Comunicado Técnico 104
ISSN 1677-8464
Dezembro, 2010
Campinas, SP

Foto: <www.sxc.hu>

Modelagem e solução numérica de sistemas biológicos utilizando ferramentas *open source*

Rafael de Oliveira Silva[1]
Sônia Ternes[2]

Introdução

A modelagem matemática desempenha um importante papel na pesquisa biológica. Uma vez que o sistema estudado é representado em linguagem matemática, torna-se possível prever seu comportamento por meio de simulações. Nesse contexto, surge a necessidade do uso de softwares robustos, capazes de computar, construir gráficos, resolver sistemas e equações, assim como gerar aproximações numéricas.

Existe, no mercado, uma infinidade de softwares matemáticos, sistemas computacionais numéricos e algébricos e ferramentas de simulação. Há produtos comerciais muito utilizados pela comunidade científica como Mathematica (WOLFRAM, 1999) e Matlab (THE MATHWORKS, 1998), e softwares não comerciais que podem ser utilizados como alternativa às ferramentas proprietárias. Além disso, as ferramentas não comerciais estão disponíveis sob uma licença de software livre (WEBER, 2004), possibilitando que qualquer usuário possa copiar, alterar e distribuir sem restrições, contribuindo para complementação e consequente melhoria da ferramenta, além de permitir a autonomia tecnológica.

Seppelt e Richter (2005) mostraram que resultados diferentes podem ser encontrados dependendo do software de modelagem escolhido. Nesse artigo é avaliado o potencial das ferramentas livres Maxima, Octave, Scilab e Sage em encontrar soluções numéricas para sistemas de equações diferenciais não lineares, problemas frequentes em modelagem de sistemas biológicos.

Metodologia

Primeiramente, foram pesquisadas e selecionadas as principais plataformas de softwares comerciais e não comerciais disponíveis para a comunidade científica. Os requisitos observados foram organizados numa tabela contendo: nome, software livre equivalente/software proprietário equivalente, sistema operacional (Windows, Linux, MacOS X), tipo de licença, tipo de interface Command Line Interface (CLI) ou Graphical User Interface (GUI), vantagem, desvantagem e forma de aquisição. Após ampla pesquisa, foram selecionadas, para avaliação, as plataformas Maxima, Octave, Scilab e Sage, descritos resumidamente na Tabela 1.

O modelo utilizado para teste foi uma modificação do sistema presa-predador proposto por Lotka-Volterra (VOLTERRA, 1927), que descreve a dinâmica da predação de uma espécie sobre outra por um sistema

[1] *Bolsista PIBIC/Embrapa Informática Agropecuária, Campinas, SP*
[2] *Doutora em Engenharia Elétrica, Pesquisadora da Embrapa Informática Agropecuária, Campinas, SP; sonia@cnptia.embrapa.br*

Embrapa

Tabela 1. Características das ferramentas escolhidas para avaliação.

Nome	Software equivalente	Sistema operacional	Licença	Interface	Vantagem	Desvantagem	Forma de aquisição
Máxima	Mathematica	Windows/Linux/ MacOS X	GNU/GPL	CLI/GUI	Manipulação simbólica, interface simples	Possui menos pacotes específicos que Mathematica	http://maxima.sourceforge.net/
Octave	Matlab	Windows/Linux/ MacOS X	GNU/GPL	CLI/GUI	Linguagem com grande compatibilidade com Matlab. Assim como Matlab, possui vários pacotes, inclusive de biomatemática	Possui menos pacotes específicos que Matlab. Não há uma ferramenta equivalente ao Simulink, presente no Matlab	http://www.gnu.org/software/octave
Scilab	Matlab	Windows/Linux/ MacOS X	GNU/GPL	CLI	Possui extensa biblioteca com aplicações em diversas áreas, como biologia, medicina, energia, finanças, química, etc. Ferramenta Scicos para simulação em bolcos (semelhante ao Simulink do Matlab)	Possui menos pacotes específicosque Matlab. Ainda não há interface gráfica	http://www.scilab.org/
Sage	Mathematica/ Matlab/Magma/ Maple	Windows/Linux/ MacOS X	GNU/GPL	CLI/GUI	Propões ser uma alternativa às ferramentas Magma, Maple, Mathematica and Matlab. Consegue tirar proveito de processadores multicore	Possui menos pacotes específicos que Matlab. Ainda não há interface gráfica	http://www.sagemath.org/

de equações diferenciais não lineares. Dado x como o número de presas, e y o número de predadores, o sistema de equações diferenciais ordinárias descrito pelas fórmulas em (1) descreve a dinâmica da interação entre as duas espécies:

$$dx/dt = r(1 - x/K)x - af(x)y \quad (1)$$

$$dy/dt = \gamma af(x)y - \mu y$$

No sistema descrito em (1), todos os parâmetros são positivos, **r** é a taxa de crescimento da presa, **K** é a capacidade do meio em relação à presa na ausência de predadores, **a** é a taxa máxima de consumo de presas por um único predador , **γ** é o coeficiente de eficiência da predação, **μ** é a taxa de mortalidade de predadores. Para **f(x)**, que representa a resposta da população de presas em relação à predação, foi utilizada uma resposta funcional sigmal do tipo Holling III (HOLLING, 1959), porque respostas sigmais podem estabilizar o equilíbrio presa-predador em modelos de Lotka-Volterra (MAY, 1974; MURDOCH; OATEN, 1975; TAKAHASHI, 1964). Assim:

$$f(x) = x^2/ (x^2+L) \quad (2)$$

onde **L** é a constante de meia-saturação, isto é, o nível de presa em que ocorre metade da taxa de consumo.

O modelo dado por (1) e (2) é o mesmo utilizado por Seppelt e Richter (2005), e por motivo de comparação também foram utilizados os mesmos valores para as constantes do sistema (1), assim como os valores iniciais das populações de presa e predadores.

O modelo proposto é resolvido numericamente pelo método de Runge-Kutta (RUGGIERO; LOPES, 1996), que é uma família de métodos iterativos para resolução numérica (aproximação) de soluções de equações diferenciais ordinárias. Tais métodos requerem apenas derivadas de primeira ordem e podem fornecer aproximações precisas com erros de truncamento da ordem h2, h3, h4 etc, onde h é o passo.

Considere a seguinte equação diferencial ordinária:

f ' = f(t ,z) com z(b) = t0 = β, no intervalo de tempo b ≤ t ≤ c, [*b*, *c*] ∈ IR

O método de Runge-Kutta de 4º ordem (Press et al. 1992), conhecido também como RK4, usa as fórmulas:

$$t(j+1) = t(j) + h,$$

$$z(j+1) = z(j) + h/6(k1+2k2+ 2k3+k4)$$

onde z(j+1) é a aproximação por RK4 de z(k+1), e:

k1 = f(t(j),z(j))
k2 = f(t(j)+h/2 , z(j)+hk1/2)
k3 = f(t(j)+h/2 , z(j)+hk2/2)
k4 = f(t(j)+h , z(j)+k3)

Então, o próximo valor, z(j+1) é determinado pelo valor atual, z(j), somado ao produto do tamanho do intervalo h e uma inclinação estimada. A inclinação é uma média ponderada de inclinações:

- k1 é a inclinação no início do intervalo;
- k2 é a inclinação no ponto médio do intervalo, usando a inclinação k1 para determinar o valor de z no ponto t(j) + h/2 pelo método de Euler (RUGGIERO; LOPES, 1996);
- k3 é novamente a inclinação no ponto médio do intervalo, mas agora usando a inclinação k2 para determinar o valor de z;
- k4 é a inclinação no final do intervalo, com seu valor z determinado usando k3.

Ao fazer a média das quatro inclinações, um peso maior é dado para as inclinações no ponto médio: inclinação = 1/6(k1+2k2+2k3+k4)

O método RK4 é um método de quarta ordem, significando que o erro por passo é da ordem de $h5$, enquanto o erro total acumulado tem ordem $h4$. Neste artigo será utilizado passo h = 0.1.

A Tabela 2 mostra a descrição, os valores escolhidos e as dimensões para as constantes do sistema (1).

Em posse do modelo escolhido para teste, dos valores das constantes (Tabela 2) e do método numérico para solução do sistema (1), as ferramentas selecionadas foram testadas com relação à eficiência para solucionar o problema estudado. Para cada ferramenta foi implementado o método de Runge-Kutta de 4º ordem (RK4) e foram construídos os gráficos para os valores das populações de presas x(t) e predadores y(t), para um tempo de simulação de T=2000 passos.

Softwares testados

Maxima

A ferramenta Maxima tem como objetivo efetuar manipulação algébrica, diferenciação, integração, séries, transformadas, equações diferenciais ordinárias, e operações com matrizes, além da construção de gráficos em 3D.

A sua linguagem de programação possui sintaxe baseada em ALGOL (WEXELBLAT, 1981) e semântica em Lisp (MCCARTHY, 1958). Como Maxima foi desenvolvido em Lisp, comandos puros dessa linguagem podem ser executados diretamente no software.

Para aplicar o método de Runge-Kutta de 4º ordem, fez-se uso da função “rk( )”. Para utiliza-la foi preciso carregar o pacote “dynamics.lisp” pelo comando “load( )”. O código para resolução do sistema (1), via algoritmo de Runge-Kutta e plotagem dos valores, é mostrado na Figura 1.

Tabela 2. Valores dos parâmetros do sistema(1).

Parâmetros	Descrição	Valor	Unidade
r	Taxa de crescimento da presa	0.006	[tempo-1]
K	Capacidade de presas	1000	[no. de pres.]
a	Taxa de predação	5	[tempo-1]
L	Limitação de sucesso de predação	50	[no. de pres.²]
γ	Coeficiente de eficiência de predação	0.2	[-]
μ	Taxa de mortalidade do predador	0.9	[tempo-1]
xo	População inicial de presas	10	[no. de pres.]
yo	População inicial de predadores	0.02	[no. de pred.]
T	Tempo de simulação	2000	[tempo]

```
load("dynamics")$
sol : rk([0.06*x*(1-x/1000) -(5*(x^2)*y)/( (x^2) + 50),
( 0.2*5*y*(x^2) )/(x^2 + 50) - 0.9*y],[x,y],[10,0.02],[t,0,2000,0.1])$
l1 : makelist([ sol[i][1] , sol[i][2] ] , i ,1 ,length(sol) )$
l2 : makelist([ sol[i][1] , sol[i][3] ] , i ,1 ,length(sol) )$
l12 : makelist([ sol[i][2] , sol[i][3] ] , i ,1 ,length(sol) )$
load("graph2d")$
graph2d(l1,l2)$
```

Figura 1. Código para solução numéria e plotagem dos resultados do sistema (1) utilizando o método de RK4 no software Maxima.

Octave

A ferramenta Octave tem como objetivo encontrar soluções numéricas para problemas lineares e não lineares, manipular polinômios, EDO's, diferenciação, integração, realizar experimentos numéricos. Plotagem 2D e 3D.

Octave é uma linguagem de alto nível, voltada para computação numérica. É praticamente idêntica à linguagem Matlab, os programas e funções possuem extensão .m assim como no Matlab. Rotinas em linguagem C, C++ e Fortran podem ser escritas em Octave utilizando módulos dinamicamente carregados.

No teste da ferramenta, utilizou-se trechos de código implementados previamente para execução no software Matlab, visando testar a compatibilidade entre os dois softwares.

Nesse código, o sistema (1) é definido como uma função vetorial, conforme mostrado na Figura 2a.

Em seguida, aplicou-se o método de RK4, chamando a função ode45( ) e plotou-se os resultados, conforme os comandos da Figura 2b.

O código acima não pode ser executado inicialmente no Octave. O erro foi gerado devido a função ode45. Embora esta função exista em Octave, com o mesmo nome no pacote "odepkg"[1], seus argumentos são diferentes. A alternativa usada para que o código escrito em linguagem Matlab funcionasse em Octave, sem alteração, foi realizar uma busca na internet pela função ode45.m que foi encontrada no endereço[2]. Dessa forma o código da Figura 2b funcionou perfeitamente.

Scilab

A ferramenta Scilab tem como objetivo encontrar soluções numéricas para problemas lineares e não lineares, manipular polinômios, EDO's, realizar experimentos numéricos e construir gráficos 2D e 3D.

Scilab é uma linguagem de alto nível. Embora seja parecida com o Matlab, seu desenvolvimento é independente e não tem como objetivo ser compatível com Matlab, ao contrário da Octave. Os arquivos escritos em Scilab possuem extensão "sci".

O sistema (1) é definido como função vetorial na linguagem do Scilab, conforme o trecho de código mostrado pela Figura 3a.

Em Scilab, para resolver sistemas de equações diferenciais ordinárias, deve-se usar a função ode.sci. Nesta, o primeiro argumento refere-se ao método de solução numérica. Para aplicar o algoritmo RK4, por exemplo, o argumento usado é "rk", conforme mostrado na Figura 3b.

Sage

Sage é uma ferramenta para uso em matemática pura e aplicada, álgebra, cálculo, teoria dos números, cripto-

a)

```
function ydot = syspp(t,y)
ydot(1)=.06*y(1)*(1-y(1)/1000)-(5*(y(1)^2)*y(2))/( (y(1)^2) + 50);
ydot(2)=( .2*5*y(2)*(y(1)^2) )/(y(1)^2 + 50) - .9*y(2);
ydot = [y(1) y(2)];
```

b)

```
clear;
t0 = 0;
tf = 2000;
y0 = [10  0.02];


sol = [t y] = ode45('syspp',[t0 tf],y0);
plot(t,y(:,1),t,y(:,2))
```

Figura 2. a) sistema (1) escrito em forma de função vetorial. b) Corpo do programa feito em Octave.

a)

```
function xdot = syspp(t,x)
xdot(1) = 0.06*x(1)*(1-x(1)/1000) -(5*(x(1)^2)*x(2))/( (x(1)^2) + 50);
xdot(2) = ( 0.2*5*x(2)*(x(1)^2) )/(x(1)^2 + 50) - 0.9*x(2);
```

b)

```
exec("syspp.sci");

x0 = [10 , 0.02]';
t0 = 0;
T = 0: 0.1 : 2000;
T = T';

sol = ode("rk",x0,t0,T,syspp);
sol =sol';

plot2d(T,sol);
```

Figura 3. a) sistema (1) escrito em forma de função vetorial. b) Corpo do programa feito em Scilab.

[1] *Disponível em: <http://octave.sourceforge.net/odepkg/index.html>*

[2] *Disponível em: <http://cns.bu.edu/~tanc/pub/matlab_octave_compliance/ode_v1.11/ode45.m>*

grafia, computação numérica, grafos, e construção de gráficos 2D e 3D.

O sistema Sage utiliza a linguagem de alto nível Phyton. Portanto, para domínio do ambiente Sage, é preciso ter conhecimentos dessa linguagem de programação.

Nessa ferramenta a implementação do método de Runge Kutta de 4º ordem é a função desolve_system_rk4( ), conforme mostrado na Figura 4.

Resultados

Resolver sistemas de equações ordinárias não lineares, como é o caso do sistema (1), é tarefa extremamente difícil, por isso foram utilizados métodos numéricos. A simulação foi feita com tempo igual a 2000 iterações. Nas quatro ferramentas avaliadas, o resultado foi de oscilação entre as populações simuladas de presas e predadores (Figura 5).

```
x,y,t = var('x y t')
r = 0.06
K = 1000
a = 5
L = 50
g = 0.2
m = 0.9
x_0 = 10
y_0 = 0.02

P = desolve_system_rk4([ r*x*(1-x/K)-a*(x^2)*y/((x^2) + L) ,
g*a*y*(x^2)/(x^2 + L) - m*y ],[x,y],ics=[0,x_0,y_0],ivar=t,end_points=2000)

Q1 = [ [i,j] for i,j,k in P]
Q2 = [ [i,k] for i,j,k in P]
Lista1 = list_plot(Q1,plotjoined=True,color='blue')
Lista2 = list_plot(Q2,plotjoined=True,color='red')
show(Lista1+Lista2)
```

Figura 4. Código feito em ambiente Sage, linguagem python.

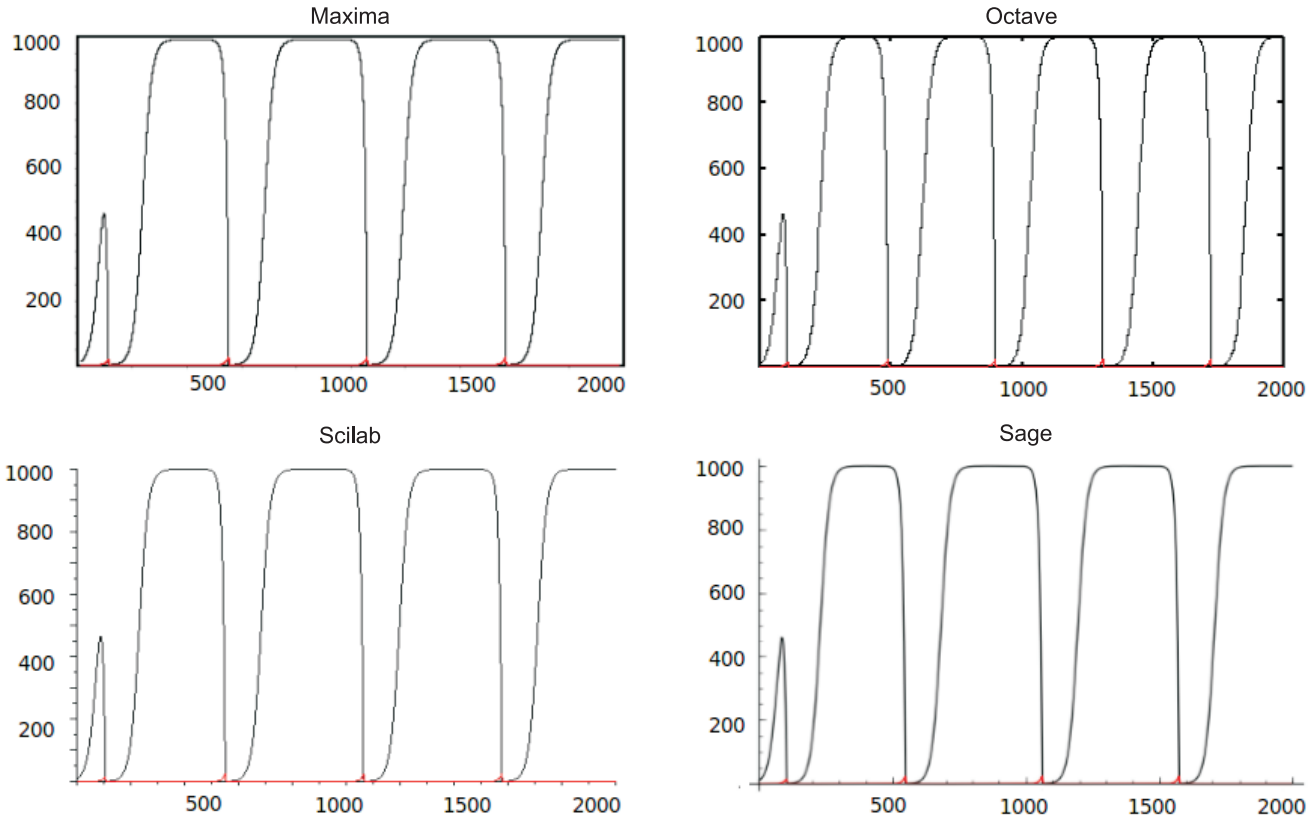

Figura 5. Plotagens das populações de presas x(t) e predadores y(t), em função do tempo, geradas pelo método de RK4 nas ferramentas avaliadas.

Observa-se pelas plotagens que com os parâmetros utilizados e as condições iniciais do modelo Presa-Predador, as populações se encontram em equilíbrio. É possível interpretar os gráficos da Figura 5 da seguinte forma: com disponibilidade de presas, o número de predadores aumenta; as presas têm, então, menor taxa de crescimento e sua população diminui. Depois de um certo tempo, a população de predadores atinge um máximo, e, por falta de alimento, começa também a diminuir. Diminuída a população de predadores, o número de presas começa a aumentar e o ciclo se repete.

Embora o passo h do método de Runge-Kutta tenha sido controlado nas ferramentas avaliadas, com h=0.1, a precisão de ponto flutuante não foi controlada, ou seja, foram utilizados os valores "default" de cada um dos softwares, o que pode ter resultado nas diferenças dos valores aproximados para os valores máximos de y(t) e o período T das oscilações (Tabela 3).

Tabela 3. Valores aproximados do período de oscilação e valores máximos de x(t) e y(t).

Ferramenta	Período T	Max x	Max y
Máxima	515	1000	22
Octave	412	1000	22
Scilab	515	1000	16
Sage	515	1000	22

Discussão

Uma importante questão a ser levantada sobre as ferramentas livres disponíveis para a comunidade científica é se elas podem substituir eficientemente as ferramentas proprietárias. Testes de comparação são necessários para avaliação da eficiência de ferramentas livres.

Sistemas não lineares complexos podem não ter soluções analíticas e, dessa forma, é preciso confiar nas soluções obtidas por meio de software de resolução numérica. As quatro ferramentas avaliadas neste trabalho tiveram resultados satisfatórios: a implementação do método de Runge-Kutta de 4º ordem foi facilmente encontrada e foram obtidos, como resultado, o bem conhecido comportamento oscilatório do sistema presa-predador em todas as ferramentas (Figura 5).

Maxima, Octave, Scilab e Sage são ferramentas robustas, bastante usadas pela comunidade científica, com centenas de pacotes aplicáveis às mais diversas áreas, como modelagem ambiental, bioinformática, matemática financeira, inteligência artificial, pesquisa operacional etc. Dessa forma, o principal problema que surge na utilização ou migração de softwares é a dificuldade de se aprender uma nova linguagem, uma vez que tempo e dinheiro são investidos no aprendizado e aperfeiçoamento da linguagem. Seguindo essa linha, a ferramenta Octave propõe ter o máximo de compatibilidade com o consagrado Matlab, já que a linguagem é muito semelhante a essa ferramenta, inclusive programas inteiros feitos em linguagem Matlab podem ser facilmente compilados em Octave, gerando eventualmente algum erro ocasionado por diferenças em nomes ou argumento das funções, como foi o caso verificado em 3.2. De uma forma geral, a dificuldade quanto à programação nas quatro ferramentas é mínima, já que possuem grande variedade de pacotes e funções específicas.

No processo de migração para software livre, é preciso avaliar também suas vantagens. As ferramentas avaliadas neste trabalho têm licença do tipo GNU GPL "General Public License", o que quer dizer que cientistas podem compartilhar e modificar livremente o código fonte, de acordo com suas necessidades específicas. O suporte à criatividade é maior do que nos softwares proprietários, não depende da disponibilidade de recursos financeiros e permite que quaisquer pessoas e instituições possam fazer parte do seu desenvolvimento científico.

Portanto, considerando os argumentos supra citados e os resultados obtidos no sistema teste em comparação com trabalhos anteriores, conclui-se que é viável a migração para as ferramentas avaliadas, das quais a ferramenta Octave apresenta o mínimo esforço de aprendizagem.

Referências

BASSANEZI, R. **Modelagem Matemática**. Blumenau: Dynamis, 1997. 389 p.

HARRISON, G. W. Comparing predator-prey models to Luckinbill's experiment with didinium and paramecium. **Ecology**, v. 76, n. 2, p. 357-374, Mar. 1995.

MATHWORKS Simulink user's guide: the mathworks.Natick, 1998.

MATLAB ReferenceGuide: the MathWorks. Natick, MA, 1992. Paginação irregular.

MCCARTHY, J. **History of lisp**. SIGPLAN Notices, 1978.

MURRAY, J.D. **Mathematical Biology**. Seattle: Springer, 1989.

PRESS W. H.; FIANNERY, B. P.; TEUKOLSKY, S. A.; VETTERLING, W. T. Runge-Kutta method and adaptive step size control for Runge-Kutta. §16.1 and 16.2. In: NUMERICAL recipes in fortran: the art of scientific computing, 2nd ed. Cambridge, England: Cambridge University Press, 1992. p. 704-716.

RITCHIE, D. M. **The development of the C programming language**. ACM, 1996.

RUGGIERO, M. A.G.; LOPES, V. L. R. **Cálculo numérico – aspectos teóricos e computacionais**. 2 ed. São Paulo: Makron Books, 1996. 406 p.

SEPPELT, R.; RICHTER, O. ‘‘It was an artefact not the results’’ - a note on system dynamic model development. **Environmental Modelling and Software**, v. 20, 1543-1548. 2005.

VOLTERRA, V., Variations and fluctuations in the numbers of coexisting animal species. In: SCUDO, F. M.; ZIEGLER, J. R. (Ed.). **The Golden Age of Theoretical Ecology**: 1923–1940. Berlin: Springer-Verlag,1927. p. 65–236.

WEBER, S. **The success of open source**. Harvard University Press, 2004. 320 p.

WEXELBLAT, R. L. (Ed.). **History of programming languages**. New York: Academic Press, 1981. pp. 758.

WOLFRAM, S. **The Mathematica book**. 4th ed. Cambridge: University Press,1999. 1496 p.

ZHU, H.; CAMPBELL, S. A.; WOLKOWICZ, G. S. K. Bifurcation analysis of a predator-prey system with nonmonotonic functional response. **Applied Mathematics**, v. 63, n. 2, p. 636–682 .

Comunicado Técnico, 104

Embrapa Informática Agropecuária
Endereço: Caixa Postal 6041 - Barão Geraldo
13083-886 - Campinas, SP
Fone: (19) 3211-5700
Fax: (19) 3211-5754
http://www.cnptia.embrapa.br
e-mail: sac@cnptia.embrapa.com.br

Embrapa

Ministério da Agricultura, Pecuária e Abastecimento

BRASIL UM PAÍS DE TODOS GOVERNO FEDERAL

1ª edição on-line - 2010

CGPE 9101