Simulazione Software di un Sistema Dinamico

Corrado Santoro

ARSLAB - Autonomous and Robotic Systems Laboratory
Dipartimento di Matematica e Informatica - Università di Catania, Italy
santoro@dmi.unict.it

Programmazione Sistemi Robotici

Dal continuo al discreto

- L'implementazione/simulazione di un sistema di controllo può essere effettuata con un approccio
  - **tempo-continuo**
  - **tempo-discreto**
- **tempo-continuo:** l'implementazione si effettua risolvendo le equazioni differenziali che caratterizzano il sistema e determinando la forma analica $y(t)$
- **tempo-discreto:** l'implementazione si effettua **discretizzando** il sistema, ossia trasformandolo da tempo-continuo a **tempo-discreto**

Discretizzazione di un sistema

- L'operazione di discretizzazione implica "campionare" tutti i segnali con un intervallo di tempo prefissato Δt:

$$
\begin{aligned}
x(t), t \in \mathcal{R} &\Rightarrow x(k\Delta t), k \in \mathcal{N} \\
\dot{x}(t), t \in \mathcal{R} &\Rightarrow \frac{x((k+1)\Delta t) - x(k\Delta t)}{\Delta t}, k \in \mathcal{N} \\
x(t) &\Rightarrow x(k) \\
\dot{x}(t) &\Rightarrow \frac{x(k+1) - x(k)}{\Delta t}
\end{aligned}
$$

Discretizzazione di un sistema

Il sistema

$$\begin{cases} \dot{x} &= Ax + Bu \\ y &= Cx + Du \end{cases}$$

discretizzato con tempo di campionamento Δt diventa:

$$\begin{cases} \frac{x(k+1)-x(k)}{\Delta t} &= A\,x(k) + B\,u(k) \\ y(k) &= C\,x(k) + D\,u(k) \end{cases}$$

$$\begin{cases} x(k+1) - x(k) &= A\,x(k)\,\Delta t + B\,u(k)\,\Delta t \\ y(k) &= C\,x(k) + D\,u(k) \end{cases}$$

$$\begin{cases} x(k+1) &= (A\Delta t + I)\,x(k) + B\,u(k)\,\Delta t \\ y(k) &= C\,x(k) + D\,u(k) \end{cases}$$

Discretizzazione di un sistema

Il sistema

$$\begin{cases} \dot{x} &= Ax + Bu \\ y &= Cx + Du \end{cases}$$

discretizzato con tempo di campionamento Δt diventa:

$$\begin{cases} x(k+1) &= A'x(k) + B'u(k) \\ y(k) &= Cx(k) + Du(k) \end{cases}$$

dove:

$$\begin{aligned} A' &= A\Delta t + I \\ B' &= B\Delta t \end{aligned}$$

Sebbene la forma analitica sia diversa, i **due sistemi sono equivalenti**

In un sistema discreto:

$$\begin{cases} x(k+1) &= A'x(k) + B'u(k) \\ y(k) &= Cx(k) + Du(k) \end{cases}$$

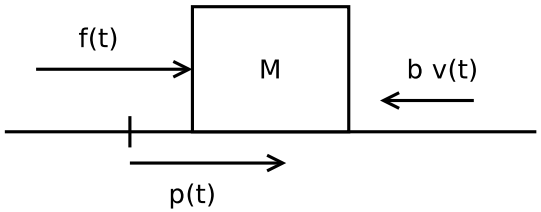
scompare l'equazione differenziale che descrive l'evoluzione dello stato nel *tempo*

Essa è sostituita dalla **equazione alle differenze**:

$$x(k+1) = A'x(k) + B'u(k)$$

che indica qual è il valore che assumerà lo stato al *successivo istante di campionamento* (legge di aggiornamento dello stato)

Esempio: Massa su piano con attrito



$$\begin{cases} \begin{bmatrix} \dot{v} \\ \dot{p} \end{bmatrix} = \begin{bmatrix} -\frac{b}{M} & 0 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} v \\ p \end{bmatrix} + \begin{bmatrix} \frac{1}{M} \\ 0 \end{bmatrix} \begin{bmatrix} f \end{bmatrix} \\ \begin{bmatrix} y \end{bmatrix} = \begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ p \end{bmatrix} \end{cases}$$

$$A = \begin{bmatrix} -\frac{b}{M} & 0 \\ 1 & 0 \end{bmatrix} \quad B = \begin{bmatrix} \frac{1}{M} \\ 0 \end{bmatrix}$$

$$C = \begin{bmatrix} 0 & 1 \end{bmatrix} \quad D = \begin{bmatrix} 0 \end{bmatrix}$$

Discretizziamo

$$
\begin{aligned}
A' &= A\Delta t + I = \begin{bmatrix} -\frac{b}{M} & 0 \\ 1 & 0 \end{bmatrix} \Delta t + \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} -\frac{b}{M}\Delta t + 1 & 0 \\ \Delta t & 1 \end{bmatrix} \\
B' &= B\Delta t = \begin{bmatrix} \frac{1}{M} \\ 0 \end{bmatrix} \Delta t = \begin{bmatrix} \frac{\Delta t}{M} \\ 0 \end{bmatrix}
\end{aligned}
$$

$$
\begin{cases}
\begin{bmatrix} v(k+1) \\ p(k+1) \end{bmatrix} = \begin{bmatrix} -\frac{b}{M}\Delta t + 1 & 0 \\ \Delta t & 1 \end{bmatrix} \begin{bmatrix} v(k) \\ p(k) \end{bmatrix} + \begin{bmatrix} \frac{\Delta t}{M} \\ 0 \end{bmatrix} \begin{bmatrix} f(k) \end{bmatrix} \\
\begin{bmatrix} y(k) \end{bmatrix} = \begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} v(k) \\ p(k) \end{bmatrix}
\end{cases}
$$

Discretizziamo

$$\begin{cases} \begin{bmatrix} v(k+1) \\ p(k+1) \end{bmatrix} = \begin{bmatrix} -\frac{b}{M}\Delta t + 1 & 0 \\ \Delta t & 1 \end{bmatrix} \begin{bmatrix} v(k) \\ p(k) \end{bmatrix} + \begin{bmatrix} \frac{\Delta t}{M} \\ 0 \end{bmatrix} \begin{bmatrix} f(k) \end{bmatrix} \\ \begin{bmatrix} y(k) \end{bmatrix} = \begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} v(k) \\ p(k) \end{bmatrix} \end{cases}$$

$$\begin{cases} v(k+1) = (-\frac{b}{M}\Delta t + 1)v(k) + \frac{\Delta t}{M} f(k) \\ p(k+1) = v(k)\Delta t + p(k) \\ y(k) = p(k) \end{cases}$$

Implementazione di un **Sistema Dinamico**

Un **sistema dinamico** è una **black box** caratterizzata da:

- uno **stato** (**variabili persistenti**)
- un **comportamento** (**codice**)

La sua implementazione più naturale è un **oggetto** in cui:

- gli **attributi** rappresentano lo **stato**
- i **metodi** rappresentano il **comportamento**

Implementazione del sistema massa con attrito

Generalizziamo.... possiamo pensare ad una **classe base** dove rappresentiamo:

- il tempo di campionamento, **`m_delta_t`**
- il metodo (virtuale ed astratto), **`evaluate`**, che rappresenta il comportamento del sistema

```
class DynamicSystem {

public:
  DynamicSystem(float delta_t) { m_delta_t = delta_t; };
  virtual float evaluate(float input) = 0;

protected:
  float m_delta_t;
};
```

Implementazione del sistema massa con attrito

$$\begin{cases} v(k+1) &= (-\frac{b}{M}\Delta t + 1)v(k) + \frac{\Delta t}{M} f(k) \\ p(k+1) &= v(k)\Delta t + p(k) \\ y(k) &= p(k) \end{cases}$$

Specializziamo....

```
class MassaAttrito : public DynamicSystem {

public:
  MassaAttrito(float m, float b, float delta_t);
  float evaluate(float input);

private:
  float m_massa, m_b;   // costanti del sistema
  float v_k, p_k;       // stato del sistema
};
```

Implementazione del sistema massa con attrito

$$\begin{cases} v(k+1) &= (-\frac{b}{M}\Delta t + 1)v(k) + \frac{\Delta t}{M} f(k) \\ p(k+1) &= v(k)\Delta t + p(k) \\ y(k) &= p(k) \end{cases}$$

Specializziamo....

```
MassaAttrito::MassaAttrito(float m, float b, float delta_t)
  : DynamicSystem(delta_t)
{
  m_massa = m;  m_b = b;  // impostazione attributi
  v_k = 0;  p_k = 0;      // stato iniziale
}

float MassaAttrito::evaluate(float input)
{
  float y = p_k;                          // calcolo uscita
  v_k = (-m_b / m_massa * m_delta_t + 1) * v_k
        + m_delta_t / m_massa * input;  // aggiornamento v_k
  p_k = v_k * m_delta_t + p_k;          // aggiornamento p_k
  return y;
}
```

Implementazione di un **Sistema Dinamico**

Organizzazione dei sorgenti:

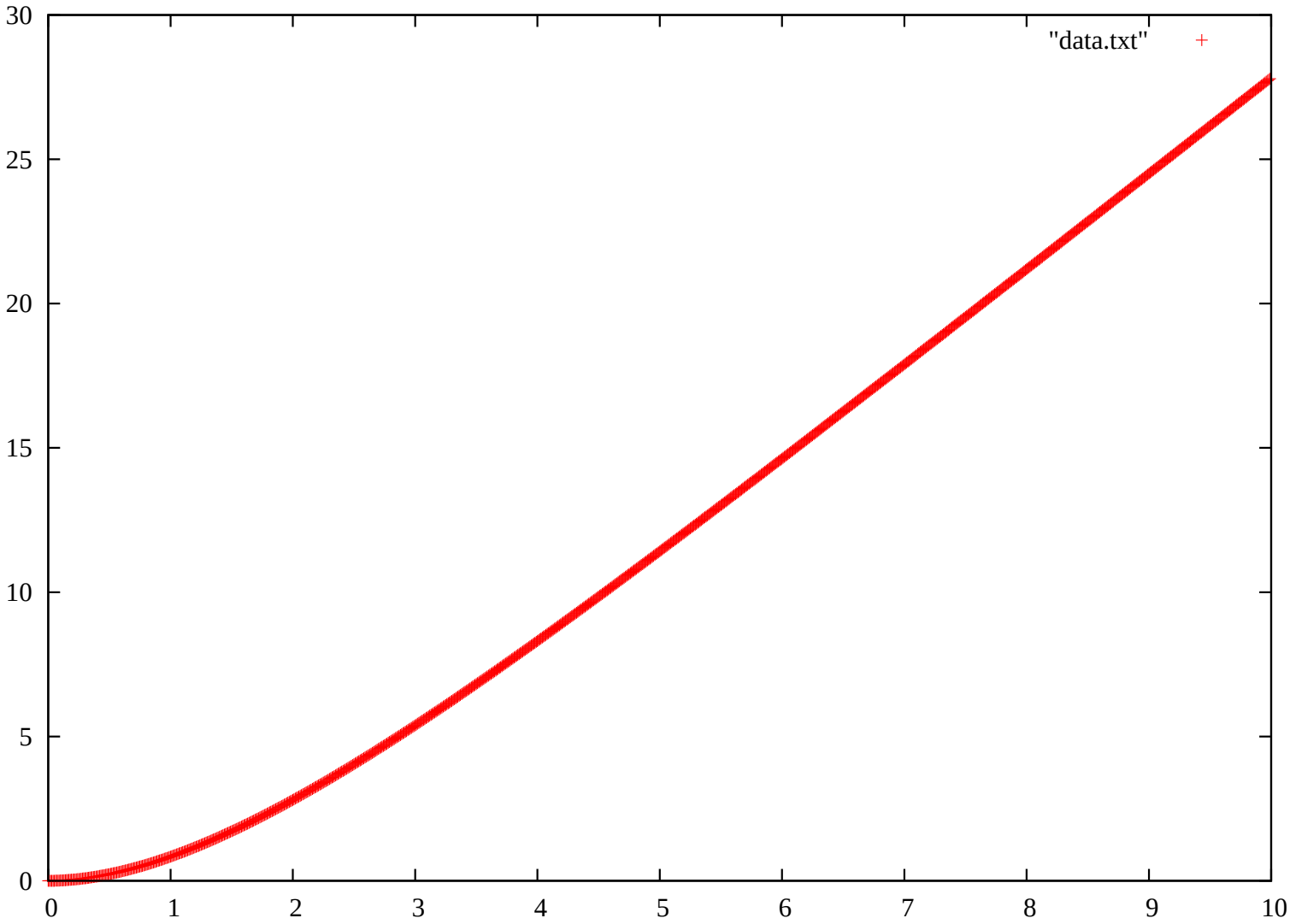
- `dynamic_system.h`: definizione della classe **DynamicSystem**
- `massa_attrito.h`: definizione della classe **MassaAttrito**
- `massa_attrito.cpp`: implementazione dei metodi della classe **MassaAttrito**
- `main.cpp`: test del sistema

Implementazione del sistema massa con attrito

main program....

```
#include <fstream>
#include "dynamic_system.h"
#include "massa_molla.h"

int main(int argc, char **argv)
{
  MassaAttrito m(1.0, 0.6, 0.01);
  // Massa = 1 kg
  // Coeff.Attrito = 0.6 N s/m
  // Delta T = 10 ms

  float input = 2; // spinta di 2 N

  std::ofstream output_file("data.txt"); // output file dei dati

  for (int i = 0; i < 1000;i++) {   // 10 secs = 1000 iterazioni
    float output = m.evaluate(input);
    output_file << (i*0.01) << " " << output << std::endl;
  }

  output_file.close();
}
```

Output della simulazione: Posizione



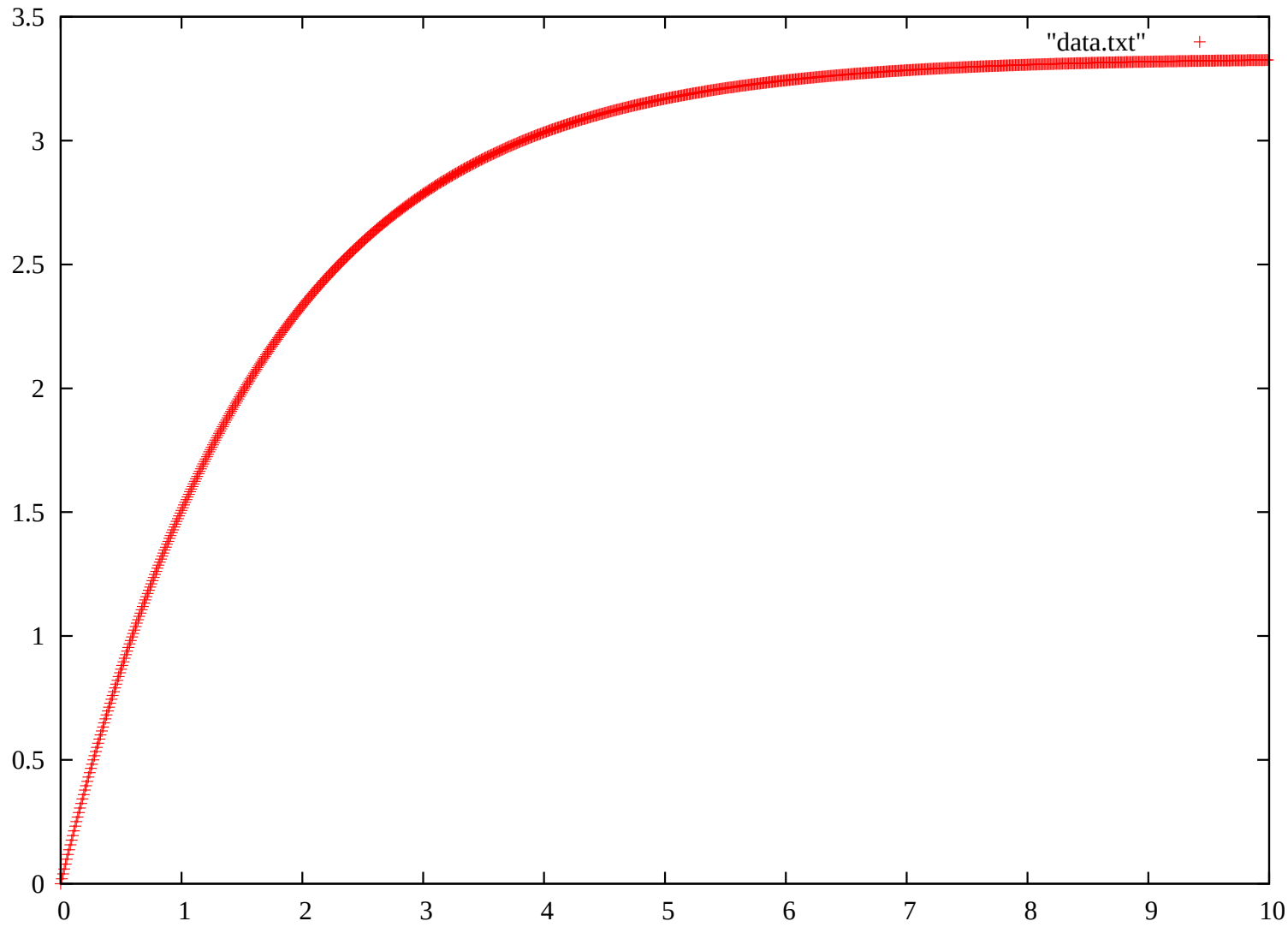
Implementazione del sistema massa con attrito

$$\begin{cases} v(k+1) &= (-\frac{b}{M}\Delta t + 1)v(k) + \frac{\Delta t}{M} f(k) \\ p(k+1) &= v(k)\Delta t + p(k) \\ y(k) &= v(k) \end{cases}$$

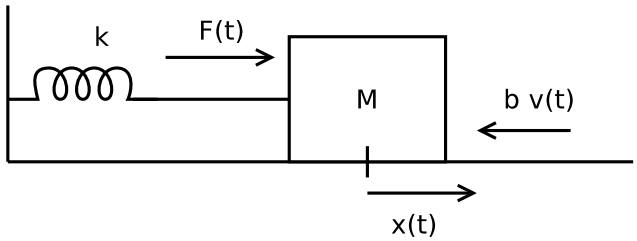
Consideriamo come output la velocità....

```
MassaAttrito::MassaAttrito(float m, float b, float delta_t)
  : DynamicSystem(delta_t)
{
  m_massa = m;  m_b = b;  // impostazione attributi
  v_k = 0;  p_k = 0;      // stato iniziale
}

float MassaAttrito::evaluate(float input)
{
  float y = v_k;                          // calcolo uscita
  v_k = (-m_b / m_massa * m_delta_t + 1) * v_k
        + m_delta_t / m_massa * input;  // aggiornamento v_k
  p_k = v_k * m_delta_t + p_k;          // aggiornamento p_k
  return y;
}
```

Output della simulazione: Velocità



Esercitazione: Massa-molla



$$\begin{cases} \begin{bmatrix} \dot{x} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\frac{k}{M} & -\frac{b}{M} \end{bmatrix} \begin{bmatrix} x \\ v \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{M} \end{bmatrix} \begin{bmatrix} F \end{bmatrix} \\ \begin{bmatrix} y \end{bmatrix} = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ v \end{bmatrix} \end{cases}$$

$$A = \begin{bmatrix} 0 & 1 \\ -\frac{k}{M} & -\frac{b}{M} \end{bmatrix} \quad B = \begin{bmatrix} 0 \\ \frac{1}{M} \end{bmatrix}$$

$$C = \begin{bmatrix} 1 & 0 \end{bmatrix} \quad D = \begin{bmatrix} 0 \end{bmatrix}$$

Simulazione Software di un Sistema Dinamico

Corrado Santoro

ARSLAB - Autonomous and Robotic Systems Laboratory
Dipartimento di Matematica e Informatica - Università di Catania, Italy
santoro@dmi.unict.it

Programmazione Sistemi Robotici