**README**

**Introduction.**

This is a repository containing results I have obtained from MATLAB during an attempt to make a complex valued conditional GAN.

Below is the experimentally obtained results, when attempting to compare the conditional GAN using complex manifolds, vs real ones. Complex is shown on top, real ones shown below.

Note that the one shown below is the original from MathWorks, and has much less accuracy than the one that I have created, which apart from the change in training loop is the exact same algorithm as the one that they created in the first place.

**Rationale (i.e. Why do this?).**

The reason why I have attempted to do this, is because at the time of attempting this experiments GAN’s including Bayesian GAN’s like this one, were known to be computationally expensive to train and very time consuming to use.

Having done some previous experiments on less complicated types of Bayesian networks where the accuracy was increased and training time reduced, I decided to try and do one on a conditional GAN to see what would happen if I did attempt to use complex manifolds instead of real ones.

This is also because of a very recent development in game theory which as far as I know lies at the heart, of how exactly a GAN actually works whether it is a Bayesian one or not, due to the interaction between the discriminator and the generator components of this type of algorithm.

Finally it is because having done some research prior to this I have found out that real manifolds used in the original algorithm, do not actually behave very well with regards to convergence over a measure, if the manifold has more than 4 dimensions where as complex manifolds do not have this particular problem.

From (9) real manifolds over a measure are semi continuous and only bounded below or above, not both in general which suggests that the training process may not work as well as it would, if the alternative is used as demonstrated here.

**Methods:**

This was done using a modified version of a tutorial from MathWorks (authors of MATLAB), by simply converting all of the numerical components to complex valued data, using the complex command in MATLAB. (1). It is not known why but using this method maps the numerical values to floating point variables (for the real part), and floating point imaginary with value 0.

Note that it was not actually possible to encode all of the data involved in this particular experiment to complex valued data, as not all of it is numerical and some of it is categorical, so at the time when I did this I did not know how to do that and I am not sure what difference if any that would have made.

The training loop used to do this uses Wirtinger derivatives, and so this can only really be used if there exists a closed form partial derivative, of the function being differentiated which is very similar to the case with real valued partial derivatives. (5).

**Results.**

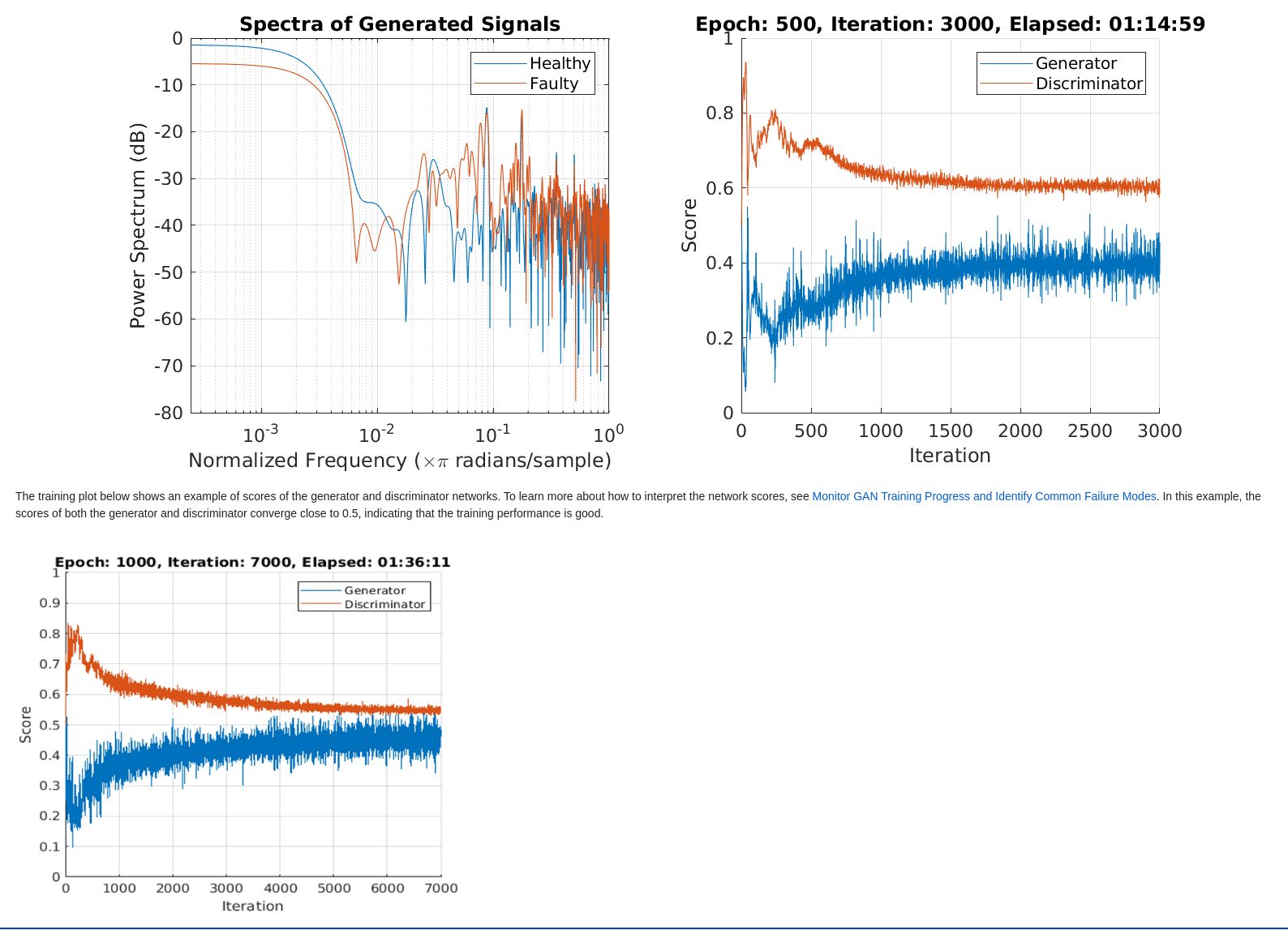


Figure 1, results obtained from the experiments I have attempted when I changed the number of Epochs to 500 instead of 1000. Note that the above experiment needs less time to gain a higher score on both the discriminator and the generator, than the original.

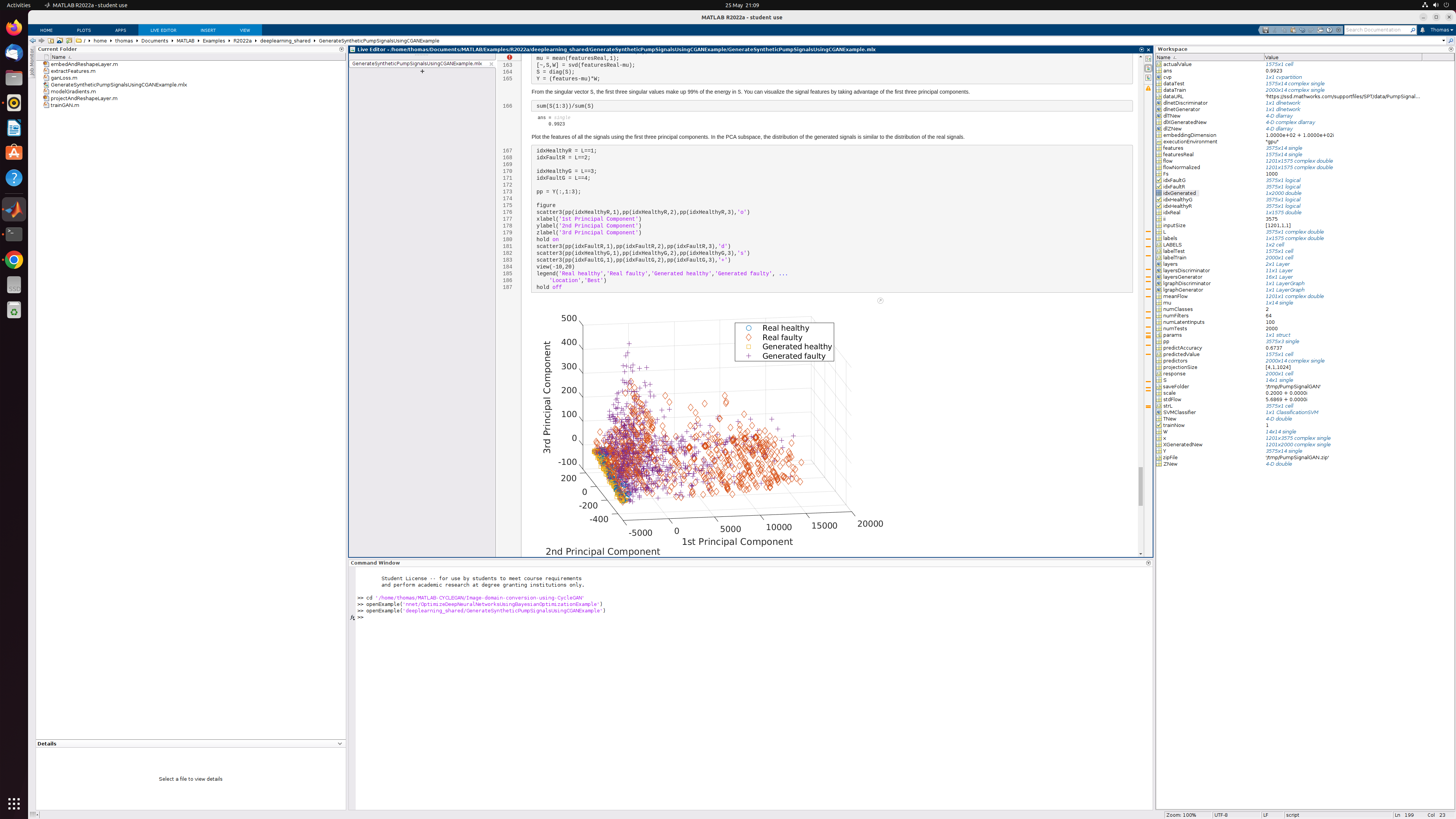
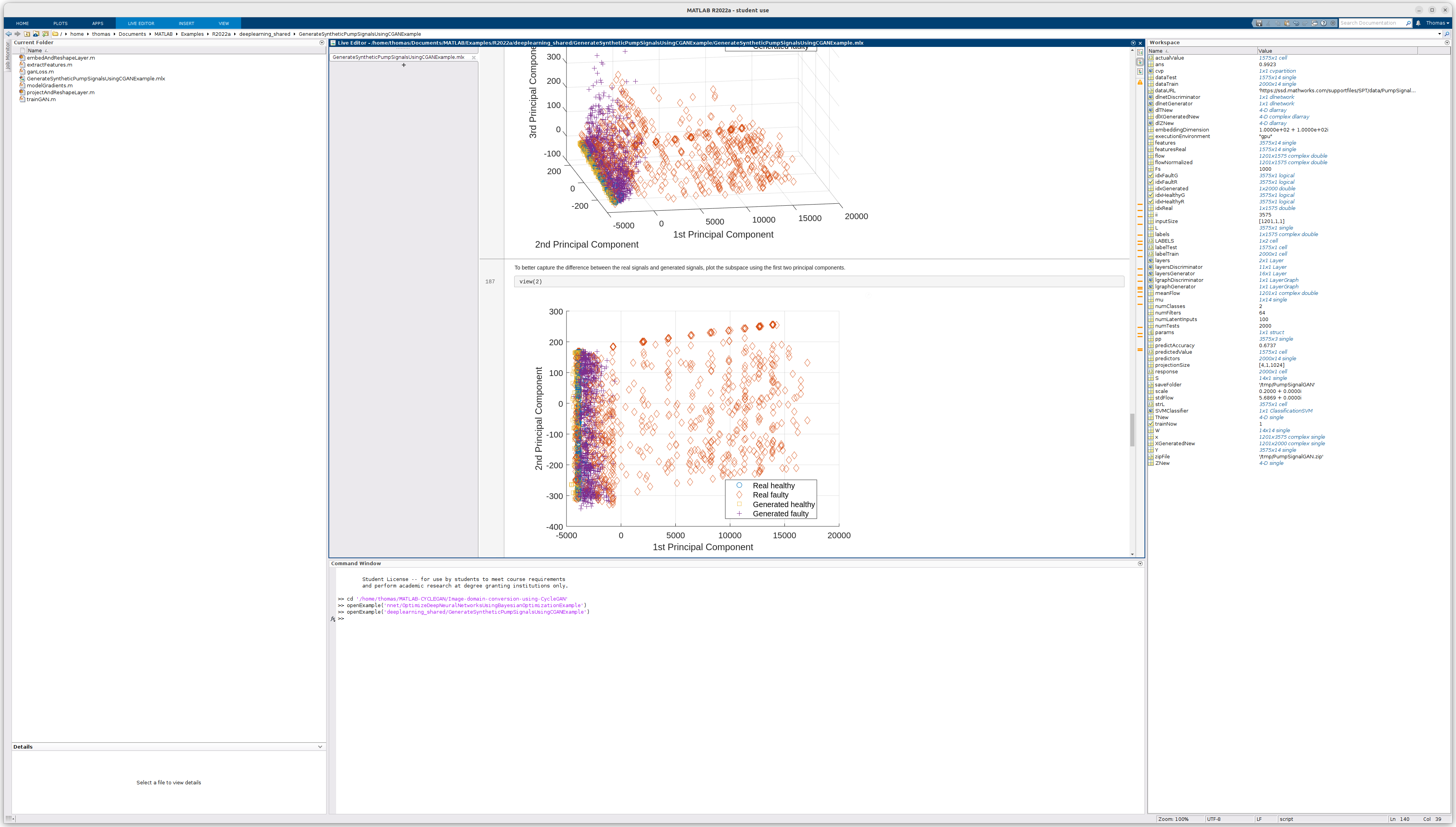
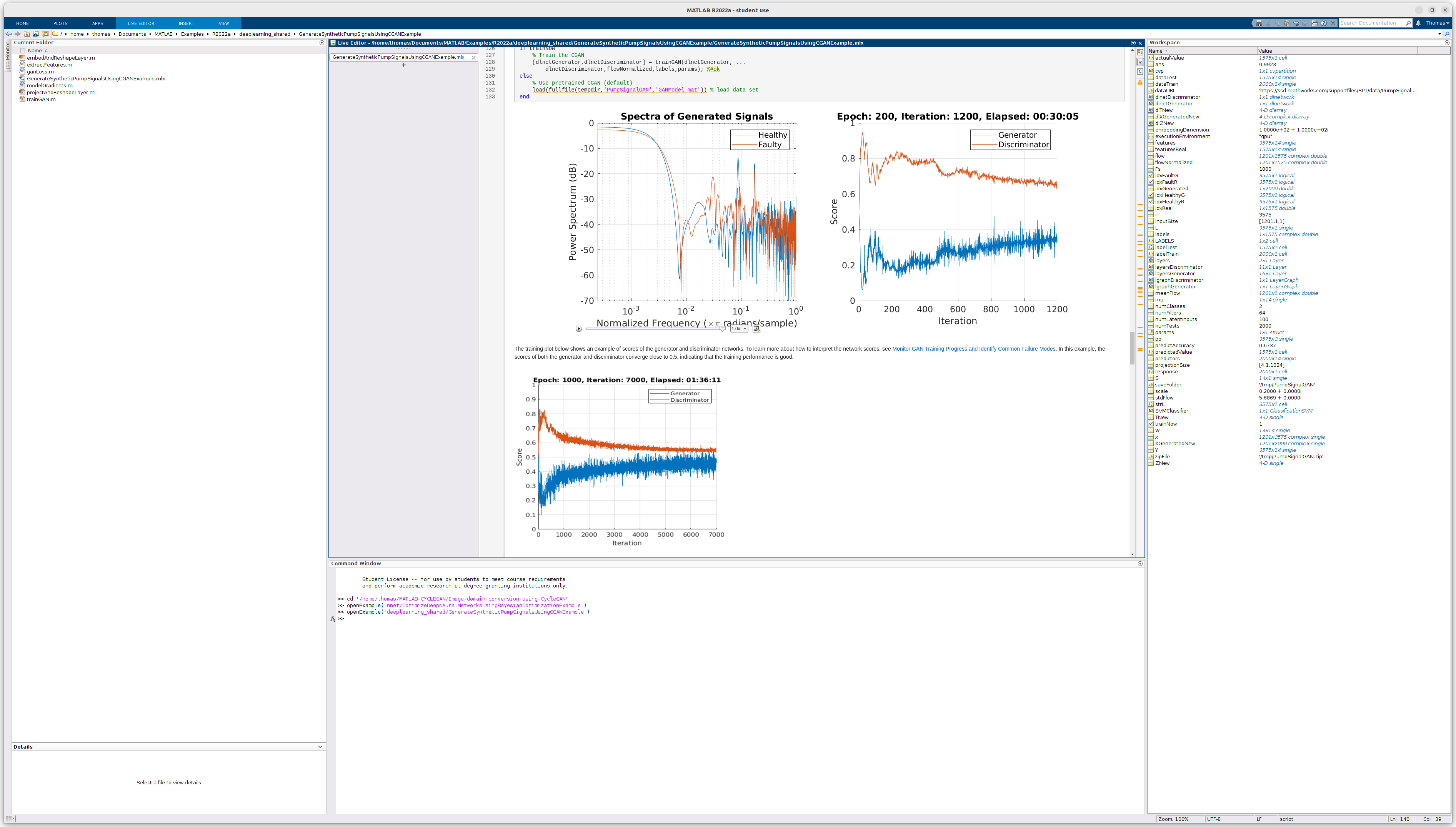


Figure 2, clustering PCA results from this attempt at making the GAN work.

Above is further results obtained, although this one was done with fewer epochs. Figure 3.

****

Above is an example of what happens if this is attempted with 200 Epochs instead of the above 500 that were later attempted, or the real valued version which uses 1,000 Epochs to do.

**Discussion of results.**

What has been found here is that the complex valued version of this algorithm requires less training time overall and fewer iterations in order to train the algorithm, and it has also resulted in higher scores as shown in figure 1.

However this also resulted in longer training time per iteration to get a result out, and so the reason for that may well be down to the algorithm used, to calculate the derivatives and integrals used in training this GAN. Integration is used in this algorithm to attempt to calculate probabilities from the probability distributions, although this can be expensive to do and again it can be easier to do with complex manifolds than in real ones, especially on a high dimensional problem like this one.

**Conclusion.**

I have attempted to find out here what would happen if complex manifolds are used to train a Bayesian GAN such as this conditional GAN, and what I have found is that using this particular implementation, the training time is reduced and the accuracy increased.

The major purpose of this was to see whether this was actually possible using these particular methods, because it has worked before on other types of algorithms such as a variational auto encoder or a subspace KNN used for image classification.

**Future work.**

I would suggest here that this should be retested using alternative methods for the analysis parts of the training loop (e.g. integration used for probability calculations), and potentially by using alternative methods to encode the data, such as that the imaginary values do not necessarily start at zero.

I would also like to suggest that this should be tested on other types of GAN’s such as a cycle GAN, or even attempting to make this work on a Quantum computer such as IonQ’s Aria or Harmony in order to further reduce the computational complexity, as these types of algorithm are normally very slow to train and often take a lot longer than an hour or 2.

References:

<https://uk.mathworks.com/help/signal/ug/generate-synthetic-pump-signals-using-conditional-generative-adversarial-network.html> [online] (accessed 30/05/2022). (1).

<https://arxiv.org/abs/2204.02277> [online] (accessed 30/05/2022). (2).

<https://courses.maths.ox.ac.uk/course/view.php?id=156> [online] (accessed 30/05/2022).(3).

Appropriate resources from: <https://uk.mathworks.com/help/index.html> [online] (accessed 30/05/2022). (4).

<https://onlinelibrary.wiley.com/doi/pdf/10.1002/0471439002.app1> [online] (accessed 30/05/2022).(5).

<https://courses.maths.ox.ac.uk/course/view.php?id=182> [online] (accessed 30/05/2022).(6).

<https://courses.maths.ox.ac.uk/course/view.php?id=156> [online] (accessed 30/05/2022). (7).

<https://courses.maths.ox.ac.uk/course/view.php?id=148>[online] (accessed 30/05/2022). (8).

Ye-Lin Ou, Bang-Yen-Chen Biharmonic Submanifolds and Biharmonic Maps in Riemannian Geometry. (9).