TD-moteur_asynchrone

June 13, 2022

$\Phi = B.N.S.\cos(\omega_s t - \omega t) = \Phi_M \cos(\omega_r t)$ avec $\Phi_M = B.N.S$

$e = -\frac{d\Phi}{dt} = \omega_r \Phi_M \cos\left(\omega_r t - \frac{\pi}{2}\right)$

On dessine le circuit équivalent et on applique la loi d'Ohm en RSF:

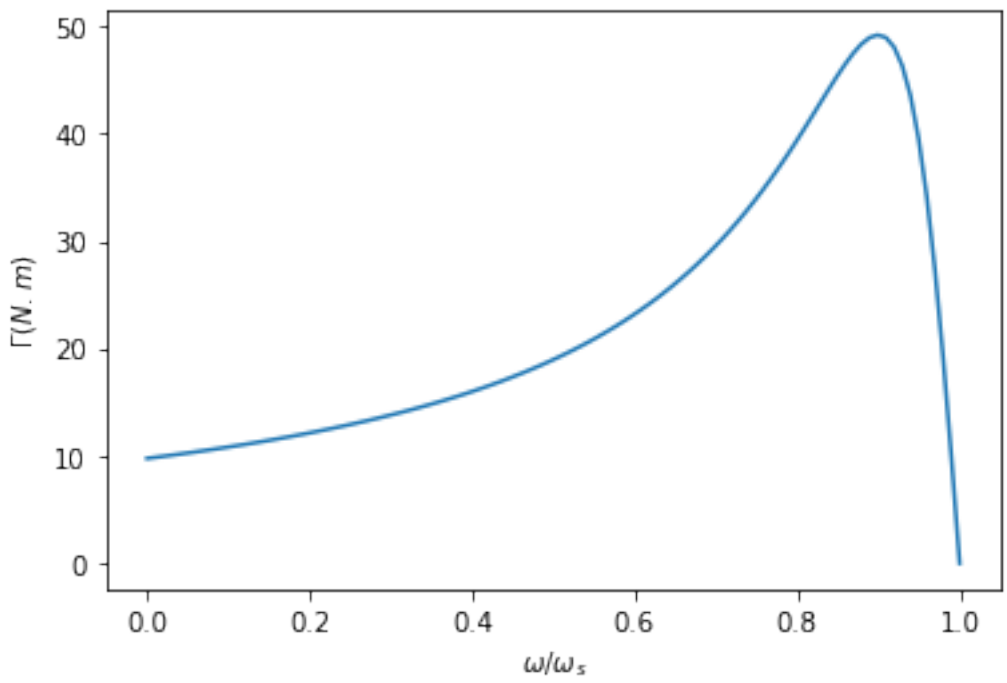
$$\underline{i} = \frac{\underline{e}}{R + jL\omega_r} = \frac{-j\omega_r \Phi_M e^{j\omega_r t}}{R + jL\omega_r}$$

On a alors $\vec{m} = i.N.S.\vec{n}$ et un couple $\gamma = \left(\vec{m} \wedge \vec{B}\right).\vec{e_z} = m.B.\sin(\theta - \phi) = i.B.N.S.\sin(\omega_r t)$ soit

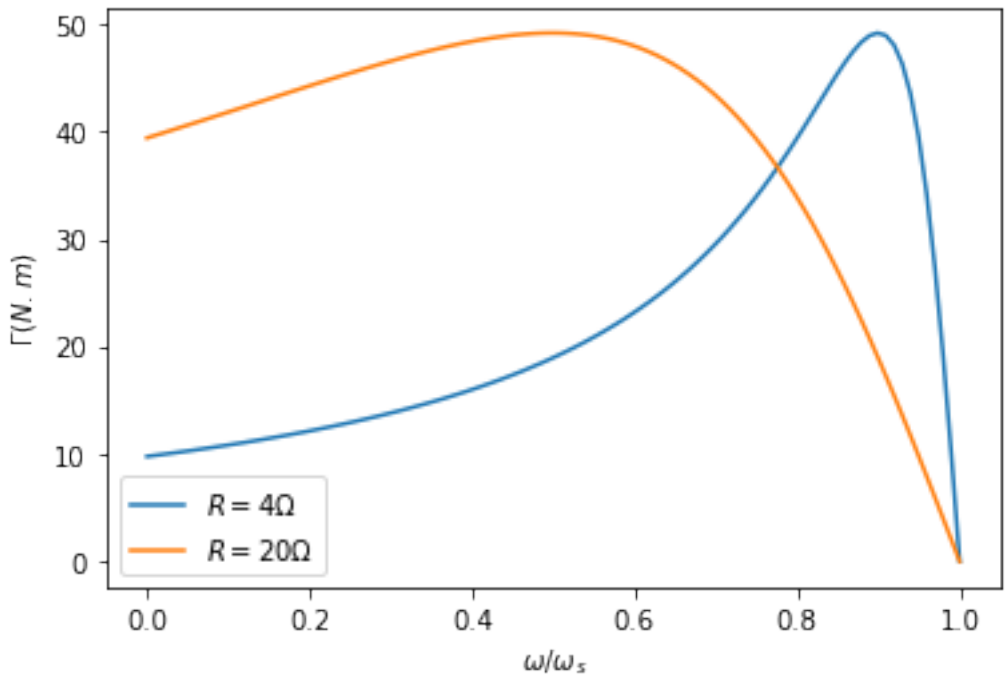
$$\gamma = i.\Phi_M.\cos\left(\omega_r t - \frac{\pi}{2}\right)$$

Le couple moyen vaut donc $\Gamma = \frac{1}{2}\Re\left[j\Phi_M \underline{i}(0)\right]$

[1]:
```
import numpy as np
import matplotlib.pyplot as plt
fs=50
ws=2*np.pi*fs
R,L,PhiM=4,127e-3,5
tab_w=np.linspace(0,ws,100)
tab_wr=ws-tab_w
tab_i=-1j*PhiM*tab_wr/(R+1j*L*tab_wr)
tab_Gamma=0.5*np.real(1j*PhiM*tab_i)

plt.figure()
plt.plot(tab_w/ws,tab_Gamma)
plt.xlabel(r"$\omega/\omega_s$")
plt.ylabel(r"$\Gamma(N.m)$")
plt.show()
```

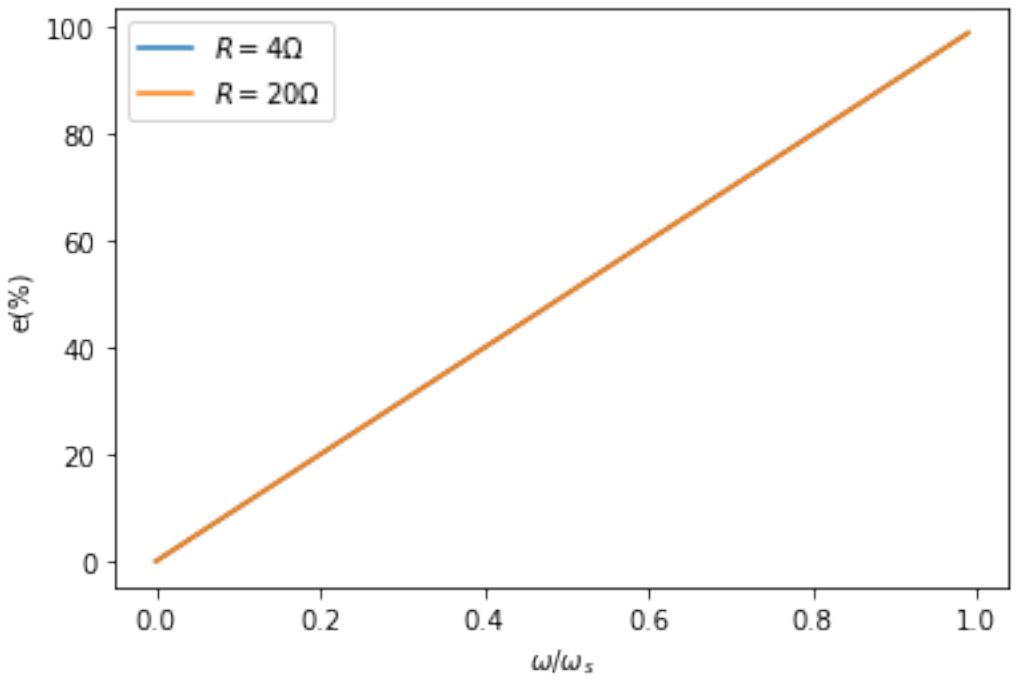


[2]:
```
R2=20
tab_i2=-1j*PhiM*tab_wr/(R2+1j*L*tab_wr)
tab_Gamma2=0.5*np.real(1j*PhiM*tab_i2)

plt.figure()
plt.plot(tab_w/ws,tab_Gamma,label=r"$R=4\Omega$")
plt.plot(tab_w/ws,tab_Gamma2,label=r"$R=20\Omega$")
plt.xlabel(r"$\omega/\omega_s$")
plt.ylabel(r"$\Gamma(N.m)$")
plt.legend()
plt.show()
```

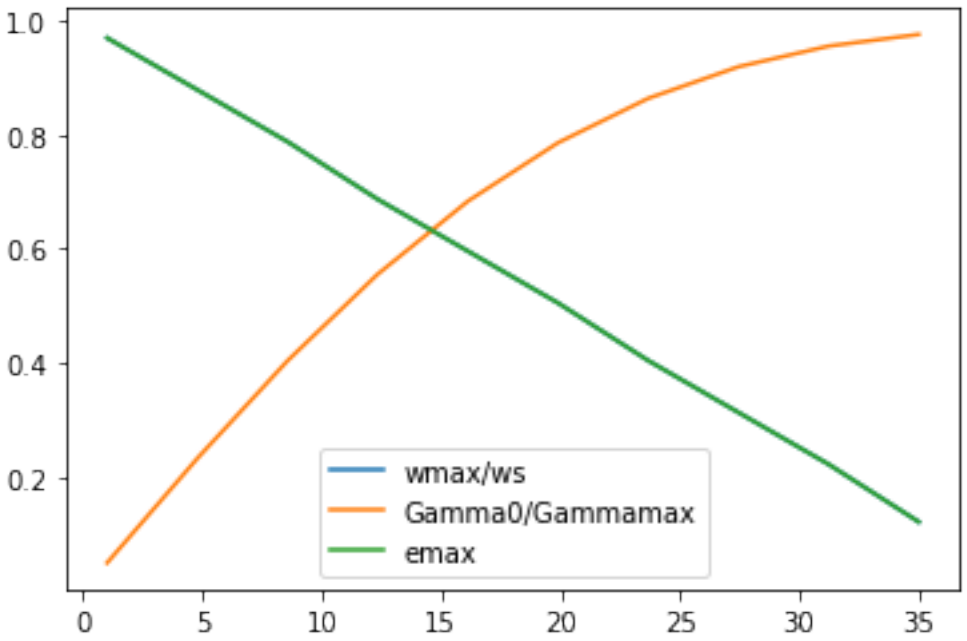


[3]:
```
tab_Pmeca=tab_w*tab_Gamma
tab_PJ=R/2*np.abs(tab_i)**2
tab_e=tab_Pmeca/(tab_Pmeca+tab_PJ)
tab_Pmeca2=tab_w*tab_Gamma2
tab_PJ2=R2/2*np.abs(tab_i2)**2
tab_e2=tab_Pmeca2/(tab_Pmeca2+tab_PJ2)

plt.figure()
plt.plot(tab_w/ws,tab_e*100,label=r"$R=4\Omega$")
plt.plot(tab_w/ws,tab_e2*100,label=r"$R=20\Omega$")
plt.xlabel(r"$\omega/\omega_s$")
plt.ylabel("e(%)")
plt.legend()
plt.show()
```

```
/tmp/ipykernel_5681/1436891269.py:3: RuntimeWarning: invalid value encountered
in true_divide
  tab_e=tab_Pmeca/(tab_Pmeca+tab_PJ)
/tmp/ipykernel_5681/1436891269.py:6: RuntimeWarning: invalid value encountered
in true_divide
  tab_e2=tab_Pmeca2/(tab_Pmeca2+tab_PJ2)
```



[4]:
```
tab_R=np.linspace(1,35,10)
tab_i0=-1j*PhiM*ws/(tab_R+1j*L*ws)
tab_Gamma0=0.5*np.real(1j*PhiM*tab_i0)
tab_wmax=np.zeros_like(tab_R)
tab_emax=np.zeros_like(tab_R)
for i in range(len(tab_R)):
    R=tab_R[i]
    tab_w=np.linspace(0,ws,100)
    tab_wr=ws-tab_w
    tab_i=-1j*PhiM*tab_wr/(R+1j*L*tab_wr)
    tab_Gamma=0.5*np.real(1j*PhiM*tab_i)
    iwmax=np.argmax(tab_Gamma)
    wmax=tab_w[iwmax]
    tab_wmax[i]=wmax
    Pmecamax=wmax*tab_Gamma[iwmax]
    PJmax=R/2*np.abs(tab_i[iwmax])**2
    emax=Pmecamax/(Pmecamax+PJmax)
    tab_emax[i]=emax
plt.figure()
plt.plot(tab_R,tab_wmax/ws,label="wmax/ws")
plt.plot(tab_R,tab_Gamma0/50,label="Gamma0/Gammamax")
plt.plot(tab_R,tab_emax,label="emax")
plt.legend()
plt.show()
```

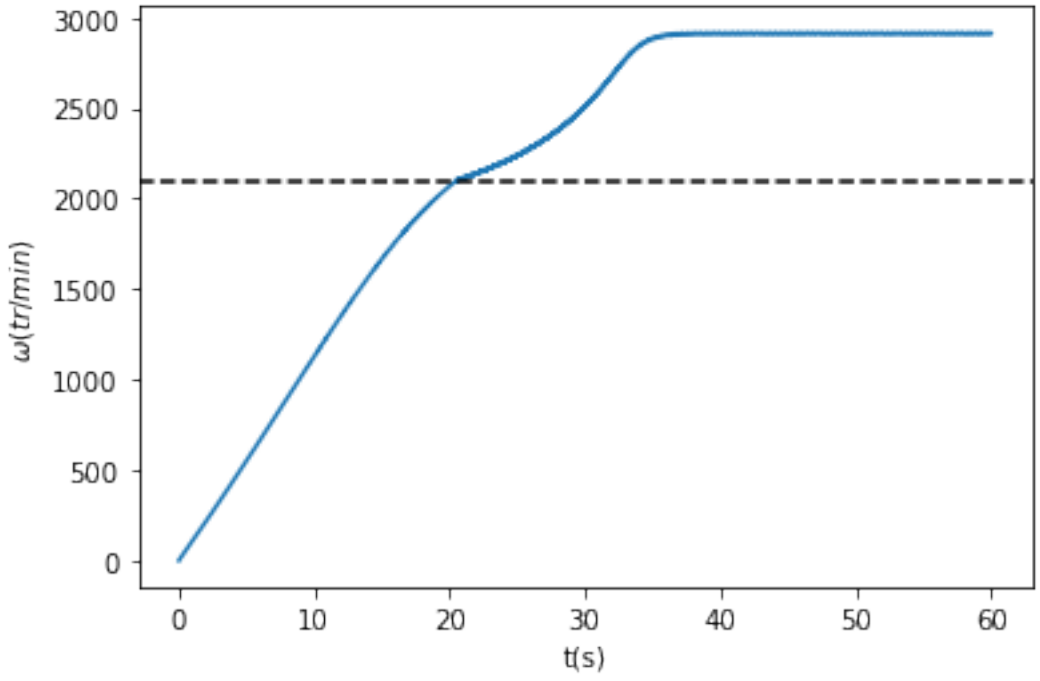


R forte: bon démarrage, maivais rendement

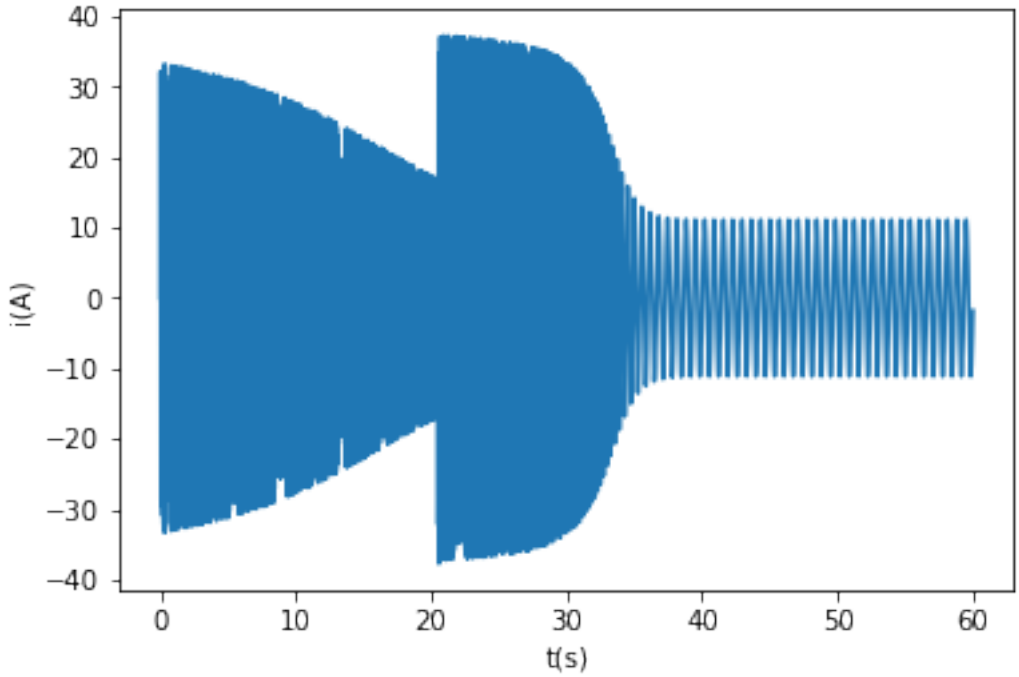
R faible: mauvais démarrage, bon rendement

[5]:
```
from scipy.integrate import odeint
ws=100*np.pi
L,PhiM=127e-3,5
def Cf(w):
    return -0.02*w-3e-5*w**2
Cu=30*9.8*0.12/2
J=2.4
def R(w):
    if w<ws*0.7:
        return 25
    else:
        return 4
def e(phi,dphidt,t):
    return PhiM*np.sin(ws*t-phi)*(ws-dphidt)
def gamma(phi,t,i):
    return i*PhiM*np.sin(ws*t-phi)
def derivee(inconnues,t):
    phi,dphidt,i=inconnues
    ddphidtt=(gamma(phi,t,i)+Cf(dphidt)-Cu)/J
    didt=(e(phi,dphidt,t)-R(dphidt)*i)/L
    return [dphidt,ddphidtt,didt]
ci=[0,0,0]
```

```
tab_t=np.linspace(0,60,6000)
sol=odeint(derivee,ci,tab_t)
tab_phi=sol[:,0]
tab_omega=sol[:,1]
tab_i=sol[:,2]

plt.figure()
plt.plot(tab_t,tab_omega*60/(2*np.pi))
plt.axhline(ws*60/(2*np.pi)*0.7,ls="--",color="black")
plt.xlabel("t(s)")
plt.ylabel(r"$\omega(tr/min)$")
plt.show()
```



[6]:
```
plt.figure()
plt.plot(tab_t,tab_i)
plt.xlabel("t(s)")
plt.ylabel("i(A)")
plt.show()
```


40
30
20
10
0
−10
−20
−30
−40
i(A)
0
10
20
30
40
50
60
t(s)

[ ]: