# Introduction

We wish to maximize:

$$f(\mathbf{U}) = \log|\det \mathbf{U}|, \tag{1}$$

where $\mathbf{U} \in \mathbb{R}^{n\times n}$, subject to the following constraint:

$$\|\mathbf{U}\mathbf{Y}\|_\infty \leq 1 \tag{2}$$

where $\mathbf{Y} \in \mathbb{R}^{n\times k}$ are some random observed samples. The gradient of the objective function is given as:

$$\nabla f(\mathbf{U}) = \left(\mathbf{U}^{-1}\right)^\intercal \tag{3}$$

A priori, we know that the problem has no critical points, and that all strict optima lie on problem vertices. Our solution methodology will proceed in two parts. First we will use dynamic programming to find a vertex of the feasible region that is non-singular. From there, we will use the simplex algorithm to find the optimal vertex. For $n \leq 5$, all optima are global, and hence simplex will find the correct solution with probability 1. For $n \geq 6$, local optima exist on vertices. In this case we may be able to use the simplex algorithm in combination with backtracking, or a similar method, to find an optimal vertex. However, we will defer on this issue until we have a handle on the lower dimensional problems.

# Dynamic Programming

The main idea of this algorithm is to break the problem of finding a feasible, non-singular vertex up into a series of one-dimensional optimization problems. Our algorithm will begin by choosing a feasible point at random. From this point, we will move in the direction of maximal gradient until we reach the boundary of the feasible region. For the next step of the algorithm, we wish to move in the null space of the active constraints; by doing so we may move to activate other constraints in a manner that does not disturb our original constraint. We choose the direction within the null space of the active constraints by computing the gradient and then projecting into the null space of the active constraints. For certain values of $n$, it is possible that this projection will be zero (i.e. that the gradient is constant over the entire null space), in this case, we will simply choose a direction at random within the null space and move until we have activated another constraint. This process will repeat until we reach a vertex.

More concretely, we describe the algorithm as follows:

1. Pick an initial feasible $\mathbf{U}$ at random.

2. The first step of our algorithm is straight-forward, which we describe separately from the rest of the steps for clarity. We first compute $\Delta = \nabla\mathbf{U}$, and then solve the following one-dimensional optimization program:

$$\begin{aligned} \underset{t\in\mathbb{R}}{\text{maximize}} \quad & f(\mathbf{U} + t\Delta) && (4)\\ \text{subject to} \quad & t \geq 0,\ \|(\mathbf{U} + t\Delta)\mathbf{Y}\|_\infty \leq 1 && (5) \end{aligned}$$

   There are several approaches to solving this optimziation problem. Because there are no critical points in the problem domain, the problem reduces to finding the maximum feasible value of $t$. One simple approach would be to perform a binary search on $t$. (Probably a better approach would be to solve for $t$ by finding the smallest element (in absolute value) of the matrix $(|\mathbf{U} * \mathbf{Y}| - 1)/\Delta$, with division being done element-wise. I had the binary search as hold-over from a previous algorithm I was playing with). Ultimately, this step will be limited computationally by finding the gradient of the objective function.

3. We now must do the following $n^2 - 1$ times:

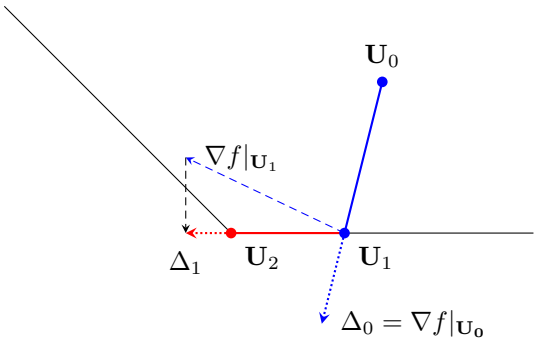


Figure 1: A graphical depiction of the first two steps of the dynamic programming algorithm. $\mathbf{U}_0$ is chosen uniformly at random. We then proceed in the direction given by the gradient until we reach the boundary of the feasible region. The next step is taken by projecting the gradient at $\mathbf{U}_1$ onto the nullspace of the basis formed by the set of active constraints.

(a) Compute $\Delta = \nabla f(\mathbf{U}_i)$.

(b) Project $\Delta$ onto the nullspace of the active constraints. If $\Delta$ is orthogonal to the nullspace, then simply choose a (feasible) direction at random within the nullspace and use this as $\Delta$.

(c) Find the maximum feasible value of $t$ from program given in (4) – (5).

This process is depicted in Figure 1. This will take $O(n^3)$ for each search step (since we can't get around computing the gradient) and will require $n^2$ steps. Thus our total runtime should be $O(n^5)$. This runtime might not be quite as bad as it seems: MIMO decoding *with CSI* can at best be approximated in $O(n^3)$ *per transmission* (on the order of once per microsecond), and maximum-likelihood decoding of MIMO signals is at worst exponential-time. Additionally, MIMO channels change somewhat slowly over time so we'd only have to perform our algorithm on the order of once per 100ms or several second depending on our operating conditions.

## Simplex

There is no guarantee that value of $\mathbf{U}$ obtained above will be optimal. In fact, empirically, it seems like we end up on a non-singular vertex almost uniformly at random. From here, we need to apply simplex to get to an optimal solution. Notice that exactly what we will need to do will depend on the dimension:

- For $n = 2, 3$, all non-singular vertices are optimal and solutions to our problem. In this case, simplex is not needed.
- For $n = 4$, we will have to hop at most one vertex to get to an optimal solution.
- For $n = 5$, we will have to hop at most 3 vertices to get to an optimal soltuion.
- For $n \geq 6$, there are local optima so we might have to do some backtracking or something.