spr

February 1, 2021

1 Sprawozdanie 6

1.1 Jan Bronicki, Denis Firat, Borys Staszczak

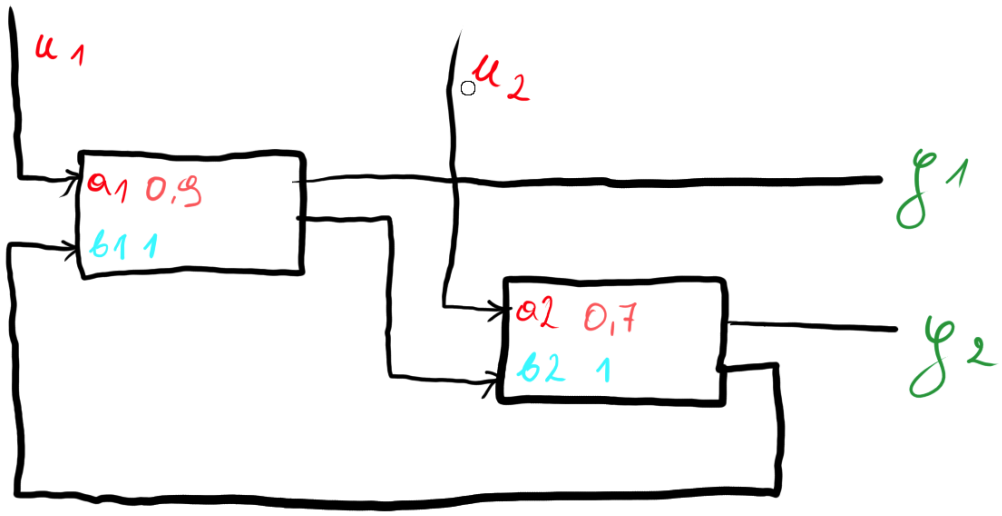
1.2 1. Cel ćwiczenia

Zoptymalizować działanie dowolnego 2-elementowego systemu kaskadowego, przyjmując kryterium kosztów

1.3 2. Optymalizację działania systemu bez ograniczeń na zasoby metodami: globalną, lokalną.

$$Q(u_1, u_2) = (y_1 - 1)^2 + (y_2 - 2)^2$$

System:



Z powyższego schematu uzyskujemy następujące macierze:

$$A = \begin{bmatrix} 0.9 & 0 \\ 0 & 0.7 \end{bmatrix}, \quad B = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad H = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

1.3.1 2.1 Metoda lokalna

Obliczamy u_i:

$$u_i = \frac{y_{z,i} - a_i \cdot H_i \cdot y_z}{b_i}$$

Podstawiamy:

$$u_1 = \frac{1 - 0.9 \cdot 2}{1} = -0.8 u_2 = \frac{2 - 0.7}{1} = 1.3$$

Otrzymujemy:

$$u = \begin{bmatrix} -0.8 \\ 1.3 \end{bmatrix}$$

1.3.2 2.2 Metoda globalna

Ponownie wyznaczamy u_1 i u_2:

$$K = (I - AB)^{-1} B u_i = K^{-1} y$$

Uzyskujemy:

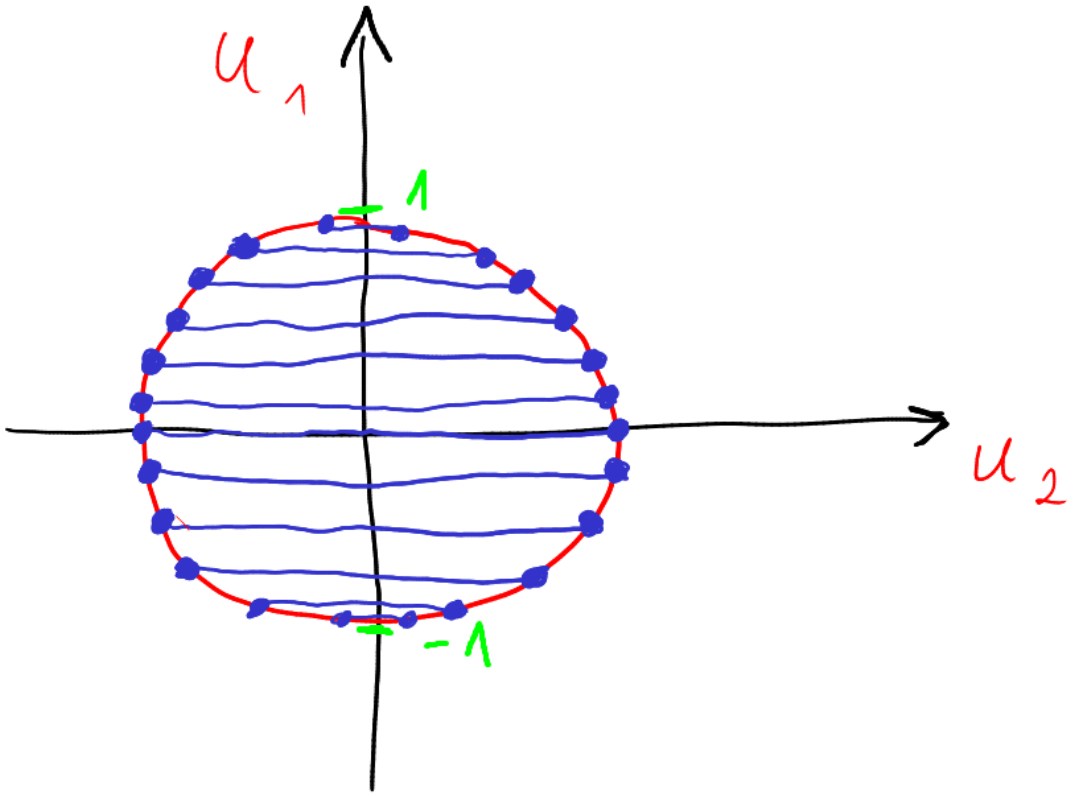
$$u = \begin{bmatrix} -0.8 \\ 1.3 \end{bmatrix}$$

1.4 3. Optymalizację działania systemu z ograniczeniem zasobów:

$$u_1^2 + u_2^2 \leq 1$$

używając metody 2-poziomowej z wybranym typem koordynacji

Dla każdego u_1 znajdujemy przedział u_2. Następnie, dla każdej możliwości u_1 i u_2 obliczamy Q i patrzymy gdzie jest najmniejsze. Na końcu printujemy najmniejsze Q, u_1 oraz u_2



[1]:
```python
# Importujemy potrzebne biblioteki
from sympy import *
import numpy as np

def kryterium_Q(u1q: float, u2q: float) -> float:
    """Kryterium_Q $Q(u_{1}, u_{2}) = (y_{1}-1)^{2} + (y_{2}-2)^{2}$

    Parameters
    ----------
    u1q : float
        u1, dla którego liczymy Q
    u2q : float
        u2, dla którego liczymy Q

    Returns
    -------
    float
        Zwraca Q, zakładając, że będzie float'em
    """
    return (u1q - 1) ** 2 + (u2q - 2) ** 2
```

[2]:
```python
# Step komputacji
step = 0.01
```

```
# Promień koła
radius_Q = 1
# Nasze u1 od -1 do 1, jako array NumPy
u1 = np.arange(-1 + step, 1, step)
# u2 jako symbol SymPy, będziemy na bierząco symbolicznie
# obliczać jego przedziały
u2 = Symbol("u2")

u1_od_u2 = np.array([])
```

[3]:
```
# Główna pętla komputacji
for each_u1 in u1:
    # Rozwiązujemy równanie żeby dostać przedział u2, dla danego u1
    fKar = Poly(each_u1 ** 2 + u2 ** 2 - radius_Q, u2, domain="RR")
    sol = solve_poly_inequality(fKar, "<=")
    u_start = sol[0].start.evalf()
    u_end = sol[0].end.evalf()
    # Tworzymy przedział u2, dla danego u1
    xd = np.append(u1_od_u2, np.arange(u_start, u_end, 0.01))

    # Obliczamy Q, dla każdego u1 i u2 z danego przedziału i znajdujemy␣
 ↪najmniejsze
    for each_u2 in np.arange(u_start, u_end, step):
        if each_u2 == u_start and each_u1 == -1 + step:
            q_min = kryterium_Q(each_u1, each_u2)
        elif q_min > kryterium_Q(each_u1, each_u2):
            q_min = kryterium_Q(each_u1, each_u2)
            u1_min = each_u1
            u2_min = each_u2
```

[4]:
```
print("Qmin   = "+str(q_min))
print("u1_min = "+str(u1_min))
print("u2_min = "+str(u2_min))
```

```
Qmin   = 1.54009766451582
u1_min = 0.5100000000000013
u2_min = 0.859825599078887
```