Solução numérica de sistemas dinâmicos

I. F. F. dos Santos

Universidade Federal de Alagoas
Instituto de Física

Sumário

Sistemas dinâmicos

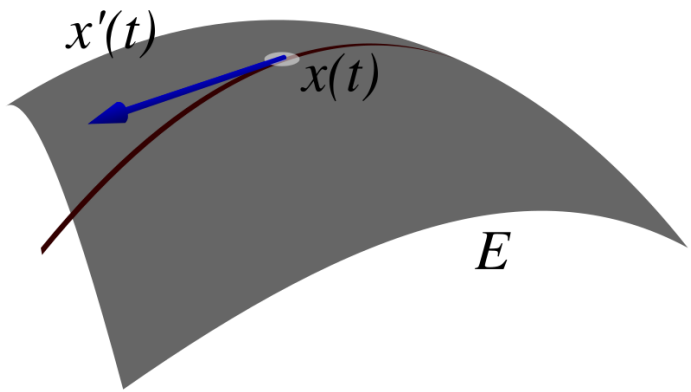
$\mathcal{E}$ Espaço d'estados: um conjunto onde cada ponto representa um possível estado do sistema físico, nesse caso um aberto de $\mathbb{R}^n$ ou $\mathbb{C}^n$.

$x(t)$ Evolução: uma função do tempo que diz qual será o estado *futuro*, após a evolução no tempo.

Em geral, uma lei de evolução é da forma

$$\frac{d}{dt}x(t) = F(t, x(t)), \qquad (1)$$

sujeito à condição de que $x(t_0) = x_0$.



Problema de valor inicial

Problema: Dada uma função $F : \mathbb{R} \times \mathcal{E} \to \mathcal{E}$ e um estado inicial $x_0 \in \mathcal{E}$ no instante inicial $t_0 \in \mathbb{R}$ encontre uma evolução $x : \mathbb{R} \to \mathcal{E}$ tal que a eq. 1 é satisfeita e $x(t_0) = x_0$.

O PVI pode ser reescrito na sua forma integral usando o TFC

$$x(t_0 + \delta t) = x(t_0) + \int_{t_0}^{t_0+\delta t} F(t, x(t))\, dt. \tag{2}$$

- Se F é contínua então o PVI possui solução.
- Se F é Lipschitz contínua no segundo argumento então o PVI possui uma única solução.

Representando em C

No $\mathbb{R}^n$ ou $\mathbb{C}^n$,

$$\begin{bmatrix} x_1(t_0+\delta t) \\ \vdots \\ x_n(t_0+\delta t) \end{bmatrix} = \begin{bmatrix} x_1(t_0) \\ \vdots \\ x_n(t_0) \end{bmatrix} + \int_{t_0}^{t_0+\delta t} \begin{bmatrix} f_1(t,x(t)) \\ \vdots \\ f_n(t,x(t)) \end{bmatrix} dt$$

```c
typedef struct estado_s {
    // _Complex
   double *x;
   int dim;
} estado_t;

static inline double dot_x(int i, double t, estado_t sistema){ /* ... */ }

/* Inicialize da seguinte maneira */
estado_t sistema;
sistema.dim = /* ... */ ;
sistema.x = (double*)malloc((size_t)(sistema.dim) * sizeof(double));
```

Dicas

- Simplifique seu problema, i.e., resolva outro equivalente porém mais simples.
- Sempre que possível use variáveis adimensionais (defina $u = v_0^{-1} v$).
- Escalas quânticas ou astronômicas DEVEM ser reescaladas.
- Identifique os pontos fixos ($x^* \in \mathcal{E}$ t.q. $F(t, x^*) = 0$) e outros casos de solução conhecida.
- Identifique constantes de evolução, funções $f : \mathcal{E} \to \mathbb{R}$ tais que $f(x(t_0 + \delta t)) = f(x_0)$, e as use para testar sua solução.

Método de Euler

Método:

$$x(t_0 + \delta t) \leftarrow x_0 + F(t_0, x_0)\, \delta t \tag{3}$$

Recursão:

$$x_0 \leftarrow x(t_0 + \delta t) \tag{4}$$

$$t_0 \leftarrow t_0 + \delta t \tag{5}$$

```
t = t0;
while(t <= tf){
   for(int i = 0; i < sistema.dim; ++i) k[i] = dot_x(i, t, sistema);
   for(int i = 0; i < sistema.dim; ++i) sistema.x[i] += k[i] * dt;
   t += dt;
}
```

Modelo SIR

- $\mathcal{E} \sim \mathbb{R}^3$.
- Lei de evolução:

$$\frac{d}{dt}\begin{bmatrix} S \\ I \\ R \end{bmatrix} = \begin{bmatrix} -\frac{\beta}{N} SI \\ \frac{\beta}{N} SI - \gamma I \\ \gamma I \end{bmatrix} \quad \sim \quad \frac{d}{d\tau}\begin{bmatrix} s \\ i \\ r \end{bmatrix} = \begin{bmatrix} -\alpha\ si \\ \alpha\ si - i \\ i \end{bmatrix} \tag{6}$$

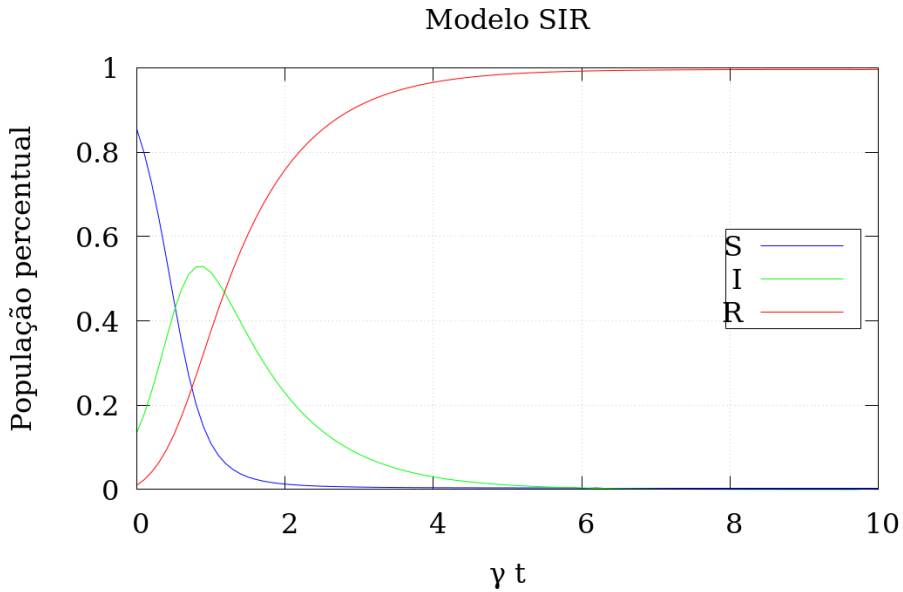
  onde $\alpha = {}^{\beta}/_{\gamma}$, $\tau = \gamma t$, $s = \frac{1}{N} S$, $i = \frac{1}{N} I$ e $r = \frac{1}{N} R$.
- $N = S + I + R$ é uma constante de evolução.

```c
typedef struct estado_s { double S, I, R; } estado_t;

static inline double dot_S(estado_t sistema){
   return (alpha * sistema.S * sistema.I);
}

static inline double dot_R(estado_t sistema){
   return (sistema.I);
}
static inline double N(estado_t sistema){
   return (sistema.S + sistema.I + sistema.R);
}
```

Modelo SIR




```
t = 0.0;
while(t <= tf){
   k_S = dot_S(sistema) * dt;
   k_I = dot_R(sistema) * dt;
   sistema.S -= k_S;
   sistema.I += k_S - k_I;
   sistema.R += k_I;
   t += dt;
   if(fabs(N(sistema) - 1.0) > 1.0e-8)
      fputs("A populacao nao estah conservando.\n", stderr);
}
```

Equações de Hamilton

- $\mathcal{E} \sim \mathbb{R}^{2f}$, $\mathcal{H} : \mathcal{E} \to \mathbb{R}$.
- Leis de evolução:

$$\frac{d}{dt}\begin{bmatrix} q(t) \\ p(t) \end{bmatrix} = \begin{bmatrix} v(t, q(t), p(t)) \\ f(t, q(t), p(t)) \end{bmatrix} = \begin{bmatrix} \frac{\partial}{\partial p}\mathcal{H} \\ -\frac{\partial}{\partial q}\mathcal{H} \end{bmatrix} \tag{7}$$

- $E = \mathcal{H}$ é uma constante de evolução.

```c
typedef struct estado_s {
   double *Q, *P;
   int dim;
} estado_t;

static inline double dot_Q(int i, estado_t sistema){ /* ... */ }
static inline double dot_P(int i, estado_t sistema){ /* ... */ }
```

Método de Euler semi-implícito

Derivada da forma $(v, f) = (v(t, q, p),\ f(t, q))$.

Método:

$$p(t_0 + \delta t) \leftarrow p_0 + f(t_0, q_0)\ \delta t \tag{8}$$

$$q(t_0 + \delta t) \leftarrow q_0 + v(t_0, q_0, p(t_0 + \delta t))\ \delta t \tag{9}$$

```
for(int i = 0; i < sistema.dim; ++i) k_P[i] = dot_P(i, sistema);
for(int i = 0; i < sistema.dim; ++i) sistema.P[i] += k_P[i] * dt;
for(int i = 0; i < sistema.dim; ++i) k_Q[i] = dot_Q(i, sistema);
for(int i = 0; i < sistema.dim; ++i) sistema.Q[i] += k_Q[i] * dt;
```

Método de Verlet

Derivada da forma $(\dot{q}, \dot{p}) = (v(t,p),\ f(t,q,p))$.

Método:

$$q(t_0 + \tfrac{1}{2}\delta t) \leftarrow q_0 + v(t_0, p_0)\ \tfrac{1}{2}\delta t \tag{10}$$

$$p(t_0 + \delta t) \leftarrow p_0 + f(t_0, q(t_0 + \tfrac{1}{2}\delta t), p_0)\ \delta t \tag{11}$$

$$q(t_0 + \delta t) \leftarrow q(t_0 + \tfrac{1}{2}\delta t) + v(t_0, p(t_0 + \delta t))\ \tfrac{1}{2}\delta t \tag{12}$$

```
dt2 = 0.5 * dt;
for(int i = 0; i < sistema.dim; ++i) k_Q[i] = dot_Q(i, sistema);
for(int i = 0; i < sistema.dim; ++i) sistema.Q[i] += k_Q[i] * dt2;
for(int i = 0; i < sistema.dim; ++i) k_P[i] = dot_P(i, sistema);
for(int i = 0; i < sistema.dim; ++i) sistema.P[i] += k_P[i] * dt;
for(int i = 0; i < sistema.dim; ++i) k_Q[i] = dot_Q(i, sistema);
for(int i = 0; i < sistema.dim; ++i) sistema.Q[i] += k_Q[i] * dt2;
```

Método de Verlet

```
/* OU ainda */
for(int i = 0; i < sistema.dim; ++i){
   k_Q[i] = dot_Q(i, sistema);
   sistema.Q[i] += k_Q[i] * dt2;
}
for(int i = 0; i < sistema.dim; ++i) k_P[i] = dot_P(i, sistema);
for(int i = 0; i < sistema.dim; ++i) sistema.P[i] += k_P[i] * dt;
for(int i = 0; i < sistema.dim; ++i){
   k_Q[i] = dot_Q(i, sistema);
   sistema.Q[i] += k_Q[i] * dt2;
}
```

Oscilador vertical

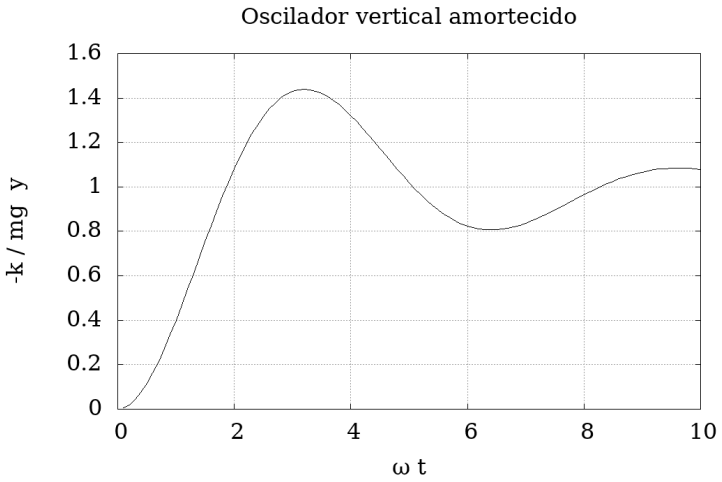
- $\mathcal{E} \sim \mathbb{R}^2$.
- Lei de evolução:

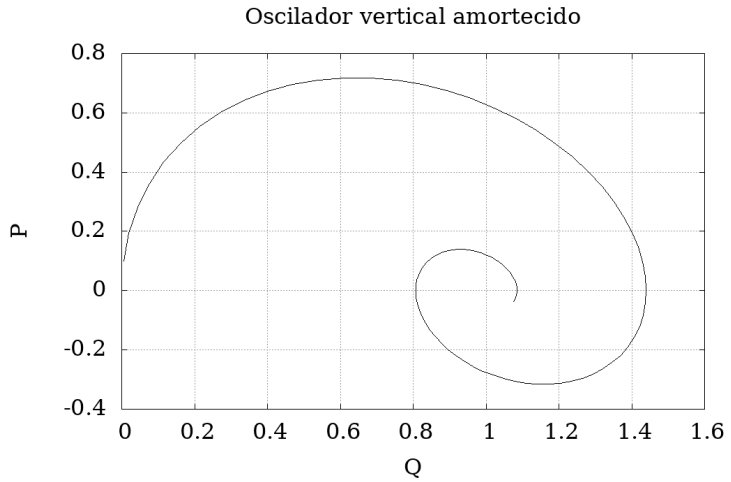
$$\frac{d}{dt}\begin{bmatrix} y \\ p \end{bmatrix} = \begin{bmatrix} \frac{1}{m}p \\ -ky - \frac{\beta}{m}p - mg \end{bmatrix} \quad \sim \quad \frac{d}{d\tau}\begin{bmatrix} Q \\ P \end{bmatrix} = \begin{bmatrix} P \\ -(Q + \gamma P) + 1 \end{bmatrix} \tag{13}$$

onde $\gamma = \frac{\beta}{m}\sqrt{\frac{m}{k}}$, $\tau = \sqrt{\frac{k}{m}}t$, $Q = -\frac{k}{mg}y$ e $P = -\frac{1}{mg}\sqrt{\frac{k}{m}}p$.

```
typedef struct estado_s { double Q, P; } estado_t ;
static inline double dot_Q(estado_t sistema){
   return (sistema.P);
}
static inline double dot_P(estado_t sistema){
   return (-(sistema.Q + gamma * sistema.P) + 1.0);
}
```

Oscilador vertical






```
/* Com Euler */
sistema.Q += dot_Q(sistema) * dt;
sistema.P += dot_P(sistema) * dt;

/* Com Verlet */
dt2 = 0.5 * dt;
sistema.Q += dot_Q(sistema) * dt2;
sistema.P += dot_P(sistema) * dt;
sistema.Q += dot_Q(sistema) * dt2;
```

Cadeia de massas idênticas com interação linear

- $\mathcal{E} \sim \mathbb{R}^{2N}$.
- Lei de evolução:

$$\frac{d}{dt}\begin{bmatrix} q_n \\ p_n \end{bmatrix} = \begin{bmatrix} \frac{1}{m} p_n \\ k(q_{n-1} + 2q_n + q_{n+1}) \end{bmatrix} \quad \sim \quad \frac{d}{d\tau}\begin{bmatrix} Q_n \\ P_n \end{bmatrix} = \begin{bmatrix} P_n \\ Q_{n-1} + 2Q_n + Q_{n+1} \end{bmatrix} \tag{14}$$

onde $\tau = \sqrt{\frac{k}{m}}t$, $Q_n = q_0^{-1} q_n$ e $P_n = q_0 \sqrt{mk}\, p_n$.

```
typedef struct estado_s { double *Q, *P; int N; } estado_t ;
static inline double dot_Q(int n, estado_t *sistema){
   return (sistema.P[n]);
}
static inline double dot_P(int n, estado_t sistema){
   return (sistema.Q[n-1] - 2.0 * sistema.Q[n] + sistema.Q[n+1]);
}
```

Cadeia de massas idênticas com interação linear

```
/* Com Verlet */
for(int n = 1; n <= sistema.dim; ++n){
   k_Q[n] = dot_Q(n, sistema);
   sistema.Q[n] += k_Q[n] * dt2;
}
for(int n = 1; n <= sistema.dim; ++n){
   k_P[n] = dot_P(n, sistema);
   sistema.P[n] += k_P[n] * dt;
}
for(int n = 1; n <= sistema.dim; ++n){
   k_Q[n] = dot_Q(n, sistema);
   sistema.Q[n] += k_Q[n] * dt2;
}
```

Mudança de variáveis

- $s : \mathbb{R} \to \mathbb{R} : t \mapsto \tau$
- $T : \mathcal{E} \to \mathcal{E} : x \mapsto y$
- $y(\tau) = Tx(s^{-1}(\tau))$
- $G(\tau, y(\tau)) = \frac{d}{d\tau} s^{-1}(\tau)\, TF(s^{-1}(\tau), T^{-1}(y(\tau)))$

$$\frac{d}{dt}x(t) = F(t, x(t)) \quad \sim \quad \frac{d}{d\tau}y(\tau) = G(\tau, y(\tau)) \tag{15}$$

Outros métodos

- Runge-Kutta
- Adams
- Integradores simpléticos