

3) Nel 1° quadrante del piano cartesiano  $X$  e  $Y$  è dato il problema di programmazione lineare

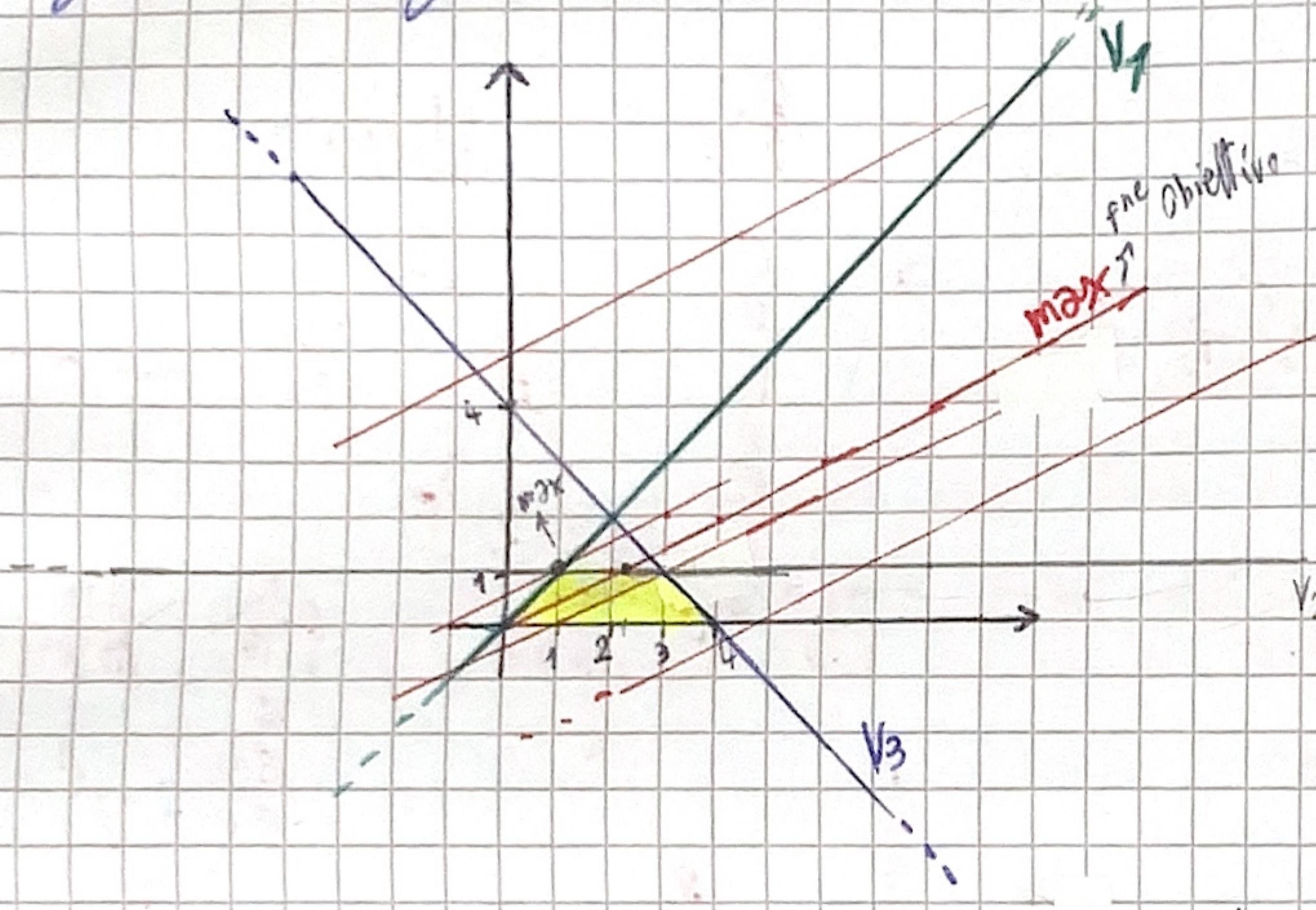
$$\max 2y - x$$

$$\text{con } 2y - x \leq 0$$

$$v_2: y - 1 \leq 0$$

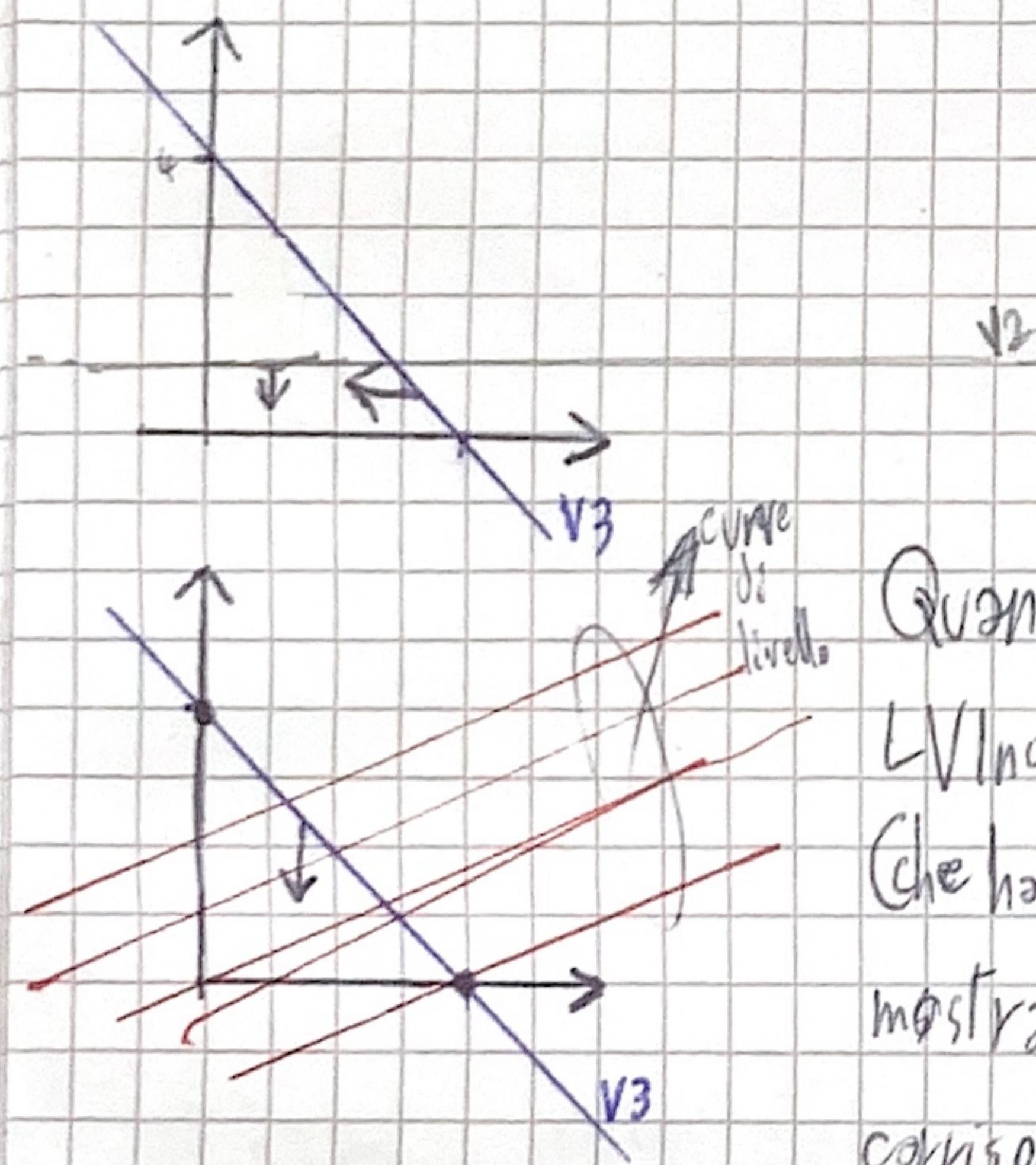
$$v_3: y + x - 4 \leq 0$$

Disegna la regione ammissibile e il fascio improprio --



Come visto a lezione e nelle note, il punto massimo si ottiene cercando la curva di livello che tocca la regione ammissibile nel vertice in alto a sinistra del trapezio (in questo caso, il punto più lontano e superiore all'interno di  $LVIncrementalLP(v_1, v_2, v_3)$  è stato campionato  $v_1$ ).

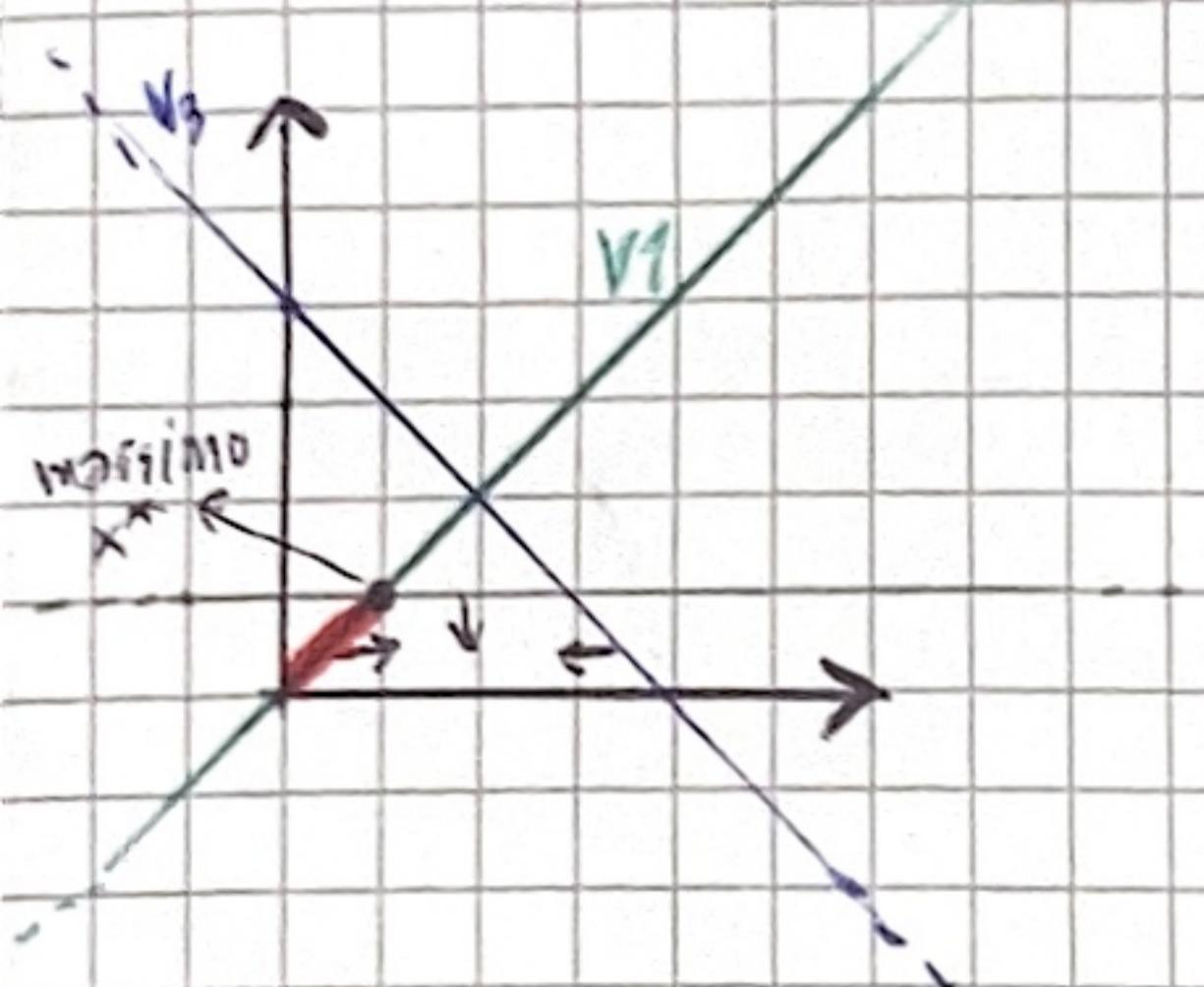
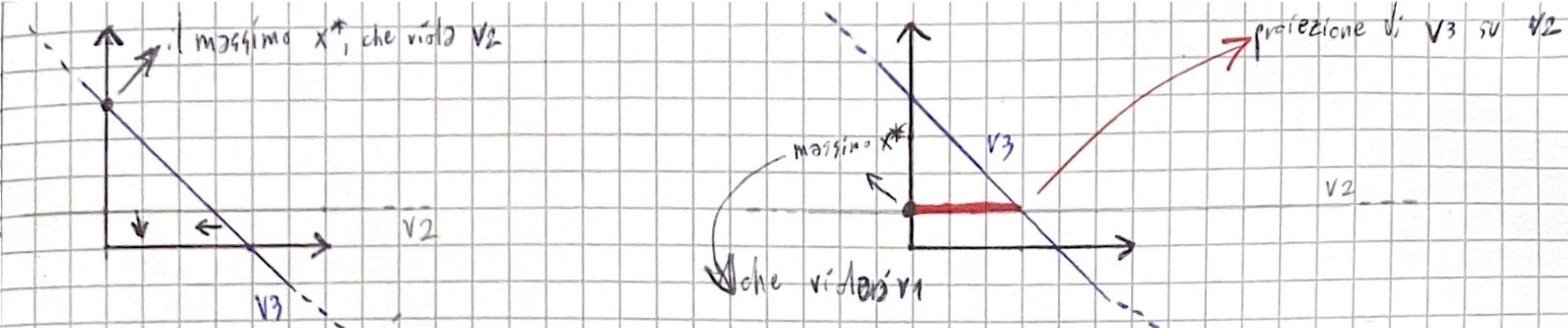
Qui di fianco sono mostrati i 2 vincoli dopo che dopo che all'interno di  $LVIncrementalLP(v_1, v_2, v_3)$  è stato campionato  $v_1$ .



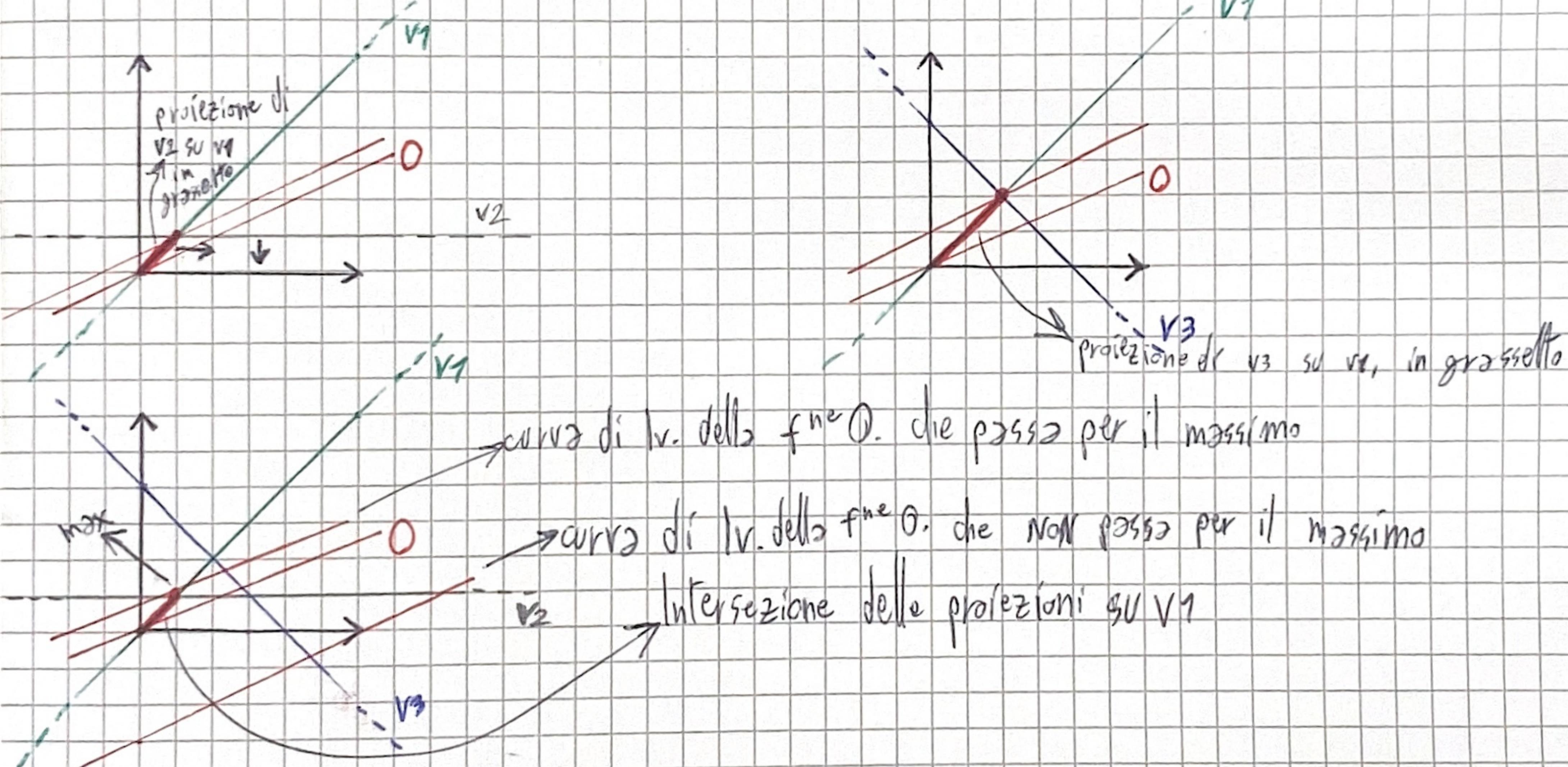
Quando all'interno di  $LVIncrementalLP(v_2, v_3)$  (che ha ricevuto il controllo da  $LVIncrementalLP(v_1, v_2, v_3)$ ) è campionato il vincolo  $v_2$ ,  $LVIncrementalLP(v_3)$  (che ha ricevuto il controllo da  $LVIncrementalLP(v_2, v_3)$ ) restituisce il massimo  $x^*(0)$  mostrato qui di fianco. In questo caso esiste un  $x^*$  ed una curva di livello corrispondente (a differenza dei casi su aula web e visti a lezione) perché, essendo i limiti al 1° quadrante, ci sono altri 2 vincoli inneschiati all'esercizio e di conseguenza (per la natura e la direzione di  $v_3$ ) le curve di livello sono limitate.

Nel disegni sottostanti, il grafico in alto a sinistra mostra cosa succede quando il controllo torna da  $LVIncrementalLP(v_3)$  (iniziano i ritorni alle funzioni chiudenti) a  $LVIncrementalLP(v_2, v_3)$ . Il massimo trovato viola il vincolo  $v_2$ . La proiezione del vincolo  $v_3$  sul vincolo  $v_2$  è mostrata nelle figure sottostanti (in alto a destra) in rosso, ma come segmento più spesso. Il nuovo massimo  $x^*$  è ottenuto confrontando i valori delle curve di livello che passano per i due estremi del segmento. Il controllo passa a  $LVIncrementalLP(v_1, v_2, v_3)$  e il massimo viola il vincolo  $v_1$ ; il nuovo massimo viene calcolato confrontando i valori delle curve di livello che passano per i due estremi del nuovo segmento in rosso, più spesso (ottenuto proiettando i vincoli  $v_2$  e  $v_3$  su  $v_1$ ).

La figura a sinistra riprende il problema (non uguale) visto a lezione e nelle note. I vincoli sono in verde, grigio, e blu ( $v_1, v_2, v_3$ ) e le frecce nere indicano il semipiano ammissibile per ogni vincolo (il 1° quadrante del piano cartesiano). La regione ammissibile è evidenziata in giallo. Le curve di livello della fnc obiettivo ( $\max$ ) sono in rosso.



Infine, nei grafici ancora sottostanti vengono mostrate le proiezioni di  $v_2$  e  $v_3$  su  $v_1$  e la loro intersezione.



In "allegato", lascio sotto gli stessi grafici disegnati con un python notebook (per completezza) e con l'aiuto di una libreria (pulp).

## APA a.a. 22/23 - Programmazione Lineare

Lontani dal caso peggiore: Programmazione Lineare (Compito 3.1 delle note)

```
%pip install pulp
%pip install numpy
%pip install matplotlib

import pulp

my_lp_problem = pulp.LpProblem("My LP Problem", pulp.LpMinimize)

# Variables
x = pulp.LpVariable('x', lowBound=0, cat='Continuous')
y = pulp.LpVariable('y', lowBound=0, cat='Continuous')

# Objective function
my_lp_problem += - y + 1/2*x, "Z"

# Constraints
my_lp_problem += y - x <= 0
my_lp_problem += y - 1 <= 0
my_lp_problem += y + x - 4 <= 0

# Solve
my_lp_problem.solve()
print("Status:", pulp.LpStatus[my_lp_problem.status])

# Print the solution
for variable in my_lp_problem.variables():
    print("{} = {}".format(variable.name, variable.varValue))

print("Z = {}".format(pulp.value(my_lp_problem.objective)))

import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline

# Definizione del primo Quadrante
x = np.linspace(0, 10, 100)
y = np.linspace(0, 10, 100)

# Definizione dei vincoli

# Vincolo v1 y -x <= 0
y1 = x
# Vincolo v2 y -1 <= 0
y2 = 1
```

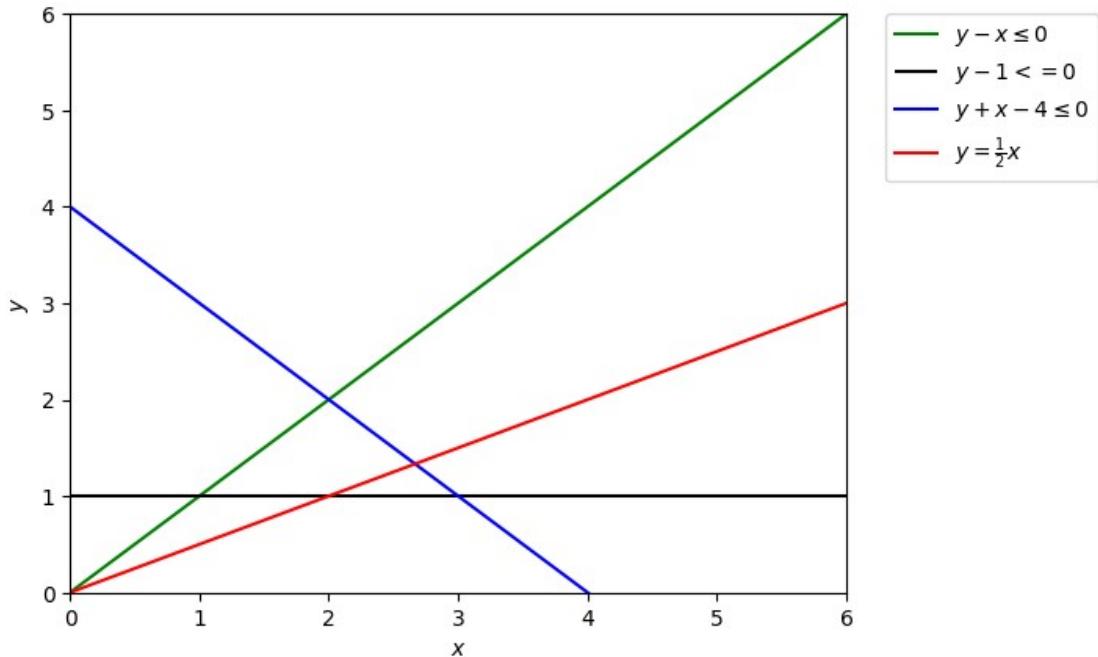
```

# Vincolo v3 y + x - 4 <= 0
y3 = -x + 4

# Definizione della funzione obiettivo
y4 = 1/2*x

# Plotting
plt.plot(x, y1, label=r'$y - x \leq 0$', color='g')
plt.hlines(y=1, xmin=0, xmax=100, label=r'$y - 1 \leq 0$', color='k')
plt.plot(x, y3, label=r'$y + x - 4 \leq 0$', color='b')
plt.plot(x, y4, label=r'$y = \frac{1}{2}x$', color='r')
plt.xlim(0, 6)
plt.ylim(0, 6)
plt.xlabel(r'$x$')
plt.ylabel(r'$y$')
plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.)
plt.show()

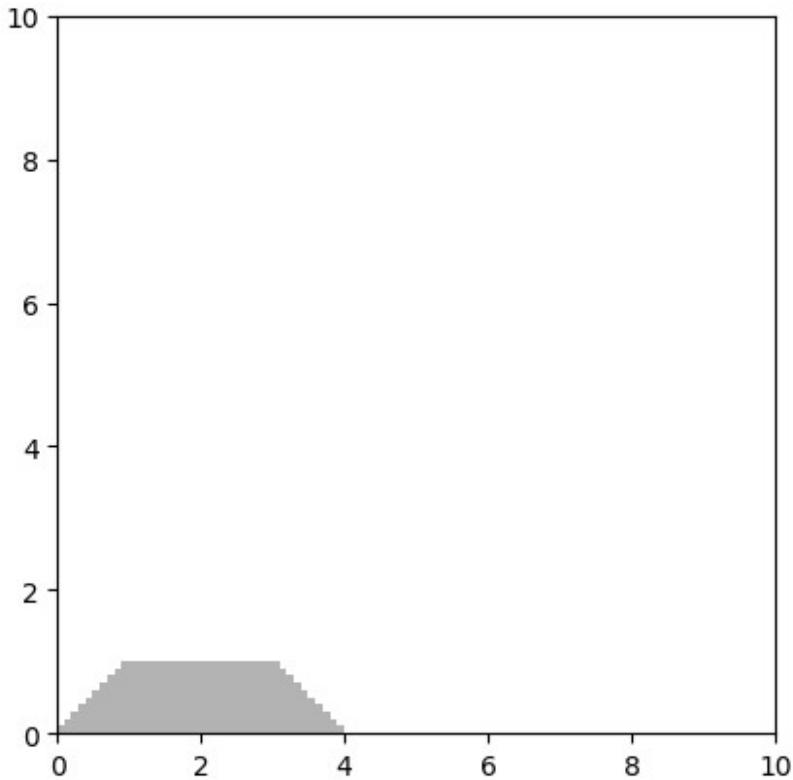
```



```

# Calcolo della regione ammissibile
X, Y = np.meshgrid(x, y)
Z = (Y>=0) & (Y - X<= 0) & (Y - 1<=0) & (Y + X - 4<=0)
plt.imshow(Z, extent=(x.min(),x.max(),y.min(),y.max())),
origin="lower", cmap="Greys", alpha = 0.3)
plt.show()

```



```
y = np.linspace(0, 10, 100)
x = np.linspace(0, 10, 100)
X, Y = np.meshgrid(x, y)
plt.imshow((Y>=0) & (Y - X<= 0) & (Y - 1<=0) & (Y + X - 4<=0).astype(int),
           extent=(x.min(),x.max(),y.min(),y.max()),origin="lower", cmap="Greys",
           alpha = 0.3)
plt.plot(x, y1, label=r'$y - x \leq 0$', color='g')
plt.hlines(y=1, xmin=0, xmax=100, label=r'$y - 1 \leq 0$', color='k')
plt.plot(x, y3, label=r'$y + x - 4 \leq 0$', color='b')
plt.plot(x, y4, label=r'$y = \frac{1}{2}x$', color='r')
plt.xlim(0, 6)
plt.ylim(0, 6)
plt.xlabel(r'$x$')
plt.ylabel(r'$y$')
plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.)
plt.show()
```

