Tutorial LINUX

Tutorial SCILAB

Sistema Operacional Linux aplicado ao uso de Scilab

Existem algumas diferenças na utilização do Scilab a partir do Linux e a partir do Windows, como o Windows é um sistema a que em geral estamos mais familiarizados serão discutidos alguns pontos da utilização do Linux.

Para iniciar:

Login: `PME2371` (em alguns computadores o login é `pme2371`)
Password: `modelagem`

Utilizando disquetes

1. Para utilizar disquetes em Linux é necessário "avisar" ao sistema, através do comando mount, através do ícone "floppy" da tela. Para isso basta um duplo clique no mesmo com o botão da esquerda ou selecionar a opção mount clicando com o botão da direita. Ao fim da utilização o inverso, agora é necessário fazer o "Unmount" no mesmo ícone, utilizando o botão da direita e selecionando o comando.
2. Repare que o sistema de direcionamento para o arquivo é um pouco diferente, a linha de comando para o arquivo que no Windows seria (A:\diretório\arquivo). No Linux fica, por exemplo.,para o drive de disquetes (\mnt\floppy\arquivo). É necessário então um certo cuidado para salvar ou apontar arquivos no Linux.

Intercâmbio de Arquivos

O Linux possui alguns programas que fazem o intercambio com arquivos do Windows, como o OpenOffice com o Microsoft Office.

Editores de texto para Macros

1. Na utilização de editores de texto para a programação de funções para o Scilab existe um cuidado adicional, quanto a sinais e formatações do texto, sendo diferente x^2 de x^2, geralmente para obter x^2 devemos digitar duas vezes o acento circunflexo (digitando x^^2). Em alguns computadores a ação pode ser digitar espaço após digitar o acento circunflexo. Entre os editores de texto, podemos utilizar o Kwrite.
2. Copiar e colar: selecione o trecho de interesse com o botão esquerdo do mouse. Mantendo a seleção, vá para a janela do Scilab. Para colar posicione o mouse no local onde se quer colar e aperte o botão do meio do mouse. Entretanto, quando queremos executar algumas vezes uma mesma função mais extensa e complexa, pode ser vantajoso utilizar um editor de texto e salvar uma função com extensão sce ou sci e executa-la a partir do Scilab.
3. A versão 4.1 do Scilab conta com um editor de texto inserido em sua janela (SciPad), mais prático, mas que não impede o uso de outros editores.

Nomeando arquivos

Não use nomes com espaços em branco nem com acentos ou o cedilha.

Tutorial Prático do Scilab

Notas importantes

-O Scilab diferencia letras maiúsculas e minúsculas.
-É recomendável a não utilização de acentos no Scilab.
-Existe a função Help no menu, que também pode ser visualizada utilizando a sintaxe "help função".
-Algumas constantes já são definidas e podem ser visualizadas através do comando WHO, algumas vem precedida pelo símbolo de porcentagem (%).
-Ex. %pi, %e, %i.
-As linhas de comando só são executadas quando é acionado o Enter.
-As funções básicas como raiz (sqrt) e determinante (det) entre outras já são definidas.
-Quando colocamos ponto e virgula (;) ao final da expressão, seu resultado não é mostrado.
-Ao importar funções de editores de textos, é necessário que não existam formatações, como por exemplo, x^2 que não deve ser escrito dessa forma, mas sim como x^2. Caso contrário o scilab não reconhecerá a função.
-Para inserir um comentário que não será avaliado pelo Scilab utilizamos duas barras invertidas (//) no início da linha.

Trabalhando com constantes

Definindo

-Um elemento é definido quando lhe é atribuído um valor.
Ex. a=7

Utilizando elementos já definidos

-Caso já tenhamos definido um elemento podemos utilizá-lo para definir outro.
Ex. c= 9
a=3*c+8 obtenho a=35.

Números Complexos

-Da mesma forma posso trabalhar com números complexos.
Ex. f=3+%i
a=sqrt(-1)
b=f+2*a obtenho b=3+3i

Vetores e matrizes

-Um vetor ou matriz é criado e se comporta como os exemplos vistos até aqui utilizando a sintaxe "matriz=[linha 1;linha 2;....;linha n,]" utilizando espaços entre os elementos da linha e ponto e virgula (;) entre as linhas.
Ex. A=[1 2 3; 4 5 6; 7 8 9]
B=[a+b 9 1] desde que a e b estejam previamente definidos.

Operações

-Operar com as matrizes é similar a operar com constantes. Posso ainda, por exemplo, multiplicar matrizes da forma tradicional ou multiplicar elemento a elemento da matriz, utilizando ponto (.) após o nome da Matriz.
Ex. C=B.*M com B e M definidos faz a multiplicação de cada elemento

Tipos de matrizes – zeros, uns, diagonais, identidades

-Algumas matrizes tradicionais podem ser criadas através de funções como zeros (zeros), uns (ones). Estas funções resultam em uma matriz com uma dada dimensão.

Ex. A=zeros(2,3) fornece uma matriz de zeros com 2 linhas por 3 colunas.

B=ones(A) fornece uma matriz de uns com a mesma dimensão de A.

-Uma matriz diagonal pode ser criada através da sintaxe "diag(V)", sendo V o vetor com os valores da diagonal.

Ex. C=diag(V) cria uma matriz diagonal com os elementos de V.

D=diag(ones(1,4)) cria uma matriz identidade de dimensão 4X4.

Características de uma Matriz

-Podemos obter características de uma matriz através de operações como as de autovalores e autovetores (spec), traço (trace), determinante (det), rank (rank) ou matriz inversas (inv) com a mesma sintaxe: "função(matriz)".

Ex. spec(A) retorna os autovalores da matriz A.

[w,v]=spec(A)

v recebe a matriz diagonal cujos elementos são os autovalores de A, e w é uma matriz cujas colunas são os autovetores de A

Vetor com passo

-Podemos criar um vetor de tempo, por exemplo, que possua um determinado passo, definindo valores iniciais e finais para o mesmo, através da sintaxe "V=(inicial:passo:final)".

Ex. V=(2:0.1:10) gera o vetor iniciado em 2 e teminado em 10, com passo 0.1.

Vetor logaritmico

-A função logspace cria um vetor de espaço entre duas potências de dez, com um numero definido de elementos, espaçados de um mesmo fator (em progressão geométrica). A sintaxe fica "logspace(inicial,final,numero de elementos)".

Ex. v=logspace(3,4,10) gera um vetor logarítmico com dez elementos entre 10^3 e 10^4.

Extraindo elementos de uma matriz

-Pode-se definir um vetor ou uma constante a partir de uma matriz utilizando a sintaxe "nome=matriz(linha,coluna)" sendo que dois pontos (:) engloba tudo e cifra ($) é a ultima.

Ex. a=A(:,$) gera um vetor com a ultima coluna da matriz A

b=A(3,4) gera uma constante com o valor do elemento da terceira linha e quarta coluna da matriz A.

Cálculo do valor de uma função definida por uma matriz de caracteres (string)

-Para retornar uma matriz valida para o Scilab a partir de uma matriz de caracteres (string) ou eventualmente avaliar sua validade temos o comando evstr, de forma "matriz=evstr(matriz-string)".

Ex. A=evstr(Z) onde A recebe o valor de uma matriz Z de caracteres, definidos.

Criando Polinômios

A partir dos coeficientes ou das raizes

-Utilizando a sintaxe "nome=poly([v], "variável", "processo") apresentamos um polinômio onde o processo pode ser coeff ou roots, sendo o vetor os coeficientes em coeff ou as raízes em roots.

Ex. p=poly([3 2 7], "x", "coeff") cria o polinômio $p=3+2x+7x^2$.

Extraindo raízes

-Para extrair as raízes de um polinômio definido utilizo a sintaxe "roots(nome do polinômio)".

Ex. roots(p1) com p1 definido, retorna as raízes de p1.

Valores Booleanos

-Podemos atribuir às variáveis valores booleanos %F (falso) ou %T (verdadeiro). Para comparações, por exemplo.

Ex. a=3

a>=1 obteremos a resposta T (verdadeiro)

Funções

Definindo a função

-Para definir uma função teremos de maneira mais simples a seguinte sintaxe "deff("[resposta]=nome(variável)","função")"

Ex. deff("[y]=func(x)","y=x^2+3*x+%i") criando a função complexa $y=x^2+3x+i$.

Calculando a função

-Podemos então obter o valor numérico da função definida da forma "nome(valor)"

Ex. deff("[y]=func(x)","y=x^2+3*x+%i") função já definida

func(%pi) retorna a função calculada em pi (3,14...)

Plotando

-Podemos plotar gráficos através da sintaxe "plot(variavel1,variavel2)" sendo as variáveis vetores de uma mesma dimensão. Uma possibilidade é plotar uma função em um intervalo de tempo, utilizando um vetor de tempo, definido na seção de vetor com passo, calculando em outro vetor a função naqueles instantes.

-Para criar novas janelas para plotar as funções utilizamos a sintaxe "xset("window",numero)" sendo os números 1, 2, 3...

-Se quisermos mudar as características da marcação no gráfico, podemos utilizar a sintaxe "plot2d(x,y,estilo)" onde estilo é um número e indica o tipo de marcação, asterisco por exemplo corresponde a –3, já a cor vermelha da linha corresponde a 5.

-Para adicionar títulos ao gráfico utilizamos a sintaxe "xtitle("titulografico", "titulox", "tituloy")".

-Podemos adicionar linhas de grades ao gráfico, através da sintaxe "xgrid(estilo)", onde estilo é um número, que define por exemplo linhas de cores diferentes na grade.

-Podemos subdividir uma janela em linhas e colunas e plotar um gráfico um uma das divisões com a sintaxe "subplot(linhas,colunas,divisão)".

Ex. xset("window",1) cria a janela numero 1.

Plot2d(x,y,-3) plota a função y(x) com asteriscos

xtitle(“titulo”, “eixox”, “eixoy”) adiciona títulos ao gráfico.
xgrid(3) cria linhas de grade verdes na área do gráfico.
Subplot(4,5,2) prepara a segunda divisão da área de plotagem de quatro linhas por cinco colunas.

Sistemas lineares

Gerando

-Um sistema linear é determinado definindo-se seu domínio, continuo (c) ou discreto (d) e sua representação, por meio da sintaxe “syslin(dominio,representação)”.

Ex. sistema: $g = \frac{s}{s+1}$, contínuo no tempo.

```
n=poly(0,'s','roots');
d=poly([1  1],'s','coeff');
g=syslin('c',n,d)
```

Ex. sistema: $\dot{\mathbf{x}} = \mathbf{Ax} + \mathbf{Bu}$
$\mathbf{y} = \mathbf{Cx} + \mathbf{Du}$

```
A=[0  1; -1 -2];
B=[0; 1]
C=[1  0];
D=0;
sistema=syslin('c',A,B,C,D);
```

Pólos e zeros

-Para plotar os pólos e zeros de um sistema linear utilizo a função plzr, na forma “plzr(sistema linear)”.

Ex. plzr(g) plota os pólos e zeros do sistema linear g definido.

Simulação

-A resposta da simulação do sistema linear a uma entrada no tempo com uma condição inicial é efetuada através da sintaxe “csim(entrada,tempo,função linear,condição inicial)”

Ex. [y,x]=csim(u,t,sl,x0) y recebe a resposta do sistema linear sl, à entrada u, no tempo t e condição inicial x0, e x recebe os estados do sistema.

Diagrama de Bode

-O diagrama de Bode, ou seja, magnitude e fase da resposta em freqüência do sistema linear, é obtido com a sintaxe “bode(sistemalinear,frequência)” onde a freqüência é dada em Hz.

Ex. bode(sl,frq) plota o diagrama de bode do sistema sl na freqüência frq, com frq um vetor de passo.

Vetor de resposta em freqüência do sistema

-A resposta em freqüência do sistema linear em cada ponto de um vetor de freqüência é calculada através da sintaxe “repfreq(sistemalinear,frequencias)”.

Ex. vf=repfreq(sl,frq) cria em vf o vetor de resposta em freqüência do sistema linear nas freqüências do vetor frq.

Magnitude e fase

-O retorno das magnitudes e fases do sistema linear é dada pelo comando “phasemag”.

Ex. [fase,mag]=phasemag(vf) retorna as fases e magnitudes de um vetor vf resposta de um sistema em freqüências.

Macros e Funções

-Objetivando executar rotinas uma alternativa importante é importar arquivos feitos em editores de texto para o scilab. A linha de comando deve ser criada e o arquivo salvo com a extensão sce (conjunto de comandos) ou sci (implementação de funções). Através do menu File – File Operations elas podem ser disponibilizadas para uso “Getf” (extensão sci) ou executadas “Exec” (extensão sce).

-As linhas de comando podem ter sintaxes como o laço for, if, end, while, function, deff, return, com símbolos de comparação como ==, >=, <=, |, =&, etc.

-A partir do “Getf” adicionamos uma função ao Scilab, e através do “Exec” executamos a linha de comando presente no arquivo, gerando uma resposta.

Exemplo de implementação de função (arquivo do tipo sci):

```
function [z]=funcao(y)
z=y+sin(y);
endfunction
```