Observations and Outputs:

The first process rank 0 fetches all the coefficients from the input text file. It then calculates the d value. The process passes the data to process of rank 2 to calculate the root values. The type of root depends on the value of d as it is square rooted in the calculation process.

Finally the roots are passed to process 3 to save to the output.

In total,l process rank 2 has n number of send and receives where n is then number of rows in the text file.

This is also more efficient than without using MPI or any parallelism because task 1 can continue to fetch coefficients and calculate d while process rank 2 and rank 3 can perform operations on previous rows.

Creating of custom data type in MPI:

The key learnings is that you can pass structs in MPI but creating a custom MPI Type by specifying its primitive data types, displacements between internal data and also the length of each data inside.

```
MPI_Datatype create_root_datatype(root the_root) {
  // The number data type is 1 and just float
  MPI_Datatype array_of_types[1];
  array_of_types[0] = MPI_FLOAT;

  // The number of blocks length are 1 and it is 2 because 2 blocks.
  int array_of_blocklengths[1];
  array_of_blocklengths[0] = 2;

  // The distance between the first and second block
```

```
  MPI_Aint array_of_displaysments[1];
  MPI_Aint address1, address2;
  MPI_Get_address(&the_root.real, &address1);
  MPI_Get_address(&the_root.img, &address2);
  array_of_displaysments[0] = address2 - address1;
  MPI_Datatype mpi_root;

  MPI_Type_create_struct(1, array_of_blocklengths, array_of_displaysments,
                         array_of_types, &mpi_root);

  // Need to commit structure type in MPI
  MPI_Type_commit(&mpi_root);
  return mpi_root;
}
```

Input

```
3
a b c
1.0 -4.0 3.0
1.0 3.0 1.0
2.0 2.0 1.0
```

Standard Output:

```
Row 0, Value d is 4.00.
Row 1, Value d is 5.00.
Row 2, Value d is -4.00.
Row 0: Coffients are d 4.00, a 1.00, b -4.00
```

```
Row 0: Roots are 3.00+0.00i, 1.00-0.00i.
Row 1: Coffients are d 5.00, a 1.00, b 3.00
Row 1: Roots are -0.38+0.00i, -2.62-0.00i.
Row 2: Coffients are d -4.00, a 2.00, b 2.00
Row 2: Roots are -0.50+0.50i, -0.50-0.50i.
```

Output

```
3
x1    x2
3.00   1.00
x1    x2
-0.38   -2.62
x1    x2    x1_real    x1_img    x2_real    x2_img
          -0.50       0.50     -0.50     -0.50
```