Autor

Carlos Augusto Jardim Chiarelli **165685**

In [1]:
```
# bibliotecas
from    pandas              import DataFrame
from    numpy               import pi, sqrt, cos, sin, tan, interp
from    platform            import python_version
from    warnings            import filterwarnings
from    IPython.display     import Image

def tabela(var, nome, unidade, arred=1, mul=None):

    if mul: var = {k:v*mul for k, v in var.items()}

    df = DataFrame(var, index=[0])
    df = df.round(arred)
    df[nome] = unidade
    return df

def tabelaProGeomet():

    propGeomet = {'parâmetro':['primitivo 1','primitivo 2','base 1
','base 2','passo circular',
                               'adendo','dedendo','dist centro 1','d
ist centro 2','Z 1','Z 2',
                               'razão contato 1','razão contato 2'],

                  'valor':[D_primitivo[1], D_primitivo[2], D_base
[1],    D_base[2],
                           pc,             adendo,         dedendo,
dist_cent[1],
                           dist_cent[2],   Z[1],           Z[2],
razaoCont[1],
                           razaoCont[2]],

                  'unidades':['m','m','m','m','in','in','in','m','m
','in','in','in','in']
                 }

    return DataFrame(propGeomet).round(2)

filterwarnings('ignore') # remove warnings

print('\nVersão da Linguagem Python usada neste relatório:', python_
version())
```

```
Versão da Linguagem Python usada neste relatório: 3.7.6
```

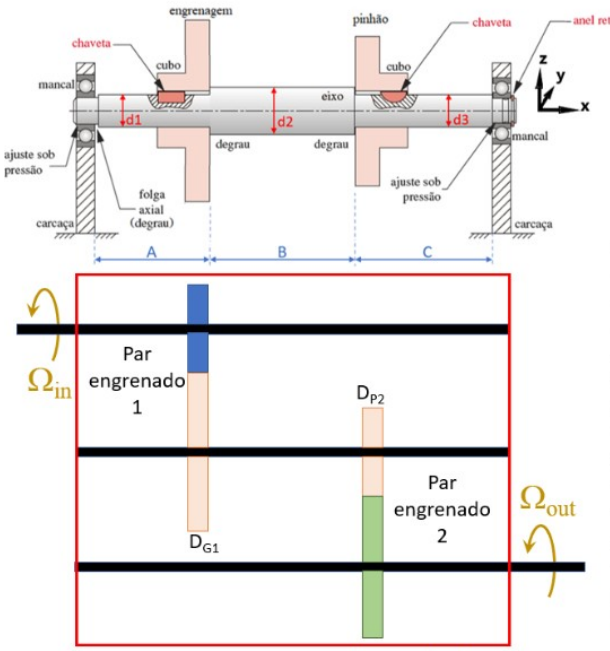
Projeto 03

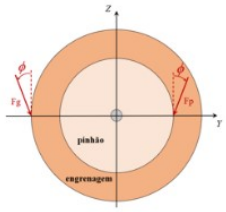
Etapa 1

In [2]:
```
Image('../dados/imagens/proj03_fig_projeto01.PNG')
```

Out[2]:

A Figura 1 apresenta em destaque o eixo de transmissão intermediário utilizado em um trem de engrenagem composto. Este trem de engrenagem representa um redutor de velocidade, cuja relação de velocidade é dada por $m_v = m_{v1} * m_{v2} = 0{,}5 * 0{,}4 = 0{,}2$. Sabendo que a potência deste redutor de velocidade é 3HP e que o eixo de transmissão intermediário opera sob rotação cíclica entre 1500 a 2000RPM, deseja-se agora dimensionar as engrenagens cilíndricas de dentes retos considerando os dados apresentados a seguir.





Dados básicos do Projeto:
- Potência = 3HP
- Rotação do Eixo intermediário = 1500 a 2000RPM
- Ângulo de pressão das engrenagens = 20°
- Razão de velocidade: m_{v1}=0,5 e m_{v2}= 0,4
- Diâmetros: D_{G1}=150mm e D_{P2}=80mm
- Temperatura em Operação < 150°C
- Todas as engrenagens são sólidas e usinadas em aço com índice de qualidade igual à 8.
- Nível de confiança das engrenagens igual 99%.
- Condição de aplicação comercial com máquinas movidas e motoras uniformes
- Dimensionamento com base na norma AGMA, considerando dente sem alongamento de adendo e carregamento HPSTC.

Figura 1 – Trem de engrenagem composto.
Dimensões: A=50mm; B=100mm; C=50mm.

Diante das informações apresentadas, determinar os coeficientes de segurança para fadiga de flexão e fadiga superficial, bem como as características geométricas das engrenagens (diâmetro primitivo, diâmetro da base, passo circular, adendo, dedendo, largura da face, passo diametral e número de dentes) e também do par engrenado (distância entre centros, comprimento da linha de ação e razão de contato).

Dados:

In [3]:
```
# características do projeto

omega_min  = 1500          # velocidade angular [RPM]
omega_max  = 2000          # velocidade angular [RPM]
omega_rpm  = 2000          # velocidade angular [RPM]
P          = 3             # potência [HP]
phi        = 20            # graus (°)

# diâmetros engrenagem e pinhão [m] (RAIOS PRIMITIVOS)
D          = {'g1':150e-3, 'p2':80e-3}

mv1, mv2   = 0.5, 0.4      # razão de velocidade
Temp_op    = 150           # °C
Qv         = 8

especif    = ['Todas as engrenagens são sólidas e usinadas em aço co
m índice de qualidade igual 8',
              'Nível de confiança das engrenagens igual a 99%',
              'Condição de aplicação comercial com máquinas movidas
e motores uniformes',
              'Dimensionamento com base na norma AGMA, considerando
dente sem alongamento de adendo e carregamento HPSTC']

obter      = ['Nf para fadiga de flexão e fadiga superficial',
              'Engrenagem: diâmetro primitivo, diâmetro de base,pass
o circular,adendo,dedendo, largura da fase, passo diametral , número
de dentes',
              'Par engrenado: distância entre centro, comprimento da
linha de ação e razão de contato']
```

In [4]:
```
# conversão unidades
omega_min = omega_min * (2*pi) / 60 # velocidade angular [rad/s]
omega_max = omega_max * (2*pi) / 60 # velocidade angular [rad/s]

phi = phi * (pi / 180) # radianos

P = P * 745.7 # potência [W]
```

In [5]:
```
Image('../dados/imagens/proj03_fig_projeto02.PNG')
```

Out[5]:

Tabela 12-3
Módulos métricos padronizados

Módulo métrico (mm)	Equivalente p_d (in^{-1})
0,3	84,67
0,4	63,50
0,4	50,80
0,8	31,75
1	25,40
1,25	20,32
1,5	16,93
2	12,70
3	8,47
4	6,35
5	5,08
6	4,23
8	3,18
10	2,54
12	2,12
16	1,59
20	1,27
25	1,02

In [6]:
```
# diâmetros
D['p1'] = D['g1']*mv1
D['g2'] = D['p2']/mv2

# dentes da engrenagem e pinhão
# assumindo pd (tabelado)
#pd     = 20.32
pd     = 8.47
calc_N = lambda D: round(pd*D*1e3/25.4, 0) # converte para polegada

N = {}

for local, d in D.items(): N[local] = calc_N(d)

tabela(D, 'Diâmetro', 'metro', 3)
```

Out[6]:

	g1	p2	p1	g2	Diâmetro
0	0.15	0.08	0.075	0.2	metro

In [7]:
```
tabela(N, 'N° engrenagens', 'quantidade')
```

Out[7]:

	g1	p2	p1	g2	N° engrenagens
0	50.0	27.0	25.0	67.0	quantidade

In [8]:
```
# velocidades
w        = {'p1':omega_max}

w['g1'] = w['p1']*mv1
w['p2'] = w['g1']
w['g2'] = w['p2']*mv2

tabela(w, 'Veloc. angular', 'rad/s')
```

Out[8]:

	p1	g1	p2	g2	Veloc. angular
0	209.4	104.7	104.7	41.9	rad/s

In [9]:
```
# Torque
T        = {'p1':P/w['p1']} # entrada

T['g1'] = T['p1']*mv1       # saída
T['p2'] = T['g1']           # entrada
T['g2'] = T['p2']*mv2       # saída

tabela(T, 'Torque', 'N.m')
```

Out[9]:

	p1	g1	p2	g2	Torque
0	10.7	5.3	5.3	2.1	N.m

In [10]:
```
# Cargas
Wt = {1: T['p1']/(D['p1']/2),
      2: T['p2']/(D['p2']/2)}

W  = {1: Wt[1]/cos(phi),
      2: Wt[2]/cos(phi)}

tabela(W, 'Carga total', 'N')
```

Out[10]:

	1	2	Carga total
0	303.1	142.1	N

Encontrando o fator geométrico J.

In [11]:
```
tabela(N, 'N° engrenagens', 'quantidade')
```

Out[11]:

	g1	p2	p1	g2	N° engrenagens
0	50.0	27.0	25.0	67.0	quantidade

In [12]:
```
Image('../dados/imagens/proj03_fig_projeto03.PNG')
```

Out[12]:

Table 11-9 AGMA Bending Geometry Factor J for 20°, Full-Depth Teeth with HPSTC Loading

	Pinion teeth															
Gear teeth	12		14		17		21		26		35		55		135	
	P	G	P	G	P	G	P	G	P	G	P	G	P	G	P	G
12	U	U														
14	U	U	U	U												
17	U	U	U	U	U	U										
21	U	U	U	U	U	U	0.33	0.33								
26	U	U	U	U	U	U	0.33	0.35	0.35	0.35						
35	U	U	U	U	U	U	0.34	0.37	0.36	0.38	0.39	0.39				
55	U	U	U	U	U	U	0.34	0.40	0.37	0.41	0.40	0.42	0.43	0.43		
135	U	U	U	U	U	U	0.35	0.43	0.38	0.44	0.41	0.45	0.45	0.47	0.49	0.49

In [13]:
```
# interpolações
J = {'p1':0, 'g1':0, 'p2':0, 'g2':0}

# par 1
PP_l1   = interp(N['p1'], [21,26],  [.34,.36])
PP_l2   = interp(N['p1'], [21,26],  [.34,.37])
GG_l1   = interp(N['p1'], [21,26],  [.37,.38])
GG_l2   = interp(N['p1'], [21,26],  [.40,.41])

J['p1'] = interp(N['g1'], [35,55],  [PP_l1,PP_l2])
J['g1'] = interp(N['g1'], [35,55],  [GG_l1,GG_l2])

# par 2
PP_l1   = interp(N['p2'], [26,35],  [.34,.40])
PP_l2   = interp(N['p2'], [26,35],  [.38,.41])
GG_l1   = interp(N['p2'], [26,35],  [.41,.42])
GG_l2   = interp(N['p2'], [26,35],  [.44,.45])

J['p2'] = interp(N['g2'], [55,135], [PP_l1,PP_l2])
J['g2'] = interp(N['g2'], [55,135], [GG_l1,GG_l2])

tabela(J, 'Fator geométrico (J)', 'adimensional', 2)
```

Out[13]:

	p1	g1	p2	g2	Fator geométrico (J)
0	0.36	0.4	0.35	0.42	adimensional

In [14]:
```
# Face
F = (12/pd) / 39.37 # converte para m

# ka (maquinas movidas e motoras uniformes)
ka = 1
```

In [15]:
```
tabela({'F':F*1e3}, 'Largura da face', 'mm')
```

Out[15]:

	F	Largura da face
0	36.0	mm

In [16]:
```
# km (F < 50mm)
km = 1.6

# ks (recomendado pela AGMA)
ks = 1

# kb (espessura da bora)
# supondo mb > 1.2
kb = 1

# ki (fator de ciclo de carga) - não há engrenagem livre
ki = 1

# kv (fator dinâmico)
B   = (12-Qv)**(2/3)/4
A   = 50+56*(1-B)
Vt  = {1: D['p1']/2*w['p1'],
       2: D['p2']/2*w['p2']}

calc_kv = lambda vt: (A/(A+vt**.5))**B

kv  = {1: calc_kv(Vt[1]),
       2: calc_kv(Vt[2])}

tabela(kv, 'Fator dinâmico (kv)', 'adimensional', 3)
```

Out[16]:

	1	2	Fator dinâmico (kv)
0	0.976	0.982	adimensional

In [17]:
```
# Tensão de flexão
def tensao_flexao(wt, pd, F, J, ka, km, ks, kb, ki, kv):

    num1 = wt*pd/(F*J)
    num2 = ka*km*ks*kb*ki/kv

    return num1*num2

pd_m = pd*39.37

sigma_n = {'p1': tensao_flexao(Wt[1], pd_m, F, J['p1'], ka, km, ks,
kb, ki, kv[1]),
           'g1': tensao_flexao(Wt[1], pd_m, F, J['g1'], ka, km, ks,
kb, ki, kv[1]),
           'p2': tensao_flexao(Wt[2], pd_m, F, J['p2'], ka, km, ks,
kb, ki, kv[2]),
           'g2': tensao_flexao(Wt[2], pd_m, F, J['g2'], ka, km, ks,
kb, ki, kv[2])}

tabela(sigma_n, 'Tensão de flexão', 'MPa', 0, mul=1e-6)
```

Out[17]:

	p1	g1	p2	g2	Tensão de flexão
0	12.0	11.0	6.0	5.0	MPa

In [18]:
```
# Tensões de contato
Ca, Cm, Cs, Cv = ka, km, ks, kv

# fator de acabamento superficial (Cf)
Cf = 1

# Coeficiente elástico (Cp)
Ep, Eg = 2*1e5*1e6, 191e6
vp, vg = 0.3,       0.3

num1 = (1-vp**2)/Ep
num2 = (1-vg**2)/Eg

Cp   = sqrt(1/(pi*(num1+num2)))

tabela({'Cp':Cp}, 'Coeficiente elástrico', 'Pa^0.5')
```

Out[18]:

	Cp	Coeficiente elástrico
0	8169.8	Pa^0.5

In [19]:
```
# converte m para in
m2in = lambda x: x*39.37

# raios de curvatura
p_raioCurv = {}

calc_p1 = lambda r1:          sqrt((r1+1/pd)**2 - (r1*cos(phi))**2) -
pi/pd*cos(phi)
calc_p2 = lambda r1, r2, p1: (r1+r2)*sin(phi)-p1

# par 1
p_raioCurv['p1'] = calc_p1(m2in(D['p1']/2))
p_raioCurv['g1'] = calc_p2(m2in(D['p1']/2), m2in(D['g1']/2), p_raioC
urv['p1'])

# par 2
p_raioCurv['p2'] = calc_p1(m2in(D['p2']/2))
p_raioCurv['g2'] = calc_p2(m2in(D['p2']/2), m2in(D['g2']/2), p_raioC
urv['p2'])

# Fator de geometria de superfíce I
calc_I = lambda pp, pl, Dp: cos(phi)/((1/pp + 1/pl)*Dp)

I = {1: calc_I(p_raioCurv['p1'], p_raioCurv['g1'], m2in(D['p1'])),
     2: calc_I(p_raioCurv['p2'], p_raioCurv['g2'], m2in(D['p2']))}

tabela(I, 'Fator de geometria', 'adimensional', 3)
```

Out[19]:

	1	2	Fator de geometria
0	0.099	0.106	adimensional

In [20]:
```
I_min = min(I[1], I[2])

# tensões de contato
calc_sigma_c = lambda wt, D, Cv: Cp*sqrt((wt*Ca*Cm*Cs*Cf)/(F*I_min*D
*Cv))

sigma_c = {1: calc_sigma_c(Wt[1], D['p1'], Cv[1]),
           2: calc_sigma_c(Wt[2], D['p2'], Cv[2])}

tabela(sigma_c, 'Tensões de contato', 'MPa', mul=1e-6)
```

Out[20]:

	1	2	Tensões de contato
0	10.8	7.1	MPa

Coeficientes de segurança: fadiga de flexão e fadiga superficial

In [21]:
```
# coef segurança
calc_N = lambda Sf, sigma: Sf/sigma

# resistência à fadiga de flexão corrigida
calc_Sfb = lambda Kl, Kt, Kr, Sfb_linha: Kl*Sfb_linha/(Kt*Kr)

# resistência à fadiga de flexão não-corrigido (Grau 2)
calc_Sfb_linha = lambda HB: 6235 + 174*HB - 0.126*HB**2
```

Será assumido que a vida em serviço requerida é 6 anos em operação de um turno.

In [22]:
```
# Kl

# número de ciclos
anos    = 6
N_clico = omega_rpm*60*2080*anos*1 # rpm * min/hora * hora/turno ano
* anos * turno

tabela({'N':N_clico}, 'N° ciclos', '1E9', arred=2, mul=1e-9)
```

Out[22]:

	N	N° ciclos
0	1.5	1E9

In [23]:
```
# não é um serviço crítico
Kl = 1.3558*N_clico**-.0178

tabela({'K':Kl}, 'Kl', 'adimensional', 2)
```

Out[23]:

	K	Kl
0	0.93	adimensional

In [24]:
```
Temp_f = Temp_op*9/5 + 32

tabela({'T':Temp_f}, 'Temperatura operação', '°F')
```

Out[24]:

	T	Temperatura operação
0	302.0	°F

In [25]:
```
# Kt (temperatura menor que 250°C)
Kt = (460+Temp_f)/620

# Kr (99% de confiança)
Kr = 1

tabela({'valor':Kt}, 'Kt', 'adimensional')
```

Out[25]:

	valor	Kt
0	1.2	adimensional

In [26]:
```
pa2psi = lambda x: x/6895

# aço endurecimento completo
HB = 300
Sfb_linha = calc_Sfb_linha(300)

Sfb = calc_Sfb(Kl, Kt, Kr, Sfb_linha)

Nb = {k : calc_N(Sfb, pa2psi(tensao)) for k, tensao in sigma_n.items
()}

tabela(Nb, 'Coef segurança flexão', 'adimensional')
```

Out[26]:

	p1	g1	p2	g2	Coef segurança flexão
0	20.6	22.8	43.0	50.7	adimensional

In [27]:
```
# resistência à fadiga de superficie não-corrigido
Sfc_linha = 27000 + 364*HB

# fator de vida Cl
Cl = 1.4488*N_clico**-.023

Ct = Kt
Cr = Kr

# fator de razão de dureza (engrenagens de mesma dureza)
Ch = 1

# resistência à fadiga de superficie corrigida
Sfc = Cl*Ch*Sfc_linha/(Ct*Cr)

Nc = {k : calc_N(Sfc, pa2psi(tensao)) for k, tensao in sigma_c.items
()}

tabela(Nc, 'Coef segurança superfície', 'adimensional')
```

Out[27]:

	1	2	Coef segurança superfície
0	63.0	95.4	adimensional

Após ocálculo dos coeficientes de segurança podemos obter as características geométricas das engrenagens e do par engrenado restantes.

Engrenagem:

- diâmetro primitivo *(obtido)*
- diâmetro de base
- passo circular
- adendo
- dedendo
- largura da fase *(obtido)*
- passo diametral *(obtido)*
- número de dentes

Par engrenado:

- distância entre centros
- comprimento da linha de ação
- razão de contato

In [28]:
```
# diâmetro primitivo [m]
D_primitivo = {1 : D['g1'],
               2 : D['p2']}

# diâmetro de base [m]
D_base = {1: D_primitivo[1]*cos(phi),
          2: D_primitivo[2]*cos(phi)}

# passo circular
pc = pi/pd

# adendo (pd < 20)
adendo = 1/pd

# dedendo (pd < 20)
dedendo = 1.25/pd

# distância entre centros [m]
dist_cent = {1: (D['g1']+D['p1'])/2,
             2: (D['g2']+D['p2'])/2}

# comprimento de ação Z
def calc_z(rp, rg, ap, ag, dist_cent):

    num1 = (rp+ap)**2 - (rp*cos(phi))**2
    num2 = (rg+ag)**2 - (rg*cos(phi))**2

    Z = sqrt(num1) + sqrt(num2) - dist_cent*sin(phi)

    return Z

# [in]
Z = {1: calc_z(m2in(D['p1']/2), m2in(D['g1']/2), adendo, adendo, m2i
n(dist_cent[1])),
     2: calc_z(m2in(D['p2']/2), m2in(D['g2']/2), adendo, adendo, m2i
n(dist_cent[2]))}

# razão de contato
calc_razaoCont = lambda Z: pd*Z/(pi*cos(phi))

razaoCont = {1: calc_razaoCont(Z[1]),
             2: calc_razaoCont(Z[2])}

tabelaProGeomet()
```

Out[28]:

	parâmetro	valor	unidades
0	primitivo 1	0.15	m
1	primitivo 2	0.08	m
2	base 1	0.14	m
3	base 2	0.08	m
4	passo circular	0.37	in
5	adendo	0.12	in
6	dedendo	0.15	in
7	dist centro 1	0.11	m
8	dist centro 2	0.14	m
9	Z 1	0.59	in
10	Z 2	0.60	in
11	razão contato 1	1.68	in
12	razão contato 2	1.71	in

Etapa 2

In [29]:
```
Image('../dados/imagens/proj03_fig_projeto04.PNG')
```

Out[29]:

A Figura 1 mostra a caixa para os mancais nas extremidades do eixo de transmissão composto por duas engrenagens na região central.

Figura 1 – Caixas para montagem dos mancais.

Com base nas análises de forças e esforços já realizadas no dimensionamento do eixo, das chavetas e dos mancais (Projetos 1 e 2), pede-se agora a seguinte etapa:

1. Dimensionar as junções parafusadas para montagem das caixas dos mancais de rolamento e hidrodinâmicos à estrutura, com fatores de segurança (escoamento, separação de junta e fadiga) para um carregamento dinâmico de 0 a 2000 N por junção. O fator de segurança à faiga deve ser no mínimo igual a 2. Considerar:

Padrão ISO – rosca normal
Padrão ISO – rosca normal – classe a definir
Material do parafuso: aço (módulo de elasticidade E = 207 GPa)
Material da caixa: ferro fundido (módulo de elasticidade E = 170 GPa)
Material da base: ferro fundido (módulo de elasticidade E = 170 GPa)
Diâmetro equivalente do material: 30 mm
Diâmetro do furo: 8 mm
Espessura do material da caixa: 10 mm
Espessura do material da base: 50mm
Rosca cortada e acabamento usinado.
Temperatura de operação da máquina.

Será usado escolhida a **Classe 5.8** e parafuso **M8** para efeito dos calculos.

In [30]:
```
# DADOS (enunciado)

F = {'min':0, 'max':2000}                                  # força no paraf
uso [N]

E = {'parafuso':207e9, 'caixa':170e9, 'base':170e9} # constante mate
rial [Pa]

D = {'eff':30e-3, 'furo':8e-3}                             # diametros [m]

espess = {'caixa':10e-3, 'base':50e-3}                     # espessura da m
aterial [m]
```

In [31]:
```
Image('../dados/imagens/proj03_fig_projeto05.PNG')
```

Out[31]:

	Roscas Grossas			Roscas Finas		
Diâmetro Maior d (mm)	Passo p mm	Diâmetro Menor dr (mm)	Área sob tração At (mm²)	Passo p mm	Diâmetro Menor dr (mm)	Área sob tração At (mm²)
3.0	0.50	2.39	5.03			
3.5	0.60	2.76	6.78			
4.0	0.70	3.14	8.78			
5.0	0.80	4.02	14.18			
6.0	1.00	4.77	20.12			
7.0	1.00	5.77	28.86			
8.0	1.25	6.47	36.61	1.00	6.77	39.17
10.0	1.50	8.16	57.99	1.25	8.47	61.20
12.0	1.75	9.85	84.27	1.25	10.47	92.07
14.0	2.00	11.55	115.44	1.50	12.16	124.55
16.0	2.00	13.55	156.67	1.50	14.16	167.25
18.0	2.50	14.93	192.47	1.50	16.16	216.23
20.0	2.50	16.93	244.79	1.50	18.16	271.50
22.0	2.50	18.93	303.40	1.50	20.16	333.06
24.0	3.00	20.32	352.50	2.00	21.55	384.42
27.0	3.00	23.32	459.41	2.00	24.55	495.74
30.0	3.50	25.71	560.59	2.00	27.55	621.20
33.0	3.50	28.71	693.55	2.00	30.55	760.80
36.0	4.00	31.09	816.72	3.00	32.32	864.94
39.0	4.00	34.09	975.75	3.00	35.32	1028.39

In [32]:
```
Image('../dados/imagens/proj03_fig_projeto06.PNG')
```

Out[32]:

Número de Classe	Faixa do diâmetro externo [mm]	Resistência de Prova Mínima [MPa]	Limite de Escoamento Mínimo [MPa]	Resistência a Tração Mínima [MPa]
4.6	M5-M36	225	240	400
4.8	M1.6-M16	310	340	420
5.8	M5-M24	380	420	520
8.8	M16-M36	600	660	830
9.8	M1.6-M16	650	720	900
10.9	M5-M36	830	940	1040
12.9	M1.6-M36	970	1100	1220

In [33]:
```
# PARÂMETRO ESCOLHIDO (classe 5.8 e parafuso M8)

p  = 8         # passo           [mm]
dr = 6.47      # diâmetro menor  [mm]
At = 36.61e-6 # área sob tração [m2]
```

In [34]:
```
# PRÉ-CARGA

# estimando comprimento parafuso [cm]
l = {'l' : (espess['caixa'] + espess['base'])*1e2}

l['parafuso'] = l['l'] * 1.25

Sp = 380e6 # resistência de prova mínima [MPa]

preCarg_Resist = 0.75 # porcentagem mínima para cargas dinâmicas
```

In [35]:
```
# COMPRIMENTO ROSCA

l['rosca'] = 2*D['furo']*1e2 + .25*2.54 # [cm]

l['s']     = l['parafuso'] - l['rosca']

l['t']     = l['l'] - l['s']

tabela(l, 'comprimento', 'cm')
```

Out[35]:

	l	parafuso	rosca	s	t	comprimento
0	6.0	7.5	2.2	5.3	0.7	cm

In [36]:
```
# RIGIDEZ DO PARAFUSO

Ab = (pi/4)*D['furo']**2

num1 = l['t']*1e-2 / (At*E['parafuso'])
num2 = l['s']*1e-2 / (Ab*E['parafuso'])

kb = ( num1 + num2 )**-1

tabela({'kb':kb*1e-9}, 'unidade', 'E9 N/m', arred=3)
```

Out[36]:

	kb	unidade
0	0.166	E9 N/m

In [37]:
```
# RIGIDEZ MATERIAL JUNTA

Am = pi*(D['eff']**2 - D['furo']**2)/4

num1 = espess['caixa'] / (Am*E['caixa'])
num2 = espess['base']  / (Am*E['base'])

km = (num1 + num2)**-1

tabela({'km':km*1e-9}, 'unidade', 'E9 N/m', arred=3)
```

Out[37]:

	km	unidade
0	1.86	E9 N/m

In [38]:
```
# FATOR DE RIGIDEZ DA JUNTA

C = kb / (km+kb)

tabela({'C':C}, 'unidade', 'adimensional', arred=3)
```

Out[38]:

	C	unidade
0	0.082	adimensional

In [39]:
```
# COEFICIENTES SEGURANÇA ESCOAMENTO

Sy = 420e6 # [Pa]


# CARGA PARA SEPARAR A JUNTA

P = {'p': F['max']}

tabela(P, 'P (sep junta)', 'N')
```

Out[39]:

	p	P (sep junta)
0	2000	N

In [40]:
```
# parâmetros obtidos -> Sp, At, kb, km, c

# PORÇOES DA FORÇA APLICADA

P['b'] = C     * P['p']
P['m'] = (1-C) * P['p']

tabela(P, 'força aplicada', 'N')
```

Out[40]:

	p	b	m	força aplicada
0	2000	163.7	1836.3	N

In [41]:
```
# PRÉ-CARGA

F['i'] = preCarg_Resist * Sp * At

tabela({'Fi':F['i']} , 'Força', 'N')
```

Out[41]:

	Fi	Força
0	10433.8	N

In [42]:
```
# CARGAS RESULTANTES (pós pré-carga)

F['b'] = F['i'] + P['b']
F['m'] = F['i'] + P['m']

tabela(F, 'Forças', 'N')
```

Out[42]:

	min	max	i	b	m	Forças
0	0	2000	10433.8	10597.5	12270.2	N

In [43]:
```
# COMPONETES MÉDIA E ALTERNADA

F['a_paraf'] = (F['b']-F['i']) / 2
F['m_paraf'] = (F['b']+F['i']) / 2

tabela(F, 'Forças', 'N')
```

Out[43]:

	min	max	i	b	m	a_paraf	m_paraf	Forças
0	0	2000	10433.8	10597.5	12270.2	81.8	10515.7	N

In [44]:
```
# TENSÕES MÉDIA/ALTERNADA PARAFUSO

sigma = {}

sigma['a_nom'] = F['a_paraf']/At
sigma['m_nom'] = F['m_paraf']/At

tabela(sigma, 'Tensão', 'MPa', mul=1e-6)
```

Out[44]:

	a_nom	m_nom	Tensão
0	2.2	287.2	MPa

In [45]:
```
# FATOR DE CONCENTRAÇÃO (TENSÃO FADIGA)

# parafuso roscas cortadas

kf = 2.8

print('\nkf * |tensao_máx| > Sy ->', (kf * (sigma['a_nom'] + sigma['
m_nom']) > Sy), '\n')

kfm = (Sy - kf*sigma['a_nom']) / (sigma['a_nom'] + sigma['m_nom'])

tabela({'kfm':kfm}, 'Fator concentração', 'adimensional')
```

```
kf * |tensao_máx| > Sy -> True
```

Out[45]:

	kfm	Fator concentração
0	1.4	adimensional

In [46]:
```
# TENSÕES REAIS MÉDIA/ALTERNADA

sigma['a'] = kf  * sigma['a_nom']
sigma['m'] = kfm * sigma['m_nom']

# TENSÃO PRÉ-CARGA INICIAL

sigma['i'] = kfm * F['i'] / At

tabela(sigma, 'Tensão', 'MPa', mul=1e-6)
```

Out[46]:

	a_nom	m_nom	a	m	i	Tensão
0	2.2	287.2	6.3	410.5	407.3	MPa

In [47]:
```
# LIMITE RESISTÊNCIA À FADIGA

Sut      = 520 # MPa

Se_linha = .5 * Sut

C_lim = {'carreg': .7,               # força normal
         'tam':     1,               # d = 8 mm
         'sup':     4.51*Sut**-.265, #
         'temp':    1,               # Temp < 450°C
         'conf':    .753}            # 99.9% confiabilidade

Se = Se_linha*C_lim['carreg']*C_lim['tam']*C_lim['sup']*C_lim['temp
']*C_lim['conf'] * 1e6

tabela({'Se':Se}, 'Limite de resistência', 'MPa', mul=1e-6)
```

Out[47]:

	Se	Limite de resistência
0	117.8	MPa

In [48]:
```
# COEFICIENTE SEGURANÇA Nf3 (Goodman) - FADIGA

N2 = {}

N2['f'] = Se*(Sut*1e6 - sigma['i']) / (Se*(sigma['m']-sigma['i']) +
Sut*1e6*sigma['a'])

# COEFICIENTE SEGURANÇA ESCOAMENTO

sigma['b'] = F['b']/At

N2['y'] = Sy/sigma['b']

# COEFICIENTE SEGURANÇA SEPARAÇÃO JUNTA

N2['separacao'] = F['i'] / (P['p']*(1-C))

tabela(N2, 'coefs seguranca', 'adimensional')
```

Out[48]:

	f	y	separacao	coefs seguranca
0	3.7	1.5	5.7	adimensional

In [49]:
```
tabela(sigma, 'Tensão', 'MPa', mul=1e-6)
```

Out[49]:

	a_nom	m_nom	a	m	i	b	Tensão
0	2.2	287.2	6.3	410.5	407.3	289.5	MPa

O projeto atende ao que foi pedido, pois a restrição era é que o **coeficiente de segurança de fadiga** não fosse **menor que 2**.