

PROGETTO PROGRAMMAZIONE SISTEMI ROBOTICI LAUREA IN INFORMATICA MAGISTRALE

Michele Mazzamuto W82000176
Santino Isgrò 1000000617





Sviluppo di una **web app** capace di simulare **sistemi** e **controllori**.

System UI offre all'utente la possibilità di analizzare il comportamento di vari sistemi, senza doversi curare della loro implementazione.



Scelta tra i **sistemi** disponibili

Per prima cosa l'utente deve scegliere quale tipologia di sistema simulare. Una volta scelto, sulla sinistra potrà settare le variabili di controllo per ciascun sistema ($y'' = 0/1$), durata e input della simulazione

 Systems 

Input/Output

Open available systems

G1:

1

$y''=2$

y'_2

y_2

u

G2:

1

$y''=2$

y'_2

y_2

u

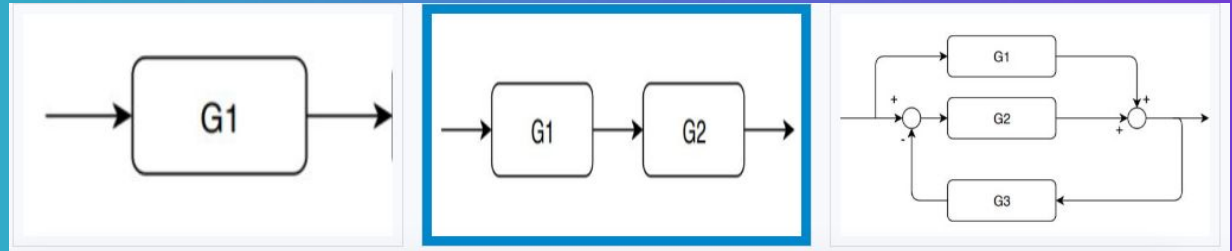
Time

10

Input

1

Ogni sistema è implementato come un modulo a se stante, il che permette di implementare nuove tipologie di sistemi in caso di sviluppi futuri



Scelta dei **controllori**

Per l'utente sarà inoltre possibile, opzionalmente, scegliere di aggiungere un controllore. Le opzioni attualmente disponibili sono : PI, PI con saturazione, PID e PID con saturazione

Controllers

Use controllers ? ☒

Type of controllers

PID

Ref.

2

if(t < 5) return 2

if(t >= 5) return t*3

Sarà inoltre possibile settare una propria funzione di riferimento, scegliendo fino a 4 condizioni differenti (if elif) come mostrato nella figura a sinistra.

Analisi degli **Output**

Dopo il tempo necessario per la simulazione, il tool mostrerà un report nel quale è possibile apprezzare sulla sinistra il grafico del sistema complessivo, mentre sulla destra una sintesi dettagliata di ogni sistema comprensiva di autovalori e stabilità

Dashboard

 Generate Report

Plot



System information

N. of system: 3

G1: $1y'' = -4y' - 2y + 3u$

Autovalori: $-2 + \sqrt{2}$, $-2 - \sqrt{2}$

Stability: Asintoticamente stabile

G2: $1y'' = -5y' - 2y + 6u$

Autovalori:

$-5/2 - \sqrt{17}/2$, $-5/2 + \sqrt{17}/2$

Stability: Asintoticamente stabile

G3: $1y'' = -2y' - 6y + 3u$

Autovalori: $-1 + \sqrt{5}i$, $-1 - \sqrt{5}i$

Stability: Asintoticamente stabile

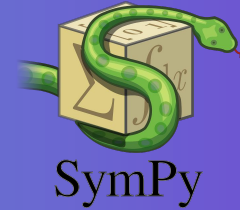
Tramite questo
tasto è possibile
salvare il grafico
in formato .png

Tecnologie Utilizzate

Front-End



Back-End



Esempi di **codice**

Di seguito alcuni estratti di codice riguardanti le funzioni principali da noi sviluppate:

```
def autovalori(y,y1):  
    A = Matrix([[0,1],[y,y1]])  
    lamb = symbols('lamb')  
    lamb_I = Matrix([[lamb,0],[0,lamb]])  
    aut = lamb_I - A  
    #print(aut)  
    determinante = det(aut)  
    return solveset(Eq(determinante,0))
```

```
class secondDegree:  
    def __init__(self,A,B):  
        self.x1 = 0  
        self.x2 = 0  
        self.A = A  
        self.B = B  
  
    def evaluate(self, u, delta_t1):  
        expr =self.A[1]  
        delta_t_val = expr.evalf(subs = {self.A[1]: delta_t1})  
        temp_x1 = self.x1 * self.A[0] + delta_t_val * self.x2  
  
        A1_10 = self.A[2]  
        A1_10 = A1_10.evalf(subs = {self.A[1]: delta_t1})  
  
        A1_11 = self.A[3]  
        A1_11 = A1_11.evalf(subs = {self.A[1]: delta_t1})  
  
        B1_01 = self.B[1]  
        B1_01 = B1_01.evalf(subs = {self.A[1]: delta_t1})  
  
        op1 = A1_10 * self.x1  
        op2 = A1_11 * self.x2  
        op3 = u * B1_01  
        temp_x2 = op1 + op2 + op3  
        self.x1 = temp_x1  
        self.x2 = temp_x2  
        output = self.x1  
        return output
```

Esempi di codice 2

```
def choose_system(equations):
    stabilita1 = ""
    #print(equations)
    if (equations['y2'] == "0"):
        A= float(equations['y'])
        B = float(equations['u'])
        G1 = firstDegree(A,B)
        auto_val = stability1(A)
        if auto_val < 0:
            stabilita1 = "Asintoticamente stabile"
        elif auto_val > 0:
            stabilita1 = "Instabile"
        else:
            stabilita1 = "Stabile semplicemente"
        print("identificato primo grado")
    else:
        A,B = sistema(equations['y2'],equations['y1'],equations['y'],equations['u'])
        G1 = secondDegree(A,B)

        print("identificato secondo grado")
        auto_val_1_1,auto_val_1_2 = autovalori([equations['y'],equations['y1']])
        auto_val = str(auto_val_1_1) + ", " + str(auto_val_1_2)
        auto_val_1_1,auto_val_1_2 = stability(auto_val_1_1,auto_val_1_2,equations['y1'],equations['y'])

        if float(auto_val_1_1) < 0 and float(auto_val_1_2) < 0:
            stabilita1 = "Asintoticamente stabile"
        elif float(auto_val_1_1) > 0 or float(auto_val_1_2) > 0:
            stabilita1 = "Instabile"
        else:
            stabilita1 = "Stabile semplicemente"
    return G1, auto_val, stabilita1, A,B
```

choose_system

rappresenta il metodo principale di System UI. Tale metodo ci permette di distinguere se un sistema possiede una derivata di primo o di secondo grado, e di conseguenza richiamare i metodi per il calcolo degli autovalori e della stabilità.

Conclusioni

Pur ottenendo dei risultati soddisfacenti l'implementazione del tool mostra alcuni limiti riguardanti principalmente la scalabilità dei sistemi. Poiché ogni tipologia di sistema complessivo necessita di un modulo a parte, sviluppato a priori.

Stesso discorso vale per la gestione delle reference, al momento l'utente può inserire nel sistema solamente 4 condizioni che non supportano l'utilizzo di operatori logici.

Possibili **sviluppi futuri** sono sicuramente l'implementazione di più moduli per sistemi complessivi e un refactoring della classe Reference in modo da gestire più tipologie di condizioni, supportando anche operatori logici e matematici.