Método de Euler Explícito para Resolver EDOs

Victoria Tanaka - 8988806
victoria.tanaka@usp.br
Escola Politécnica - Universidade de São Paulo

O Método de Euler é um algoritmo de primeira ordem, explícito e de um passo para resolução de Equações Diferenciais Ordinárias com condição inicial. Neste trabalho, foi estudado seu comportamento e convergência variando o valor de n.

I. Introdução e Conceitos

Dado o seguinte problema de Cauchy na forma normal:

$$\dot{y}(t) = f(t, y(t)) \quad \text{com} \quad y(t_0) = y_0 \tag{1}$$

O Método de Euler estima uma solução numérica a partir da noção básica de derivada, i. e., podemos aproximar y'(t) por:

$$\dot{y}(t) \approx \frac{y(t + \Delta t) - y(t)}{\Delta t} \tag{2}$$

Quanto menor Δt, melhor a aproximação. O primeiro passo, portanto, é discretizar o domínio de definição do problema. Dado um intervalo $I = [t_0, t_f] \in Dom(y(t))$ e $n \in \mathbb{N}$ o número de passos, define-se $\Delta t = \frac{t_f - t_0}{n}$. Note que o intervalo é igualmente espaçado.

Voltando à ideia da derivada, nota-se que $\dot{y}(t)$ é igual a $f(t, y(t))$, por definição. Podemos então reescrever (2) da seguinte forma:

$$y_{k+1} \approx y_k + \Delta t f(t_k, y_k) \tag{3}$$

Ou seja, a partir de um valor inicial, temos uma relação de recorrência para obter uma estimativa do valor de $y(t_k) \approx y_k$ a partir do anterior.

O Método de Euler pode ser estendido para EDO's de ordens superiores adotando o modelo n-dimensional. Por exemplo, para n=2, temos $\ddot{y}(t) = f(t, y(t), \dot{y}(t))$ e basta introduzir uma variável auxiliar $z_1(t) = \dot{y}(t)$. Teremos o seguinte sistema de equações diferenciais:

$$\dot{\mathbb{Y}}(t) = \begin{pmatrix} \dot{y}(t) \\ \dot{z}_1(t) \end{pmatrix} = \mathbb{F}(t, \mathbb{Y}) = \begin{pmatrix} z_1(t) \\ f(t, y(t), z_1(t)) \end{pmatrix} \tag{4}$$

II. Implementação e testes

O algoritmo foi implementado na linguagem C, organizado em 3 arquivos: main.c, euler.c, util.c. Os gráficos foram gerados usando a biblioteca *matplotlib* do Python (ver apêndice D). As partes relevantes dos códigos podem ser conferidos nos apêndices.

A. Arquivos do projeto

Abaixo segue um breve resumo do conteúdo dos códigos fonte e interfaces.

main.c: apenas implementa a função main(). Veja o código na íntegra no Apêndice C.

euler.c: implementa o algoritmo em si. Um pedaço de sua interface é apresentado a seguir:

```
void
euler(double(*f)(), double(*g)(), Exemplo ex);

double
rate_x(double t, double x, double y, Exemplo
    ex);

double
rate_y(double t, double x, double y, Exemplo
    ex);
```

util.c: implementa funções auxiliares para formatação de arquivos, impressão na tela, entre outros. Sua interface contém também enumeradores e constantes como y_0, x_0, t_0, t_f, n e constantes dos modelos (ver apêndice A).

B. Algoritmo

A implementação do algoritmo em questão é apresentado abaixo. A função recebe como argumento as funções definidas por $\dot{x}(t)$ e $\dot{y}(t)$ (ver apêndice B). "Exemplo" é apenas um enumerador criado na interface *util.h* para definir qual problema está sendo analisado.

```
void euler(double(*f)(double, double, double,
    Exemplo), double(*g)(double, double,
    double, Exemplo), Exemplo ex)
{
    int k;
    double xk, yk, xk_1, yk_1, tk, dt = (TF -
        T0)/N_;
    FILE *arq;

    [...]
    // essa parte do codigo foi suprimida
    // formatacao e manipulacao de arquivos (
        cabecalhos, nomes, possiveis erros de
        abertura, etc)
    // "switch" para cobrir os dois casos
        exemplos (massa-mola e presa-predador)

    xk = X0;
    yk = Y0;
    tk = T0;

    for(k=1; k <= N_; k++) {
        xk_1 = xk + dt*f(tk, xk, yk, ex);
        yk_1 = yk + dt*g(tk, xk, yk, ex);
        tk = T0 + k*dt;
        xk = xk_1;
        yk = yk_1;
        fprintf(arq,"%f %e %e\n", tk, xk, yk);
    }
    fclose(arq);
}
```

C. Testes

Para validar o código numérico para aplicações reais, primeiro aplicamos para vários casos simples com solução analítica conhecida.

O método convergiu rapidamente para os diversos testes, e o tempo de execução foi praticamente instantâneo (da ordem de milisegundos). Vale lembrar, entretanto, que para problemas práticos essa convergência pode não ser tão eficiente. Para equações com comportamentos menos suaves, o tamanho do passo deve ser extremamente pequeno, o que aumenta o tempo de resposta do programa.

Nos exemplos estudados, o tempo de execução se limitou na casa dos milissegundos. Para n muito grande ($\approx 10M$), o tempo foi de ≈ 10 segundos.

III. Exemplos

Os dois problemas abordados por este trabalho são o Modelo Massa-Mola Forçado e o Modelo de Lotka-Volterra. Apesar de simples, ambos os exemplos ilustram EDOs que não podem ser resolvidas facilmente com métodos analíticos aprendidos nos cursos de Cálculo (existe uma solução exata para o caso Massa-Mola, mas calculá-la é extremamente trabalhoso).

A. Modelo Massa-Mola Forçado

O comportamento de um sistema clássico massa-mola pode ser modelado pela seguinte equação diferencial:

$$\ddot{x} = -w_0^2 x + A\cos wt \tag{5}$$

onde x é a posição, A e w são parâmetros da força externa e w_0 é a frequência natural.
Adotando as condições $t_0 = 0$, $t_f = 3$, $x_0 = 0$, $y_0 = 1$ e parâmetros do modelo $w_0^2 = 5$, $w = 1$ e $A = 2$, variando o valor de n temos os seguintes resultados:

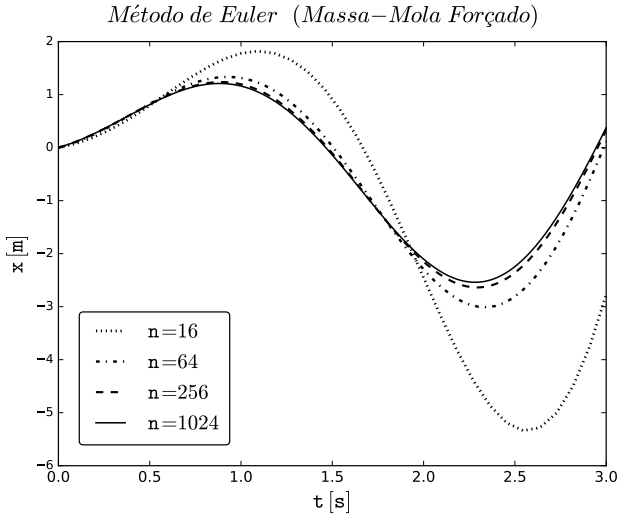

Figura 1: Posição x em função do tempo do sistema massa-mola forçado para diferentes valores de n. Observou-se que para valores maiores que 1024 as soluções se sobrepõem.

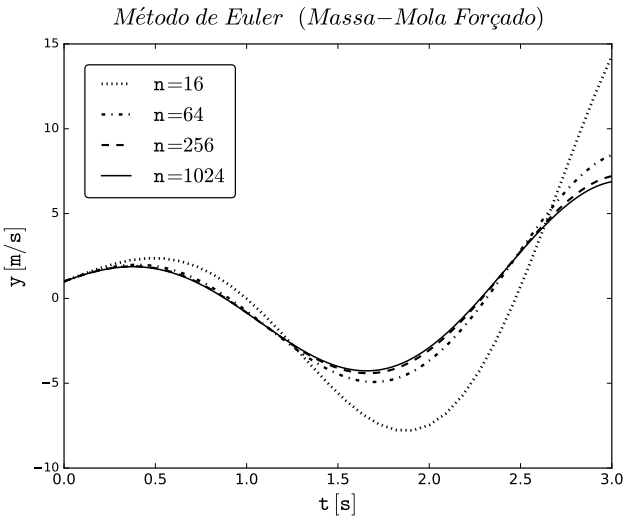

Figura 2: Velocidade y em função do tempo do sistema massa-mola forçado para diferentes valores de n. Observou-se que para valores maiores que 1024 as soluções se sobrepõem.

Para observar melhor o comportamento do sistema, podemos aumentar o intervalo de estudo e o Δt:

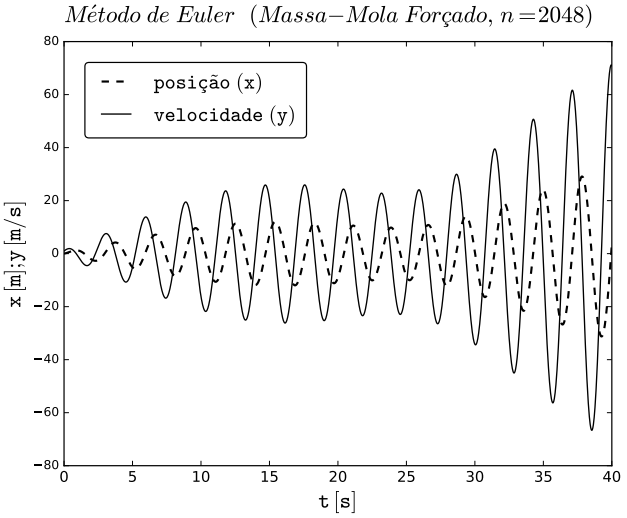

Figura 3: Posição x e velocidade y em função do tempo para o sistema massa-mola forçado.

B. Modelo de Lotka-Volterra

Em primeira instância, a dinâmica biológica de um sistema presa-predador pode ser modelada pelas seguintes equações:

$$\begin{cases} \dot{x} = x(\alpha - \beta y) \\ \dot{y} = y(\delta x - \gamma) \end{cases} \tag{6}$$

onde x é o número de presas, y é o número de predadores e $\alpha, \beta, \gamma, \delta \in \mathbb{R}^+$.

Apresentaremos aqui apenas a evolução das equações para ilustrar um exemplo de um sistema de EDOs que não tem solução analítica.

Adotando as condições $t_0 = 0$, $t_f = 140$, $x_0 = 10$, $y_0 = 10$, $n = 2500$ e parâmetros do modelo $\alpha = 0.1$, $\beta = 0.02$, $\gamma = 0.4$ e $\delta = 0.02$ temos:

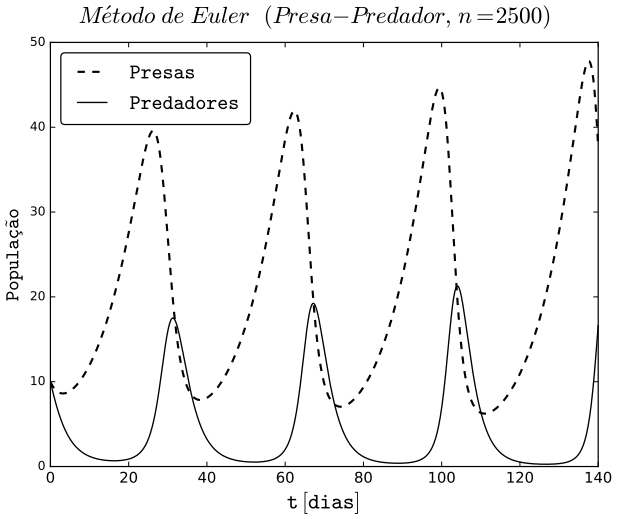

Figura 4: Evolução das populações de predador e presa no modelo simples de Lotka-Volterra.

Nota: para este exemplo, o comportamento das soluções variou fortemente com os valores escolhidos para os parâmetros $(\alpha, \beta, \gamma, \delta)$ do modelo. Para alguns casos, $x(t)$ e $y(t)$ divergiram muito para valores pequenos de n, o que indica que o problema requer um valor de Δt pequeno.

[1] "Max232 texas instruments." `http://www.ti.com/lit/ds/symlink/max232.pdf`, 2014. [Online; accessed 2016-09-25].

[2] "Db9 pinout." `http://www.db9-pinout.com/`. [Online; accessed 2016-09-28].

[3] "Manual da placa de2 da altera," 2008.

[4] "Aplicação da metodologia de projeto com dispositivos programáveis." `http://www.pcs.usp.br/~labdig/pdffiles_2016/metodologia-projeto-v1.pdf`, 2016. [Online; accessed 2016-10-16].

[5] "Conceitos de comunicação serial assíncrona." `http://www.pcs.usp.br/~labdig/pdffiles_2016/conceitos-comunicacao-serial-v2.pdf`, 2016. [Online; accessed 2016-10-16].