1

ECE 236B - HW 5

Peter Racioppo

103953689

T4.26, A12.6, A3.11, A3.13, T4.43, Q56

$\boxed{\text{T4.26}}$ ~~Verify~~ $x^T x \le yz$, $y \ge 0$, $z \ge 0$

if and only if $\left\| \begin{bmatrix} 2x \\ y-z \end{bmatrix} \right\|_2 \le y+z$, $y \ge 0$, $z \ge 0$.

Suppose $x^T x \le yz$

$$\left\| \begin{bmatrix} 2x \\ y-z \end{bmatrix} \right\|^2 = \begin{bmatrix} 2x^T & y^T - z^T \end{bmatrix} \begin{bmatrix} 2x \\ y-z \end{bmatrix} = 4x^T x + y^T y - y^T z - z^T y + z^T z$$

$$= 4x^T x + y^T y - 2y^T z + z^T z$$

$$= 4x^T x + y^2 + z^2 - 2yz \quad (y, z \text{ scalars})$$

$$\le 4yz - 2yz + y^2 + z^2$$

$$= 2yz + y^2 + z^2$$

$$= (y+z)^2$$

Taking the square root of both sides,

$\left\| \begin{bmatrix} 2x \\ y-z \end{bmatrix} \right\|_2 \le y+z$. Conversely, if this inequality holds, we have that:

$$4x^T x + y^2 + z^2 - 2yz \le (y+z)^2 = y^2 + z^2 + 2yz$$

$$\Rightarrow 4x^T x \le 4yz \Rightarrow x^T x \le yz.$$

(a.) maximize $\left(\sum_{i=1}^m 1/(a_i^T x + b_i) \right)^{-1}$, with domain $\{x \mid Ax > b\}$, where a_i^T is the ith row of A.

<u>Recall</u>: Given a proper cone K, $x \le_K y \Leftrightarrow y - x \in K$.

<u>Def: SOCP</u>

$$\min \ c^T x$$

$$\text{s.t.} \ -(A_i x + b_i, c_i^T x + d_i) \le_{K_i} 0, \ i = 1, \ldots, m$$

where $K_i = \{(y,t) \in \mathbb{R}^{n_i+1} \mid \|y\|_2 \le t\}$

$$\text{maximize } \left(\sum_{i=1}^{m} 1/(a_i^T x + b_i)\right)^{-1} \quad (1)$$

is equivalent to:

$$\begin{aligned} \max\ & 1/\mathbf{1}^T t \\ \text{s.t. } & 1/(a_i^T x + b_i) \le t_i \quad \forall i \in [1,m]. \quad (2) \\ & t \succeq 0. \end{aligned}$$

$$\Updownarrow$$

$$\begin{aligned} \min\ & \mathbf{1}^T t \\ \text{s.t. } & t_i(a_i^T x + b_i) \ge 1, \quad \forall i \in [1,m]. \\ & t \succeq 0 \end{aligned}$$

Let $x = 1$, $y_i = a_i^T x + b_i$, $z_i = t_i$. Then,
$t_i(a_i^T x + b_i) \ge 1 \Leftrightarrow \left\| \begin{bmatrix} 2 \\ a_i^T x + b_i - t_i \end{bmatrix} \right\|_2 \le a_i^T x + b_i + t_i$,
$a_i^T x + b_i \ge 0$, $t_i \ge 0$, $\forall i \in [1,m]$.

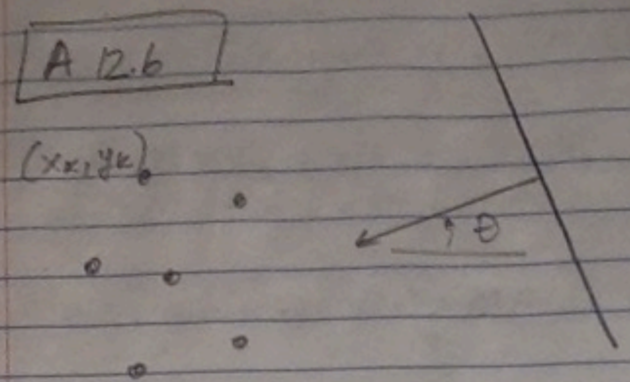
Let $K_i = \{ y \mid \|y\|_2 \le a_i^T x + b_i + t_i \}$.
This gives: $-\begin{bmatrix} 2 \\ a_i^T x + b_i - t_i \end{bmatrix} \preceq_{K_i} 0$.

In summary, we have:

$$\boxed{\begin{aligned} \min\ & \mathbf{1}^T t \\ \text{s.t. } & \left\| \begin{bmatrix} 2 \\ a_i^T x + b_i - t_i \end{bmatrix} \right\|_2 \le a_i^T x + b_i + t_i, \\ & a_i^T x + b_i \ge 0,\ t_i \ge 0,\ \forall i \in [1,m]. \\ & \Updownarrow \\ \min\ & \mathbf{1}^T t \\ \text{s.t. } & -\begin{bmatrix} 2 \\ a_i^T x + b_i - t_i \end{bmatrix} \preceq_{K_i} 0 \\ & a_i^T x + b_i \ge 0,\ t_i \ge 0,\ \forall i \in [1,m], \\ \text{where } & K_i = \{ y \mid \|y\|_2 \le a_i^T x + b_i + t_i \end{aligned}}$$

③

$\boxed{A\ 12.6}$



$$G(\theta) = \sum_{k=1}^{n} w_k e^{i(x_k \cos\theta + y_k \sin\theta)}$$

$$= \sum_{k=1}^{n} \left[w_{re,k} \cos(\gamma_k(\theta)) - w_{im,k} \sin(\gamma_k(\theta)) \right] + i\left[w_{re,k} \sin(\gamma_k(\theta)) + w_{im,k} \cos(\gamma_k(\theta)) \right],$$

$$\gamma_k(\theta) = x_k \cos\theta + y_k \sin\theta, \quad w_k = w_{re,k} + i w_{im,k}.$$

$$\min_{w_k} \left(\max_{n} \left\{ |G(\theta_N)| \;\middle|\; \underbrace{|\theta_n - \theta^{tar}| \geq \Delta}_{-\Delta \leq \theta_n - \theta^{tar} \leq \Delta} \right\} \right).$$

(a.) $\min \ t$

$\text{s.t.} \ |G(\theta_n)| \leq t, \ \forall n.$

Let $\gamma_{kn} = x_k \cos\theta_n + y_k \sin\theta_n$.

$$|G(\theta_n)| = \left| \sum_k w_{re} \cos\gamma_n - w_{im} \sin\gamma_n + i \sum_k w_{re} \sin\gamma_n + w_{im} \cos\gamma_n \right| \leq t$$

$$\Leftrightarrow \left(\sum_k w_{re} \cos\gamma_n - w_{im} \sin\gamma_n \right)^2 + \left(\sum_k w_{re} \sin\gamma_n + w_{im} \cos\gamma_n \right)^2 \leq t^2$$

$$\Leftrightarrow \boxed{\left\| \begin{matrix} \sum_k w_{re,k} \cos\gamma_n - w_{im,k} \sin\gamma_n \\ \sum_k w_{re,k} \sin\gamma_n + w_{im,k} \cos\gamma_n \end{matrix} \right\|_2 \leq t, \quad \forall n.}$$

12.6. Antenna array weight design.

We consider an array of n omnidirectional antennas in a plane, at positions (xk,yk), k = 1,...,n.

A unit plane wave with frequency ω is incident from an angle θ. This incident wave induces in the kth antenna element a (complex) signal exp(i(xk cos θ + yk sin θ − ωt)), where i = $\sqrt{-1}$. (For simplicity we assume that the spatial units are normalized so that the wave number is one, i.e., the wavelength is λ = 2π.) This signal is demodulated, i.e., multiplied by eiωt, to obtain the baseband signal (complex number) exp(i(xk cos θ + yk sin θ)). The baseband signals of the n antennas are combined linearly to form the output of the antenna array

The complex weights in the linear combination, wk =wre,k +iwim,k, k=1,...,n, are called the antenna array coefficients or shading coefficients, and will be the design variables in the problem. For a given set of weights, the combined output G(θ) is a function of the angle of arrival θ of the plane wave. The design problem is to select weights wi that achieve a desired directional pattern G(θ).

We now describe a basic weight design problem. We require unit gain in a target direction θtar, i.e., G(θtar) = 1. We want |G(θ)| small for |θ − θtar| ≥ Δ, where 2Δ is our beamwidth. To do this, we can minimize max |G(θ)|, |θ−θtar|≥Δ where the maximum is over all θ ∈ [−π, π] with |θ − θtar| ≥ Δ. This number is called the sidelobe level for the array; our goal is to minimize the sidelobe level. If we achieve a small sidelobe level, then the array is relatively insensitive to signals arriving from directions more than Δ away from the target direction. This results in the optimization problem minimize max|θ−θtar|≥Δ |G(θ)| subject to G(θtar) = 1, with w ∈ Cn as variables.

The objective function can be approximated by discretizing the angle of arrival with (say) N values (say,uniformlyspaced)θ1,...,θN over the interval [−π,π],and replacing the objective with max{|G(θk)| | |θk − θtar| ≥ Δ}

In [1]:
```
import numpy as np
import cvxpy as cp
import matplotlib.pyplot as plt
```

In [73]:
```
n = 40
N = 400
theta_tar = 15*(np.pi/180)
Delta = 15*(np.pi/180)

X = 30*np.random.uniform(low=0,high=1,size=n)
Y = 30*np.random.uniform(low=0,high=1,size=n)

# Theta = np.linspace(start=-np.pi,stop=np.pi,num=N)
# Exclude theta's less than delta from theta_tar:
ratio = np.round(np.abs(-180-15)/(np.abs(180-15)+np.abs(-180-15))*1000)/1000 # (Ratio of number of poin
Theta1 = np.linspace(start=-np.pi,stop=theta_tar-Delta,num=round(N*ratio))
Theta2 = np.linspace(start=theta_tar+Delta,stop=np.pi,num=round(N*(1-ratio)))
Theta = np.concatenate((Theta1,Theta2))
```

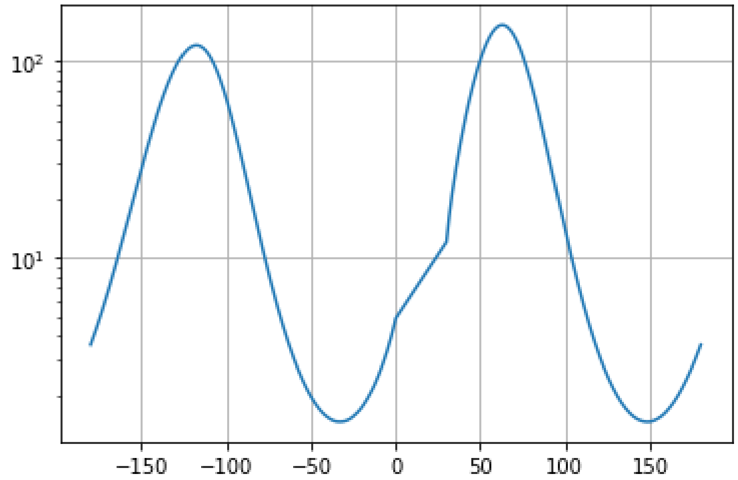
In [18]:
```
## Failed attempts:

# Gamma_kn = np.outer(X,np.cos(Theta)) + np.outer(Y,np.sin(Theta))
# gamma_tar = X*np.cos(theta_tar) + Y*np.sin(theta_tar)

# # gamma = np.matrix(Gamma_kn[:,1])
# # Gamma_kn_cos = np.cos(Gamma_kn)
# # Gamma_kn_sin = np.sin(Gamma_kn)

# # np.shape(np.cos(gamma))
# # np.dot(w,np.cos(gamma)) + np.dot(w,np.sin(gamma))
# # cos_gamma = np.matrix(Gamma_kn_cos[:,1])
# # np.shape(cos_gamma)

# w_re = cp.Variable(n)
# w_im = cp.Variable(n)
# t = cp.Variable(1)


# constr = []
# for i in np.arange(N):
#     theta = Theta[i]
#     gamma = X*np.cos(theta) + Y*np.sin(theta)
#     term1 = cp.sum(np.cos(gamma)*w_re - np.sin(gamma)*w_im)
#     term2 = cp.sum(np.sin(gamma)*w_re + np.cos(gamma)*w_im)
#     constr += [cp.norm(term1,term2) <= t]
#     constr += [theta - theta_tar <= Delta, -(theta - theta_tar) <= Delta]

# term1 = cp.sum(np.cos(gamma_tar)*w_re - np.sin(gamma_tar)*w_im)
# term2 = cp.sum(np.sin(gamma_tar)*w_re + np.cos(gamma_tar)*w_im)
# constr += [term1 == 1,term2 == 0]
# problem = cp.Problem(cp.Minimize(t), constr)
# problem.solve()

## ------------------------

# w_re = cp.Variable(n)
# w_im = cp.Variable(n)
# t = cp.Variable(1)

# constr = []
# for i in np.arange(N):
```

```
#     theta = Theta[i]
#     gamma = np.matrix(Gamma_kn[:,i])
#     cos_gamma = Gamma_kn_cos[:,i].T
#     sin_gamma = Gamma_kn_sin[:,i].T
#     term1 = cp.sum(cp.multiply(cos_gamma,w_re) - cp.multiply(sin_gamma,w_im))
#     term2 = cp.sum(cp.multiply(sin_gamma,w_re) + cp.multiply(cos_gamma,w_im))

#     constr += [cp.sum(cp.multiply(term1,term1)) + cp.sum(cp.multiply(term2,term2)) <= t**2]
#     constr += [theta - theta_tar <= Delta]
#     constr += [-(theta - theta_tar) <= Delta]

# term1 = cp.sum(cp.multiply(np.cos(gamma_tar),w_re) - cp.multiply(np.sin(gamma_tar),w_im))
# term2 = cp.sum(cp.multiply(np.sin(gamma_tar),w_re) + cp.multiply(np.cos(gamma_tar),w_im))
# constr += [term1 == 1,term2 == 0]
# problem = cp.Problem(cp.Minimize(t), constr)
# problem.solve(solver=cp.ECOS)
```

In [*]:
```
# Attempt 3:

w_re = cp.Variable(n)
w_im = cp.Variable(n)
t = cp.Variable(1,pos=True)

constr = []
for i in np.arange(N):
    theta = Theta[i]
    gamma = X*np.cos(theta) + Y*np.sin(theta)
    term1 = 0
    term2 = 0
    for j in np.arange(n):
        term1 += np.cos(gamma[j])*w_re[j] - np.sin(gamma[j])*w_im[j]
        term2 += np.sin(gamma[j])*w_re[j] + np.cos(gamma[j])*w_im[j]

    constr += [cp.SOC(term1 + term2,t)] # I think this should actually be reversed, but this doesn't wo

term1 = 0
term2 = 0
for j in np.arange(n):
    term1 += np.cos(gamma_tar[j])*w_re[j] - np.sin(gamma_tar[j])*w_im[j]
    term2 += np.sin(gamma_tar[j])*w_re[j] + np.cos(gamma_tar[j])*w_im[j]
constr += [term1 == 1,term2 == 0]

problem = cp.Problem(cp.Minimize(t), constr)
problem.solve()
```

In [95]:
```
w_re_opt = w_re.value
w_im_opt = w_im.value

G = np.zeros(N)
for i in np.arange(N):
    theta = Theta[i]
    gamma = X*np.cos(theta) + Y*np.sin(theta)
    term1 = 0
    term2 = 0
    for j in np.arange(n):
        term1 += np.cos(gamma[j])*w_re_opt[j] - np.sin(gamma[j])*w_im_opt[j]
        term2 += np.sin(gamma[j])*w_re_opt[j] + np.cos(gamma[j])*w_im_opt[j]
    G[i] = np.sqrt(term1**2+term2**2)
```

In [99]:
```
plt.plot(Theta*(180/np.pi),G)
plt.grid()
plt.yscale('log')
plt.show()
```



In [ ]:
```
# (Obviously wrong)
```

(4)

Formulate as SDP:
minimize $c^T x$
s.t. $x_1F_1 + \dots + x_nF_n + G \preceq 0$

$\boxed{\text{A3.11 (a,b,c)}}$

$F(x) = F_0 + x_1F_1 + \dots + x_nF_n \quad , x \in \mathbb{R}^n, F_i \in S^m.$
$\mathrm{dom}(f) = \{x \in \mathbb{R}^n \mid F(x) \succ 0\}.$

(a.) minimize $f(x) = c^T F(x)^{-1} c$, where $c \in \mathbb{R}^m$.

$$\Updownarrow$$

$$\begin{aligned} \min \quad & t \\ \text{s.t.} \quad & c^T F(x)^{-1} c \leq t \\ & F(x) \succ 0 \end{aligned}$$

$$\Updownarrow$$

$$\begin{aligned} \min \quad & t \\ \text{s.t} \quad & \begin{bmatrix} F(x) & c \\ c^T & t \end{bmatrix} \succeq 0 \\ & F(x) \succ 0 \end{aligned}$$

$S = t - c^T F^{-1} c \geq 0$
$\Leftrightarrow c^T F^{-1} c \leq t.$

$$-\begin{bmatrix} F(x) & c \\ c^T & t \end{bmatrix} = \begin{bmatrix} -F_0 & -c \\ -c^T & 0 \end{bmatrix} + x_1 \begin{bmatrix} -F_1 & 0 \\ 0 & 0 \end{bmatrix} + \dots + x_n \begin{bmatrix} -F_n & 0 \\ 0 & 0 \end{bmatrix} + t \begin{bmatrix} 0 & 0 \\ 0 & -1 \end{bmatrix} \preceq 0$$

$$\boxed{\begin{aligned} & \text{minimize} \quad t \\ & \text{s.t.} \\ & \begin{bmatrix} -F_0 & -c \\ -c^T & 0 \end{bmatrix} + x_1 \begin{bmatrix} -F_0 & 0 \\ 0 & 0 \end{bmatrix} + \dots + x_n \begin{bmatrix} -F_n & 0 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & -1 \end{bmatrix} \preceq 0 \\ & F(x) \succ 0 \end{aligned}}$$

(5)

(b.) Minimize $f(x) = \max_{1 \le i \le k} c_i^T F(x)^{-1} c_i$,

where $c_i \in \mathbb{R}^m$, $i = 1, \ldots, k$.

$$\Updownarrow$$

$$\begin{aligned} \min \quad & t \\ \text{s.t.} \quad & \max_i c_i^T F(x)^{-1} c_i \le t \end{aligned}$$

$$\Updownarrow$$

$$\begin{aligned} \min \quad & t \\ \text{s.t.} \quad & c_i^T F(x)^{-1} c_i \le t, \ \forall i \end{aligned}$$

$$\Updownarrow$$

$$\begin{aligned} \min \quad & t \\ \text{s.t.} \quad & \begin{bmatrix} F(x) & c_i \\ c_i^T & t \end{bmatrix} \succeq 0, \ \forall i \end{aligned}$$

Now $A \succ 0$ and $B \succ 0$ if and only if $\begin{bmatrix} A & 0 \\ 0 & B \end{bmatrix} \succ 0$. So we can write:

$$\boxed{\begin{aligned} \min \quad & t \\ \text{s.t.} \quad & \begin{bmatrix} A_1 & & 0 \\ & \ddots & \\ 0 & & A_n \end{bmatrix} \succeq 0 \\ & \\ \text{where} \quad & A_i = \begin{bmatrix} F(x) & c_i \\ c_i^T & t \end{bmatrix}. \end{aligned}}$$

⑥

(c.) minimize $f(x) = \sup_{\|c\|_2 \leq 1} c^T F(x)^{-1} c$

$$\Updownarrow$$

$$\begin{aligned} &\min \ t \\ &\text{s.t.} \ \sup_{\|c\|_2 \leq 1} c^T F(x)^{-1} c \leq t \end{aligned}$$

$$\Updownarrow$$

$$\begin{aligned} &\min \ t \\ &\text{s.t.} \ c^T F(x)^{-1} c \leq t \\ &\quad \text{for } \forall \ c \text{ s.t. } \|c\|_2 \leq 1. \end{aligned}$$

$$\Updownarrow$$

$$\begin{aligned} &\min \ t \\ &\text{s.t.} \ \begin{bmatrix} F(x) & c \\ c^T & t \end{bmatrix} \succeq 0 \\ &\qquad \|c\|_2 \leq 1 \end{aligned}$$

$$\begin{bmatrix} F(x) & c \\ c^T & t \end{bmatrix} = \begin{bmatrix} F_0 & 0 \\ 0 & 0 \end{bmatrix} + x_1 \begin{bmatrix} F_1 & 0 \\ 0 & 0 \end{bmatrix} + \cdots + x_n \begin{bmatrix} F_n & 0 \\ 0 & 0 \end{bmatrix} + t \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}$$
$$+ c_1 \begin{bmatrix} 0 & \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \\ [1\ 0\ 0 \cdots 0] & 0 \end{bmatrix} + \cdots + c_n \begin{bmatrix} 0 & \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \\ [0 \cdots 0\ 1] & 0 \end{bmatrix}$$

Let $X = \begin{bmatrix} x \\ c \\ t \end{bmatrix}$ and $C = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}$, and $D = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$

We can then write:

$$\begin{aligned} &\min \ C^T X \\ &\text{s.t.} \ \begin{bmatrix} F(x) & c \\ c^T & t \end{bmatrix} \succeq 0 \\ &\qquad \|D^T X\|_2 \leq 1. \end{aligned}$$

But this isn't an SDP because of the norm constraint.

⑦

(c.) minimize $f(x) = \sup_{\|c\|_2 \leq 1} c^T F(x)^{-1} c$

$f(x) = \lambda_{max}(F(x)^{-1})$

$$\begin{aligned} &\min \ t \\ &\text{s.t } \lambda_{max}(F(x)^{-1}) \leq t \end{aligned}$$

$\lambda_{max}(F(x)^{-1}) \leq t \Rightarrow (F(x)^{-1} - tI) c \leq 0 \quad \forall c$

$\Rightarrow c^T F(x)^{-1} c \leq tI\, c^T c \leq t$ (since $c^T c \leq 1$).

(since $F(x) \succ 0 \Rightarrow F(x)^{-1} \succ 0$).

$\Rightarrow c^T (F(x)^{-1} - tI) c \leq 0.$

$e_i^T F(x)^{-1} e^i \leq t \quad \forall i$

$\Rightarrow I^T F(x)^{-1} I \leq tI$

$$\Rightarrow \begin{bmatrix} F(x) & I \\ I & tI \end{bmatrix} \succeq 0$$

We have:

$$\begin{aligned} &\min \ t \\ &\text{s.t } \begin{bmatrix} F(x) & I \\ I & tI \end{bmatrix} \succeq 0. \end{aligned}$$

⑧

$\boxed{\text{A 3.13}}$ $H(A,B) = 2(A^{-1}+B^{-1})^{-1}$.

Show that $X = \frac{1}{2}H(A,B)$ solves the SDP

$$\max \ \mathrm{tr}\, X$$
$$\text{s.t.} \begin{bmatrix} X & X \\ X & X \end{bmatrix} \preceq \begin{bmatrix} A & 0 \\ 0 & B \end{bmatrix},$$

$X \in S^n$, $A \in S^n_{++}$, $B \in S^n_{++}$.

Let $R = \begin{bmatrix} A^{-1} & I \\ B^{-1} & -I \end{bmatrix}$.

$A, B \in S^n_{++} \Rightarrow A^{-1}, B^{-1} \in S^n_{++}$
$\Rightarrow \mathrm{rank}(A^{-1}) = \mathrm{rank}(B^{-1}) = n$.

We need to show that every row of $[A^{-1}\ I]$ and $[B^{-1}\ -I]$ are linearly independent.

Suppose $v \in \mathrm{Null}(R)$, $v = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}$.

$\Rightarrow \begin{bmatrix} A^{-1} & I \\ B^{-1} & -I \end{bmatrix}\begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = 0 \Rightarrow \begin{bmatrix} A^{-1}v_1 + Iv_2 \\ B^{-1}v_1 - Iv_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$?

Suppose row i of $[A^{-1}\ I]$ is linearly dependent on row i of $[B^{-1}\ -I]$.

$\Rightarrow -[A^{-1}\ I]_i = [B^{-1}\ -I]_i \Rightarrow -A^{-1}_i = B^{-1}_i$

$\Rightarrow A^{-1}_i + B^{-1}_i = 0$ for some i.

$\Rightarrow (v^T(A^{-1}+B^{-1})v)_i = (v^TA^{-1}v)_i + (v^TB^{-1}v)_i = 0$
$\Rightarrow A^{-1} \preceq 0$ or $B^{-1} \preceq 0$, a contradiction.

9

Thus, $[A^{-1}\ I]$ and $[B^{-1}\ -I]$ are linearly independent and hence $\text{rank}(R)=n$, so R is nonsingular. Thus,

$$R^T\begin{bmatrix} X & X \\ X & X \end{bmatrix} R \preceq R^T \begin{bmatrix} A & 0 \\ 0 & B \end{bmatrix} R$$

$$\begin{bmatrix} X & X \\ X & X \end{bmatrix}\begin{bmatrix} A^{-1} & I \\ B^{-1} & -I \end{bmatrix} = \begin{bmatrix} X(A^{-1}+B^{-1}) & 0 \\ X(A^{-1}+B^{-1}) & 0 \end{bmatrix}$$

Note: $B^{-T}=B^{-1}$.

$$X(A+B)^{-1}\begin{bmatrix} A^{-1} & B^{-1} \\ 0 & -I \end{bmatrix}\begin{bmatrix} I & 0 \\ I & 0 \end{bmatrix} = X(A^{-1}+B^{-1})\begin{bmatrix} A^{-1}+B^{-1} & 0 \\ -I & 0 \end{bmatrix}$$

$$\begin{bmatrix} A & 0 \\ 0 & B \end{bmatrix}\begin{bmatrix} A^{-1} & I \\ B^{-1} & -I \end{bmatrix} = \begin{bmatrix} I & A \\ I & -B \end{bmatrix}$$

$$\begin{bmatrix} A^{-1} & B^{-1} \\ 0 & -I \end{bmatrix}\begin{bmatrix} I & A \\ I & -B \end{bmatrix} = \begin{bmatrix} A^{-1}+B^{-1} & 0 \\ -I & B \end{bmatrix}$$

So we have:

$$X(A^{-1}+B^{-1})\begin{bmatrix} A^{-1}+B^{-1} & 0 \\ -I & 0 \end{bmatrix} \preceq \begin{bmatrix} A^{-1}+B^{-1} & 0 \\ -I & B \end{bmatrix}$$

Letting $X = \frac{1}{2}H(A,B) = (A^{-1}+B^{-1})^{-1}$, we have:

$$\begin{bmatrix} A^{-1}+B^{-1} & 0 \\ -I & 0 \end{bmatrix} \preceq \begin{bmatrix} A^{-1}+B^{-1} & 0 \\ -I & B \end{bmatrix},$$

which is true since $B \succeq 0$.

The constraint is satisfied with equality for all other elements, which implies that X is optimal.

(10) Suppose $\exists$ Y s.t. $tr(Y) > tr(X)$
and $\begin{bmatrix} Y & Y \\ Y & Y \end{bmatrix} \preccurlyeq \begin{bmatrix} A & 0 \\ 0 & B \end{bmatrix}$.

WLOG, let $Y = X + Q_i$ where $Q_i = \begin{bmatrix} 0 & & \\ & \varepsilon & \\ & & 0 \end{bmatrix}$,

That is, any matrix with $tr(Q_i) = \varepsilon$
where $\varepsilon > 0$, so that $tr(Y) = tr(X) + \varepsilon$.

Then, we have that:

$$Y(A^{-1} + B^{-1}) = I + \varepsilon(A^{-1} + B^{-1}) \Rightarrow$$

$$\begin{bmatrix} A^{-1} + B^{-1} & 0 \\ -I & 0 \end{bmatrix} + \varepsilon \begin{bmatrix} (A^{-1} + B^{-1})^2 & 0 \\ -(A^{-1} + B^{-1}) & 0 \end{bmatrix} \preccurlyeq \begin{bmatrix} A^{-1} + B^{-1} & 0 \\ -I & B \end{bmatrix}$$

$$\Leftrightarrow \varepsilon \begin{bmatrix} (A^{-1} + B^{-1})^2 & 0 \\ -(A^{-1} + B^{-1}) & 0 \end{bmatrix} \preccurlyeq \begin{bmatrix} 0 & 0 \\ 0 & B \end{bmatrix}$$

But $\varepsilon(A^{-1} + B^{-1})^2 \not\preccurlyeq 0$ for $\forall$ $\varepsilon > 0$
since $A^{-1}, B^{-1} \in S^n_{++}$, a contradiction.

Thus, X is the solution to the SDP.

It follows that $tr((A^{-1} + B^{-1})^{-1})$ is
a concave function of (A, B)
because $tr(X)$ is a concave function of
X for $\forall$ X satisfying the given
linear matrix inequality.

(11)

$\boxed{\text{T 4.43 (b,c)}}$ Suppose $A: \mathbb{R}^n \to S^m$ is affine,

$A(x) = A_0 + x_1 A_1 + \dots + x_n A_n$, $A_i \in S^m$

Let $\lambda_1(x) \geq \lambda_2(x) \geq \dots \geq \lambda_m(x) = \text{eigen}(A)$.

Pose as SDP:

(b.) $\min (\lambda_1(x) - \lambda_m(x))$

$\lambda_{max} = \sup_{\|c\|_2 \leq 1} c^T A(x) c$ ... But no simple expression for $\lambda_{min}(x)$.

$$\min_x (\lambda_1(x) - \lambda_m(x))$$

$$= \min_x \left(\max_{i,j} (\lambda_i(x) - \lambda_j(x)) \right)$$

$$\Leftrightarrow \min_{x,t} \; t$$

$$\text{s.t. } \max_{i,j} (\lambda_i(x) - \lambda_j(x)) \leq t$$

$$\Leftrightarrow \min_{x,t} \; t$$

$$\text{s.t. } \lambda_i(x) - \lambda_j(x) \leq t \; , \forall \, i,j$$

~~Let v_i be such that $A(x) v_i = \lambda_i v_i$~~

~~$\lambda_i(x) v_i - \lambda_j(x) v_j = A(x) v_i - A(x) v_j = A(x)(v_i - v_j)$~~

~~Consider $A(x) - I\lambda_m$. $(A(x) - I\lambda_m) v_m = 0$~~

~~$\min t$~~

~~s.t. $\lambda_{max} \leq t + \lambda_{min}$~~

⑫

I 4.43 (b) (contd).

Consider $\lambda I - A(x)$. Let $A(x) = U^T \Lambda U$, $U^T U = I$

$\Rightarrow \lambda I - A = U^T(\lambda I - \Lambda) U$. $\lambda I - A \succeq 0 \Leftrightarrow \lambda I - \Lambda \succeq 0$

$\lambda I - \Lambda \succeq 0 \quad \Leftrightarrow \quad \lambda \geq \lambda_{max}(\Lambda) = \lambda_{max}(A)$

$\Leftrightarrow \quad \lambda_{max}(A) = \inf_{\lambda I - A \succeq 0} \lambda \quad \Leftrightarrow \lambda_{max}(A) = -\sup_{\lambda I - A \succeq 0}(-\lambda)$.

Consider $A(x) - \lambda I = U^T(\Lambda - \lambda' I) U$

$\Lambda - \lambda' I \succeq 0 \quad \Leftrightarrow \quad \lambda_{min}(A) = \lambda_{min}(\Lambda) \geq \lambda'$

$\Leftrightarrow \quad \lambda_{min} = \sup_{A - \lambda' I \succeq 0} \lambda' \quad \Leftrightarrow \quad \lambda_{min} = -\inf_{\lambda' I - A \preceq 0}(-\lambda')$

Thus, $\min_x (\lambda_{max}(A) - \lambda_{min}(A)) = \min_x \inf_{\substack{\lambda, \lambda' \\ \lambda' I - A \preceq 0 \preceq \lambda I - A}} \lambda - \lambda'$

$$= \boxed{\begin{array}{ll} \min_{x, \lambda, \lambda'} & \lambda - \lambda' \\ \text{s.t.} & \lambda' I - A(x) \preceq 0 \\ & \lambda I - A(x) \succeq 0 \end{array}}$$

(c.) $\kappa(A(x)) = \lambda_1(x)/\lambda_m(x)$, $\text{dom}\{x \mid A(x) \succ 0\}$.

$A(x) \succ 0$ for at least one x.

Make change of variables $y = x/\gamma$, $t = \lambda/\gamma$, $s = 1/\gamma$ ($\Rightarrow y = sx$)

$\min \lambda_1(x)/\lambda_m(x)$ s.t. $A(x) \succeq 0$.

$$\begin{array}{l} \min \lambda/\gamma \\ \text{s.t. } \gamma I - A \preceq 0 \preceq \lambda I - A \end{array} \quad \Leftrightarrow \quad \begin{array}{l} \min t \\ \text{s.t. } I - sA \preceq 0 \preceq tI - sA \end{array}$$

$$\Leftrightarrow \begin{array}{l} \min t \\ \text{s.t. } I \preceq sA \preceq tI,\ s > 0 \end{array} \quad \Leftrightarrow \quad \begin{array}{l} \min t \\ \text{s.t. } I \preceq s(A_0 + x_1 A_1 + \cdots + x_n A_n) \preceq tI,\ s > 0 \end{array}$$

$$\Leftrightarrow \begin{array}{l} \min t \\ \text{s.t. } I \preceq sA_0 + y_1 A_1 + \cdots + y_n A_n \preceq tI,\ -Is \preceq 0 \end{array}$$

13

There are 3 constraints, so we can rewrite them using 3×3 matrices.

$$s\begin{bmatrix} -I & & 0 \\ & A_0 & \\ 0 & & -A_0 \end{bmatrix} + y_1\begin{bmatrix} 0 & & 0 \\ & A_1 & \\ 0 & & -A_1 \end{bmatrix} + \cdots + y_n\begin{bmatrix} 0 & & 0 \\ & A_n & \\ 0 & & -A_n \end{bmatrix}$$

$$+\begin{bmatrix} 0 & & 0 \\ & 0 & \\ 0 & & I \end{bmatrix} + t\begin{bmatrix} 0 & & 0 \\ & -I & \\ 0 & & 0 \end{bmatrix} \preceq \underline{0}$$

The problem is now an SDP.

$\boxed{6.}$ <u>GP</u>: $\min f_0(x)$ s.t. $f_i(x) \le 1$, $h_i(x) = 1$,

$f_0, \dots, f_m$ posynomial & $h_1, \dots, h_p$ monomial.

That is, $h_i(x) = c x_1^{a_1} x_2^{a_2} \cdots x_n^{a_n}$, $c > 0$, $a_i \in \mathbb{R}$,

$f_i(x) = \sum c_k x_1^{a_{1k}} \cdots x_n^{a_{nk}}$, $c_k > 0$.

---

minimize $\left(A + \mu T = \sum w_i + \mu \max(T_i)\right)$.

s.t. $T_1 = \rho k_0 \left[(w_3 + C_{L1})\left(\frac{1}{w_1} + \frac{1}{w_2} + \frac{1}{w_3}\right) + w_2\left(\frac{1}{w_1} + \frac{1}{w_2}\right) + (w_1 + w_4 + w_5 + w_6 + C_{L2} + C_{L3}) \frac{1}{w_1} \right]$.

$$T_2 = \rho k_0 \left[(w_5 + C_{L2})\left(\frac{1}{w_1} + \frac{1}{w_4} + \frac{1}{w_5}\right) + w_4\left(\frac{1}{w_1} + \frac{1}{w_4}\right) + (w_6 + C_{L3})\left(\frac{1}{w_1} + \frac{1}{w_4}\right) + (w_1 + w_2 + w_3 + C_{L1})\frac{1}{w_1} \right]$$

$$T_3 = \rho k_0 \left[(w_6 + C_{L3})\left(\frac{1}{w_1} + \frac{1}{w_4} + \frac{1}{w_6}\right) + w_4\left(\frac{1}{w_1} + \frac{1}{w_4}\right) + (w_1 + w_2 + w_3 + C_{L1})\frac{1}{w_1} + (w_5 + C_{L2})\left(\frac{1}{w_1} + \frac{1}{w_4}\right) \right].$$

$$T_j = \sum_i C_i R_{ij} = \rho k_0 \sum_i w_i R_{ij}$$

$$\min \left(\sum w_i + \mu \max(T_i) \right)$$

$\Leftrightarrow \min\left(\sum w_i + \mu t\right)$ s.t. $\max(T_i) \le t$ $\Leftrightarrow \min\left(\sum w_i + \mu t\right)$ s.t. $T_i \le t, \forall i$.

$T_i \le t \Leftrightarrow \frac{1}{t} T_i \le 1$ since $t > 0$.

Each T_i restraint is clearly a posynomial inequality of the form $f_i(x) \le 1$. The objective is also a posynomial. Thus, the scalarized problem is a GP.

6. Interconnect sizing. We consider the problem of sizing the interconnecting wires of the simple circuit shown below, in which one voltage source drives three different capacitive loads Cload1, Cload2, and Cload3.

We divide the wires into 6 segments of fixed length li; the optimization variables in the problem will be the widths wi of the segments. (The height of the wires is related to the particular integrated circuit technology process, and is fixed.) We take the lengths li to be one, for simplicity.

In the next figure each of the wire segments is modeled by a simple RC circuit, with the resistance inversely proportional to the width of the segment and the capacitance proportional to the width.

The capacitance and resistance of the ith segment is thus Ci = k0wi, Ri = ρ/wi, i = 1,...,6, where k0 and ρ are positive constants, which we take to be one for simplicity. We also take Cload1 = 1.5, Cload2 = 1, and Cload3 = 5.

We are interested in the trade-off between area and delay. The total area used by the wires is the sum of the wi's

We use the Elmore delay to model the delay from the source to each of the loads. The Elmore delays to loads 1, 2, and 3 are:

T1 = (C3 + Cload1)*(R1 + R2 + R3) + C2*(R1 + R2) + (C1 + C4 + C5 + C6 + Cload2 + Cload3)*R1 T2 = (C5 + Cload2)*(R1 + R4 + R5) + C4*(R1 + R4) + (C6 + Cload3)*(R1 + R4) + (C1 + C2 + C3 + Cload1)*R1 T3 = (C6 + Cload3)*(R1 + R4 + R6) + C4*(R1 + R4) + (C1 + C2 + C3 + Cload1)*R1 + (C5 + Cload2)*(R1 + R4) (The general rule is as follows: the Elmore delay from the source to node j is given by Sum of CiRij all nodes i where Ci is the capacitance at node i and Rij is the sum of the resistances on the intersection of the path from the source to node i and the path from the source to node j.) Our main interest is in the maximum of these delays, T = max {T1, T2, T3}.

We also impose minimum and maximum allowable values for the wire widths: Wmin ≤ wi ≤ Wmax, i = 1,...,6. For our specific problem, we take Wmin = 0.1 and Wmax = 10. We compare two choices of wire widths.

In [34]:
```
import numpy as np
import cvxpy as cp
import matplotlib.pyplot as plt
```

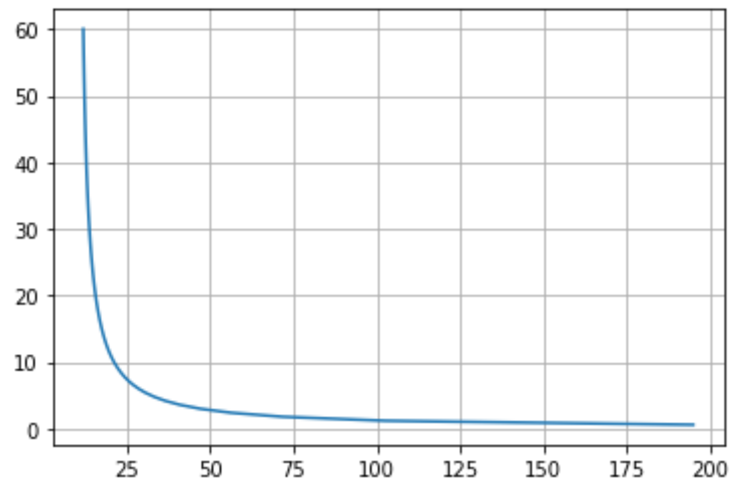
In [118]:
```
Cload1 = 1.5
Cload2 = 1
Cload3 = 5
W_min = 0.1
W_max = 10
N = 100
w_v = np.linspace(start=W_min,stop=W_max,num=N)
A_v = 6*w_v
```

(a) Equal wire widths. Plot the values of area A versus delay T , obtained if you take equal wire widths wi (varying between Wmin and Wmax).

In [119]:
```
T_v = np.zeros(N)
for i in np.arange(N):
    w = w_v[i]
    C1 = C2 = C3 = C4 = C5 = C6 = w
    R1 = R2 = R3 = R4 = R5 = R6 = 1/w
    T1 = (C3 + Cload1)*(R1 + R2 + R3) + C2*(R1 + R2) + (C1 + C4 + C5 + C6 + Cload2 + Cload3)*R1
    T2 = (C5 + Cload2)*(R1 + R4 + R5) + C4*(R1 + R4) + (C6 + Cload3)*(R1 + R4) + (C1 + C2 + C3 + Cload1
    T3 = (C6 + Cload3)*(R1 + R4 + R6) + C4*(R1 + R4) + (C1 + C2 + C3 + Cload1)*R1 + (C5 + Cload2)*(R1 +
    T = np.max([T1,T2,T3])
    T_v[i] = T
```

Area vs Delay:

In [120]:
```
plt.plot(T_v,A_v)
plt.grid()
plt.show()
```



(b) Optimal wire widths. The optimal area-delay trade-off curve can be computed by scalarization, i.e., by minimizing A + μT , subject to the constraints on w, for a large number of different positive values of μ. Verify that the scalarized problem is a geometric program (GP). For the specific problem parameters given, compute the area-delay trade-off curve using CVX or CVXPY. You can choose the values of μ logarithmically spaced between 10^-3 and 10^3. Compare the optimal trade-off curve with the one obtained in part (a).

Consult chapter 7 of the CVX user guide for details on how to solve GPs. For reasons explained in the user guide, CVX is not very fast when solving GPs. If needed, you can limit the number of weights μ, for example, to 10 or 20.

In [86]:
```
# Set up the problem:

N = 21
mu_v = np.geomspace(start=1e-3,stop=1e3,num=N)
w = cp.Variable(shape=(6,), pos=True, name="w")
t = cp.Variable(1,pos=True)

T1 = (w[2] + Cload1)*(w[0]**-1 + w[1]**-1 + w[2]**-1) + w[1]*(w[0]**-1 + w[1]**-1) + (w[0]+w[3]+w[4]+w[5
T2 = (w[4]+Cload2)*(w[0]+w[3]+w[4]) + w[3]*(w[0]**-1+w[3]**-1) + (w[5] + Cload3)*(w[0]**-1 + w[3]**-1) +
T3 = (w[5]+Cload3)*(w[0]**-1+w[3]**-1+w[5]**-1) + w[3]*(w[0]**-1+w[3]**-1) + (w[0]+w[1]+w[2]+Cload1)*w[0
T1 /= t
T2 /= t
T3 /= t
print("T1:", T1.log_log_curvature)
print("T2:", T2.log_log_curvature)
print("T3:", T3.log_log_curvature)
```

```
T1: LOG-LOG CONVEX
T2: LOG-LOG CONVEX
T3: LOG-LOG CONVEX
```

In [87]:
```
# Solve a single instance:

mu = 1
objective_fn = cp.sum(w) + mu*t
constraints = [T1 <= 1, T2 <= 1, T3 <= 1]
assert objective_fn.is_log_log_convex()
assert all(constraint.is_dgp() for constraint in constraints)
problem = cp.Problem(cp.Minimize(objective_fn), constraints)

print(problem)
print("Is this problem DGP?", problem.is_dgp())
```

```
minimize Sum(w, None, False) + 1.0 * var151138
subject to ((w[2] + 1.5) * (power(w[0], -1) + power(w[1], -1) + power(w[2], -1)) + w[1] * (power(w[0], -1) + power(w[1], -1)) + (w[0] + w[3] + w[4] + w[5] + 1.0 + 5.0) * power(w[0], -1)) / var151138 <= 1.0
           ((w[4] + 1.0) * (w[0] + w[3] + w[4]) + w[3] * (power(w[0], -1) + power(w[3], -1)) + (w[5] + 5.0) * (power(w[0], -1) + power(w[3], -1)) + (w[0] + w[1] + w[2] + 1.5) * power(w[0], -1)) / var151138 <= 1.0
           ((w[5] + 5.0) * (power(w[0], -1) + power(w[3], -1) + power(w[5], -1)) + w[3] * (power(w[0], -1) + power(w[3], -1)) + (w[0] + w[1] + w[2] + 1.5) * power(w[0], -1) + (w[4] + 1.0) * (power(w[0], -1) + power(w[3], -1))) / var151138 <= 1.0
Is this problem DGP? True
```

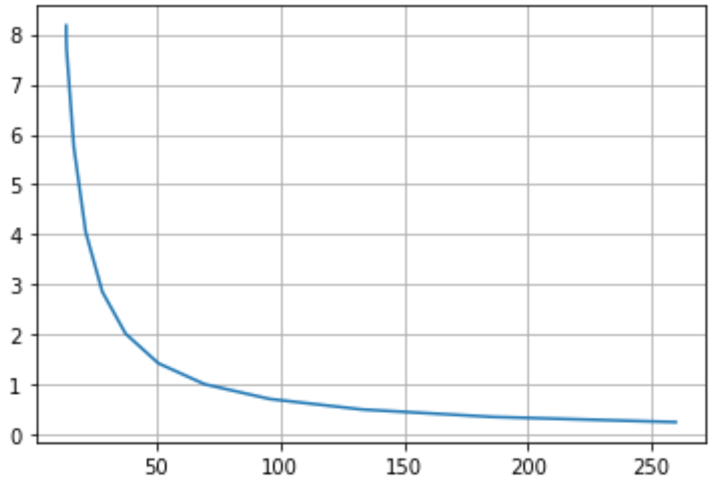
In [88]:
```
problem.solve(gp=True)
print("Optimal value:", problem.value)
print(w, ":", w.value)
print(t, ":", t.value)
print("Dual values: ", list(c.dual_value for c in constraints))
```

```
Optimal value: 21.486547050628886
w : [3.26297989e+00 5.36887644e-01 4.19543272e-01 2.03482520e+00
 3.65756051e-08 1.42861434e+00]
var151138 : [13.80369632]
Dual values:  [array([0.12603657]), array([0.0702077]), array([0.44619011])]
```

In [102]:
```
def f_T(w,Cload1,Cload2,Cload3):
    T1 = (w[2] + Cload1)*(w[0]**-1 + w[1]**-1 + w[2]**-1) + w[1]*(w[0]**-1 + w[1]**-1) + (w[0]+w[3]+w[4
    T2 = (w[4]+Cload2)*(w[0]+w[3]+w[4]) + w[3]*(w[0]**-1+w[3]**-1) + (w[5] + Cload3)*(w[0]**-1 + w[3]**
    T3 = (w[5]+Cload3)*(w[0]**-1+w[3]**-1+w[5]**-1) + w[3]*(w[0]**-1+w[3]**-1) + (w[0]+w[1]+w[2]+Cload1
    T = np.max([T1,T2,T3])
    return T
```

In [108]:
```
# Solve for multiple values of mu:
opt_val = np.zeros(N)
opt_w = np.zeros((6,N))
opt_t = np.zeros(N)
opt_T = np.zeros(N)

for i in np.arange(N):
    w = cp.Variable(shape=(6,), pos=True, name="w")
    t = cp.Variable(1,pos=True)
    T1 = (w[2] + Cload1)*(w[0]**-1 + w[1]**-1 + w[2]**-1) + w[1]*(w[0]**-1 + w[1]**-1) + (w[0]+w[3]+w[4
    T2 = (w[4]+Cload2)*(w[0]+w[3]+w[4]) + w[3]*(w[0]**-1+w[3]**-1) + (w[5] + Cload3)*(w[0]**-1 + w[3]**
    T3 = (w[5]+Cload3)*(w[0]**-1+w[3]**-1+w[5]**-1) + w[3]*(w[0]**-1+w[3]**-1) + (w[0]+w[1]+w[2]+Cload1
    T1 /= t
    T2 /= t
    T3 /= t

    mu = mu_v[i]
    objective_fn = cp.sum(w) + mu*t
    constraints = [T1 <= 1, T2 <= 1, T3 <= 1]
    problem = cp.Problem(cp.Minimize(objective_fn), constraints)
    problem.solve(gp=True)
    opt_val[i] = problem.value
    opt_w[:,i] = w.value
    opt_t = t.value
    opt_T[i] = f_T(w.value,Cload1,Cload2,Cload3)
```

In [125]:
```
A_opt = np.sum(opt_w,axis=0)

plt.plot(opt_T,A_opt)
plt.grid()
# plt.xscale('log')
plt.show()
```



In [126]:
```
# Optimal solution and non-optimal solution from part a overlaid:

plt.plot(opt_T,A_opt)
plt.plot(T_v,A_v,'--')
plt.grid()
# plt.xscale('log')
plt.show()
```