ECE-210-A HW3

Instructor: Jonathan Lam

Spring 2022

Images and layer masks This question uses `imshow` (see the help page!) and logical matrices to produce some simple shapes and show how we can use them as masks for image manipulation. This should hopefully reinforce your intuition of logical indexing in a visual manner.

1. Create the logical matrix $A \in M_{256\times256}(\mathbb{F}_2)$ where a_{ij} is true iff $\sqrt{(i-128)^2+(j-128)^2} < 64$. (Use `meshgrid` or broadcasting.)
   ($\mathbb{F}_2$, or Galois-field 2, is the field consisting of only the values $\{0,1\}$ and whose operations are akin to logical operators. For our purposes, these are the two logical values corresponding to false and true.)
2. Create the logical matrix $B \in M_{256\times256}(\mathbb{F}_2)$ where b_{ij} is true iff $\sqrt{(i-96)^2+(j-96)^2} < 64$.
3. Create the following logical matrices. For each one, use `figure` and `imshow` to visualize it. Briefly describe each one in a comment.
   (a) A
   (b) B
   (c) $C = A \cap B^C$ (The ^{C} superscript indicates logical complement.)
   (d) $D = A^C \cup B$
4. Visualize the following matrices using `figure` and `imshow`. What do the `.*` and `+` operators do when dealing with logical arrays? (Think layer masks.)

   ```
   E = rand(256, 256, 3);
   F = linspace(0, 0.25, 256) + linspace(0, 0.25, 256).';
   G = E .* C + F .* D;
   ```
5. *(Optional)* Rather than putting each plot in a new figure, use subplots using `subplot` or `tiledlayout`. Label each plot! We will talk about these plotting functions next class.
6. *(Optional)* Implement a function, `generate_circle(x, y, rad)` that takes in the coordinates of the circle's center and its radius, and generates a 256×256 logical matrix, and use it for parts (a) and (b). Alternatively, use it to generate arbitrary circles and visually demonstrate the inclusion-exclusion principle by `xor`-ing all the circles.

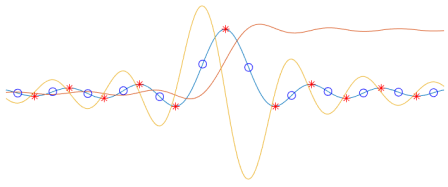
Calculus time! Hopefully the last homework assignment should motivate the use of functions to avoid code duplication.

1. Write a function `deriv(y, x)` and `antideriv(y, x)` which take vectors `y` and `x`, and which perform numerical differentiation and integration on the function $y = y(x)$. These should each output vectors of the same length as the input; you may need to pad your result. You did this already; now turn it into a function.
2. Write a function `switchsign(x)` which takes a vector `x` and returns a logical array with the same length as `x` that is true when `x` switches sign. E.g., for `x=[2 3 -1 0 -1 5]` it would return `[0 0 1 0 0 1]`. Do not use a `for` loop.
   (Hint: One way to vectorize this is to use the `sign` and `diff` functions. Another way is to write conditions on the vector and use a shifted version of itself. You will probably need to pad the resulting vector to make it the same length as the original vector; one way to do this is to repeat the first element of the resulting vector twice.)
3. Write a function `extrema(y, x)` that uses the first derivative test to find local extrema (minima and maxima). Recall that local extrema on a differentiable function occur when $y'(x)$ changes sign. Use your `deriv` and `switchsign` functions here. The output of this function should also be two vectors: one representing the y values of the extrema, and one representing the corresponding x values.
4. Write a function `inflections(y, x)` that uses the second derivative test to find inflection points (i.e., when $y''(x)$ changes sign).
5. Let x be a 1001-point vector linearly spaced from -2π to 2π. Use `sinc` to generate $y = \mathrm{sinc}(x)$ over this interval. Generate the following variables using the functions you just wrote:
   (a) `y_antideriv`, the antiderivative of y
   (b) `y_deriv`, the derivative of y
   (c) `y_extr` and `x_extr`, the coordinates of the local extrema of y
   (d) `y_infl` and `x_infl`, the coordinates of the inflection points of y

   Plot these using:

   ```
   plot(x, y, x, y_antideriv, x, y_deriv, ...
        x_extr, y_extr, 'r*', x_infl, y_infl, 'bo');
   ```

   Title your plot. It should look something like:

   
6. Repeat step 5 on a different function $y_2(x)$ of your choice.