1 Contributions

- **C1.** We introduce the concept of "approximation", a reduced representation generated from an initial compression using an existing error-controlled lossy compressor. This decouples compression with bitplane encoding, which enables simpler recomposition and better isolation of data points. We further develop an optimized preprocessing technique for a target unstructured dataset, which is often randomly ordered and challenging to compress, and integrate it into our refactoring pipeline for better data approximation.
  Supporting artifacts: **A3**
- **C2.** We introduce a novel paradigm that encodes the data based on their impacts on the target QoIs. We further explore strategies for assigning weights to data based on QoI functions and integrating the weights into bitplane encoding, the key stage that enables progressive representations.
  Supporting artifacts: **A4**
- **C3.** We enhance the error estimation by considering the distinct impact of different variables in the case of multivariate QoIs using a coordinate error bound descent algorithm.
  Supporting artifacts: **A5**
- **C4.** We rigorously evaluate `QPro` against prior works using a case study on a computational fluid dynamics (CFD) code from Generic Electric (GE) and four other real-world datasets. Experimental results demonstrate that `QPro` provides guaranteed error control under given QoI tolerances with significantly reduced data size.
  Supporting artifacts: **A3-A8**

2 Artifacts

- **A1. Project source code**:
  The Github repository qoi-control contains the full implementation of PSZ3-delta, and PMGARD-HB. The Github repository ProDM contains the full implementation of our proposed efficient progressive retrieval framework with QoI error control, including QPro-N, QPro-NW, QPro, which are detalied in A**.

  Note that QPro-NW is already integrated into QPro together with coordinate error bound descent algorithm. If you'd like to test QPro-NW only, please undo the comment of the part below "// estimate error bound based on maximal errors" (uniform error bound descent algorithm) and then comment the part below "// **********..." (coordinate error bound descent algorithm) in the function `halfing_error_QoI_uniform(..., std::vector<std::vector<int> weights)` where `QoI` varies according to the QoI types.
- **A2. Dataset**:
  - In our work, we utilized six double precision 3D datasets, namely: GE-small (unstructure), Hurricane ($100 \times 500 \times 500$), NYX ($512 \times 512 \times 512$), SCALE($98 \times 1200 \times 1200$), S3D($500 \times 500\times$), and GE-large(unstructure). In our evaluation, we converted the original single precision data of Hurricane, NYX, and SCALE into double precision.
  - Variables of each dataset:
    - GE: V_x, V_y, V_z, $Pressure$, $Density$
    - Hurricane: V_x, V_y, V_z
    - NYX: V_x, V_y, V_z
    - SCALE: V_x, V_y, V_z
    - S3D: V_x, V_y, V_z
  - Hurricane, NYX, SCALE, and S3D are available online. Note that, S3D dataset is composed by multiple snapshots, each with 11 fields while the last 3 fields are V_x, V_y, and V_z. We used the last 3 fields of the last snapshot of S3D dataset as our test example. Access to GE-small, and GE-large datasets is upon request.
- **A3. Data approximation algorithm (QPro-N)**
  We propose this method to allows for easy integration of filtering mechanisms and modifications to bitplane encoding, as the use of error-controlled lossy compressors is decoupled from the encoding procedure. Detailed description can be found in **Section 5**. This method is applied in all of our evaluations and we show the computation results in **Fig. 4 - Fig. 8**.

- **A4. Weighted bitplane encoding (QPro-NW)**
  We propose this method to encode the data based on their impacts on the target QoIs to mitigate the issue that maximal QoI error is dominated by a few sensitive data points. Detailed explanation can be found in **Section 6**. This method is applied in all of our evaluations and we show the computational results in **Fig. 4 - Fig. 8**.

- **A5. Coordinate error bound descent algorithm (QPro)**
  We propose this method to find the new data error bound after knowing that the supremum of QoI errors with the current data representation exceeds the tolerance, and it is only applied to a single data point with the maximal estimation of QoI error for performance consideration. Detailed illustration can be found in **Section 7**. This method is applied in all of our evaluations and we show the computational result in **Fig. 5 - Fig. 8**.

- **A6. Progressive compression with bitplane using PMGARD as underlying compressor (PMGARD-HB)**
  The PMGARD-HB method which omits L^2 projection in PMGARD's decomposition algorithm for more accurate error estimation. The elaboration of this method can be found in **Section V.B of Citation 45**. And the related computational results are shown using the cyan and purple lines in **Fig. 6 - Fig. 7**, and also blue bars in **Fig. 8**.

- **A7. Error-controlled delta compression using SZ3 as underlying compressor (PSZ3-delta)**
  The PSZ3-delta method which reduces data into multiple snapshots but eliminates the redundancy by compressing the residues. The elaboration of this method can also be found in **Section V.B of Citation 45**. And the related computational results are shown using the brown lines in **Fig. 6 - Fig. 7**, and also green bars in **Fig. 8**.

- **A8. Progressive compression using QPro**
  The QPro method integrates A3, A4, and A5 together for more accurate error estimation and more efficient retrieval. The elaboration of this method can be found in **Section 5 - Section 7**. And the related computational results are shown using the red lines in **Fig. 5 - Fig. 7**, and also yellow bars in **Fig. 8**.

3 Artifact Evaluation

- **Installation**

```
git clone https://github.com/Linus-Li-1037/ProDM.git
cd ProDM
sh build_script.sh
cd ..
git clone https://github.com/Linus-Li-1037/qoi-control.git
cd qoi-control
sh build_script.sh
cd ..
```

- **Refactoring the Data** This is the initial step before running any tests to evaluate QoI error control.
  - To generate the refacotred data for one of the five datasets among GE-small, Hurricane, NYX, SCALE, and S3D, the usage of command is as follows:

    ./qoi-control/build/test/refactor_data [1 - PMGARD-HB, 3 - PSZ3-delta] [Name of dataset] [Path to dataset]

    ./ProDM/build/test/refactor_d64 [1 - SZ3-based approximation for structured datasets, 3 - Customized approximation for GE datasets] [0 - Bitplane encoding **(QPro-N)**, 1 - Weighted bitplane encoding **QPro**] [Name of dataset] [Path to dataset] [max_weight_V] (**For structured datasets** [block_size], **for GE datasets** [max_weight_for_T] [max_weight_for_T])

  - The above commands only includes **QPro** and **QPro-N**, if you'd like to test with **QPro-N**, just modify the second parameter from 1 to 0. If you'd like to test with **QPro-NW**, you can use the same parameter like **QPro**, however, please refer to the usage of QPro-NW in **A1** to use uniform error bound descent algorithm. Note that, the commands of ProDM on same dataset would cover the files of previous results, so please finish the evaluation before you move to next one.

- **Results for Fig. 4**
  - QPro-N is our baseline. **Error bounds(eb)** includes 1e-1, 1e-2, ..., 1e-6, and 5e-2, 5e-3, ..., 5e-7. Here, [PathR] represents the path to the reordered GE dataset.

```
./ProDM/build/test/refactor_d64 3 0 GE_small [PathR]
./ProDM/build/test/qoi_Vtot_d64 3 0 <eb> [PathR]
./ProDM/build/test/qoi_T_d64 3 0 <eb> [PathR]
./ProDM/build/test/refactor_d64 3 0 Hurricane [PathR]
./ProDM/build/test/qoi_Vtot_d64 3 0 <eb> [PathR]
./ProDM/build/test/refactor_d64 3 0 NYX [PathR]
./ProDM/build/test/qoi_Vtot_d64 3 0 <eb> [PathR]
```

  - QPro-NW is tested with both different **max_weight_V** and **max_weight_T** which both include 3, 4, 5, 7, 9, and different **block_size** which includes 3, 4, 5, 6. To test QPro-NW instead of QPro, please refer to the usage of QPro-NW in **A1**. Here, [Path] refers to the path to the corresponding dataset while [PathR] refers to the path to the reordered GE dataset.

```
./ProDM/build/test/refactor_d64 3 1 GE_small [PathR] <max_weight_V> \
<max_weight_T> <max_weight_T>
./ProDM/build/test/qoi_Vtot_d64 3 1 <eb> [PathR]
./ProDM/build/test/qoi_T_d64 3 1 <eb> [PathR]
./ProDM/build/test/refactor_d64 1 1 Hurricane [Path] <max_weight_V> <block_size>
./ProDM/build/test/qoi_Vtot_d64 1 1 <eb> [Path]
./ProDM/build/test/refactor_d64 1 1 NYX [Path] <max_weight_V> <block_size>
./ProDM/build/test/qoi_Vtot_d64 1 1 <eb> [Path]
```

- **Results for Fig. 5**
  - The whole part is tested with QPro with max_weight_V = 4 while max_weight_T = 3. Here, [PathR] represents the path to the reordered GE dataset.

```
./ProDM/build/test/refactor_d64 3 1 GE_small [PathR] 4 3 3
./ProDM/build/test/qoi_Vtot_d64 3 1 <eb> [PathR]
```

- **Results for Fig. 6**
  - Here, [Path] refers to the path to GE dataset while [PathR] refers to the path to the reordered GE dataset. And, [QoI] includes six real world QoIs which are V_{total}, T, C, $Mach$, PT and mu. Note that the second to last command is for QPro-NW. To test QPro-NW instead of QPro, please refer to the usage of QPro-NW in **A1**.

```
./qoi-control/build/test/refactor_data 1 GE [Path]
./qoi-control/build/test/refactor_data 3 GE [Path]
./qoi-control/build/test/halfing_Vtot <eb> [Path]
./qoi-control/build/test/halfing_Vtot_sz3delta <eb> [Path]
./qoi-control/build/test/refactor_data 1 GE [PathR]
./qoi-control/build/test/halfing_Vtot <eb> [PathR]
./ProDM/build/test/refactor_d64 3 0 GE_small [PathR]
./ProDM/build/test/qoi_[QoI]_d64 3 0 <eb> [PathR]
./ProDM/build/test/refactor_d64 3 1 GE_small [PathR] 4 3 3
```

```
./ProDM/build/test/qoi_Vtot_d64 3 1 <eb> [PathR]
./ProDM/build/test/qoi_Vtot_d64 3 1 <eb> [PathR]
```

- **Results for Fig. 7**
  - Here, [Dataset] denotes four structured datasets Hurricane, NYX, SCALE, and S3D, while [Path] denotes the path to the corresponding dataset. Note that the second to last command is for QPro-NW. To test QPro-NW instead of QPro, please refer to the usage of QPro-NW in **A1**.

```
./qoi-control/build/test/refactor_data 1 [Dataset] [Path]
./qoi-control/build/test/refactor_data 3 [Dataset] [Path]
./qoi-control/build/test/halfing_Vtot <eb> [Path]
./qoi-control/build/test/halfing_Vtot_sz3delta <eb> [Path]
./ProDM/build/test/refactor_d64 1 0 [Dataset] [Path]
./ProDM/build/test/qoi_Vtot_d64 1 0 <eb> [Path]
./ProDM/build/test/refactor_d64 1 1 [Dataset] [Path] 7 4
./ProDM/build/test/qoi_Vtot_d64 1 1 <eb> [Path]
./ProDM/build/test/qoi_Vtot_d64 1 1 <eb> [Path]
```

- **Results for Fig. 8**
  - To generate the result for Fig. 8 under relative error bound 1e-1, first run the following command for data refactoring.

```
mpirun -n 96 ./ProDM/build/prl_src/refactorROD
mpirun -n 96 ./qoi-control/build/parallel_src/refactorGE
mpirun -n 96 ./qoi-control/build/parallel_src/refactorGE_SZ3_delta
```

  - Then run the command below for data retrieval

```
mpirun -n 96 ./ProDM/build/prl_src/test_PT 1e-1
mpirun -n 96 ./qoi-control/build/parallel_src/testGE_PT 1e-1
mpirun -n 96 ./qoi-control/build/parallel_src/testGE_PT_SZ3_delta 1e-1
```