# resistance_model_test

October 18, 2024

## 1 KRISO KCS container ship - računska naloga

S pomočjo testa modela ladje v bazenu, določimo upor realne ladje.

[1]:
```
import math as mat
import numpy as np
import sympy as sym
import scipy.interpolate as interpolate

import matplotlib as mpl
import matplotlib.pyplot as plt

# set LaTeX fonts
mpl.rc('text', usetex = True)
mpl.rc('font', family = 'serif')
sym.init_printing()
```

[2]:
```
# Parametri modela

# Lpp - length between perpendiculars [m]:
Lpp_m = 7.28;  Lpp_s = 250.0;

# Lwl - length water line [m]:
Lwl_m = 7.42;  Lwl_s = 255.0;

# Los - length over surface [m]:
Los_m = 7.85;  Los_s = 262.0;

# B - breadth [m]:
B_m = 1.02;  B_s = 35.0;

# T - draught [m]:
Tf_m = 0.342;  Tf_s = 11.74;
Ta_m = 0.342;  Ta_s = 11.74;

# D - displacemnt [m3]:
D_m = 1.649;  D_s = 66822.61;
```

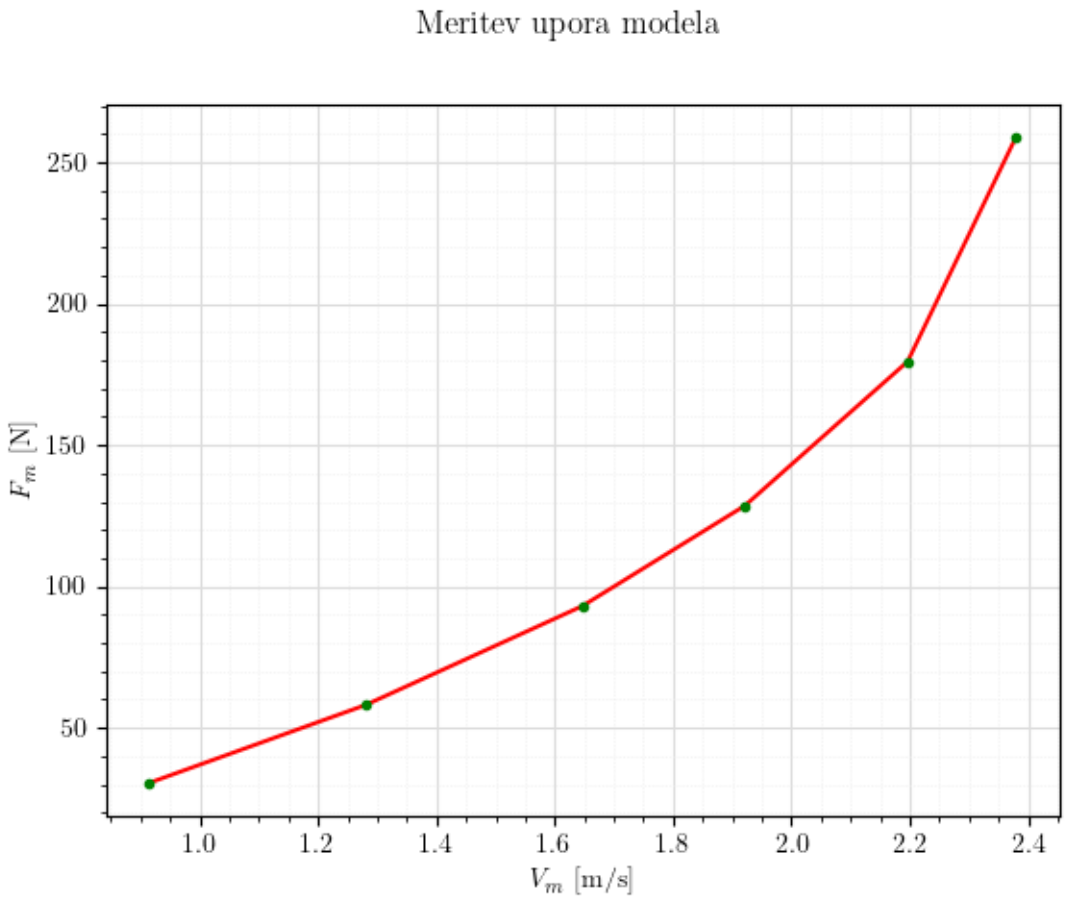
```
# S - wetted surface [m2]:
S_m = 9.95;  S_s = 11738.32;

# x_b Position centre of buoyancy forward of midship
xb = -0.571; # Lpp

# Cb - block coefficinet:
Cb = 0.6505;

# Ratios
# rlb =  5.582; # Lpp/B
# rbt =  2.673; # B/T
# rlt = 14.922; # L/T

# Caam - Air resistance:
Cair = 2.880E-05;

# Propeler diameter [m]
Dp = 8.0;

# Kinematic viscosity [Pa s]:
mu_m = 0.001;  mu_s = 0.001;

# Density [kg/m3]:
rho_m = 1000; rho_s = 1025;
```

### 1.1 Podatki meritev modela v bazenu

V bazenu merimo celoten upor modela $F_m$ [N] za različne hitrosti $V_m$ [m/s]:

[3]:
```
# Data [V_m, F_m]

data = np.array([
    [0.9151, 30.5507],
    [1.2810, 58.1317],
    [1.6469, 93.0954],
    [1.9215, 128.6038],
    [2.1962, 179.5267],
    [2.3795, 259.0008]
])

dDim = data.shape[0]

V_m = data.T[0];
F_m = data.T[1];

fig, ax = plt.subplots()
```

```
fig.suptitle('Meritev upora modela');

ax.plot(V_m, F_m, 'r-')
ax.plot(V_m, F_m, 'g.')
ax.set_xlabel('$V_m$ [m/s]')
ax.set_ylabel('$F_m$ [N]')
ax.grid(which='both')
ax.grid(which='major', color='#DDDDDD', linewidth=0.8)
ax.grid(which='minor', color='#EEEEEE', linestyle=':', linewidth=0.5)
ax.minorticks_on()
fig.savefig('model_resistance_measurements.pdf')
```



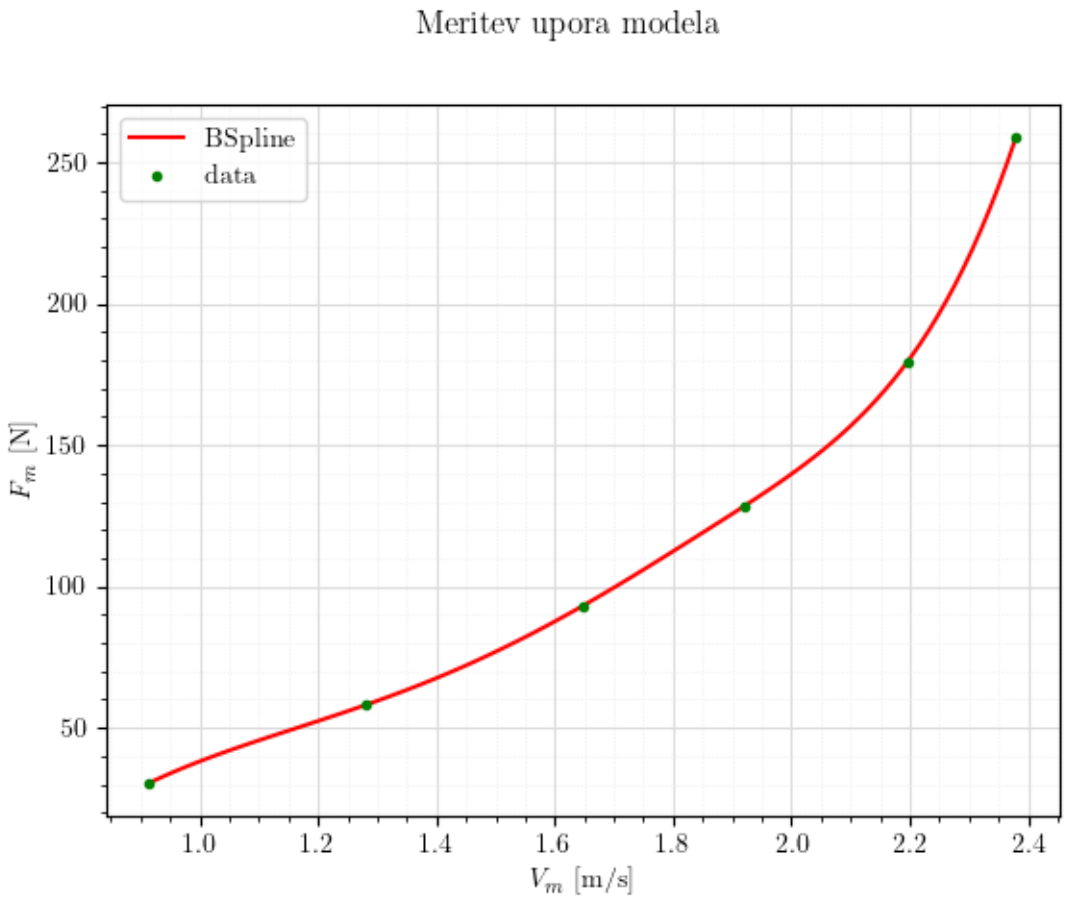

## 1.2 Interpoliranje podatkov meritve

Podatke meritve interpoliramo s pomočjo kubičnih zlepkov, da dobimo gladko krivuljo. Nakar vzamemo 100 merskih toč iz interpolirane krivulje.

[4]:
```
# izdelamo BSpline krivuljo na merskih točkah
t, c, k = interpolate.splrep(V_m, F_m, s=0, k=4)
print('t: {:}\nc: {:}\nk: {:}'.format(t, c, k))
spline = interpolate.BSpline(t, c, k, extrapolate=False)

N = 100
v_min, v_max = V_m.min(), V_m.max()
vv = np.linspace(v_min, v_max, N)
ff = spline(vv)
```

```
t: [0.9151 0.9151 0.9151 0.9151 0.9151 1.7842 2.3795 2.3795 2.3795 2.3795
 2.3795]
c: [ 30.5507      52.21157697  58.70692262 127.9551111  165.69969387
 259.0008       0.           0.           0.           0.
   0.        ]
k: 4
```

[5]:
```
fig, ax = plt.subplots()
fig.suptitle('Meritev upora modela');

ax.plot(vv, ff, 'r-', label='BSpline')
ax.plot(V_m, F_m, 'g.', label='data')
ax.set_xlabel('$V_m$ [m/s]')
ax.set_ylabel('$F_m$ [N]')
ax.grid(which='both')
ax.grid(which='major', color='#DDDDDD', linewidth=0.8)
ax.grid(which='minor', color='#EEEEEE', linestyle=':', linewidth=0.5)
ax.minorticks_on()
ax.legend()

fig.savefig('model_resistance_interpolation.pdf')
```




### 1.3 Pretvorba meritev v brezdimenzijske vrednosti

Podatke pretvorimo v brezdimenzijsko formo, kjer pretvorimo

- hitrost modela $V_m$ v Froude število

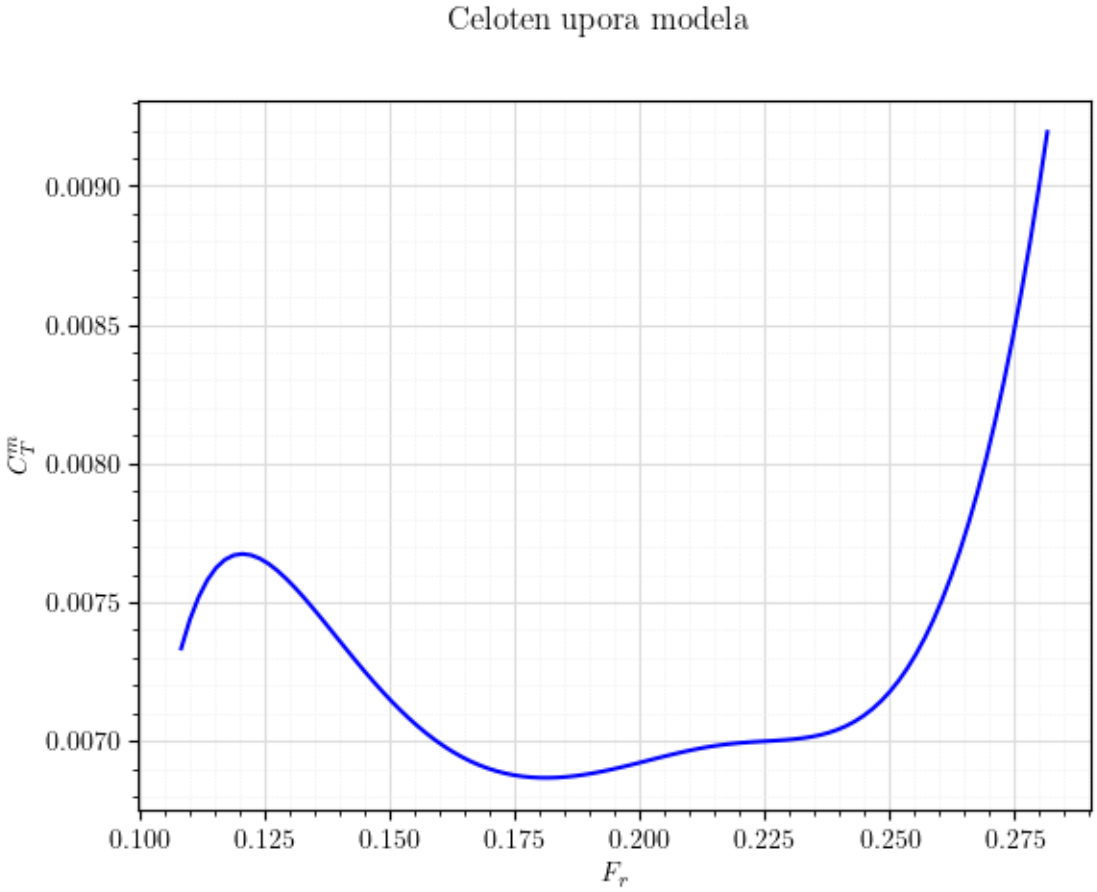
$$F_r = \frac{V_m}{\sqrt{g\, L_{pp}}}$$

- silo $F_m$ v koeficient celotnega upora modela

$$C_T^m = \frac{F_m}{\frac{1}{2}\rho_m\, V_m^2\, S_m}$$

[6]:
```
dDimInt = vv.shape[0]

Fn = vv/mat.sqrt(9.81*Lpp_m)
Ct_m = np.zeros(dDimInt);
for i in range(dDimInt):
```

```
    Ct_m[i] = ff[i]/(0.5*rho_m*vv[i]**2*S_m);

fig, ax = plt.subplots()
fig.suptitle('Celoten upora modela');

ax.plot(Fn, Ct_m, 'b-')
ax.set_xlabel('$F_r$')
ax.set_ylabel('$C_T^m$')
ax.grid(which='both')
ax.grid(which='major', color='#DDDDDD', linewidth=0.8)
ax.grid(which='minor', color='#EEEEEE', linestyle=':', linewidth=0.5)
ax.minorticks_on()
fig.savefig('model_resistance_total_dimless.pdf')
```




### 1.3.1 Določitev minimalnega koeficienta celotnega upora modela

S pomočjo interpolirane funkcije, lahko poiščemo tudi njen odvod. V primeru, ko je ta enak 0 smo v lokalnem nimimu ali maksimu. Poiščemo lokalni minimum, ki je naš optimalno Froudejevo število za minimalni koeficient upora.

[7]:
```
# izdelamo BSpline krivuljo na merskih točkah
t, c, k = interpolate.splrep(Fn, Ct_m, s=0, k=4)
#print('t: {:}\nc: {:}\nk: {:}'.format(t, c, k))
Ct_m_spl = interpolate.BSpline(t, c, k, extrapolate=False)

d_Ct_m_spl = interpolate.splder(Ct_m_spl)
extrema = interpolate.sproot(d_Ct_m_spl)

opt_Fn = extrema[1]
opt_Ct_m = Ct_m_spl(opt_Fn)
print('Optimalen rezim plovbe modela:\n\t Fn: {:.5f}\n\t Ct_m: {:.5e}'.
 ↪format(opt_Fn, opt_Ct_m))
```

```
Optimalen rezim plovbe modela:
         Fn: 0.18125
         Ct_m: 6.86697e-03
```

### 1.4 Rezidualni del upora

*Od sedaj naprej uporabimo **interpolirane podatke**, da je meritev bolj zvezna!!*

Preostali ali residualni del upora, določimo s formulo

$$C_R^m = C_T^m - (1+k)C_F^m + C_A^m.$$

Celotnemu brezdimenzijskem uporu $C_T^m$ tako odštejemo upora trenja $C_F^m$, upor forme $k\,C_F^m$ in upora zraka $C_A^m$. Koeficient forme lahko določimo tudi empirično

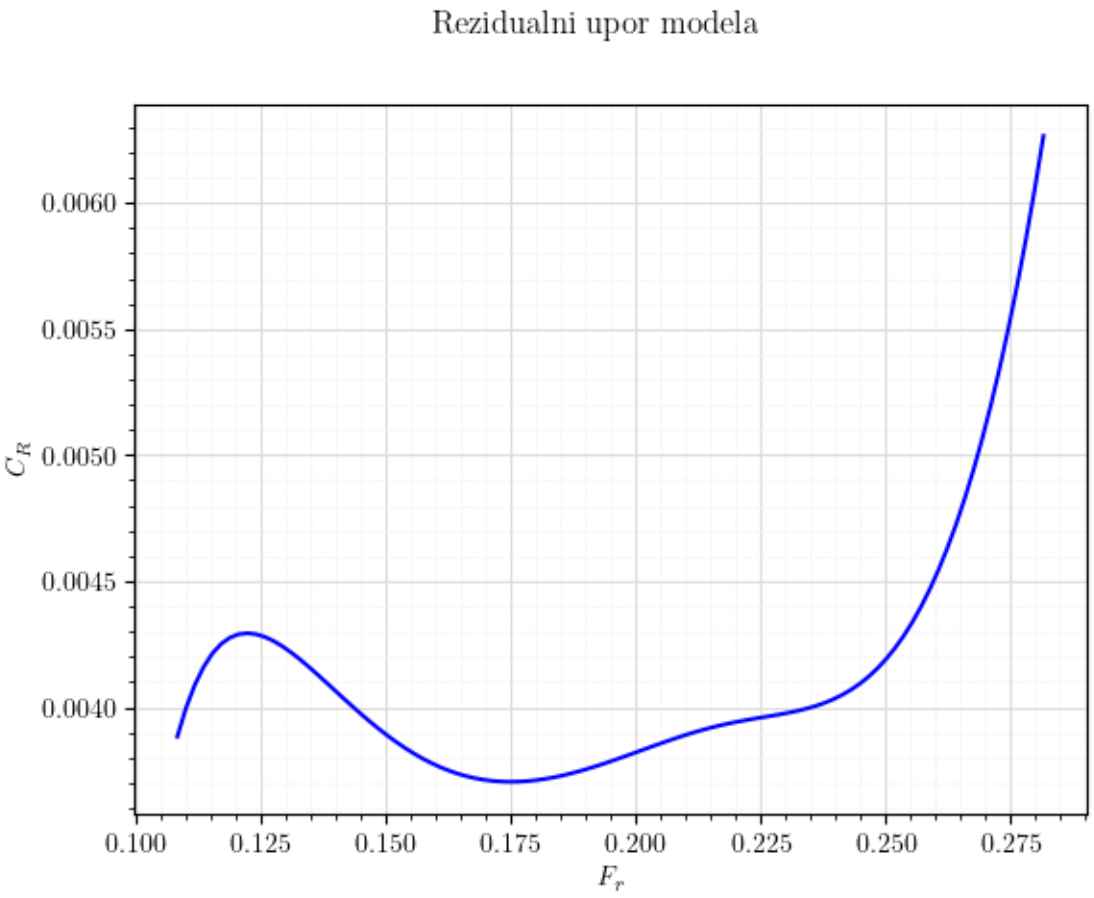
$$k = 0.6\,\phi + 145.0\,\phi^{3.5},$$

kjer je

$$\phi = \frac{C_B}{L_{WL}}\sqrt{B(T_A + T_F)}.$$

[8]:
```
Rn_m = vv*Lpp_m/(mu_m/rho_m)
phi = Cb/Lpp_m*mat.sqrt(B_m*(Tf_m + Ta_m))
k = 0.6*phi + 145.0*phi**3.5

Cf_m = np.zeros(dDimInt)
for i in range(dDimInt):
    Cf_m[i] = 0.075 / (mat.log10(Rn_m[i]) - 2)**2

Cr = np.zeros(dDimInt)
for i in range(dDimInt):
    Cr[i] = Ct_m[i] - (1+k)*Cf_m[i] - Cair
```

```
fig, ax = plt.subplots()
fig.suptitle('Rezidualni upor modela');

ax.plot(Fn, Cr, 'b-')
ax.set_xlabel('$F_r$')
ax.set_ylabel('$C_R$')
ax.grid(which='both')
ax.grid(which='major', color='#DDDDDD', linewidth=0.8)
ax.grid(which='minor', color='#EEEEEE', linestyle=':', linewidth=0.5)
ax.minorticks_on()
fig.savefig('model_resistance_rezidual.pdf')
```




## 1.5 Določitev upora ladje s pomočjo upora modela

Sedaj lahko določimo celoten upor ladje, kjer pa je potrebno upoštevati še podatek o hrapavosti površine. Popravek trenja zaradi hrapavosti površine določimo s pomočjo Townsin (1990) formule, ki jo je privzel tudi ITTC

$$\Delta C_F = 0.044 \left[ \left( \frac{H\, 10^{-6}}{L_{WL}} \right)^{1/3} - 10.0\, R_n^{-1/3} \right] + 1.25\, 10^{-4}$$

V primeru testov vzamemo za hrapavost vedno vrednost $H = 150$ (hrapavost se meri v *mikronih* [$10^{-6}$m]).

Koeficient trenja določimo po ITTC'59 formuli

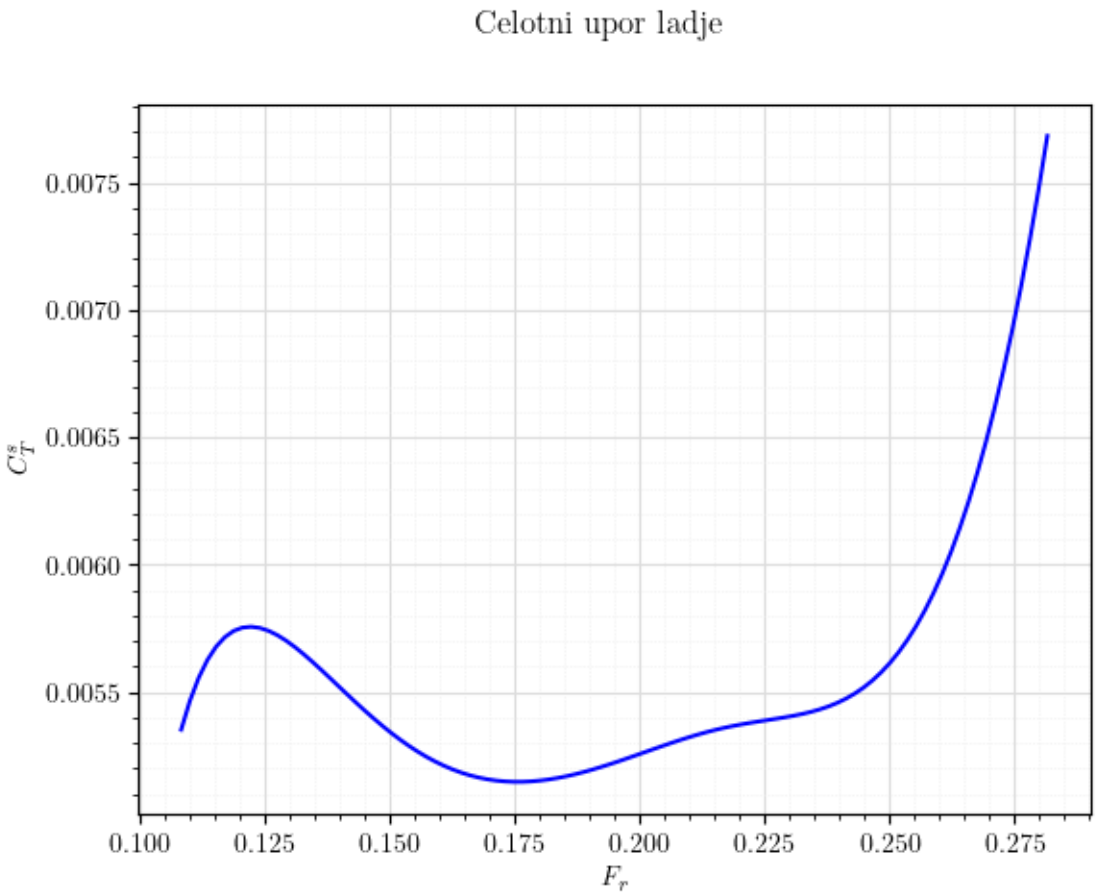
$$C_F^s = \frac{0.075}{(\log R_e - 2)^2}, \quad Re = \frac{V\, L}{\nu}, \quad \nu = \frac{\mu}{\rho},$$

kjer je $R_e$ Reynoldsovo število, $\nu$ pa kinematična viskoznost.

Poleg dodatnega popravka zaradi hrapavosti, je potrebno upoštevati še korelacijski koeficient $C_{ms}$, ki ga določa vsak inštitut zase na podlagi dolgoletnih izkušenj. Korelacijski koeficient določa popravek za meritev med uporom modela in rešitvijo upora ladje. Celoten upor ladje je sedaj enak

$$C_T^s = (1+k)(C_F^s + \Delta C_F) + C_R + C_A^s + C_{ms}$$

[9]:
```
H = 150
Cms = -2.28e-4
V_s = Fn*mat.sqrt(9.81*Lpp_s)
phi = Cb/Lpp_s * mat.sqrt(B_s*(Tf_s + Ta_s))
k = 0.6 * phi + 145 * phi**3.5

Ct_s = np.zeros(dDimInt)
for i in range(dDimInt):
    Rn_s = V_s[i] * Lpp_s/(mu_s/rho_s);
    Cf_s = 0.075 / (mat.log10(Rn_s) - 2)**2
    DCf = 0.044 * ((H*1e-6/Lpp_s)**(1/3) - 10*Rn_s**(-1/3)) + 1.25e-4
    Ct_s[i] = (1+k)*(Cf_s + DCf) + Cr[i] + Cair + Cms

fig, ax = plt.subplots()
fig.suptitle('Celotni upor ladje');

ax.plot(Fn, Ct_s, 'b-')
ax.set_xlabel('$F_r$')
ax.set_ylabel('$C_T^s$')
ax.grid(which='both')
ax.grid(which='major', color='#DDDDDD', linewidth=0.8)
ax.grid(which='minor', color='#EEEEEE', linestyle=':', linewidth=0.5)
ax.minorticks_on()
fig.savefig('model_resistance_total_ship_Fr.pdf')
```



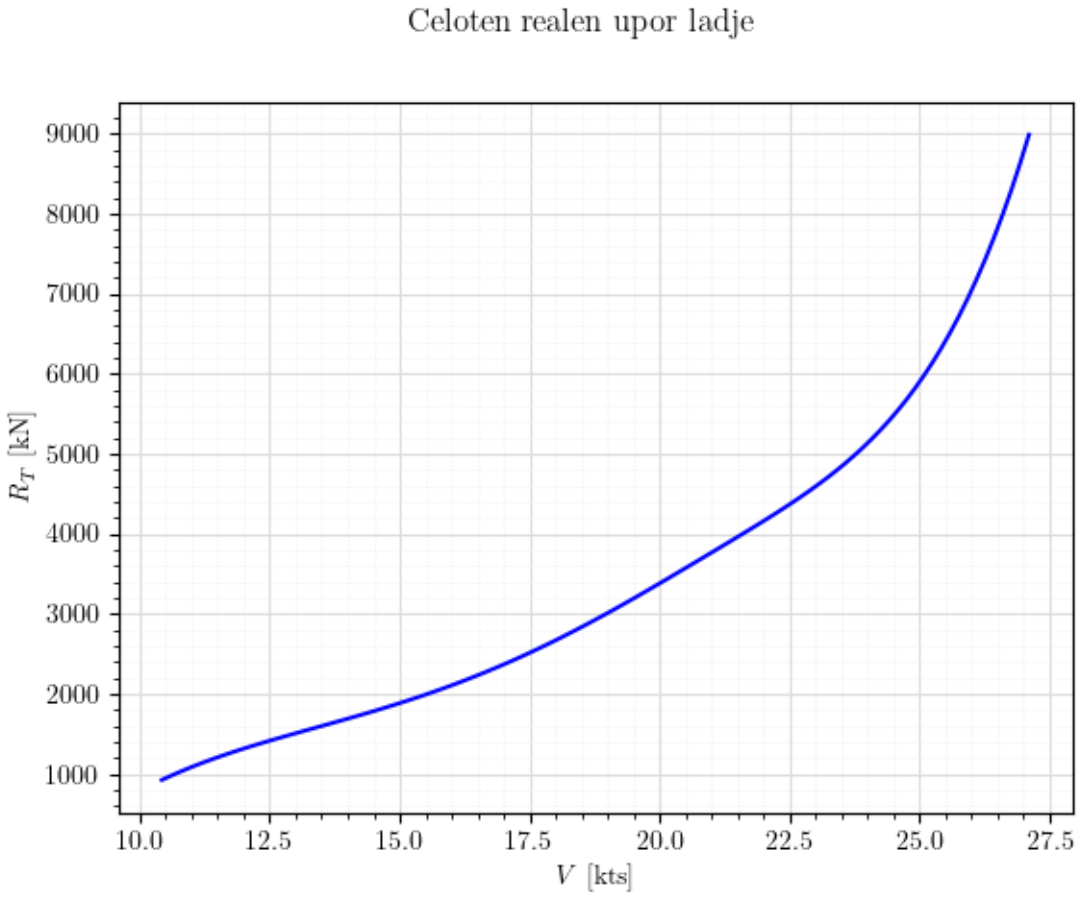

### 1.6 Celoten upor ladje in efektivna moč

Celoten upor ladje določimo s pomočjo koeficienta celotnega upora

$$R_T = \frac{1}{2} C_T^s \, \rho \, V_s^2 \, S$$

[10]:
```
R_s = np.zeros(dDimInt);
for i in range(dDimInt):
    R_s[i] = 0.5*rho_s*V_s[i]**2*Ct_s[i]*S_s/1000;

fig, ax = plt.subplots()
fig.suptitle('Celoten realen upor ladje');

ax.plot(V_s*3600/1852, R_s, 'b-')
ax.set_xlabel('$V$ [kts]')
ax.set_ylabel('$R_T$ [kN]')
ax.grid(which='both')
ax.grid(which='major', color='#DDDDDD', linewidth=0.8)
```

```
ax.grid(which='minor', color='#EEEEEE', linestyle=':', linewidth=0.5)
ax.minorticks_on()
fig.savefig('model_resistance_total_ship.pdf')
```




Iskanje optimalne hitrosti poteka na način, da poiščemo najnižji koeficient celotnega upora v intervalu načrtovane hitrosti ladje, ki jo določa ladjar

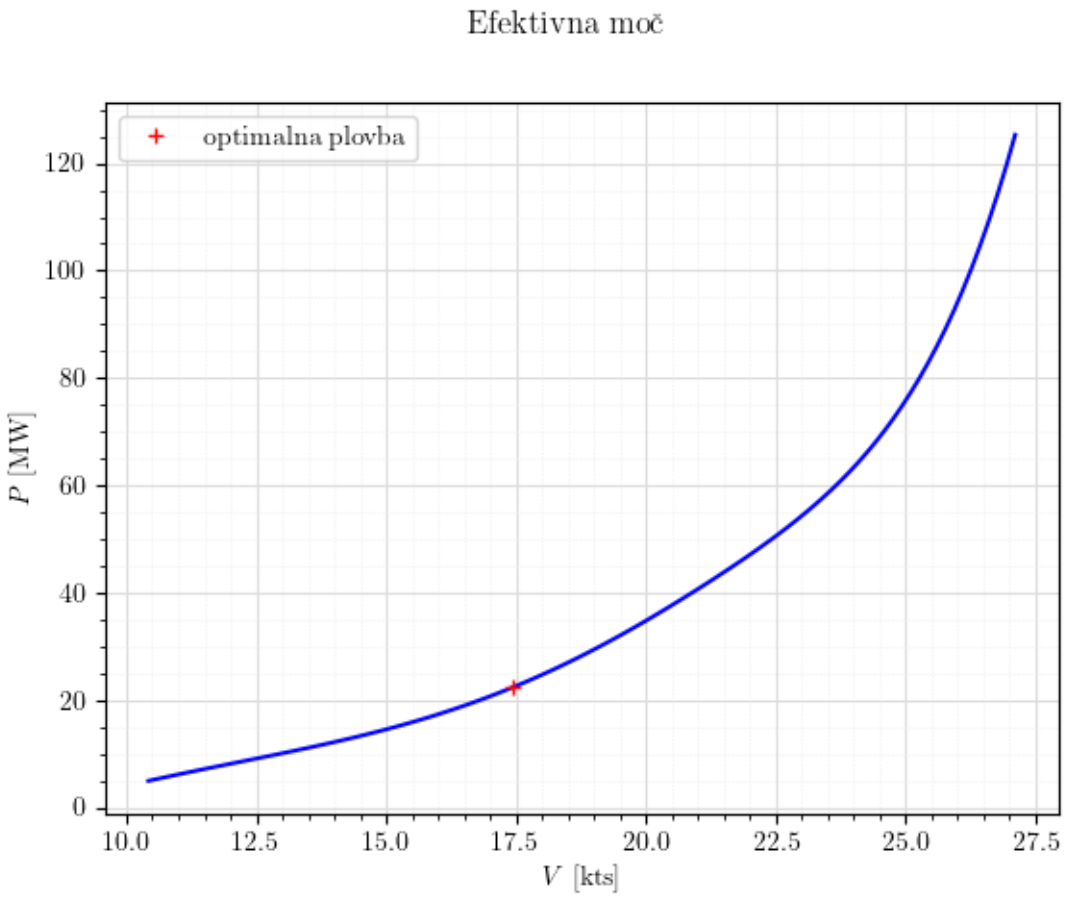
[11]:
```
Fr_opt = opt_Fn;
Vx_opt = Fr_opt * mat.sqrt(9.81 * Lpp_s) * 3600/1852
Rx_opt = np.interp(Vx_opt, V_s*3600/1852, R_s) # smo interpolirali tabelirane␣
 ↪podatke
print('Rx = {:.2f} kN for speed Vx = {:.2f} kts (Fr={:.2f})'.format(Rx_opt,␣
 ↪Vx_opt, Fr_opt))
```

```
Rx = 2499.19 kN for speed Vx = 17.45 kts (Fr=0.18)
```

Efektivna moč, ki je potrebna za doseganje določene hitrosti

$$P_e = R_s \, V_s$$

[14]:
```
P_eff = np.multiply(R_s,V_s)
P_eff_opt = Rx_opt * Fr_opt * mat.sqrt(9.81 * Lpp_s)

fig, ax = plt.subplots()
fig.suptitle('Efektivna moč');

ax.plot(V_s*3600/1852, P_eff/1000, 'b-')
ax.plot(Vx_opt,P_eff_opt/1000, 'r+', label='optimalna plovba')
ax.set_xlabel('$V$ [kts]')
ax.set_ylabel('$P$ [MW]')
ax.grid(which='both')
ax.grid(which='major', color='#DDDDDD', linewidth=0.8)
ax.grid(which='minor', color='#EEEEEE', linestyle=':', linewidth=0.5)
ax.minorticks_on()
ax.legend()
fig.savefig('model_eff_power_ship.pdf')
```



[13]:
```
print('P_eff = {:.2f} kW @ v= {:.2f} kts, Fr = {:.3f}, H = {:.1f}'.
  ↪format(P_eff_opt, Fr_opt * mat.sqrt(9.81 * Lpp_s)*3.6/1.852, Fr_opt, H))
```

```
P_eff = 22432.80 kW @ v= 17.45 kts, Fr = 0.181, H = 150.0
```