# Uniform Sampling (Using Hammersley Sequence)

Close agreement. Whilst there are no exact numbers for higher number of points in Wu,2023, from graph (next page) results look similar for 5k and 10k points as well.

|  |
| --- |
| **C:\Users\mn17jilf\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Ham_all3 Points.png** |
| **Point Distribution using Hammersley Sequence.** |

|  |
| --- |
| **C:\Users\mn17jilf\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Ham_2k Points.png** |
| **Close up of 2000 point case** |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Number | Time Average (minutes) | Time Standard Deviation | Error | Error Standard Deviation | % Difference | % Error |
| 2 000 |  |  | 3.0200% | ± 2.98% | - | - |
| 2 000 | 15.22 | 1.54E+00 | 3.1219% | 1.93E-02 | 3.3% | 3.4% |
| 5 000 | 37.66 | 4.51E+00 | 0.0845% | 4.66E-04 | - | - |
| 10 000 | 63.89 | 7.01E+00 | 0.0150% | 7.01E-05 | - | - |

|  |
| --- |
|  |
| **Error vs Points used (Hammersley Uniform) –** Hard to see orange cross as right on top. |

|  |
| --- |
|  |
| **Error vs Points –** From Wu, 2023. For different uniform sampling sequences. |

|  |
| --- |
|  |
| **Time vs Points used (Hammersley Uniform Method)** |

|  |
| --- |
| C:\Users\mn17jilf\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Uniform cases loss.png |
| **Loss graph –** I need to see how the stopping is decided to see if this can be run for longer to improve accuracy. However, I need to check that this is not an intentional feature to prevent overfitting. |

# RAD – Residual-based Adaptive Distribution

This method replaces all the 2000 points every N iterations. The default algorithm re-samples the points 100 times, allowing the algorithm to keep running further reducing error, but taking a long time.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Time Average (min) | Time Standard Deviation | Error | Error Standard Deviation | % Difference | % Error |
| Wu, 2023 (100) |  |  | 0.02% | ± 0.00% | - | - |
| 10 resamples | 27.37 | 1.01E+00 | 9.347% | 2.22E-01 | 199.1% | 46634.2% |
| 100 resamples | 709.58 | 3.87E+02 | 0.078% | 6.88E-04 | 118.5% | 290.6% |

|  |
| --- |
| C:\Users\mn17jilf\AppData\Local\Microsoft\Windows\INetCache\Content.Word\RAD_Default points.png |
| **RAD point distribution at end.** |

|  |
| --- |
|  |
| **Isometric view of sample solution obtained by RAD algorithm** |

# RAR-D - Residual-Based Adaptive Refinement with Distribution.

In this case, starts with a random distribution and in theory adds new points based on residual information every re-sampling period. In the default algorithm, this was done 100 times. This is done by starting with a sample of 1000 points, and adding (1000/100 = 10) points every 2000 steps.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Time Average (min) | Time Standard Deviation | Error | Error Standard Deviation | % Difference | % Error |
| Validation (100) |  |  | 0.03% | ± 0.01% | - | - |
| 10 resamples | 72.48 | 6.24E+01 | 17.558% | 7.09E-02 | 199.3% | 58426.0% |
| 100 resamples | 445.02 | 2.13E+02 | 20.228% | 1.01E-01 | 199.4% | 67326.7% |

|  |
| --- |
| C:\Users\mn17jilf\AppData\Local\Microsoft\Windows\INetCache\Content.Word\RAR-D_Default points.png |
| **RAR-D point distribution at end** – This random distribution highlights hyper-parameters must’ve been set wrong as algorithm is purely randomly sampling. |

# Comparing all methods

|  |
| --- |
|  |
| **Error of Different Methods** – Blue dots in Uniform represent 2k, 5k and 10k points. In RAD, 10 re-samples and 100 re-samples. |

|  |
| --- |
|  |
| **Time of different methods –** Shortest RAD cases took ~ 4 hours, but subsequent runs would slow down drastically. Using a reasonable number of re-samples kept the time lower than using 5k or 10k points with a Uniform sampling method, but impacted the accuracy. |

|  |
| --- |
|  |
| **Test loss of different methods –** Helps visualise just how much quicker the Uniform method is done. However, for better visualising this I should use a smoothing function (like a moving average). It seems like RAD loss overall keeps dropping towards end, not so for RAR-D as it’s not properly working. |