README 1.1

PHASE_REC_GUI VERSION

[Getting Started]

The first thing to do is download the entire **phase_rec_GUI** folder and navigate to it in MATLAB. Since certain features of this software depend on GPU compatibility, it requires the use of a **2011 or later** release. While using the GUI, it is imperative that the user **remains in this folder**, or else the application may crash. Do not remove or rename any of the subdirectories, as they are referenced by name in the application.

For convenience, the user may wish to move datasets into the subdirectory called **datasets**. When the application prompts the user to browse for a dataset to load, the file explorer will start in this subdirectory.

To run the application, simply navigate to the folder indicated above, and type '**phase_rec_GUI**' into the MATLAB command window.

[Data and Algorithm Selection]

The user may type the file path of the dataset to be loaded into the edit box that initially contains "Enter file path …", but will probably find it easier to click the **Browse** button instead. This will open a file explorer window in the datasets folder which will allow the user to navigate to and select a dataset.

Some notes about datasets:

- The dataset must be of type ***.mat**.
- For convenience, the dataset should include a stack of images, a variable called '**lambda**' indicating wavelength, a variable called '**ps**' indicating the pixel size, and a vector indicating the z-depths of the images in the stack.
- The length of the z-vector should be equal to the number of images in the intensity stack.
- **Pixel size and wavelength must be measured in meters.**

Once a dataset has been selected, press the '**Load'** button – this loads the variables of the dataset into the MATLAB base workspace. (Note that this base workspace cannot be directly accessed in the middle of a GUI function.) If variables named 'lambda' and 'ps' exist in the dataset, their values will automatically be entered into the appropriate boxes under 'General Parameters.'

Once the variables have been loaded, they will appear in a dropdown menu underneath the file path edit box. (Note that the file path edit box now only displays the name of the dataset the user is currently working with.) Select the image stack variable from this list and then click the '**Display'** button. The user should now be able to see his/her image stack on the left-side axes. The **slider** allows the user to cycle through the images in the stack, with the text box next to it displaying the index of the image in the stack. The '**Plot Contrast**' button opens a new figure, quantifies the contrast of each image in the stack, and plots it against the indices of the stack. This may be useful for determining the plane of minimum defocus in the stack. The '**Color'** and '**Grayscale'** buttons change the colormap used when displaying the image stack (and result). Note that both plots will change simultaneously when either button is pressed.

The final item in this panel is a dropdown menu that allows the user to choose an algorithm by which phase will be recovered. As of the current version of this software, there are 5 choices of algorithms. Following are some detailed instructions pertaining to each of the algorithms.

(The fearless user may wish to skip the following segments and rely instead on the help buttons (?) scattered throughout the GUI.)

[Algorithm: Standard TIE]

This algorithm requires:

I0 Position – the stack index of the center image *(z = 0)*
I (+) Position – the stack index of the image at a **positive** distance *(z = +dz)*
I (-) Position – the stack index of the image at a **negative** distance *(z = -dz)*
dz – the distance **in meters** between the center image and either of the other two images
Pixel Size and Wavelength, both **in meters**

Laplacian – regularization for solving the Poisson equation
Intensity – regularization for division by intensity

Note that the use of this algorithm requires a certain symmetry in the dataset; that is, it is assumed that an image exists at $z = 0$, and that whenever an image exists at $z = \alpha$, an image also exists at $z = -\alpha$.

This algorithm will produce a single image of the same size as the images in the stack, which will be displayed on the right-side axes.

[Algorithm: Iterative TIE]

This algorithm requires the same input parameters as the **Standard TIE** and also:

Iterations – the number of iterations to perform
Factor – a weighting factor used to change the relative contribution of consecutive iterations' results

The use of this algorithm requires the same symmetry required by the Standard TIE.

This algorithm will produce **a stack** of images of the same size as those in the original stack, displayed on the right-side axes. The index in this result stack corresponds to the resulting phase after that many iterations of the algorithm. The image of **maximal index** corresponds to the **final result**, and the slider in the lower right hand corner can be used to cycle through **intermediate results**. The box next to the slider will display the index of the currently visible image in the result stack.

(It is now appropriate to digress and explain the following panel.)

[Alternate & Extra Parameters]

If the dataset includes a vector indicating z distance values, this may be an easier means of entering data than filling out each box in the 'General Parameters' panel individually. To load and use such a vector, first click the **radio button** to demonstrate intent. Next choose the appropriate variable from the dropdown menu and click the nearby '**Load'** button.

The vector will be considered valid in the context of **Standard TIE** and **Iterative TIE** if it obeys the symmetry requirement stated above.

If the vector is valid, the user will be prompted to **choose a dz value** from a second dropdown menu. **Note that a value immediately appears when the vector is loaded – this value has NOT been chosen. The user must click the dropdown menu and make a selection.** When a value has been chosen, the 'I0 Position', 'I (+) Position', 'I (-) Position', and 'dz' fields will automatically be filled in with appropriate values.

The other option in this panel is to reflect the image before solving the Poisson equation. This procedure involves reflecting the original image about the vertical axis to obtain a new image of twice the width. This image is then reflected about the horizontal axis, producing a final image **four times as large** as the original image. This helps eliminate boundary value errors in the recovered phase.

(Back to algorithms.)

[Algorithm: Higher Order TIE]

Unlike the previous algorithms, this one **requires a z-vector** to run. Only **Pixel Size** and **Wavelength** are required from the General Parameters panel. **Laplacian** and **Intensity** regularization are also required, as before. New parameters specific to this algorithm include:

Num. Images – the number of images to use in either direction from the center *(the center is identified by a z value of 0)*

Index Step – images used in algorithm will be:

```
center – (Num. Images)*(Index Step)
        . . .
center – 1*(Index Step)
center
center + 1*(Index Step)
        . . .
center + (Num. Images)*(Index Step)
```

Degree – degree of the polynomial to interpolate intensity to; for well-defined interpolation to **degree *d***, there must be ***d+1* images** being used

Interpolation – regularization for interpolation

Like the Standard TIE, this algorithm produces a single result image of the same size as the input stack images, displayed on the right-side axes.

[Algorithm: GP Regression]

In the GP TIE algorithm, we first perform **Gaussian process regression** over the defocused intensity images [in Frequency domain] to estimate the intensity axial derivative; then we recover phase from the axial derivative by the transport intensity of equation (TIE). GP TIE alleviates the nonlinearity error in the derivative estimation by using the prior knowledge of how intensity varies with defocus propagation in the spatial frequency domain. It doesn't require the intensity images to be equally spaced, so the input intensity stack can be exponentially spaced, which is shown to be an efficient scheme to transfer the phase information into the measured intensity.

Like the Higher Order TIE, this algorithm **requires a z-vector** to run. It also requires an additional variable called '**Bins**'. This values specifies the number of bins we divide the frequency domain into – a large number of bins will result in a finer grid, i.e. more computation time and a more accurate result.

(It is now appropriate to digress and explain the following panel.)

[Regularization Sweep]

For any of the above algorithms, the user may be interested in seeing many results with slight modifications to regularization parameters. In order to do so, the user must change the selection in the dropdown menu on the '**Regularization Sweep**' panel from '**(None)**' to **some other choice**.

The box on the TIE panel corresponding to the variable chosen will become inactive, and any value in it will be ignored. The user must then input **an expression** in the edit box on the 'Regularization Sweep' panel. Given the following line: '*svals =* ___________________________;' the user must input an appropriate expression that would fill the blank. This expression will be evaluated to find the list of regularization values to run the selected algorithm on. Note that this feature is only available for **Standard TIE**, **Iterative TIE**, **Higher Order TIE**, and **GP Regression**.

Since Standard and Higher Order TIE only produced a single image, they will now produce a stack of images corresponding to each regularization value. If the Iterative TIE is chosen, however, only the final result (the end image of the stack) of each run will be saved and displayed at the end. Regularization values can be seen in the box next to the slider in the lower left hand corner.

(Back to algorithms.)

[Algorithm: Gerchberg-Saxton]

There are two versions of this algorithm, determined by the dropdown menu on the '**Gerchberg-Saxton**' panel in the upper right hand corner of the GUI. The default version is the '**Single-plane propagation**' version, which takes a center plane intensity and an initial phase guess of zeros, propagates to a plane at positive z, replacing intensity with that of the image at +z, and back to the center, replacing the intensity of the result with that of the center image. Then this process is repeated for the image at -z.

This version of the algorithm requires all of the parameters in the '**General Parameters**' panel. Entry of the center, +z, and -z images can be simplified by **loading a z-vector** in the '**Alternate & Extra Parameters**' panel.

The other version is the '**Multiple-plane propagation**', which takes an additional argument, '**Num. Images**'. In each step, the algorithm propagates the specified center image to *(Num. Images)* images with positive z in the stack, replaces intensity, and propagates all images back to the center. The results are wrapped to 2π and then averaged, and the process is repeated using *(Num. Images)* images with negative z. Note that this version does not need all arguments in the 'General Parameters' panel. **However, this version, in the manner of Higher Order TIE, requires a z-vector to run.**

Both versions of the Gerchberg-Saxton algorithm require the user to specify the number of iterations to perform. This is the slowest algorithm, as many iterations on large images can take several hours to complete, depending on computer specifications.

This algorithm will access the computer's GPU, if possible, for faster computation.

[Running and Saving Results, Resetting]

Once all appropriate parameters have been entered, the user can click the '**Run**' button in the lower right hand corner. Depending on the algorithm selected, the user may need to wait for some time before the results are displayed. Once the algorithm terminates, a variable called '**RESULT**' will exist in the base workspace. If the user would like to perform any operations on the result not supported by the GUI, this variable can be used freely from the MATLAB command window.

If the user would like to save the result, he/she can input a name into the text box on the '**Save Results**' panel, and click '**Save**'. This will produce a ***.mat** file in the '**results'** subdirectory of '**phase_rec_GUI**' with a single variable '**Result**'. It is up to the user to organize files in the 'results' subdirectory – all saved *.mat files will simply be put into that folder.

If the user would like to choose a different dataset, he/she must simply click the 'Reset' button in the lower right hand corner. Note that this will clear the base workspace of relevant/internal variables and reset the display axes.

[Troubleshooting]

The earliest release of MATLAB compatible with this software is R2011a. Earlier releases of Matlab do not have proper GPU support for some functions.

GP Regression does not have proper documentation in its code yet, and as a result, error messages do not provide much help. This will be improved in a later update.

Expressions for various parameters must be entered in such a way that MATLAB can convert them from strings to doubles. For example, "10e2" will be recognized as 100, but "10^2" will not; "0.5" will be recognized as a valid input, but "1/2" will not. The application converts an expression from string to double, but DOES NOT evaluate the expression unless otherwise stated.

If nothing works, check the current directory. THE CURRENT DIRECTORY MUST BE '**…/phase_rec_GUI/**'. MANUALLY CHANGING DIRECTORIES WILL CAUSE THE APPLICATION TO CRASH.

[About]

This application was created by Gautam Gunjala, B.S. Electrical Engineering & Computer Science, B.S. Engineering Mathematics and Statistics, 2016, at the University of California, Berkeley. It is affiliated with the Computational Imaging Lab and Prof. Laura Waller (http://www.laurawaller.com). This software is intended to be open-source. Please do not attempt to profit off of this software, and please give credit where it is due. Thank you!

Back-end software contributors include:
Gautam Gunjala *(myself)*
Prof. Laura Waller
Aamod Shanker
Daniel Shuldman
Jingshan Zhong

[Bug Fixes]

-- Version 1.0 Release: 10/8/2014 –

-- Version 1.1 Release 10/16/2014 –
- GP Regression bugs fixed, now compatible with regularization sweep

(If you encounter any problems, please email me at gautam.gunjala@berkeley.edu*)*