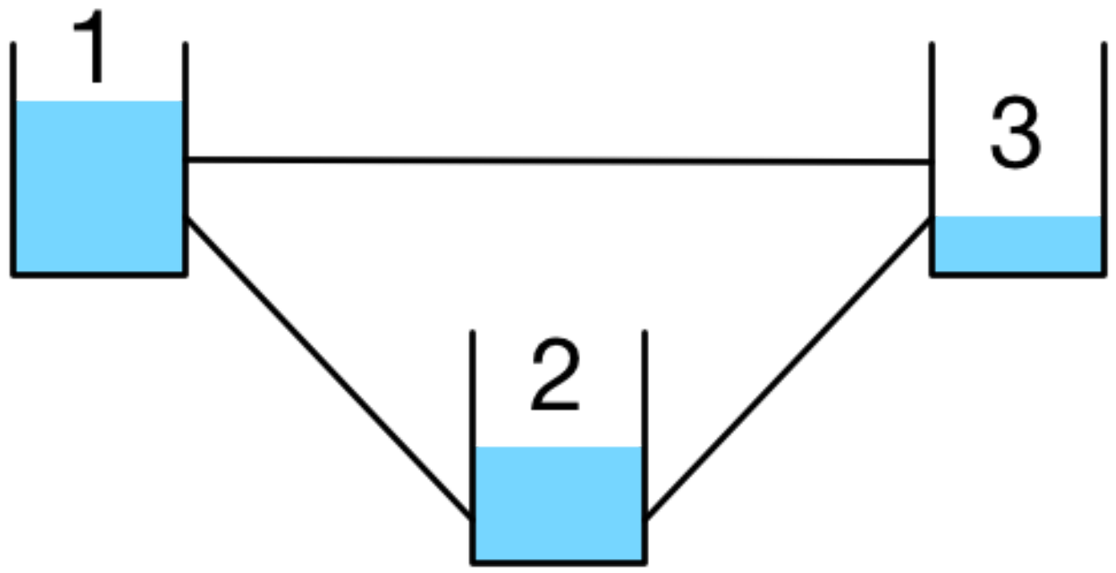
In [1]:
```
%load_ext autoreload
%autoreload 2
```

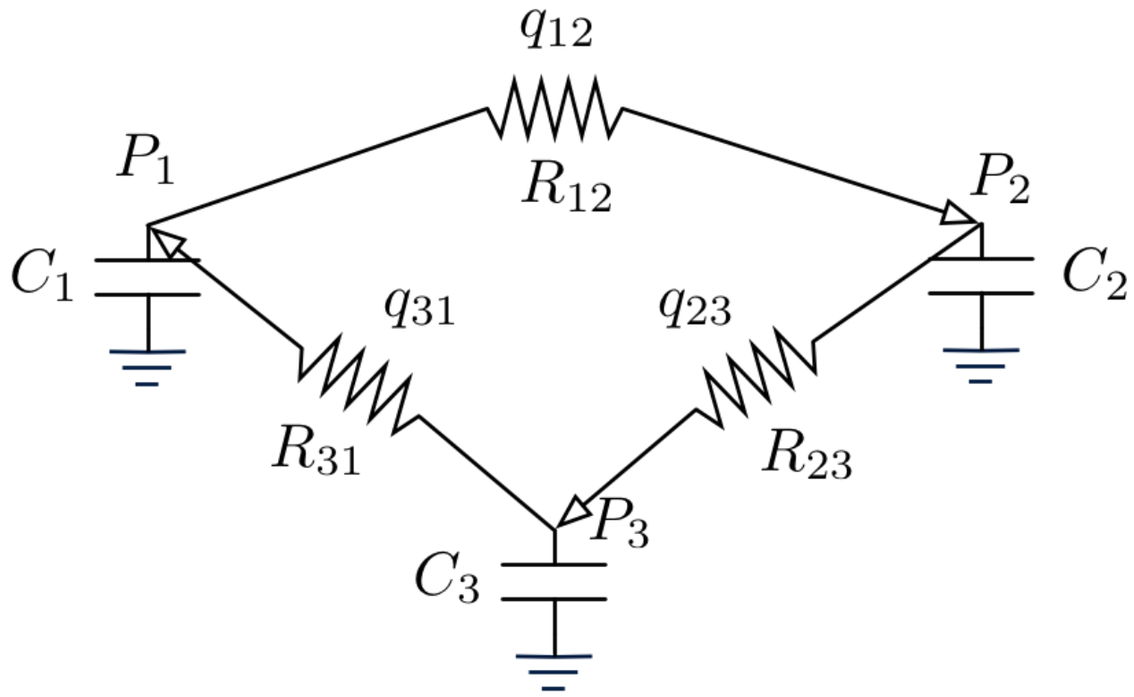
Esercizio: Serbatoi Comunicanti

Tre serbatoi comunicano attraverso condotte



Esercizio: Serbatoi Comunicanti

Tre serbatoi comunicano attraverso condotte



- Il problema può essere modellato con un circuito RC
- La tensioni corrispondono a valori di pressione (i.e. P_1, P_2, P_3)
- Le correndi a flussi di acqua (i.e. q_{12}, q_{23}, q_{31})
- Il modello è approssimativo, ma a noi basterà

Esercizio: Serbatoi Comunicanti

Il sistema è descritto dall'ODE:

$$\begin{pmatrix} \dot{P}_1 \\ \dot{P}_2 \\ \dot{P}_3 \end{pmatrix} = \begin{pmatrix} \frac{1}{C_1}(q_{31} - q_{12}) \\ \frac{1}{C_2}(q_{12} - q_{23}) \\ \frac{1}{C_3}(q_{23} - q_{31}) \end{pmatrix}$$

Con:

$$q_{12} = \frac{1}{R_{12}}(P_1 - P_2) \qquad q_{23} = \frac{1}{R_{23}}(P_2 - P_3)$$

$$q_{31} = \frac{1}{R_{31}}(P_3 - P_1)$$

Esercizio: Serbatoi Comunicanti

Prima di tutto, procediamo a caricare i dati del problema

Potete farlo usando la cella seguente:

In [2]:
```
# Dati intermedi
g = 9.81
S1 = 1 # Superfici
S2 = 1
S3 = 1
h1 = 3 # Livelli dei serbatoi
h2 = 2
h3 = 1
qmax12 = 0.0002 # Portata per unita' di pressione
qmax23 = 0.0007
qmax31 = 0.0008

# Capacita'
C1 = S1/g
C2 = S2/g
C3 = S3/g

# Resistenze
R12 = 1/qmax12
R23 = 1/qmax23
R31 = 1/qmax31
```

Esercizio: Serbatoi Comunicanti

Nel modulo `sol.tubes` si definisca una classe:

```
class Dstate:
    def __init__(self, C1, C2, C3, R12, R23, R31):
        ...

    def __call__(self, X, t):
        ...
```

...Che rappresenti la funzione che definisce l'ODE

- Il metodo `__call__` deve calcolare le derivate
- ...E restiuirle sotto forma di `numpy.array`

Nella cella seguente:

- Si utilizzi la classe per calcolare il gradiente
- ...Per lo stato fornito nella cella ed il tempo $t_0 = 0$

Esercizio: Serbatoi Comunicanti

Nel modulo `sol.tubes` si definisca una funzione:

```
def simulate(f, X0, t)
```

...Che si simuli il comportamento dei serbatoi:

- La funzione deve restituire una tupla contenente (nell'ordine):
  - La matrice con gli stati visitati
  - Il vettore con i valori del tempo
- La funzione deve anche disegnare un grafico utilizzando

`base.util.plot_state_evolution`

Si utilizzi la funzione per determinare il comportamento dei serbatoi

- Per un periodo di 600 secondi
- ...A partire dallo stato iniziale indicato nella cella

```
In [4]: X0 = np.array([P1_0, P2_0, P3_0])
```

Esercizio: Serbatoi Comunicanti

Nel modulo `sol.tubes` si definisca una funzione:

```
def final_P1(X, t)
```

- Che restituisca il valore finale della pressione P_1

Si stampi a video il risultato

In [ ]:

Esercizio: Serbatoi Comunicanti

Nel modulo `sol.tubes` si definisca una funzione:

```
def times_P2_eq_P3(X, t)
```

- Che restituisca come array
- ...I valori di tempo per cui $|P_2 - P_3| < 1e-2$

Si stampi a video il risultato

In [ ]: