Final Project Due Date Extension: Devon Webb

The date I would like to extend my final project to is December 1.

The project I am accomplishing is to implement a version of the algorithm called Square Root SAM described in the paper found at this link: https://www.cc.gatech.edu/~dellaert/pub/Dellaert06ijrr.pdf. The Square root SAM algorithm is a least squares approach to solving the least squares problem using a cholesky factorization of a sparse jacobian matrix. I currently have a 1D of this problem working in C++ and it involves solving the following equation. (taken from the above article).

$$x^* = \mathrm{argmin}_x \| Ax - b \|$$

Where b contains odometery and landmark measurement data recorded by a robot, x contains all poses that a robot reaches along it trajectory as well as the location of each landmark in reference to the world plane, and A contains jacobians relating the poses and landmarks to the various odometery Solving this problem involves first redefining the problem as

$$x^* = \mathrm{argmin}_x \| A'Ax - A'b \|$$

and then doing a cholesky decomposition of the, now square, matrix A’A which we define as I. This I matrix is factored into a LL’ where L is a lower Left hand matrix. Which is then used to solve for the argmin value of x.

For 1D with 25 poses and 5 landmarks you end up with an A matrix that is 30 x 85 and do a cholesky decomposition on a 30 x 30 matrix. But, because a 1D example only contains linear jacobians, the A matrix is filled with only -1, 1, and zero which is not very interesting. So to finish the project I am continuing the square root SAM proccess on a 2D example with non linear jacobians which should lead to a more interesting least squares problem than the linear case. For this 2D example I have generated a b vector with “measurement data” that corresponds to a predefined ground truth for robot poses and landmark positions. I have also calculated the Jacobians for 2D. The last thing I have to do for my project is write a function that inserts the Jacobian into the A matrix properly for square root SAM. Then the algorithm for solving least squares should be the same as 1D.