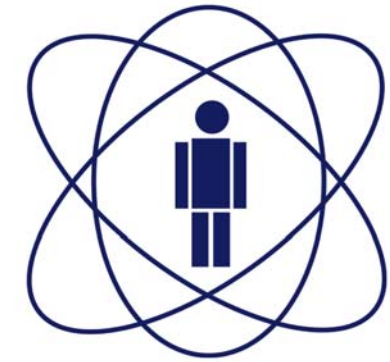


Introdução a Programação em MATLAB para o Cluster Linux

AUTORES:
Alan Tavares Miranda
Márcio Portes de Albuquerque
Marcelo Portes de Albuquerque
Elisângela Lopes de Faria

Nota Técnica CBPF - CBPF-NT-004/2008

SUMÁRIO

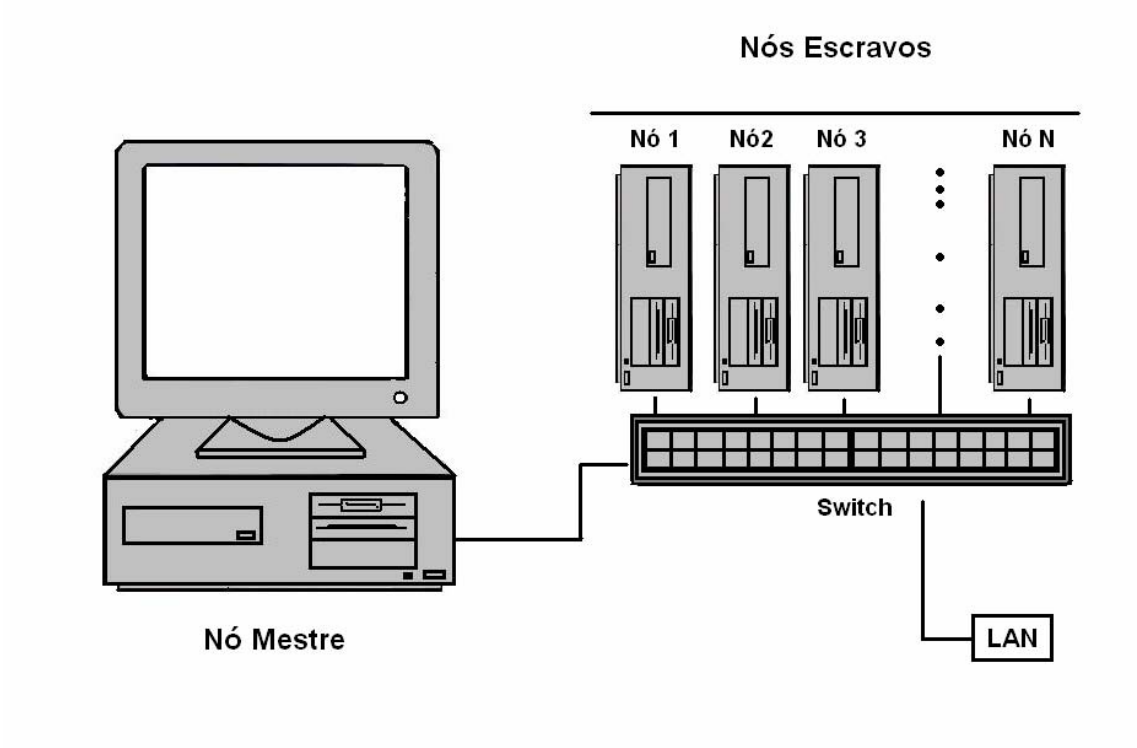
1.Introdução

Cluster é um sistema que consiste em dois ou mais computadores ou sistemas na qual trabalham em conjunto para executar aplicações ou realizar outras tarefas, de tal forma que os usuários que os utilizam tenham a impressão que somente um único sistema responde para eles, criando assim uma ilusão de um recurso único (**computador virtual**). Como características fundamentais para a construção destas plataformas incluem-se **elevação da: confiança, distribuição de carga e performance.**

O CBPF dispõe hoje de uma infra-estrutura central de computação de alto desempenho baseada em um Cluster de computadores do tipo **Beowulf**.

Um sistema de cluster do tipo Beowulf funciona a partir de duas partes fundamentais: o **nó mestre** e os **nós escravos**. O nó mestre tem como função controlar o cluster monitorando e distribuindo as tarefas, funciona como servidor de arquivos e é a ligação entre os usuários e o cluster; já os nós escravos têm como objetivo executar as tarefas determinadas pelo nó mestre. O número de nós escravos pode variar, sendo que quanto mais deles existirem no sistema melhor será a distribuição de carga.

Abaixo temos um esquema do cluster do tipo Beowulf.



- O *switch* mostrado no esquema tem como objetivo organizar a transferência de dados entre os nós mestre e escravos.

Um recurso amplamente utilizado pela comunidade científica e disponível aos usuários do Cluster são softwares comercias, como por exemplo, o pacote Mathematica ou MATLAB que oferecem um completo ambiente de programação.

No CBPF existem diversos ramos de pesquisas físicas que utilizam a computação em cluster para otimizar os resultados a fim de obtê-los de forma mais rápida e precisa como por exemplo : cálculo de estruturas eletrônicas, materiais e sistemas magnéticos, astrofísica, física estatística, sistemas dinâmicos, física de altas e energias e cosmologia.

Além disso, o cluster é utilizado como auxilio em redes neurais artificiais, processamento digital de imagens e de sinais (como por exemplo em um projeto de reconhecimento de placas de automóveis).

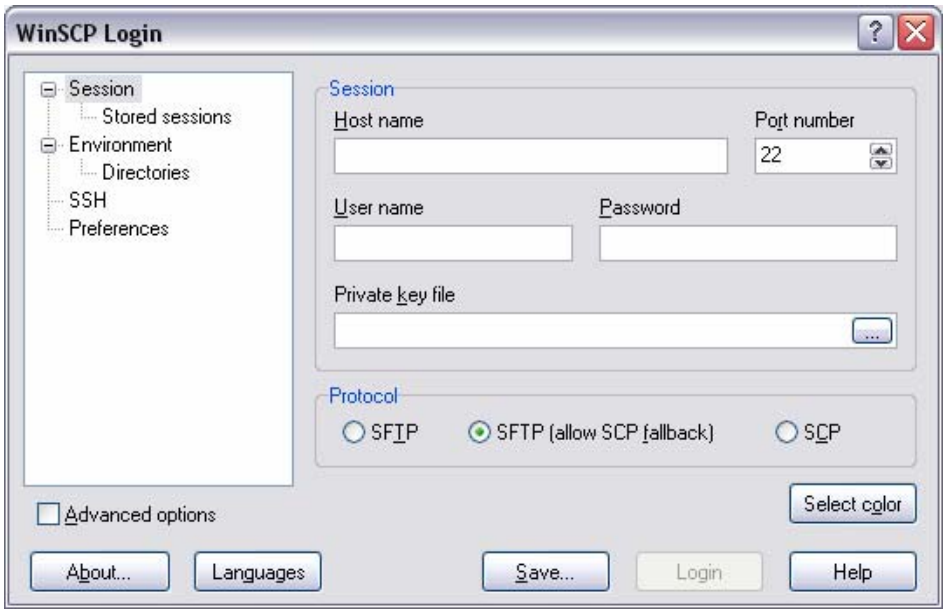
É importante ressaltar que o sistema de cluster é escalável, ou seja, é possível incluir outros computadores para partilhar da execução das tarefas caso seja necessário.

O enfoque desta nota técnica será feito na utilização do cluster no CBPF para a execução de programas feitos no MATLAB.

2.Transferência de Arquivos

Para que o usuário possa conectar-se ao cluster do CBPF ele deverá possuir um cadastramento que pode ser feito entrando em contato com o **CBPF/CAT**.

Após a criação de sua conta no SSOLAR, serão possíveis transferências de arquivos entre o computador do usuário e o servidor do CBPF. Para este fim é sugerido o uso do programa **WinSCP** que pode ser obtido em http://winscp.net/eng/download.php, abaixo temos a interface inicial do programa.

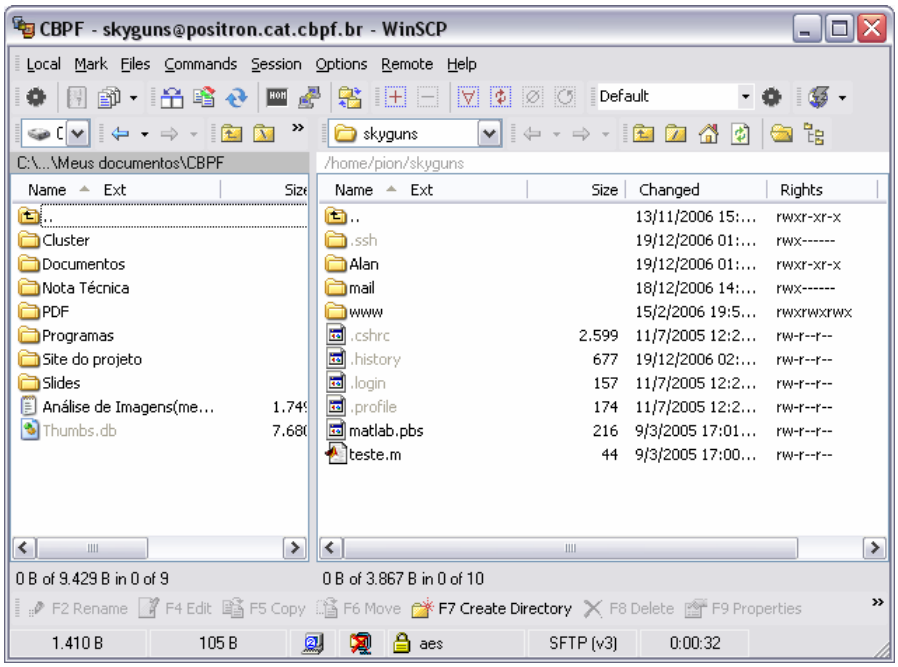


Transferência a partir de um computador dentro do CBPF.

No campo *Host name : ssolar.cat.cbpf.br.*
Utilizar a porta 22.
Os arquivos poderão ser transferidos diretamente pelo programa.

O nome de usuário e senha deverão ser fornecidos pelo CAT.

Após a conexão abaixo temos a interface do programa:



Na coluna esquerda do programa temos o seu computador e na coluna direita sua pasta no servidor do CBPF.

Os arquivos podem ser copiados diretamente do seu computador para o servidor e vice-versa.

Transferência a partir de um computador qualquer.

Os usuários que precisarem transferir arquivos para o SSOLAR a partir de um computador de fora do CBPF não poderão fazê-lo somente a partir do WinSCP, pois para conectar-se ao SSOLAR é necessário estar conectado na Positron, e o programa não permite que seja transferido arquivos da Positron para o SSOLAR, então teremos que usar o WinSCP para transferir os arquivos para a Positron e depois recorrer a alguns comandos básicos do **Linux** para transferi-los para o SSOLAR(veja o Apêndice no fim do artigo para maiores informações).

1) Conectar-se ao SSOLAR(ver o tópico número 4 deste artigo)
2) Crie uma pasta para sua aplicação : *mkdir* nome da pasta
3) Entre na pasta criada : *cd* nome da pasta
4) Copie os arquivos **.m** e **.pbs** da Positron para o SSOLAR : *scp usuário@positron.cat.cbpf.br:arquivo.extensao ~/destino*

~/destino significa que a pasta de destino esta na raiz.

Caso se deseje copiar todos os arquivos de uma só vez de uma determinada pasta na Positron, trocar arquivo.extensão por *origem/*.**

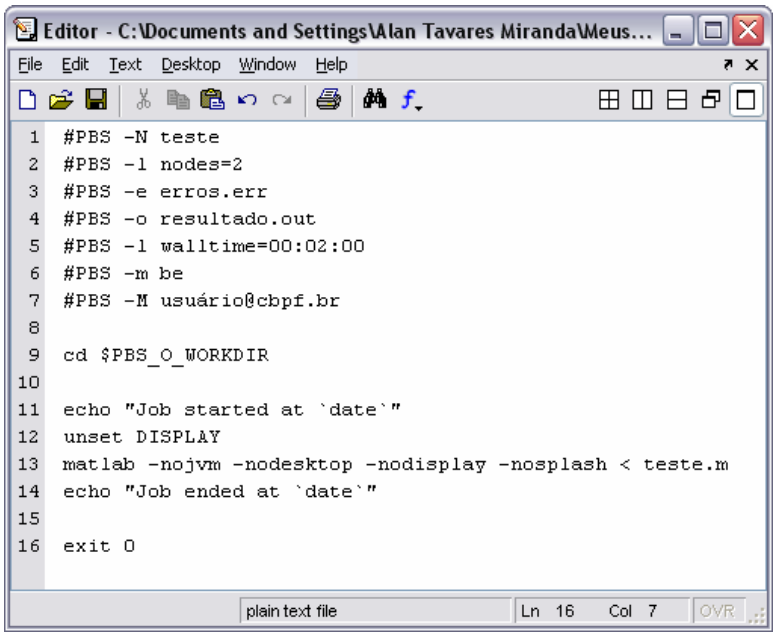
3.O arquivo PBS

O arquivo PBS é um script onde o usuário pode configurar as opções que o SSOLAR disponibiliza através do gerenciador de filas OpenPBS. É através do PBS que o usuário insere seu **job** para ser executado no servidor e determina arquivos de saída e notificações por exemplo.

● *Job é o programa na fila execução*.

3.1 Criando um script PBS utilizando o MATLAB

De forma simples podemos criar um arquivo PBS como no exemplo abaixo:



As linhas iniciais #PBS serão interpretadas pelo qsub como parâmetros diretos para o PBS. As demais linhas serão processadas como um shell script normal, o usuário pode colocar quaisquer comandos que deseje alem do programa principal.

Parâmetros do PBS

1) *–N* deve ser usado para especificar o nome do arquivo PBS.

2) *-1 nodes=x*, onde x é o numero de nós que deseja usar(cpu's).

3) *–e* especifica o arquivo que conterá os erros(caso existam)

4) *-o* especifica qual é o arquivo de saída.

5) *-1 walltime=HH:mm:ss* especifica um tempo de execução para o job em HH:mm:ss (horas, minutos e segundos).

***Nota**: Deve-se inserir um tempo superior ao esperado como margem de segurança, pois uma vez que a execução ultrapasse este tempo, o job será interrompido.*

Nota2: *O fato de o usuário especificar um tempo de execução otimiza o gerenciador de filas.*

6) *–m be* envia notificação ao iniciar e ao finalizar uma execução.

7) *–M e-mail* especifica para onde as notificações serão enviadas(o e-mail deverá ser especificado no lugar da palavra e-mail).

9) *cd $PBS_O_WORKDIR* direciona o ponteiro para a área de trabalho do usuário no servidor.

11) *echo "Job started at `date`"* comando que guarda a informação de quando o job iniciou a execução.

12) *unset display* é um comando que deve ser utilizado quando se deseja iniciar remotamente o MATLAB através de um terminal e iniciar a execução de um programa, e poder sair do terminal sem finalizar a execução do programa.

13) *matlab -nojvm -nodesktop -nodisplay -nosplash < teste.m* linha de comando que especifica qual é script que contem o programa do Matlab.

14) *echo "Job ended at `date`"* guarda informações de quando o job terminou a execução.

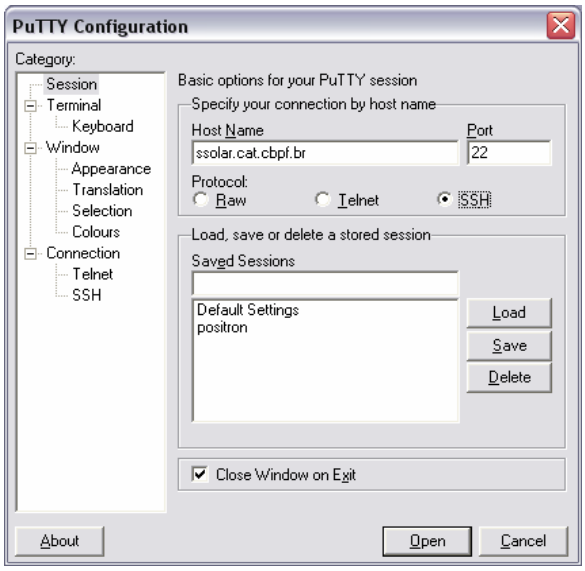
16) *exit 0* é usado para finalizar o arquivo PBS.

Nota3**: Caso seja importante armazenar as mensagens de erro e as saídas num só arquivo deve-se excluir a linha de comando que especifica o arquivo de erros e inserir a seguinte : #**PBS –j oe

4.Conexão ao Cluster

Após a criação do seu programa e o respectivo arquivo PBS, o usuário deverá acessar o cluster para manipular a execução de seu programa. Para este fim é sugerido o uso do programa **PuTTY** que pode ser obtido gratuitamente em http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html

Abaixo segue a configuração do programa para utilização dentro do CBPF.



Utilize o protocolo **SSH**(a porta 22 será automaticamente selecionada)

Após clicar em open surgirá um ***prompt*** onde deverá ser inserido o login e a senha do usuário no servidor.

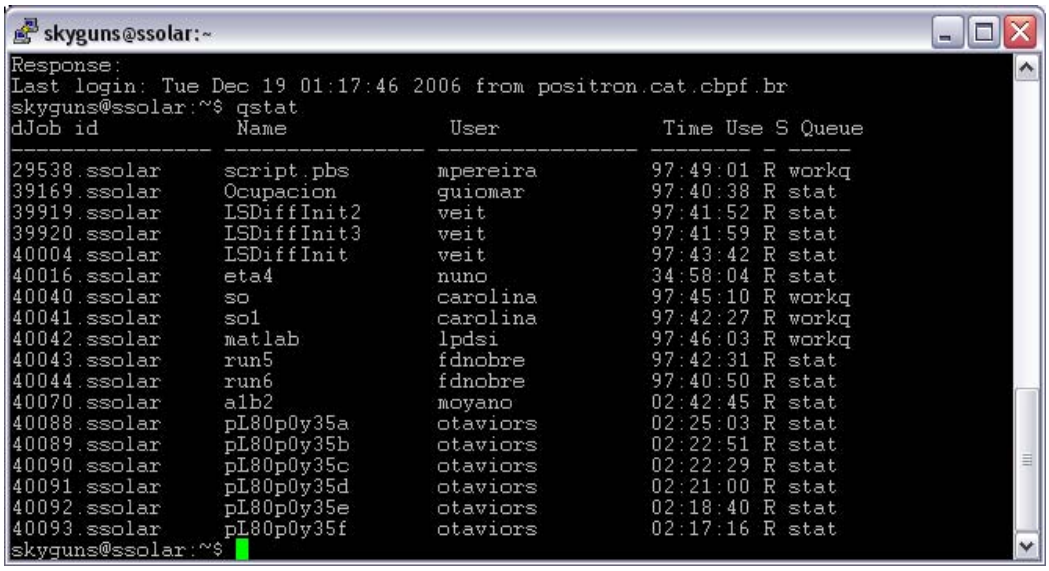
Caso o usuário queira conectar-se a partir de um computador fora da rede do CBPF, deverá ser usado *o host name : positron.cat.cbpf.br.*

Após clicar em open o usuário deverá inserir seu login e senha de **e-mail,** digitar o comando ssh ssolar.cat.cbpf.br -l loginname para conectar-se ao SSOLAR(onde o usuário insere seu login no **servidor**).

Para finalizar digite sua senha do SSOLAR.

4.1 Comandos PBS Básicos

Comando	Descrição
qsub nome.pbs	Submissão do script nome.pbs
qstat	Mostra o status de todos os jobs já submetidos
qstat –fid	Mostra status detalhado do job id
qdel id	Encerra execução do job id



Acima temos uma imagem do prompt onde são mostrados os jobs em execução e em espera no momento

Stat – Job em execução.

Workq – Job em espera.

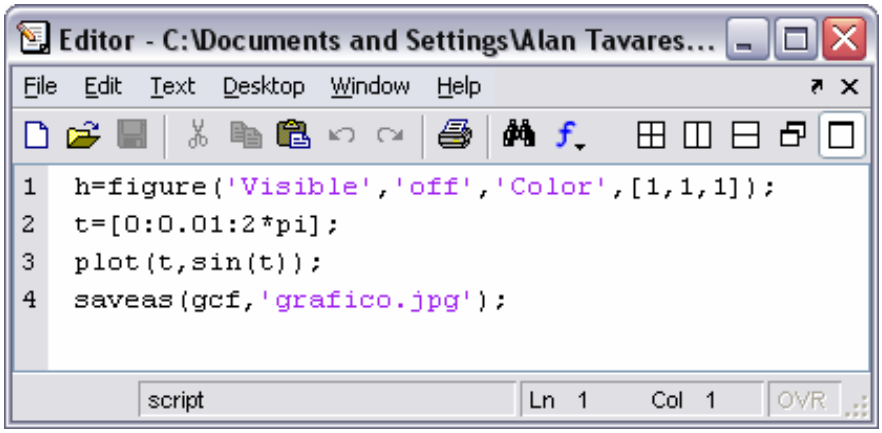
5. Salvando Gráficos

Construindo gráficos diretamente no ambiente do programa MATLAB, é possível exportá-lo para arquivo facilmente através da interface do programa, porém caso o usuário julgue necessário fazê-lo através de comandos inseridos em um script, é necessário conhecer alguns códigos que permitam que essa tarefa seja realizada diretamente.

Antes da construção do gráfico propriamente dito, é interessante apresentarmos sucintamente o conceito de *handle*.

Handle: *É um valor numérico que é atribuído a um gráfico ou imagem quando este é alocado na memória.*

Ex. H=plot(x,y). H é o handle atribuído ao gráfico construído da função y e do vetor x que devem estar previamente declarados (consultar o apêndice II).



1) O comando *figure cria um espaço destinado ao gráfico.*

a) *Visible – off: faz com que o espaço criado não seja mostrado na tela, apenas criando o* handle.

b) Color – vetor: preenche o espaço criado com uma cor (procedimento necessário ao modo não visível); a cor é especificada por um vetor, no caso descrito no exemplo [1,1,1] é a cor branca.

2) Variável livre declarada como vetor.

3) *plot: constrói o gráfico a partir dos pontos t e seno de t*

4) saveas: exporta o gráfico para arquivo, onde gcf chama o handle *atual.*

- *É possível adicionar outros parâmetros a função* figure*, como por exemplo 'title' tendo como próximo argumento o título do gráfico. Para maiores informações consultar a ajuda do Matlab.*

6. Exportando dados (variáveis) para arquivo

Em muitos casos é interessante exportar dados para construir gráficos para que seja possível uma melhor manipulação dos mesmos ao invés de exportar o gráfico já construído. Para este fim apresentamos dois comandos úteis

save('file.mat')- salva todas as variáveis já declaradas ate a linha atual para o arquivo file.mat

save('partial.mat',c) – salva a variável c para o arquivo partial.mat.

save fname contrast* : salva para o arquivo fname todas as variáveis declaradas até o momento que comecem com o termo contrast.

Para ler o arquivo das variáveis use
load('filename.mat'): carrega o arquivo de dados filename.mat para o Matlab.

APÊNDICE I – Comandos básicos do Linux

Comando	Função
clear	Limpa a tela
ls	Lista diretório atual

ls –l	Lista arquivos em formato detalhado
ls –a	Lista arquivos ocultos
ls –r	Lista também os subdiretórios encontrados
cd	Entra e sai de diretórios
cd ..	Diretório anterior
cd ~	Diretório HOME do usuário
cd -	Último diretório
pwd	Mostra o caminho completo do diretório atual
mkdir	Cria diretório
rm	Apaga arquivo
rm –r	Apaga diretório vazio
rm –rf	Apaga arvore de diretórios
mv	Renomeia arquivo ou diretório
rmdir	Apaga diretório vazio
cp	Copia arquivo
cp –v	Mostra o que esta sendo copiado
cp –r	Copia todo o diretório incluindo os subdiretórios
more/less	Mostra conteúdo de arquivo paginando
Os	Lista processos
kill	Finalizar um processo
unzip	Descompacta arquivos .zip

gzip	Compacta arquivo para extensão .gz
du –hs	Mostra espaço ocupado pelo diretório atual
df –h	Mostra espaço de todas as partições
uptime	Mostra quanto tempo o sistema esta rodando
uname -a	Mostra informações de versão do kernel
mcedit	Edita arquivo
Date	Mostra data
telnet	Conecta em portas remotas
ftp	Cliente ftp
ping	Testa conectividade entre hosts
scp	Copia arquivos ou diretórios entre computadores remotos

APÊNDICE II – Geração de Vetores

Definição: Vetor é um elemento que armazena valores, podendo ser um vetor linha ou coluna. Matriz é um conjunto de vetores especialmente disposto.

Geração de Vetores

O operador ‘:’ indica da onde ate aonde

v=[a:b] – gera vetor com valores de a até b espaçados de 1.

v2=[a:c:b] - gera vetor com valores de a até b espaçados de c.

v3=linspace(a,b,n) – gera vetor linearmente espaçado de a até b com n elementos.

APÊNDICE III – Outras formas de construção de gráficos

Gráficos 2D

ezplot('x(t),y(t)') - plota função 2D parametrizada ou não sem intervalo definido.

ezplot('x(t),y(t)'),[a,b],[b,c]) - função 2D parametrizada ou não com intervalo definido.

* A função deve ser escrita de forma implícita ou explicita não escrevendo, neste caso, a variável dependente.

ezpolar('h(t)') - plota gráfico em coordenadas polares; precisa de uma única função.

fplot('[f(x),g(x)]',[a,b]) - plota vários gráficos num único plano cartesiano. É necessário especificar um intervalo de domínio.

Gráficos 3D

ezplot3('x(t)','y(t),'z(t)') – plota curva 3D parametrizada.

ezsurf('f(x,y)') - plot de superfície

Argumentos opcionais: *colorbar* – escala de cores

shading interp – retira linhas

shading faceted – retorna as linhas

* Ezsurf também pode plotar gráfico de funções parametrizadas.

ezsurfc('f(x,y)') - plota gráfico e curva de contorno.

ezmesh('f(x,y)') – plota gráfico da superfície com linhas.

ezmeshc('f(x,y)') - plota gráfico da superfície com linhas + curva de contorno.

ezcontour('f(x,y)') - curva de contorno.

surf([matriz]) – Utiliza plotagem através da informação de uma matriz de pixels.

Linhas de Dados

Sendo x,y,z e w vetores temos :

plot(x,y,'x') – plota pontos (não sob a forma de gráfico)

plot(x,y) - plota gráfico dos pontos

plotyy(x,y,z,w) – plota gráfico tendo-se referência num eixo coordenado à esquerda e outro a direita.

plot(y) – plota um eixo imaginário e um real de um número complexo.

APÊNDICE IV – Editor de texto VI

O VI é um editor de texto que pode facilitar a edição de pequenas alterações a serem feitas no código dos programas em matlab, arquivos.txt, etc. quando estes já estiverem no ambiente do cluster. Para editar um arquivo basta digitar a partir do Shell: $ **vi nome-arquivo.** Uma vez carregado o arquivo, várias modificações poderão ser feitas no mesmo.

Na tabela abaixo segue alguns comandos básicos muito utilizados. A tecla ESC do teclado indica o início de um comando a ser executado.

ALGUNS COMANDOS BÁSICOS PARA EDITAR O TEXTO
i = insere texto antes do cursor atual
a = insere texto depois do cursor atual
I = insere texto no início da linha
A = insere texto no final da linha
s = substitui texto no cursor atual
S = substitui texto depois do cursor atual
o = abre uma linha abaixo do cursor atual
x = deleta o caracter que está sobre o cursor

:wq = salva o arquivo e sai do editor
:w nome_arquivo = salva o arquivo corrente com o nome especificado.
:w! nome_arquivo = salva arquivo corrente no arquivo especificado.
:q = sai do editor
:q! = sai do editor sem salvar as alterações.

BIBLIOGRAFIA

MATLAB 7.0 Help

CBPF-NT-010/06

http://www.damtp.cam.ac.uk/computing/numerical/faq-matlab.html

http://www.planetarium.com.br/planetarium/noticias/2004/2/1076629936

http://www.devin.com.br/eitch/comandos_linux/

http://www.debianfordummies.org/wiki/index.php/Transferir_Arquivos_com_SCP

2004 - Albuquerque, M. P., Albuquerque, M. P., Alves, N., Peixoto, D. e Maia, A. (2004). “Computação Científica no CBPF”, Anais do II Workshop de Grade Computacional e Aplicações.

2004 - CBPF-NT-001/04 - "Programação Científica Estruturada Linguagem C, Paralelização, Cluster/ Grids" Nilton Alves, Deyse M. Peixoto e Alexandre Maia.

2003 - Albuquerque, M. P., Albuquerque, M. P., Alves, N. e Peixoto, D. (2003). “Ambiente de Computação de Alto Desempenho do CBPF: Projeto SSolar”, Anais do I Workshop de Grade Computacional e Aplicações, pg.13-18