INSTITUTO FEDERAL
Mato Grosso
Campus Cuiabá
Cel. Octayde
Jorge da Silva

Departamento da Área de Informática
Curso: Bacharelado em Engenharia da Computação **Semestre:** 9
Curso: Bacharelado em Engenharia de Controle e Automação **Semestre:** Optativa
Disciplina: Processamento Digital de Imagens.
Professor: Esp. Giuliano Robledo Zucoloto Moreira.

Cuiabá-MT, 28 de julho de 2022.

NOTA EXPLICATIVA:
GNU Octave básico para a disciplina de PDI

Está fora do escopo desta nota esgotar o polimorfismo e herança de cada comando utilizado bem como todos os itens da ementa. Nela são apresentados os passos iniciais necessários ao caminho do estudo da disciplina de processamento digital de imagens (PDI). A nota muitas vezes pode ser objetiva pois considera-se que o estudante tenha conhecimentos prévios de programação dada a evolução no curso até à disciplina de PDI. O *software* utilizado no desenvolvimento dos exemplos foi o *GNU Octave* (modo gráfico) versão 6.1.0. **Grande parte deste material está baseada na documentação oficial das das versões 4.2.1 [1] e 6.1.0 [2] do *software* que, na período desta redação, foi encontrada no endereço eletrônico:** `https://octave.org/doc`.

1 Antes de iniciar é necessário configurar!

1.1 Exibição de caracteres

Caso utilize o sistema operacional *Windows* e tenha problemas com a correta exibição de caracteres tanto em mensagens na **Janela de Comandos** quanto em elementos gráficos como títulos de imagens, rótulos de eixos etc continue a leitura deste tópico, caso contrário, prossiga para o próximo tópico.

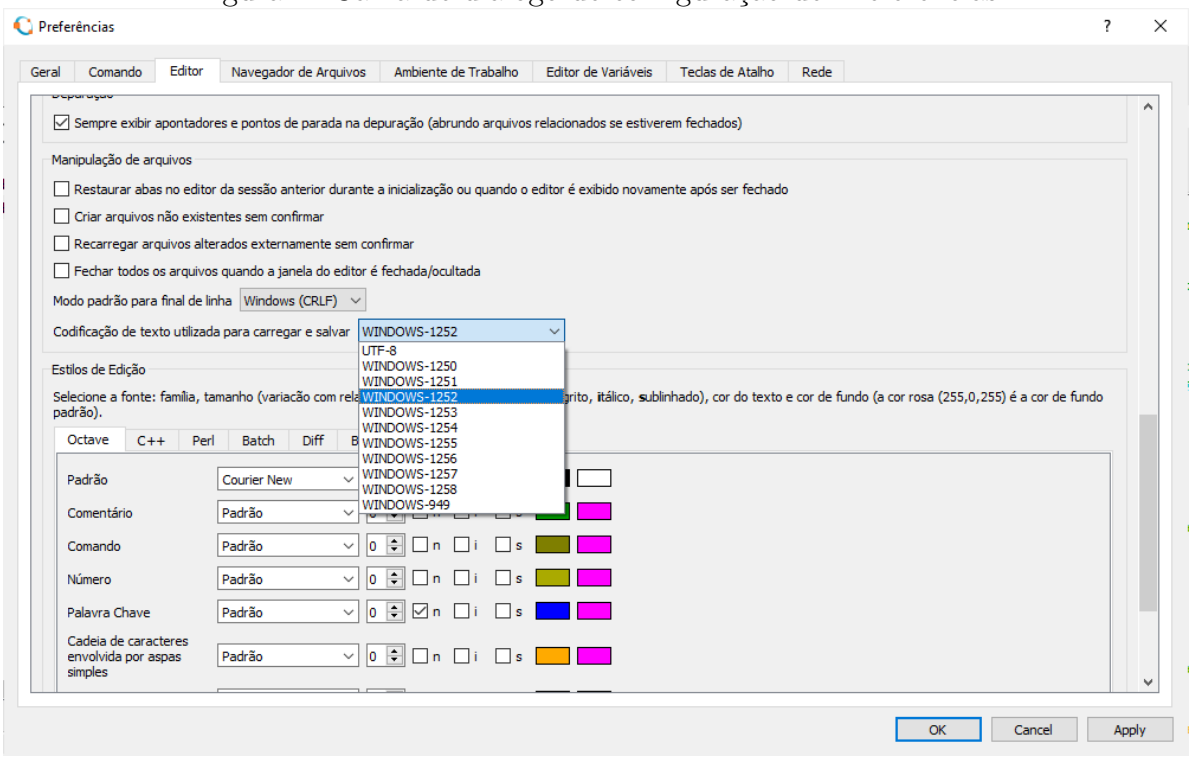
Segue em destaque o passo-a-passo anotado do procedimento de configuração da codificação de texto:

Procedimento de configuração da codificação de texto
Menu Editar → Preferências → Aba Editor → opção "Codificação de texto utilizada para carregar e salvar"→ Selecionar "Windows-1252", aplicar e confirmar a configuração (botão Ok).

Fonte: Do Autor (2021).

A Figura 1 apresenta a caixa de diálogo de configuração das preferências, onde é possível observar o detalhe da seleção da codificação "Windows-1252".

Figura 1: Caixa de diálogo de configuração de Preferências



Fonte: Do Autor (2021).

1.1.1 Quebras de linhas para conforto visual(opcional)

Esta configuração pode ser realizada para oferecer conforto visual na redação dos códigos-fonte dos *scripts* no Editor. Segue em destaque o passo-a-passo anotado do procedimento de configuração da quebra de linhas:

Procedimento de configuração da quebra das linhas

Menu Editar → Preferências → Aba Editor → Selecionar "Quebrar linhas longas na borda da janela atual", aplicar e confirmar a configuração (botão Ok).

Fonte: Do Autor (2021).

2 Primeiros passos

O objetivo desta seção é apresentar diversos códigos-fonte de *scripts* da linguagem *m* do GNU Octave versão 6.1.0 para demonstrar a aplicação de alguns comandos. Os códigos-fonte em si não tem um objetivo de realizar uma tarefa específica, são apenas aglutinados de exemplos de comandos.

2.1 *Script* 01: comandos básicos

Segue o código-fonte do *script* 01:

Código-fonte do *script* 01

```
01. clc;
02. clear all;
03. clf;
04. disp("Escrevendo uma mensagem na Janela de Comandos.\n");
05. disp("O valor da variável v é: ");
06. v = 10;
07. disp(v);
08. do
09.  t = input('\nInforme o valor inteiro de tempo em segundos a esperar e pressione a tecla enter: ');
10. until (isnumeric(t))
11.  printf('Aguardando %i segundo(s) para dar continuidade ao processamento.',t);
12. pause(t);
13. printf('\n\n\nImpressão de mensagem e variável intercaladas:\n\nO valor da variável v é %i.\n',v);
14. x = 15.123456789;
15. y = 20.123456789*10^35;
16. printf("\n\n\nMensagem com valores de variáveis recebendo formatação na impressão:\n\n\nO valor da variável v é: %05i, da variável x: é %04.3f e da variável y é: %02.15e.",v,x,y);
```

Fonte: Do Autor (2021).

Linha de comando 01: Utilizamos o comando **clc** para limpar a Janela de Comandos.

Linha de comando 02: Utilizamos o comando **clear all** para limpar as variáveis locais e globais definidas pelo usuário.

Linha de comando 03: Utilizamos o comando **clf** para limpar o conteúdo da janela de figura ativa. Se a janela de figura ainda não existir é criada a primeira janela.

Linha de comando 04: Utilizamos o comando **clear all** para exibir uma mensagem na Janela de Comandos;

Linha de comando 05: Utilizamos o comando **clear all** para exibir uma mensagem na Janela de Comandos da mesma forma da linha de comando 04;

Linha de comando 06: Utilizamos o comando **clear all** para atribuir o valo 10 (dez) à variável v;

Linha de comando 07: Utilizamos o comando **clear all** para exibir o valor da variável v na Janela de Comandos;

Linhas de comandos 08 a 10: estas linhas de comandos formam um laço de repetição que força o usuário a informar um valor numérico de tempo. O laço é formado pelas linhas 08 e 10, respectivamente os comandos **do** e **until**. Este laço executa o(s) comando(s) pelo menos uma vez antes de testar a condição de repetição. Utilizamos o comando **input** para exibir a mensagem na Janela de comandos, coletar o valor informado pelo usuário e armazená-lo na variável (t);

Linha de comando 11: Utilizamos o comando **printf** de forma semelhante ao comando **disp**, para exibir mensagem ao usuário na Janela de Comandos, contudo o comando **printf** apresenta algumas vantagens que serão apresentadas em outras linhas de comando deste *script*.

Linha de comando 12: Utilizamos o comando **pause** para gerar um intervalo de tempo na execução do programa;

Linha de comando 13: Já explicado o **printf**, esta linha apresenta uma de suas vantagens, que é poder intercalar junto à mensagem os valores de variáveis. A intercalação ocorre por meio da inserção de um símbolo de porcentagem (%) justaposto ao tipo de variável que será impressa na posição da mensagem onde ocorreu a inserção. Neste caso a variável é do tipo inteiro (%i). Após as inserções encerra-se a redação da mensagem e o próximo parâmetro a ser informado após a vírgula é a variável (v) que será apresentada na inserção (%i).

É possível exibir outros formatos numéricos:

Números inteiros sem sinal

%o → base octal
%u → base decimal
%x → base hexadecimal
%c → código do caractere

Fonte: Do Autor (2021).

Números de ponto flutuante

%f → notação de ponto fixo
%e → notação de exponencial
%g → apresenta o número em notação de ponto fixo ou notação exponencial de acordo com a melhor forma de exibição para a magnitude do número

Fonte: Do Autor (2021).

Linha de comando 14: Atribui um valor numérico à variável x;

Linha de comando 15: Atribui um valor numérico em notação exponencial à variável y;

Linha de comando 16: Já explicado o **printf**, esta linha apresenta, além da vantagem de poder intercalar junto à mensagem os valores de variáveis, a vantagem de permitir a formatação dos números a serem exibidos na mensagem. Em destaque os detalhes da formatação numérica:

Detalhes da formatação numérica

%05i → se a parte inteira do número contiver menos de cinco dígitos este será preenchido com zeros à esquerda na impressão;
%04.3f → se a parte inteira do número contiver menos de quatro dígitos este será preenchido com zeros à esquerda e o número de dígitos de casas decimais será de três na impressão;
%02.15e → se a parte inteira do número contiver menos de dois dígitos este será preenchido com zeros à esquerda e o número de dígitos de casas decimais será de quinze na impressão.

Fonte: Do Autor (2021).

2.2 *Script* 02: comandos básicos

Segue o código-fonte do *script* 02:

Código-fonte do *script* 02

```
01. m = [1 2 3 4 5; 6 7 8 9 10; 11 12 13 14 15]
02. n = m(1:2,1:4)
03. printf('\nObserve que a matriz n é o resultado do comando n = m(1:2,1:4). Compare as matrizes m no intervalo e n.\n\n');
04. m0 = zeros(5,5)
05. m0(2:4,2:4) = m (1:3,1:3)
06. printf("\nObserve que a matriz m0 foi atualizada pelo comando m0(2:4,2:4) = m (1:3,1:3), que faz uma cópia de um intervalo de nove elementos de m para m0.\n");
```

Fonte: Do Autor (2021).

Linha de comando 01: Cria uma matriz m formada por quinze elementos distribuídos em três linhas com cinco colunas cada.

Linha de comando 02: Cria uma matriz n formada por um intervalo de oito elementos da matriz m. **Compreender de forma plena a manipulação dos índices de matrizes é de suma importância para o desenvolvimento do aprendizado na disciplina**. A matriz n foi formada a partir de elementos de m como já informado, mas quais elementos? Os oito elementos localizados nas quatro colunas das duas primeiras linhas! Veja o detalhamento da manipulação dos índices da matriz m:

Detalhamento da manipulação dos índices da matriz m

m(1:2,1:4);
O **primeiro índice** da matriz m se refere às **linhas da matriz**, neste caso foi informado um intervalo 1 : 2, onde o caractere dois pontos (:) significa até, resultando na referência às linhas 1 até 2 da matriz m;
O **segundo índice** da matriz m se refere às **colunas da matriz**, neste caso foi informado um intervalo 1 : 4, onde o caractere dois pontos (:) também significa até, resultando na referência às colunas 1 até 4 da matriz m;
podemos fazer a leitura simples desta referência à matriz m como sendo: a seleção do intervalo que envolve as quatro colunas das duas primeiras linhas da matriz;
ATENÇÃO! os intervalos informados devem estar contidos no limite dimensional da matriz.

Fonte: Do Autor (2021).

Linha de comando 03: Impressão utilizando o comando **printf** já explicado no *script* 01;

Linha de comando 04: Utilizamos a função **zeros** para criar uma variável e preencher seus elementos com o valor zero. Pode ser criado um vetor ou uma matriz, dependendo dos parâmetros informados nos argumentos da função. Esta linha de comando cria uma matriz $m0$ cujas dimensões são cinco linhas de cinco colunas cada linha, totalizando vinte e cinco elementos, todos preenchidos com o valor zero.

Linha de comando 05: Esta linha apresenta uma operação de cópia de intervalo de elementos **entre matrizes**. Para realizar este tipo de operação e necessário lembrar que: **os intervalos informados devem estar contidos no limite dimensional da matriz**, nesta operação, dentro do limite de cada matriz. A interpretação dos intervalos é análoga à explicação apresentada no **Detalhamento da manipulação dos índices da matriz** m.

Linha de comando 06: Idem à linha de comando 03.

3 Funções de modo gráfico (*GUI*)

3.1 Função *figure*

A função ***figure*** abre uma janela para exibição de componentes visuais como gráficos, imagens entre outros. É importante salientar o uso de algumas das propriedades desta função:

Propriedades importantes da função *figure*

numbertitle permite exibir ou ocultar o numero que está associado ao nome da janela. O número é exibido por padrão, para ocultar basta acrescentar aos argumentos da função a setença: **"numbertitle","off"**;

name é o título que será exibido na janela. Para usar esta propriedade acrescente aos parâmetros da função a seguinte sentença: **"name","Título desejado"**.

menubar permite exibir ou ocultar a barra de menus da janela; por padrão a barra de menus é exibida na janela; para ocultar acrescente aos parâmetros a sentença: **"menubar","none"**.

toolbar permite exibir ou ocultar a barra de ferramentas da janela; por padrão a barra de ferramentas é exibida na janela; para ocultar acrescente aos parâmetros a sentença: **"toolbar","none"**.

papertype permite configurar o tamanho da página associado à janela, o que é relevante para a impressão e/ou exportação. Como exemplo segue a sentença de configuração para o papel tamanho A4: **"papertype","a4"**;

paperorientation permite configurar a orientação do papel para retrato ou paisagem; para retrato a sentença é: **"paperorientation","portrait"** e para paisagem: **"paperorientation","landscape"**.

position permite configurar a posição e o tamanho da janela na tela. Por exemplo segue a sentença para configurar uma janela iniciada na origem da tela e de dimensões de 400 x 300 pixels: **"position",[0 0 400 300]**; sobre os valores entre colchetes, o primeiro deles é a posição inicial no eixo x, o segundo a posição inicial no eixo y, o terceiro é a largura da janela e o quarto a altura;

windowstyle define a forma/estilo de exibição da janela; para que a janela fique sobreposta às demais janela do GNU Octave 6.1.0 sentença é: **"windowstyle","modal"**.

Fonte: Do Autor (2021).

Na data da redação desta nota havia outras informações e propriedades disponibilizadas através do endereço eletrônico: `https://octave.org/doc/v4.2.0/Figure-Properties.html` <<Acesso em 20 jan. 2021>>.

Veja em destaque um exemplo de utilização da função ***figure***. O número no argumento da função é o identificador da janela para a execução do *script*. Podem ser criadas outras figuras e com números diferentes.

Exemplo de utilização da função *figure*

figure(1,"numbertitle","off", "name","Título da janela", "menubar","none", "toolbar","none", "papertype","a4", "paperorientation","landscape", "position",[0 0 400 300] ,"windowstyle","modal");

Fonte: Do Autor (2021).

Uma janela criada através da função ***figure*** pode ser referenciada por uma variável por meio de atribuição. Esta possibilidade é muito importante para manipulações como a inserção de outros objetos gráficos na janela.

Veja o detalhe da atribuição da janela criada pela função ***figure*** a uma variável:

Detalhe da atribuição da janela a uma variável

f = figure();

Fonte: Do Autor (2021).

No detalhe a função ***figure*** foi apresentada com os parâmetros vazios para simplificar o entendimento da atribuição.

Nas próximas subseções serão apresentadas alguns componentes gráficos que podem ser inseridos nas janelas, então será possível observar a necessidade da referência à janela através de uma variável.

3.2 Função *uipanel*

Iniciamos a apresentação dos componentes pelo painel por facilitar a organização de outros componentes na janela, simplificando o processo de criação do *layout*.

ATENÇÃO! O comportamento do atributo ***position*** é diferente na função ***uipanel***, as coordenadas e dimensões são informadas no intervalo 0 a 1.

Segue o código-fonte onde se insere um painel em uma janela:

Código-fonte da inserção de um painel em uma janela

```
01. clc;
02. clear all;
03. f = figure(1, "numbertitle", "off", "name", "Título da figura", "menubar",
"none", "toolbar", "none");
04. clf;
05. p1 = uipanel(f,"title","Painel 1","position",[.025 .75 .95 .225]);
06. p2 = uipanel(f,"title","Painel 2","position",[.025 .025 .2 .7]);
07. p3 = uipanel(1,"title","Painel 3","position",[.25 .025 .725 .7]);
```

Fonte: Do Autor (2021).

As linhas de comando 1 a 4 já contam com explicações em outras partes desta nota, de agora em diante nos ateremos à explicação apenas das linhas de comando do que ainda não foram explicadas evitando repetições cansativas de explicações.

Linha de comando 05: Nesta linha fazemos uso de uma variável $p1$ para receber o painel criado pela função ***uipanel***. O primeiro parâmetro da função ***uipanel*** é onde indicamos a janela onde o painel será inserido, no caso foi utilizada a variável f para inserir

o painel na figura criada na linha de comando 03, aqui está a importância da referência à figura por meio de uma variável conforme apontamento feito na subseção **Função *figure***. O segundo parâmetro na verdade é a sentença "title", "Painel 1", que fará aparecer no painel o título Painel e o terceiro parâmetro é a sentença "position", [.025 .75 .95 .225], que informa a posição e as dimensões do painel na janela. Os dois primeiros valores são as coordenadas onde o painel será criado na janela e os dois últimos valores respectivamente a largura e altura do painel. As linhas de comando 06 e 07 seguem e mesmo raciocínio.

É possível fazer outras modificações significativas no painel, recomenda-se a leitura do manual do software.

3.3 Função *uicontrol* para criação de botão

A função ***uicontrol*** permite a inserção de vários tipos de componentes de interface com o usuário. Nesta subseção vamos nos dedicar a inserção de botões, que são inseridos nativamente pela função quando o tipo de componente não é informado.

Segue o código-fonte onde se insere um botão diretamente em uma janela e em um painel na janela:

Código-fonte da inserção de botão usando a função *uicontrol*

```
01. clc;
02. clear all;
03. f = figure(1, "numbertitle", "off", "name", "Título da figura", "menubar",
"none", "toolbar", "none");
04. clf;
05. p = uipanel(f,"title","Painel","position",[.025 .75 .95 .225]);
06. b1 = uicontrol(f, "string", "Botão na janela", "position", [10 10 180 45]);
07. b2 = uicontrol(p, "string", "Botão no painel", "position", [200 10 320 45]);
```

Fonte: Do Autor (2021).

Linha de comando 06: É atribuída à variável $b1$ a saída da função ***uicontrol***. O primeiro parâmetro da função ***uicontrol*** é utilizado para informar em qual componente o botão será criado, neste caso na janela criada na linha de comando 03 e atribuída à variável f; o segundo parâmetro consiste da sentença "string", "Botão na janela"que faz a segunda parte da sentença ser impressa no botão inserido na janela. A posição é informada em pixels. A interpretação da linha de comando 07 é análoga.

É possível fazer outras modificações significativas através da função ***uicontrol***, recomenda-se a leitura do manual do software.

3.4 Função *uicontrol* para criação de outros componentes

Nesta etapa vamos trabalhar a criação de outros componentes por meio da função ***uicontrol***.

Segue o código-fonte onde se faz a inserção de componentes:

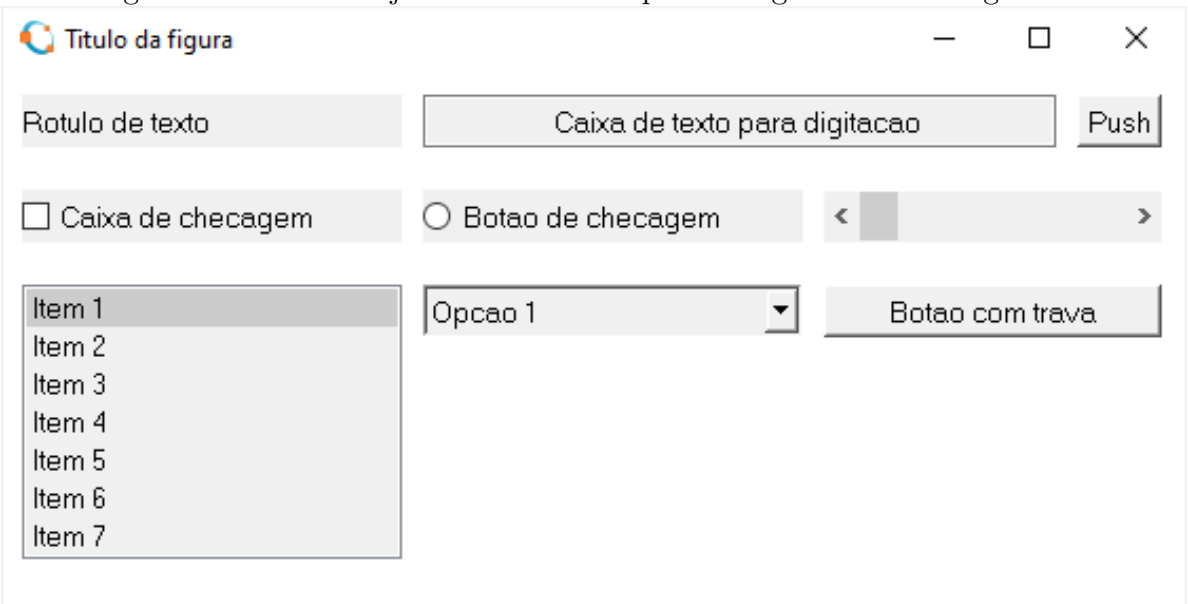
Código-fonte da inserção de outros componentes através da função *uicontrol*

```
01. clc;
02. clear all;
03. f = figure(1, "numbertitle", "off", "name", "Título da figura", "menubar", "none", "toolbar", "none");
04. clf;
05. r = uicontrol(f, "style", "text", "string", "Rótulo de texto", "horizontalalignment", "left", "position", [10 385 180 25]);
06. cx = uicontrol(f, "style", "edit", "string", "Caixa de texto para digitação", "Position", [200, 385, 300, 25]);
07. pb = uicontrol(f, "style", "pushbutton", "string", "Push", "position", [510 385 40 25]);
08. cb = uicontrol(f, "style", "checkbox", "string", "Caixa de checagem", "position", [10 350 180 25]);
09. rb = uicontrol(f, "style", "radiobutton", "string", "Botão de checagem", "position", [200 350 180 25]);
10. s = uicontrol(f, "style", "slider", "string", "Escorregador", "position", [390 350 160 25]);
11. lb = uicontrol(f, "style", "listbox", "string", {"Item 1"; "Item 2"; "Item 3"; "Item 4"; "Item 5"; "Item 6"; "Item 7";}, "position", [10 190 180 130]);
12. pm = uicontrol(f, "style", "popupmenu", "string", {"Opção 1", "Opção 2", "Opção 3Opção 4", "Opção 5"}, "position", [200 295 180 25]);
13. tb = uicontrol(f, "style", "togglebutton", "string", "Botão com trava", "position", [390 295 160 25]);
```

Fonte: Do Autor (2021).

Para o código-fonte apresentado faz-se interessante a apresentação de seu resultado devido ao vulto significativo de componentes inseridos de uma única vez, esta apresentação é feita na Figura 2:

Figura 2: Recorte da janela com os componentes gráficos do código-fonte



Fonte: Do Autor (2021).

Linha de comando 05: Cria na figura f o rótulo de texto na posição e dimensões determinadas. Os rótulos são criados por meio da sentença **"style","text"**. O que há de novidade é uma mudança de alinhamento no texto realizável por meio da sentença: "horizontalalignment","left", pois o alinhamento horizontal padrão do rótulo é centralizado "center".

Linha de comando 06: Cria na figura f a caixa de texto para digitação na posição e dimensões determinadas. As caixas de texto para digitação são criados por meio da sentença **"style","edit"**.

Linha de comando 07: Cria na figura f um botão com mola (volta ao ser pressionado) na posição e dimensões determinadas. Os botões com mola são criados por meio da sentença **"style","pushbutton"** e seus rótulos a eles atrelados por meio da sentença "string", "Rótulo do botão".

Linha de comando 08: Cria na figura f a caixa de checagem na posição e dimensões determinadas. As caixas de checagem são criadas por meio da sentença **"style","checkbox"** e o texto a elas atrelados por meio da sentença "string", "texto atrelado".

Linha de comando 09: Cria na figura f o botão de checagem na posição e dimensões determinadas. Os botões de checagem são criados por meio da sentença **"style","radiobutton"** e o texto a eles atrelados por meio da sentença "string", "texto atrelado".

Linha de comando 10: Cria na figura f o controle deslizante na posição e dimensões determinadas. Os controles deslizantes são criados por meio da sentença **"style","slider"**.

Linha de comando 11: Cria na figura f a caixa de listagem na posição e dimensões determinadas. As caixas de listagem são criadas por meio da sentença **"style","listbox"** e os itens a elas atrelados por meio da sentença "string", {lista de itens dentro das duas chaves com cada item delimitado por aspas duplas e separados por ponto-e-vírgula}.

Linha de comando 12: Cria na figura f um *popupmenu* (caixa de seleção) na posição e dimensões determinadas. Os *popupsmenu* são criados por meio da sentença **"style", "popupmenu"** e as opções a eles atrelados por meio da sentença "string", {lista de itens dentro das duas chaves com cada item delimitado por aspas duplas e separados por ponto-e-vírgula}.

Linha de comando 13: Cria na figura f um botão com trava na posição e dimensões determinadas. Os botões com trava são criadas por meio da sentença **"style","togglebutton"** e seus rótulos a eles atrelados por meio da sentença "string", "Rótulo do botão".

Relembramos que é possível fazer outras modificações significativas nos componentes através da função ***uicontrol***, recomenda-se a leitura do manual do software.

Agora que concluímos o estudo básico da criação dos componentes gráficos criados com a função ***uicontrol*** nos interessa a interagir com estes através da manipulação em *background*, assunto das duas próximas subseções.

3.5 Obtendo informações de propriedades dos componentes gráficos *uicontrol*

Já aprendemos a criar os componentes gráficos utilizando a função ***uicontrol***, agora vamos trabalhar a manipulação destes componentes em *background*. Esta subseção está focada em obter (*get*) informações presentes nos componentes gráficos.

Para obter dados de um componente utilizamos a função ***get***, cuja sintaxe básica é **get(componente, "propriedade")**, onde o parâmetro componente é a variável que está associada ao componente que criamos e queremos saber informação sobre a propriedade dele, e o parâmetro "propriedade"é a informação que queremos obter do objeto. Por exemplo, em determinado momento da execução de um *script* pode ser do interesse saber para qual posição um usuário deslizou um controle deslizante (associado a uma variável s), e isto pode ser feito por meio da função ***get***: get(s,"value"); ou também pode haver a necessidade de se

coletar o valor informado por um usuário em uma determinada caixa de texto (associada a uma variável cx), o que pode ser feito de forma similar: get(cx,"string").

3.6 Modificar informações das propriedades dos componentes gráficos *uicontrol*

Já aprendemos a criar e obter informações dos componentes gráficos ***uicontrol***, agora vamos modificar as propriedades destes componentes em *background*. Esta subseção está focada em modificar (*set*) informações presentes nos componentes gráficos.

Para obter dados de um componente utilizamos a função ***get***, cuja sintaxe básica é **set(componente, "propriedade", "valor da propriedade")**, onde o parâmetro componente é a variável que está associada ao componente que criamos e queremos modificar a propriedade dele, o parâmetro "propriedade"é a propriedade que queremos modificar do objeto e o parâmetro valor da propriedade é o valor que será atribuído à propriedade do objeto em modificação. Por exemplo, em determinado momento da execução de um *script* pode ser necessário restaurar a posição de um controle deslizante (associado a uma variável s), e isto pode ser feito por meio da função ***set***: set(s,"value",.5), o que coloca o cursor do controle deslizante no meio da barra; ou também pode haver a necessidade de se modificar o conteúdo em uma determinada caixa de texto (associada a uma variável cx), o que pode ser feito de forma similar: set(cx,"string","nova mensagem").

Segue um exemplo onde se utilizam as funções ***get*** e ***set***:

Exemplo de utilização das funções *get* e *set* em componentes *uicontrol*

```
01. clc;
02. clf;
03. f = figure(1, "numbertitle", "off", "name", "Titulo da figura", "menubar",
"none", "toolbar", "none", "position", [100, 100, 350, 150]);
04. r = uicontrol(f, "style","text","string", "Posição do coontrole deslizante: ",
"horizontalalignment","left","position", [50 100 250 25]);
05. s = uicontrol(f, "style", "slider", "string", "Escorregador", "position", [50 50
160 25]);
06. while(1)
07. x = get(s,"value");
08. set(r,"string",strcat("Posição do controle deslizante: ",num2str(x)));
09. pause(0.05);
10. endwhile
```

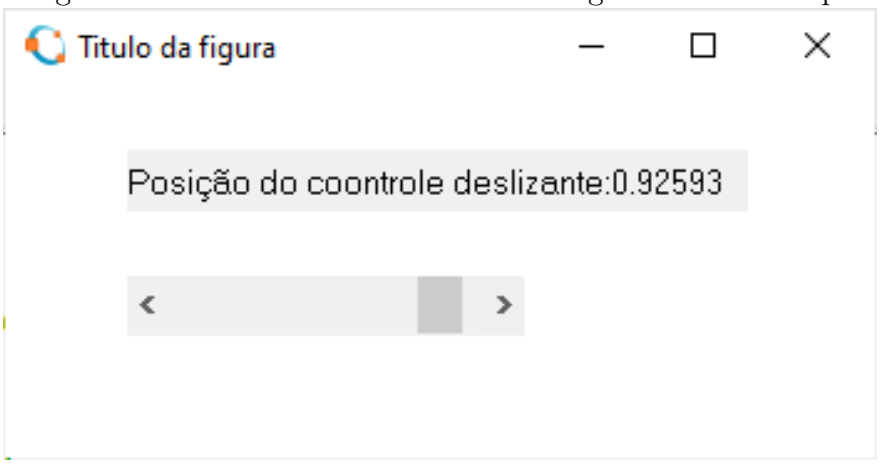
Fonte: Do Autor (2021).

É útil a apresentação da janela com o resultado do código-fonte do exemplo (Figura 3) para fins de complementar a explicação do raciocínio:

O código-fonte do exemplo cria os componentes gráficos como já estudamos e a novidade está no laço de repetição *while* (linhas de comando 06 a 10), este repete continuamente os comandos nele contidos até a execução do *script* ser finalizada. Na linha de comando 07 é lido o último valor fixado no controle deslizante e este é armazenado na variável x; na linha de comando 08 é repetido o rótulo de texto acrescido do último valor fixado no controle deslizante e na linha de comando 09 faz-se uma pausa de 50 (cinquenta) milissegundos e então o laço de repetição ocorre novamente.

Chamamos a atenção para a expressão *"último valor fixado no controle deslizante"*, pois o valor que é armazenado na variável somente após o controle deslizante estabilizar na nova

Figura 3: Janela com o resultado do código-fonte do exemplo



Fonte: Do Autor (2021).

posição e ainda depende do decurso de tempo de cada repetição do laço para que esta nova posição seja lida como a do "último valor fixado no controle deslizante".

3.7 Barra de menus

A barra de menus padrão da janela foi ocultada nas explicações de propósito, para que neste momento possamos estudar a construção de uma barra de menu. Vamos ao exemplo de construção de uma barra de menu:

Exemplo da construção de uma barra de menu

```
01. clc;
02. f = figure(1, "numbertitle", "off", "name", "Titulo da figura", "menubar",
"none", "toolbar", "none");
03. clf;
04. menuA = uimenu(f,"label","Menu &A");
05. uimenu(menuA,"label", "Opção A&1");
06. uimenu(menuA,"label", "Opção A&2", "accelerator","Y");
07. uimenu(menuA,"label", "Opção A&3", "accelerator","y");
08. menuB = uimenu(f,"label","Menu &B");
09. uimenu(menuB,"label", "Opção B&1");
10. uimenu(menuB,"label", "Opção B&2", "accelerator", "Z");
11. uimenu(menuB,"label", "Opção B&3", "accelerator", "z");
```

Fonte: Do Autor (2021).

3.8 Barra de ferramentas

Da mesma forma que a barra de menus, a barra de ferramentas padrão da janela foi ocultada nas explicações de propósito, para que neste momento possamos estudar a construção de uma barra de ferramentas. Vamos ao exemplo de construção de uma barra de ferramentas:

Exemplo da construção de uma barra de ferramentas

```
01. clc;
02. f = figure(1, "numbertitle", "off", "name", "Titulo da figura", "menubar", "none", "toolbar", "none");
03. clf;
04. bf = uitoolbar(f);
05. icone = rand(19,19,3);
06. fr1 = uitoggletool(bf,"cdata",icone,"tooltipstring","Ferramenta 01 com trava","clickedcallback","msgbox('Funcionou a ferramenta 1 com trava!')");
07. icone = rand(19,19,3);
08. fr2 = uitoggletool(bf,"cdata",icone,"tooltipstring","Ferramenta 02 com trava","clickedcallback","msgbox('Funcionou a ferramenta 2 com trava!')");
09. icone = rand(19,19,3);
10. fr3 = uitoggletool(bf,"cdata",icone,"tooltipstring","Ferramenta 03 com trava","clickedcallback","msgbox('Funcionou a ferramenta 3 com trava!')");
11. icone = randi([0 16],19,19,3);
12. fr4 = uipushtool(bf,"cdata",icone,"tooltipstring","Ferramenta 01 com mola","clickedcallback","msgbox('Funcionou a ferramenta 1 com mola!')");
13. icone = randi([0 16],19,19,3);
14. fr5 = uipushtool(bf,"cdata",icone,"tooltipstring","Ferramenta 02 com mola","clickedcallback","msgbox('Funcionou a ferramenta 2 com mola!')");
```

Fonte: Do Autor (2021).

3.9 Barra de progresso

Esta ferramenta é útil principalmente quando desejamos informar o andamento de tarefas que estão sendo executadas em *background*.

Vamos tratar de um exemplo simples para apresentar o andamento de uma operação de laço de repetição conhecido.

Exemplo de uso de uma barra de progresso para exibir o andamento de um laço de repetição conhecido

```
01. clc;
02. f = figure(1, "numbertitle", "off", "name", "Titulo da figura", "menubar",
"none", "toolbar", "none");
03. clf;
04. bp = waitbar(0,"Progresso","name","Barra de progresso");
05. I = 10;
06. J = 10;
07. numOperacoes = I*J;
08. salto = 1/numOperacoes;
09. preenchimento = 0;
10. for(i=1:1:I)
11. for(j=1:1:J)
12. pause(1);
13. preenchimento += salto;
14. waitbar(single(preenchimento),bp);
15. endfor
16. endfor
```

Fonte: Do Autor (2021).

Para o uso coerente da barra de progresso necessitamos saber qual o número de operações que ela representará e quanto cada operação representa. Havendo a coerência vamos ao entendimento da função *waitbar*:

Diferentemente do que aprendemos até o momento, o primeiro parâmetro da função é um valor fracionário no intervalo [0, 1] que representa a porcentagem do preenchimento da barra, porcentagem esta que deve estar relacionada à operação cujo progresso está sendo representado por meio do preenchimento da barra. Vamos à interpretação do código-fonte do exemplo de utilização da barra de progresso.

Linha de comando 04: Cria uma caixa de diálogo com a barra de progresso iniciada sem preenchimento (zero) e a associa à variável h. O segundo parâmetro da função exibe uma mensagem acima da barra de progresso, neste caso específico é exibida a mensagem "Progresso". O nome da caixa de diálogo que será exibido em sua barra de título é configurado pela propriedade "name", seguida do nome que se deseja exibir na barra de título da caixa de diálogo, neste exemplo "Barra de progresso". A função ***waitbar*** herda propriedades da função ***figure***, ou seja propriedades da função ***figure*** são comuns à função ***waitbar***.

No exemplo o número de operações a serem representadas (**linha de comando 07**) é dado pelo produto das variáveis (**linhas de comando 05 e 06**) que delimitam os laços de repetição (**linhas de comando 10 e 11**). Conhecido o número de operações calcula-se o tamanho do salto que cada operação representa no preenchimento da barra (**linha de comando 08**). Criamos uma variável para representar o preenchimento da barra (**linha de comando 09)** e esta variável é atualizada a cada execução do laço de repetição interno (**linha de comando 13**) por meio de um incremento de um salto calculado (**linha de comando 08**). O último passo é atualizar a barra para que esta represente o progresso da operação (**linha de comando 14**), o que se faz chamando a função, informando o valor do preenchimento e qual barra será preenchida, no caso, a barra representada pela variável h. Cabe ressaltar que a função ***waitbar*** gera erro se o valor informado for de precisão dupla, o que faz necessária a conversão do valor do preenchimento para precisão simples por meio da função ***single*** (**linha de comando 14**).

4 Outros componentes gráficos: plotagem de dados

Esta seção foi criada para tratar de alguns modelos de gráficos e principalmente a parte de formatação destes, o que pode ser muito útil no dia-a-dia do profissional da Engenharia e que são úteis para o PDI.

Sobre gráficos é importante saber que existem alguns recursos como título, rótulos de eixos, formatações dos eixos, rótulos de dados na seção de apresentação de dados, gráficos animados entre outros.

4.1 Configurações comuns aos gráficos

É possível utilizar o modo de interpretação em ***Latex*** para reproduzir diversos símbolos especiais nos gráficos, como letras Gregas por exemplo. No endereço eletrônico `https://octave.org/doc/v6.1.0/Use-of-the-interpreter-Property.html` são apresentadas as formas de inserção destes caracteres. Uma observação experimental utilizando o interpretador ***latex*** é que o texto deve ser inserido nos argumentos das funções ***entre apóstrofos*** e não aspas duplas.

4.1.1 Título

O título do gráfico é redigido através da função ***title***, como por exemplo title("Título do gráfico"). Se for necessário é possível imprimir múltiplas linhas no título do gráfico por meio da fragmentação do parâmetro, como, por exemplo, title({"Primeira parte"; "Segunda parte"}).

Em algumas situações pode ser útil formatar o título do gráfico, então para isto lançamos mão, dentre outras, das propriedades "fontname"para ajustar a fonte, "fontweight"para aplicar negrito, "fontsize"para ajustar o tamanho, "horizontalalignment"e "verticalalignment"para alinhamento do título. Maiores detalhes sobre a utilização destas propriedades podem ser encontradas no endereço eletrônico: `https://octave.org/doc/v6.1.0/Text-Properties.html`.

Um exemplo de linha de comando aplicando formatação no título: title({"Linha 1";"Linha 2"}, "fontname","arial","horizontalalignment","left").

4.1.2 Eixos

Eixos contam com dois recursos importantes: rótulo e intervalo. O rótulo é uma **legenda do eixo** que acompanha o eixo ao qual está vinculado e intervalo são os valores que no eixo do gráfico são exibidos.

Para configurar os rótulos dos eixos, configuramos eixo a eixo os rótulos; para o eixo x utiliza-se a função ***xlabel***, como por exemplo xlabel("Rótulo X"), para o eixo y utiliza-se a função ***ylabel*** e para z a função ***zlabel***.

Os limites do gráfico, podemos obter ou configurar; quando queremos saber os limites dos eixos, devemos consultar para cada eixo, a consulta para o eixo x é feita através da função *xlim* com parâmetro vazio: xlim(); esta função retorna um vetor com os dois valores dos limites do eixo x do gráfico; o processo é análogo para o eixo y, porém a função utilizada é a ***ylim***, e para o eixo z, a função ***zlim***. Se necessário, é possível consultar os limites de todos os eixos de uma única vez através de uma única função, a função ***axis***, também com parâmetro vazio, que retornará um vetor com pares de elementos, um par para cada eixo e de acordo com o número de eixos exibidos; e seguindo a lógica x, y, z, se o gráfico contiver apenas dois eixos serão retornados dois pares de valores num vetor, os dois primeiros referentes aos limites do eixo x, os dois últimos do eixo y; se o gráfico contiver três eixos, a

função retornará um vetor com três pares de valores, respectivamente limites dos eixos x, y e z. No momento outras organizações de gráficos estão fora do escopo deste estudo.

Dadas as características do polimorfismo podemos utilizar estas mesmas funções para configurar os limites dos eixos do gráfico. O que precisamos ter em mente é que para cada eixo se configura um par de valor de limites, o primeiro valor é o limite inferior (*low*) do eixo e o segundo, o limite superior (*high*). No caso do eixo x, utilizando a função ***xlim***, fazemos a configuração do intervalo do eixo passando os dois valores do limite dentro de um vetor no parâmetro da função: xlim([low high]); o procedimento é análogo para as outras funções individuais dos eixos y e z (***ylim*** e ***zlim***).

Fazendo uso do polimorfismo também podemos configurar os limites dos eixos de uma única vez através da função ***axis***, passando como parâmetro um vetor com o número de pares e valores de limites (*low* e *high*) que desejamos configurar, respeitada a ordenação x, y e z. Como vamos fazer a configuração de um ou mais eixos, prefixamos os limites inferior (*low*) e superior (*high*) com a respectiva identificação do eixo (exemplo: *Xlow* e *Xhigh*). Para configurar apenas o eixo x redigimos a função ***axis*** com um par de valores no vetor do parâmetro, da seguinte forma: axis([Xlow Xhigh]); para configurar os eixos x e y, da seguinte forma: axis([Xlow Xhigh Ylow Yhigh]); e para configurar os eixos x, y e z, da seguinte forma: axis([Xlow Xhigh Ylow Yhigh Zlow Zhigh]).

Podemos continuar explorando o polimorfismo oferecido pela função ***axis***, que além de permitir a consulta e configuração dos valores dos limites dos eixos do gráfico, permite aplicar ajustes de aspecto visual no gráfico, tais ajustes, na data da redação desta nota, são detalhados no endereço eletrônico `https://octave.sourceforge.io/octave/function/axis.html`.

5 Exibição de imagens

A exibição de imagens exige atenção! A função ***imshow*** exibe imagens cujos tipos de variáveis para representá-las sejam variáveis dos tipos *uint8*, *uint16*, ...

Imagens com valores negativos podem até ser exibidas, porém o *Octave* realiza um ajuste automático mapeando o máximo valor negativo com o valor mínimo de intensidade (zero) e o maior valor com o maior valor de intensidade (um).

Imagens representadas por números de ponto flutuante ou de precisão dupla também são apresentados mapeados num intervalo entre zero e um.

O problema na exibição das imagens ocorre quando não há uma conversão dos números para serem representados dentro do intervalo, o que reflete em exibições incorretas das imagens, normalmente imagens sobrecarregadas de branco.

Uma forma de adaptar as imagens para o sistema *RGB* de 24 *bits* é mapear as intensidades para a faixa do sistema e depois certificar-se do mapeamento convertendo a imagem através da função *uint8*.

É possível fazer com que a exibição da imagem tenha seu intervalo determinado automaticamente pela função *imshow* utilizando-se de um par de colchetes vazio nos segundo parâmetro, **porém nem sempre este procedimento apresenta os resultados conforme esperado**. Considerando uma imagem representada por uma variável I, a parametrização da função *imshow* com a autoconfiguração de intervalo assume o seguinte formato:

Função *imshow* com autoconfiguração de intervalo

```
imshow(I,[]);
```

Fonte: Do Autor (2021).

6 Funções

ATENÇÃO!

Atenção! Salvo disposição em contrário, o retorno das funções estudadas nas próximas subseções depende das características dimensionais do intervalo informado no parâmetro. Se for passado um vetor, independentemente se vetor linha ou vetor coluna é retornado apenas um elemento do intervalo como resposta da função. Se o parâmetro for uma matriz, o retorno será um vetor linha, esta linha contendo o valor da função de cada coluna.
Tomando como função de exemplo a função ***max***, que retorna o máximo valor do intervalo, seguem-se os exemplos para as duas situações supracitadas.
Exemplo 1: vetor linha V de 5 elementos:

$$V = \begin{bmatrix} 2 & 3 & 9 & 8 & 5 \end{bmatrix};$$

O retorno da função $max(V)$ será 9.
Exemplo 2: matriz M de 15 elementos:

$$M = \begin{bmatrix} 2 & 3 & 9 & 4 & 1 \\ 7 & 6 & 9 & 5 & 1 \\ 5 & 3 & 9 & 8 & 4 \end{bmatrix};$$

O retorno da função $max(M)$ será um vetor linha V:

$$V = \begin{bmatrix} 7 & 6 & 9 & 8 & 4 \end{bmatrix};$$

Neste momento deve ter surgido uma pergunta: como encontrar o valor máximo em uma matriz? A resposta pode ser percebida nos próprios exemplos 1 e 2, ao se aplicar a função a uma matriz se tem um vetor e ao se aplicar a um vetor tem-se um elemento, então é possível concluir que ao aplicar-se duas vezes a função a uma matriz atinge-se o objetivo de obter-se apenas um elemento de retorno. Traduzindo em comando: $max(max(M))$.
A ideia de utilização é análoga nas funções que serão apresentadas nas próximas subseções.

Fonte: Do Autor (2021).

6.1 Valor da média aritmética do intervalo

A função para retornar o valor da média aritmética de um intervalo é a função ***mean***.

6.2 Valor da mediana do intervalo

A função para retornar o valor da mediana de um intervalo é a função ***median***.

6.3 Valor da moda do intervalo

A função para retornar o valor da moda de um intervalo é a função ***mode***.

6.4 Valor da média harmônica do intervalo

A função para retornar o valor da média harmônica de um intervalo é a função ***mean***, com uma pequena modificação, após o primeiro parâmetro, que é o intervalo, acrescenta-se no segundo parâmetro o termo "h". Aqui outra vez percebe-se a importância do ***polimorfismo***.

6.5 Valor da média geométrica do intervalo

A função para retornar o valor da média geométrica de um intervalo é a função ***mean***, com uma pequena modificação, após o primeiro parâmetro, que é o intervalo, acrescenta-se no segundo parâmetro o termo "g". Aqui mais uma vez percebe-se a importância do ***polimorfismo***.

6.6 Valor da média quadrática do intervalo

A função para retornar o valor da média quadrática de um intervalo é a função ***meansq***.

6.7 Valor do desvio padrão do intervalo

A função para retornar o valor do desvio padrão de um intervalo é a função ***std***.

6.8 Valor da variância do intervalo

A função para retornar o valor da variância de um intervalo é a função ***var***.

6.9 Valor máximo do intervalo

Como já explicado a função ***max*** retorna o valor máximo de um intervalo de acordo com as características dimensionais do intervalo. Vide exemplos 1 e 2 no Quadro de Atenção supracitado.

6.10 Valor mínimo do intervalo

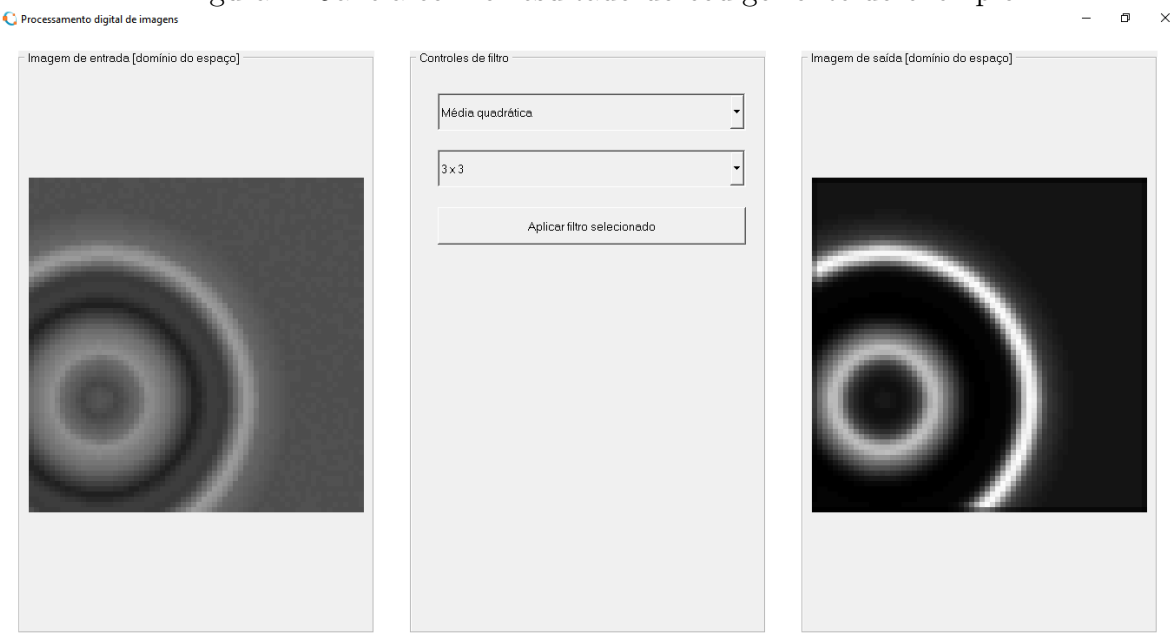
A função para retornar o valor mínimo de um intervalo é a função ***min***.

7 Exemplo de sistema simples de processamento de imagens no espaço

Dadas as explicações sobre componentes gráficos neste material e o andamento do Curso de Engenharia da Computação considera-se que seja possível a interpretação do código-fonte à seguir com pouco esforço e pesquisa.

A Figura 4 apresenta a interface gerada por meio do código-fonte apresentado no Quadro 7.

Figura 4: Janela com o resultado do código-fonte do exemplo



Fonte: Do Autor (2021).

Quadro 7: Código-fonte de um sistema simples de processamento de imagens no domínio do espaço

```
clc;
clear all;
f = figure(1, "name", "Processamento digital de imagens",
 ↪  "numbertitle", "off", "menubar", "none", "toolbar", "none",
 ↪  "papertype", "a4");
clf;
[arq cam] = uigetfile("");
img = imread(strcat(cam,arq));
S1=size(img,1);
S2=size(img,2);
imgOut = img;
## Controles
controles = uipanel(f,"title","Controles de
 ↪  filtro","position",[.345 .015 .30 .95],"units","normalized");
##rotulo = uicontrol(controles, "style", "text",
 ↪  "string","Filtro", "horizontalalignment", "left");
```

```
popUpMenuFiltro = uicontrol(controles,"style", "popupmenu",
 ↪  "string",{"Média aritmética", "Média harmônica", "Média
 ↪  geométrica", "Média quadrática", "Mediana", "Moda", "Valor
 ↪  máximo", "Valor mínimo"},"units","normalized","position",[.075
 ↪  .875 .875 .065]);
popUpMenuFiltroSegmento = uicontrol(controles,"style", "popupmenu",
 ↪  "string",{"3 x 3", "5 x 5", "7 x 7", "9 x
 ↪  9"},"units","normalized","position",[.075 .775 .875 .065]);
botaoFiltroAplicar = uicontrol(controles,"style", "pushbutton",
 ↪  "string","Aplicar filtro
 ↪  selecionado","units","normalized","position",[.075 .675 .875
 ↪  .065],"callback",{@aplicaFiltro, popUpMenuFiltro,
 ↪  popUpMenuFiltroSegmento, img, S1, S2});
imgViewer1 = uipanel(f,"title","Imagem de entrada [domínio do
 ↪  espaço]","position",[.015 .015 .30 .95]);
imgViewer2 = uipanel(f,"title","Imagem de saída [domínio do
 ↪  espaço]","position",[.675 .015 .30 .95]);
ImgFrm1 = axes ( "parent",imgViewer1,
    "position", [0.025, 0.025, .95, 0.95],
    "xtick",    [],
    "ytick",    [],
    "xlim",     [0, 1],
    "ylim",     [0, 1]);
ImgFrm2 = axes ( "parent",imgViewer2,
    "position", [0.025, 0.025, .95, 0.95],
    "xtick",    [],
    "ytick",    [],
    "xlim",     [0, 1],
    "ylim",     [0, 1]);
axes(ImgFrm1);
imshow(img);
axes(ImgFrm2);
imshow(imgOut);

function aplicaFiltro (x, y, funcao, dimensao, img, S1, S2)
disp(x)
disp(y)
  switch(get(dimensao,"value"))
    case 1
      w=3;
    case 2
      w=5;
    case 3
      w=7;
    case 4
      w=9;
    otherwise
      w=0;
  endswitch
```

```
if(w!=0)
w2 = (w+1)/2;
w3 = (w-1)/2;
imgOutLocal=zeros(2*w3+S1,2*w3+S2);
imgTemp=imgOutLocal;
imgTemp(w2:S1+w3,w2:S2+w3)=img(:,:);

switch (get(funcao,"value"))
  case 1
    ## "Média aritmética"
    for(l=w2:1:S1+w3)
      for(c=w2:1:S2+w3)
        imgOutLocal(l,c) =
         ↪  mean(mean(imgTemp(l-w3:l+w3,c-w3:c+w3)));
      endfor
    endfor
    imgOutLocal(1:w3,:)=[];
    imgOutLocal(S1+1:S1+w3,:)=[];
    imgOutLocal(:,1:w3)=[];
    imgOutLocal(:,S2+1:S2+w3)=[];
    imshow(imgOutLocal,[]);
  case 2
    "Média harmônica"
    for(l=w2:1:S1+w3)
      for(c=w2:1:S2+w3)
        imgOutLocal(l,c) =
         ↪  mean(mean(imgTemp(l-w3:l+w3,c-w3:c+w3),"h"),"h");
      endfor
    endfor
    imgOutLocal(1:w3,:)=[];
    imgOutLocal(S1+1:S1+w3,:)=[];
    imgOutLocal(:,1:w3)=[];
    imgOutLocal(:,S2+1:S2+w3)=[];
    imshow(imgOutLocal,[]);
  case 3
     "Média geométrica"
    for(l=w2:1:S1+w3)
      for(c=w2:1:S2+w3)
        imgOutLocal(l,c) =
         ↪  mean(mean(imgTemp(l-w3:l+w3,c-w3:c+w3),"g"),"g");
      endfor
    endfor
    imgOutLocal(1:w3,:)=[];
    imgOutLocal(S1+1:S1+w3,:)=[];
    imgOutLocal(:,1:w3)=[];
    imgOutLocal(:,S2+1:S2+w3)=[];
    imshow(imgOutLocal,[]);
  case 4
```

```
    "Média quadrática"
    for(l=w2:1:S1+w3)
      for(c=w2:1:S2+w3)
        imgOutLocal(l,c) =
        ↪  meansq(meansq(imgTemp(l-w3:l+w3,c-w3:c+w3)));
      endfor
    endfor
    imgOutLocal(1:w3,:)=[];
    imgOutLocal(S1+1:S1+w3,:)=[];
    imgOutLocal(:,1:w3)=[];
    imgOutLocal(:,S2+1:S2+w3)=[];
    imshow(imgOutLocal,[]);
  case 5
    "Mediana"
    for(l=w2:1:S1+w3)
      for(c=w2:1:S2+w3)
        imgOutLocal(l,c) =
        ↪  median(median(imgTemp(l-w3:l+w3,c-w3:c+w3)));
      endfor
    endfor
    imgOutLocal(1:w3,:)=[];
    imgOutLocal(S1+1:S1+w3,:)=[];
    imgOutLocal(:,1:w3)=[];
    imgOutLocal(:,S2+1:S2+w3)=[];
    imshow(imgOutLocal,[]);
  case 6
    "Moda"
    for(l=w2:1:S1+w3)
      for(c=w2:1:S2+w3)
        imgOutLocal(l,c) =
        ↪  mode(mode(imgTemp(l-w3:l+w3,c-w3:c+w3)));
      endfor
    endfor
    imgOutLocal(1:w3,:)=[];
    imgOutLocal(S1+1:S1+w3,:)=[];
    imgOutLocal(:,1:w3)=[];
    imgOutLocal(:,S2+1:S2+w3)=[];
    imshow(imgOutLocal,[]);
  case 7
    "Valor máximo"
    for(l=w2:1:S1+w3)
      for(c=w2:1:S2+w3)
        imgOutLocal(l,c) =
        ↪  max(max(imgTemp(l-w3:l+w3,c-w3:c+w3)));
      endfor
    endfor
    imgOutLocal(1:w3,:)=[];
    imgOutLocal(S1+1:S1+w3,:)=[];
    imgOutLocal(:,1:w3)=[];
```

```
      imgOutLocal(:,S2+1:S2+w3)=[];
      imshow(imgOutLocal,[]);
    case 8
      "Valor mínimo"
      for(l=w2:1:S1+w3)
        for(c=w2:1:S2+w3)
          imgOutLocal(l,c) =
          ↪  min(min(imgTemp(l-w3:l+w3,c-w3:c+w3)));
        endfor
      endfor
      imgOutLocal(1:w3,:)=[];
      imgOutLocal(S1+1:S1+w3,:)=[];
      imgOutLocal(:,1:w3)=[];
      imgOutLocal(:,S2+1:S2+w3)=[];
      imshow(imgOutLocal,[]);
    otherwise
      msgbox("Erro de lógica no código-fonte.");
  endswitch
  else
    msgbox("Não foi possível aplicar o filtro à imagem.");
  endif
endfunction
```

Fonte: Do Autor (2021).

Referências

[1] *GNU Octave*. Copyright © 1996-2017 John W. Eaton. Disponível em: `https://www.gnu.org/software/octave/doc/v4.2.1`. Acesso em 05 Maio 2021.

[2] *GNU Octave (version 6.1.0)*. 2020. Copyright © 1996-2020 John W. Eaton. Disponível em: `https://www.gnu.org/software/octave/doc/v6.1.0`. Acesso em: 05 Maio 2021.