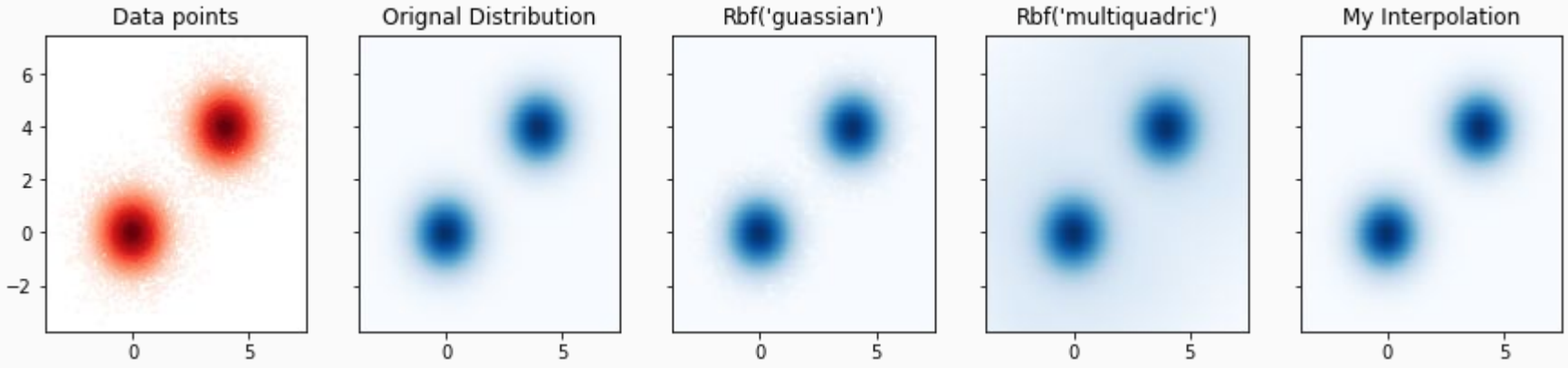
Progress Report

Kuei-Yueh (Clint) Ko

Testing sample




```
N_POINTS = 10000
DIM_GRIDS = 128
mean = ([0.0, 0.0], [4.0, 4.0])
prop = (0.5, 0.5)
```

Benchmark of Scipy Rbf function

Scipy Rbf('gaussian')

```
%%timeit
rbfi = Rbf(dat_pts[:, 0], dat_pts[:, 1], value, function='gaussian')
z_rbf_gauss = rbfi(x_c, y_c)
```

```
/local_data/env-py3/lib/python3.6/site-packages/scipy/linalg/basic.py:40: RuntimeWarning: scipy.linalg.solve
Ill-conditioned matrix detected. Result is not guaranteed to be accurate.
Reciprocal condition number/precision: 4.850221788728607e-20 / 1.1102230246251565e-16
  RuntimeWarning)
```

```
15.7 s ± 26.7 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)
```

Scipy Rbf('multiquadric')

```
%%timeit
rbfi = Rbf(dat_pts[:, 0], dat_pts[:, 1], value, function='multiquadric')
z_rbf_multiq = rbfi(x_c, y_c)
```

```
/local_data/env-py3/lib/python3.6/site-packages/scipy/linalg/basic.py:40: RuntimeWarning: scipy.linalg.solve
Ill-conditioned matrix detected. Result is not guaranteed to be accurate.
Reciprocal condition number/precision: 9.993949597828553e-17 / 1.1102230246251565e-16
  RuntimeWarning)
```

```
11.3 s ± 35.8 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)
```

Benchmark of my interpolation function

GPU (CPU input/output arrays); (threadsperblock = 32)

```
%%timeit
get_weights[blockspergrid, threadsperblock](grids, points, weights)
```

```
370 ms ± 17.9 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)
```

```
%%timeit
get_weights[blockspergrid, threadsperblock](grids, points, weights)
z_test = np.matmul(weights, value.reshape(N_POINTS, -1))
z_test = z_test.reshape(DIM_GRIDS, DIM_GRIDS)
```

```
844 ms ± 4.26 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)
```

np.matmul ~ 844 - 370 = 474 (ms)

- **Threads per block = (32, 32)**

Benchmark of my interpolation function

GPU (GPU input/output arrays); (threadsperblock = 32)

```
%%timeit
get_weights[blockspergrid, threadsperblock](grids_device, points_device, weights_device)
weights = weights_device.copy_to_host()
```

```
452 ms ± 7.32 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)
```

```
%%timeit
get_weights[blockspergrid, threadsperblock](grids_device, points_device, weights_device)
weights = weights_device.copy_to_host()
z_test = np.matmul(weights, value.reshape(N_POINTS, -1))
z_test = z_test.reshape(DIM_GRIDS, DIM_GRIDS)
```

```
932 ms ± 489 µs per loop (mean ± std. dev. of 7 runs, 1 loop each)
```

np.matmul ~ 932 - 452 = 480 (ms)

In this example, we do not gain a lot when the process of data copy from CPU to GPU is removed. I guess the reason is because this data copy process does not consume much time when comparing to calculating the weights

- **Threads per block = (32, 32)**

Note 01

test skcuda and pycuda

```
import pycuda.autoinit
import pycuda.gpuarray as gpuarray
import numpy as np
import skcuda.linalg as linalg
import skcuda.misc as misc
linalg.init()
a = np.asarray(np.random.rand(4, 2), np.float32)
b = np.asarray(np.random.rand(2, 2), np.float32)
a_gpu = gpuarray.to_gpu(a)
b_gpu = gpuarray.to_gpu(b)
c_gpu = linalg.dot(a_gpu, b_gpu)
np.allclose(np.dot(a, b), c_gpu.get())
```

```
---------------------------------------------------------------------------
OSError                                   Traceback (most recent call last)
<ipython-input-263-bcef7324e87e> in <module>()
      4 import skcuda.linalg as linalg
      5 import skcuda.misc as misc
----> 6 linalg.init()
      7 a = np.asarray(np.random.rand(4, 2), np.float32)
      8 b = np.asarray(np.random.rand(2, 2), np.float32)

/local_data/env-py3/lib/python3.6/site-packages/skcuda/misc.py in init(allocator)
    175     global _global_cusolver_handle
    176     if not _global_cusolver_handle:
--> 177         from . import cusolver
    178         _global_cusolver_handle = cusolver.cusolverDnCreate()
    179

/local_data/env-py3/lib/python3.6/site-packages/skcuda/cusolver.py in <module>()
     49         break
     50 if _libcusolver == None:
---> 51     raise OSError('cusolver library not found')
     52
     53 class CUSOLVER_ERROR(Exception):

OSError: cusolver library not found
```

Note 02

Threads per block = (64, 64)

```
/usr/lib/python3/dist-packages/numba/cuda/cudadrv/driver.py in safe_cuda_api_call(*args)
    286             _logger.debug('call driver api: %s', libfn.__name__)
    287             retcode = libfn(*args)
--> 288             self._check_error(fname, retcode)
    289         return safe_cuda_api_call
    290

/usr/lib/python3/dist-packages/numba/cuda/cudadrv/driver.py in _check_error(self, fname, retcode)
    321                     _logger.critical(msg, _getpid(), self.pid)
    322                     raise CudaDriverError("CUDA initialized before forking")
--> 323             raise CudaAPIError(retcode, msg)
    324
    325     def get_device(self, devnum=0):

CudaAPIError: [1] Call to cuLaunchKernel results in CUDA_ERROR_INVALID_VALUE
```