Ciência Computacional: Modelagem e Simulação

Roberto M. Cesar Jr.
rmcesar@usp.br

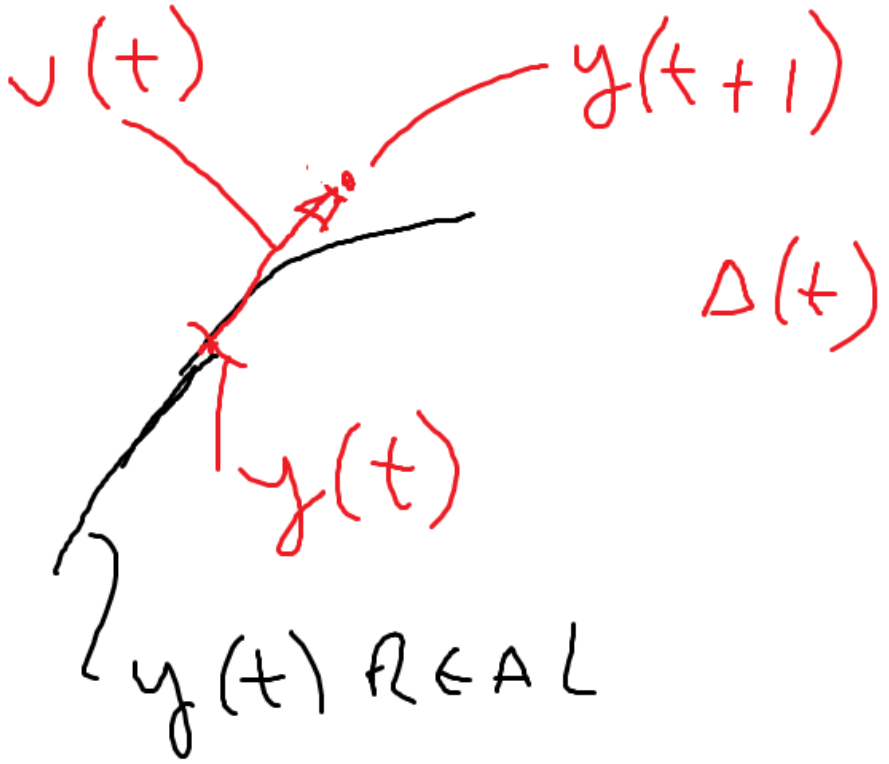
Variações do algoritmo de Euler

$$v(t + \Delta t) = v(t) + a(t)\Delta t$$
$$y(t + \Delta t) = y(t) + v(t)\Delta t,$$

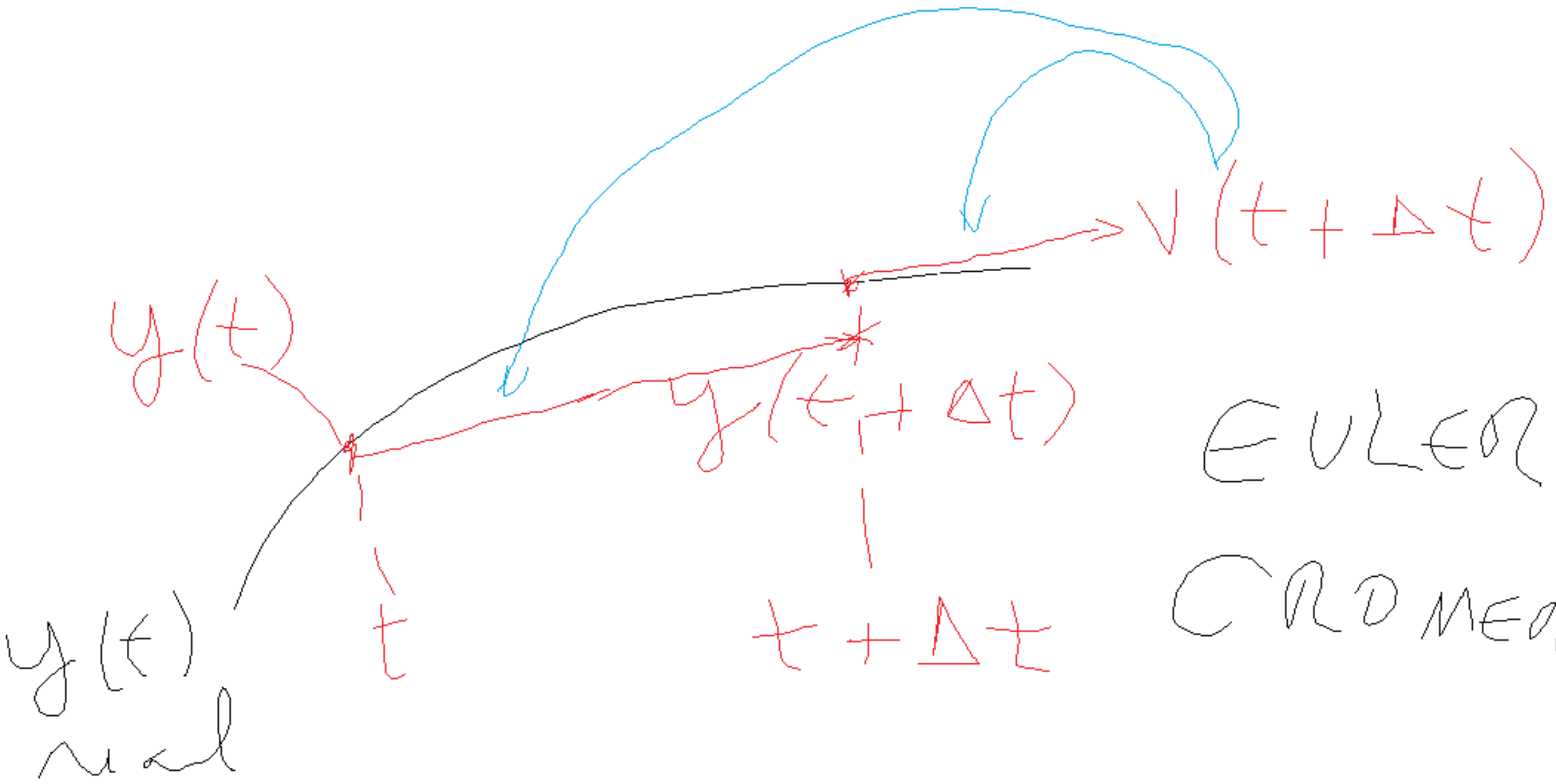
Algoritmo de Euler modificado: Euler-Cromer

To illustrate why we need algorithms other than the simple Euler algorithm, we make a very simple change in the Euler algorithm and write

$$v(t + \Delta t) = v(t) + a(t)\Delta t \tag{3.1a}$$
$$y(t + \Delta t) = y(t) + v(t + \Delta t)\Delta t, \tag{3.1b}$$



ALGORITMO DE EULER



EULER
CROMER

Problem 3.1. Comparing Euler algorithms

a. Write a class that extends `Particle` and models a simple harmonic oscillator for which $F = -kx$. For simplicity, choose units such that $k = 1$ and $m = 1$. Determine the numerical error in the position of the simple harmonic oscillator after the particle has evolved for several cycles. Is the original Euler algorithm stable for this system? What happens if you run for longer times?

b. Repeat part (a) using the Euler-Cromer algorithm. Does this algorithm work better? If so, in what way?

c. Modify your program so that it computes the total energy, $E_{\text{sho}} = v^2/2 + x^2/2$. How well is the total energy conserved for the two algorithms? Also consider the quantity $\tilde{E} = E_{\text{sho}} + (\Delta t/2)xp$. What is the behavior of this quantity for the Euler-Comer algorithm?

SISTEMAS DINÂMICOS

Sistemas Dinâmicos

- Processos que evoluem em função de alguma dimensão (ex. Tempo)
- A lei de evolução é definida por um modelo matemático como uma função ou equação diferencial
- Contínuos ou discretos
- Determinísticos ou estocásticos

Analisando Algoritmos

- Existe um algoritmo que resolve meu problema?
- Exemplo: o algoritmo vai parar algum dia?

```
Algoritmo simples
 1- enquanto x for diferente de 1 faca
      x = x - 2
 2- Imprima "Ola, cheguei!"
Fim
```

- Se x for inicialmente um número ímpar e positivo, o algoritmo irá imprimir a mensagem. Senão for, não irá imprimir nunca.
- Nós sabemos exatamente o que vai acontecer!

Analisando Algoritmos

- Exemplo 2: o algoritmo vai parar algum dia?

```
Algoritmo nao tao simples
 1- enquanto x for diferente de 1 faca
      Se x for par,
       ~~x = x - 2~~          x = x / 2
      senao
       x = 3x + 1
 2- Imprima "Ola, cheguei!"
Fim
```

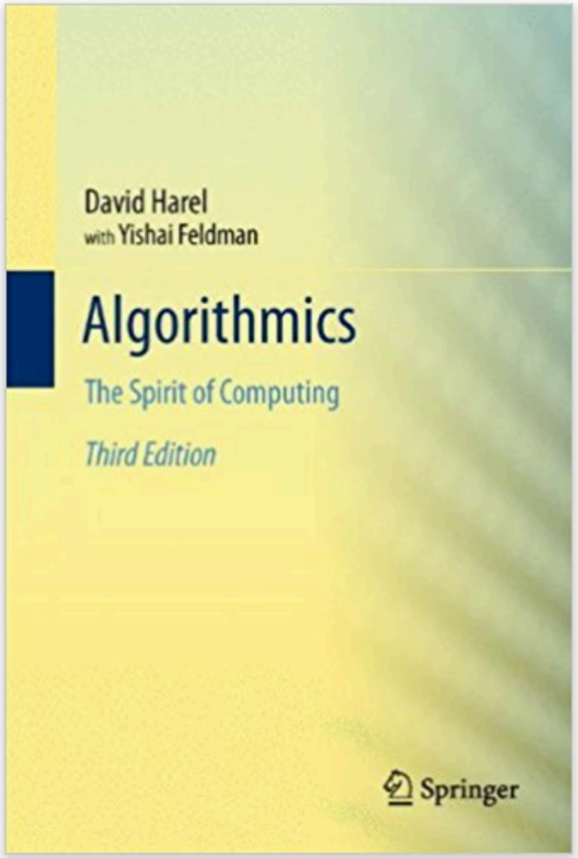
O que é fácil, o que é difícil...

- Está aqui um exemplo de seqüência produzida pelo algoritmo: 7, 22, 11, 34, 17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1
- Esse algoritmo já foi testado inúmeras vezes com diversos números de entrada, e sempre parou, mas...
- até hoje, ninguém conseguiu provar que ele sempre parará para qualquer número de entrada!

Algorithmics: The Spirit of Computing 3rd ed. 1987 Edition

by David Harel (Author), Yishai Feldman (Author)

9 customer reviews



ISBN-13: 978-3642272653
ISBN-10: 3642272657
Why is ISBN important?

Hardcover	Paperback	Other Sellers
$66.49	$65.70 - $79.95	See all 3 versions

Buy new

Only 3 left in stock (more on the way).
Ships from and sold by Amazon.com. Gift-wrap available.

This item ships to **São Paulo, Brazil. Want it Thursday, April 19?** Order within **23 hrs 42 mins** and choose **AmazonGlobal Priority Shipping** at checkout. Learn more

More Buying Choices

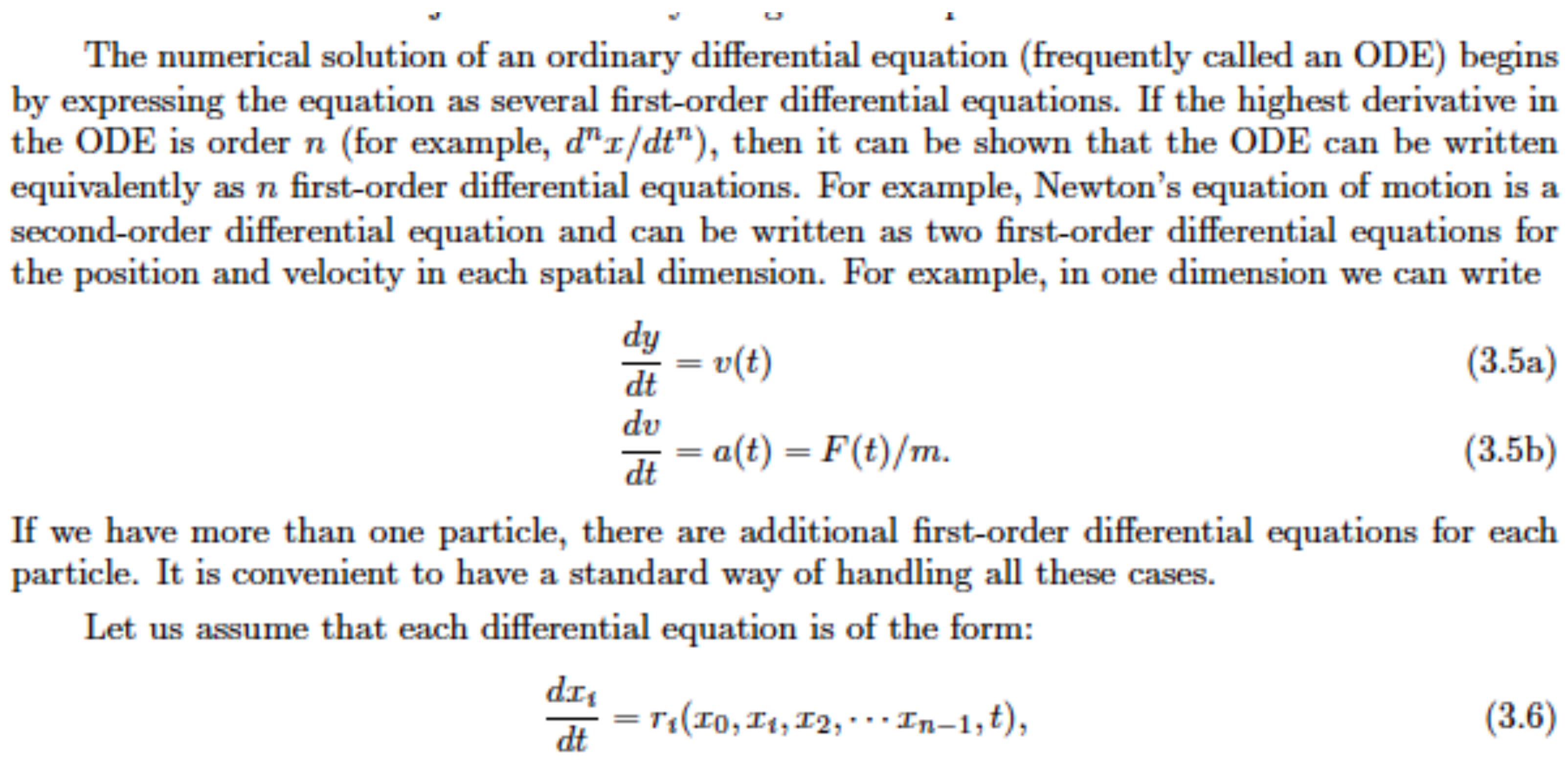
22 New from $62.49 | 20 Used from $60.01

Sistemas dinâmicos

- $x(n+1) = f(x(n))$
- x pode ser vetorial

  x = (x1, x2, …, xn)
- **x** é denominado **vetor de estados**
- Espaço de estados
- Trajetórias

3.4 Specifying the state of a system using arrays



The numerical solution of an ordinary differential equation (frequently called an ODE) begins by expressing the equation as several first-order differential equations. If the highest derivative in the ODE is order n (for example, $d^n x/dt^n$), then it can be shown that the ODE can be written equivalently as n first-order differential equations. For example, Newton's equation of motion is a second-order differential equation and can be written as two first-order differential equations for the position and velocity in each spatial dimension. For example, in one dimension we can write

$$\frac{dy}{dt} = v(t) \tag{3.5a}$$

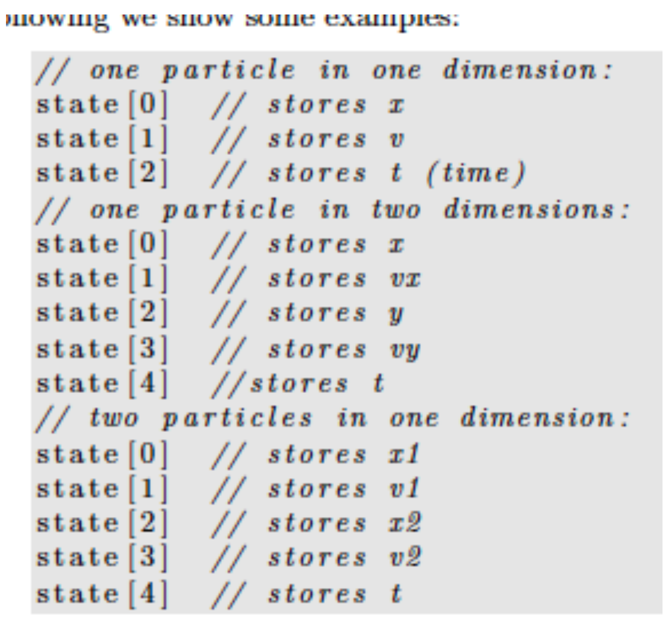
$$\frac{dv}{dt} = a(t) = F(t)/m. \tag{3.5b}$$

If we have more than one particle, there are additional first-order differential equations for each particle. It is convenient to have a standard way of handling all these cases.

Let us assume that each differential equation is of the form:

$$\frac{dx_i}{dt} = r_i(x_0, x_1, x_2, \cdots x_{n-1}, t), \tag{3.6}$$

3.4 Specifying the state of a system using arrays



llowing we show some examples:

```
// one particle in one dimension:
state[0]   // stores x
state[1]   // stores v
state[2]   // stores t (time)
// one particle in two dimensions:
state[0]   // stores x
state[1]   // stores vx
state[2]   // stores y
state[3]   // stores vy
state[4]   //stores t
// two particles in one dimension:
state[0]   // stores x1
state[1]   // stores v1
state[2]   // stores x2
state[3]   // stores v2
state[4]   // stores t
```

3.5 The ODE interface

To introduce the ODE interface, we again consider the equations of motion for a falling particle. We use a state array ordered as $s = (y, v, t)$, so that the dynamical equations can be written as:

$$\dot{s_0} = s_1 \tag{3.7a}$$

$$\dot{s_1} = -g \tag{3.7b}$$

$$\dot{s_2} = 1. \tag{3.7c}$$

```
package org.opensourcephysics.sip.ch03;
import org.opensourcephysics.numerics.*;

public class FallingParticleODE implements ODE {
   final static double g = 9.8;
   double[] state = new double[3];

   public FallingParticleODE(double y, double v) {
      state[0] = y;
      state[1] = v;
      state[2] = 0;               // initial time
   }

   public double[] getState() { // required to implement ODE interface
      return state;
   }

   public void getRate(double[] state, double[] rate) {
      rate[0] = state[1]; // rate of change of y is v
      rate[1] = -g;
      rate[2] = 1;          // rate of change of time is 1
   }
}
```

Listing 3.6: The ODE solver interface. Note the four methods that must be defined.

```
package org.opensourcephysics.numerics;
public interface ODESolver {
   public void initialize(double stepSize);

   public double step();

   public void setStepSize(double stepSize);

   public double getStepSize();
}
```

```
public class FallingParticleODEApp extends AbstractCalculation {
   public void calculate() {
      // gets initial conditions
      double y0 = control.getDouble("Initial y");
      double v0 = control.getDouble("Initial v");
      // creates ball with initial conditions
      FallingParticleODE ball = new FallingParticleODE(y0, v0);
      // creates ODE solver
      ODESolver solver = new Euler(ball); // note how particular algorithm is chosen
      // sets time step dt in the solver
      solver.setStepSize(control.getDouble("dt"));
      while(ball.state[0]>0) {
         solver.step();
      }
      control.println("final time = "+ball.state[2]);
      control.println("y = "+ball.state[0]+" v = "+ball.state[1]);
   }

   public void reset() {
      control.setValue("Initial y", 10);     // sets default input values
      control.setValue("Initial v", 0);
      control.setValue("dt", 0.01);
   }
```

```
   public static void main(String[] args) { // creates a calculation control
      CalculationControl.createApp(new FallingParticleODEApp());
   }
}
```

```java
public class Projectile implements Drawable, ODE {
   static final double g = 9.8;
   double[] state = new double[5]; // {x,vx,y,vy,t}
   int pixRadius = 6;              // pixel radius for drawing of projecti
   EulerRichardson odeSolver = new EulerRichardson(this);

   public void setStepSize(double dt) {
      odeSolver.setStepSize(dt);
   }

   public void step() {
      odeSolver.step(); // do one time step using selected algorithm
   }

   public void setState(double x, double vx, double y, double vy) {
      state[0] = x;
      state[1] = vx;
      state[2] = y;
      state[3] = vy;
      state[4] = 0;
   }

   public double[] getState() {
      return state;
   }
```

```
  public double[] getState() {
    return state;
  }

  public void getRate(double[] state, double[] rate) {
    rate[0] = state[1]; // rate of change of x
    rate[1] = 0;        // rate of change of vx
    rate[2] = state[3]; // rate of change of y
    rate[3] = -g;       // rate of change of vy
    rate[4] = 1;        // dt/dt = 1
  }

  public void draw(DrawingPanel drawingPanel, Graphics g) {
    int xpix = drawingPanel.xToPix(state[0]);
    int ypix = drawingPanel.yToPix(state[2]);
    g.setColor(Color.red);
    g.fillOval(xpix-pixRadius, ypix-pixRadius, 2*pixRadius, 2*pixRadius);
    g.setColor(Color.green);
    int xmin = drawingPanel.xToPix(-100);
    int xmax = drawingPanel.xToPix(100);
    int y0 = drawingPanel.yToPix(0);
    g.drawLine(xmin, y0, xmax, y0); // draw a line to represent the ground
  }
}
```

Exercício Jupyter – 1

Sistemas dinâmicos, implementação com vetor de estados e erro quadrático definido pela norma

ODE do movimento:

$$\frac{dy}{dt} = v(t)$$
$$\frac{dv}{dt} = a(t)$$

Euler:

$$y(t + \Delta t) = y(t) + \Delta t v(t)$$
$$v(t + \Delta t) = v(t) + \Delta t a(t)$$

Implementação com vetor de estados em uma modelagem por sistemas dinâmicos:

$$\vec{S}(t + \Delta t) = \vec{S}(t) + \Delta t \vec{R}(t)$$

em que $\vec{S}$ é o vetor de estados e $\vec{R}$ é o vetor de taxas de variação.

Exemplo

Abaixo está a solução para $\frac{d^2x}{dt^2} = 6t$ usando vetor de estados. Temos:

$\frac{d^2x}{dt^2} = 6t$

Exercício Jupyter – 2

Simple Harmonic Oscilator - SHO

$$\frac{d^2x}{dt^2} = -\omega^2 x$$

$$\omega = \sqrt{\frac{k}{m}}$$

ou

$$\frac{dv}{dt} = a(t) = -\omega^2 x(t)$$

$$\frac{dx}{dt} = v(t)$$

e a formulação de Euler:

$$x(t + \Delta t) = x(t) + \Delta t \ v(t)$$

$$v(t + \Delta t) = v(t) - \Delta t \ \omega^2 x$$

Podemos definir o vetor de estados e o vetor de taxas de variação:

$$\vec{S} = [\vec{S}[0], \vec{S}[1], \vec{S}[2]] = [x, v, t]$$

$$\vec{R} = [\vec{S}[1], -\omega^2 \vec{S}[0], 1]$$