MATLAB CVX Tutorial

We will cover some basic examples using CVX in Matlab. You can download CVX and find its documentation here: http://cvxr.com/cvx. (We appreciate the contributions of Brendon Anderson and Yatong Bai in preparing this script.)

1 Installations

CVX is supported on 64-bit versions of Linux, Mac OSX, and Windows. In generally, we strongly recommend that you use the latest version of MATLAB that you can obtain.

1.1 Installing CVX on MATLAB

(a) To download MATLAB and acquire the academic license, you can go to
https://software.berkeley.edu/matlab%C2%AE.
You can download CVX via GitHub.

(b) Unpack the file anywhere you like; a directory called cvx will be created. There are two important exceptions:

- Do not place CVX in Matlab's own toolbox directory.
- Do not unpack a new version of CVX on top of an old one. We recommend moving the old version out of the way, but do not delete it until you are sure the new version is working as you expect.

(c) Start Matlab. Do not add CVX to your path by hand.

(d) Change directories to the top of the CVX distribution, and run the `cvx_setup` command. For example, if you installed CVX into `C\personal\cvx` on Windows, type these commands:

```
cd C:\personal\cvx
cvx_setup
```

at the MATLAB command prompt. If you installed CVX into ∼`/MATLAB/cvx` on Linux or a Mac, type these commands:

```
cd ~/MATLAB/cvx
cvx_setup
```

The `cvx_setup` function performs a variety of tasks to verify that your installation is correct, sets your Matlab search path so it can find all of the CVX program files, and runs a simple test problem to verify the installation.

(e) In some cases—usually on Linux—the `cvx_setup` command may instruct you to create or modify a `startup.m` file that allows you to use CVX without having to type `cvx_setup` every time you re-start Matlab.

1.2 Installing a CVX Professional License

If you acquire a license key for CVX Professional, the only change required to the above steps is to include the name of the license file as an input to the `cvx_setup` command. For example, if you saved your license file to `~/licenses/cvx_license.mat` on a Mac, this would be the modified command:

```
cd ~/MATLAB/cvx
cvx_setup ~/licenses/cvx_license.mat
```

If you have previously run `cvx_setup` without a license, or you need to replace your current license with a new one, simply run `cvx_setup` again with the filename. Once the license has been accepted and installed, you are free to move your license file anywhere you wish for safekeeping—CVX saves a copy in its preferences.

2 Basic Usage of CVX

2.1 Defining variables, constraints, and objective functions in CVX

2.1.1 Variables

All variables must be declared using the `variable` command (or `variables` command; see below) before they can be used in constraints or an objective function. A `variable` command includes the name of the variable, an optional dimension list, and one or more keywords that provide additional information about the content or structure of the variable.

Variables can be real or complex scalars, vectors, matrices, or n-dimensional arrays. For instance,

```
variable X
variable Y(20,10)
variable Z(5,5,5)
```

Variable declarations can also include one or more keywords to denote various structures or conditions on the variable. For instance, to declare a nonnegative variable, use the `nonnegative` keyword:

```
variable x(10) nonnegative
```

2.1.2 Constraints

The following constraint types are supported in CVX:

- Equality == constraints, where both the left- and right-hand sides are affine expressions.
- Less-than <= inequality constraints, where the left-hand expression is convex, and the right-hand expression is concave.
- Greater-than >= constraints, where the left-hand expression is concave, and the right-hand expression is convex.

The non-equality operator ~= may never be used in a constraint; in any case, such constraints are rarely convex. The latest version of CVX now allows you to chain inequalities together; e.g., l <= x <= u. (Previous versions did not allow chained inequalities.)

Note the important distinction between the single equals =, which is an assignment, and the double equals ==, which denotes equality; for more on this distinction, please refer to the official website of CVX.

2.1.3 Objective functions

Declaring an objective function requires the use of the `minimize` or `maximize` function, as appropriate. The objective function in a call to minimize must be convex; the objective function in a call to maximize must be concave; for instance:

```
minimize( norm( x, 1 ) )
maximize( geo_mean( x ) )
```

At most one objective function may be declared in a CVX specification, and it must have a scalar value.

If no objective function is specified, the problem is interpreted as a feasibility problem, which is the same as performing a minimization with the objective function set to zero. In this case, `cvx_optval` is either 0, if a feasible point is found, or +Inf, if the constraints are not feasible.

2.2 Types of problems that can be solved in CVX

CVX can solve convex optimization problems. At this point in the class, we haven't formally defined convex sets, functions, or problems, but intuitively you can think of them as follows: Convex functions are the ones that are "bowl" shaped with positive curvature. Convex optimization problems are problems whose objective function to minimize is convex, and whose constraints are defined in terms of inequalities and equalities of convex and affine (linear) functions.

If you ask CVX to solve an optimization that isn't convex, it will give you an error. As an example, run the following code examples:

Example 1. (Objective is nonconvex) Consider the optimization problem:

$$\min \ |(x^2 - 1)| \tag{1}$$

We can formulate the above problem in MATLAB CVX:

```
cvx_begin
variables x
minimize( abs(x^2-1) )    % Draw this to see why it's nonconvex.
cvx_end
```

Example 2. (Constraint is nonconvex) Consider the optimization problem:

$$\begin{aligned} &\min\ x \\ &\text{s.t. } x \in \{0, 1\} \end{aligned} \tag{2}$$

We can formulate the above problem in MATLAB CVX:

```
cvx_begin
variables x
minimize( x )
subject to
    x*(x-1) == 0          % This is a nonconvex integer constraint.
cvx_end
```

Note that the above examples are very easy to solve by hand, but CVX fails since they are not in a convex formulation. If you find that your CVX code fails, make sure to double check that your problem is convex. If it isn't, you may be able to reformulate it into a convex problem, or at least relax it into one. We will see more on these techniques later in the course.

A simple univariate example.

Consider $f(x) = (x-2)^2 + 1$. Can show f is minimized at $x^* = 2$ and $f(x^*) = 1$. Let's compute this numerically.

```
cvx_begin                   % Start CVX environment.
variables x                 % Define optimization variable.
minimize( (x-2)^2 + 1 )     % Define objective function.
cvx_end                     % End CVX environment and solve.
```

Now let's print the minimizer and objective value. After CVX solves, the solution x^* is stored in the variable x, and the optimal value $f(x^*)$ is stored in `cvx_optval`.

```
disp(['Minimizer: x*=', num2str(x)]);
disp(['Optimal value: f(x*)=', num2str(cvx_optval)]);
disp('Press enter to continue to CVX status.');
pause;
```

2.3 CVX status

Let's print the variable `cvx_status` from the previous example.

```
disp(['CVX status: ',num2str(cvx_status)]);
disp('Press enter to continue.');
pause;
```

The previous optimization was successfully 'Solved'. Let's look at examples where CVX may appear to have run successfully, but in fact the optimization couldn't be solved.

Example 3. (Unbounded)

```
cvx_begin
variables x
minimize( x )
cvx_end
disp(['CVX status: ',num2str(cvx_status)]);
disp('Press enter to continue.');
pause;
```

Example 4. (Infeasible)

```
cvx_begin
variables x
minimize( x^2 )
subject to
    x <= 1
    x >= 2
cvx_end
disp(['CVX status: ', num2str(cvx_status)]);
```

The above two status' show that there are issues with your optimization problem's formulation. Double check to ensure your problem has a solution (e.g., if objective is continuous and feasible set is compact, then there exists a solution), and also that the problem is feasible.

There are also CVX status' 'Inaccurate' and 'Failed', which sometimes occur when trying to solve particularly nasty problems. If you see one of these two status', try changing the CVX solver/algorithm, or adjusting the precision/tolerance. See the CVX website for more information.

The sixth status is 'error' The optimization problem may be bounded and feasible, but CVX encountered a problem when executing.

3 Conclusion.

We've covered the basics of CVX through a series of examples. There are many more examples on the CVX website. Furthermore, there are many built-in features that CVX has to handle common convex functions and constraints. Furthermore, you can specify solver preferences such as precisions/tolerances, as well as which numerical solver/algorithm to use (if you don't want to use CVX's default choice). See the CVX website for information on these features.