou de saída
  - Opera nos dados da área de trabalho
  - Útil para automatizar uma série de passos que você precisa programar muitas vezes
- Funções
  - Pode aceitar argumentos de entrada e de saída
  - Variáveis internas são locais da função por padrão
  - Útil para extender a linguagem do Matlab para a sua aplicação

Um Script Matlab

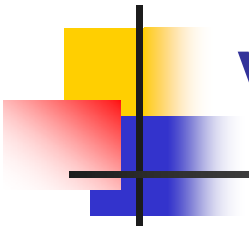
```
% Um script que imprime petalas de flores
theta = -pi:0.01:pi;          % Computaçao
rho(1,:) = 2*sin(5*theta).^2;
rho(2,:) = cos(10*theta).^3;
rho(3,:) = sin(theta).^2;
rho(4,:) = 5*cos(3.5*theta).^3;
for k = 1:4
    polar(theta,rho(k,:))     % Saida Grafica
    pause
end
```



Uma Função no Matlab

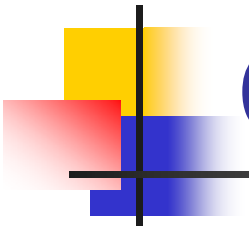
```
function y = media(x)
% MEDIA dos elementos de um vetor.
% MEDIA(X), onde X é um vetor.
% Se X não é vetor, retorna um erro.
[m,n] = size(x);
if (~((m == 1) | (n == 1)) | (m == 1 & n == 1))
    error('A entrada precisa ser um vetor')
end
y = sum(x)/length(x);      % Calculo Efetivo
```

```
Resultado:
>> A = [0 5 8 10 20];
>> media(A)

ans =

    8.6000
```



Variáveis

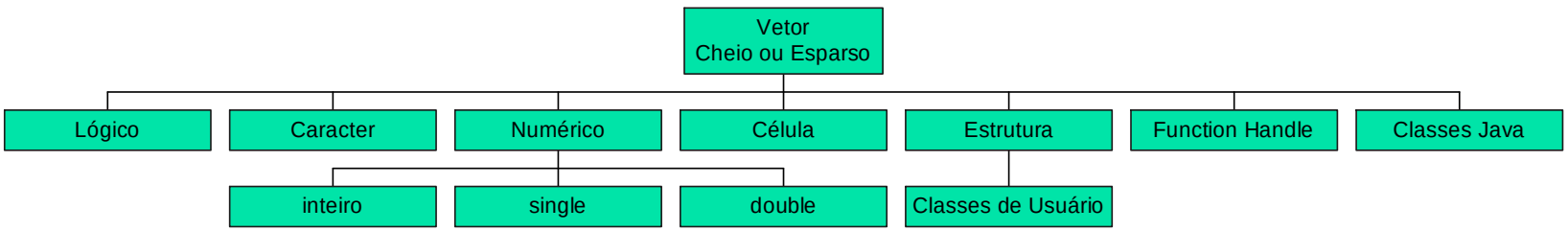
- Variáveis no Matlab não precisam ser declaradas, com exceção das variáveis globais ou persistentes
- **Variáveis Globais:** Se várias funções declaram uma variável como global, a mesma variável será usada por todas essas funções
- **Variáveis Persistentes:** só podem ser utilizadas em funções e somente a função que a criou tem acesso a ela. Elas não são apagadas quando a função termina, e seu valor é mantido para a próxima chamada da mesma função



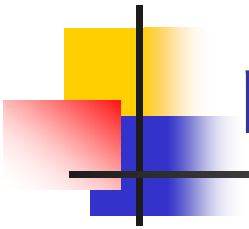
Controle de Fluxo

- if, junto com else e elseif, executa um grupo de comandos baseados em uma condição lógica.
- switch, junto com case e otherwise, executa diferentes grupos de comandos dependendo do valor de alguma condição lógica.
- while executa um grupo de comandos um número indefinido de vezes, baseado em alguma condição lógica.
- for executa um grupo de comandos por um número fixo de vezes.
- continue passa o controle para a próxima iteração de um loop while, omitindo qualquer comando remanescente no restante do loop.
- break termina a execução de um loop for ou while.
- try...catch muda o fluxo de controle se um erro é detectado durante a execução.
- return faz a execução retornar para a função que invocou.

Tipos de dados



Acessando elementos de uma matriz



- Soma = A(1,4) + A(2,4) + A(3,4) + A(4,4)
- Soma = sum(A(1:4,4))
- Soma = sum(A(:,4))
- Soma = sum(A(:,end))

Expandindo o tamanho de uma matriz

- Se você tentar acessar um elemento fora da matriz, receberá uma mensagem de erro:

```
B = A(4,5)
Index exceeds matrix dimensions
```

- Se você tentar armazenar um elemento fora da matriz, a matriz se expande para acomodar o elemento:

```
B = A;
B(4,5) = 17
B =
     16     2     3    13     0
      5    11    10     8     0
      9     7     6    12     0
      4    14    15     1    17
```

Excluindo linhas e colunas

- Você pode excluir linhas de uma matriz usando apenas um par de colchetes. Começando com:

```
X = A;
```

- Para deletar a segunda coluna de X usa-se:

```
X(:,2) = []
```

- Isto muda X para:

```
X =
    16     3    13
     5    10     8
     9     6    12
     4    15     1
```

- Se você tentar excluir o único elemento de uma matriz, ela deixará de ser uma matriz, portanto essa operação não é permitida e retorna um erro.

Matrizes vazias

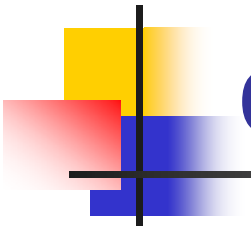
```
>> A = zeros(3)

A =

     0     0     0
     0     0     0
     0     0     0
```

```
>> B = ones(3)

B =

     1     1     1
     1     1     1
     1     1     1
```

```
>> C = rand(3)

C =

    0.9501    0.4860    0.4565
    0.2311    0.8913    0.0185
    0.6068    0.7621    0.8214
```

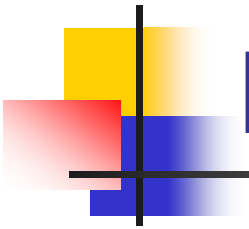
```
>> D = eye(3)

D =

     1     0     0
     0     1     0
     0     0     1
```

Programação orientada a objetos

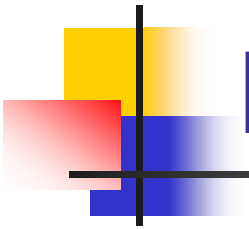


- **Sobrecarga de operadores e funções:** você pode criar métodos que sobrecarregam funções existentes do MATLAB. Quando você chama uma função com um objeto definido por usuário como argumento, o MATLAB primeiro verifica se há um método definido para o objeto na classe. Se houver o MATLAB o chama, em vez de chamar a função normal do MATLAB.
- **Encapsulamento de dados e métodos:** as propriedades dos objetos não são visíveis pela linha de comando; você pode acessálos somente com métodos da classe. Isto protege as propriedades dos objetos de operações que não são permitidas pela classe do objeto.
- **Herança:** você pode criar hierarquia de classes pais e filhos na qual a classe filho herda os métodos do pai. A classe filho pode herdar de um pai (herança simples) ou muitos pais (herança múltipla). A herança pode se expandir para a ou mais gerações. A herança habilita o compartilhamento de funções comuns dos pais e força um comportamento comum entre todas as classes filho.
- **Agregação:** você pode criar classes usando agregação, na qual um objeto contem outros objetos. Isto é apropriado quando um tipo de objeto é parte de outro tipo. Por exemplo, um objeto conta pode ser parte de um objeto banco.

Ordem de Precedência de Funções



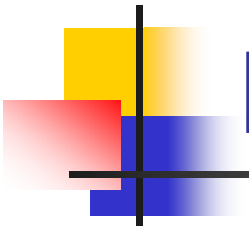
- Dois tipos de função: internas ou externas
- **Sobrecarga de método:** se há um método criado com o mesmo nome de uma função interna, então este método é chamado em vez da função interna (isto não se aplica para subfunções e funções privadas)
- Um arquivo pode conter mais de uma função, a primeira função é a função principal e as outras são subfunções, que só são visíveis para a função principal ou outras subfunções.
- Subfunções tem precedência sobre todas as outras funções, mesmo que uma função seja chamada com argumento de tipo igual a de um método sobrecarregado, ainda assim o Matlab usa a subfunção e ignora o método sobrecarregado



Funções Privadas

- Funções Privadas são funções que residem em um subdiretório com o nome “private”. Elas são visíveis apenas para as funções no diretório pai.
- Se uma função privada tem o mesmo nome de uma outra função, a função privada tem precedência, mesmo que a outra função tenha argumentos iguais aos da função chamada, o Matlab ignora a função sobrecarrega e chama a função privada.

Resolução de nomes no Matlab

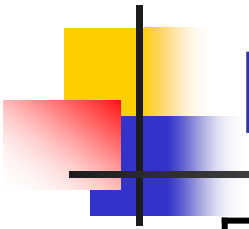


1. Verifica se o nome é uma variável
2. Verifica se o nome é uma subfunção (uma função que reside no mesmo arquivo da função chamada)
3. Verifica se o nome é uma função privada que reside em um diretório privado, que é um diretório acessível apenas para funções e scripts no diretório imediatamente acima dele.
4. Verifica se o nome é uma função no PATH do MATLAB. O MATLAB utiliza o primeiro arquivo que encontrar com o nome procurado

Se você duplicar um nome de função, o MATLAB executa a primeira encontrada segundo as regras acima. Também é possível sobrecarregar nomes de funções.

Diferenças entre Matlab, Java e C++ na Orientação a Objeto

- No MATLAB, chamada de métodos não são baseados em sintaxe, como em C++ ou Java. Quando a lista de argumentos contém objetos de igual precedência o MATLAB usa o objeto mais a esquerda para selecionar qual método chamar.
- No MATLAB não há nada equivalente a um método destrutor. Para remover um objeto é necessário usar a função clear.
- A construção dos tipos de dados do MATLAB acontecem em tempo de execução ao invés de tempo de compilação. Você registra um objeto como pertencente a uma classe chamando essa classe.
- No uso de herança no MATLAB, a relação de herança é estabelecida na classe filho através da criação do objeto pai, e então chamando a função da classe.
- No uso de herança no MATLAB, o objeto filho contém o objeto pai em uma propriedade com o nome da classe pai.
- No MATLAB não há passagem de variáveis por referência, Quando for escrever métodos que atualizam um objeto você precisa passar de volta o objeto atualizado e usar um comando de atribuição. Essa chamada para o método atualiza o campo nome de um objeto A e retorna o objeto atualizado.



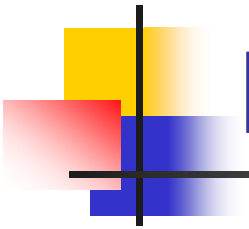
Métodos Padrão

construtor	Criar um objeto da classe
display	Chamado sempre que o MATLAB exibe o conteúdo de um objeto (exemplo: quando uma função é chamada sem o ponto-e-vírgula)
set e get	Acessa as propriedades da classe
subsref e subsasng	Habilita referência e atribuição indexada para objetos de usuário
end	Suporta a expressão end em expressões indexadas
subsindex	Suporta o uso de um objeto para indexar expressões
conversores como double e char	Métodos para converter um objeto para um tipo do MATLAB

Exemplo de Construtor

Aqui está um construtor para a classe polinomio, em @polinomio/polinomio.m

```
function p = polinomio(a)
%POLINOMIO Classe construtor do Polinômio.
%   p = POLINOMIO(v) cria um objeto polinômio a partir do vetor v
%   contendo os coeficientes das potências descendentes de x.
if nargin == 0  % construtor vazio
   p.c = [];
   p = class(p,'polinomio');
elseif isa(a,'polinomio')   % construtor de cópia
   p = a;
else % construtor com argumentos
   p.c = a(:).';
   p = class(p,'polinomio');
end
```



Método Display

Método display em @polinomio/display.m

```
function display(p)
disp(' ');
disp([inputname(1),' = '])
disp(' ');
disp(['   ' char(p)])
disp(' ');
```

```
>> p = polynom([1 0 -2 -5])

p =
    x^3 - 2*x - 5
```

Sobrecarga de Operadores e Funções

`p + q` gera uma chamada a função @polinomio/plus.m se esta existir

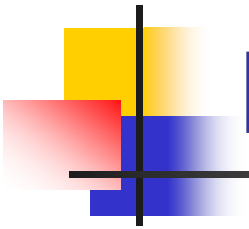
```
function r = plus(p,q)
% Implementa p + q para polinomios.
p = polinomio(p);
q = polinomio(q);
k = length(q.c) - length(p.c);
r = polinomio([zeros(1,k) p.c] + [zeros(1,-k) q.c]);
```

Para sobrecarregar outros operadores é necessário saber o nome do operador:

p – q @polinomio/minus.m
p * q @polinomio/mtimes.m

Para funções sobrecarregamos o nome da função:

@polinomio/roots.m
@polinomio/plot.m



Herança

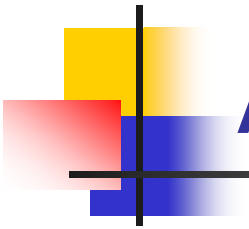
- Herança Simples:

```
child_obj = class(child_obj,'child_class',parent_obj);
```

- Herança Múltipla:

```
obj = class(structure,'class_name',parent1,parent2,...)
```

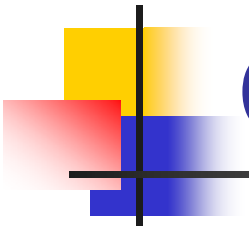
- As propriedades herdadas da classe pai só podem ser acessadas por métodos da classe pai
- Se houverem métodos de mesmo nome em mais de um pai, será chamado o método do pai que estiver primeiro na lista de chamada do construtor, e não há forma de acessar os métodos de mesmo nome dos outros pais
- A herança pode se expandir por mais de uma geração, a classe filho pode herdar métodos e propriedades do pai, avô, etc.



Agregação

- Além da herança padrão o Matlab também suporta agregação, isto é, um objeto pode conter um outro objeto como um de seus campos
- Você pode chamar um método para o objeto agregado somente através de um método do objeto externo. Quando determinando qual versão da função chamar, o MATLAB considera somente a classe mais externas dos objetos passados como argumentos, as classes dos objetos agregados são ignoradas.

Salvando e Carregando Objetos



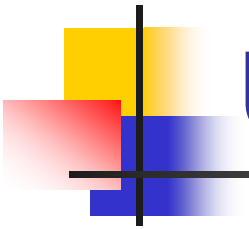
- O Matlab possui as funções save e load que permitem salvar e carregar objetos em arquivos .mat (da mesma forma que qualquer outra variável)
- Ao utilizar a função load com objetos, o Matlab chama o construtor da classe, que deve ser capaz de ser chamado sem argumentos
- Ao chamar save e load, o Matlab procura na classe por objetos chamados saveobj e loadobj respectivamente, estes métodos podem ser sobrecarregados para modificar o objeto antes de salvá-lo ou carregá-lo

Precedência de Objetos

- A precedência de objeto é um meio de resolver a questão de qual versão de um operador ou função chamar em uma dada situação.
- A precedência de objeto permite que você controle o comportamento de expressões contendo diferentes classes de objetos. Por exemplo:

  objetoA + objetoB

- Normalmente o MATLAB assume que os objetos tem igual precedência e chamam o método associado ao objeto mais a esquerda. Porém há duas exceções:
  - Classes definidas por usuário tem precedência sobre as classes internas do Matlab
  - Classes definidas por usuário podem especificar sua precedência em relação a outra(s) usando as funções inferiorto e superiorto
    - `inferiorto('class1','class2',...)`
    - `superiorto('class1','class2',...)`



Uso eficiente da memória

- Alguns comandos:
  - whos – mostra quanto da memória está alocado para variáveis e objetos
  - pack – salva todas as variáveis e objetos para a memória e os recarrega de forma contínua, eliminando a fragmentação
  - clear – limpa variáveis da memória
  - save/load – para armazenar em disco variáveis que não estão sendo utilizadas no momento
  - quit – encerra o MATLAB e devolve a memória alocada para o sistema (útil em sistemas Unix que não liberam a memória reservada a um aplicativo quando ele é encerrado)