

TUTORIAL

do

OCTAVE

GNU-Octave versão 2.1.42

Aluna: Camili Ambrósio

RA: 042426

Disciplina: MA111

Professor: Márcio Rosa

Segundo semestre 2005.

Glossário

O que é o Octave? Onde Obter?.....	2
Introdução.....	3
Como instalar.....	4
Como ligar e como sair.....	6
Como usar : As operações básicas.....	7
As Variáveis.....	10
As Constantes.....	12
Matrizes.....	13
Comando disp () e Funções	16
Fatorial, Limite e Integral.....	17
Como funciona os logs, salvar e editar.....	18
Gráficos.....	19
Gráfico 3D.....	21
Gráfico de superfície.....	28
Bibliografia.....	32

TUTORIAL do programa OCTAVE

O que é o Octave?

Resumidamente, é um software livre capaz de resolver cálculos numéricos e também pode ser usado como linguagem de programação de alto nível.

O GNU/Octave é uma linguagem de alto nível basicamente voltada para computação gráfica. Esse programa prove uma interface por linha de comandos – não há interface gráfica - para solução numérica de problemas lineares e, também, não-lineares e para implementar outros experimentos numéricos usando uma linguagem que compatível com o Matlab (que é um programa comercial). O programa pode ser utilizado também em modo script (textos de programação) e permite incorporar módulos escritos nas linguagens C++, C, Fortran e outras. O GNU/Octave foi escrito por John W. Eaton e muitos outros, estando disponível na forma GPL.

O GNU Octave utiliza o GNUPLOT.

Onde posso obter?

Na Internet em sites especializados na área de matemática, sites de professores e pesquisadores e também no site oficial do Octave. Segue abaixo alguns deles:

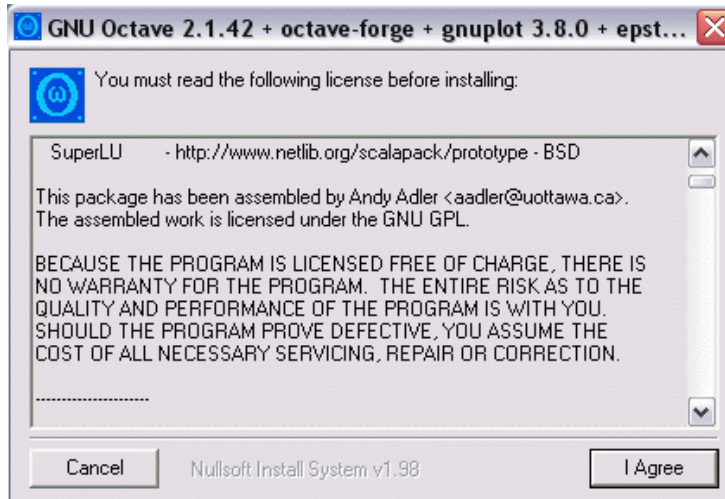
- Octave – <http://www.octave.org>
 1. mais precisamente em : <http://www.octave.org/download.html>
- GNUPlot – <http://www.gnu.org>

INTRODUÇÃO

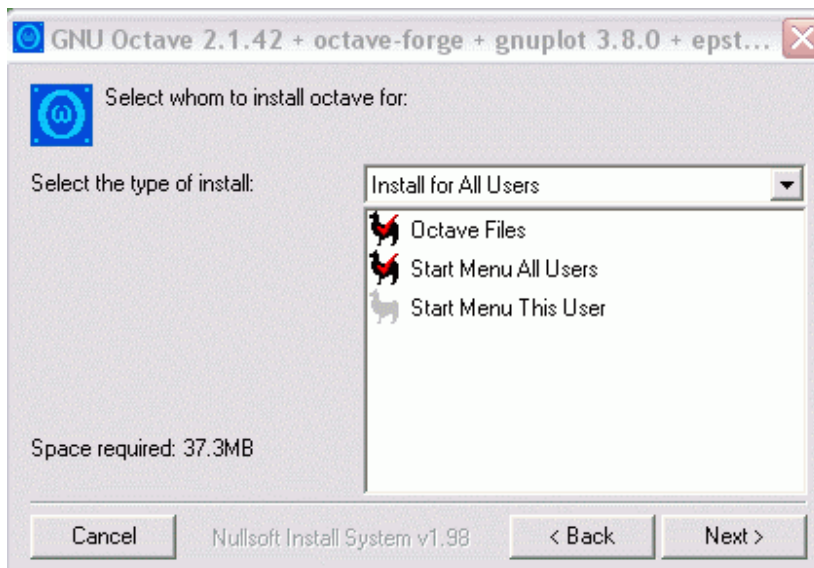
Este tutorial visa mostrar alguns comandos e exemplos de exercícios que podemos resolver com a ajuda deste programa Octave. No meu caso, instalei o GNU/Octave 2.1.42 - disponível no site do Octave acima citado – no Windows XP, mas poderia ser instalado em outros sistemas operacionais também, inclusive no Linux – onde geralmente é utilizado, pois aí já se utiliza softwares livres desde o sistema operacional. As explicações de instalação e para abrir o programa serão feitas a partir do Windows XP, lembrando que no Linux pode-se abrir o programa (depois de instalado) pelo terminal. Mas nosso objetivo é mostrar o programa. Veja na bibliografia ao final deste Tutorial para ver sites em que ensinam como instalar em outros sistemas operacionais.

Como instalar

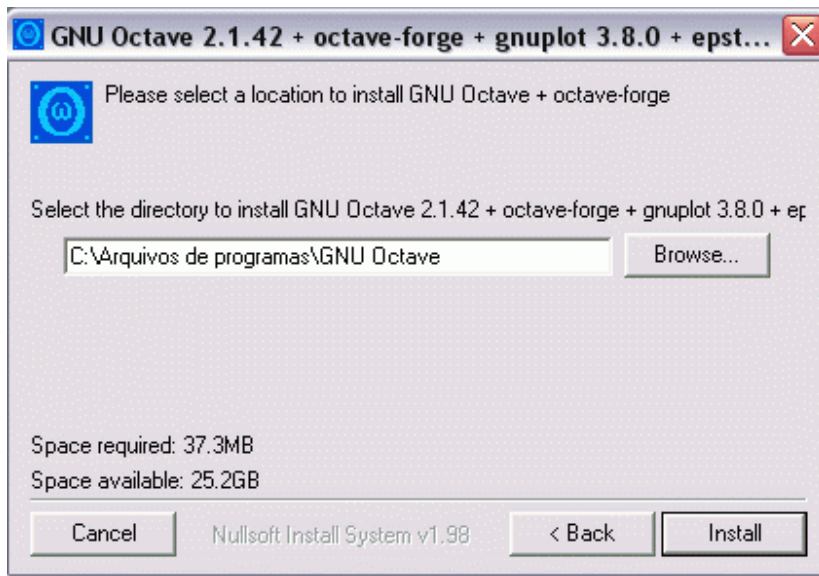
Após ter feito o download do programa, procurar na pasta em foi salva em seu computador e clicar no arquivo. Na instalação, após ler a licença, clique em “I Agree”.



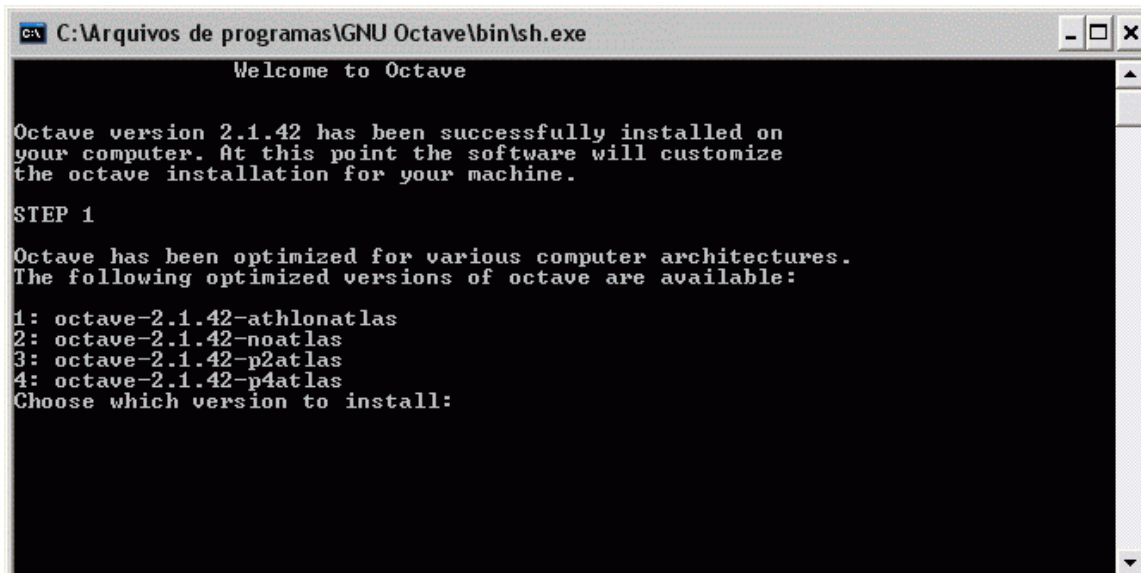
A seguir aparecerá a tela abaixo. Selecione o tipo de instalação mais adequada e clique em “Next”.



Agora selecione o diretório de seu computador que deseja que o Octave seja instalado e clique em Install.



Aparecerá a seguinte tela:



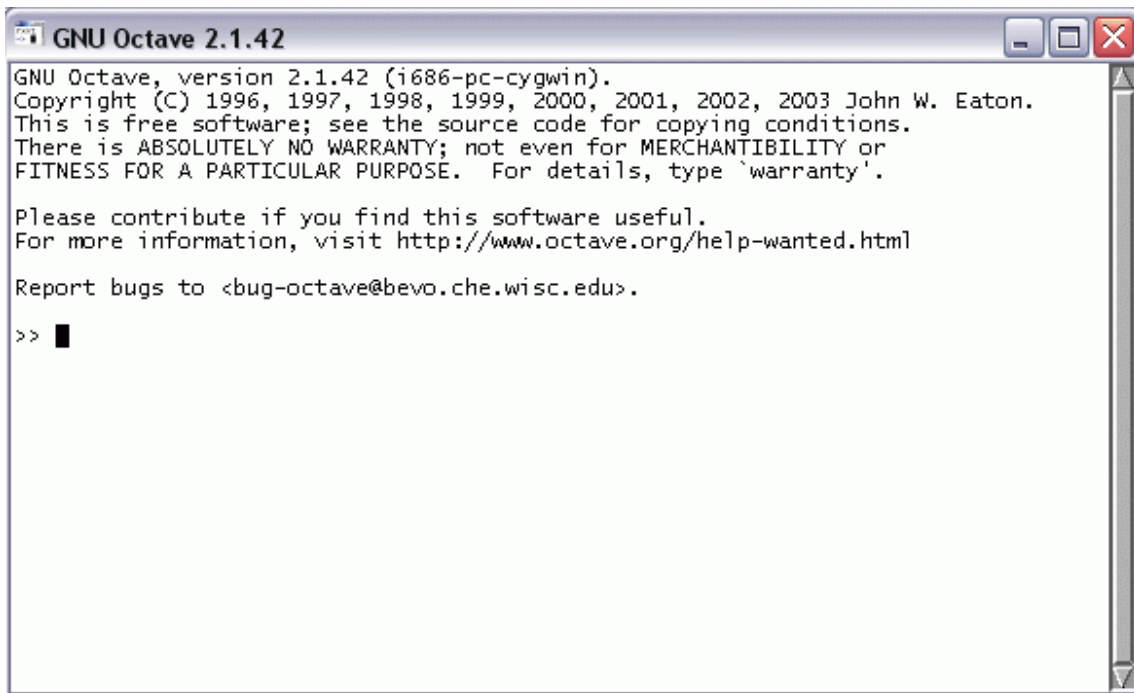
Escolha o número da versão que queira e aperte enter. Será instalado o Octave.

Como ligar o programa

Selecione o arquivo no diretório que você escolheu acima. Talvez o programa esteja também no menu ‘iniciar’ do seu computador e pode clicar no programa por lá:



Aparecerá a seguinte tela:

A screenshot of the GNU Octave 2.1.42 terminal window. The window title is 'GNU Octave 2.1.42'. The text displayed in the terminal is:

```
GNU Octave, version 2.1.42 (i686-pc-cygwin).
Copyright (C) 1996, 1997, 1998, 1999, 2000, 2001, 2002, 2003 John W. Eaton.
This is free software; see the source code for copying conditions.
There is ABSOLUTELY NO WARRANTY; not even for MERCHANTABILITY or
FITNESS FOR A PARTICULAR PURPOSE.  For details, type `warranty'.

Please contribute if you find this software useful.
For more information, visit http://www.octave.org/help-wanted.html

Report bugs to <bug-octave@bevo.che.wisc.edu>.

>> █
```

Como sair

No programa, escreva “exit” ou “quit” (somente o que está entre aspas, ou seja, sem aspas). E para ‘quebrar’ linha (geralmente quando digitou algo errado), aperte ctrl+c e estará já na nova linha!

Como usar

Agora que já instalamos e sabemos ligar o programa, precisamos utilizá-lo! Vamos então aprender a mexer com o GNUOctave. Veremos alguns comandos e tentaremos fazer alguns exercícios.

OS COMANDOS

As operações básicas

Para somar, subtrair, multiplicar e dividir, podemos fazer de tal modo:

Operação	Símbolo
Somar	+
Subtrair	-
Multiplicar	*
Dividir	/

```
>> 1+1
ans = 2
>> 2+2
ans = 4
>> 10-4
ans = 6
>> 6*4
ans = 24
>> 24/12
ans = 2
>> (1+3)-(9*0)
ans = 4
>> █
```

No programa, vemos assim “>>” para entrada de dados (o que digitamos, ou melhor, é o *prompt* do GNU Octave) e “ans = ” (de ‘answer’ em inglês, que significa resposta.), ou seja, a saída que o programa nos dá: o resultado (veja figura ao lado). Quando tiver o sinal “>>” você pode digitar. Por exemplo: “1+1” e, em seguida, aperte ‘enter’ para o programa rodar o seu comando.

O retângulo preto na figura ao lado é como o programa deixa para você ir escrevendo.

Mais operações

Operação	Comando
raiz ($\sqrt{\quad}$)	sqrt(x)
potência	x**y
conjugado	x'
Aumentar de um em um	x++

Operação	Comando
Decrementar de um em um	x--
Comentário	% ou #
Exponencial	exp ()
Logaritmo natural	log ()

Funções pré-definidas

Números Complexos	
Conjugado de x	conj(x)
Parte imaginária de x	imag(x)
Parte real de x	real(x)
Valor absoluto	abs(x)
Argumento	arg(x)

Trigonometria
sin(x) : Seno de x
cos(x) : Cosseno de x
tan(x) : Tangente de x
asin(x) : Seno inverso de x
acos(x) : Cosseno inverso de x
atan(x) : Tangente inverso de x
sinh(x) : Seno hiperbólico de x
cosh(x) : Cosseno hiperbólico de x
tanh(x) : Tangente hiperbólico de x
asinh(x) : Seno hiperbólico inverso de x
acosh(x) : Cosseno hiperbólico inverso de x
atanh(x) : Tangente hiperbólico inverso de x

Arredondamento:
floor(x) : Arredonda x para baixo
ceil(x) : Arredonda x para cima
round(x) : Arredonda x para o inteiro mais próximo

Vetores e Matrizes
length: Tamanho de um vetor
size: Dimensão de uma matriz
reshape: Muda a dimensão de uma matriz
zeros: Preenche uma matriz com Zero
ones: Preenche uma matriz com Um
eye: Matriz-Identidade
linspace: Vetor com elementos espaçados linearmente
logspace: Vetor com elementos espaçados logaritmicamente
prod: Produto dos elementos de um vetor
sum: Soma dos elementos de um vetor
sumsq: Soma do quadrado dos elementos de um vetor

Álgebra Linear:
det: Determinante de uma matriz
inv: Matriz inversa
rank: Matrix rank (?)
eig: Eigenvalues (?)
svd: Decomposição em valores singulares

Algumas aplicações destas funções:

```
>> sin(pi/6)
ans = 0.5000000000000000
>> x=2
x = 2
>> y=3
y = 3
>> x**y
ans = 8
>> log 1
parse error:

>>> log 1
      A

>> log (1)
ans = 0
>> cam = 9.21987
cam = 9.219870000000000
>> round(cam)
ans = 9
>> floor(cam)
ans = 9
>> ceil(cam)
ans = 10
>> █
```

Observação importante: você pode digitar “;” (Ponto-e-vírgula) após uma certa linha de comando. Isto serve para o programa não dar a resposta do ‘cálculo’ na tela. Ele apenas armazena na memória dele a informação. E, para exibir o valor de uma variável, basta escrever o nome da variável desejada que o Octave exibirá o seu valor logo abaixo.

As variáveis

As variáveis são criadas por você, durante o uso deste programa, e podem assumir valores numéricos (ou seja, reais ou complexos), matrizes, vetores, strings (seria um ‘vetor de caracteres’) entre outros. Para criar variáveis basta digitá-la no programa, assumir um valor para ela e pronto! Veja uns exemplos:

```
>> a = 3
a = 3
>> a
a = 3
>>
>> a = 3;
>> a
a = 3
>> a;
>> █
```

Na figura ao lado, criamos variáveis e omitindo a resposta do programa.

```
<<
>> c = 9 + 5i;
>> c
c = 9 + 5i
>> a = 9;
>> b = 34235;
>> c = a + 2b;
parse error:
>>> c = a + 2b;
      ^
>> c = a + 2*b;
>> c
c = 68479
>> █
```

Na figura ao lado, criamos uma variável $c = 9 + 5i$ (um número complexo) após declaramos a e b e $c = a + 2b$. Percebe-se que apareceu um erro. O Octave indica (^) em qual lugar está o erro. E foi a falta de asterisco (*) pra indicar a multiplicação de $2*b$.

Então, este programa não aceita a omissão deste sinal de multiplicação.

Agora, mexendo com strings. A declaração para uma variável string é diferente do que pra uma numérica. Veja:

```
>> %agora declararei uma variavel tipo string
>> nome = camili;
error: `camili' undefined near line 12 column 8
error: evaluating assignment expression near line 12, column 6
< Esta declaracao eh diferente das variaveis a, b, c.
>> nome = "camili";
>> nome
nome = camili
>> nome++;
>> nome
nome =
    100    98    110    106    109    106
```

Quando tentei declarar a variável nome como “ nome = camili; ” , não deu certo pois para declarar uma string é necessário o uso de aspas no começo e final da string, como em : nome = “camili”; . Se quisermos, podemos usar uma string como “um número”. Assim como fiz em ‘nome++;’ o símbolo ++ é para adicionar uma unidade na variável, mas como ela é um vetor de caracteres, o programa vai e usa os valores em ASCII (American Standard Code for Information Interchange) que cada caractere possui. Então:

Caractere	“Valor do caractere”	Valor + 1
c	99	100
a	97	98
m	109	110
i	105	106
l	108	109
i	105	106

E quando pedimos, após o comando nome++, para mostrar o valor de ‘nome’, ele apareceu uma matriz (ou vetor) linear (1 linha x 6 colunas) com os valores do caractere acrescido de um.

As constantes

Constantes são variáveis que o Octave cria toda vez que é inicializado. É possível substituir o valor de uma constante caso seja necessário. Alguns exemplos de constantes são : pi , e, infinito, verdadeiro (true), falso (false). Agora, no programa:

```
>> e
e = 2.7183
>> pi
pi = 3.1416
>> inf
inf = Inf
>> true
true = 1
>> false
false = 0
>> True
error: `True' undefined near line 21 column 1
>> False
error: `False' undefined near line 21 column 1
>> Inf
Inf = Inf
>> █
```

Observe que há diferenças entre maiúsculas e minúsculas para algumas constantes. Como ‘true’ e ‘True’. Algumas versões de Octave utilizam ‘true’ e outras utilizam ‘True’. Há também o ‘inf’ que é aceito com maiúscula ou minúscula em sua inicial, que é o infinito.

```
>> pi
pi = 3.1416
>> e
e = 2.7183
>> format long
>> pi
pi = 3.14159265358979
>> e
e = 2.71828182845905
>> format short
>> pi
pi = 3.14
>> e
e = 2.72
>> format
>> e
e = 2.7183
>> pi
pi = 3.1416
```

Uma coisa interessante é que você pode escolher a quantidade de casas que você quer que apareça quando digitar a constante, ou melhor, você pode formatar o tipo da constante.

Assim, vemos que há o *format long*, que apresenta 15 casas decimais. O *format short*, que aparece duas casas decimais, o *format* que nos dá 4 casas após a vírgula e, também, há *format bank*, com duas casas após a vírgula e com números complexos, não aparece a parte imaginária. Mais algumas constantes:

```
>> eps
eps = 2.2204e-16
>> format long
>> eps
eps = 2.22044604925031e-16
>> realmax
realmax = 1.79769313486232e+308
>> real min
parse error:

>>> real min
      ^

>> realmin
realmin = 2.22507385850720e-308
```

Matrizes

Para criar um vetor ou uma matriz, basta você inserir os valores destes entre colchetes: []. Sendo que valores na mesma linha são separados por vírgula, e valores na mesma coluna são separados com ponto-e-vírgula. Abaixo criei as matrizes $M_{1 \times 3}$, $N_{3 \times 1}$ e $O_{3 \times 3}$.

```
>> M = [1,2,3]
M =
     1     2     3

>> N = [1; 2; 3]
N =
     1
     2
     3

>> O = [1,2,3 ; 4,5,6 ; 7,8,9]
O =
     1     2     3
     4     5     6
     7     8     9

>> █
```

Para selecionar um valor de dentro de uma matriz, devemos inserir seu nome(como M, N, O), seguido pela posição do elemento dentro de parênteses : (). Para localizar qual valor são as seguintes posições: matriz M (1,3), matriz (2,1) e matriz O (2,2).

```
>> O(3,:)
ans =
     7     8     9

>> O(:,3)
ans =
     3
     6
     9

>> O(end,end-1)
ans = 8
>> █
```

```
>> M(1,3)
ans = 3
>> N(2,1)
ans = 2
>> O(2,2)
ans = 5
>>
```

Também podemos ver os elementos todos de uma certa linha ou coluna. Utiliza-se o “:” para exibir todos os elementos daquela linha ou coluna. E também existe o comando ‘end’ na matriz, que significa a última posição. (no caso da matriz O, é o 3) e também pode ser usado referenciais do ‘end’, que são relativos ao final da matriz, como o end-1. Veja ao lado (esquerda). O(end,end-1) é equivalente a dizer O(3,3-1) que é O(3,2).

Podemos adicionar, subtrair, achar a transposta de matrizes, entre outras operações.

```
>> A = [1,2;2,1];
>> B = [2,1;1,2];
>> A+B
ans =

     3     3
     3     3

>> A*B
ans =

     4     5
     5     4

...
>> transpose(A)
ans =

     1     2
     2     1

>> C=[1,2;3,4];
>> transpose(C)
ans =

     1     3
     2     4
```

Vimos então que para somar matriz A com a B , basta digitar : $A + B$, para multiplicar é $A*B$ e para mostrar a transposta é o comando $transpose(C)$ ou C' , por exemplo.

Para multiplicar uma matriz $M = [1,2,3]$ por um escalar n , como $n=3$, fazer: $3*M$:

```
>> 3*M
ans =

     3     6     9
```

Para achar o determinante duma matriz, como a matriz O citada acima, utilizar comando $det(O)$. Veja abaixo o $det(O)$ e, também, a transposta da matriz O feita de duas maneiras diferentes:

```
>> det(O)
ans = 0
>> O'
ans =

     1     4     7
     2     5     8
     3     6     9

>> transpose(O)
ans =

     1     4     7
     2     5     8
     3     6     9
```

Também há comandos prontos para matrizes. Tais como:

1. `ones(N,M)` para construir uma matriz $N \times M$ com elementos de valor 1.
2. `zeros(N, M)` para construir uma matriz $N \times M$ com elementos de valor 0.
3. `eye(N,M)` para construir uma matriz com elementos de valor 1 na diagonal.
4. `rand(N,M)` para construir uma matriz $N \times M$ com elementos de valor.

Veja:

```
>> ones(3,3)
ans =
     1     1     1
     1     1     1
     1     1     1

>> zeros(3,3)
ans =
     0     0     0
     0     0     0
     0     0     0

>> eye(3,3)
ans =
     1     0     0
     0     1     0
     0     0     1

>> rand(3,3)
ans =
    0.323272615671158    0.480424463748932    0.049759462475777
    0.466094881296158    0.325780987739563    0.124100692570210
    0.477467060089111    0.266319215297699    0.591843783855438
```

E se eu pedir para fazer uma matriz(2,4) (ou seja, não é uma matriz quadrada) e completá-la com 1 na diagonal? (comando `eye(2,4)`). Observe abaixo o que encontramos:

```
>> eye(2,4)
ans =
     1     0     0     0
     0     1     0     0
```


O Comando disp()

O comando disp() serve para exibir somente o valor final da entrada do usuário.

Fica melhor de entender ao visualizar:

```
>> x=9;
>> y = 5;
>> x
x = 9
>> y
y = 5
>> disp(x)
9
>> disp(x+y)
14
>> disp(1+1+3)
5
```

Aqui declarei o $x=9$ e $y=9$ e em seguida pedi para mostrá-los e o programa deu a seguinte saída: $x = 9$, por exemplo. Usando este comando disp(), vemos que com disp(x), ele apenas dá a saída o valor do que está entre parênteses (no disp()). Pode ser com variáveis e com números.

```
>> format long
>>
>> disp("O valor do numero eh:"),disp(e)
O valor do numero eh:
2.71828182845905
>> █
```

Agora testei com dois 'disp()'. Perceba que há uma vírgula entre eles! Num utilizei uma string e em outro coloquei o

número 'e'. Olhe a figura acima para ver o que foi feito e qual a saída do programa.

Funções

Para definir funções, usa-se o comando “function” e em seguida a declaração da função

($y=f(x)$), por exemplo, seguida da função:

```
>>
>> function y = f(x)
y=x^3+(3*x^2)-(2*x)+9;
endfunction
>> █
```

Agora outra função e já resolvendo-a:

```
>> function y = f(x)
y=x^3+x^2-3*x-3;
endfunction
>> [x, info] = fsolve ("f", 1.)
x = 1.73205080756888
info = 1
>> [x, info] = fsolve ("f", 0.)
x = -1
info = 1
```

Observação: o valor info=1 indica que a solução converge.

Alguns breves comandos:

Fatorial

Usar a função `factorial(n)` onde `n` é o número escolhido.

```
>> factorial(4)
ans = 24
>> factorial(3)
ans = 6
>> factorial(38)
ans = 5.23022617466601e+44
>> factorial(7)
ans = 5040
>> █
```

Limite $\lim_{x \rightarrow x_0} f(x)$

Para resolver este limite, o comando é : `limit(f(x), x = 0, Left)`.

Observar bem os parênteses, vírgulas, letras maiúsculas.

Integral

Para calcular a integral definida: $\int_0^1 f(x) dx$ usar o comando: `int(f(x), x = 0..1)`

`integrate(função, variavel);`

Calculo da integral definida em um intervalo de `a` até `b`:

`integrate(função, variável,a,b);`

Somatória

Podemos usar o comando `sum(f(i), i = 1..n)` para fazer uma somatória.

Como funciona os logs, salvar e editar

Para salvar os logs num arquivo:

```
>> diary on
>> 4+5
ans = 9
>> 7+45+3453/32
ans = 159.91
>> diary off
>> █
```

- Log diário

Digite *diary on* para ativá-lo e *diary off* para desativar. Isto funciona como criar um arquivo para guardar as informações de um certo cálculo ou função, por exemplo.

Para copiar o log também pode usar o famoso “ctrl+c” e “ctrl+v” para colá-lo em algum arquivo como num bloco de notas, por exemplo. Observem como fica um exemplo:

```
>> 6*4
ans = 24
```

A utilização do GNU-Octave apresenta um potencial maior quando se faz uso de recursos de programação. Com este propósito, é importante conhecer as formas de controlar os fluxos de cálculos **if**, **for**, **while** e **switch**. Neste tutorial, não darei ênfase a isto pois no momento, o essencial são os comandos matemáticos, para as aulas de cálculo.

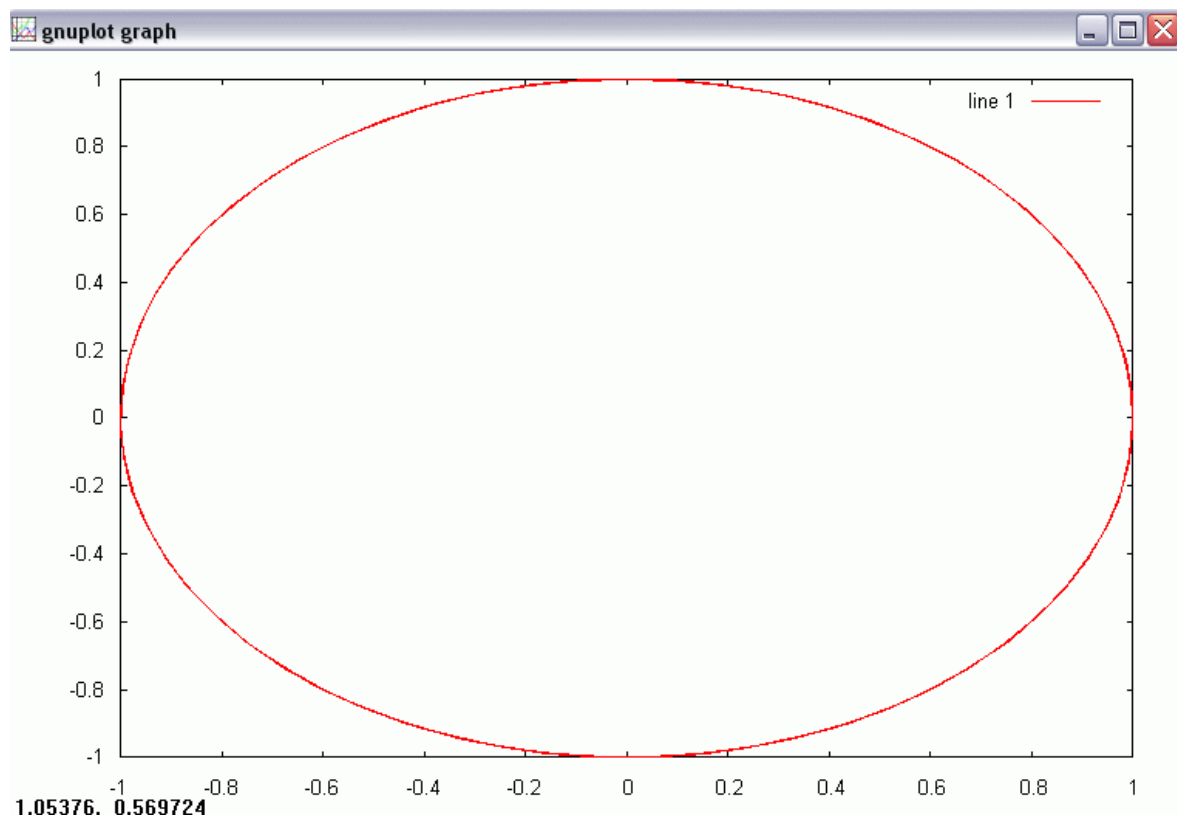
Os comandos do Gnu/Octave podem ser salvos em um arquivo texto de extensão **.m** ou **.oct**, como por exemplo **prog.m**. O que ajuda com a utilização depois dos comandos que você digitou no Octave, para o programa Matlab, que utiliza a extensão **.m** também.

Fazendo gráficos

Para fazer uma circunferência conhecida. Comando:

```
>> z=(-10:0.1:10)';
>> x=sin(z);
>> y=cos(z);
>> gset tittle "Circunferencia por Camili"
>> plot(x,y);
>> pause
```

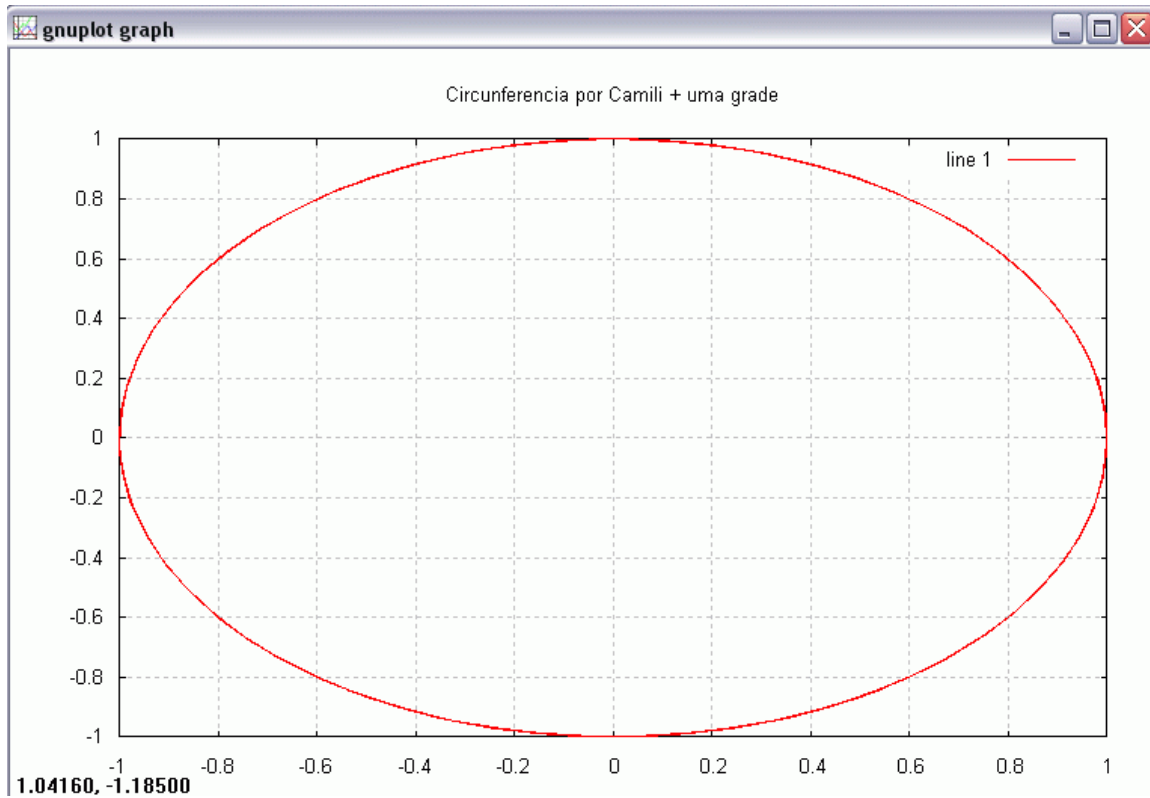
Abrirá uma nova janela com o seguinte gráfico:



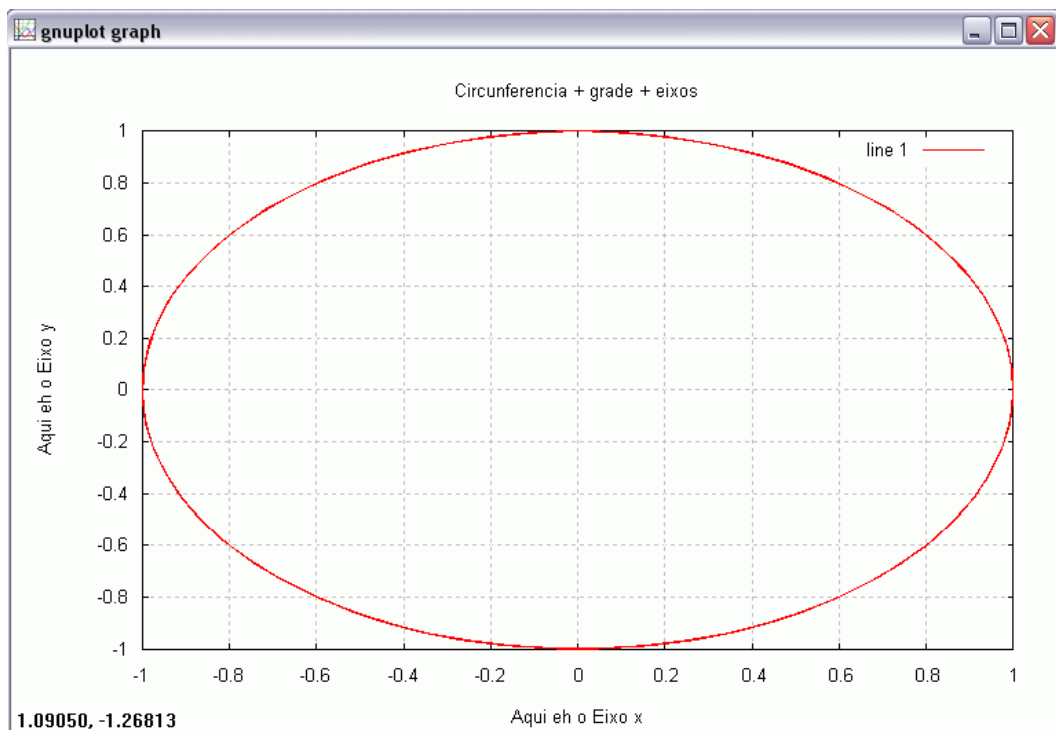
Para os gráficos da página seguinte, os comandos utilizados foram:

```
>> z=(-10:0.1:10)';
>> x=sin(z);
>> y=cos(z);
>> gset tittle "Circunferencia por Camili"
>> plot(x,y);
>> pause
>> gset title "Circunferencia por Camili + uma grade"
>> grid "on"
>> plot(x,y);
>> pause
>> gset title "Circunferencia + grade + eixos"
>> gset xlabel "Aqui eh o Eixo x"
>> gset ylabel "Aqui eh o Eixo y"
>> grid "on"
>> plot(x,y);
```

Podemos adicionar uma “grade” atrás do gráfico. Basta usar o comando `grid “on”`:

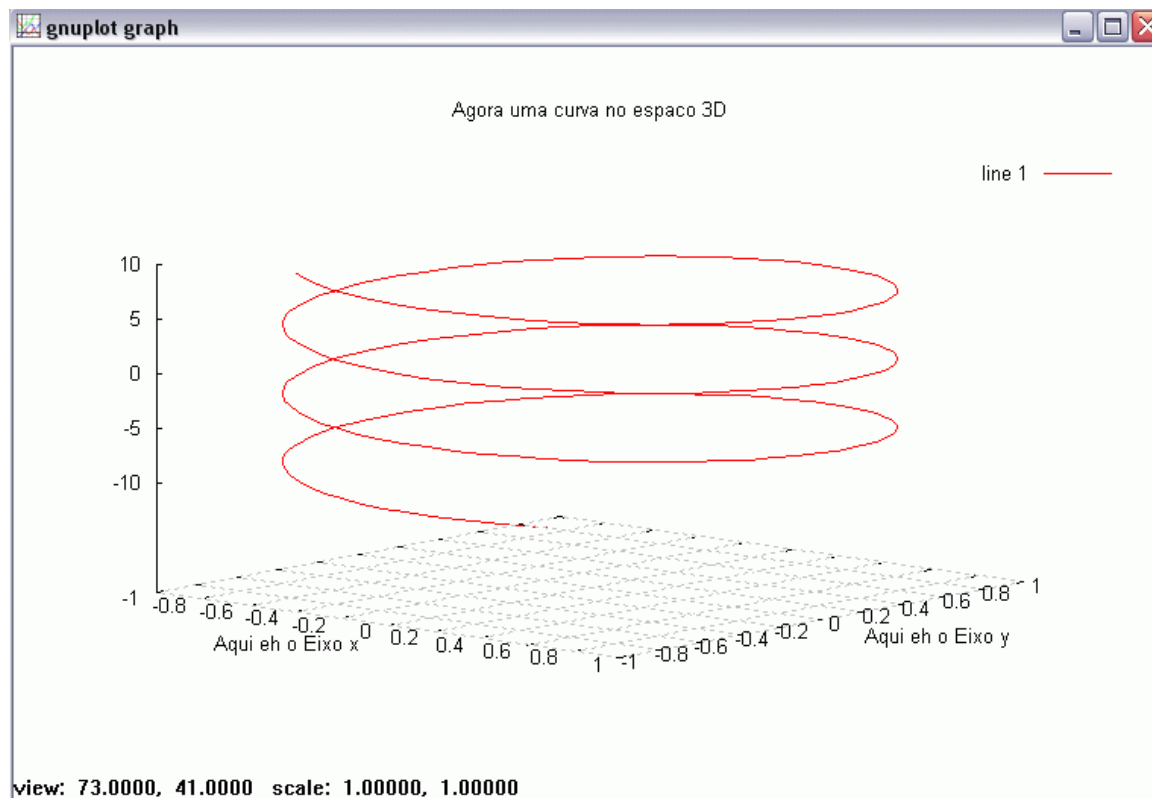


Também podemos adicionar ‘nome aos eixos’:

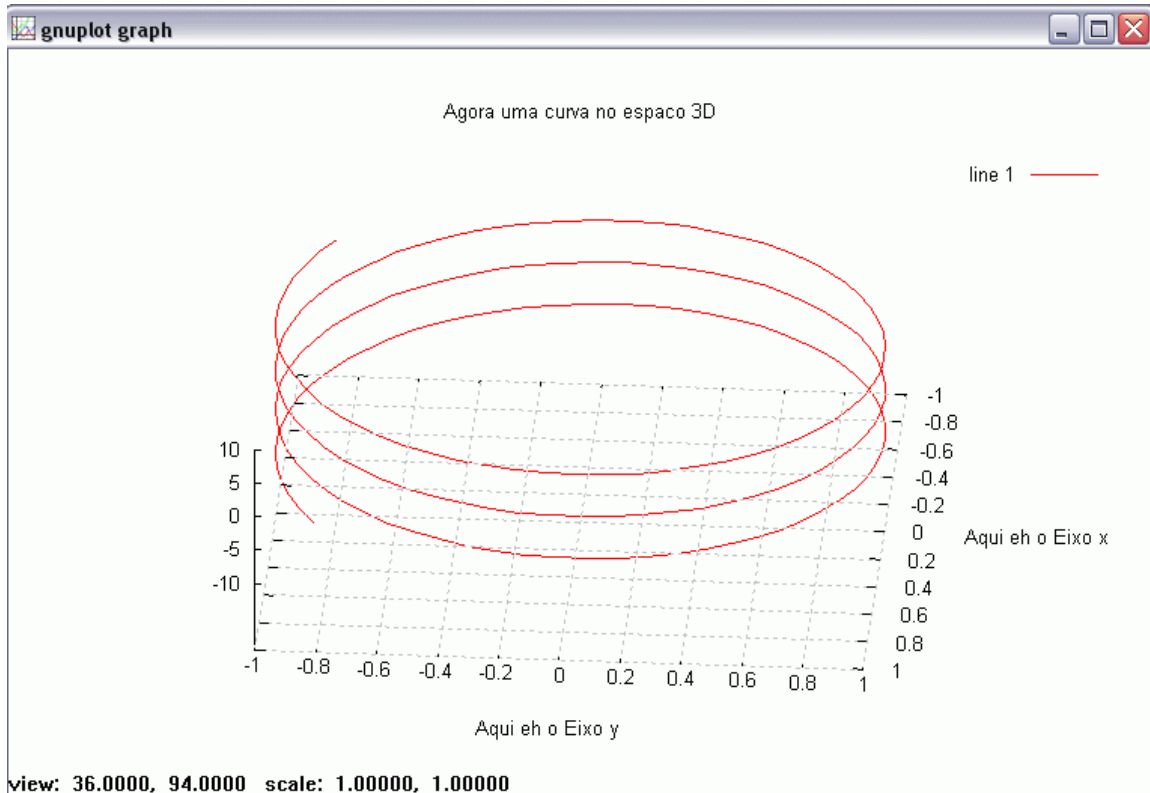


Para fazer um gráfico 3-D, basta usar:

```
>> z=(-10:0.1:10)';  
>> x=sin(z);  
>> y=cos(z);  
>> gset title "Um grafico 3D: Curva no espaco 3D"  
>> grid "on"  
>> plot3(x,y,z);  
>> █
```



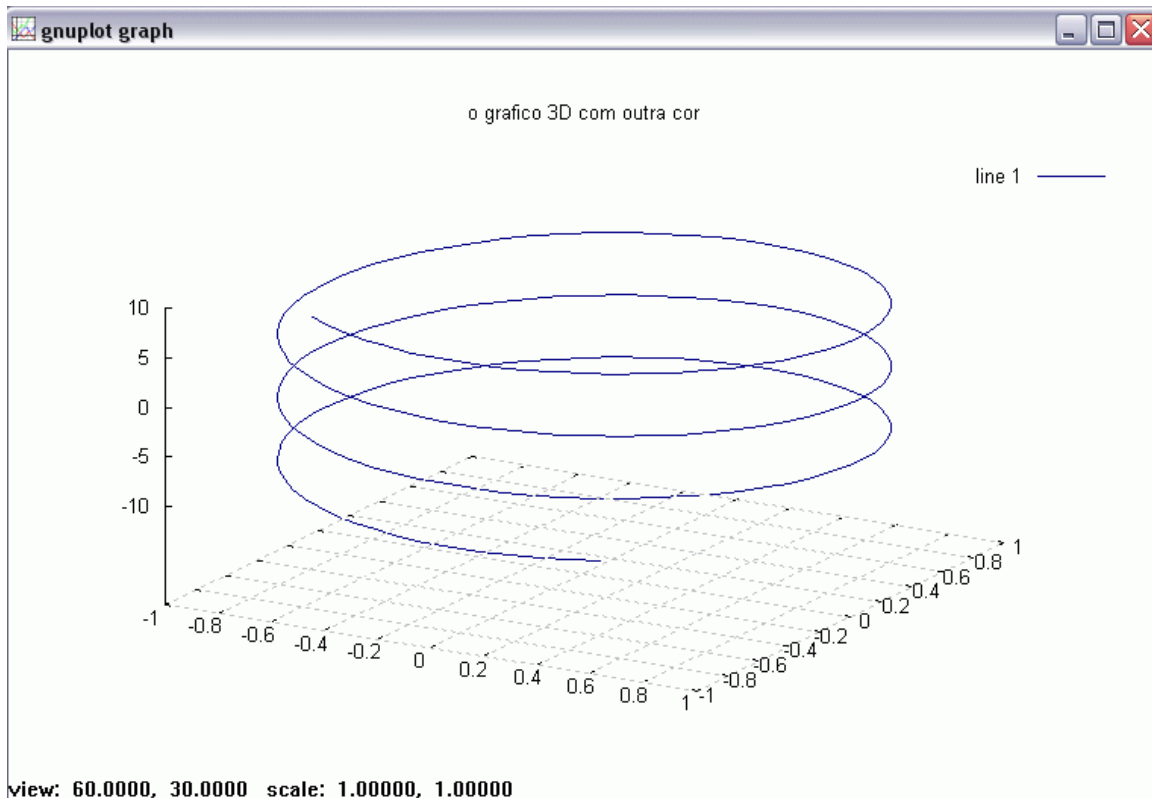
Podemos mudar o “ângulo” de ver este gráfico. A maneira mais prática é clicar em cima do gráfico plotado, com botão esquerdo do mouse, aparecerá um ‘desenho’ de duas setas, indicando um movimento e arrastar o cursor pros lados até encontrar outro ângulo que agrade. Veja abaixo o gráfico acima com uma mudança no ângulo:



Também podemos mudar a cor do gráfico:

Para cor ciano, temos `plot3(x,y,z,"5")`; . Veja:

```
>> z=(-10:0.1:10)';
>> x=sin(z);
>> y=cos(z);
>> grid "on"
>> plot3(x,y,z,"5");
>> gset title "o grafico 3D com outra cor"
>> grid "on"
>> plot3(x,y,z,"5");
```



Para um gráfico verde, basta, no comando acima, digitar : “2” , ao invés de “5”.

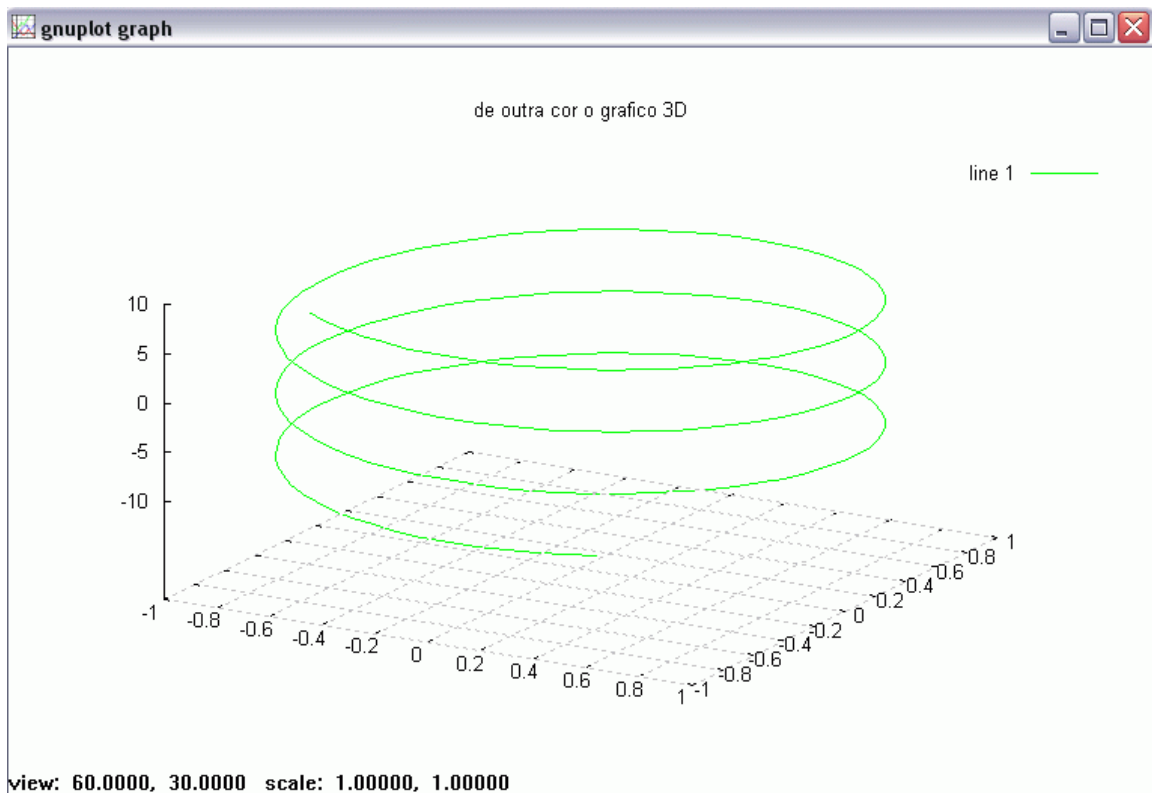


Gráfico de funções trigonométricas:

Abaixo segue o gráfico do seno e o comando utilizado no Octave para plota o gráfico seno, o gráfico do cosseno e gráfico do seno junto com o do cosseno(veja páginas seguintes).

```
>> m=(0:0.01:2*pi)';  
>> n=sin(m);  
>> a=cos(m);  
>>  
>> gset title "grafico do seno"  
>> plot(m,n);  
>> gset title "grafico do cosseno"  
>> plot(m,a)  
>> clg  
>> gset title "grafico seno mais grafico do cosseno"  
>> data=[m,n,a];  
>> gplot data with lines, data using 1:3 with impulses 8  
>> █
```

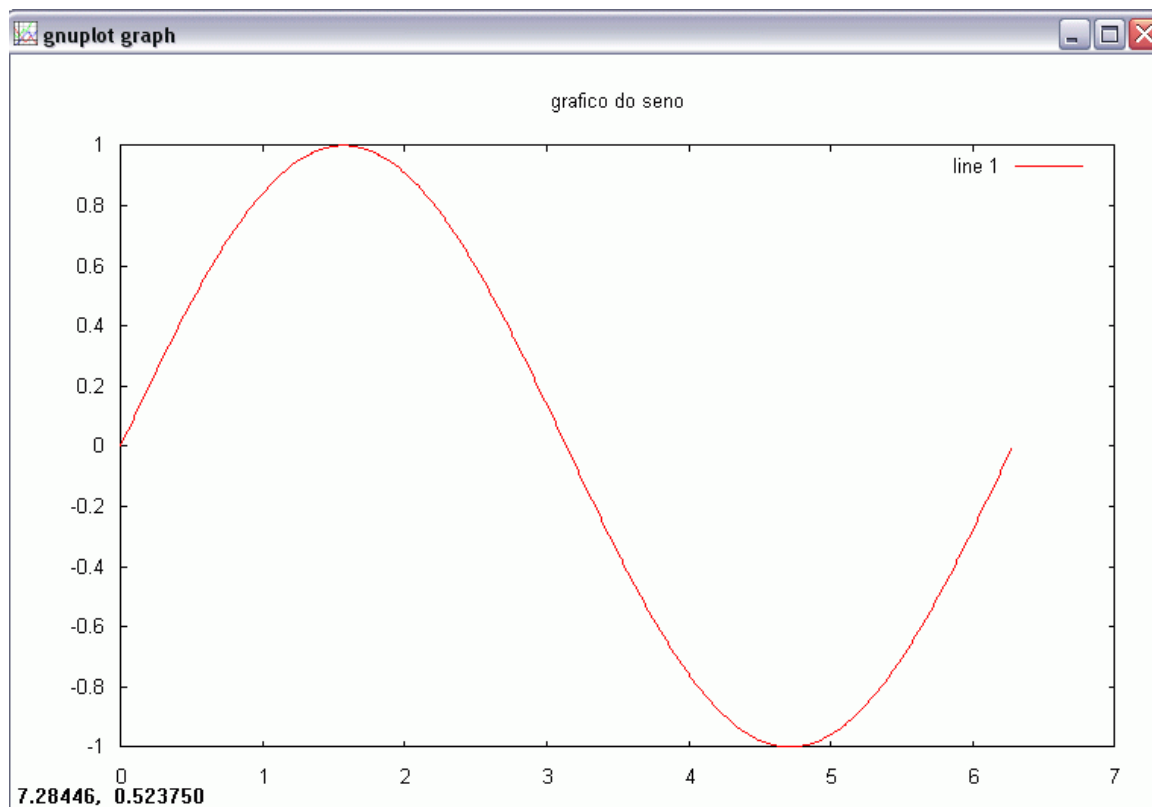
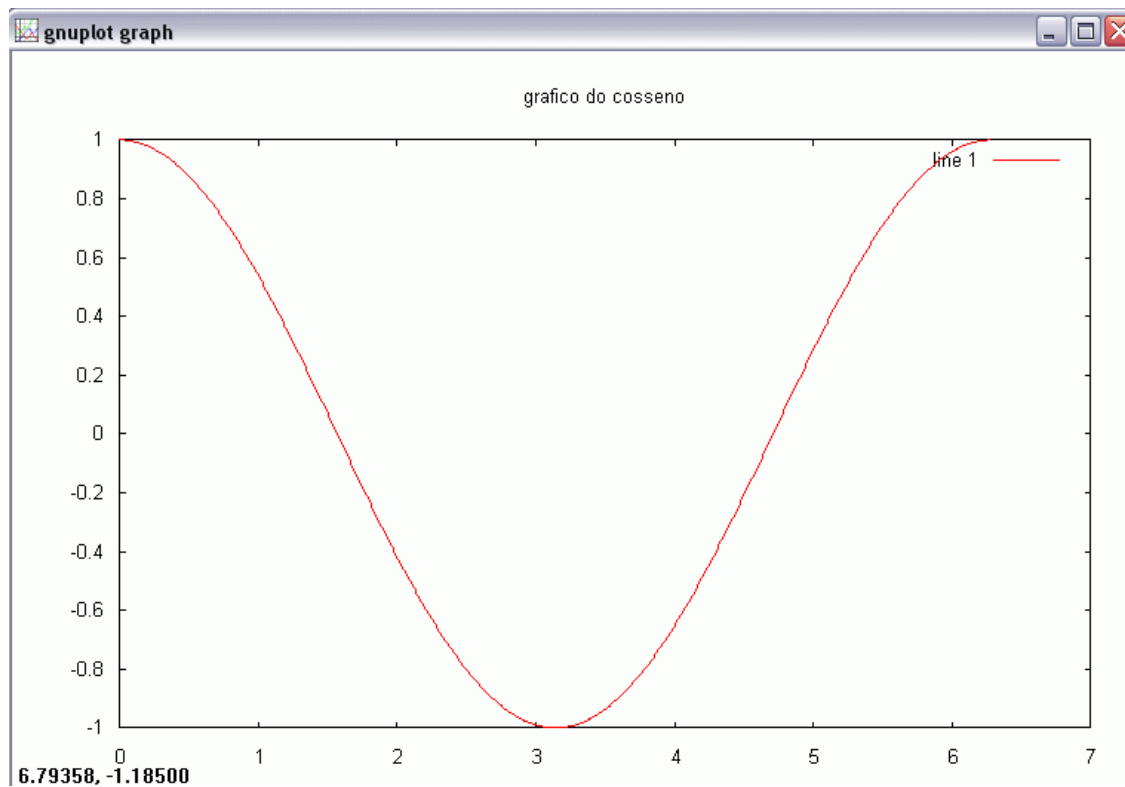
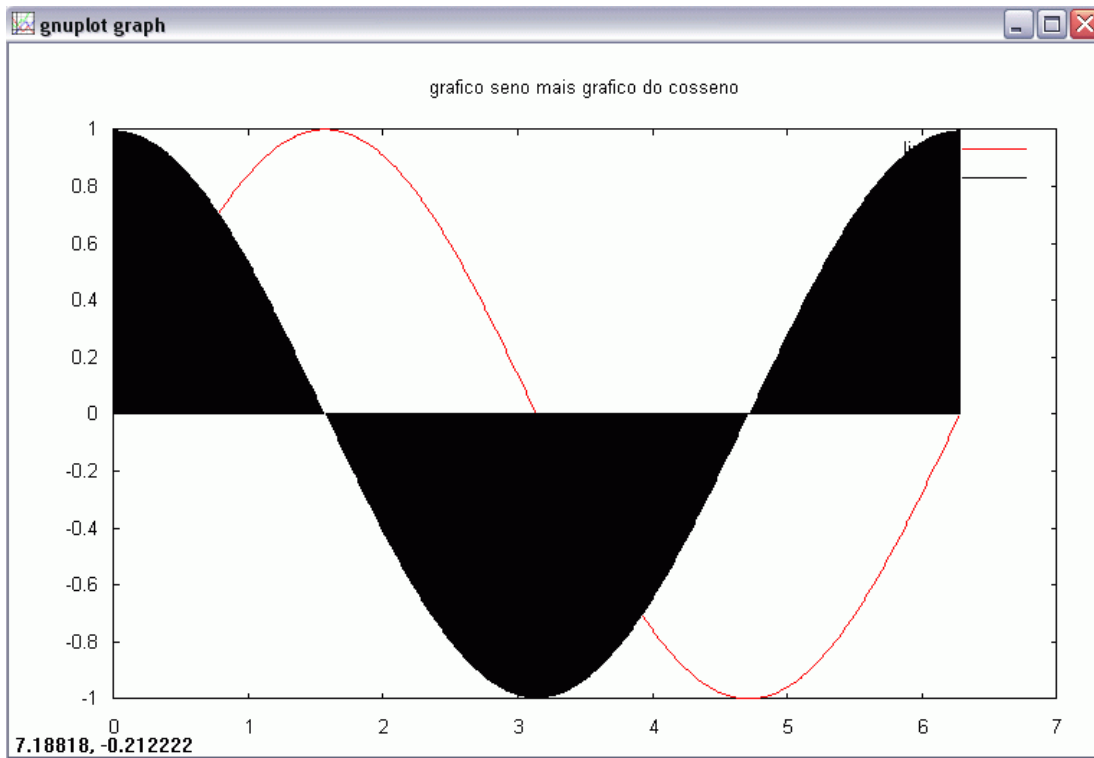


Gráfico do Cosseno

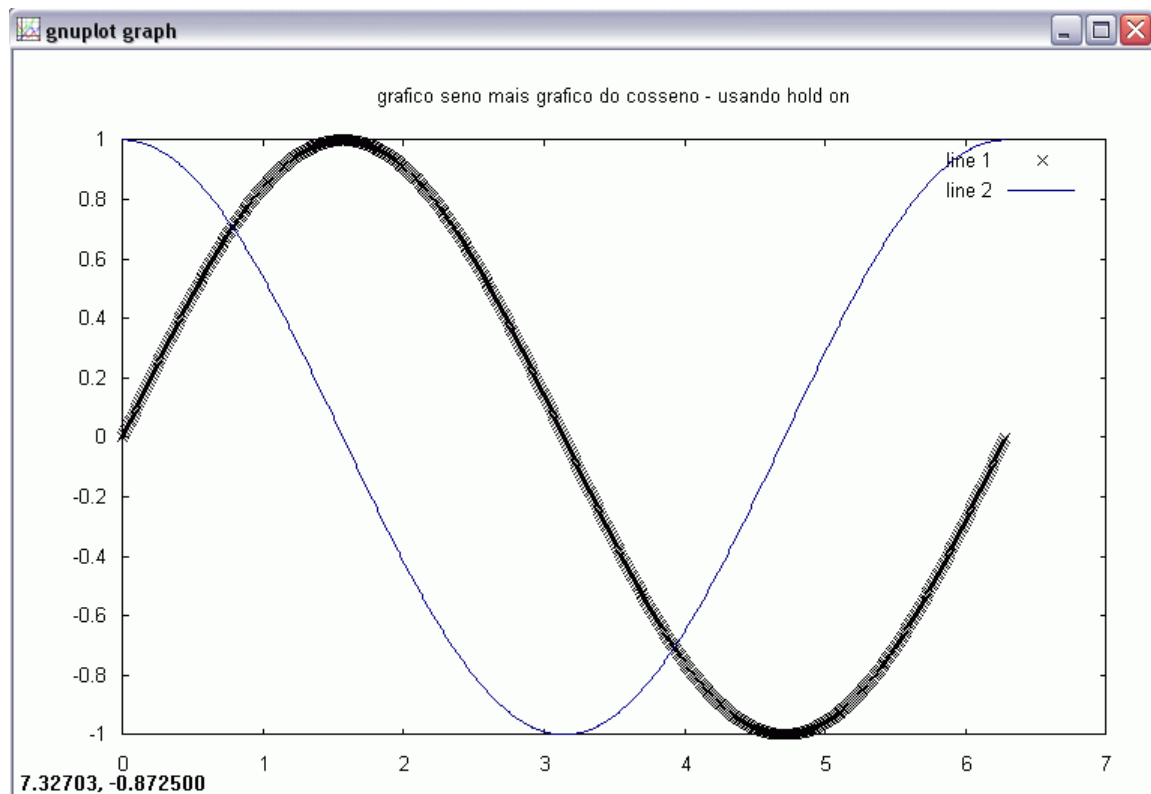


Dois gráficos em um! Gráfico do seno e do cosseno:



O mesmo gráfico, usando o comando hold on:

```
>> c!g
>> gset title "grafico seno mais grafico do cosseno - usando hold on"
>> data=[x,y,z];
error: `x' undefined near line 14 column 7
error: evaluating assignment expression near line 14, column 5
>> data=[m,n,a];
>> gplot data with points 8 4
>> hold on
>> gplot data using 1:3 with line 5
>> pause
>> hold off
```



Também podemos usar o comando `stairs(x,y)`:

```
>>  
>> c=(0:0.5:2*pi)';  
>> y=sin(x);  
error: `x' undefined near line 21 column 7  
error: evaluating argument list element number 1  
error: evaluating assignment expression near line 21, column 2  
>> y=sin(c);  
>> z=cos(c);  
>>  
>> clg  
>> gset title "Testando o comando stairs(x,y)"  
>> stairs(c,y)  
ans = []  
>> █
```

E, na janela do gnuplot aparecerá o gráfico abaixo:

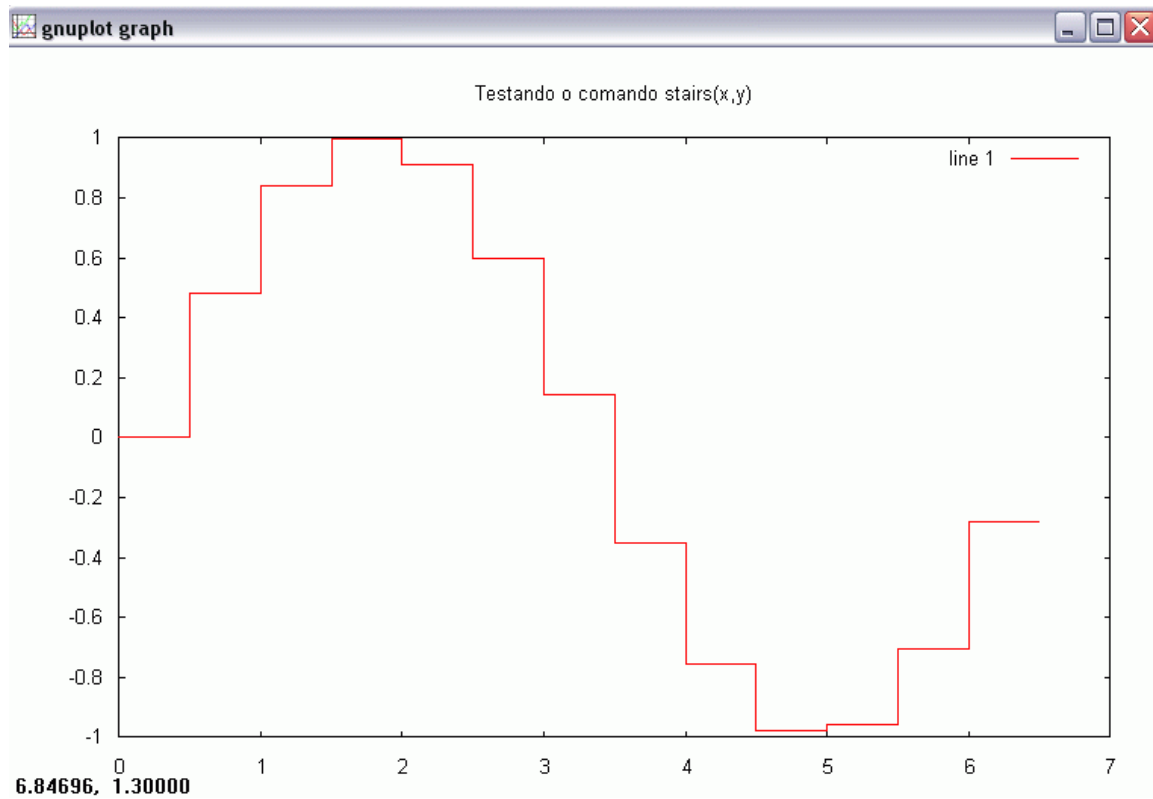


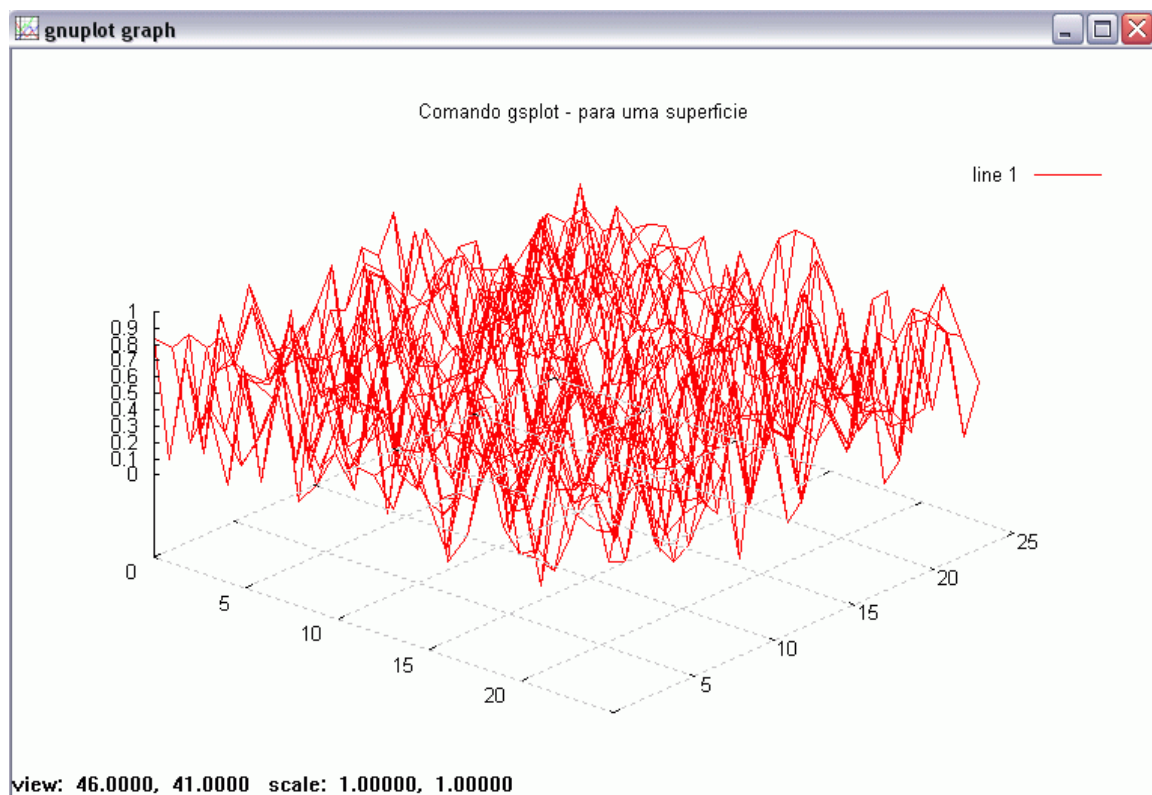
Gráfico de uma superfície. Comandos:

```
>> clg
>> gset title "Comando gplot - para uma superficie
error: unterminated string constant
parse error:

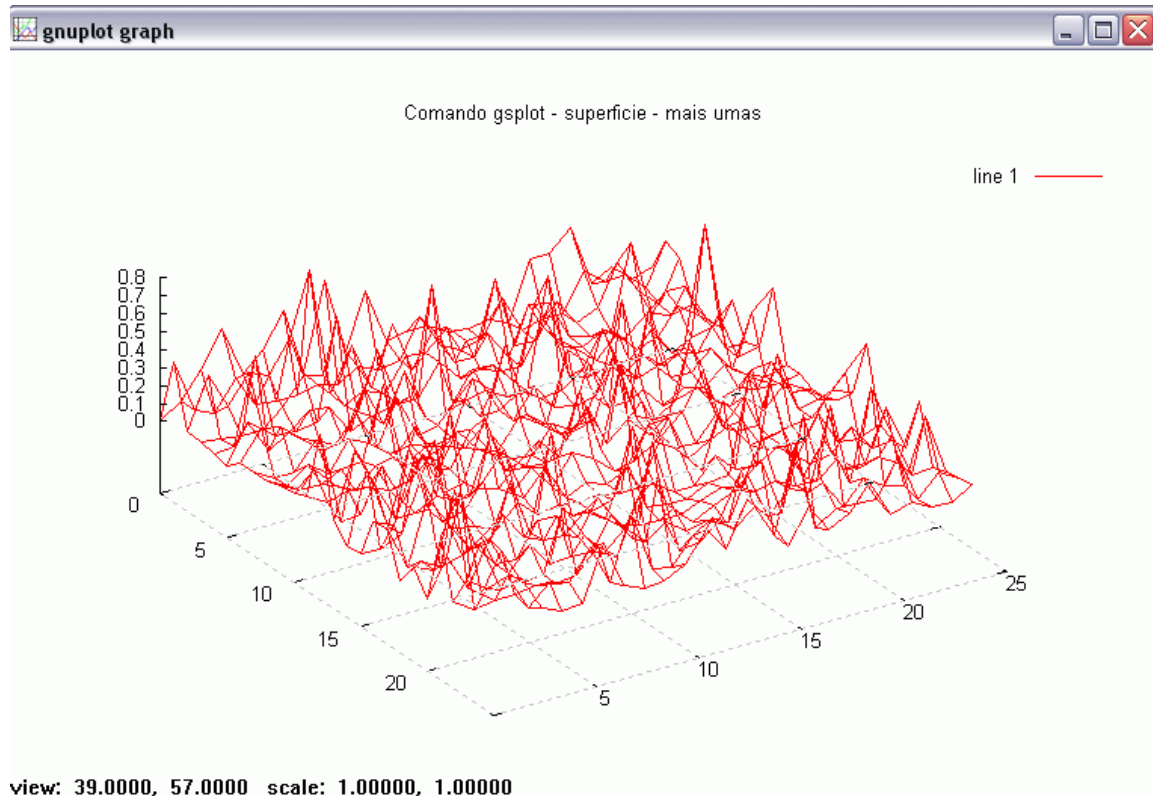
>>> gset title "Comando gplot - para uma superficie
^

>> gset title "Comando gplot - para uma superficie"
>> gspplot rand(25,25)
>> grid "on"
```

O gráfico:

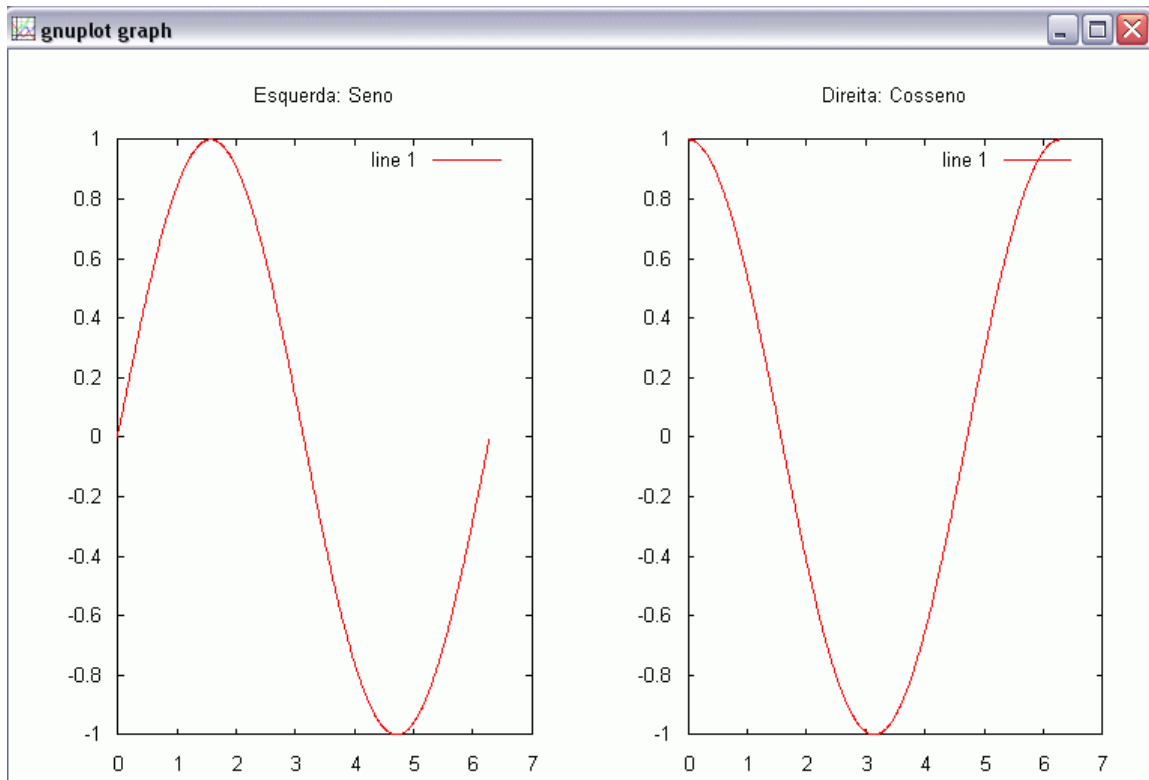


Mais uma superfície e seus comandos:



```
>> clg
>> gset title "Comando gspot - superficie - mais umas"
>> gspot (rand(25,25).*rand(25,25)).*rand(25,25)
grid "on"
parse error:
>>> grid "on"
      ^
>> gspot (rand(25,25).*rand(25,25)).*rand(25,25)
>> grid "on"
>> █
```

O legal do Octave é que podemos colocar na janela lá do gnuplot, dois gráficos distintos, um ao lado do outro. Também podemos colocar 4 gráficos! Veja:



Para 4 gráficos temos:

```
>> clg
>>
>> subplot(2,2,1)
>> gset title "Seno"
>> gplot data 1
parse error:

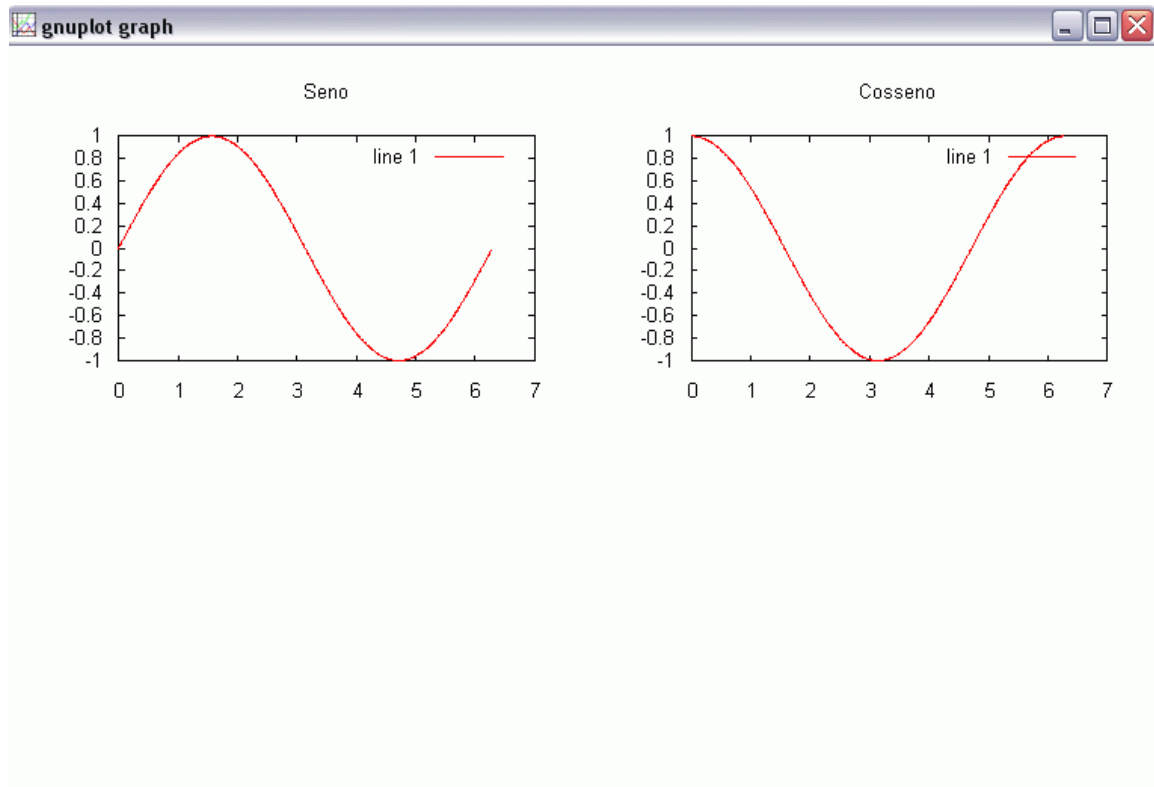
>>> gplot data 1
^

>> gplot data 1
parse error:

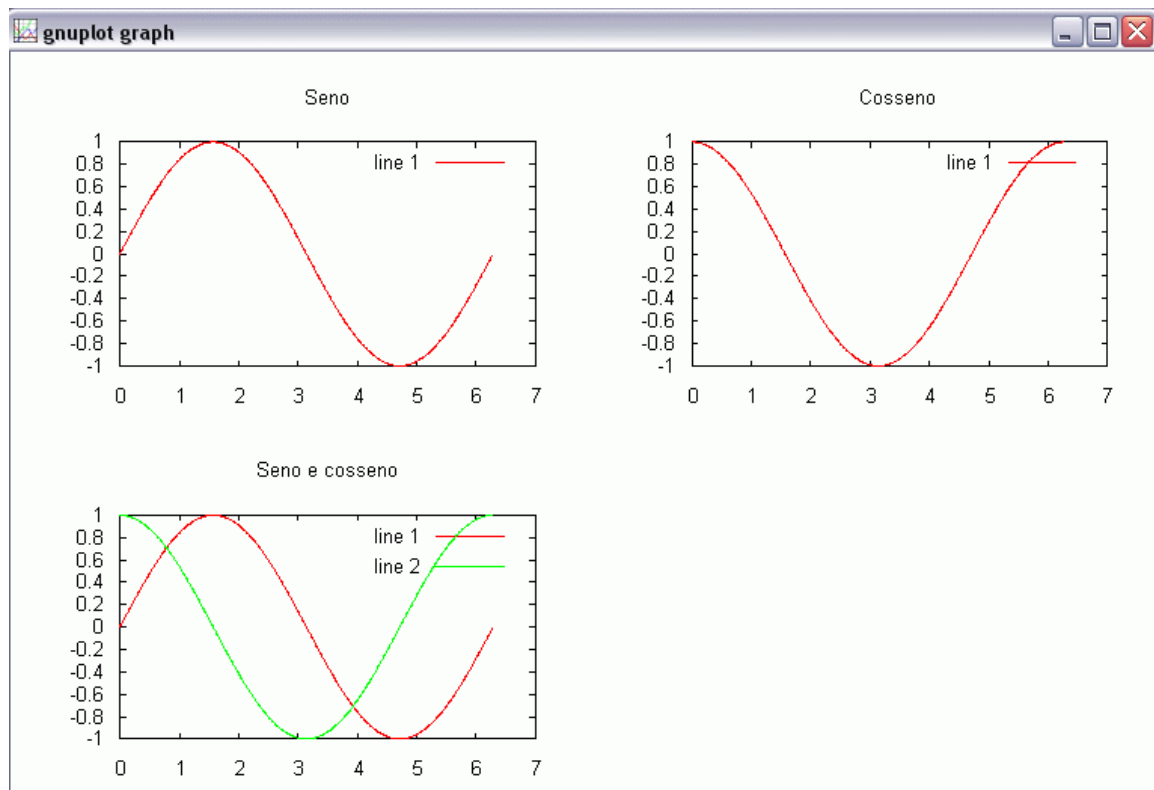
>>> gplot data 1
^

>> gplot data1
>>
>> subplot(2,2,2)
>> gset title "Cosseno"
>> gplot data2
>>
>> subplot(2,2,3)
>> gset title "Seno e cosseno"
>> gplot data1,data2
>>
>> subplot(2,2,4)
>> gset title "Raiz de x"
>> gplot sqrt(x)
>> ■
```

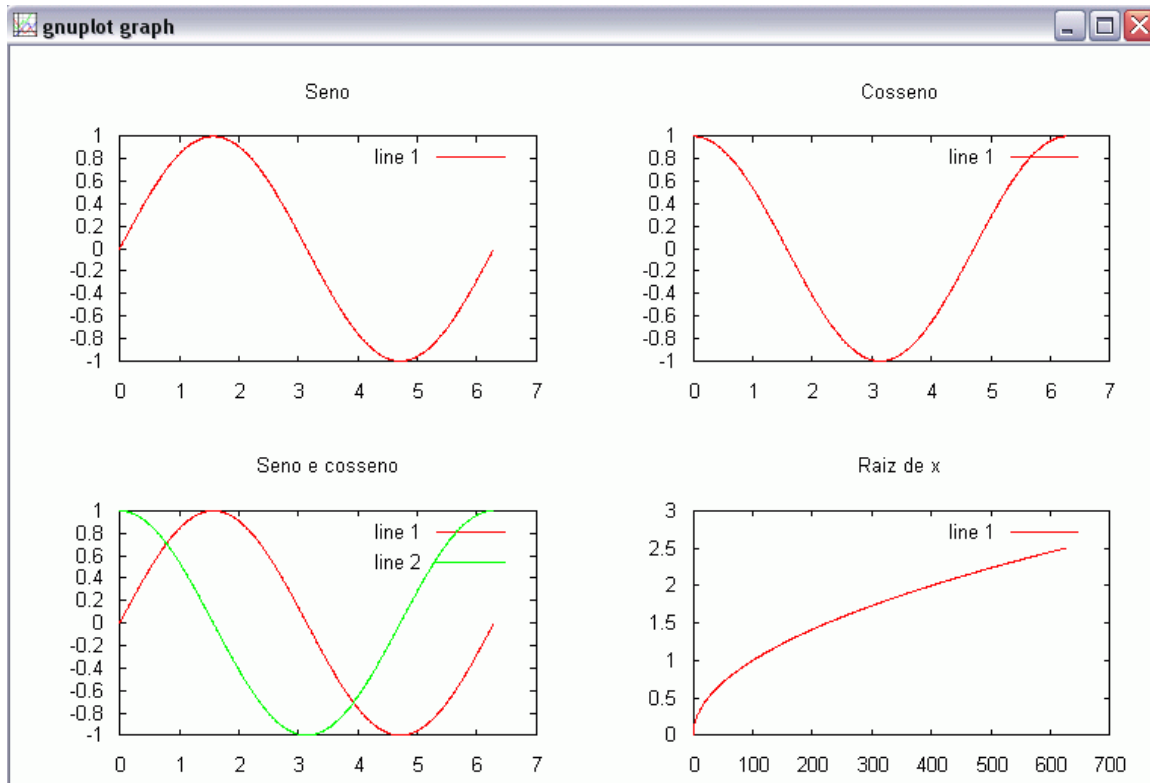
Quando digitamos os comandos para dois gráficos, a janela do gnuplot estará assim:



Com 3 comandos já enviados, teremos esta outra imagem:



E, enfim, para os quatro gráficos na mesma janela, teremos esta imagem:3



Observações:

A tabela de números e cores para gráfico é:

Número Cores no Gnuplot

- 1 vermelho
- 2 verde
- 3 azul
- 4 magenta
- 5 ciano ('azul escuro' diferente)
- 6 marrom

Explicação de alguns comandos:

`clearplot` ou `clg` : limpam a janela que abriu quando plotamos o gráfico.

`rand(,)` : pega números aleatórios no intervalo dado.

Bibliografia

Para a montagem deste projeto, utilizei diversos sites da Internet.

1. <http://www2.prudente.unesp.br/dcartog/galo/octave/fct.htm>
(muito importante para entender os gráficos e refazê-los).
2. <http://www.octave.org/FAQ.html> (próprio site do Octave. Nesta sessão há várias perguntas e suas respostas ajudaram em partes a conseguir mexer com este programa).
3. http://sunsite.univie.ac.at/textbooks/octave/octave_toc.html
4. http://ssdi.di.fct.unl.pt/cursos/pce/0405-1/material/aulas_praticas/octave/guia/guia_octave.html
5. http://paginas.fe.up.pt/~jcard/Octave_links_org.html (há links para sites sobre Octave e, também, para Matlab).
6. <http://www.aims.ac.za/resources/tutorials/octave/index.php> (outro site com um tutorial muito bom sobre Octave).
7. <http://www.math.uic.edu/~hanson/Octave/OctaveNonlinearEG.html> (há vários exemplos resolvidos neste site).
8. http://volga.eng.yale.edu/sohrab/matlab_tutorial.html#the_basics (outro site com um bom tutorial).

Comentário Final:

Infelizmente não deu para colocar todos comandos do Octave neste tutorial. Como é um tutorial básico, não dei ênfase a vários comandos, mas quem se interessar, pode visitar os sites acima – vide Bibliografia – que são muito bons (principalmente os que eu comentei ali).