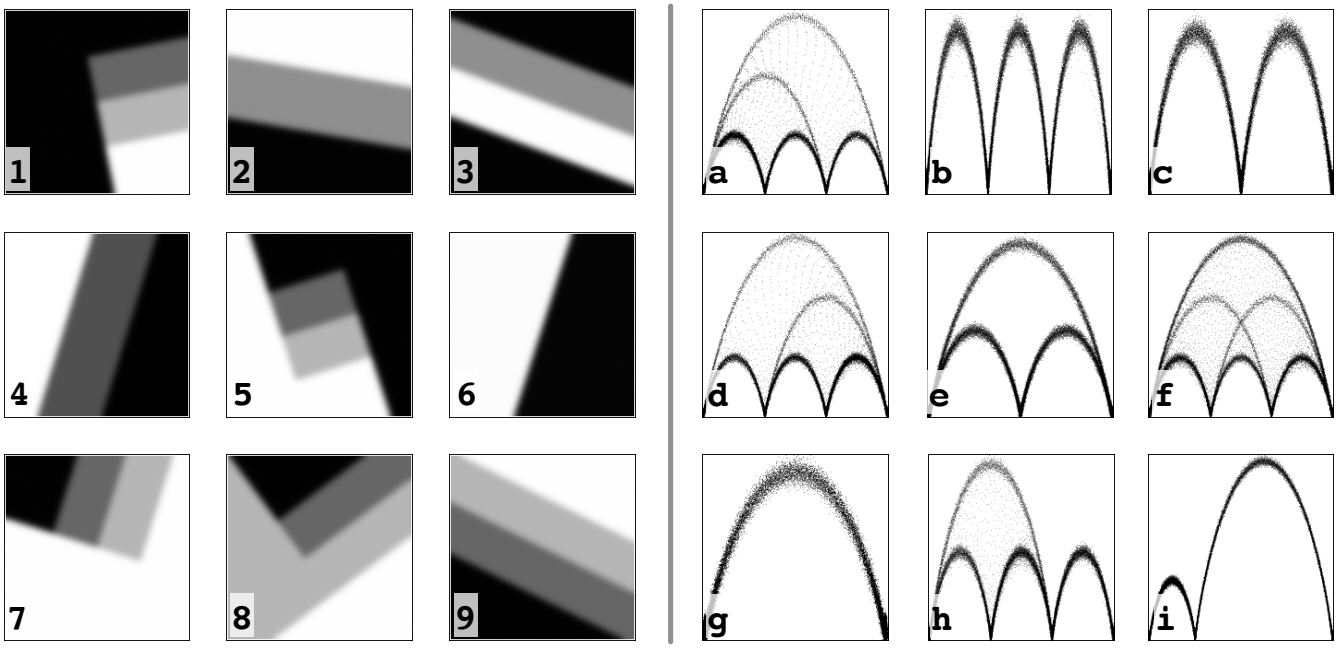
CMSC 23710/33710 **Scientific Visualization** **Homework 4**
Winter 2017 **Assigned Feb 16**
Due 9 AM Feb 22

Submit answers to questions 1 and 4 in a `hw4.pdf` file to the `hw4` subdirectory of your `CNETID-scivis-2017` repository. You may write out work by hand on paper and scan it (gray-scale is fine); but MAKE SURE THE RESULT IS LEGIBLE, and under 3 MB in size. Questions 2 and 3 require editing and committing source files; `pthdemo.c` and `tile.c` respectively.

(1) This question concerns the "Semi-Automatic Generation of Transfer Functions" reading.



In the diagram above, on the left are nine 2D synthetic datasets $f(\mathbf{x})$, and on the right are nine scatterplots of $|\nabla f(\mathbf{x})|$ (vertical, increasing upwards) and $f(\mathbf{x})$ (horizontal, increasing rightwards); scatterplots like these are described in the reading and were described in class. For each numbered dataset, give the letter of the corresponding scatterplot.

(2) You should have a `pthdemo.c` program in your `hw4` directory. Read through this program to understand what it does: reading in a 2D array, summing scanlines along the faster axis, and saving out the resulting 1D array. You will add the code necessary to make this code multi-threaded, using pthreads. As with programming projects, you insert your code in specific locations. You can check that you've respected these boundaries via:

```
./strip.sh pthdemo.c | cksum
```

which should return: "`39254702 7115`". The program should compile without warnings, using the provided compile line.

(3) You should have a `tile.c` program in your `hw4` directory. Like the `pthdemo.c` program, it takes in a 2D array, and processes it into a 2D array of 2D tiles. The array values are not changed; they are just re-arranged in memory so that what is most contiguous in memory are small 2D subsets (or "tiles") of the data, and the 2D arrangement of tiles are themselves linearized along faster and slower axes. One example input `in.txt`:

```
 0  1  2  9 10 11 18 19 20 27 28 29
 3  4  5 12 13 14 21 22 23 30 31 32
 6  7  8 15 16 17 24 25 26 33 34 35
36 37 38 45 46 47 54 55 56 63 64 65
39 40 41 48 49 50 57 58 59 66 67 68
42 43 44 51 52 53 60 61 62 69 70 71
```

Note that the ordering of values is as a 4 (faster) by 2 (slower) array of 3 by 3 tiles. These commands create `in.txt` and then process it:

```
echo 0 71 | unu resample -s 72 = -k tent -c node |
unu 1op rup | unu convert -t float |
unu reshape -s 3 3 4 2 | unu axmerge -a 2 | unu tile -a 2 0 1 -s 4 2 -o in.txt
./tile -i in.txt -ts 3 3 -o - | unu reshape -s 72 1 | unu save -f text
```

You don't need to understand the first piped command. The output of the second command should simply be an ordered list of all integers from 0 to 71 inclusive. Check that you've edited only within the permitted space by
`./strip.sh tile.c | cksum`
which should return: "`2368396283 3960`". The program should compile without warnings, using the provided compile line.

(4) Consider a 2D field of 2D vectors that satisfies

$$[\mathbf{v}(\mathbf{x})]_{\mathcal{B}} = \begin{bmatrix} -x_2 \\ x_1 \end{bmatrix} \tag{1}$$

$$[\mathbf{x}]_{\mathcal{B}} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \tag{2}$$

with orthonormal basis $\mathcal{B}$. A streamline $\mathbf{p}(t)$ is a path through the vector field such that the path derivative is the same as the local vector: $\mathbf{p}'(t) = \mathbf{v}(\mathbf{p}(t))$.

(a) In general, what is the shape of streamlines through $\mathbf{v}$? You can figure this out by hand on paper.

(b) Starting at initial point $\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$, and using integration stepsize h, find the location of the next estimated step along the streamline after taking one Euler step. **(i)** Express this new location as a column 2-vector, and then **(ii)** express taking an Euler step (through this particular vector field) as multipliplication by a 2×2 anti-symmetric matrix. The matrix should involve only real numbers and h.

(c) Same setup as previous part, but now find where one Midpoint Method step will take you. **(i)** Express this new location as a column 2-vector, and then **(ii)** express taking a Midpoint Method step as multiplication by a 2×2 anti-symmetric matrix.

(d) Comparing the matrices from **(b.ii)** and **(c.ii)**, which one is closer to a rotation matrix (which preserves lengths)? How did you assess this?