Math 260, Fall 2020
*Last updated: 9/2/20*

# HOMEWORK 3

DUE FRI. SEP. 11

**Exercises.** (Nothing to submit here).

**E1.** Suppose you have a matrix $A$ represented as a list of rows.

a) Write a function `transpose` that returns a new matrix (i.e. a new list of rows) that is the transpose of $A$. Do so using a list comprehension.

b) Test what the following code does:

```
a = 1
b = 2
a, b = b, a
```

c) Write a function `transpose(a)` that transposes $A$ 'in-place', modifying the data for $A$ directly (with no return). *Hint: you may want to use (b) to simplify your code.*

---

**Programming problems.**

**Q1 (setup for Gaussian elimination).** You'll be implementing a solver for $Ax = b$ in the next homework. This week, some preliminary steps:

a) Write a function that does back substitution, creating new space for the result.

b) Write a function `residual` that computes the residual $Ax - b$ given $A, x$ and $b$, creating new space for the result (return a new list).

c) Consider trying to solve $Hx = b$ where $H$ and $b$ are given as `hilb` and `b` in the funcftion `error_example`.

I have supplied a numerically computed solution to $Hx = b$ as `x_comp` and the true solution `x_true`. The computed solution uses `numpy`'s linear system solver, which tries to be as accurate as it can.

Calculate the size of (i) the absolute error and (iii) the residual using your code in (b). For the 'size' of the absolute error, use

$$\max_{1 \le i \le n} |x_i - y_i|$$

i.e. the largest component in absolute value. Do the same for the residual, i..e. look at

$$\max_{1 \le i \le n} |r_i|.$$

Have your code output these **two** values with reasonable formatting (scientific notation is suggested). How do the two numbers compare?

*Note:* The `abs(x)` function computes $|x|$ and `max(arr)` returns the max. value in the list.

**Q2.** In this problem you will implement the bisection algorithm along with some features to make it more like the sort of function you'd find in a library like `numpy`.

Some template code is provided if you want to use it.

a) Write a function `find_zero` that finds a zero of the function $f(x)$, starting with a bracketing interval $[a, b]$. Your function should include the following properties:

- bisection returns **both** the estimated solution and the number of iterations taken.
- Make `tol` a default parameter with a reasonable value.
- `find_zero` checks that the initial $[a, b]$ really is a bracketing interval and raises an exception if it is not.
- `find_zero` has a default parameter `show=False`. If true, then it prints the estimate $c$ (the midpoint) at each step and the iteration number, something like

```
1  1.3467
2  1.0456
3  1.0233
```

  This is 'verbose output' that a user of the function might want to see, but should be hidden by default. (Unlike temporary print statements for debugging, this gets to stay in the code).

b) Test your function on the following examples (I've put some in the template code):

i) $f(x) = x^2 - 9$ with $[a, b] = [1, 4]$

ii) $f(x) = x^2 - 9$ with $[a, b] = [2, 4]$

iii) $f(x) = \sin(x)$ with $[a, b] = [3, 4]$

iv) $f(x) = \sin(x)$ with $[a, b] = [3, 10]$

Find a solution with an (absolute) error at most $10^{-6}$ if possible and state the number of iterations required. Note that you can use the `numpy` or `math` modules for the math functions here. Briefly describe the results in a comment.

(No specific requirements on the code for part (b) here).

c) Eventually, taking more steps will stop improving the approximation. How many iterations are required to find the zero in (i) with the given interval to the maximum possible accuracy, and what is the error? (put in a comment along with part (b)).

(Note: you don't have to write automated search code to do this; I'm looking just for the answer. Your code with `show=True` may be useful).