UNIVERSITY OF TEHRAN

COLLEGE OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

CONVEX OPTIMIZATION

CA#1

MOHAMMAD HEYDARI

810197494

UNDER SUPERVISION OF:

DR. HADI AMIRI

ASSISTANT PROFESSOR

SCHOOL OF ELECTRICAL AND COMPUTER ENGINEERING

UNIVERSITY OF TEHRAN

Nov. 2021

1 CONTENTS

2 QUESTION #1

2.1 MATLAB CVX PACKAGE

CVX is a MATLAB-based modelling system for convex optimization. CVX turns MATLAB into a modelling language, allowing constraints and objectives to be specified using standard MATLAB expression syntax.

In its default mode, CVX supports a particular approach to convex optimization that we call disciplined convex programming. Under this approach, convex functions and sets are built up from a small set of rules from convex analysis, starting from a base library of convex functions and sets. Constraints and objectives that are expressed using these rules are automatically transformed to a canonical form and solved. For more information on disciplined convex programming, see these resources; for the basics of convex analysis and convex optimization, see the book Convex Optimization.

CVX also supports geometric programming (GP) through the use of a special GP mode. Geometric programs are not convex, but can be made so by applying a certain transformation. In this mode, CVX allows GPs to be constructed in their native, nonconvex form, transforms them automatically to a solvable convex form, and translates the numerical results back to the original problem.

Version 2.0 of CVX brings support for mixed integer disciplined convex programming (MIDCP). Mixed integer DCPs must obey the disciplined convex programming ruleset; however, one or more of the variables may be constrained to assume integer or binary values. It is important to note that MIDCPs are not convex, and most non-convex models cannot be expressed as an MIDCP. Not all solvers support MIDCPs, and those that do cannot guarantee a successful solution in reasonable time for all models. Nevertheless, we believe that MIDCP support is a powerful addition to CVX and we look forward to seeing how our users take advantage of it.

It is quite important to also note what CVX is not. It is not a general-purpose tool for nonlinear optimization, nor is it a tool for checking whether or not your model is convex. It is important to confirm that your model can be expressed as an MIDCP or a GP before you begin using CVX. If it is neither of these, then CVX is not the correct tool for the task.

The CVX also package includes a growing library of examples to help get you started, including examples from the book Convex Optimization and from a variety of applications.

I have used below links to install CVX package:

https://www.youtube.com/watch?v=CmskN_RzAf4

3 QUESTION #2

3.1 SOLVING OPTIMIZATION PROBLEMS WITH DIFFERENT WAYS AND USING MATLAB

In this part we intend to implement and solve an optimization problem using MATLAB, further you can find some information about CVX package and also project descriptions.

Note that in this project we have used different algorithms in a wide range which include:

1. Equal lamp powers.

2. Least-squares with rounding

3. Weighted least-squares.

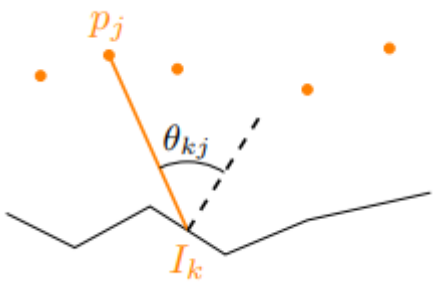
4. Chebyshev approximation.

5. Piecewise-linear approximation.

In this CA we compare several approximate least-squares and linear programming solutions for the illumination problem. The CA also serves as a review of least-squares and linear programming.

The Illumination problem:

m lamps illuminating n, small flat patches.



intensity Ik at patch k depends linearly on lamp powers pj :

$$I_k = \sum_{j=1}^{m} a_{kj} p_j = a_k^T p \qquad (1)$$

$$a_{kj} = r_{kj}^{-2} \max\{\cos\theta_{kj}, 0\} \qquad (2)$$

Now it's time to describe the optimization problem:

problem: achieve desired illumination Id with bounded lamp powers.

$$\begin{aligned} &\text{minimize} \quad \max_{k=1,\ldots,n} |\log I_k - \log I_d| \\ &\text{subject to } 0 \leq p_j \leq p_{\max}, \; j = 1, \ldots, m \end{aligned}$$

Data file has been attached in related directory please have a look and analyse vectors to find a better insight about detailed information.

There are 10 lamps **(m = 10)** and 20 patches **(n = 20)**. We take $Id = 1$ and $pmax = 1$.

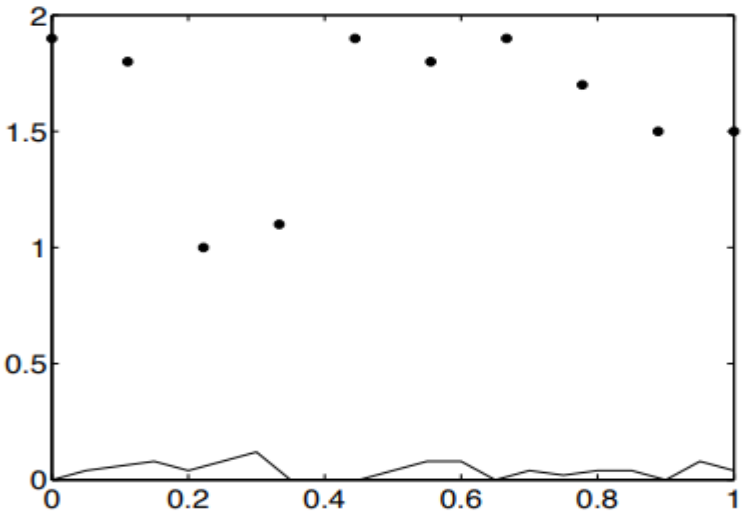


Figure 1: Patch and lamp geometry

Afterward, we should serve our knowledge to implement each algorithm in MATLAB environment and analyse each of them to obtain desired and optimal results!

4 QUESTION #3

4.1 MATLAB SIMULATIONS

First of all Note that the $f_0(p)$ is obtained from below calculations:

$$\text{minimize } f_0(p) = \max_{k=1,\dots,n} h(a_k^T p)$$
$$\text{subject to } 0 \le p_j \le p_{\max},\ j = 1,\dots,m$$

where h(u) = max {u, 1/u} for u > 0. The function h, shown in the figure below, is nonlinear, non-differentiable, and convex. To see the equivalence between (1) and (2), we note that

$$\begin{aligned} f_0(p) &= \max_{k=1,\dots,n} |\log(a_k^T p)| \\ &= \max_{k=1,\dots,n} \max\{\log(a_k^T p), \log(1/a_k^T p)\} \end{aligned}$$

$$\begin{aligned} &= \log \max_{k=1,\dots,n} \max\{a_k^T p, 1/a_k^T p\} \\ &= \log \max_{k=1,\dots,n} h(a_k^T p), \end{aligned}$$

and since the logarithm is a monotonically increasing function, minimizing f_0 is equivalent to minimizing $\max k = 1,\dots,n\ h(a_k^T p)$

1. Equal lamp powers:

Take $pj = \gamma$ for $j = 1,\dots,m$, and plot f_0(p) versus γ over the interval [0,1]. Graphically determine the optimal value of γ.

```
lamp_data = data_func();    % Loading our data

% method 1: equal lamp powers

[n,m]=size(lamp_data);
point_Num=1000;
p = logspace(-3,0,point_Num);
f = zeros(size(p));
for k=1:point_Num
f(k) = max(abs(log(lamp_data*p(k)*ones(m,1))));
end
[value_method1,idx] = min(f);
p_equal = p(idx)*ones(m,1);
```

Figure 2: Method1

2. Least-squares with rounding:

Algorithm:

If the solution has negative values for some p_i , set them to zero; if some values are greater than 1, set them to 1. Give the resulting value of $f_0(p)$. Least-squares solutions can be computed using the Matlab backslash operator:

A\b returns the solution of the least-squares problem.

$$minimize\ ||Ax - b||_2^2$$

```
% method 2: least-squares with rounding


p_state2 = lamp_data\ones(n,1);
p_state2 = max(p_state2,0);                    % rounding negative p_i to 0
p_state2 = min(p_state2,1);                    % rounding p_i > 1 to 1
value_method2 = max(abs(log(lamp_data*p_state2)));
```

Figure 3: Method2

3. Weighted least-squares:

Algorithm:

Solve the weighted least-squares problem:

$$minimize \sum_{k=1}^{n}(a_k^T p - 1)^2 + \sum_{j=1}^{m}(p_j - 0.5)^2$$

If the solution has negative coefficients, set them to zero; if some coefficients are greater than 1, set them to 1.

we can use the backslash operator in MATLAB to solve the Weighted least-squares problem.

```
vector = linspace(1e-3,1,point_Num);
store = [];
for j=1:point_Num
p = [lamp_data; sqrt(vector(j))*eye(m)]\[ones(n,1); sqrt(vector(j))*0.5*ones(m,1)];
store = [ store norm(p-0.5,inf)];
end
idx = find(store <= 0.5);
new_vec = vector(idx(1));
p_state3 = [lamp_data; sqrt(new_vec)*eye(m)]\[ones(n,1); sqrt(new_vec)*0.5*ones(m,1)];
value_method3 = max(abs(log(lamp_data*p_state3)));
```

Figure 4: Method3

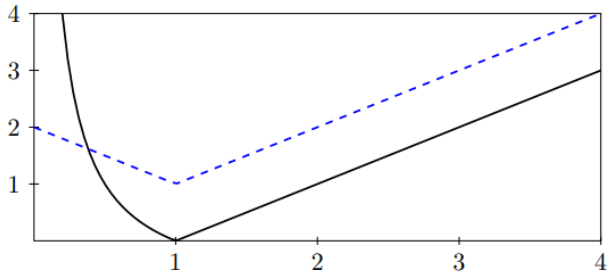
4. Chebyshev approximation:

Algorithm:

$$\text{minimize } f_0(p) = \max_{k=1,\dots,n} |a_k^T p - 1|$$

$$\text{subject to } 0 \le p_j \le p_{\max}, j = 1, \dots, m$$

using linear programming. We can think of this problem as obtained by approximating the non-linear function $h(u)$ by a piecewise-linear function $|u - 1| + 1$. As shown in the figure below, this is a good approximation around $u = 1$.



we can solve the Chebyshev approximation problem using cvx. The (convex) function $||Ap - 1||_\infty$ can be expressed in cvx as norm(A*p-ones(n,1),inf). Give the resulting value of $f_0(p)$.

```
% method 4: chebyshev approximation


cvx_begin
variable p_chebyshew(m)
minimize(norm(lamp_data*p_chebyshew-1, inf))
subject to
p_chebyshew >= 0
p_chebyshew <= 1
cvx_end
value_chebyshew = max(abs(log(lamp_data*p_cheb)));
```

Figure 5: Method4

5. Piecewise-linear approximation:

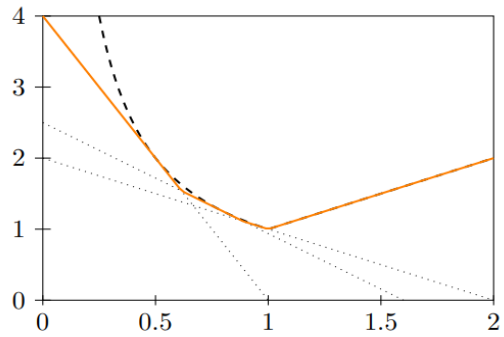
Algorithm:

We can improve the accuracy of the previous method by using a piecewise-linear approximation of h with more than two segments. To construct a piecewise-linear approximation of 1/u, we take the pointwise maximum of the first-order approximations

$$h(u) \approx 1/\hat{u} - (1/\hat{u}^2)(u - \hat{u}) = 2/\hat{u} - u/\hat{u}^2$$

at a number of different points $\hat{u}$. This is shown in the figure below, for $\hat{u} = 0.5, 0.8, 1$. In other words,

$$h_{\text{pwl}} = \max\left\{u, \frac{2}{0.5} - \frac{1}{0.5^2}u, \frac{2}{0.8} - \frac{1}{0.8^2}u, 2 - u\right\}$$



Solve the problem

$$\begin{aligned} &\text{minimize } f_0(p) = \max_{k=1,\ldots,n} h_{\text{pwl}}(a_k^T p) \\ &\text{subject to } 0 \le p_j \le p_{\max}, j = 1, \ldots, m \end{aligned}$$

using linear programmming

```
% method 5: Piecewise-linear approximation

H_new=@(u) max([ u , 2/0.5 - u/(0.5)^2 , 2/0.8-u/(0.8)^2 , 2-u]);

cvx_begin
variable p_state5(m)
minimize(max(H_new(lamp_data*p_state5)))
subject to
p_state5 >= 0
p_state5 <= 1
cvx_end

value_method5 = max(abs(log(lamp_data*p_state5)));
```

Figure 6: Method5

6. Exact Solution:

```
% exact solution:


cvx_begin
variable p_exact(m)
minimize(max([lamp_data*p_exact; inv_pos(lamp_data*p_exact)]))
subject to
p_exact >= 0
p_exact <= 1
cvx_end
val_exact = max(abs(log(lamp_data*p_exact)));
```

Figure 7: exact solution

Conclusion:

```
ans =

    0.3448    1.0000    0.4170    1.0000    1.0000    1.0000
    0.3448         0    0.4022    0.1165    0.1896    0.2023
    0.3448    1.0000    0.0950    0.0000    0.0000    0.0000
    0.3448         0    0.1393    0.0000    0.0000    0.0000
    0.3448         0    0.4133    1.0000    1.0000    1.0000
    0.3448    1.0000    0.4042    0.0000    0.0000    0.0000
    0.3448         0    0.4224    1.0000    1.0000    1.0000
    0.3448    1.0000    0.4071    0.0249    0.1640    0.1882
    0.3448         0    0.3939    0.0000    0.0000    0.0000
    0.3448    1.0000    0.4164    1.0000    1.0000    1.0000


ans =

    0.4693    0.8628    0.4609    0.4198    0.3664    0.3575
```

As you can see results match with our previous knowledge as well as we supposed,

You can see that the column number 6 is related to exact solution and as you can see in all five methods we have reached same values around exact value which abbreviated in above table.

As we expected the accuracy is increasing with using much accurate method such as Piecewise-linear approximation, chebyshev approximation and also weighted least-square but the most accurate one is **Piecewise-linear approximation** which used a new h function with Piecewise-linear approach and driven a fantastic values which is 0.3664 and also we can obtain that this is the most efficient method which implemented using CVX package in MATLAB.

Note that the description of MATLAB CVX package and also project description has been attached in previous parts.

For running this part please have an accurate look on related directory and There you can easily find all of you need and also I would appreciate it if you would consider them ☺

you can find all of my code implementation using MATLAB if there is any issue with that please let me know to provide more detailed information. (be in touch **here!**)