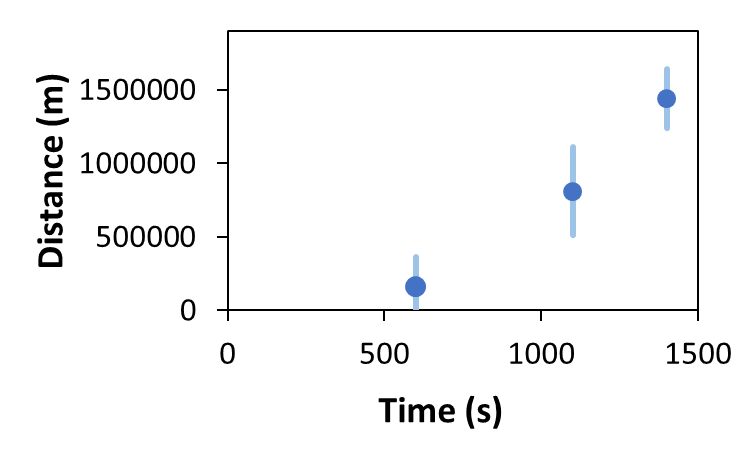
Why should you use PEUQSE to get parameters from observed data? A few lines of code will give you realistic estimates and some graphs.

Consider a situation where we have three observed experimental data points with uncertainties:



Their values, including uncertainties, are:

160500 +/- 200000

810500 +/- 300000

1440500 +/- 200000

Consider that this situation is known to be described the following equation:

y=(x-a)^2 + b

Where we know that the physically realistic values of “a” and “b” are:

a is expected to be 200 +/- 100 (this is the 1 sigma confidence interval)

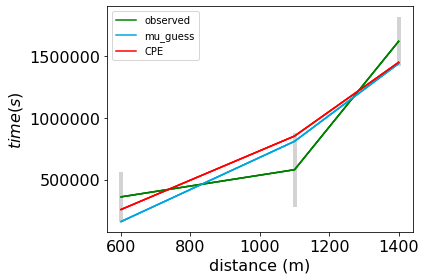
b is expected to be 500 +/- 200 (this is the 1 sigma confidence interval)

If one tries to do a sum of squares fitting (conventional parameter estimation, CPE), we will not get realistic values for “a” and “b”. We get **a = 255, b = 139153**. The value for “a” is fine, but the value for “b” is not realistic.

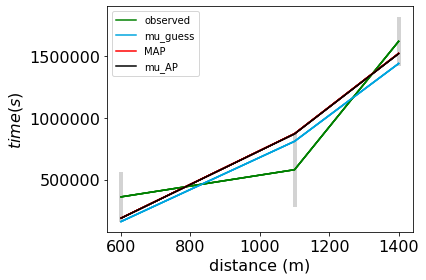
However, if we do a Bayesian Parameter Estimation (BPE), what CheKiPEUQ is designed for, then we get the following answers: **a = 166 +/- 57, b= 509 +/- 198**. Where these errors are the 1 sigma credible intervals. Notice that now ***both*** of the parameters have physically realistic values. We even have error bars that took into account the uncertainty! The covariance matrix for the parameters is also provided, so that the correlated uncertainties of estimated parameters is not lost.

**How good is the match in this example?**

The fitting (CPE) gives the red line below:



The Bayesian Parameter Estimation gives the black line below (and the red, not explained here):



We see that for this example, the CPE result from fitting and the BPE results do not look very different from each other. Both parameter estimation methods manage to stay in the error bars, yet the BPE result has a far more physically realistic pair of parameters! This is the main purpose using PEUQSE BPE: it will tend to give more realistic parameter estimates, and can even give a type of uncertainty (called credible intervals) on the final estimates.

Here is the code that was required after making the model equation:

import PEUQSE

import PEUQSE.UserInput as UserInput

UserInput.model['InputParameterPriorValues'] = [200, 500] #prior expected values for a and b

UserInput.model['InputParametersPriorValuesUncertainties'] = [100, 200] #1 sigma, in this example not correlated, but a covariance matrix can be used instead.

UserInput.model['simulateByInputParametersOnlyFunction'] = simulation\_model\_00.simulation\_function\_wrapper #This just points to the User created model equation.

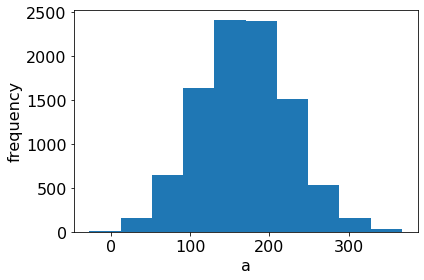
PE\_object = PEUQSE.parameter\_estimation(UserInput)

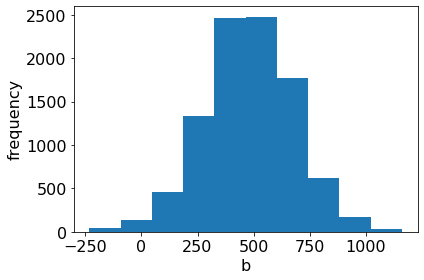
PE\_object.doEnsembleSliceSampling()

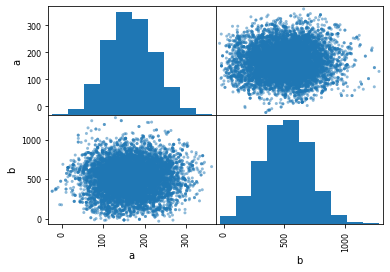
PE\_object.createAllPlots()

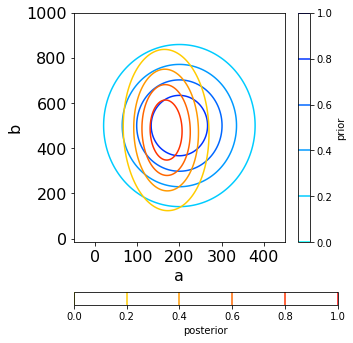
There is a logfile generated called mcmc\_log\_file.txt (along with other files in the directory).

You will also get the following plots, some of which can be further customized, such as removing the bars from the contour plots.









We can see that in this example the position and uncertainty in “a” narrowed more than that of “b”.

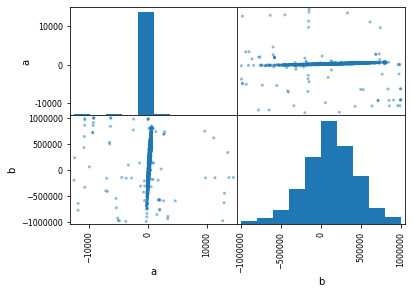
Additional Tests & Info on The two types of mcmc  
There are two types of mcmc samplings possible in PEUQSE. The **EnsembleSliceSampling** will be faster for many higher dimensional problems by needing fewer (but more sophisticated) samplings: there are normally almost no rejections. The **MetrpolisHastings** routine is what PEUQSE was originally built with, and makes more discontinuous jumps: it is recommended that ESS be tried before MH.

If we compare the outputs and performance from 00a1 and 00a2, above, we see that the outputs are similar but that the ESS is a bit slower than the MH.

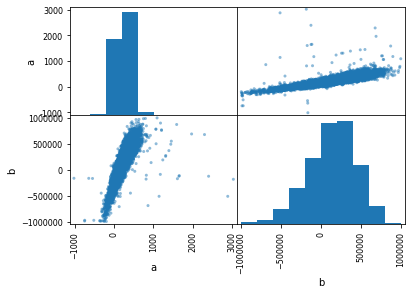
Let’s look at the harder case of 00c1 and 00c2 (which is uniform distributions for each sample) to demonstrate that ESS requires fewer samplings to arrive at the final distribution. First, let’s note that with ESS, we will be using 8 walkers (8 samplings per iteration) with ~0.05 seconds per iteration. That means around 0.00625 seconds per sample. In contrast, the MH requires around 0.0013 seconds per sampling. This means the MH is ~5 times faster per sample, for this system. Still, it is possible the ESS will be better. Let’s look at the following scenarios and outputs. We will use the mu\_AP rathe than the MAP, since what we’re looking for is convergence rather than finding the MAP. Comparing c2 and c5 in the table and the images below, we see that for this ‘harder to sample’ system: the MH method (which uses more rejections) has sparse and “slow” sampling relative to the ESS method, but more focused. However, we should note that while this system is hard to sample, it’s still got a single dense region, so it is the type of problem where we expect ESS to do better even despite being low-dimensional. This problem demonstrates the difference between ESS and MH sampling at a qualitative level. 00c2 and 00c6 have converged enough that they won’t change much.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Example: | Type of MCMC: | Samples: | Est. time: | a | b |
| 00c1 | ESS | ~1500 | ~5 seconds | 124 ± 1.7E3 | 123930 ± 3.3E5 |
| 00c2 | ESS | ~10000 | ~60 seconds | 235 ± 1.9E2 | 134329 ± 3.1E5 |
| 00c3 | MH | ~10000 | ~13 seconds | -2.4e+01 ± 1.4E2 | -3.17e+05 ± 1.9E5 |
| 00c4 | MH | ~60000 | ~50 seconds | 311 ± 1.4E2 | 278693 ± 2.3E5 |
| 00c5 | MH | ~100000 | ~80 seconds | 279 ± 1.3E2 | 218384 ± 2.0E5 |
| 00c6 | MH | ~1000000 | ~800 seconds | 243 ± 1.6E2 | 161890 ± 2.6E5 |
| 00c8 | Uniform Distribution Sampling (non-mcmc) | ~1000000 | ~990 seconds | 225 ± 1.5E2 | 125773 ± 2.6E5 |

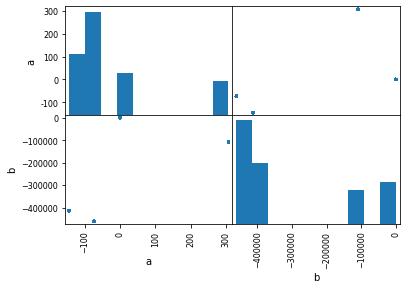
C1 sampling:



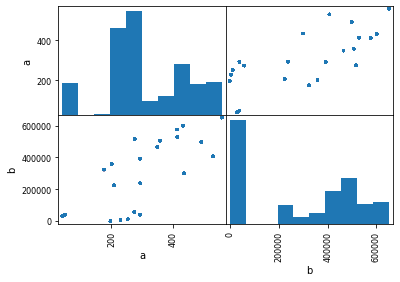
c2 sampling:



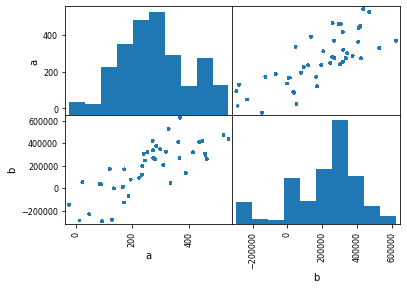
c3 sampling:



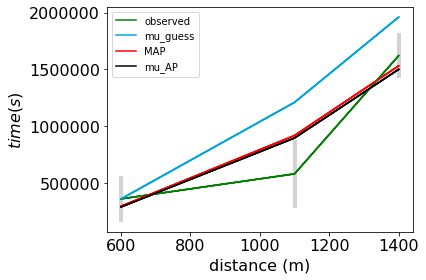
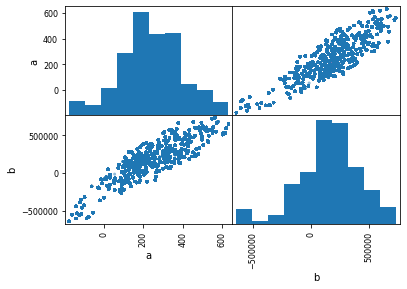
C4 sampling:



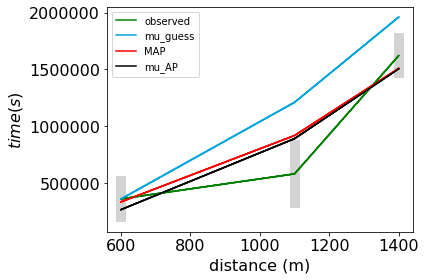
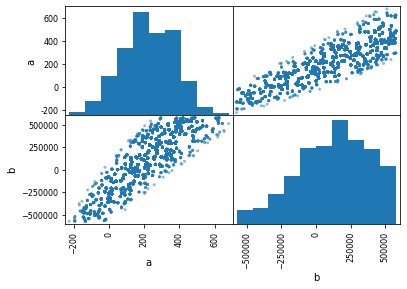
C5 sampling



C6 sampling:



C8 uniform distribution random sampling:

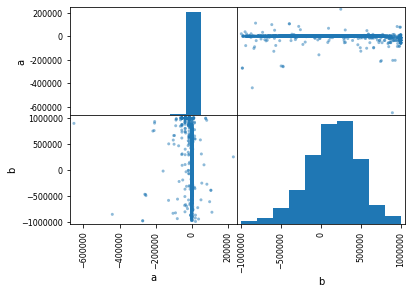
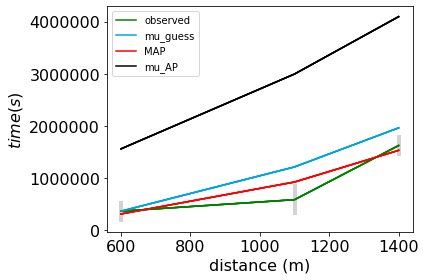


* Quite impressively, C8 gives results like C6. However, this was not arrived at ‘trivially’. With C8, leaving the default settings for the variable UserInput.parameter\_estimation\_settings['multistart\_relativeInitialDistributionSpread'] gives very horrible sampling. Unlike the mcmc, there is no guiding and biasing. In this specific case, we \*\*knew\*\* the HPD interval was not over the full upper and lower bounds of -1E6 to 1E6. So by making the relativeInitialDistributionSpread smaller from 2.0 we were able to get uniform sampling of the *relevant* region. Using 0.10 was *too* small, using 0.50 turned out to be okay. Note that in the general case, uniform distribution sampling and non-adaptive grid based sampling will do a very inefficient job of sampling if given a bounds that result in areas (‘volumes’) orders of magnitude larger than the HPD area (‘volume’). This scales nonlinearly, like d3 for 3 parameters. Still, one could take the HPD interval according to mcmc and then do uniform random sampling in a region that is simply several times that size.

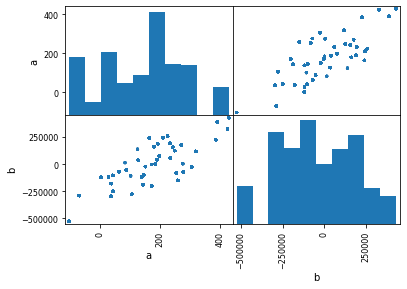
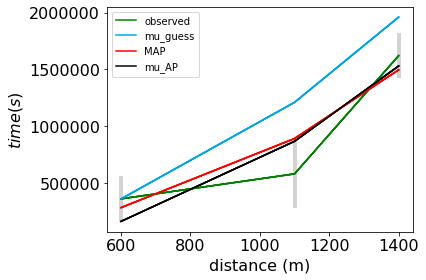
Importance of Filtering.  
For both MH and ESS, we used filtering of the tails of the distributions to avoid a bad effect on the output. Below, let’s take a look at how c2 and c5 would look like without this filtering (there are runfiles for filter removed). We see that C2 (the ESS way) is affected much more badly when there is no filtering – enough that the simulated black line on the right from mu\_AP looks terrible! Much longer sampling *is not* very effective at removing the effects of outlier low probability samples. The threshold filtering is the better solution and is on by default in PEUQSE.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Example: | Type of MCMC: | Samples: | Est. time: | a | B |
|  |  |  |  |  |  |
| 00c2  filter removed | ESS | ~10000 | ~60 seconds | -586 ± 1.2E4 | 149867 ± 3.3E5 |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
| 00c5 filter removed | MH | ~100000 | ~80 seconds | 145 ± 1.3E2 | -44578 ± 2.3E5 |

C2 sampling

C5 sampling

**Convergence Diagnostics**

During MCMC, samples are drawn from the posterior distribution in order to characterize the posterior, and the output gradually achieves a good representation of the true posterior. In principle, a good approximation of the true full posterior should be returned as the number of samples approaches infinity. Convergence diagnostics use statistical tests to indicate when the solution seems to be stable (meaning, that it is not likely to change with further sampling). These are diagnostics *as to whether the samplings of the posterior have converged statistics*. These are *not* diagnostics about *whether the “difference between the true posterior and the sampled posterior” has converged*, for which it is impossible to know for problems like what PEUQSE is designed for, since the sampled posterior has arbitrary shape and arbitrary steepness and is impossible to sample ‘completely’. Still, it is better to use convergence diagnostics rather than to not use them.

It is important to note that MCMC samplers can get “stuck” in local solutions or individual modes, such that the sampling can be converged without reflecting the true posterior. For example, a bad sampling could get stuck in a peak that represents only 1% of the posterior and converge there. Additionally, even when a convergence diagnostic shows indications of convergence, that still does not guarantee even statistical convergence. Thus, convergence diagnostics only tell us if the sampling has not converged. Convergence diagnostics do not tell us whether the solution is representative of the truth. Still, convergence diagnostics are useful. The best practice is to (a) do what is possible within the user’s computational time, using the most appropriate sampler and/or several sampler’s, (b) when possible, run until the solution seems converged.

PEUQSE includes more than one convergence diagnostic. In particular, integrated Autocorrelation time (ACT) and Geweke diagnostics.

(1) The ACT infers convergence when the ACT value steadies to a single value. For ACT plotting, heuristics line is added to give better judgement of when the chains might converge. More information can be found at <https://emcee.readthedocs.io/en/stable/tutorials/autocorr/> .

(2) Geweke diagnostics evaluate convergence on ergodicity, or the random movement of walkers. When a walker starts exploring a smaller region, then it can resemble convergence. Within PEUQSE, the Geweke plots infer chain convergence when the Geweke points fall under 1. These are z-scores that must be within 1 std when comparing the last 50% of the points to the first 10%. A Geweke percents plot shows how big the window size needs to be before the Geweke plot shows convergence. These values should fall and stay at 0%.

(3) Comparison of the MAP and the mu\_AP is also an indication of convergence for cases where the posterior is sufficiently symmetric.

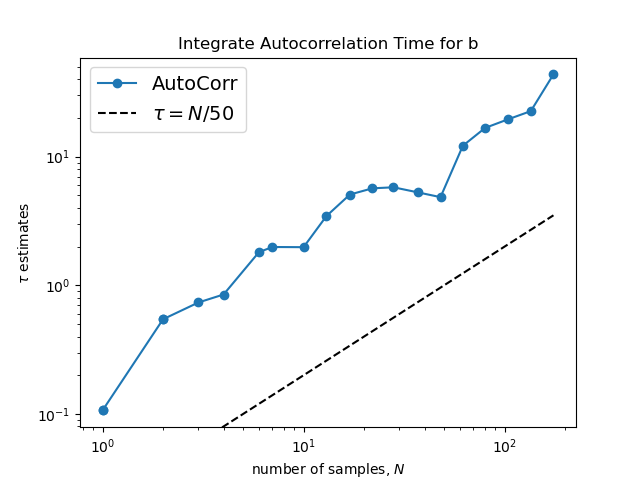
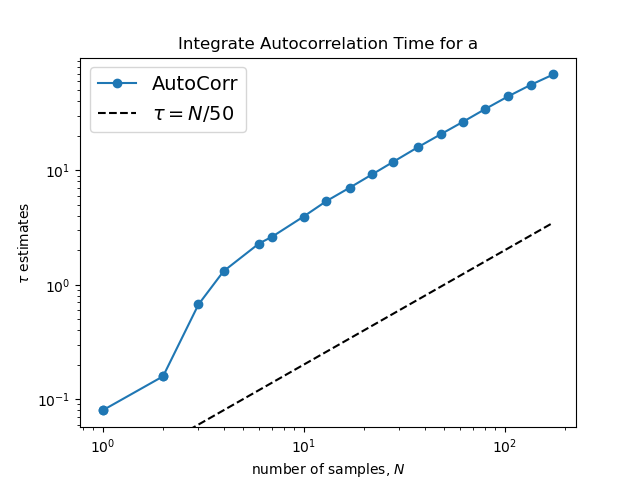
(4) if multiple samplers find similar solutions, that is also a good indication of convergence.

Remember, convergence diagnostics cannot prove convergence. Best practice is to remain skeptical and validate samplings by multiple methods. Sometimes it is obvious convergence has occurred, like in example C8. Sometimes one convergence diagnostic will infer convergence, and another will show the sampling may be starting to converge. It is always better to run more samples, but it is ultimately up to the user’s discretion.

Convergence diagnostics were run on examples C1, C2, C6, and C8, and are shown below.

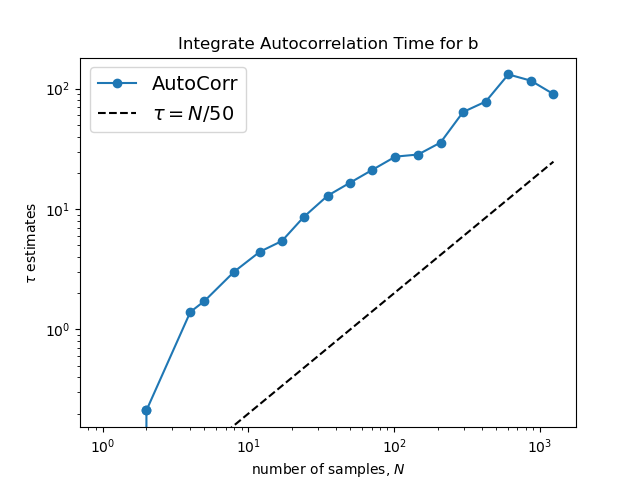
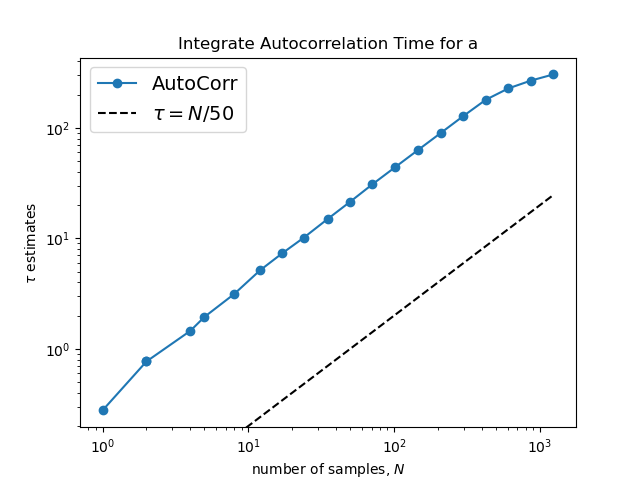
Example C1 – ESS with 1500 samples

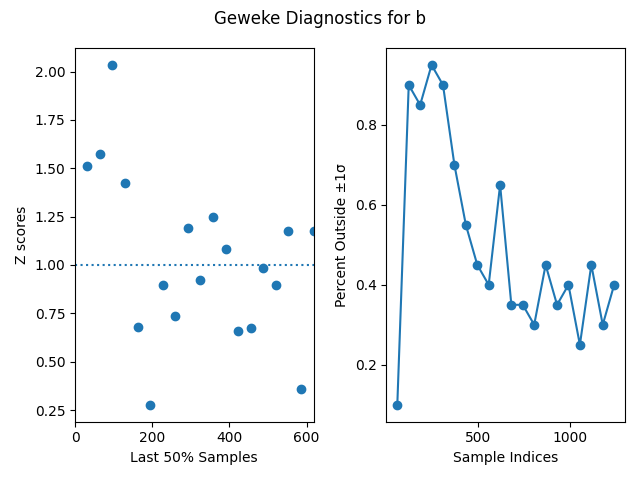
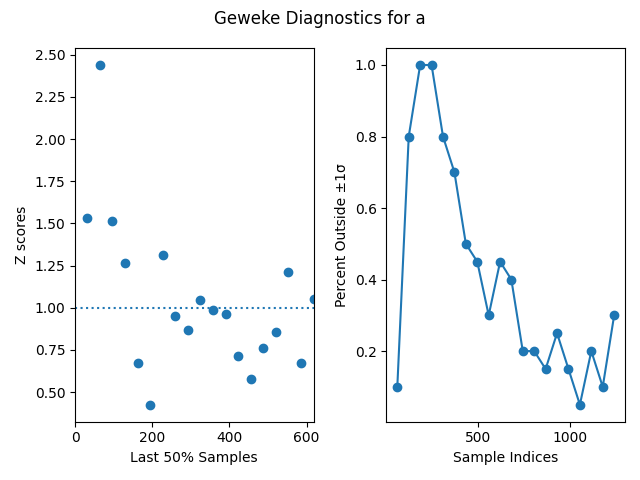
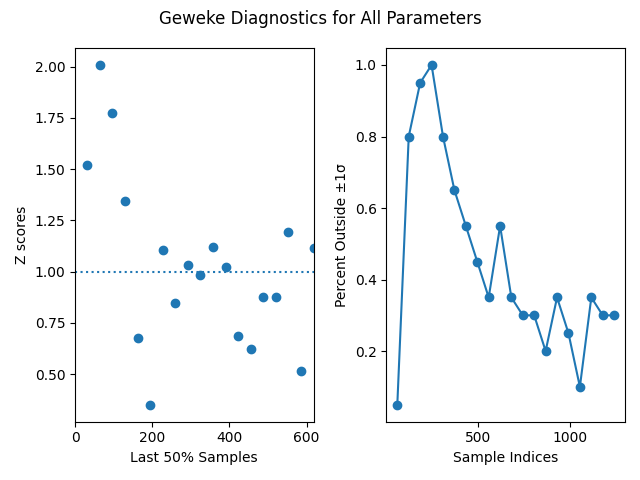
* Convergence cannot be inferred as the ACT values do not converge. The way to assess convergence is when plots flatten to a plateau, which does not occur here.
* The chains were too short for the Geweke plots to be generated.



Example C2 – ESS with 10,000 samples.

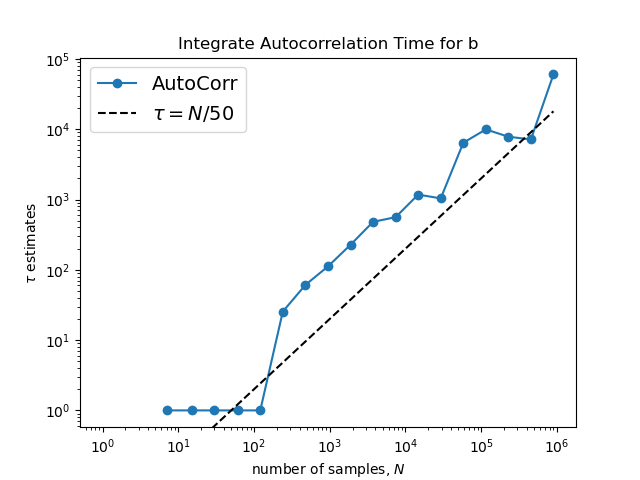
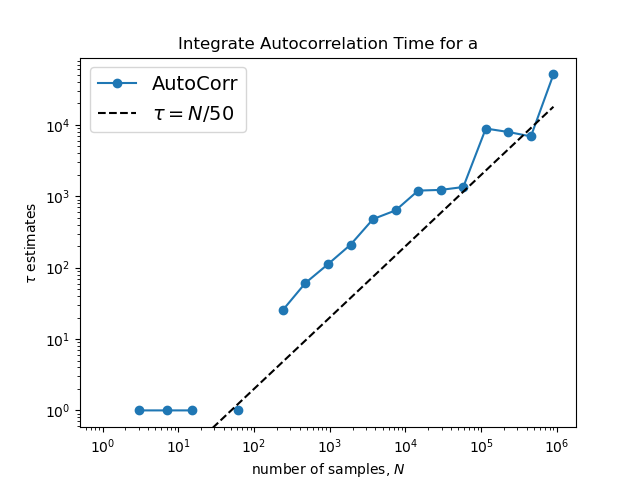
* For both parameters, the ACT diagnostics shows the sampling may be starting to converge, but more samples are needed. The way to assess convergence is when plots flatten.

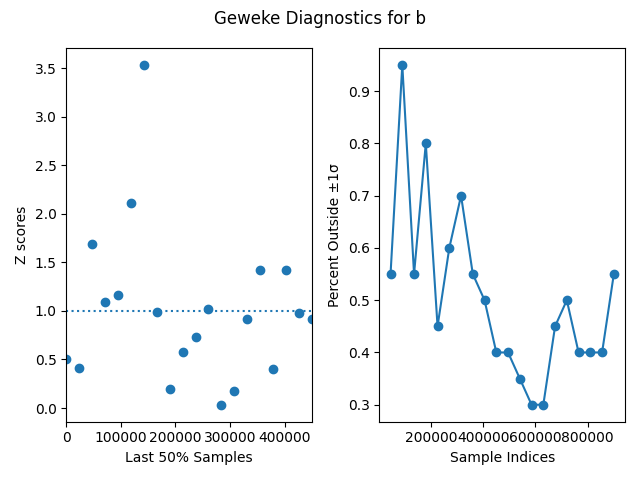
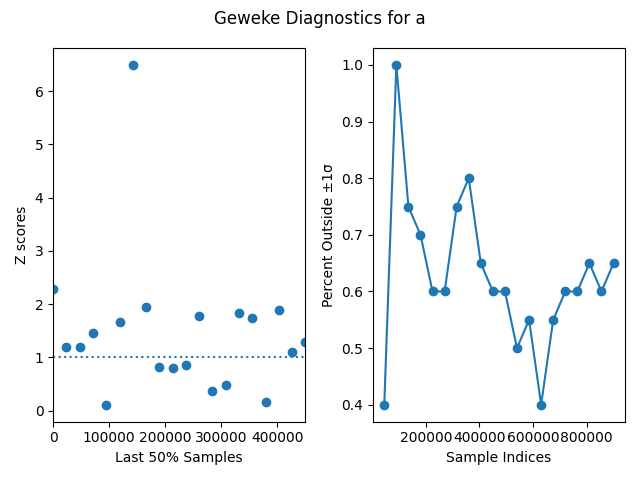


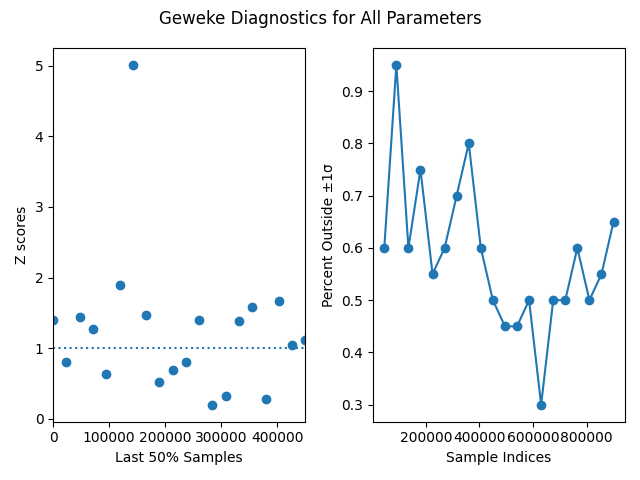
* In the Geweke diagnostic, we see the beginnings of convergence for each parameter and “all parameters”. The way to assess convergence is when the right-hand plot for the parameter goes to zero.
* 
  + - * 

Example C6 – MH with 1,000,000 samples

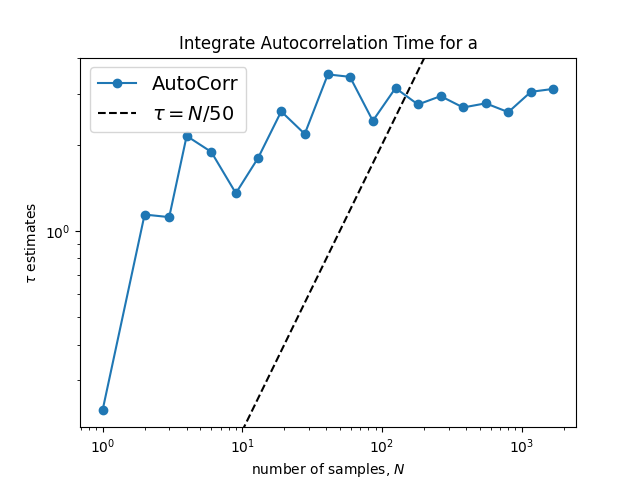
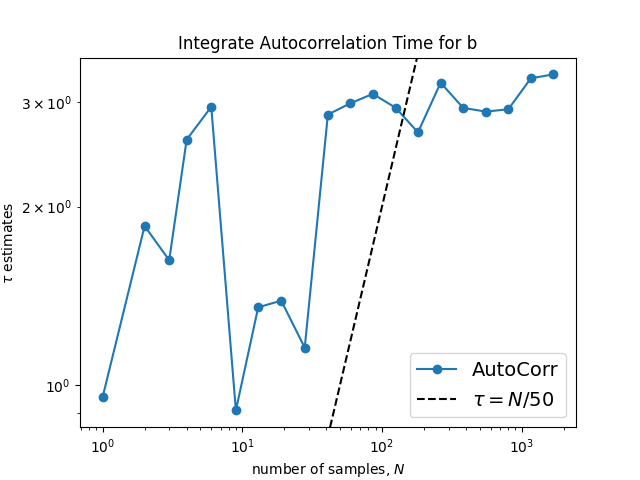
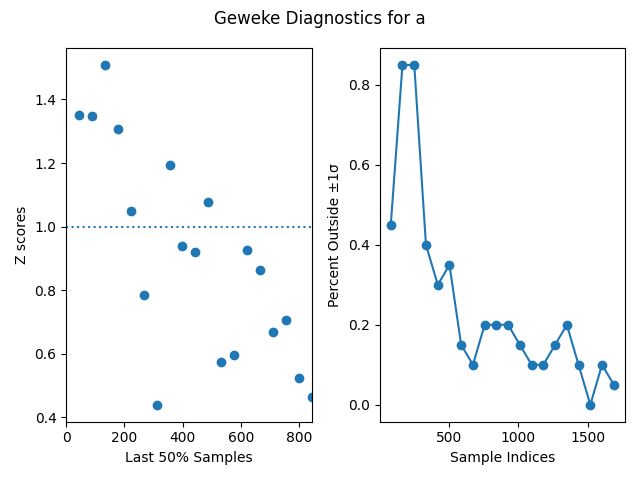
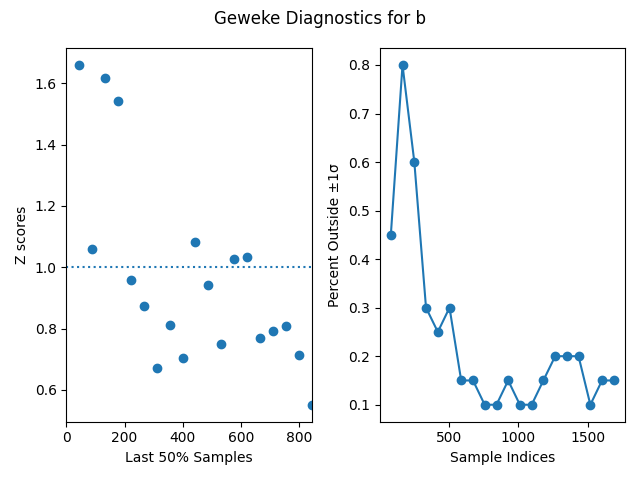
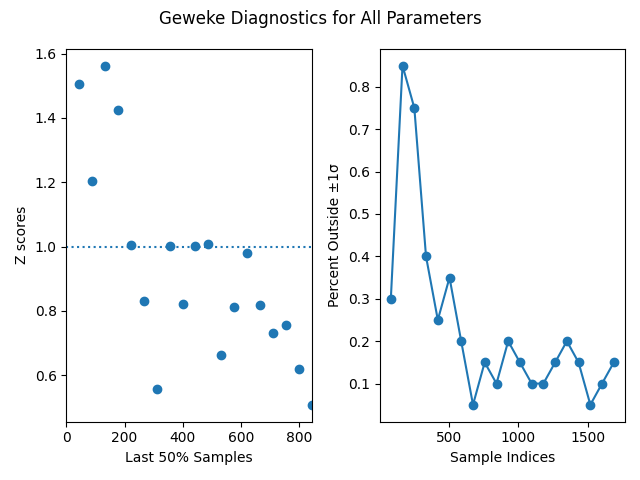
* In this example, the ACT diagnostics do not show convergence yet for either parameter (however, we know this example is actually partially converged).



* In this example, with the Geweke diagnostics, this example seems as if it is not close to convergence. Just on the way to convergence.
* 



Example C7 – ESS with upper and lower bounds and 15000 samples

* This example was made for other reasons, but shows convergence in both ACT and Geweke diagnostics.
*  
*  
* 

Example C8 – uniform sampling with 1,000,000 samples

* Evidence of partial convergence is seen in both the ACT diagnostic and Geweke diagnostic. This example had upper and lower bounds in the region already found by mcmc, which helpedt converge quickly despite being a uniform random sampling run.

Chart, line chart

Description automatically generatedChart, line chart

Description automatically generated