Solução numérica de sistemas dinâmicos

I. F. F. dos Santos

Universidade Federal de Alagoas
Instituto de Física

Sumário

Sistemas dinâmicos

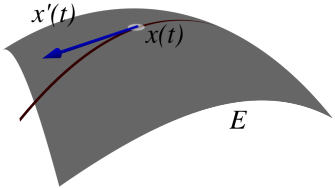
$\mathcal{E}$ Espaço d'estados: um conjunto onde cada ponto representa um possível estado do sistema físico, nesse caso um aberto de $\mathbb{R}^n$ ou $\mathbb{C}^n$.

$x(t)$ 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 $\mathbb{C}^n$,

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

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

static inline double edo_f(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){
      sistema.x[i] += f(i, t, &sistema) * dt;
   }
   t += dt;
}
```

Modelo SIR

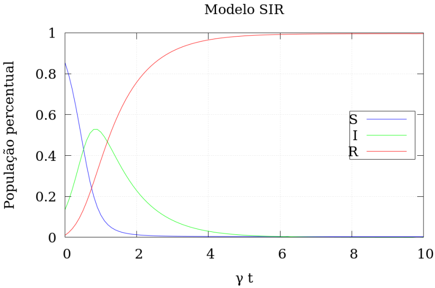
- $\mathcal{E} \sim \mathbb{R}^3$.
- Lei de evolução:

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

  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.

```
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;
}
```

Modelo SIR




```
while(t <= tf){
   aux = dot_S(&sistema) * dt;
   aux1 = dot_R(&sistema) * dt;
   sistema.S -= aux;
   sistema.I += aux - aux1;
   sistema.R += aux1;
   if(fabs(sistema.S + sistema.I + sistema.R - 1.0) > 1.0e-8)
      fputs("A populacao nao estah conservando.\n", stderr);
   t += dt;
}
```

Mudança de variáveis

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

$$\frac{d}{dt}x(t) = F(t, x(t)) \quad \sim \quad \frac{d}{d\tau}y(\tau) = G(\tau, y(\tau)) \tag{7}$$

Runge-Kutta de 4ª ordem

Método:

$$k_1 \leftarrow F(t_0, x(t_0)) \tag{8}$$

$$k_2 \leftarrow F(t_0 + {}^{\delta t}/_2,\ x(t_0) + k_1 {}^{\delta t}/_2) \tag{9}$$

$$k_3 \leftarrow F(t_0 + {}^{\delta t}/_2,\ x(t_0) + k_2 {}^{\delta t}/_2) \tag{10}$$

$$k_4 \leftarrow F(t_0 + \delta t,\ x(t_0) + k_3 \delta t) \tag{11}$$

$$x(t_0 + \delta t) \leftarrow x(t_0) + (k_1 + 2k_2 + 2k_3 + k_4)\ {}^{\delta t}/_6 \tag{12}$$

```
rk.k1[i] = f(i, t, &sistema);
rk.k2[i] = f(i, t + dt2, &sistema);
rk.k3[i] = f(i, t + dt2, &sistema);
rk.k4[i] = f(i, t + dt, &sistema);
sistema.x[i] += (rk.k1[i] + 2.0 * (rk.k2[i] + rk.k3[i]) + rk.k4[i]) * dt6;
```

Sistema de N níveis

Equações de Hamilton

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

$$\frac{d}{dt}\begin{bmatrix} q(t) \\ p(t) \end{bmatrix} = \begin{bmatrix} v(t, q(t), p(t)) \\ f(t, q(t), p(t)) \end{bmatrix} = \begin{bmatrix} \frac{\partial}{\partial p}\mathcal{H} \\ -\frac{\partial}{\partial q}\mathcal{H} \end{bmatrix} \tag{13}$$

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

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

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