Prelab –

Part 1:

Is this section we are asked to compute the bidiagonal reduction to the random 10\* 5 matrix “A”. This is done using the “bidiag\_reduction” function written in a previous lab session. From this we get the products B, U and V. Next, we are asked to produce the singular values of the matrices “A” and “B” where “B” is the upper bidiagonal matrix produced from “A”. Comparing the singular values from these matrices it is found that U1 and U2 are unrelated. However, it is found that the Sigma values for each of the matrices is the same and that the first column of V1 and V2 are the negative values of each other, all other values in V1 and V2 are unrelated also.

Part 2:

In this part, we formulate the singular values of matrix A by simple multiplication. We do this by multiplying the output of Bi-Diagonal Reduction of ‘A’ with the Singular values of B which we assume to have obtained. Sigma values of B which is an Upper Diagonal Matrix is the same as that of A.  
So the New Unitary matrixes are NewU with m\*m dimension and NewV with n\*n Dimension and NewSigma is the New Sigma value.

Lab –

In this exercise, we will formulate SVD using given Rotational Matrices.

Part1:

For that we create an identity matrix R in the dimension m and obtain theta, which is the tan inverse of the value along the Y axis which is beta by the value along the X which is alpha. The value in the rotational matrix c is given by cos of theta and the value of s is given by Sin of theta. Once we obtain the values of c and s we assign it along the axis of rotation in the Matrix and this at the end returns a Rotational Matrix R.

Part2:

In this part, we follow the pseudo-code to carry about SVD in Golub Kahan Method following the algorithm 1c.

In the first step, we identify the Block matrix B22 by setting the matrix indices to iLower and iUpper. We then create a temporary Matrix which is the Dot product of the Diagonal Block and its Transpose. In step 2, we determine the shape of the temporary matrix in order to create a new Matrix ‘C’ in the dimension 2\*2 which is the right bottom of B, to optimise the function. In step 3, we then compute the eigen values of C. In the next step, we set the eigen value which is closest to C22 as mu. In step 5, we change the value of alpha and beta with respect to the new iLower value which is set to k. In step 6, a for loop is run in the range of k from iLower to iUpper-1. We then perform rotation over the row axis first and preform rotation using the function ‘givens\_rot’ and then obtain B which is the dot product of B with the transpose of the rotational matrix and V which is the dot product of itself with the transpose of the rotational matrix. Locations of alpha and Beta are set. We then preform rotation over the other axis and at the end implement an if loop for k in the range till iUpper-2 and set new locations to alpha and beta. We then return the values B, U and V.

Part 3

We create a function called ‘golub\_reinsch\_svd’. The Bidiagonal reduction of A is done and the shape of the upper bidiagonal matrix is determined to determine the sub blocks of the matrix. The counter value is set to 0 first and the lowest value of p is always set to 0 and for q it is set to 0 first. We implement a while loop to run the counter in values less than the maximum number of iterations which is user defined. We then set a range for I from 0 to n-1 and check if the values are less than the values of eps, and if not we turn them to 0. After carrying out the zeroing steps we obtain a sub block matrix B33. We first take the sum of the diagonals in the opposite diagonal, and we obtain the opposite diagonal by using the ‘fliplr’ function. If the sum is less than eps we increment q. We then reach till the value of q where it is n-1 and the stop and obtain the diagonal value of B and store it in S. We then return the values of S, U, V and the counter value.

Post Lab –

Part1:

In the first part, we try to test 4 factors that is if the product of the diagonal matrix, the unitary matrices U and V returns the input matrix A. The second and the third tests are the product of the unitary matrices U and V by its transpose will give you back an identity matrix. The final test is to check if the diagonal elements are positive. If all the tests pass it will output as true or it would output in which test it has failed.

Part2:

Fixing the bug in the function of previous lab which works only for m>n by taking the transpose of the matrix and swapping the values of the unitary matrices.

Part 3:

We compare the time of execution of our function with the built-in function by importing ‘timeit’ and measuring the elapsed time and returning the elapsed time values at the end.

The values we got for our function was 0.003495s vs that for the built in was faster and was 0.0001724s. By this comparison, we believe further optimisation is required.