

HOMEWORK 1

A. BOLSHAKOV

ABSTRACT. If you are looking for Python code, mine is at https://github.com/atbolsh/RRR_HW1. `prob2.py` is specific to that problem, but I intend to reuse my `RRR.py` and my image-manipulation functions from `prob3.py`.

1. PROBLEM 1

Many constraint surfaces can be locally linearized. Let us examine the behavior of the dynamical system

$$\dot{\mathbf{x}} = P_A(\mathbf{x}) - P_B(\mathbf{x})$$

for this large class of systems by just looking at a simple, 2 dimensional case. Let constraint A be $x_2 = mx_1$ and constraing B be $x_2 = -mx_1$ for some slope m . (see Fig. 1).

The system is simple enough that we can explicitly solve for P_A and P_B :

$$P_A \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \frac{1}{m^2 + 1} \begin{pmatrix} x_1 + mx_2 \\ mx_1 + m^2 x_2 \end{pmatrix}$$
$$P_B \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \frac{1}{m^2 + 1} \begin{pmatrix} x_1 - mx_2 \\ m^2 x_2 - mx_1 \end{pmatrix}$$

We can ignore the $m^2 + 1$ coefficient as we only care about long term behavior. So, we get

$$\dot{\mathbf{x}} = P_A(\mathbf{x}) - P_B(\mathbf{x}) = \begin{pmatrix} 2mx_2 \\ 2mx_1 \end{pmatrix} = \begin{pmatrix} 0 & 2m \\ 2m & 0 \end{pmatrix} \mathbf{x}$$

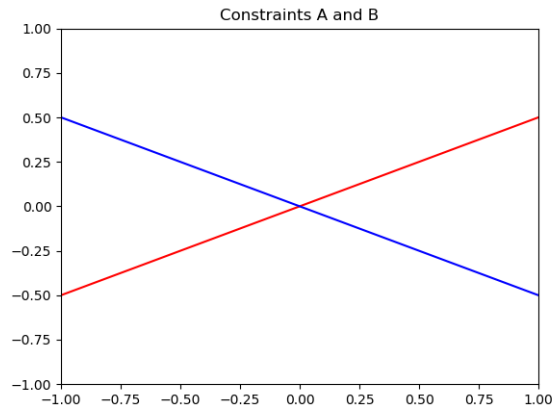


FIGURE 1. Example constraint surfaces. A is red, B is blue.

This is a classic linear hyperbolic system. The two eigenvalues are $\pm 2m$, so all trajectories except a set of measure 0 will flow away.

2. PROBLEM 2

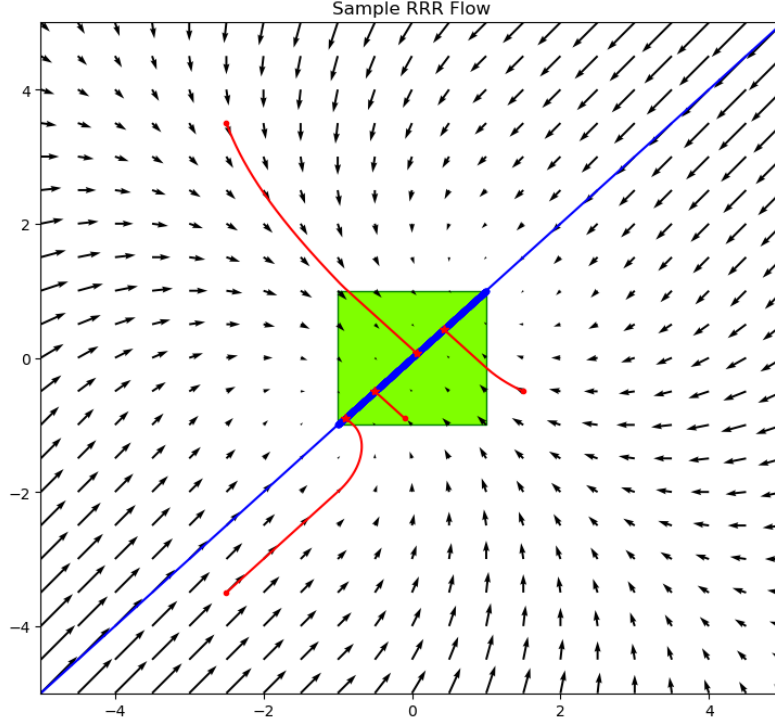


FIGURE 2. RRR flow, including constraint surfaces (blue and green), the set of fixed points (dark blue), 4 sample trajectories (red), and the vector field for this dynamical system. Notice that since the problem domain \mathbb{R}^2 is spanned by the constraint surfaces, we do not need to worry about the distinction between \mathbf{x} and $P_A(\mathbf{x})$.

Fig. 2 shows the RRR flow. Numerical integration does not suggest any cycles, at least not in the smooth limit (I use $\beta = 0.01$); all 4 of the tested trajectories converge, and my sample points represent all 4 important regions:

- (1) inside A ;
- (2) directly above or to the left of A ;
- (3) above-right / below-left of A ;
- (4) and above-left / below-right of A .

It would be possible to go through and rigorously confirm that trajectories from all 4 of these regions do, in fact, converge, but that= does not seem necessary - after all, the entire point of the algorithms is to find solutions for problems where the flow is **not** fully understood.

3. PROBLEM 3

Let N be the length of our array (image), and let P be the set of points $p \in \mathbb{R}^N$ such that $\text{sort}(p)$ gives the desired distribution (in our case, a uniform distribution).

Obviously, we wish to project some arbitrary point $x \in \mathbb{R}^N$ to this manifold P . Denote by p^* the point with indices in the same order as x .

Consider two candidate points $p, q \in P$ which differ only by a single permutation: for some $i \neq j$, $p_i = q_j$ and $p_j = q_i$, while for all $k \neq i, j$, $p_k = q_k$.

We have

$$\begin{aligned} |q - x| - |p - x| &= \\ [(x_i - q_i)^2 + (x_j - q_j)^2] - [(x_i - p_i)^2 + (x_j - p_j)^2] &= \\ [(x_i - p_j)^2 + (x_j - p_i)^2] - [(x_i - p_i)^2 + (x_j - p_j)^2] &= \\ 2x_i p_i + 2x_j p_j - 2x_i p_j - 2x_j p_i &= \\ 2(x_i - x_j)(p_i - p_j) \end{aligned}$$

Now, if x_i, x_j are in the same order as p_i, p_j (either both i -indices are smaller than the j s, or vice versa), this quantity is positive, so p is closer to x than q . If the order is mismatched, then q is closer.

There are many sorting algorithms (for instance, `quickSort`) which are based on only exchanging values when they are in the wrong order. Therefore, starting at an arbitrary point $p_0 \in P$, it is possible to use any of these algorithms to construct a sequence $p_0, p_1, p_2, \dots, p^*$ with the property

$$|p_{i+1} - x| < |p_i - x|$$

Therefore, p^* is closer to x than any arbitrary point in P , as desired.

The flattened images are below; the python code is online in `prob3.py` (the code was written to be efficient, easy to read, and reusable).

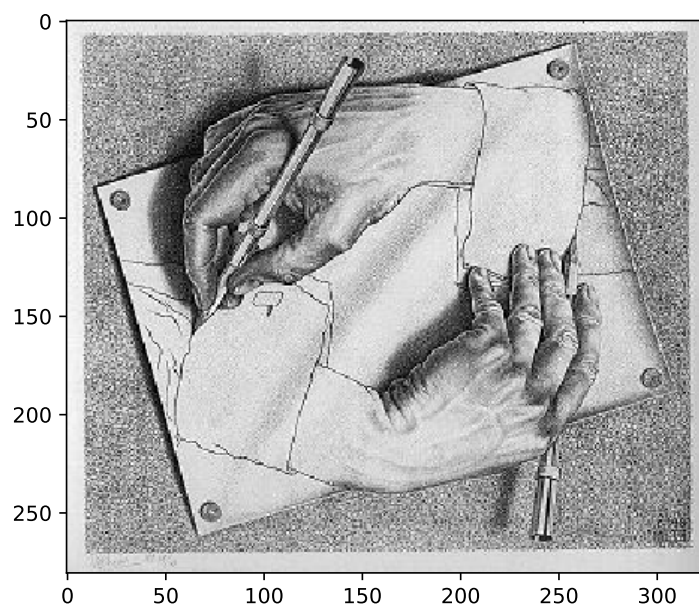


FIGURE 3. Original Picture

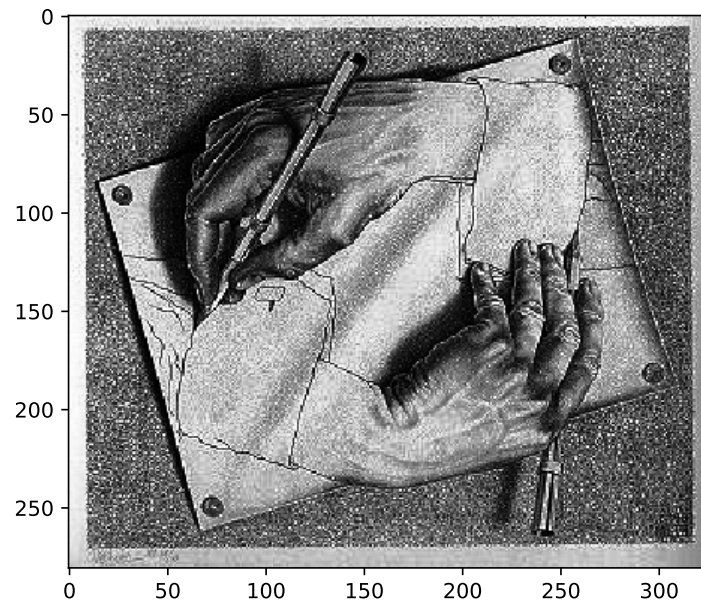


FIGURE 4. Output image with uniform distribution.

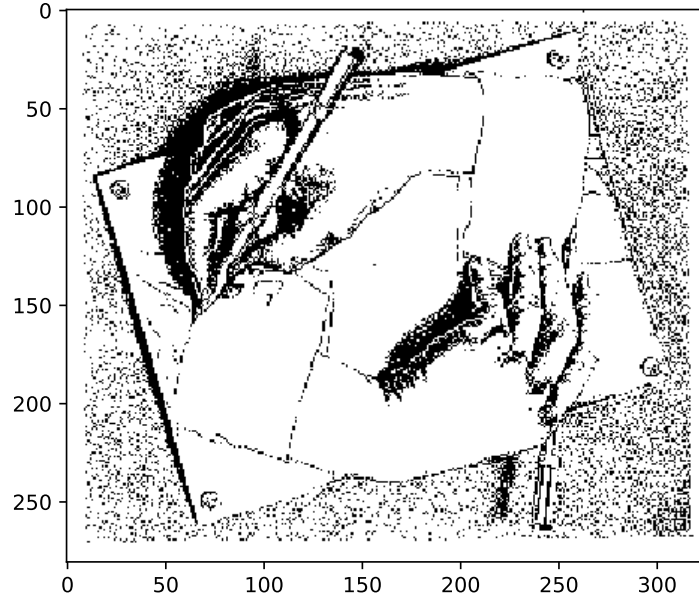


FIGURE 5. Projection onto a skewed bimodal distribution; the darkest $\frac{1}{6}$ th pixels are black, the rest are white.