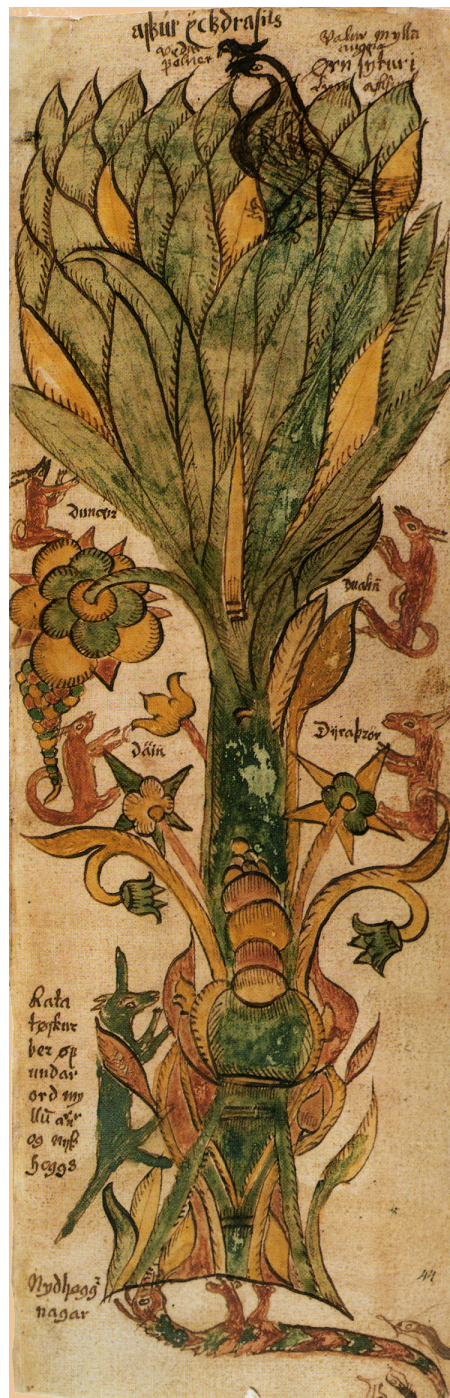


# YGGDRASIL Exercise

Philip Müller, 13-928-304

July 13, 2019



**Front Image** Shows YGGDRASIL the source is [https://de.wikipedia.org/wiki/Yggdrasil#/media/File:AM\\_738\\_4to\\_Yggdrasil1.png](https://de.wikipedia.org/wiki/Yggdrasil#/media/File:AM_738_4to_Yggdrasil1.png).

# 1 General Remarks

It is expected that the user that has devoted a reasonable amount of time is able to solve the exercises presented here. At least the one about Python not the one about C++.

The correction and assessment of your solution has multiple aspects.

- It must be correct and perform the requested functionality. This is the most important aspect.
- The artistic and ascetic aspects of your solution. Is the solution written in an nice, clean and beautiful manner?
- The degree of comments on your code. A good commented code maintains a one to one ratio. Good comments also explain why something is trivial, and explain how hard stuff is done.
- Exploitation of functionality. How good are the resources of PYYGDRASIL used? This aspect mostly measures how good you have studied the manual.
- The second most important aspect of the assessment is *the art of the hack*. How beautiful is your hack. Note that ugly and gross hack are punished. The *bending* of the rules however is encouraged.

## 2 Python

This section asks some questions about Python. You can solve them by a script or with a notebook, however a notebook is recommended.

The exercises must be done in a folder that is different from the build folder. They will be tested under `/test` which contains a symlink to the module.

Note doing the C++ *exceptionally* well will result in full points on the Python part, regardless your performance in it.

### 2.1 Basics

1. Compile the library.
2. Explain the differences between:
  - Root scaled data space.
  - Rescaled root data space.
  - Node data space.
  - Rescaled data space.
3. What is the efficient container to fit a tree. Justify your answer.

### 2.2 Sample Creation

Create a sample with at least 5 dimension with 3 different methods. All samples must be distinct.

### 2.3 Cube Creation

1. Create a hyper cube of dimension 7.
2. Create a hyper cube that is equivalent to

$$[-4.0, 5[\times[1991.5, 77852[\times[878, 962[\times[2.718, 3.1415[\times[687, 1000[\times[-9000, 9000[, \quad (1)$$

*without* the use of an explicit constructor.

3. Create the same cube from above again, but this time all constructors, beside the copy constructor, are allowed.

## 2.4 Random Samples

1. Create a multidimensional distribution of your choice with 4 dimensions.
2. Verify that the first and the centralized second moment are consistent with your choice. Try to be as efficient as possible.
3. Create a one dimensional distribution of your choice, and generate a few sample. Again test if the distribution is consistent also verify that the pdf function of the distribution is consistent with our choice.
4. Redo the last exercise but with a two modal distribution. The mean of the different distributions must be different from each other.

## 2.5 Containers

For all exercises in this section generate first a random distribution of your choice.

1. Create a container of your choice, and fill it with samples from your distributions. You are not allowed to use sampling methods that returns containers only the one that return samples.  
To pass this exercise you must be as efficient as possible, so choose the *right* functions.

## 2.6 Tree

### 2.6.1 Warm up

1. Create your favourite one dimensional distribution. Use it to fit a tree.
2. Compute MISE on the tree. At the same time integrate the theoretical pdf and the tree PDF over the domain. Do not use important sampling neither for the pdf nor the MISE. Try to be as efficient as possible.  
Justify that your results are accurate, no mathematics is needed just argue.
3. Visualize the domains, also how the domain is partitioned. In the same plot show the theoretical pdf and the pdf that is estimated from the tree. Justify divisions of the two.

### 2.6.2 First Part

1. Generate a two dimensional distributions, with four modes. The only restriction is that the mean of the distribution must be  $(2, 5)^T$  and the domain must be centred on that point, and have an extension in each dimension of  $\pm 10$  units. All mode must have different parameters from each other and no mode is allowed to have the mean given above.  
Justify your choice and show your steps.
2. Check that the distribution works, by checking the first and second (centralized) moment. Also check if the range capture most of the distribution. Try to be as efficient as possible.
3. Visualize the tree on a plot, similar as done for the one dimensional case. In the same plot visualize the density that is given by the tree.  
Repeat this but use the theoretical density instead.

### 2.6.3 Second Part

Create a distribution with at least four dimension of your choice. Perform a scaling experiment on it with the following corner points.

- Compare at least the linear and the constant model with each other, in the same plot.
- Use the following sizes for the samples that are used to fit the tree:

$$N_i = 10^{2+\frac{1}{2} \cdot i}, \quad i \in [0, 15] \cap \mathbb{N}$$

- For computing the MISE generate at least 6 times more samples than you have used for fitting.
- Repeat each size at least 4 times to get the mean.

Note that this exercise is more about how you use your tools than anything else.

## 2.7 Tree Sampler

In this section the tree Sampler will be tested.

1. Generate a distribution of your choice and generate a sampler from it. Draw samples from that tree distribution and verify that the sampler generates the underlying distribution.
2. From the same tree generate a new tree sampler object, but this time apply a condition to it. Verify that the sampler still generates samples that are distributed correctly. For that you have to compute the covariance, you can use Mathematica for that.
3. Perform a scaling like experiment. Fit a tree with a certain, in the beginning small amount of samples. Use the tree to get a tree sampler object and use it to judge the accuracy of the quality of the sampler, as it was shown in the notebooks. Increase the amount of samples and show the results.
4. Generate a tree dimensional Dirichlet distribution, this distribution will have 4 parameters, of your choice. Generate samples from it and fit a tree. From the tree construct a tree sampler that restricts one dimension. The tree sampler will have “dimension 2” visualize the pdf of the sampler.

## 3 C++

This is the test on C++ it is not expected that after studying the manual a user, beside some software developer, could solve them. Also whereas the Python part needs a few hour to do, this part will take several days, at least.

Also note that every new feature you add must also find its way to PYYGGDRASIL.

1. Extend PYYGGDRASIL by adding a wrapper to a member function of your choice.
2. Implement a new one dimensional random number distribution. It is recommended to do the exponential distribution.
3. Implement the multi dimensional t distribution.
4. Implement references into the sample array and the sample collection.
5. Extend the sample list such that it is impossible to insert a sample of the wrong dimension.
6. Implement the coping of the tree.  
This is considered rather easy.
7. Extend the container such that they have a “natural array” constructor. This is a constructor which takes an array that is tailored for their internal format.
8. Implement the dumping of the tree to hdf5.
9. Implement a new model and perform the integration in the code base. Since this part only goes about how well you can integrate a new model, you are allowed to copy one of the existing ones and rename it.
10. Implement a new test, the same as for the model applies.
11. Implemented a task based parallelism into the tree.
12. Implement a new sample container. That behaves as the array container, but is only a reference to an other array.
13. Implement the inserting scheme. Note this is considered hard.
14. Implemented “adaptive rescaling”. This means only apply rescaling if a cell falls below a certain size.
15. Implement the parallel generation of random samples from the tree and the distributions. Note that a simple openMP `pragma` is not enough, the reason is that the generator is not thread safe. Implement a generator that bypass that problem.