Drawing Sankeys: Notes

Sam Calisch

August 26, 2011

1 General Structure

Nodes are specified as a list. The columns are subsets of this list. Flow Data input as matrix (a_{ij}) where a_{ij} is the value of flow from node i to node j.

There are (at least) three drawing factors affecting readability of Sankey diagrams.

- Position of Nodes,
- Branching order,
- Horizontal Offset, and
- The curves of the flows.

We'll address them in the order of dependency.

2 Node positioning

To position the columns horizontally, I use two parameters:

1. $gaps$: the distances between consecutive columns. It's a list of length $numColumns - 1$.The same value works for most all sankeys, but in extreme cases with a lot of stuff flying around, I bump it up.
2. $widths$: the horizontal width of nodes in each column. It's a list of length $numColumns$. It probably could just be one value for all columns.

To position the nodes within each column, I use two other parameters:

1. $betweenSpaces$: the vertical distance between nodes in each column.
2. $startys$: the starting Y value for each column.

From these, we draw the bottom-left corner of the j^{th} node in the i^{th} column at

$$X_{min} = \sum_{k=0}^{i-1}(widths(k) + gaps(k))$$

$$Y_{min} = startys(i) + j * betweenSpace(i) + \sum_{k=0}^{j-1} nodesheights(k)$$

The top-right corner of the same node is:

$$X_{max} = \sum_{k=0}^{i} widths(k) + \sum_{k=0}^{i-1} gaps(k)$$

$$Y_{max} = startys(i) + j * betweenSpace(i) + \sum_{k=0}^{j} nodesheights(k)$$

3 Branching Order

Figure 1 and Figure 2 show how to determine branching order. Figure 2 is really the better diagram.

Essentially, for two flows terminating at the same column, the branching order is determined by the change in y values between the bottom of the start node and the bottom of the terminating node. The flow which falls the most is drawn before (i.e. below) flows falling less.

This works unless we compare flows terminating in different columns. We deal with this recursively. First determine the order of the flows terminating one column from the origin column. Then determine the order of flows terminating two columns away and inject this into the first ordering at the spot representing a rise/fall of zero. Continue recursively.

d

4 Horizontal Offset

Where flows collect at the input of output of a node things can get complicated – especially when we start making curves as described in the next section. To try to manage this, I use a differing horizontal offset, X, for each flow before they start to curve. Figure 3 and Figure 4 show this.

My method keeps some flows off of each other, but it could be better. If you can figure out a better heuristic, I'd be appreciative.

The way it works is opposite for up-sloping and down-sloping flows. Consider down-sloping ones first. In plain english, we want the flows early in the branch order (i.e., the bottom flows) to bend away quickly, while the flows later (i.e., higher) should continue horizontally for a bit to give the earlier flows time to get out of the way. I know this is complicated, hopefully the picture helps.

To do this, I set y_{bottom} and y_{top} to be the y values of the bottom and top of the node rectangle, respectively. For the i^{th} downsloping flow, let y_i be its starting y value (this is determined by branching order and the thickness of the previous flows). Then, I let

$$X = \frac{1}{2}\left(y_i - y_{bottom}\right).$$

So the horizontal offset grows from zero to almost $\frac{1}{2}(y_{top} - y_{bottom})$ as we proceed through the branch ordering of down-sloping flows.

For up-sloping strands, I set

$$X = \frac{1}{2}\left(y_{top} - y_i\right).$$

so the horizontal offset shrinks from nearly $\frac{1}{2}(y_{top} - y_{bottom})$ to zero as we proceed.

Any node will usually have both down-sloping and up-sloping flows. The branch order will segregate these to the bottom and top, respectively, so the horizontal offset will start at zero at the bottom, grow to a maximum, and shrink again down to zero.

For flows which do not terminate in the next column, we add the width of the intermediate columns and gaps to the horizontal offset and compute as above.

We've focused on outgoing flows, but the same convention governs horizontal offsets of incoming flows to nodes, albeit reflected about a vertical axis. The effect of this is simply to interchange the up- and down-sloping equations.

5 Drawing Flows

Now to the trickiest and sexiest part. First, let's define the input to drawing a flow (sometimes called "strands" in the code):

- t: the thickness of a flow. At all points this must remain constant, as measured perpendicularly from the bottom border of the flow. This value comes from the data file.

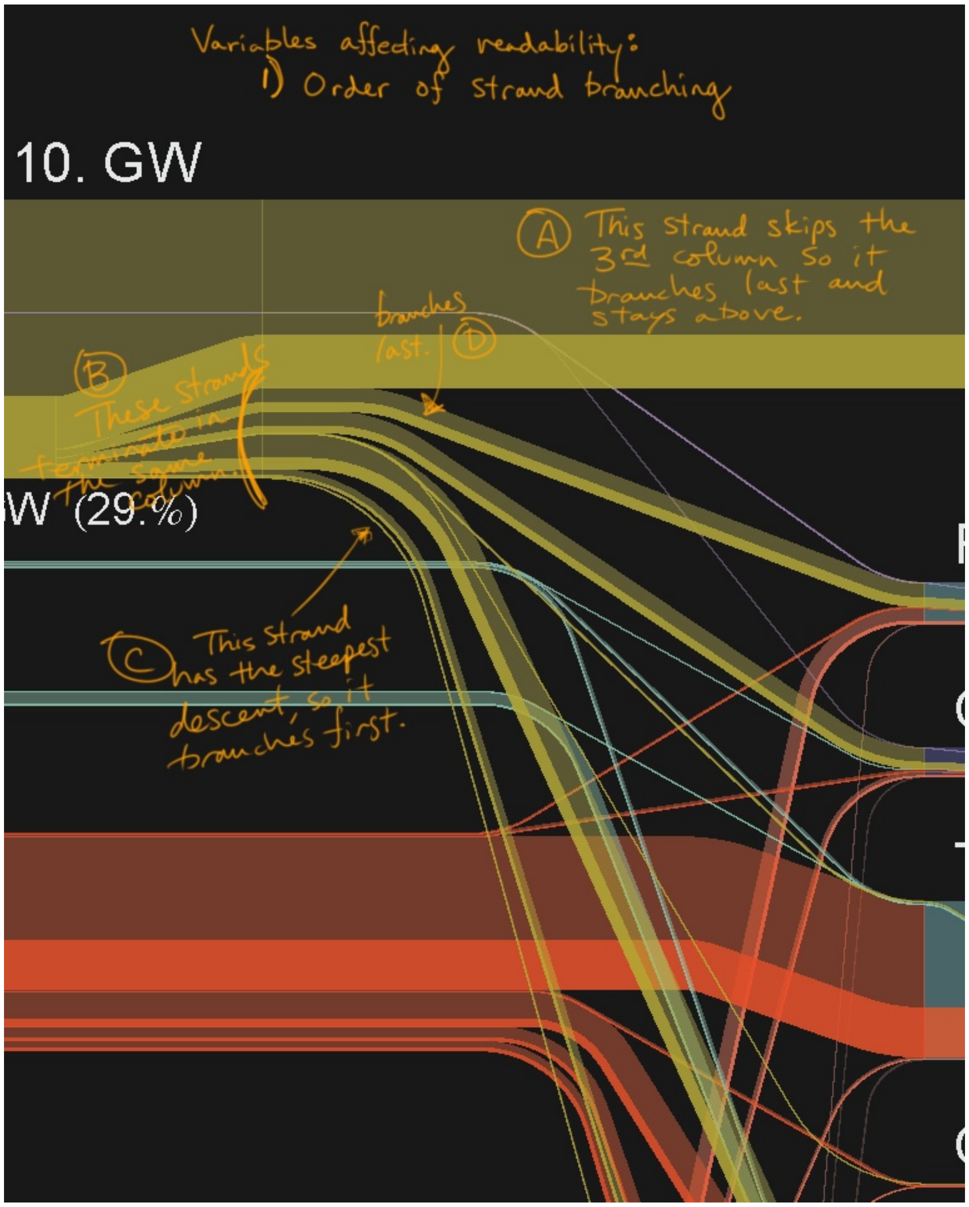

Variables affecting readability:
1) Order of strand branching
10. GW
A This strand skips the 3rd column so it branches last and stays above.
branches last. D
B These strands terminate in the same column.
W (29.%)
C This strand has the steepest descent, so it branches first.

Figure 1: Determining branching order.

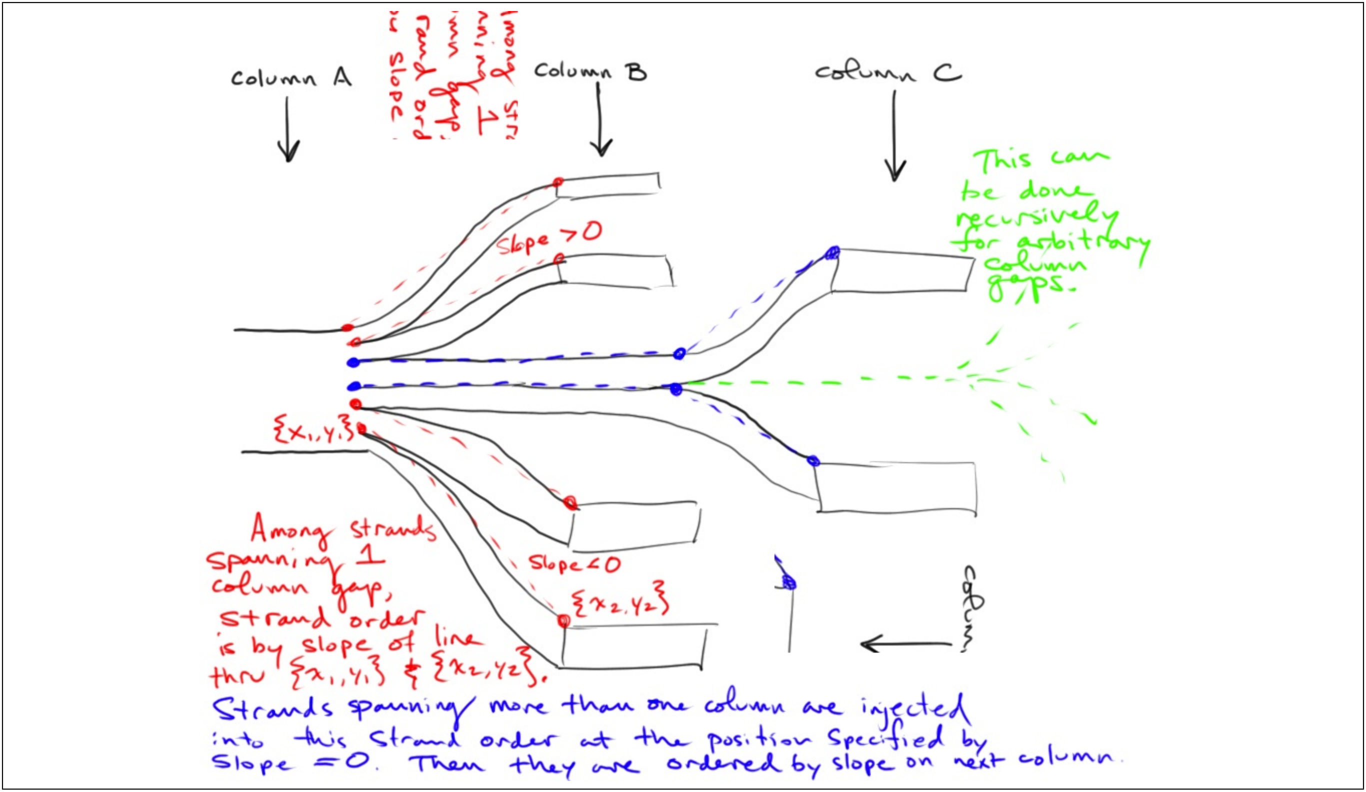

mong stra
ning 1
mn gap,
rand ord
s slope
column A
column B
column C
This can be done recursively for arbitrary column gaps.
Slope > 0
{x1, y1}
Slope < 0
{x2, y2}
column
Among strands spanning 1 column gap, strand order is by slope of line thru {x1, y1} & {x2, y2}.
Strands spanning more than one column are injected into this strand order at the position specified by slope = 0. Then they are ordered by slope on next column.

Figure 2: Determining Branching order. Drawing 2.

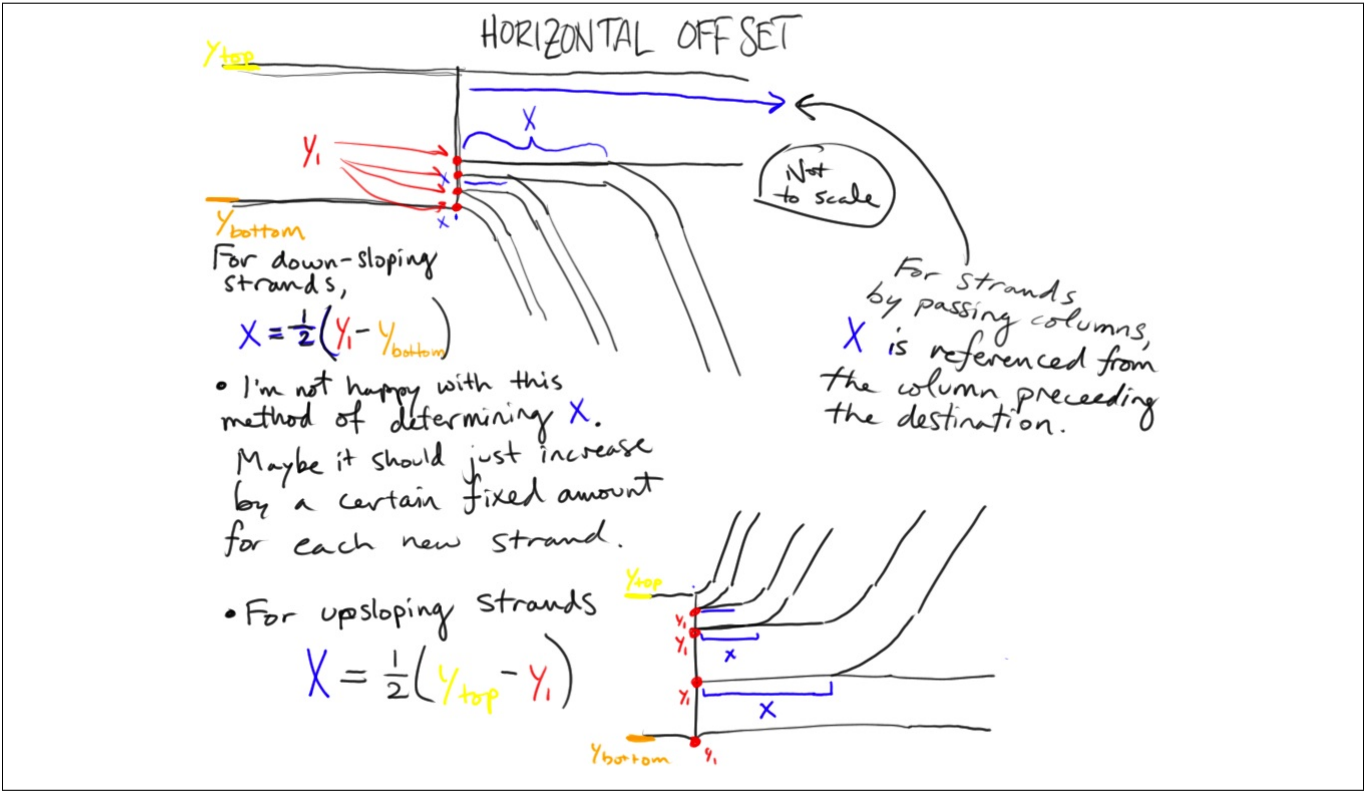

HORIZONTAL OFFSET
Ytop
Y1
X
Ybottom
Not to scale
For down-sloping strands,
X = 1/2 (Y1 - Ybottom)
• I'm not happy with this method of determining X. Maybe it should just increase by a certain fixed amount for each new strand.
• For upsloping strands
X = 1/2 (Ytop - Y1)
For strands by passing columns, X is referenced from the column preceeding the destination.
Ytop
Y1
Y1
Y1
X
X
Ybottom
Y1

Figure 3: Horizontal offset.

- $\{x_1, y_1\}$: the point specifying the bottom-left corner of the curvy part of the flow. This starts after the horizontal offset and the translation from branch ordering.
- $\{x_2, y_2\}$: the point specifying the bottom-right corner of the curvy part. Similar.

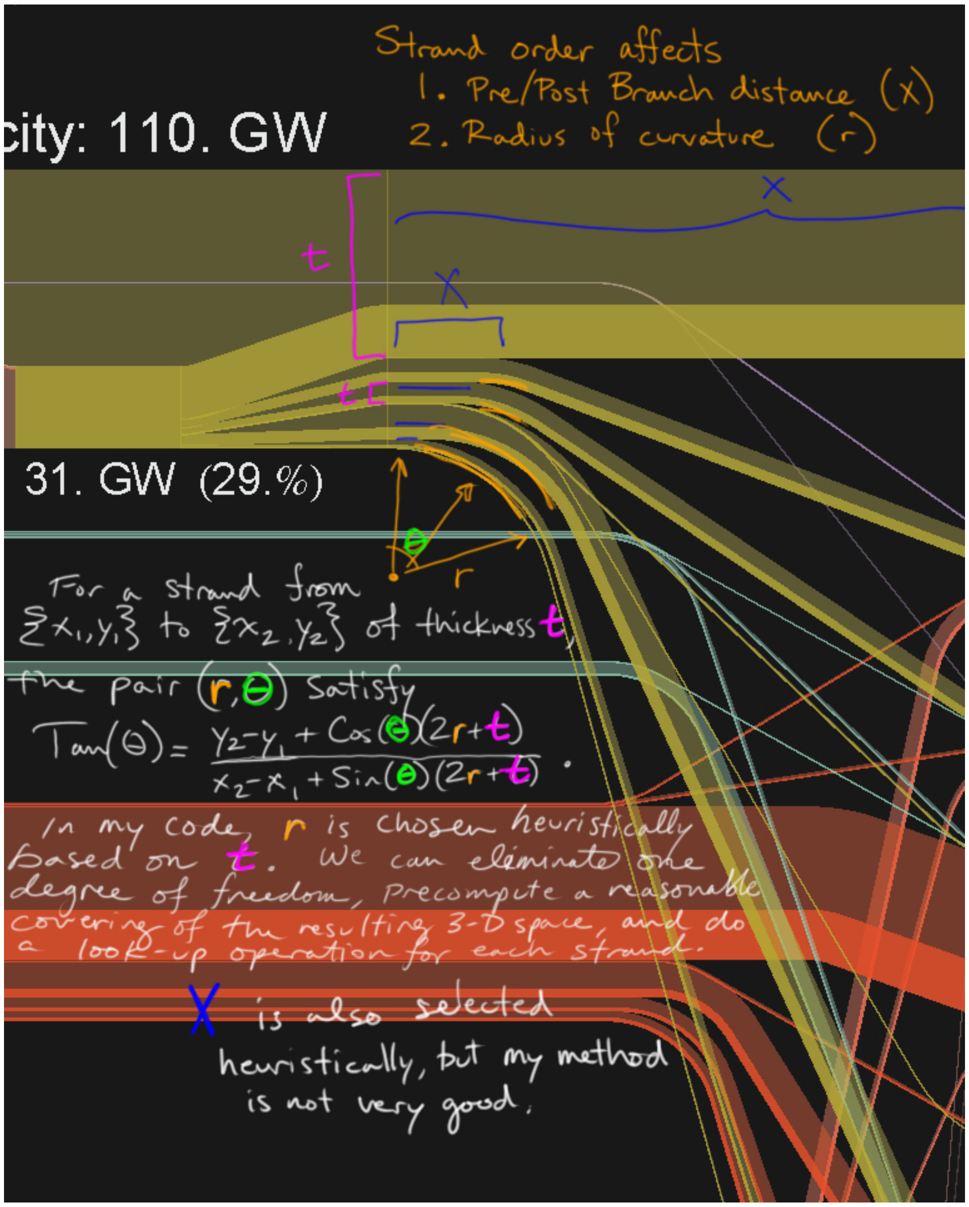
In the following, I'll describe my method for computing the curves. It could be possible to use splines, but care must be taken to ensure the flow has constant thickness at all points. In order to do this, I use a region defined by concentric circular sections, followed by a sloped rectangle, followed by the same concentric circular section region (but rotated 180°). See Figure 4 and Figure 5.

Using the conventions shown in these figures, the variables must satisfy

$$\tan\theta = \frac{y_2 - y_1 + (2r + t)\cos\theta}{x_2 - x_1 + (2r + t)\sin\theta}$$

in order to flow smoothly.

We set $r = \frac{1}{4}(x_2 - x_1)$ (for no other reason than that it seems to work) and solve for θ. Without loss of generality, say $(x_1, y_2) = (0, 0)$. Thus, θ is determined by $(x_2 - x_1)$, $(y_2 - y_1)$, and t. Note that if θ_0 solves the system for a choice of $(x_2 - x_1)$, $(y_2 - y_1)$, and t, then it also solves the system for $a(x_2 - x_1)$, $a(y_2 - y_1)$, and at for a scale factor a. In loose language, the same θ works for a short, thin strand as well as a long, thick strand of the same slope. This means we can eliminate one variable. If we precompute a reasonable grid over the resulting two-dimensional space, we can avoid doing any equation solving in the javascript. I'll give you this look-up table.



Strand order affects
1. Pre/Post Branch distance (x)
2. Radius of curvature (r)

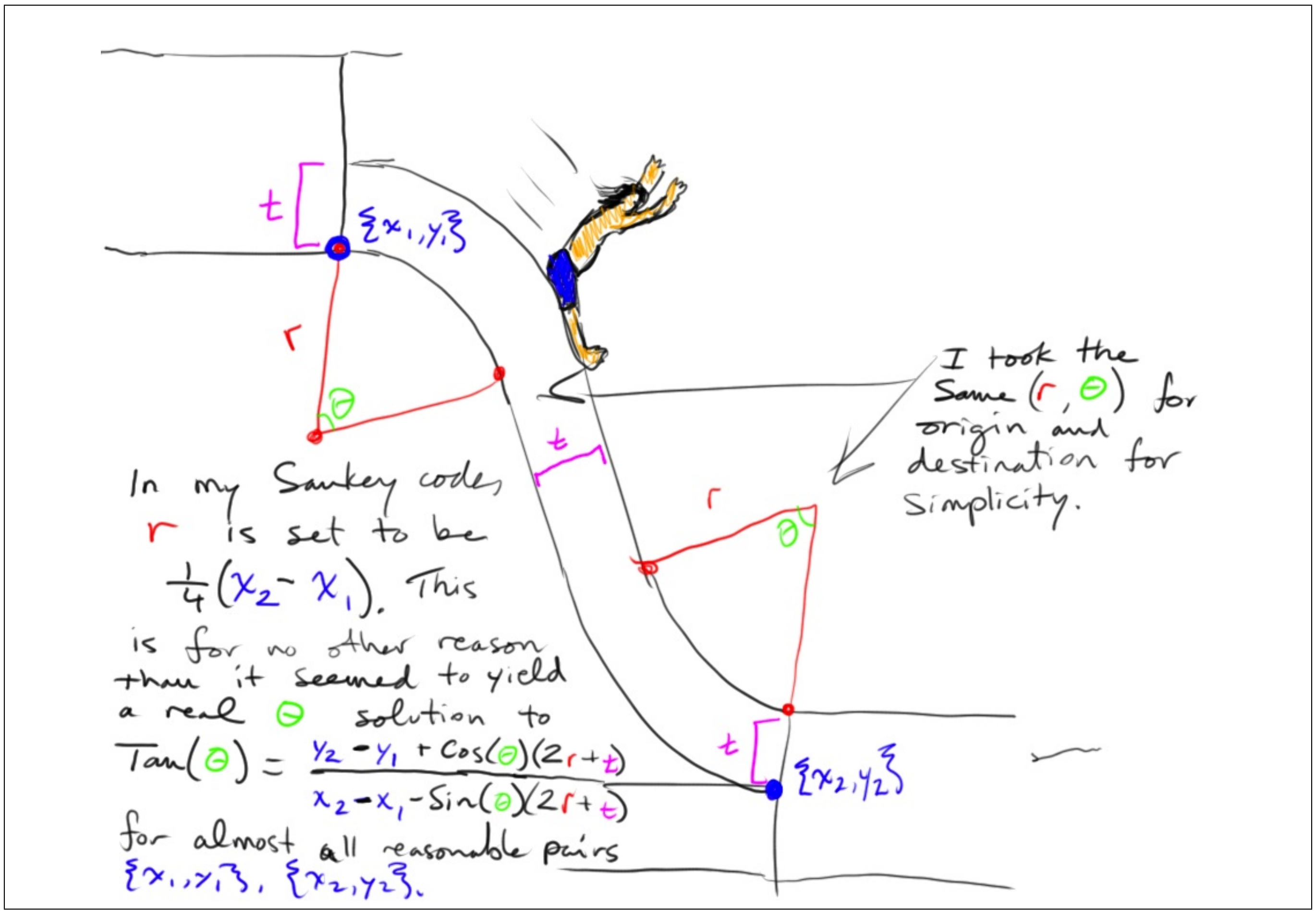
For a strand from $\{x_1,y_1\}$ to $\{x_2,y_2\}$ of thickness t, the pair (r,θ) satisfy

$$\tan(\theta)=\frac{y_2-y_1+\cos(\theta)(2r+t)}{x_2-x_1+\sin(\theta)(2r+t)}.$$

In my code, r is chosen heuristically based on t. We can eliminate one degree of freedom, precompute a reasonable covering of the resulting 3-D space, and do a look-up operation for each strand.

X is also selected heuristically, but my method is not very good.

Figure 4: Computing the flow curves.



In my Sankey codes r is set to be $\frac{1}{4}(x_2 - x_1)$. This is for no other reason than it seemed to yield a real θ solution to

$$\mathrm{Tan}(\theta) = \frac{y_2 - y_1 + \mathrm{Cos}(\theta)(2r+t)}{x_2 - x_1 - \mathrm{Sin}(\theta)(2r+t)}$$

for almost all reasonable pairs $\{x_1, y_1\}$, $\{x_2, y_2\}$.

I took the Same (r, θ) for origin and destination for Simplicity.

Figure 5: Computing flow curves