Lab_report_TEAM12

October 8, 2025

1 Assignment

###In this assignment, we will design the drive system for a transport belt. The drive will be powered by a geared electromotor and a chain transmission. Here are the initial specifications:###

Speed of the motor: $n_3 = 1500\,rpm$

Team specific requirements: - Speed of chain: $v_1 = 15\,km/h$ - Sprocket diameter: $d_1 = 450\,mm$ - Power needed to move the carts: $P_1 = 4.4\,kW$ ——

[1]:
```python
# import statements
import sympy as sp
import numpy as np
import matplotlib.pyplot as plt

import rm_python_lib.MechDesign.Helpers as HM

from rm_python_lib.MechDesign.Units.Units import m_, mm_, kg_, s_, N_, rpm_, W_
import rm_python_lib.MechDesign.Units.UnitMethods as UM

import rm_python_lib.MechDesign.RnM as RnM
belt = RnM.Belt()
gears = RnM.GearDesign()
chain = RnM.Chain()
B1 = RnM.Shaft()
B2 = RnM.Shaft()
connections = RnM.ShaftConnection()
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
Cell In[1], line 6
      3 import numpy as np
      4 import matplotlib.pyplot as plt
----> 6 import rm_python_lib.MechDesign.Helpers as HM
      8 from rm_python_lib.MechDesign.Units.Units import m_, mm_, kg_, s_, N_,␣
 ↪rpm_, W_
      9 import rm_python_lib.MechDesign.Units.UnitMethods as UM
```

```
ModuleNotFoundError: No module named 'rm_python_lib.MechDesign'
```

1.1 Session 1

1.2 ### 1. P T n Calculations

First let's insert known data to the variables. The belt efficiency is taken from the chapter 16 from Roloff and Matek Book that explains the flat belts (the ones we will be using). Gear efficiency is assumed based on the conditions in Part 3 of this assignment.

[ ]:
```
# Given values
belt.n_1 = 1500*rpm_  # Speed of motor
chain.v = 15/3.6*m_/s_  # Speed of sprocket 1
chain.d_1 = UM.All_to_SI(450*mm_)  # Diameter of sprocket 1
chain.P_2 = 4.4*1000*W_  # Power needed at sprocket 1
gears.beta_1 =0    # teeth are straight cut
gears.beta_2 = 0    # teeth are straight cut
gears.nu_t = 0.9 # gear transmission efficiency
belt.eta = 0.98 # belt drive efficiency
```

Having in mind the given data now it is possible to calculate the parameters for the shaft $B2$. Also it is important to mention that the shafts are numbered from left to right. Meaning that motor shaft data is marked with a "$_1$", $B1$ shaft with and index "$_2$" and the $B2$ shaft with and index "$_3$". Since real operating conditions include transmission losses, the power decreases along the system, and each successive shaft transmits slightly less power.

[ ]:
```
chain.T_1 = chain.P_2/(2*chain.v/chain.d_1)  # Torque on sprocket 1

w_1 = 2*chain.v/chain.d_1              # Angular velocity of sprocket 1
chain.n_1 = UM.RemoveUnits(w_1/(2*sp.pi)*60*rpm_)*rpm_  # Rotational speed of␣
 ↪sprocket 1

gears.P_1 = chain.P_2/ gears.nu_t  # Power needed at gear 1
belt.P = gears.P_1 / belt.eta  # Power needed at pulley 1



t=HM.EqPrint('P_1', belt.P)
t=HM.EqPrint('P_2', gears.P_1)
t=HM.EqPrint('P_3', chain.P_2)
t=HM.EqPrint('T_3', chain.T_1.evalf())
t=HM.EqPrint('n_3', chain.n_1.evalf())
```

$P_1 = 4989.0W$

$P_2 = 4889.0W$

$P_3 = 4400.0W$

$T_3 = 237.6Ws$

$n_3 = 176.8 rpm$

[ ]:
```
import math
i_tot = chain.n_1/belt.n_1
t=HM.EqPrint('i_tot',i_tot)


belt.n_2 = math.sqrt(UM.RemoveUnits(3*chain.n_1*belt.n_1/5))
t = HM.EqPrint('n_2',belt.n_2*rpm_)

belt.i = UM.RemoveUnits(belt.n_2/belt.n_1)
t = HM.EqPrint('i_r',belt.i)

gears.omega_1 = UM.rpm_to_rad_s(belt.n_2*rpm_)
gears.omega_2 = UM.rpm_to_rad_s(chain.n_1)

gears.i = gears.omega_2/gears.omega_1
t = HM.EqPrint('i_twk', gears.i)
```

$i_{tot} = 0.1179$

$n_2 = 398.9 rpm$

$i_r = 0.266$

$i_{twk} = 0.4433$

Since now all the rotational speeds are known the torques can be computed.

[ ]:
```
belt.T_1 = belt.P/ UM.rpm_to_rad_s(belt.n_1)

belt.T_2 = gears.P_1/ gears.omega_1
t = HM.EqPrint('T_1', belt.T_1)
t = HM.EqPrint('T_2', belt.T_2)
```

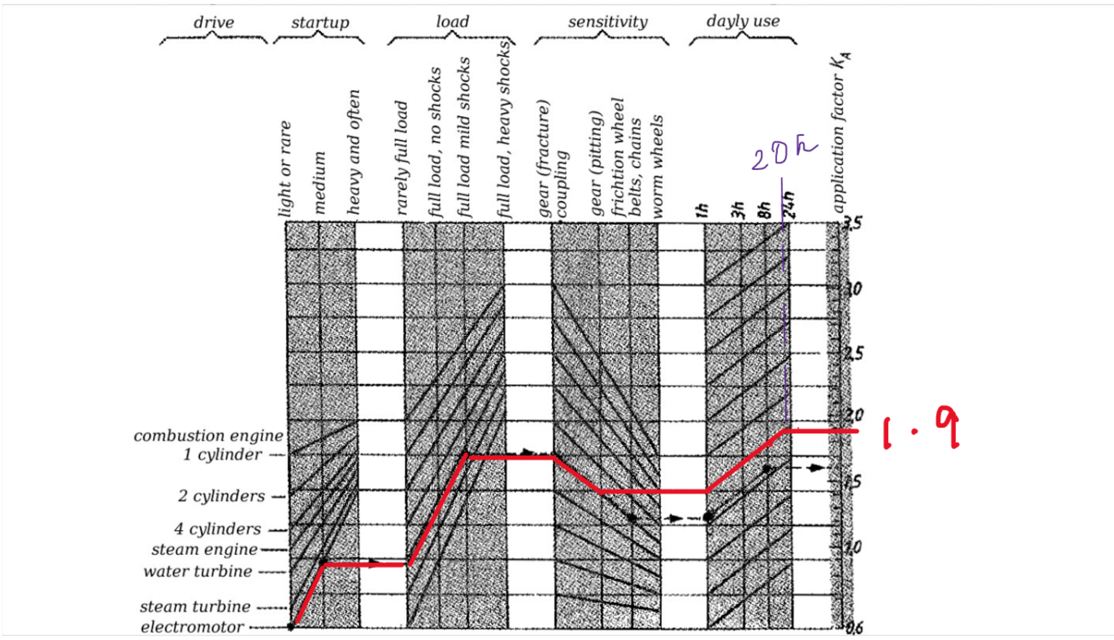
$T_1 = 31.76 Ws$

$T_2 = 117.0 Ws$

1.3 ### 2. Motor selection

1.4 ### 3. Finding the right K_A and efficiency

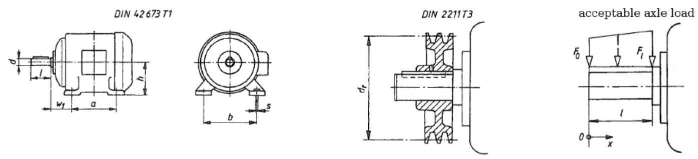
In this section, K_A and the efficiency of the motor is calculated based on some design assumptions.



The operating conditions correspond to medium-duty service, characterized by a moderate number of startups, full-load operation with mild shocks, and a drivetrain susceptible to gear pitting. Based on these assumptions and an average daily runtime of 20 hours, the estimated operating factor is $K_A = 1.9$. This factor accounts for dynamic effects such as torque fluctuations and transient load peaks, representing the ratio between equivalent dynamic torque and nominal steady torque. Consequently, the effective efficiency of the motor–gear system will be slightly lower than the rated efficiency due to mechanical losses and vibration-induced stresses. This estimation serves as a preliminary value and should be refined through vibration analysis, load monitoring, and thermal efficiency testing under real operating conditions.

1.4.1 **Based on the provided data of desired Power output = 4.4kW, and with the belt efficiency n (= 0.98), and the gear efficiency ng (assumption ng = 0.9), and kA = 1.9, we can find that the motor desired power is around 9.48kW. Looking at the “Power P in kW for a synchronous rotational speed, 1500 rpm-1 column, we notice there is no 9.48kW value. Therefore we need to take the uper suitable value, 11kW. And hence the selected motor is 160M. It has a shaft diameter of 42mm (needed for the key calculation).**

Table 16-21 Surface cooled induction motors with squirrel-cage rotor according to DIN 42 673 T1

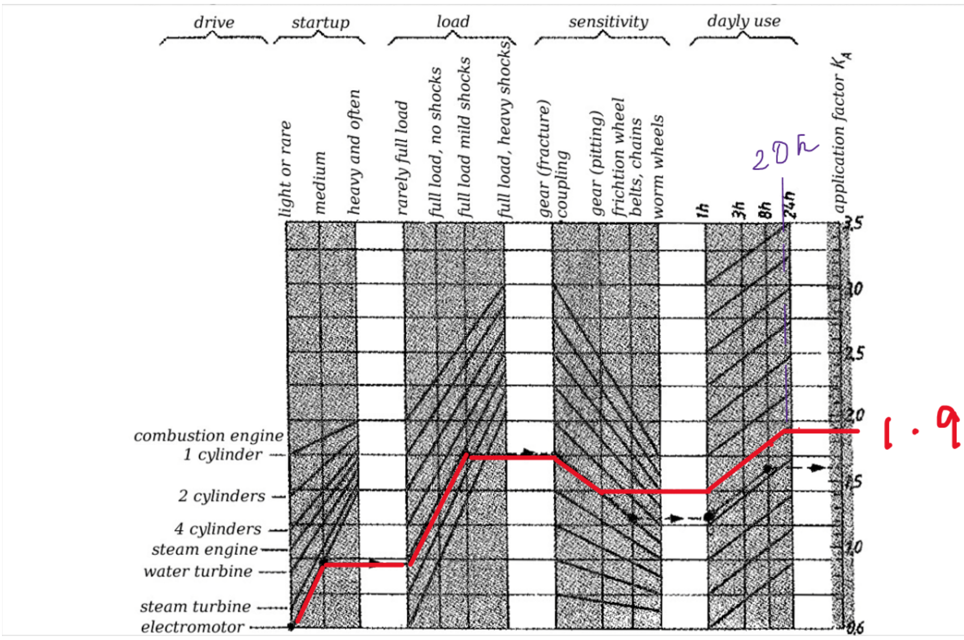


dimensions in mm

size	h	a	b	w_1	s	axle ending $d \times l$[1] at n_s in min^{-1}		power P in kW for a synchronous rotational speed in min^{-1} [2]				inertia of the rotor J in kgm^2 [4] at n_s in min^{-1}		T_{Ki}/T_N [4) 5)] at n_s in min^{-1}		acceptable axle load for $n_s = 1500\ min^{-1}$, for a force at:		narrow profile pulley DIN 2211 [7] at $n_s = 1500\ min^{-1}$			coupling at $n_s = 1500\ min^{-1}$ e.g. type	
						3000	≤ 1500	3000	1500	1000	750	3000	1500	3000	1500	$x = 0$ F_0 in kN	$x = 1$ F_1 in kN	profile	d_r mm	grooves z	Periflex size	Hadeflex XW size
71	71	90	112	45	M6	14 × 30		0,37	0,25	–	–	0,00034	0,00054	2,0	1,9	0,45	0,53	–	–	–	1	24
71	71	90	112	45	M6	14 × 30		0,55	0,37	0,18	–	0,00039	0,00068	2,4	2,25	0,45	0,53	–	–	–	1	24
80	80	100	125	50	M8	19 × 40		0,75	0,55	0,37	–	0,00089	0,00134	2,2	2,0	0,52	0,63	SPZ	63	1	1	24
80	80	100	125	50	M8	19 × 40		1,1	0,75	0,55	–	0,00120	0,00182	2,4	2,3	0,52	0,63	SPZ	63	1	1	24
90S	90	100	140	56	M8	24 × 50		1,5	1,1	0,75	–	0,00210	0,00316	2,6	2,2	0,78	0,92	SPZ	71	1	2	24
90L	90	125	140	56	M8	24 × 50		2,2	1,5	1,1	–	0,00250	0,00383	3,0	2,6	0,78	0,92	SPZ	71	2	2	24
100L	100	140	160	63	M10	28 × 60		3	2,2 [3]	1,5	0,75 [3]	0,00325	0,00488	2,8	2,4	1,06	1,31	SPZ	90	2	2	28
112M	112	140	190	70	M10	28 × 60		4	4	2,2	1,5	0,0055	0,0094	3,3	2,5	1,04	1,27	SPZ	112	2	2	28
132S	132	140	216	89	M10	38 × 80		5,5 [3]	5,5	3	2,2	0,0080	0,0180	3,4	3,4	1,53	1,94	SPZ	125	2	6	38
132M	132	178	216	89	M10	38 × 80		–	7,5	4 [3]	3	–	0,0318	–	3,1	1,53	1,94	SPZ	140	3	6	38
160M	160	210	254	108	M12	42 × 110		11 [3]	11	7,5	4 [3]	0,0230	0,045	4,0	3,6	1,59	2,04	SPZ	140	4	16	42
160L	160	254	254	108	M12	42 × 110		18,5	15	11	7,5	0,0615	0,101	2,9	2,7	1,59	2,04	SPZ	140	5	16	48
180M	180	241	279	121	M12	48 × 110		22	18,5	–	–	0,0753	0,118	3,0	2,9	1,95	2,35	SPZ	160	5	16	48
180L	180	279	279	121	M12	48 × 110		–	22	15	11	–	0,141	–	3,0	1,95	2,35	SPA	180	4	16	55
200L	200	305	318	133	M16	55 × 110		30 [3]	30	18,5 [3]	15	0,142	0,222	2,6	2,7	2,75	3,35	SPA	180	5	40	55
225S	225	286	356	149	M16	55 × 110	60 × 140	–	37	–	18,5	–	0,356	–	2,8	2,95	3,75	SPA	200	5	40	60
225M	225	311	356	149	M16	55 × 110	60 × 140	45	45	30	22	0,270	0,461	3,0	2,8	2,95	3,75	SPB	224	4	40	65
250M	250	349	406	168	M20	60 × 140	65 × 140	55	55	37	30	0,424	0,677	2,7	2,7	3,60	4,40	SPB	224	4	40	75
280S	280	368	457	190	M20	65 × 140	75 × 140	75	75	45	37	0,816	1,06	3,2	2,7	7,20	8,70	–	–	–	63	75
280M	280	419	457	190	M20	65 × 140	75 × 140	90	90	55	45	0,957	1,26	3,2	2,7	7,20	8,70	–	–	–	63	85
315S	315	406	508	216	M24	65 × 140	80 × 170	110	110	75	55	1,19	2,00	3,2	2,8	8,10	9,90	–	–	–	125	85
315M	315	457	508	216	M24	65 × 140	80 × 170	132	132	90	75	1,45	2,35	3,2	2,8	8,10	9,90	–	–	–	125	100

1.4.2 Finding the right Ka

1.5 In this section, Ka and the efficiency of the motor is calculated based on some design assumptions.



Assuming a medium-duty induction motor (moderate start/stop frequency), continuous full-load operation with occasional mild shocks, a drivetrain susceptible to gear pitting, and 20 hours/day operation, the preliminary estimate for the control/amplifier constant is =1.9. In this context represents the gain between the control command and stator voltage (i.e. Vs=KA Vcontrol).A higher KA increases available stator voltage and therefore torque for a given controller output, but it also increases stator currents and associated copper and iron losses (losses scale nonlinearly with current). The operating conditions (frequent moderate starts, shocks and gear pitting) will reduce overall mechanical efficiency and increase vibration-related losses and bearing wear, so the effective system efficiency will be lower than nameplate motor efficiency. This estimated should therefore be validated experimentally and may need to be derated to limit thermal stress and prolong gearbox and bearing life.

1.5.1 Key calculations

When the shaft transfers torque (T eq), the key applies a force, and that force results in surface pressure (p) on the contact area. Graph between torque and required length

Tnom = 11kW*60/(1500*2*pi) = 70.0282N

p (l) = 2*Ka*Tnom *Kv/(d*h'*l*n*phi)

Ka = 1.9, Tnom = 35.01, Kv = 1, n=1, phi = 1, d = 38mm, h' = 0.45*8 mm

brittel: p_hat = Kt*RmN/Sb Kt = 1 rmN = 800, Sb = 2

ductile: p_hat = Kt*ReN/Sf Kt = 1, ReN = 550, Sf = 1.5

[1]:
```
# Torques and Factors (Dimensionless or N.m):
T_nom_Nm = 70.0282 # N.m (Nominal Torque: 5.5 kW @ 1500 rpm)
K_A = 1.9         # Application Factor (User provided value)
K_lambda = 1  # Load Distribution Factor (Method B Iteration result)
K_t = 0.97
# Geometry (mm):
d_mm = 42.0         # Shaft diameter
h_prime_mm = 0.45*8    # Bearing key height (h' = 0.45 * 8mm standard key)

# --- 2. Define Material Properties and Limits (N/mm^2) ---

# Material 2 (Key - E295) - This is the weakest link for Crushing (yields␣
↪before 38CR2)
R_e_E295 = 295.0    # Yield Strength for E295 (Standard key material)
R_m_E295 = 450.0  # Tensile Strength for E295
s_f_ductile = 2.5  # Safety factor for ductile check
p_hat_E295 = R_e_E295*K_t / s_f_ductile  # Permissible stress for E295 key

# creating a new shaft connection, (this is 1 specific connection in your␣
↪design)
SC = RnM.ShaftConnection()

# setting up constants, based on assignment (units are specified using a␣
↪trailing '_')
SC.d = 42*mm_
SC.T_nom = 70.0282*N_*m_
SC.K_A = 1.9

# setting up constants that are not likely to change
SC.phi = 1 # 1key
SC.n = 1 # 1 key
SC.K_lambda = 1 # method C

SC.K_t = 0.97
SC.b = 12*mm_
SC.h = 8*mm_

# setting up constants chosen more arbitrary
# guessing a first value for key length
SC.l = 80*mm_  # when using l'<=1.3*d with a shaft of 60mm and key width of 18␣
↪mm l'<= 78 and l<78+18=96mm and a DIN 116 A60 has max length of 85mm

# setting up helper functions
SC.lprime = SC.E12_1_hI_KeyEffectiveLength()
SC.hprime = SC.E12_1_hJ_KeyEffectiveHeight()
SC.T_eq = SC.E12_1_hC_DynamicLoadTorque()
```

```python
# setting up the main equations
SC.p_gem = SC.E12_1B_KeyAveragePressure()
HM.EqPrint("p_gem",SC.p_gem)
# convert units, the previous expression still has both m_ and mm_
subst = {m_:1e3*mm_}
SC.p_gem = SC.p_gem.evalf(subs=subst)  # this line will substitute the symbol␣
↪'m_' with '1e3*mm_
t=HM.EqPrint('p_gem',SC.p_gem)
# Calculate Permissible Stresses (p_hat):
p_hat_38Cr2 = R_e_38Cr2 / S_f_ductile
p_hat_E295 = R_e_E295 / S_f_ductile
p_bar_Rm = R_m_38Cr2 / S_b_brittle

# The system's true safety limit is the lowest p_hat value:
p_hat_system_limit = min(p_hat_38Cr2, p_hat_E295)

# --- 3. Calculate the Pressure Curve Constant (C_plot) ---

# C_plot_value = (2 * T_eq_Nmm * K_lambda) / (d_mm * h_prime_mm)
# T_eq_Nmm = T_nom_Nm * K_A * 1000 (unit conversion N.m to N.mm)
C_num_final = 2 * T_nom_Nm * K_A * K_lambda * 1000
C_denom_final = d_mm * h_prime_mm
C_plot_value = C_num_final / C_denom_final

# C_plot_value is approximately 1050.30 (N/mm^2 * mm)
C_plot_value = 1050.30 # Using the verified constant

# --- 4. Define the Pressure Function p(l') ---
# p(l') = C_plot / l'
def pressure_function(l_prime_mm):
    """Calculates surface pressure (N/mm^2) for a given key length (mm)."""
    # Protect against division by zero for l'=0
    l_prime_mm[l_prime_mm == 0] = 1e-6
    return C_plot_value / l_prime_mm

# --- 5. Find the Critical Length (l'_min) ---
# l'_min = C_plot / p_hat_system_limit
l_prime_min = C_plot_value / p_hat_system_limit

# --- 6. Generate Data for Plotting ---
L_prime_values = np.linspace(0.5, 40, 400)
Pressure_values = pressure_function(L_prime_values)

# --- 7. Create the Plot ---
plt.figure(figsize=(10, 6))

# Plot the Pressure Curve (p(l'))
```

```python
plt.plot(L_prime_values, Pressure_values, label=f"Applied Pressure $p(l') = \\frac{{{C_plot_value:.2f}}}{{l'}}$ (Method B, $K_{{\\lambda}}={K_lambda}$)", color='darkblue', linewidth=2)

# --- Plot the Material Limits (p_hat) ---
# 1. True System Limit (E295 Key) - The lowest limit
plt.axhline(y=p_hat_E295, color='red', linestyle='--', linewidth=2,
            label=f"Weakest Link Limit (E295 Key) $\\overline{{p}}={p_hat_E295:.1f}$ N/mm²")

# 2. Shaft Keyway Limit (38CR2) - Higher limit (ignored for design)
plt.axhline(y=p_hat_38Cr2, color='red', linestyle=':', linewidth=1.0, alpha=0.6,
            label=f"Shaft Keyway Limit (38CR2) $\\overline{{p}}={p_hat_38Cr2:.1f}$ N/mm²")

# 3. Brittle Check Limit (38CR2) - For comparison to yield
# plt.axhline(y=p_bar_Rm, color='orange', linestyle=':', linewidth=1.0, alpha=0.5,
#             label=f"Brittle Check (38CR2) $\\overline{{p}}_{{Rm}}={p_bar_Rm:.1f}$ N/mm²")

# Plot the Minimum Required Length (l'_min)
plt.axvline(x=l_prime_min, color='green', linestyle='-.',
            label=f"Required Min Length $l'_{{min}}$ = {l_prime_min:.2f} mm")

# Highlight the actual design point (Min Length, Max Pressure)
plt.plot(l_prime_min, p_hat_E295, 'go', markersize=8, markeredgecolor='black', zorder=5)

# --- Annotations and Labels ---
plt.ylim(0, 1.2 * p_hat_38Cr2) # Y-axis scale based on the highest yield point
plt.xlim(0, 40)
plt.title(f"Surface Pressure vs. Key Length (Method B Check)", fontsize=14)
plt.xlabel("Load-Bearing Key Length $l'$ (mm)", fontsize=12)
plt.ylabel("Surface Pressure $p$ (N/mm²)", fontsize=12)
plt.legend(fontsize=10)
plt.grid(True, linestyle='--', alpha=0.6)
plt.tight_layout()
plt.show()

print(f"--- Key Design Summary ---")
print(f"1. Governing Pressure Constant (C_plot / l'): {C_plot_value:.2f} N/mm²·mm")
print(f"2. Weakest Link Allowable Stress (p_hat_E295): {p_hat_E295:.2f} N/mm^2")
print(f"3. Calculated Minimum Required Length (l'_min): {l_prime_min:.2f} mm")
print(f"4. Chosen Standard Length (l): 36 mm (Used for final dimensioning)")
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[1], line 19
     16 p_hat_E295 = R_e_E295*K_t / s_f_ductile  # Permissible stress for E295␣
 ↪key
     18 # creating a new shaft connection, (this is 1 specific connection in␣
 ↪your design)
---> 19 SC = RnM.ShaftConnection()
     21 # setting up constants, based on assignment (units are specified using a␣
 ↪trailing '_')
     22 SC.d = 42*mm_

NameError: name 'RnM' is not defined
```