SESSÃO DE MATLAB (Matrix Laboratory)

Lab. 0S1, 14.06.2014

Rosário Laureano

Symbolic Math Toolbox (uma cx. ferramentas do MATLAB

DERIVAÇÃO PARCIAL de 1ª ordem

Command window

Definir as variáveis (como símbolos):

```
>> syms x
>> syms y
```

Uso da Função diff indicando qual a função a derivar:

```
>> diff(y*sin(x))
```

Não sendo indicada a var. de derivação, é assumido ser *x* ou ser *y*?

```
ans =

y*cos(x)
```

(assume *x* quando não é indicada outra variável)

Para obter a derivada parcial em ordem a outra variável há que indicá-la:

```
>> diff(y*sin(x),y)

ans =

sin(x)
```

Se *x* não está presente na expressão da função, é assumida como variável?

```
>> diff(sin(y))
```

(seria esperada a resposta 0)

```
ans =

cos(y)
```

(curiosamente y é assumida como a var. de derivação)

As duas variáveis podem ser definidas em simultâneo:

```
>> syms x y
>> diff(y*exp(x))

ans =

y*exp(x)

>> diff(y*exp(x),y)

ans =

exp(x)
```

ATENÇÃO: **Clear Command Window** **só limpa o ambiente de trabalho , mantendo as variáveis simbólicas já definidas.**

ATENÇÃO: para limpar as variáveis simbólicas usar Clear WorkPlace e seleccionar Variables.

Clear WorkPlace **Variables**

```
>> syms x  (apenas x está a ser definida)
>> diff(sin(x)) (seria esperada a resposta x*sin(y))

ans =

cos(x)
```

Quando *y* não foi definida como símbolo e está presente na função, é assumida como constante?

```
>> diff(y*sin(x))
Undefined function or variable 'y'.
```

DERIVAÇÃO PARCIAL de ordem superior

Para obter derivação de ordem superior:

```
>> diff(x^3*y^4)

ans =

3*x^2*y^4
```

(deriv. parcial de 1ª ordem em *x*, denotada por f'_x ou Df_x)

```
>> diff(x^3*y^4,2)

ans =

6*x*y^4
```

(deriv. parcial de 2ª ordem em *x*, denotada por f''_{xx} ou D^2f_x)

```
>> diff(x^3*y^4,y)

ans =

4*x^3*y^3
```

(deriv. parcial de 1ª ordem em *y*, denotada por f'_y ou Df_y)

```
>> diff(x^3*y^4,2,y)

ans =

12*x^3*y^2
```

(deriv. parc. de 2ª ordem em *y*, denotada por f''_{yy} ou D^2f_y)

```
>> diff(x^3*y^4,2,x,y)
```

(TENTATIVA para obter deriv. de 2ª ordem mista)

```
Error using sym/diff (line 18)
Too many input arguments.
```

Para obter derivadas de ordem superior mistas:

```
>> diff(exp(2*x)*sin(y))

ans =

2*exp(2*x)*sin(y)
```

(derivada parcial de 1ª ordem em *x*)

```
>> syms dfx
```

(nova definição simbólica)

```
>> dfx=ans

dfx =

2*exp(2*x)*sin(y)
```

(OK)

```
>> diff(dfx,y)

ans =
```

2*exp(2*x)*cos(y) **(deriv. parcial de 2ª ordem primeiro em *x* e depois em *y*, denotada por f''_{xy} ou D^2f_{xy})**

Prosseguindo com derivação de df **na variável y:**

```
>> diff(dfx,2,y)

ans =

-2*exp(2*x)*sin(y)
```

(deriv. parcial de 3ª ordem $f^{(3)}_{xyy}$ ou D^3f_{xyy})

Obviamente, df **também pode ser usada para obter f''_{xx} ou D^2f_x:**

```
>> diff(dfx)

ans =

4*exp(2*x)*sin(y)
```

(deriv. parcial de 2ª ordem em *x*, ou seja, f''_{xx} ou D^2f_x)

embora esta derivada possa ser obtida de forma mais rápida:

```
>> diff(exp(2*x)*sin(y),2)

ans =

4*exp(2*x)*sin(y)
```

(confirma!)

De modo análogo, para obter outras derivadas parciais mistas:

```
>> diff(exp(2*x)*sin(y),y)

ans =

exp(2*x)*cos(y)
```

(derivada parcial de 1ª ordem em *y*)

```
>> syms dfy
```

(nova definição simbólica)

```
>> dfy=ans

dfy =

exp(2*x)*cos(y)
```

(OK)

```
>> diff(dfy,y)

ans =
```

-exp(2*x)*sin(y) **(deriv. parcial de 2ª ordem em *y*, denotada por f''_{yy} ou D^2f_y)**

E, não necessitando de especificar x como variável de derivação:

>> diff(dfy)

ans =

2*exp(2*x)*cos(y) **(deriv. parcial de 2ª ordem primeiro em *y* e depois em x, denotada por f''_{yx} ou D^2f_{yx})**

NOTA: confirma o Teorema de Schwartz em funções regulares

Modo rápido que evita usar repetidamente a expresão da função:

>> syms x y z
>> f=x^2*y^3*z^4 **(definir simbolicamente a função em estudo)**

f =

x^2*y^3*z^4 **(OK)**

>> diff(f)

ans =

2*x*y^3*z^4 **(derivada parcial de 1ª ordem em *x* da função f)**

>> diff(f,y)

ans =

3*x^2*y^2*z^4 **(derivada parcial de 1ª ordem em *y* da função f)**

>> diff(f,z)

ans =

4*x^2*y^3*z^3 **(derivada parcial de 1ª ordem em *z* da função f)**

Prosseguindo para derivadas de ordem superior:

>> diff(diff(f))

ans =

2*y^3*z^4 **(deriv. parcial de 2ª ordem em *x*, ou seja, f''_{xx} ou D^2f_x)**

>> diff(diff(f),y)

ans =

6*x*y^2*z^4 **(deriv. parcial de 2ª ordem primeiro em *x* e depois em *y*)**

>> diff(diff(f,y),y)

ans =

6*x^2*y*z^4 **(deriv. parcial de 2ª ordem em *y*, ou seja, f''_{yy} ou D^2f_y)**

embora esta derivada possa ser obtida de forma mais rápida:

>> diff(f,2,y)

ans =

6*x^2*y*z^4 **(confirma!)**

Uso da Função gradient para obter o vetor gradiente de uma função escalar:

>> f=x^2*y^3*z^4 **(definir simbolicamente a função em estudo)**
>> gradient(f) **(também** gradient(f, [x, y, z])**)**

ans =

```
  2*x*y^3*z^4
 3*x^2*y^2*z^4
 4*x^2*y^3*z^3
```

Cada componente do vetor gradient pode ser pedida individualmente:

>> gradient(f,y)

ans =

3*x^2*y^2*z^4

Escolha das components do vetor gradient que interessa considerar:

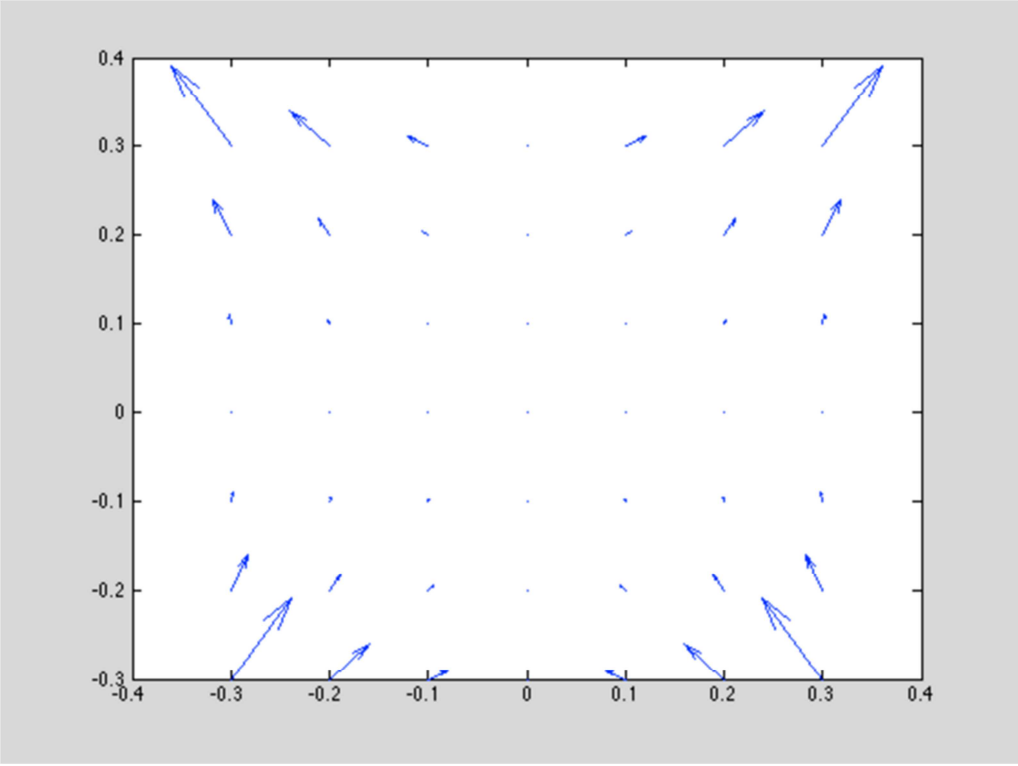
```
>> gradient(f, [x, z])

ans =

  2*x*y^3*z^4
4*x^2*y^3*z^3
```

Representação gráfica do campo de vetores dado pelo gradiente da função:

```
>> g=gradient(f)   (definição de nova variável simbólica)

g =

  2*x*y^3
3*x^2*y^2
```

O MATLAB possui a função gráfica quiver para esta tarefa. Contudo, esta função não reconhece argumentos simbólicos. Como tal, nas expressões das components de *g*, há que substituir as variáveis simbólicas *x* e *y* por valores numéricos e só depois aplicar a função quiver.

```
>> [X, Y] = meshgrid(-.3:.1:.3,-.3:.1:.3);
   G1 = subs(g(1), [x y], {X,Y}); G2 = subs(g(2), [x y], {X,Y});
   quiver(X, Y, G1, G2)
```



(vetores orientados conforme as expressões do gradiente ($2xy^3$, $3x^2y^2$)

PRIMITIVAÇÃO de FUNÇÕES com (uma ou) várias variáveis

Command window

Definir as variáveis (como simbólicas):

```
>> syms x  y
```

Uso da Função int indicando qual a função a primitivar:

```
>> int(3*x^2*sin(y))
```

Não sendo indicada a var. de primitivação, é assumido ser *x* ou ser *y*?

```
ans =

x^3*sin(y)
```

(assume *x* quando não é indicada outra variável)

Para primitivar em relação a outra variável há que indicá-la:

```
>> int(3*x^2*sin(y),y)

ans =
 -3*x^2*cos(y)
```

```
>> syms x y
>> int(sin(x))

ans =

-cos(x)
```

(obviamente assume x)

```
>> int(sin(x),y)

ans =

y*sin(x)
```

Se *x* não está presente na expressão da função, é assumida como variável?

>> int(sin(y)) **(seria esperada a resposta x*sin(y))**

ans =

-cos(y) **(curiosamente *y* é assumida como a var. de primitivação)**

Neste caso, para primitivar em relação à variável *x*, há que indicá-la:

>> int(sin(y),x)

ans =

x*sin(y)

ATENÇÃO: **Clear Command Window** **só limpa o ambiente de trabalho , mantendo as variáveis definidas;**

ATENÇÃO: para limpar as variáveis usar Clear WorkPlace e seleccionar Variables.

Clear WorkPlace **Variables**

>> syms x **(apenas *x* está a ser definida)**
>> int(sin(x))

ans =

-cos(x)

Quando *y* não foi definida e está presente na função, é assumida como constante?

>> int(sin(y),x)
Undefined function or variable 'y'.

Com que "grau" de simplificação são dadas as primitivas?

>> syms x y

```
>> int(x+y)

ans =

(x*(x + 2*y))/2
```

(em vez da expressão mais simples $xy + y^2/2$)

NOTA: o MATLAB privilegia produto e quocientes como operações principais presentes na resposta com vista à sua utilização posterior)

Conforme é esperado, a primitivação na variável *y* conduz a uma expressão que é a "simétrica" da anterior:

```
>> int(x+y,y)

ans =

(y*(2*x + y))/2
```

Primitivaçao de funções racionais não-próprias:

```
>> syms x y
>> int(x*y/((x+1)*(x-3)))

ans =

(y*(log(x + 1) + 3*log(x - 3)))/4
```

Primitivação por partes:

```
>> syms x y
>> int(x*y*log(x))

ans =

(x^2*y*(log(x) - 1/2))/2
```

Primitivação por substituição ou mudança de variável (m.v.):

```
>> syms x y
>> int((sqrt(x)/(x+1)))

ans =

2*x^(1/2) - 2*atan(x^(1/2))
```

(arctan é atan no MatLab) (idem para arcsin, arccos e arcotan)

(a resposta é dada com x*(1/2) em vez se sqrt(x); para raízes de outros indices terá de ser usada a notação em expoente, embora "abusar" de parentesis seja uma chatice!...)

```
>> int(x^(1/2)/(x+1)))
 int(x^(1/2)/(x+1)))              |
Error: Unbalanced or unexpected parenthesis or bracket.

>> int((x^(1/2))/(x+1)))
 int((x^(1/2))/(x+1)))                |
Error: Unbalanced or unexpected parenthesis or bracket.

>> int(x^(1/2)/(x+1))

ans =

2*x^(1/2) - 2*atan(x^(1/2))
```

INTEGRAIS SIMPLES E MÚLTIPLOS

Command window

Há vantagens em definir simbolicamente a função integranda:

```
>> f=exp(2*x)

f =

exp(2*x)
```

(OK)

Mantém-se o uso da Função int, indicando qual a função integranda (através do seu símbolo) e os extremos superior e superior de integração:

```
>> int(f,0,2)

ans =

exp(4)/2 - ½
```

(valor do integral de f no intervalo [0,2]) (trata-se de uma área visto que exp(x)>0)

Uso da Função integral num modo mais avançado, em que a função pode depender de um parâmetro:

```
>> syms x
```

```
>> f=@(x,p) exp(2*x+p)

f =

   @(x,p)exp(2*x+p)
```

(OK)

```
>> h=integral(@(x)f(x,-1),0,3)

h =

  74.0226
```

O cálculo de integrais de funções negativas no intervalo de integração não é problema:

```
>> m=-2*x

m =

-2*x

>> int(m,2,4)

ans =

-12
```

(trata-se do valor simétrico da área pois -2x<0 no intervalo [0,2])

Usa a decomposição quando as funções alternam entre si de "superioridade":

```
>> h=sin(x)-cos(x)

h =

sin(x) - cos(x)

>> int(h,0,pi/4)

ans =

1 - 2^(1/2)
```

(não se trata do cálculo de uma área)

O cálculo de integrais duplos em domínios planos que sejam retângulos segue o processo iterativo (e intuitivo) conhecido do Cálculo:

```
>> syms x y
>> g=exp(2*x)*y

g=

y*exp(2*x)
```
(OK)

```
>> int(g,0,2)

ans =

(y*(exp(4) - 1))/2
```
(assume naturalmente x como var. de integração)

```
>> int(ans,y,-3,4)
```
(integral na var. y da expressão obtida na resposta)

```
ans =

(7*exp(4))/4 - 7/4
```
(valor do integr. duplo de g no retângulo [0,2]x[-3,4])

embora se possa também utilizar:

```
>> int(int(g,0,2),y,-3,4)

ans =

(7*exp(4))/4 - 7/4
```
(confirma!)

Pela ordem inversa obtemos, obviamente, o mesmo valor final:

```
>> int(g,y,-3,4)
```
(para integrar relativamente à variável *y*, há que indicá-la)

```
ans =

(7*exp(2*x))/2

>> int(ans,0,2)

ans =

(7*exp(4))/4 - 7/4
```

O cálculo de integrais duplos em domínios planos que não sejam retângulos segue o mesmo processo processo iterativo do Cálculo:

```
>> syms x y
>> f=x+y
```

f =

x + y

>> int(f,y,x^2,2-x) **(OK)**

ans =

-((x - 1)*(x + 2)*(x^2 + x + 2)) **(expressão fatorizada do polinómio $-x^4/2 - x^3 -+ x^2/2 - x -2$)**

>> int(ans,0,1)

ans =

89/60 **(na forma de fração reduzida)**

Uso da Função integral2 para um modo mais avançado (que também permite uso de parâmetros):

```
>> syms x y
>> f=@(x,y)(x+y)
```

f =

@(x,y)(x+y) **(OK)**

>> ymax=@(x) (2-x)

ymax =

@(x)(2-x) **(OK)**

>> ymin=@(x) x.^2

ymin =

@(x)x.^2 **(OK)**

>> integral2(f,0,1,ymin,ymax)

ans =

1.4833 **(na forma de dízima infinita periódica)**

Pela ordem inversa obtemos, obviamente, o mesmo valor final (...)

Uso da Função integral num modo mais avançado,

```
>> syms x y
>> f=@(x,y) 2./(x+y+2).^2).*(sqrt(3.*x+y)
 f=@(x,y) 2./(x+y+2).^2).*(sqrt(3.*x+y)                    |
Error: Unbalanced or unexpected parenthesis or bracket.

>> f=@(x,y) 2./((x+y+2).^2).*(sqrt(3.*x+y))

f =

   @(x,y)2./((x+y+2).^2).*(sqrt(3.*x+y))
```

(OK)

```
>> ymax=@(x)(1-x)

ymax =

   @(x)(1-x)
```

(OK)

```
>> g=integral2(f,0,1,0,ymax)

g =

   0.1550
```

Uso de coordenadas polares no cálculo de integrais duplos:

```
>> syms x y
>> f=@(x,y) 2./((x+y+2).^2).*(sqrt(3.*x+y))

f =

   @(x,y)2./((x+y+2).^2).*(sqrt(3.*x+y))
```

(OK)

```
>> polarf=@(theta,r) f(r.*cos(theta),r.*sin(theta)).*r;
```

(definição das coordenadas polares e produto pelo Jacobiano)

```
>> rmax=@(theta) 1./(sin(theta)+cos(theta));
```

(a partir de y=1-x obtem-se r=1/(sin(theta)-cos(theta))

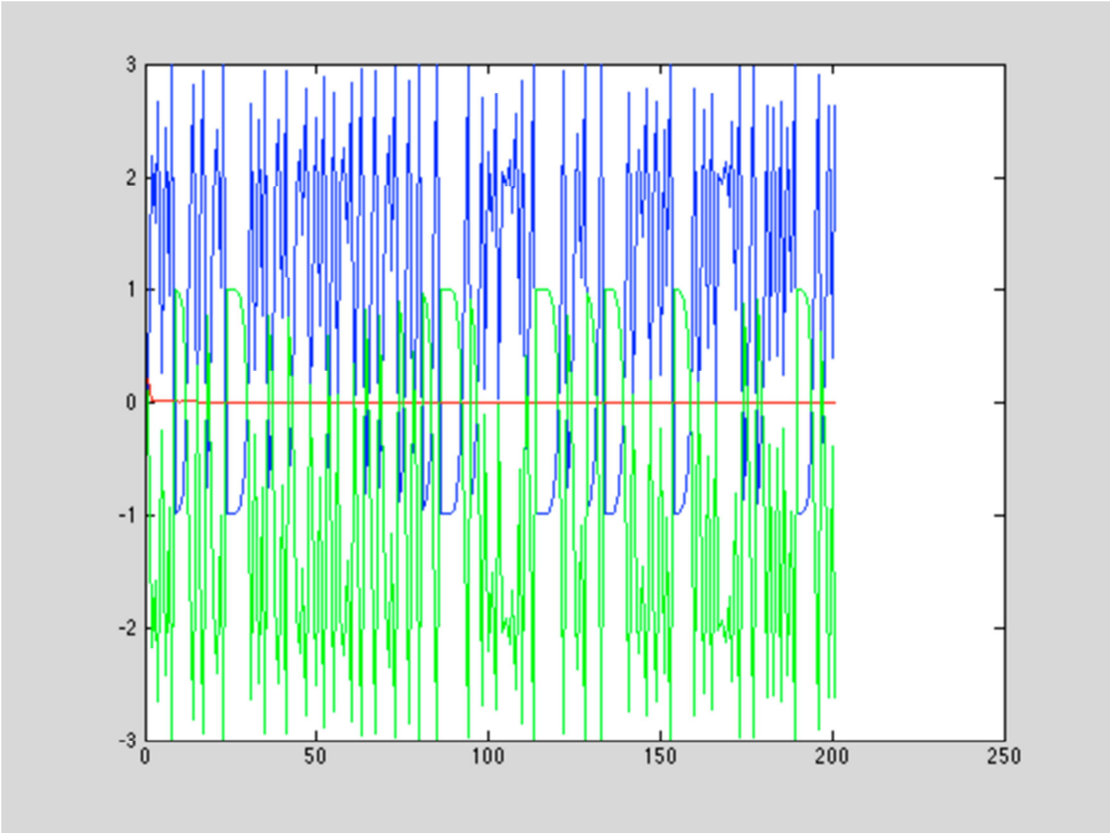
```
>> h=integral2(polarf,0,pi/2,0,rmax)

h =

   0.1550
```

(confirma!)

Editor – pequenos programas que são "corridos" para os valores paramétricos que interessa estudar

SINCRONIZAÇÃO de sistemas caóticos em tempo discreto

```
x(1)=0.1; a=2; N=200; y(1)=0.2; b=2; c=0.1;
for i=1:N;
    x(i+1)=a-(x(i))^2+b*x(i);
    y(i+1)=c*x(i)*y(i)*(1-y(i));
end
figure
plot(x);
hold on
plot(y,'r');
plot(y-x,'g');
```



SINCRONIZAÇÃO de sistemas caóticos em tempo discreto

```
%CASO L2
%Integrates two coupled  LORENZ circuits with synchronization
%Unidireccional por controle feedback negativo deslocado aplicado ‡
%2™ equÁ„o da resposta e ainda com x2 substituida por x1 nos termos
n„o
%lineares da resposta


format compact

options= odeset('Reltol',1.e-5);
%options= odeset('Reltol',1.e-
```

```
22,'OutputFcn','odephas3','Refine',222);
%options= odeset('Reltol',1.e-3,'OutputFcn','odephas3','Refine',2);
%options= odeset('Reltol',1.e-3);

 trans=0*100
 tf=100;
 fac=5;
 c3=25.58;
 DC = 0.1;
 CR = 0.48/2;

 N=3;
 S = 0.1;
 x0 = 0.1+S*(2*(rand(1,1)-0.5));
 y0 = 0.0+S*(2*(rand(1,1)-0.5));
 z0 = 1.0+S*(2*(rand(1,1)-0.5));
 v0 = [x0 y0 z0];
 x0 = 0.1+S*(2*(rand(1,1)-0.5));
 y0 = 0.0+S*(2*(rand(1,1)-0.5));
 z0 = 1.0+S*(2*(rand(1,1)-0.5));
 v0 = [v0 x0 y0 z0]
 tspan=[0:tf/2000:tf];

 v01=v0; v02=v0; v03=v0;
 if(trans)
  [T1,Y1]=ode45('lorenzfeedback_synchro',trans,v01,options,c3,CR-DC);
  [T2,Y2]=ode45('lorenzfeedback_synchro',trans,v02,options,c3,CR);
  [T3,Y3]=ode45('lorenzfeedback_synchro',trans,v03,options,c3,CR+DC);
  v01 = Y1(end,:); v02 = Y2(end,:); v03 = Y3(end,:);
 end

 [T1,Y1]=ode45('lorenzfeedback_synchro',tspan,v01,options,c3,CR-DC);
 [T2,Y2]=ode45('lorenzfeedback_synchro',tspan,v02,options,c3,CR);
 [T3,Y3]=ode45('lorenzfeedback_synchro',tspan,v03,options,c3,CR+DC);

 ifac=floor(length(tspan)/fac);
 Tfac = T1(1:ifac);

 figure(1);
 clf;
  subplot(3,3,1)
  plot(Y1(1:ifac,1),Y1(1:ifac,2),Y1(1:ifac,N+1),Y1(1:ifac,N+2))
  hold on
  plot(Y1(1,1),Y1(1,2),'*',Y1(1,N+1),Y1(1,N+2),'*')
  plot(Y1(ifac,1),Y1(ifac,2),'^',Y1(ifac,N+1),Y1(ifac,N+2),'^')
  axis tight; ylabel('y');
  subplot(3,3,4)
  plot(Y2(1:ifac,1),Y2(1:ifac,2),Y2(1:ifac,N+1),Y2(1:ifac,N+2))
  hold on
  plot(Y2(1,1),Y2(1,2),'*',Y2(1,N+1),Y2(1,N+2),'*')
  plot(Y2(ifac,1),Y2(ifac,2),'^',Y2(ifac,N+1),Y2(ifac,N+2),'^')
  axis tight; ylabel('y');
  subplot(3,3,7);
  plot(Y3(1:ifac,1),Y3(1:ifac,2),Y3(1:ifac,N+1),Y3(1:ifac,N+2))
  hold on
  plot(Y3(1,1),Y3(1,2),'*',Y3(1,N+1),Y3(1,N+2),'*')
  plot(Y3(ifac,1),Y3(ifac,2),'^',Y3(ifac,N+1),Y3(ifac,N+2),'^')
  axis tight; xlabel('x');ylabel('y');

  subplot(3,3,2)
  plot(Y1(1:ifac,1:N),Y1(1:ifac,N+1:2*N))
  hold on
  plot(Y1(1,1:N),Y1(1,N+1:2*N),'*')
  plot(Y1(ifac,1:N),Y1(ifac,N+1:2*N),'^')
  axis tight; ylabel('x2,y2,z2');
  subplot(3,3,5)
  plot(Y2(1:ifac,1:N),Y2(1:ifac,N+1:2*N))
  hold on
```

```
plot(Y2(1,1:N),Y2(1,N+1:2*N),'*')
plot(Y2(ifac,1:N),Y2(ifac,N+1:2*N),'^')
axis tight; ylabel('x2,y2,z2');
subplot(3,3,8)
plot(Y3(1:ifac,1:N),Y3(1:ifac,N+1:2*N))
hold on
plot(Y3(1,1:N),Y3(1,N+1:2*N),'*')
plot(Y3(ifac,1:N),Y3(ifac,N+1:2*N),'^')
axis tight; ylabel('x2,y2,z2'); xlabel('x1,y1,z1');

subplot(3,3,3)
semilogy(T1,sqrt(sum((Y1(:,1:N)-Y1(:,N+1:2*N)).^2,2)))
axis tight; ylabel('||x-y||');
subplot(3,3,6)
plot(T2,sqrt(sum((Y2(:,1:N)-Y2(:,N+1:2*N)).^2,2)))
axis tight; ylabel('||x-y||');
subplot(3,3,9)
semilogy(T3,sqrt(sum((Y3(:,1:N)-Y3(:,N+1:2*N)).^2,2)))
axis tight; ylabel('||x-y||');xlabel('t')
```

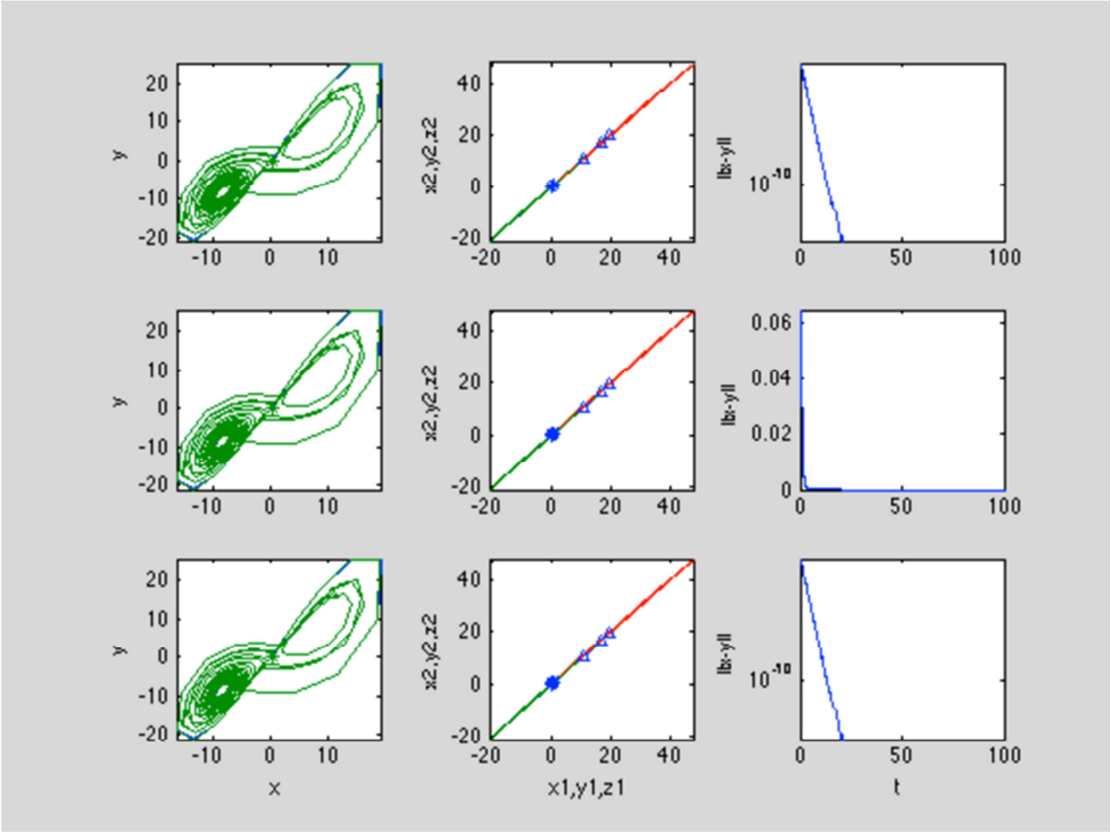


Diagrama de bifurcação da aplicação logística

```
% Program_2d - Bifurcation diagram of the LOGISTIC MAP

clear
itermax=100;
finalits=30; finits=itermax-(finalits-1);
for r=0:0.005:4
    x=0.4;
    xo=x;
    for n=2:itermax
    xn=r*xo*(1-xo);
    x=[x xn];
    xo=xn;
    end
```

```
    plot(r*ones(finalits),x(finits:itermax),'.','MarkerSize',1)
    hold on
end
fsize=15;
set(gca,'xtick',[0:1:4],'FontSize', fsize)
set(gca,'ytick',[0,0.5,1],'FontSize', fsize)
xlabel('{\mu}','FontSize', fsize)
ylabel('\itx','FontSize', fsize)
hold off
% End of Program_2d.
```