ExaGeoStatCPP

Generated by Doxygen 1.9.1

Chapter 1

ExaGeoStat

The **Exascale GeoStatistics** project (ExaGeoStat) is a parallel high-performance unified framework for computational geostatistics on many-core systems. The project aims to optimize the likelihood function for a given spatial data to efficiently predict missing observations in the context of climate/weather forecasting applications. This machine learning framework proposes a unified simulation code structure to target various hardware architectures, from commodity x86 to GPU accelerator-based shared and distributed-memory systems. ExaGeoStat enables statisticians to tackle computationally challenging scientific problems at large-scale while abstracting the hardware complexity through state-of-the-art high-performance linear algebra software libraries.

ExaGeoStatCPP

ExaGeoStatCPP is a C++ API for ExaGeoStat that aims to offer a user-friendly and efficient API for C++ developers, essentially maintaining traditional practices and embracing contemporary C++ elements like namespaces, templates, and exceptions to enhance functionality.

ExaGeoStatR : R Interface of ExaGeoStat

R is a powerful and versatile tool for scientific computing, offering a wide range of statistical and graphical techniques, strong community support, and the flexibility to integrate with other programming languages. Its open-source nature and extensive package ecosystem make it an invaluable resource for researchers and data scientists. Therefore, we decided to create ExaGeoStatR: An interface for functionalities provided by ExaGeoStatCPP to make use of R's various benefits.

Vision of ExaGeoStat/ExaGeoStatCPP

The ExaGeoStat/ExaGeoStatCPP project is a collaboration between the KAUST Spatial Statistics group and the Extreme Computing Research Center (ECRC). Lies not in a new algorithm nor a new dataset, but in demonstrating the routine use of the larger datasets becoming available to geospatial statisticians, thanks to the implementation of state-of-the-art statistical algorithms on High Performance Computing (HPC) hardware.

We have built a standalone software framework (ExaGeoStat/ExaGeoStatCPP) that can run on a variety of hardware resources, including GPUs and massively distributed systems such as Shaheen-II, KAUST's Cray XC40 supercomputer, HLRS HPE Apollo (Hawk), ORNL Summit (OLCF-4) supercomputer, and Riken Fugaku supercomputer, to create a statistical model to predict environmental data (i.e., temperature, flow rates, soil moisture, wind speed, air pollution, etc.) at spatial locations on which data is missing, and to exploit large amounts of data to reduce the effect of individual measurement errors. The best-known methods for such statistical processing have a cost that grows rapidly in the size of the dataset, namely, in proportion to its cube or third power. Thus, increasing the size of the dataset by a factor of ten drives up the cost of the computation by a factor of a thousand while simultaneously driving up the memory requirements by a factor of a hundred.

For instance, according to this cubic growth in complexity, a computation that requires one minute would require nearly 17 hours on a dataset just ten times larger. This creates a computational strain on standard statistics software, for which contemporary data sizes were not anticipated, and even if possible, it puts the computation beyond the interactive attention span of the analyst. Parallelism (assigning thousands of processors to a single task) and Moore's Law allow leading-edge computers to handle such "big data" with ease, but the software bridge must be built. Furthermore, the software interface must resemble the interactive one with which working statisticians are familiar.

To summarize, the combination of emerging computing capabilities and emerging datasets promises significant advances in statistical analyses of environmental and many other phenomena. Such cross-disciplinary advances are natural at KAUST, so this relatively low-hanging fruit was ours to harvest earliest. Our roadmap now takes Exa↵ GeoStat a step further on the algorithmic side by integrating tile low-rank matrix approximation. This low-rank matrix approximation permits the exploitation of the data sparsity of the operator with user-controlled numerical accuracy. This further expands practical problem sizes for statisticians with modest computational resources.

Installation

> Note: Installation requires at least **CMake of version 3.2**. to build ExaGeoStatCPP.

C++ source code installation

To install the `ExaGeoStat` project locally, run the following commands in your terminal:

1. Clone the project from the remote gitHub repository into your local machine using the following command
   `git clone https://github.com/ecrc/ExaGeoStatCPP.git`
2. Change your current directory by getting into the `ExaGeoStatCPP` project directory
   `cd ExaGeoStatCPP`
3. Run `configure` script with the flag `-h` for help, to know the supported options and their corresponding flags.
   `./configure -h`
4. Run `clean_build.sh` script with the flag `-h` for help, to know the needed arguments to run with your specific options.
   `./clean_build.sh -h`
5. Export the installation paths of the dependencies to your `.bashrc` file, e.g.
   `export PKG_CONFIG_PATH=$PWD/installdir/_deps/DEPENDENCY_NAME/lib/pkgconfig:$PKG_CONFIG_PATH`

Now, you can use the pkg-config executable to collect compiler and linker flags for ExaGeoStatCPP.

R package installation

1. Open the R prompt window by simply running `R` command in the terminal, inside the prompt, we will install needed packages by running the following commands:
   ```
   install.packages(Rcpp)
   install.packages("assert")
   ```
2. close the R prompt and return to the terminal. Run the following command, make sure your current path is the ExaGeoStat project directory

```
R CMD INSTALL . --configure-args="-r"
```

> For more detailed information on installing ExaGeoStat with different configurations and enabling technologies such as CUDA, MPI, R, etc., please refer to the User Manual

Usage

C++ Example

```
{C++}
int main(int argc, char **argv) {
    // Create a new configurations object.
    Configurations configurations;
    // Initialize the arguments with the provided command line arguments
    configurations.InitializeArguments(argc, argv);
    // Initialize the ExaGeoStat Hardware
    auto hardware = ExaGeoStatHardware(configurations.GetComputation(), configurations.GetCoresNumber(),
       configurations.GetGPUsNumbers());
    // Load data by either read from file or create synthetic data.
    std::unique_ptr<ExaGeoStatData<double» data;
    ExaGeoStat<double>::ExaGeoStatLoadData(configurations, data);
    // Modeling module.
    ExaGeoStat<double>::ExaGeoStatDataModeling(configurations, data);
    // Prediction module
    ExaGeoStat<double>::ExaGeoStatPrediction(configurations, data);
    return 0;
}
```

R Example:

```
hardware <- new(Hardware, computation, ncores, ngpus, p, q)
exageostat_data <- simulate_data(kernel=kernel, initial_theta=initial_theta, problem_size=problem_size,
       dts=dts, dimension=dimension)
estimated_theta <- model_data(data=exageostat_data, kernel=kernel, dts=dts,
       dimension=dimension,lb=lower_bound, ub=upper_bound, mle_itr=10)
predict_data(train_data=list(x, y, z_measurement), test_data=list(test_x, test_y), kernel=kernel, dts=dts,
       estimated_theta=estimated_theta)
```

Please take a look at the end-to-end examples as a reference for using all the operations.

Contributing

Find detailed information on how to contribute to ExaGeoStatCPP here

References

1. Sameh Abdulah, Hatem Ltaief, Ying Sun, Marc G. Genton, and David E. Keyes. "ExaGeoStat: A high performance unified software for geostatistics on manycore systems." IEEE Transactions on Parallel and Distributed Systems 29, no. 12 (2018): 2771-2784.

2. Sameh Abdulah, Hatem Ltaief, Ying Sun, Marc G. Genton, and David E. Keyes. "Parallel approximation of the maximum likelihood estimation for the prediction of large-scale geostatistics simulations." In 2018 IEEE International Conference on Cluster Computing (CLUSTER), pp. 98-108. IEEE, 2018.

3. Sameh Abdulah, Hatem Ltaief, Ying Sun, Marc G. Genton, and David E. Keyes. "Geostatistical modeling and prediction using mixed precision tile Cholesky factorization." In 2019 IEEE 26th international conference on high performance computing, data, and analytics (HiPC), pp. 152-162. IEEE, 2019.

4. Mary Lai O. Salvana, Sameh Abdulah, Huang Huang, Hatem Ltaief, Ying Sun, Marc G. Genton, and David E. Keyes. "High performance multivariate geospatial statistics on manycore systems." IEEE Transactions on Parallel and Distributed Systems 32, no. 11 (2021): 2719-2733.

5. Mary Lai O. Salvaña, Sameh Abdulah, Hatem Ltaief, Ying Sun, Marc G. Genton, and David E. Keyes. "Parallel Space-Time Likelihood Optimization for Air Pollution Prediction on Large-Scale Systems." In the Proceedings of the Platform for Advanced Scientific Computing Conference (PASC'22). Association for Computing Machinery, New York, NY, USA, Article 17, 1–11. ACM, 2022.

6. Sameh Abdulah, Qinglei Cao, Yu Pei, George Bosilca, Jack Dongarra, Marc G. Genton, David E. Keyes, Hatem Ltaief, and Ying Sun. "Accelerating geostatistical modeling and prediction with mixed-precision computations: A high-productivity approach with PaRSEC." IEEE Transactions on Parallel and Distributed Systems 33, no. 4 (2021): 964-976.
7. Sagnik Mondal, Sameh Abdulah, Hatem Ltaief, Ying Sun, Marc G. Genton, and David E. Keyes. "Parallel Approximations of the Tukey g-and-h Likelihoods and Predictions for Non-Gaussian Geostatistics." 2022 IEEE International Parallel and Distributed Processing Symposium (IPDPS), Lyon, France, 2022, pp. 379-389. IEEE, 2022.
8. Qinglei Cao, Sameh Abdulah, Rabab Alomairy, Yu Pei, Pratik Nag, George Bosilca, Jack Dongarra et al. "Reshaping geostatistical modeling and prediction for extreme-scale environmental applications." In 2022 SC22: International Conference for High-Performance Computing, Networking, Storage and Analysis (SC), pp. 13-24. IEEE Computer Society, 2022. (ACM GORDON BELL PRIZE Finalist).
9. Sagnik Mondal, Sameh Abdulah, Hatem Ltaief, Ying Sun, Marc G. Genton, and David E. Keyes. "Tile low-rank approximations of non-Gaussian space and space-time Tukey g-and-h random field likelihoods and predictions on large-scale systems." Journal of Parallel and Distributed Computing 180 (2023): 104715.
10. Qinglei Cao, Sameh Abdulah, Hatem Ltaief, Marc G. Genton, David E. Keyes, and George Bosilca. "Reducing Data Motion and Energy Consumption of Geospatial Modeling Applications Using Automated Precision Conversion." In 2023 IEEE International Conference on Cluster Computing (CLUSTER), IEEE, 2023.

License

Handout

Chapter 2

Include Subdirectory

This contains all the header files for the project.

File structure

- `api`: Directory contains the high-level drivers for the ExaGeoStat-cpp functionalities that are provided to library users. These functions help users interact with the ExaGeoStat-cpp framework and perform various statistical operations.
- `common`: Directory contains all ExaGeoStat-cpp common functionalities that might be used across the different modules of the ExaGeoStat-cpp framework.
- `configurations`: Directory contains all ExaGeoStat-cpp configurations arguments and parsers. These functions are used to parse and set the configuration parameters for the ExaGeoStat-cpp framework.
- `data-generators`: Directory contains the required methods to generate datasets, i.e., dense...etc
- `data-units`: Directory is used for all ExaGeoStat-cpp base data structures that the user should utilize and interact with. These data units are used to represent the data and perform operations on it.
- `hardware`: Directory contains the required methods to manage hardware allocations and de-allocations.
- `helpers`: Directory contains helper functions that can be used across the different modules of the Exa↩ GeoStat-cpp framework.
- `kernels`: Directory provide low-level implementations of the supported kernels offered by the ExaGeoStat-cpp framework.
- `linear-algebra-solvers`: Directory is used for all ExaGeoStat-cpp integrated linear algebra solvers libraries.
- `operators`: Directory contains various operators used by the ExaGeoStat-cpp framework. These operators are used to perform various mathematical operations on the data sets.

Chapter 3

Namespace Index

3.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 4

Hierarchical Index

4.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 5

Class Index

5.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 6

File Index

6.1 File List

Here is a list of all files with brief descriptions:

Chapter 7

Namespace Documentation

7.1 exageostat Namespace Reference

Namespaces

- adapters
- api
- common
- configurations
- dataLoader
- dataunits
- generators
- helpers
- kernels
- linearAlgebra
- plugins
- prediction
- results
- runtime

7.2 exageostat::adapters Namespace Reference

Functions

- std::vector< std::vector< double > > R_GetLocations (ExaGeoStatData< double > *apData)

  Retrieves locations from ExaGeoStat data.
- Rcpp::NumericVector R_GetDescZValues (ExaGeoStatData< double > *apData, const std::string &aType)

  Retrieves descriptive Z values from ExaGeoStat data based on type.
- ExaGeoStatData< double > * R_ExaGeoStatLoadData (const std::string &aKernelName, const std::vector< double > &aInitialTheta, const std::string &aDistanceMatrix, const int &aProblemSize, const int &aSeed, const int &aDenseTileSize, const int &aLowTileSize, const std::string &aDimension, const std::string &aLog↩ Path, const std::string &aDataPath, const std::string &aRecoveryFilePath, const std::string &aObservations↩ FilePath)

  Function to load ExaGeoStat data.

- std::vector< double > R_ExaGeoStatModelData (const std::string &aComputation, const std::string &aKernelName, const std::string &aDistanceMatrix, const std::vector< double > &aLowerBound, const std::vector< double > &aUpperBound, const int &aTolerance, const int &aMleIterations, const int &aDenseTileSize, const int &aLowTileSize, const std::string &aDimension, const int &aBand, const int &aMaxRank, SEXP apData, Rcpp::Nullable< Rcpp::NumericVector > aMeasurementsVector=R_NilValue, Rcpp::Nullable< Rcpp::NumericVector > aLocationsX=R_NilValue, Rcpp::Nullable< Rcpp::NumericVector > aLocationsY=R_NilValue, Rcpp::Nullable< Rcpp::NumericVector > aLocationsZ=R_NilValue)

  Models ExaGeoStat data using specified arguments.
- std::vector< double > R_ExaGeoStatPredictData (const std::string &aKernelName, const std::string &aDistanceMatrix, const std::vector< double > &aEstimatedTheta, const int &aDenseTileSize, const int &aLowTileSize, const std::string &aDimension, std::vector< std::vector< double >> &aTrainData, std::vector< std::vector< double >> &aTestData)

  Predicts outcomes using ExaGeoStat data and configurations.
- std::vector< double > R_ExaGeoStatMLOE_MMOM (const std::string &aKernelName, const std::string &aDistanceMatrix, const std::vector< double > &aEstimatedTheta, const std::vector< double > &aTrueTheta, const int &aDenseTileSize, const int &aLowTileSize, const std::string &aDimension, std::vector< std::vector< double >> &aTrainData, std::vector< std::vector< double >> &aTestData)

  Calculates the Mean Logarithmic Error (MLOE) and the Mean Measure of Model Output (MMOM) for ExaGeoStat predictions.
- std::vector< double > R_ExaGeoStatFisher (const std::string &aKernelName, const std::string &aDistanceMatrix, const std::vector< double > &aEstimatedTheta, const int &aDenseTileSize, const int &aLowTileSize, const std::string &aDimension, std::vector< std::vector< double >> &aTrainData, std::vector< std::vector< double >> &aTestData)

  Computes the Fisher information matrix for ExaGeoStat models.
- std::vector< double > R_ExaGeoStatIDW (const std::string &aKernelName, const std::string &aDistanceMatrix, const std::vector< double > &aEstimatedTheta, const int &aDenseTileSize, const int &aLowTileSize, const std::string &aDimension, std::vector< std::vector< double >> &aTrainData, std::vector< std::vector< double >> &aTestData, std::vector< double > &aTestMeasurementsValues)

  Applies Inverse Distance Weighting (IDW) for spatial interpolation using ExaGeoStat data.
- double * GetDataFromArguments (Rcpp::Nullable< Rcpp::NumericVector > aMeasurementsVector, Rcpp::Nullable< Rcpp::NumericVector > aLocationsX, Rcpp::Nullable< Rcpp::NumericVector > aLocationsY, Rcpp::Nullable< Rcpp::NumericVector > aLocationsZ, std::unique_ptr< ExaGeoStatData< double >> &aData, configurations::Configurations &aConfigurations, const std::string &aKernelName, const std::string &aDistanceMatrix, const int &aDenseTileSize, const int &aLowTileSize, const std::string &aDimension, const common::Computation &aComputation)

  Extracts and prepares data from given arguments for ExaGeoStat operations.
- void ValidateDataDimensions (const std::vector< std::vector< double >> &aData, const std::string &aDataType)

  Validates the dimensions of input data.
- void PredictionSetupHelper (configurations::Configurations &aConfigurations, const std::string &aKernelName, const std::string &aDistanceMatrix, const int &aDenseTileSize, const int &aLowTileSize, const std::string &aDimension, std::vector< std::vector< double >> &aTrainData, std::vector< std::vector< double >> &aTestData, const std::vector< double > &aEstimatedTheta, const std::vector< double > &aTestMeasurementsValues)

  Sets up the prediction environment.

7.2.1 Function Documentation

7.2.1.1 R_GetLocations()

```
std::vector<std::vector<double> > exageostat::adapters::R_GetLocations (
            ExaGeoStatData< double > * apData )
```

Retrieves locations from ExaGeoStat data.

Extracts and returns the locations stored in an ExaGeoStatData object,

Parameters

in	*apData*	Pointer to ExaGeoStatData object containing the spatial data.

Returns

vector of locations coordinates.

7.2.1.2 R_GetDescZValues()

```
Rcpp::NumericVector exageostat::adapters::R_GetDescZValues (
            ExaGeoStatData< double > * apData,
            const std::string & aType )
```

Retrieves descriptive Z values from ExaGeoStat data based on type.

Extracts and returns Z values from an ExaGeoStatData object, aiding in targeted spatial data analysis and visualization within ExaGeoStat.

Parameters

in	*apData*	Pointer to ExaGeoStatData object containing the spatial data.
in	*aType*	String specifying the type of descriptor value to retrieve (e.g., "Chameleon", "HiCMA").

Returns

Numeric vector of descriptive Z values.

7.2.1.3 R_ExaGeoStatLoadData()

```
ExaGeoStatData<double>* exageostat::adapters::R_ExaGeoStatLoadData (
            const std::string & aKernelName,
            const std::vector< double > & aInitialTheta,
            const std::string & aDistanceMatrix,
            const int & aProblemSize,
            const int & aSeed,
```

```
            const int & aDenseTileSize,
            const int & aLowTileSize,
            const std::string & aDimension,
            const std::string & aLogPath,
            const std::string & aDataPath,
            const std::string & aRecoveryFilePath,
            const std::string & aObservationsFilePath )
```

Function to load ExaGeoStat data.

This function loads data into an ExaGeoStatData object using the provided configuration and computational settings. It is designed to initialize the data structure necessary for subsequent statistical model operations within the ExaGeoStat framework.

Parameters

in	*aKernelName*	Name of the computational kernel to be utilized.
in	*aInitialTheta*	Initial parameter values for the statistical model.
in	*aDistanceMatrix*	Type of distance matrix to be used ("euclidean", "manhattan", etc.).
in	*aProblemSize*	Size of the problem or dataset.
in	*aSeed*	Seed for random number generation, ensuring reproducibility.
in	*aDenseTileSize*	Size of the tile for dense computations.
in	*aLowTileSize*	Size of the tile for low-rank computations.
in	*aDimension*	Dimensionality of the problem ("2D" for two dimensions, "3D" for three dimensions).
in	*aLogPath*	Path to the log file where execution details will be stored.
in	*aDataPath*	Path to the data file containing spatial observations.
in	*aRecoveryFilePath*	Path for saving intermediate computation states, aiding in recovery from interruptions.
in	*aObservationsFilePath*	Path to the file containing observation data.

Returns

A pointer to an ExaGeoStatData object containing the loaded data.

7.2.1.4 R_ExaGeoStatModelData()

```
std::vector<double> exageostat::adapters::R_ExaGeoStatModelData (
            const std::string & aComputation,
            const std::string & aKernelName,
            const std::string & aDistanceMatrix,
            const std::vector< double > & aLowerBound,
            const std::vector< double > & aUpperBound,
            const int & aTolerance,
            const int & aMleIterations,
            const int & aDenseTileSize,
            const int & aLowTileSize,
            const std::string & aDimension,
            const int & aBand,
            const int & aMaxRank,
```

```
            SEXP apData,
            Rcpp::Nullable< Rcpp::NumericVector > aMeasurementsVector = R_NilValue,
            Rcpp::Nullable< Rcpp::NumericVector > aLocationsX = R_NilValue,
            Rcpp::Nullable< Rcpp::NumericVector > aLocationsY = R_NilValue,
            Rcpp::Nullable< Rcpp::NumericVector > aLocationsZ = R_NilValue )
```

Models ExaGeoStat data using specified arguments.

Applies statistical modeling to ExaGeoStatData based on the provided configurations. This function is essential for preparing the data for in-depth statistical analysis and predictions, optimizing internal representations and parameters for the modeling process.

Parameters

in	*aComputation*	Computational method to be used.
in	*aKernelName*	Name of the kernel for computations.
in	*aDistanceMatrix*	Type of distance matrix ("euclidean", "manhattan", etc.).
in	*aLowerBound*	Lower bound for optimization parameters.
in	*aUpperBound*	Upper bound for optimization parameters.
in	*aTolerance*	Tolerance level for the optimization algorithm.
in	*aMleIterations*	Maximum number of iterations for the Maximum Likelihood Estimation (MLE) algorithm.
in	*aDenseTileSize*	Tile size for dense matrix computations.
in	*aLowTileSize*	Tile size for low-rank approximations.
in	*aDimension*	Dimensionality of the problem ("2D" or "3D").
in	*aBand*	Bandwidth for band matrices, applicable in certain computational kernels.
in	*aMaxRank*	Maximum rank for low-rank approximations.
in	*apData*	Pointer to ExaGeoStatData object to be modeled.
in	*aMeasurementsVector*	Optional vector of measurements to enhance modeling, can be nullable.
in	*aLocationsX*	Optional vector of X coordinates for locations, can be nullable.
in	*aLocationsY*	Optional vector of Y coordinates for locations, can be nullable.
in	*aLocationsZ*	Optional vector of Z coordinates for locations, can be nullable.

Returns

Vector of doubles represents the modeled theta.

7.2.1.5 R_ExaGeoStatPredictData()

```
std::vector<double> exageostat::adapters::R_ExaGeoStatPredictData (
            const std::string & aKernelName,
            const std::string & aDistanceMatrix,
            const std::vector< double > & aEstimatedTheta,
            const int & aDenseTileSize,
            const int & aLowTileSize,
            const std::string & aDimension,
            std::vector< std::vector< double >> & aTrainData,
            std::vector< std::vector< double >> & aTestData )
```

Predicts outcomes using ExaGeoStat data and configurations.

Utilizes a modeled ExaGeoStatData object to perform predictions, leveraging specified computational settings and statistical models. This function is integral for generating spatial predictions based on the data and models within the ExaGeoStat framework.

Parameters

in	*aKernelName*	Name of the kernel used for prediction computations.
in	*aDistanceMatrix*	Type of distance matrix used ("euclidean", "manhattan", etc.).
in	*aEstimatedTheta*	Vector of estimated parameters from the model.
in	*aDenseTileSize*	Tile size for dense matrix operations.
in	*aLowTileSize*	Tile size for low-rank matrix operations.
in	*aDimension*	Dimensionality of the spatial data ("2D" or "3D").
in	*aTrainData*	Training data set used for predictions.
in	*aTestData*	Test data set for which predictions are made.

Returns

Vector of predicted values based on the test data.

7.2.1.6 R_ExaGeoStatMLOE_MMOM()

```
std::vector<double> exageostat::adapters::R_ExaGeoStatMLOE_MMOM (
            const std::string & aKernelName,
            const std::string & aDistanceMatrix,
            const std::vector< double > & aEstimatedTheta,
            const std::vector< double > & aTrueTheta,
            const int & aDenseTileSize,
            const int & aLowTileSize,
            const std::string & aDimension,
            std::vector< std::vector< double >> & aTrainData,
            std::vector< std::vector< double >> & aTestData )
```

Calculates the Mean Logarithmic Error (MLOE) and the Mean Measure of Model Output (MMOM) for ExaGeoStat predictions.

Assesses the accuracy of spatial predictions made by the ExaGeoStat framework by computing the MLOE and MMOM, which provide insights into the predictive performance and uncertainty of the models.

Parameters

in	*aKernelName*	Kernel used for the prediction computations.
in	*aDistanceMatrix*	Type of distance matrix ("euclidean", "manhattan", etc.).
in	*aEstimatedTheta*	Vector of estimated parameters from the model.
in	*aTrueTheta*	Vector of true parameter values for validation.
in	*aDenseTileSize*	Tile size for dense matrix operations.
in	*aLowTileSize*	Tile size for low-rank matrix operations.
in	*aDimension*	Dimensionality of the spatial data ("2D" or "3D").
in	*aTrainData*	Training data set used in the model.
in	*aTestData*	Test data set used for validation.

Returns

Vector containing the calculated MLOE and MMOM values.

7.2.1.7 R_ExaGeoStatFisher()

```
std::vector<double> exageostat::adapters::R_ExaGeoStatFisher (
            const std::string & aKernelName,
            const std::string & aDistanceMatrix,
            const std::vector< double > & aEstimatedTheta,
            const int & aDenseTileSize,
            const int & aLowTileSize,
            const std::string & aDimension,
            std::vector< std::vector< double >> & aTrainData,
            std::vector< std::vector< double >> & aTestData )
```

Computes the Fisher information matrix for ExaGeoStat models.

Utilizes the estimated parameters and the Fisher information matrix to evaluate the information content and parameter uncertainties within the ExaGeoStat framework, contributing to the understanding of model reliability and sensitivity.

Parameters

in	*aKernelName*	Kernel used for computations.
in	*aDistanceMatrix*	Type of distance matrix ("euclidean", "manhattan", etc.).
in	*aEstimatedTheta*	Vector of estimated parameters from the model.
in	*aDenseTileSize*	Tile size for dense matrix operations.
in	*aLowTileSize*	Tile size for low-rank matrix operations.
in	*aDimension*	Dimensionality of the spatial data ("2D" or "3D").
in	*aTrainData*	Training data set used in the model.
in	*aTestData*	Test data set used for validation.

Returns

Vector represents the Fisher information matrix.

7.2.1.8 R_ExaGeoStatIDW()

```
std::vector<double> exageostat::adapters::R_ExaGeoStatIDW (
            const std::string & aKernelName,
            const std::string & aDistanceMatrix,
            const std::vector< double > & aEstimatedTheta,
            const int & aDenseTileSize,
            const int & aLowTileSize,
            const std::string & aDimension,
            std::vector< std::vector< double >> & aTrainData,
```

```
            std::vector< std::vector< double >> & aTestData,
            std::vector< double > & aTestMeasurementsValues )
```

Applies Inverse Distance Weighting (IDW) for spatial interpolation using ExaGeoStat data.

Implements the IDW interpolation method to estimate spatial variables at unsampled locations based on the distances and values of nearby sampled points within the ExaGeoStat framework, enhancing spatial prediction capabilities.

Parameters

in	*aKernelName*	Kernel used for IDW computations.
in	*aDistanceMatrix*	Type of distance matrix ("euclidean", "manhattan", etc.).
in	*aEstimatedTheta*	Vector of parameters, typically used for weighting in IDW.
in	*aDenseTileSize*	Tile size for dense matrix operations.
in	*aLowTileSize*	Tile size for low-rank matrix operations.
in	*aDimension*	Dimensionality of the spatial data ("2D" or "3D").
in	*aTrainData*	Training data set providing sampled locations and values.
in	*aTestData*	Test data set providing unsampled locations for which values are interpolated.
in	*aTestMeasurementsValues*	Vector of measured values at the test locations, used as reference in some IDW implementations.

Returns

Vector of interpolated values at the test locations.

7.2.1.9 GetDataFromArguments()

```
double* exageostat::adapters::GetDataFromArguments (
            Rcpp::Nullable< Rcpp::NumericVector > aMeasurementsVector,
            Rcpp::Nullable< Rcpp::NumericVector > aLocationsX,
            Rcpp::Nullable< Rcpp::NumericVector > aLocationsY,
            Rcpp::Nullable< Rcpp::NumericVector > aLocationsZ,
            std::unique_ptr< ExaGeoStatData< double >> & aData,
            configurations::Configurations & aConfigurations,
            const std::string & aKernelName,
            const std::string & aDistanceMatrix,
            const int & aDenseTileSize,
            const int & aLowTileSize,
            const std::string & aDimension,
            const common::Computation & aComputation )
```

Extracts and prepares data from given arguments for ExaGeoStat operations.

This function is designed to parse and prepare spatial and measurement data from provided arguments, making it suitable for processing within the ExaGeoStat framework. It handles optional data vectors for measurements and locations (X, Y, Z coordinates), and configures an ExaGeoStatData object based on these inputs along with other computational and configuration parameters.

Parameters

in	*aMeasurementsVector*	vector of measurements to enhance modeling, can be nullable.
in	*aLocationsX*	vector of X coordinates for locations, can be nullable.
in	*aLocationsY*	vector of Y coordinates for locations, can be nullable.
in	*aLocationsZ*	vector of Z coordinates for locations, can be nullable.
in	*aData*	Pointer to ExaGeoStatData object to be modeled.
in	*aConfigurations*	Configuration settings specifying computational details such as the kernel type, matrix storage format, etc.
in	*aKernelName*	Name of the kernel for computations.
in	*aDistanceMatrix*	Type of distance matrix ("euclidean", "manhattan", etc.).
in	*aDenseTileSize*	Tile size for dense matrix computations.
in	*aLowTileSize*	Tile size for low-rank approximations.
in	*aDimension*	Dimensionality of the problem ("2D" or "3D").
in	*aComputation*	Computational method to be used.

Returns

Pointer to a double array containing the prepared data, ready for use in ExaGeoStat operations.

7.2.1.10 ValidateDataDimensions()

```
void exageostat::adapters::ValidateDataDimensions (
            const std::vector< std::vector< double >> & aData,
            const std::string & aDataType )
```

Validates the dimensions of input data.

This function checks the dimensions of the provided data vectors to ensure they meet the expected format and size requirements for a given data type. It's used to verify that the data structures passed into algorithms or processes are correctly formatted, preventing errors or inconsistencies in data processing.

Parameters

aData	A constant reference to a vector of vectors containing the data to be validated.
aDataType	A string describing the type of data being validated, which influences the expected dimensions and format of the data.

Returns

void

7.2.1.11 PredictionSetupHelper()

```
void exageostat::adapters::PredictionSetupHelper (
            configurations::Configurations & aConfigurations,
```

```
const std::string & aKernelName,
const std::string & aDistanceMatrix,
const int & aDenseTileSize,
const int & aLowTileSize,
const std::string & aDimension,
std::vector< std::vector< double >> & aTrainData,
std::vector< std::vector< double >> & aTestData,
const std::vector< double > & aEstimatedTheta,
const std::vector< double > & aTestMeasurementsValues )
```

Sets up the prediction environment.

This function prepares the necessary configurations and data structures for making predictions. It involves setting up various parameters, including kernel names, distance matrices, tile sizes, dimensions, and training/test data. The function is crucial for initializing the prediction process with the appropriate settings and data.

Parameters

aConfigurations	Reference to a Configurations object containing various prediction and algorithm configurations.
aKernelName	Name of the kernel to be used in predictions.
aDistanceMatrix	String representation of the distance matrix to be used.
aDenseTileSize	Size of the dense tiles in the matrix.
aLowTileSize	Size of the low-resolution tiles in the matrix.
aDimension	String representation of the dimensionality of the data.
aTrainData	Reference to a vector of vectors containing the training data.
aTestData	Reference to a vector of vectors containing the test data.
aEstimatedTheta	Vector containing estimated theta values for the model.
aTestMeasurementsValues	Vector containing the test measurement values.

Returns

Pointer to a double array containing the prepared data, ready for use in ExaGeoStat operations.

7.3 exageostat::api Namespace Reference

Classes

- class ExaGeoStat

  High-Level Wrapper class containing the static API for ExaGeoStat operations.

7.4 exageostat::common Namespace Reference

Typedefs

- typedef enum exageostat::common::TileStorage ExaGeoStatTileStorage

Enumerations

- enum Verbose { QUIET_MODE = 0 , STANDARD_MODE = 1 , DETAILED_MODE = 2 }
- enum Dimension { Dimension2D = 0 , Dimension3D = 1 , DimensionST = 2 }

  Enum denoting the dimension of generated data.
- enum Side { EXAGEOSTAT_LEFT = 141 , EXAGEOSTAT_RIGHT = 142 }

  Enum denoting the side on which the matrix appears in an equation.
- enum Trans { EXAGEOSTAT_NO_TRANS = 111 , EXAGEOSTAT_TRANS = 112 , EXAGEOSTAT_CONJ_TRANS = 113 }

  Enum denoting whether or not to transpose a matrix.
- enum Diag { EXAGEOSTAT_NON_UNIT = 131 , EXAGEOSTAT_UNIT = 132 }

  Enum denoting whether the diagonal is unitary.
- enum DistanceMetric { EUCLIDEAN_DISTANCE = 0 , GREAT_CIRCLE_DISTANCE = 1 }
- enum DescriptorType { CHAMELEON_DESCRIPTOR = 0 , HICMA_DESCRIPTOR = 1 }
- enum DataSourceType { SYNTHETIC = 0 , CSV_FILE = 1 }
- enum DescriptorName : int {
  DESCRIPTOR_C = 0 , DESCRIPTOR_Z = 1 , DESCRIPTOR_Z_COPY = 2 , DESCRIPTOR_PRODUCT = 3 , DESCRIPTOR_DETERMINANT = 4 , DESCRIPTOR_CD = 5 , DESCRIPTOR_CUV = 6 , DESCRIPTOR_CRK = 7 ,
  DESCRIPTOR_Z_OBSERVATIONS = 8 , DESCRIPTOR_Z_Actual = 9 , DESCRIPTOR_Z_MISS = 10 , DESCRIPTOR_MSPE = 11 ,
  DESCRIPTOR_Z_1 = 12 , DESCRIPTOR_Z_2 = 13 , DESCRIPTOR_Z_3 = 14 , DESCRIPTOR_PRODUCT_1 = 15 ,
  DESCRIPTOR_PRODUCT_2 = 16 , DESCRIPTOR_PRODUCT_3 = 17 , DESCRIPTOR_C11 = 18 , DESCRIPTOR_C12 = 19 ,
  DESCRIPTOR_C22 = 20 , DESCRIPTOR_C12D = 21 , DESCRIPTOR_C12UV = 22 , DESCRIPTOR_C12RK = 23 ,
  DESCRIPTOR_C22D = 24 , DESCRIPTOR_C22UV = 25 , DESCRIPTOR_C22RK = 26 , DESCRIPTOR_MSPE_1 = 27 ,
  DESCRIPTOR_MSPE_2 = 28 , DESCRIPTOR_k_T = 29 , DESCRIPTOR_k_A = 30 , DESCRIPTOR_k_A_TMP = 31 ,
  DESCRIPTOR_k_T_TMP = 32 , DESCRIPTOR_K_T = 33 , DESCRIPTOR_K_T_TMP = 34 , DESCRIPTOR_K_A = 35 ,
  DESCRIPTOR_EXPR_1 = 36 , DESCRIPTOR_EXPR_2 = 37 , DESCRIPTOR_EXPR_3 = 38 , DESCRIPTOR_EXPR_4 = 39 ,
  DESCRIPTOR_MLOE = 40 , DESCRIPTOR_MMOM = 41 , DESCRIPTOR_MLOE_MMOM = 42 , DESCRIPTOR_ALPHA = 43 ,
  DESCRIPTOR_TRUTH_ALPHA = 44 , DESCRIPTOR_TIMATED_ALPHA = 45 , DESCRIPTOR_CK = 46 , DESCRIPTOR_CJ = 47 ,
  DESCRIPTOR_C_TRACE = 48 , DESCRIPTOR_C_DIAG = 49 , DESCRIPTOR_A = 50 , DESCRIPTOR_RESULTS = 51 ,
  DESCRIPTOR_SUM = 52 , DESCRIPTOR_R = 53 , DESCRIPTOR_R_COPY = 54 }
- enum TileStorage {
  EXAGOSTAT_CM = 101 , EXAGOSTAT_RM = 102 , EXAGOSTAT_CCRB = 103 , EXAGOSTAT_CRRB = 104 ,
  EXAGOSTAT_RCRB = 105 , EXAGOSTAT_RRRB = 106 }
- enum Computation { EXACT_DENSE = 0 , DIAGONAL_APPROX = 1 , TILE_LOW_RANK = 2 }

  Enum denoting the types of computations that can be requested, to use the required Linear Algebra solver library.
- enum Precision { SINGLE = 0 , DOUBLE = 1 , MIXED = 2 }

  Enum denoting the precision of operations that are supported to be done on the matrix.
- enum FloatPoint : int {
  EXAGEOSTAT_BYTE = 0 , EXAGEOSTAT_INTEGER = 1 , EXAGEOSTAT_REAL_FLOAT = 2 , EXAGEOSTAT_REAL_DOUBLE = 3 ,
  EXAGEOSTAT_COMPLEX_FLOAT = 4 , EXAGEOSTAT_COMPLEX_DOUBLE = 5 }

  Enum denoting the floating point arithmetic of the matrix.

- enum UpperLower : int { EXAGEOSTAT_UPPER = 121 , EXAGEOSTAT_LOWER = 122 , EXAGEOSTAT_UPPER_LOWER = 123 }

  Enum denoting the Upper/Lower part.
- enum CopyDirection : int { CHAMELEON_TO_HICMA = 0 , HICMA_TO_CHAMELEON = 1 }

  Enum denoting the copy descriptors flow.

Variables

- static const std::set< std::string > availableKernels

  Set denoting the available kernels supported in matrix generation.

7.4.1 Typedef Documentation

7.4.1.1 ExaGeoStatTileStorage

```
typedef enum exageostat::common::TileStorage exageostat::common::ExaGeoStatTileStorage
```

7.4.2 Enumeration Type Documentation

7.4.2.1 Verbose

```
enum exageostat::common::Verbose
```

Enumerator

QUIET_MODE	
STANDARD_MODE	
DETAILED_MODE	

7.4.2.2 Dimension

```
enum exageostat::common::Dimension
```

Enum denoting the dimension of generated data.

Enumerator

Dimension2D	
Dimension3D	
DimensionST	

7.4.2.3 Side

```
enum exageostat::common::Side
```

Enum denoting the side on which the matrix appears in an equation.

Enumerator

EXAGEOSTAT_LEFT	
EXAGEOSTAT_RIGHT	

7.4.2.4 Trans

```
enum exageostat::common::Trans
```

Enum denoting whether or not to transpose a matrix.

Enumerator

EXAGEOSTAT_NO_TRANS	
EXAGEOSTAT_TRANS	
EXAGEOSTAT_CONJ_TRANS	

7.4.2.5 Diag

```
enum exageostat::common::Diag
```

Enum denoting whether the diagonal is unitary.

Enumerator

EXAGEOSTAT_NON_UNIT	
EXAGEOSTAT_UNIT	

7.4.2.6 DistanceMetric

```
enum exageostat::common::DistanceMetric
```

Enumerator

EUCLIDEAN_DISTANCE	
GREAT_CIRCLE_DISTANCE	

7.4.2.7 DescriptorType

`enum exageostat::common::DescriptorType`

Enumerator

CHAMELEON_DESCRIPTOR	
HICMA_DESCRIPTOR	

7.4.2.8 DataSourceType

`enum exageostat::common::DataSourceType`

Enumerator

SYNTHETIC	
CSV_FILE	

7.4.2.9 DescriptorName

`enum exageostat::common::DescriptorName : int`

Enumerator

DESCRIPTOR_C	
DESCRIPTOR_Z	
DESCRIPTOR_Z_COPY	
DESCRIPTOR_PRODUCT	
DESCRIPTOR_DETERMINANT	
DESCRIPTOR_CD	
DESCRIPTOR_CUV	
DESCRIPTOR_CRK	
DESCRIPTOR_Z_OBSERVATIONS	
DESCRIPTOR_Z_Actual	
DESCRIPTOR_Z_MISS	
DESCRIPTOR_MSPE	

Enumerator

DESCRIPTOR_Z_1	
DESCRIPTOR_Z_2	
DESCRIPTOR_Z_3	
DESCRIPTOR_PRODUCT_1	
DESCRIPTOR_PRODUCT_2	
DESCRIPTOR_PRODUCT_3	
DESCRIPTOR_C11	
DESCRIPTOR_C12	
DESCRIPTOR_C22	
DESCRIPTOR_C12D	
DESCRIPTOR_C12UV	
DESCRIPTOR_C12RK	
DESCRIPTOR_C22D	
DESCRIPTOR_C22UV	
DESCRIPTOR_C22RK	
DESCRIPTOR_MSPE_1	
DESCRIPTOR_MSPE_2	
DESCRIPTOR_k_T	
DESCRIPTOR_k_A	
DESCRIPTOR_k_A_TMP	
DESCRIPTOR_k_T_TMP	
DESCRIPTOR_K_T	
DESCRIPTOR_K_T_TMP	
DESCRIPTOR_K_A	
DESCRIPTOR_EXPR_1	
DESCRIPTOR_EXPR_2	
DESCRIPTOR_EXPR_3	
DESCRIPTOR_EXPR_4	
DESCRIPTOR_MLOE	
DESCRIPTOR_MMOM	
DESCRIPTOR_MLOE_MMOM	
DESCRIPTOR_ALPHA	
DESCRIPTOR_TRUTH_ALPHA	
DESCRIPTOR_TIMATED_ALPHA	
DESCRIPTOR_CK	
DESCRIPTOR_CJ	
DESCRIPTOR_C_TRACE	
DESCRIPTOR_C_DIAG	
DESCRIPTOR_A	
DESCRIPTOR_RESULTS	
DESCRIPTOR_SUM	
DESCRIPTOR_R	
DESCRIPTOR_R_COPY	

7.4.2.10 TileStorage

```
enum exageostat::common::TileStorage
```

Enumerator

EXAGOSTAT_CM	
EXAGOSTAT_RM	
EXAGOSTAT_CCRB	
EXAGOSTAT_CRRB	
EXAGOSTAT_RCRB	
EXAGOSTAT_RRRB	

7.4.2.11 Computation

`enum exageostat::common::Computation`

Enum denoting the types of computations that can be requested, to use the required Linear Algebra solver library.

Enumerator

EXACT_DENSE	
DIAGONAL_APPROX	
TILE_LOW_RANK	

7.4.2.12 Precision

`enum exageostat::common::Precision`

Enum denoting the precision of operations that are supported to be done on the matrix.

Enumerator

SINGLE	
DOUBLE	
MIXED	

7.4.2.13 FloatPoint

`enum exageostat::common::FloatPoint : int`

Enum denoting the floating point arithmetic of the matrix.

Enumerator

EXAGEOSTAT_BYTE	

Enumerator

EXAGEOSTAT_INTEGER	
EXAGEOSTAT_REAL_FLOAT	
EXAGEOSTAT_REAL_DOUBLE	
EXAGEOSTAT_COMPLEX_FLOAT	
EXAGEOSTAT_COMPLEX_DOUBLE	

7.4.2.14 UpperLower

```
enum exageostat::common::UpperLower :  int
```

Enum denoting the Upper/Lower part.

Enumerator

EXAGEOSTAT_UPPER	Use lower triangle of A
EXAGEOSTAT_LOWER	Use upper triangle of A
EXAGEOSTAT_UPPER_LOWER	Use the full A

7.4.2.15 CopyDirection

```
enum exageostat::common::CopyDirection :  int
```

Enum denoting the copy descriptors flow.

Enumerator

CHAMELEON_TO_HICMA	
HICMA_TO_CHAMELEON	

7.4.3 Variable Documentation

7.4.3.1 availableKernels

```
exageostat::common::availableKernels  [static]
```

Set denoting the available kernels supported in matrix generation.

This set is updated automatically to add new kernels. The set is initialized with a lambda function that iterates through a directory and extracts the kernel names from the filenames. It also adds lowercase versions of the kernel names with underscores before each capital letter.

Returns

set of all available kernels names

7.5 exageostat::configurations Namespace Reference

Classes

- class Configurations

    Contains methods to set and get.

7.6 exageostat::dataLoader Namespace Reference

Namespaces

- csv

Classes

- class DataLoader

    Extends DataGenerator to include data loading functionalities.

7.7 exageostat::dataLoader::csv Namespace Reference

Classes

- class CSVLoader

    A class for creating data by reading CSV files.

7.8 exageostat::dataunits Namespace Reference

Namespaces

- descriptor

Classes

- union BaseDescriptor

    Union represents the base descriptor.
- class DescriptorData

    Manages geo-statistical descriptor data with functions for retrieving and manipulating descriptors.
- class Locations

    A class containing methods to set and get location data.
- struct mModelingData

    Struct containing all the data needed for modeling.

7.9 exageostat::dataunits::descriptor Namespace Reference

Classes

- class ChameleonDescriptor

  ChameleonDescriptor is a class for creating matrix descriptors by CHAMELEON library.
- class HicmaDescriptor

  HicmaDescriptor is a class for creating matrix descriptors by HICMA library.
- class ExaGeoStatDescriptor

  ExaGeoStatDescriptor is a class for creating matrix descriptors used in CHAMELEON and HiCMA libraries.

7.9.1 Detailed Description

Tile matrix descriptor

Matrices are stored in a contiguous data chunk containing in order A11, A21, A12, A22 with :

```
     n1       n2
+----------+---+
|          |   |    With m1 = lm - (lm%mb)
|          |   |         m2 = lm%mb
```

m1 | A11 |A12| n1 = ln - (lnnb) | | | n2 = lnnb | | | +--------—+—+ m2 | A21 |A22| +--------—+—+

7.10 exageostat::generators Namespace Reference

Namespaces

- synthetic

Classes

- class DataGenerator

  Abstract base class for generating synthetic or real data.
- class LocationGenerator

  Generates spatial locations based on given parameters.

7.11 exageostat::generators::synthetic Namespace Reference

Classes

- class SyntheticGenerator

  A class for generating synthetic data.

7.12 exageostat::helpers Namespace Reference

Classes

- class BasselFunction

  The BasselFunction class provides methods for computing various derivatives of the modified Bessel function of the second kind, (K_{\nu}). This class is templated to support both float and double data types, enabling precision-based computations as required by different applications.
- class CommunicatorMPI

  A class for Communicating MPI rank.
- class DistanceCalculationHelpers

  Class to calculate the distance between two points.

Functions

- uint64_t SpreadBits (uint64_t aInputByte)

  Spread bits by three spaces.
- uint64_t ReverseSpreadBits (uint64_t aInputByte)

  Reverse Spread bits operation.
- bool CompareUint64 (const uint64_t &aFirstValue, const uint64_t &aSecondValue)

  Compares two Unit64 values.

7.12.1 Function Documentation

7.12.1.1 SpreadBits()

```
uint64_t exageostat::helpers::SpreadBits (
            uint64_t aInputByte )
```

Spread bits by three spaces.

Parameters

in	*aInputByte*	The input 64 bit to be spread.

Returns

The byte after being spread.

7.12.1.2 ReverseSpreadBits()

```
uint64_t exageostat::helpers::ReverseSpreadBits (
            uint64_t aInputByte )
```

Reverse Spread bits operation.

Parameters

in	*aInputByte*	The input spread 64 bit to be compacted.

Returns

The byte after being compacted.

7.12.1.3 CompareUint64()

```
bool exageostat::helpers::CompareUint64 (
            const uint64_t & aFirstValue,
            const uint64_t & aSecondValue )
```

Compares two Unit64 values.

Parameters

in	*aFirstValue*	Constant reference to the first input 64 bit value.
in	*aSecondValue*	Constant reference to the second input 64 bit value.

Returns

True if the second value is bigger than the first value, false otherwise.

7.13 exageostat::kernels Namespace Reference

Classes

- class BivariateMaternFlexible

  A class represents a Bivariate Matern Flexible kernel.
- class BivariateMaternParsimonious

  A class represents a Bivariate Matern Parsimonious kernel.
- class BivariateSpacetimeMaternStationary

  A class represents a Bivariate Spacetime Matern Stationary kernel.
- class TrivariateMaternParsimonious

  A class represents a Trivariate Matern Parsimonious kernel.
- class UnivariateExpNonGaussian

  A class represents a Univariate Exp Non Gaussian kernel.
- class UnivariateMaternDbeta

  A class represents a Univariate Matern Dbeta kernel.
- class UnivariateMaternDdbetaBeta

  A class represents a Univariate Matern Ddbeta Beta kernel.
- class UnivariateMaternDdbetaNu

  A class represents a Univariate Matern Ddbeta Nu kernel.

- class UnivariateMaternDdnuNu

  A class represents a Univariate Matern Ddnu Nu kernel.
- class UnivariateMaternDdsigmaSquare

  A class represents a Univariate Matern Ddsigma Square kernel.
- class UnivariateMaternDdsigmaSquareBeta

  A class represents a Univariate Matern Ddsigma Square Beta kernel.
- class UnivariateMaternDdsigmaSquareNu

  A class represents a Univariate Matern Ddsigma Square Nu kernel.
- class UnivariateMaternDnu

  A class represents a Univariate Matern Dnu kernel.
- class UnivariateMaternDsigmaSquare

  A class represents a Univariate Matern Dsigma Square kernel.
- class UnivariateMaternNonGaussian

  A class represents a Univariate Matern Non Gaussian kernel.
- class UnivariateMaternNuggetsStationary

  A class represents a Univariate Matern Nuggets Stationary kernel.
- class UnivariateMaternStationary

  A class represents a Univariate Matern Stationary kernel.
- class UnivariatePowExpStationary

  A class represents a Univariate PowExp Stationary kernel.
- class UnivariateSpacetimeMaternStationary

  A class represents a Univariate Spacetime Matern Stationary kernel.
- struct KernelsConfigurations
- class Kernel

7.14 exageostat::linearAlgebra Namespace Reference

Namespaces

- dense
- diagonalSuperTile
- tileLowRank

Classes

- class ChameleonImplementation

  ChameleonImplementation is a concrete implementation of LinearAlgebraMethods class for dense or diagonal-super tile matrices.
- class LinearAlgebraFactory

  A class that creates linear algebra solvers based on the input computation type.
- class LinearAlgebraMethods

  A class that defines the interface for linear algebra solvers.

7.15 exageostat::linearAlgebra::dense Namespace Reference

Classes

- class ChameleonDense

  ChameleonImplementationDense is a concrete implementation for dense matrices using Chameleon..

7.16 exageostat::linearAlgebra::diagonalSuperTile Namespace Reference

Classes

- class ChameleonDST

  ChameleonImplementationDST is a concrete implementation of LinearAlgebraMethods class for diagonal super tile matrices.

7.17 exageostat::linearAlgebra::tileLowRank Namespace Reference

Classes

- class HicmaImplementation

  HicmaImplementation is a concrete implementation of LinearAlgebraMethods class for tile low-rank matrices.

7.18 exageostat::plugins Namespace Reference

Classes

- class PluginRegistry

  Template class for registering and creating plugins.

7.19 exageostat::prediction Namespace Reference

Classes

- class Prediction

  Class to handle different Prediction Module calls.
- class PredictionAuxiliaryFunctions

  Class to define and implement different Prediction Module Auxiliary Functions.
- class PredictionHelpers

  Class to define and implement different Prediction Module helpers functions.

7.20 exageostat::results Namespace Reference

Classes

- class Results

7.21 exageostat::runtime Namespace Reference

Classes

- class RuntimeFunctions

  A class that defines runtime static functions.
- class DCMGCodelet
- class DDOTPCodelet
- class DMDETCodelet
- class DmloeMmomCodelet
- class DMSEBivariateCodelet
- class DMSECodelet
- class DTRACECodelet
- class DZCPYCodelet
- class GaussianCodelet
- class NonGaussianLoglike

  A class for starpu codelet non gaussian loglike.
- class NonGaussianTransform

  A class for starpu codelet non gaussian transform.
- class STRIDEVECCodelet
- class TriStrideVecCodelet

  A class for starpu codelet tri_stride_vec.
- class ChameleonStarPuHelpers

  ChameleonStarPuHelpers is a concrete implementation of StarPuHelpers interface for Chameleon library.
- class HicmaStarPuHelpers

  HicmaStarPuHelpers is a concrete implementation of StarPuHelpers interface for Hicma library.
- class StarPuHelpers

  A class that defines the interface for StarPu helpers.
- class StarPuHelpersFactory

  A class that creates StarPu helpers based on the input computation type.

Chapter 8

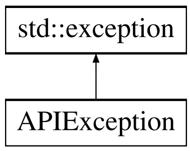
Class Documentation

8.1 APIException Class Reference

Custom exception class for handling API errors and warnings.

```
#include <ErrorHandler.hpp>
```

Inheritance diagram for APIException:



Public Member Functions

- APIException (const std::string &aMessage, const ErrorType &aErrorCode)

    Constructor for APIException.
- ∼APIException () override=default

    Destructor for APIException.

8.1.1 Detailed Description

Custom exception class for handling API errors and warnings.

8.1.2 Constructor & Destructor Documentation

8.1.2.1 APIException()

```
APIException::APIException (
            const std::string & aMessage,
            const ErrorType & aErrorCode )  [inline]
```

Constructor for APIException.

Parameters

in	*aMessage*	The error or warning message.
in	*aErrorCode*	The error type.

8.1.2.2 ∼APIException()

```
APIException::~APIException ( ) [override], [default]
```

Destructor for APIException.

The documentation for this class was generated from the following file:

- ErrorHandler.hpp

8.2 exageostat::dataunits::BaseDescriptor Union Reference

Union represents the base descriptor.

```
#include <DescriptorData.hpp>
```

Public Attributes

- CHAM_desc_t ∗ chameleon_desc

8.2.1 Detailed Description

Union represents the base descriptor.

This union is used to store different types of descriptors based on the configuration.

8.2.2 Member Data Documentation

8.2.2.1 chameleon_desc

```
CHAM_desc_t* exageostat::datacunits::BaseDescriptor::chameleon_desc
```

The documentation for this union was generated from the following file:

- DescriptorData.hpp

8.3 exageostat::helpers::BasselFunction< T > Class Template Reference

The BasselFunction class provides methods for computing various derivatives of the modified Bessel function of the second kind, (K_{\nu}). This class is templated to support both float and double data types, enabling precision-based computations as required by different applications.

```
#include <BasselFunction.hpp>
```

Static Public Member Functions

- static T CalculateDerivativeBesselNu (const T &aOrder, const T &aInputValue)

  Calculates the derivative of the modified Bessel function of the second kind (K_nu) with respect to its order, evaluated at input_value and order aOrder.
- static T CalculateSecondDerivativeBesselNu (const T &aOrder, const T &aInputValue)

  Calculates the second derivative of the modified Bessel function of the second kind (K_nu) with respect to its input, evaluated at input_value and order aOrder.
- static T CalculateSecondDerivativeBesselNuInput (const T &aOrder, const T &aInputValue)

  Calculates the second derivative of the modified Bessel function of the second kind (K_nu) with respect to its input, evaluated at input_value and order aOrder.

8.3.1 Detailed Description

template<typename T>
class exageostat::helpers::BasselFunction< T >

The BasselFunction class provides methods for computing various derivatives of the modified Bessel function of the second kind, (K_{\nu}). This class is templated to support both float and double data types, enabling precision-based computations as required by different applications.

Template Parameters

T	Data Type: float or double

8.3.2 Member Function Documentation

8.3.2.1 CalculateDerivativeBesselNu()

```
template<typename T >
static T exageostat::helpers::BasselFunction< T >::CalculateDerivativeBesselNu (
            const T & aOrder,
            const T & aInputValue )  [static]
```

Calculates the derivative of the modified Bessel function of the second kind (K_nu) with respect to its order, evaluated at input_value and order aOrder.

Parameters

in	*aOrder*	The order of the Bessel function.
in	*aInputValue*	The input value at which to evaluate the derivative.

Returns

The value of the derivative of K_nu with respect to its order, evaluated at input_value and order aOrder.

8.3.2.2 CalculateSecondDerivativeBesselNu()

```
template<typename T >
static T exageostat::helpers::BasselFunction< T >::CalculateSecondDerivativeBesselNu (
            const T & aOrder,
            const T & aInputValue ) [static]
```

Calculates the second derivative of the modified Bessel function of the second kind (K_nu) with respect to its input, evaluated at input_value and order aOrder.

Parameters

in	*aOrder*	The order of the Bessel function.
in	*aInputValue*	The input value at which to evaluate the second derivative.

Returns

The value of the second derivative of K_nu with respect to its input, evaluated at input_value and order aOrder.

8.3.2.3 CalculateSecondDerivativeBesselNuInput()

```
template<typename T >
static T exageostat::helpers::BasselFunction< T >::CalculateSecondDerivativeBesselNuInput (
            const T & aOrder,
            const T & aInputValue ) [static]
```

Calculates the second derivative of the modified Bessel function of the second kind (K_nu) with respect to its input, evaluated at input_value and order aOrder.

Parameters

in	*aOrder*	The order of the Bessel function.
in	*aInputValue*	The input value at which to evaluate the derivative.

Returns

The value of the derivative of K_nu with respect to its input, evaluated at input_value and order aOrder.

The documentation for this class was generated from the following file:

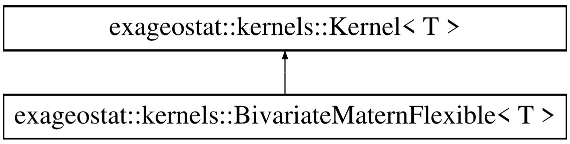
- BasselFunction.hpp

8.4 exageostat::kernels::BivariateMaternFlexible< T > Class Template Reference

A class represents a Bivariate Matern Flexible kernel.

```
#include <BivariateMaternFlexible.hpp>
```

Inheritance diagram for exageostat::kernels::BivariateMaternFlexible< T >:



Public Member Functions

- BivariateMaternFlexible ()

    Constructs a new BivariateMaternFlexible object.
- ∼BivariateMaternFlexible () override=default

    Virtual destructor to allow calls to the correct concrete destructor.
- void GenerateCovarianceMatrix (T ∗apMatrixA, const int &aRowsNumber, const int &aColumns←Number, const int &aRowOffset, const int &aColumnOffset, dataunits::Locations< T > &aLocation1, dataunits::Locations< T > &aLocation2, dataunits::Locations< T > &aLocation3, T ∗apLocalTheta, const int &aDistanceMetric) override

    Generates a covariance matrix using a set of locations and kernel parameters.

Static Public Member Functions

- static Kernel< T > ∗ Create ()

    Creates a new BivariateMaternFlexible object.

Static Private Attributes

- static bool plugin_name

Additional Inherited Members

8.4.1 Detailed Description

template<typename T>
class exageostat::kernels::BivariateMaternFlexible< T >

A class represents a Bivariate Matern Flexible kernel.

This class represents a Bivariate Matern Flexible, which is a subclass of the Kernel class. It provides a method for generating a covariance matrix using a set of input locations and kernel parameters.

8.4.2 Constructor & Destructor Documentation

8.4.2.1 BivariateMaternFlexible()

```
template<typename T >
exageostat::kernels::BivariateMaternFlexible< T >::BivariateMaternFlexible ( )
```

Constructs a new BivariateMaternFlexible object.

Initializes a new BivariateMaternFlexible object with default values.

8.4.2.2 ∼BivariateMaternFlexible()

```
template<typename T >
exageostat::kernels::BivariateMaternFlexible< T >::~BivariateMaternFlexible ( ) [override],
[default]
```

Virtual destructor to allow calls to the correct concrete destructor.

8.4.3 Member Function Documentation

8.4.3.1 GenerateCovarianceMatrix()

```
template<typename T >
void exageostat::kernels::BivariateMaternFlexible< T >::GenerateCovarianceMatrix (
            T * apMatrixA,
            const int & aRowsNumber,
            const int & aColumnsNumber,
            const int & aRowOffset,
            const int & aColumnOffset,
            dataunits::Locations< T > & aLocation1,
            dataunits::Locations< T > & aLocation2,
            dataunits::Locations< T > & aLocation3,
            T * apLocalTheta,
            const int & aDistanceMetric ) [override], [virtual]
```

Generates a covariance matrix using a set of locations and kernel parameters.

Generates a covariance matrix using a set of locations and kernel parameters.

Parameters

out	*apMatrixA*	The output covariance matrix.
in	*aRowsNumber*	The number of rows in the output matrix.
in	*aColumnsNumber*	The number of columns in the output matrix.
in	*aRowOffset*	The row offset for the input locations.
in	*aColumnOffset*	The column offset for the input locations.
in	*apLocation1*	The set of input locations 1.
in	*apLocation2*	The set of input locations 2.
in	*apLocation3*	The set of input locations 3.
in	*aLocalTheta*	An array of kernel parameters.
in	*aDistanceMetric*	Distance metric to be used (1 = Euclidean, 2 = Manhattan, 3 = Minkowski).

Returns

void

Implements exageostat::kernels::Kernel< T >.

8.4.3.2 Create()

```
template<typename T >
static Kernel<T>* exageostat::kernels::BivariateMaternFlexible< T >::Create ( )  [static]
```

Creates a new BivariateMaternFlexible object.

This method creates a new BivariateMaternFlexible object and returns a pointer to it.

Returns

A pointer to the new BivariateMaternFlexible object.

8.4.4 Member Data Documentation

8.4.4.1 plugin_name

```
template<typename T >
bool exageostat::kernels::BivariateMaternFlexible< T >::plugin_name  [static], [private]
```

The documentation for this class was generated from the following file:

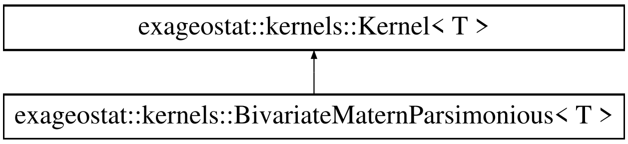
- BivariateMaternFlexible.hpp

8.5 exageostat::kernels::BivariateMaternParsimonious< T > Class Template Reference

A class represents a Bivariate Matern Parsimonious kernel.

```
#include <BivariateMaternParsimonious.hpp>
```

Inheritance diagram for exageostat::kernels::BivariateMaternParsimonious< T >:



Public Member Functions

- BivariateMaternParsimonious ()

  Constructs a new BivariateMaternParsimonious object.
- ∼BivariateMaternParsimonious () override=default

  Virtual destructor to allow calls to the correct concrete destructor.
- void GenerateCovarianceMatrix (T ∗apMatrixA, const int &aRowsNumber, const int &aColumns←Number, const int &aRowOffset, const int &aColumnOffset, dataunits::Locations< T > &aLocation1, dataunits::Locations< T > &aLocation2, dataunits::Locations< T > &aLocation3, T ∗apLocalTheta, const int &aDistanceMetric) override

  Generates a covariance matrix using a set of locations and kernel parameters.

Static Public Member Functions

- static Kernel< T > ∗ Create ()

  Creates a new BivariateMaternParsimonious object.

Static Private Attributes

- static bool plugin_name

Additional Inherited Members

8.5.1 Detailed Description

template<typename T>
class exageostat::kernels::BivariateMaternParsimonious< T >

A class represents a Bivariate Matern Parsimonious kernel.

This class represents a Bivariate Matern Parsimonious, which is a subclass of the Kernel class. It provides a method for generating a covariance matrix using a set of input locations and kernel parameters.

8.5.2 Constructor & Destructor Documentation

8.5.2.1 BivariateMaternParsimonious()

```
template<typename T >
exageostat::kernels::BivariateMaternParsimonious< T >::BivariateMaternParsimonious ( )
```

Constructs a new BivariateMaternParsimonious object.

Initializes a new BivariateMaternParsimonious object with default values.

8.5.2.2 ∼BivariateMaternParsimonious()

```
template<typename T >
exageostat::kernels::BivariateMaternParsimonious< T >::∼BivariateMaternParsimonious ( ) [override],
[default]
```

Virtual destructor to allow calls to the correct concrete destructor.

8.5.3 Member Function Documentation

8.5.3.1 GenerateCovarianceMatrix()

```
template<typename T >
void exageostat::kernels::BivariateMaternParsimonious< T >::GenerateCovarianceMatrix (
            T * apMatrixA,
            const int & aRowsNumber,
            const int & aColumnsNumber,
            const int & aRowOffset,
            const int & aColumnOffset,
            dataunits::Locations< T > & aLocation1,
            dataunits::Locations< T > & aLocation2,
            dataunits::Locations< T > & aLocation3,
            T * apLocalTheta,
            const int & aDistanceMetric ) [override], [virtual]
```

Generates a covariance matrix using a set of locations and kernel parameters.

Generates a covariance matrix using a set of locations and kernel parameters.

Parameters

out	*apMatrixA*	The output covariance matrix.
in	*aRowsNumber*	The number of rows in the output matrix.
in	*aColumnsNumber*	The number of columns in the output matrix.
in	*aRowOffset*	The row offset for the input locations.
in	*aColumnOffset*	The column offset for the input locations.
in	*apLocation1*	The set of input locations 1.
in	*apLocation2*	The set of input locations 2.
in	*apLocation3*	The set of input locations 3.

Returns

void

Implements exageostat::kernels::Kernel< T >.

8.5.3.2 Create()

```
template<typename T >
static Kernel<T>* exageostat::kernels::BivariateMaternParsimonious< T >::Create ( ) [static]
```

Creates a new BivariateMaternParsimonious object.

This method creates a new BivariateMaternParsimonious object and returns a pointer to it.

Returns

A pointer to the new BivariateMaternParsimonious object.

8.5.4 Member Data Documentation

8.5.4.1 plugin_name

```
template<typename T >
bool exageostat::kernels::BivariateMaternParsimonious< T >::plugin_name [static], [private]
```

The documentation for this class was generated from the following file:

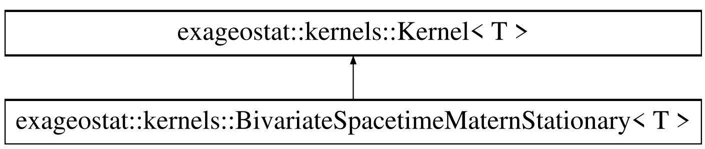
- BivariateMaternParsimonious.hpp

8.6 exageostat::kernels::BivariateSpacetimeMaternStationary< T > Class Template Reference

A class represents a Bivariate Spacetime Matern Stationary kernel.

```
#include <BivariateSpacetimeMaternStationary.hpp>
```

Inheritance diagram for exageostat::kernels::BivariateSpacetimeMaternStationary< T >:



Public Member Functions

- BivariateSpacetimeMaternStationary ()

    Constructs a new BivariateSpacetimeMaternStationary object.
- ∼BivariateSpacetimeMaternStationary () override=default

    Virtual destructor to allow calls to the correct concrete destructor.
- void GenerateCovarianceMatrix (T ∗apMatrixA, const int &aRowsNumber, const int &aColumns←Number, const int &aRowOffset, const int &aColumnOffset, dataunits::Locations< T > &aLocation1, dataunits::Locations< T > &aLocation2, dataunits::Locations< T > &aLocation3, T ∗apLocalTheta, const int &aDistanceMetric) override

    Generates a covariance matrix using a set of locations and kernel parameters.

Static Public Member Functions

- static Kernel< T > ∗ Create ()

    Creates a new BivariateSpacetimeMaternStationary object.

Static Private Attributes

- static bool plugin_name

Additional Inherited Members

8.6.1 Detailed Description

template<typename T>
class exageostat::kernels::BivariateSpacetimeMaternStationary< T >

A class represents a Bivariate Spacetime Matern Stationary kernel.

This class represents a Bivariate Spacetime Matern Stationary, which is a subclass of the Kernel class. It provides a method for generating a covariance matrix using a set of input locations and kernel parameters.

8.6.2 Constructor & Destructor Documentation

8.6.2.1 BivariateSpacetimeMaternStationary()

```
template<typename T >
exageostat::kernels::BivariateSpacetimeMaternStationary< T >::BivariateSpacetimeMaternStationary
( )
```

Constructs a new BivariateSpacetimeMaternStationary object.

Initializes a new BivariateSpacetimeMaternStationary object with default values.

8.6.2.2 ∼BivariateSpacetimeMaternStationary()

```
template<typename T >
exageostat::kernels::BivariateSpacetimeMaternStationary< T >::~BivariateSpacetimeMaternStationary
( ) [override], [default]
```

Virtual destructor to allow calls to the correct concrete destructor.

8.6.3 Member Function Documentation

8.6.3.1 GenerateCovarianceMatrix()

```
template<typename T >
void exageostat::kernels::BivariateSpacetimeMaternStationary< T >::GenerateCovarianceMatrix (
            T * apMatrixA,
            const int & aRowsNumber,
            const int & aColumnsNumber,
            const int & aRowOffset,
            const int & aColumnOffset,
            dataunits::Locations< T > & aLocation1,
            dataunits::Locations< T > & aLocation2,
            dataunits::Locations< T > & aLocation3,
            T * apLocalTheta,
            const int & aDistanceMetric ) [override], [virtual]
```

Generates a covariance matrix using a set of locations and kernel parameters.

Generates a covariance matrix using a set of locations and kernel parameters.

Parameters

out	*apMatrixA*	The output covariance matrix.
in	*aRowsNumber*	The number of rows in the output matrix.
in	*aColumnsNumber*	The number of columns in the output matrix.
in	*aRowOffset*	The row offset for the input locations.
in	*aColumnOffset*	The column offset for the input locations.
in	*apLocation1*	The set of input locations 1.
in	*apLocation2*	The set of input locations 2.
in	*apLocation3*	The set of input locations 3.
in	*aLocalTheta*	An array of kernel parameters.
in	*aDistanceMetric*	Distance metric to be used (1 = Euclidean, 2 = Manhattan, 3 = Minkowski).

Returns

void

Implements exageostat::kernels::Kernel< T >.

8.6.3.2 Create()

```
template<typename T >
static Kernel<T>* exageostat::kernels::BivariateSpacetimeMaternStationary< T >::Create ( )
[static]
```

Creates a new BivariateSpacetimeMaternStationary object.

This method creates a new BivariateSpacetimeMaternStationary object and returns a pointer to it.

Returns

A pointer to the new BivariateSpacetimeMaternStationary object.

8.6.4 Member Data Documentation

8.6.4.1 plugin_name

```
template<typename T >
bool exageostat::kernels::BivariateSpacetimeMaternStationary< T >::plugin_name [static],
[private]
```

The documentation for this class was generated from the following file:

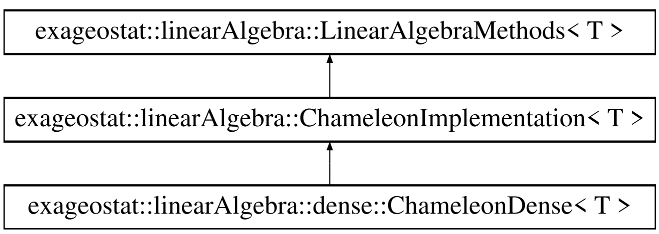
- BivariateSpacetimeMaternStationary.hpp

8.7 exageostat::linearAlgebra::dense::ChameleonDense< T > Class Template Reference

ChameleonImplementationDense is a concrete implementation for dense matrices using Chameleon..

```
#include <ChameleonDense.hpp>
```

Inheritance diagram for exageostat::linearAlgebra::dense::ChameleonDense< T >:



Public Member Functions

- ChameleonDense ()=default

    Default constructor.
- ∼ChameleonDense () override=default

    Virtual destructor to allow calls to the correct concrete destructor.
- void ExaGeoStatPotrfTile (const common::UpperLower &aUpperLower, void ∗apA, int aBand, void ∗apCD, void ∗apCrk, const int &aMaxRank, const int &aAcc) override

    Computes the Cholesky factorization of a symmetric positive definite or Symmetric positive definite matrix.

8.7.1 Detailed Description

template<typename T>
class exageostat::linearAlgebra::dense::ChameleonDense< T >

ChameleonImplementationDense is a concrete implementation for dense matrices using Chameleon..

Template Parameters

T	Data Type: float or double

8.7.2 Constructor & Destructor Documentation

8.7.2.1 ChameleonDense()

```
template<typename T >
exageostat::linearAlgebra::dense::ChameleonDense< T >::ChameleonDense ( )  [explicit], [default]
```

Default constructor.

8.7.2.2 ∼ChameleonDense()

```
template<typename T >
exageostat::linearAlgebra::dense::ChameleonDense< T >::~ChameleonDense ( )  [override], [default]
```

Virtual destructor to allow calls to the correct concrete destructor.

8.7.3 Member Function Documentation

8.7.3.1 ExaGeoStatPotrfTile()

```
template<typename T >
void exageostat::linearAlgebra::dense::ChameleonDense< T >::ExaGeoStatPotrfTile (
            const common::UpperLower & aUpperLower,
            void * apA,
            int aBand,
            void * apCD,
            void * apCrk,
            const int & aMaxRank,
            const int & aAcc )  [override], [virtual]
```

Computes the Cholesky factorization of a symmetric positive definite or Symmetric positive definite matrix.

Computes the Cholesky factorization of a symmetric positive definite or Symmetric positive definite matrix.

Parameters

in	*aUpperLower*	Whether upper or lower part of the matrix A.
in,out	*apA*	Symmetric matrix A.
in	*aBand*	Diagonal thickness parameter.
in	*apCD*	Additional matrix CD.
in	*apCrk*	Additional matrix Crk.
in	*aMaxRank*	Maximum rank parameter.
in	*aAcc*	Accuracy parameter.

Returns

void

Implements exageostat::linearAlgebra::LinearAlgebraMethods< T >.

The documentation for this class was generated from the following file:

- ChameleonDense.hpp

8.8 exageostat::dataunits::descriptor::ChameleonDescriptor< T > Class Template Reference

ChameleonDescriptor is a class for creating matrix descriptors by CHAMELEON library.

`#include <ChameleonDescriptor.hpp>`

Static Public Member Functions

- static CHAM_desc_t * CreateChameleonDescriptor (void *apDescriptor, const bool &aIsOOC, void *apMatrix, const common::FloatPoint &aFloatPoint, const int &aMB, const int &aNB, const int &aSize, const int &aLM, const int &aLN, const int &aI, const int &aJ, const int &aM, const int &aN, const int &aP, const int &aQ, const bool &aValidOOC)

    Create a chameleon descriptor for a matrix with the given parameters.
- static int DestroyChameleonDescriptor (void *apDescriptor)

    destroys and finalize a descriptor

8.8.1 Detailed Description

template<typename T>
class exageostat::dataunits::descriptor::ChameleonDescriptor< T >

ChameleonDescriptor is a class for creating matrix descriptors by CHAMELEON library.

Template Parameters

T	Data Type: float or double

8.8.2 Member Function Documentation

8.8.2.1 CreateChameleonDescriptor()

```
template<typename T >
static CHAM_desc_t* exageostat::dataunits::descriptor::ChameleonDescriptor< T >::Create↵
ChameleonDescriptor (
            void * apDescriptor,
            const bool & aIsOOC,
            void * apMatrix,
            const common::FloatPoint & aFloatPoint,
            const int & aMB,
            const int & aNB,
            const int & aSize,
            const int & aLM,
            const int & aLN,
            const int & aI,
            const int & aJ,
            const int & aM,
            const int & aN,
            const int & aP,
            const int & aQ,
            const bool & aValidOOC )  [static]
```

Create a chameleon descriptor for a matrix with the given parameters.

Parameters

in	*apDescriptor*	A pointer to the existing CHAM_desc_t descriptor. The new descriptor will be created based on this descriptor.
in	*aIsOOC*	A boolean value indicating whether the matrix is out-of-core or not.
in	*apMatrix*	A pointer to the beginning of the matrix.
in	*aFloatPoint*	The precision of the matrix.
in	*aMB*	The number of rows in a tile.
in	*aNB*	The number of columns in a tile.
in	*aSize*	The size of the matrix in elements including padding.
in	*aLM*	The number of rows of the entire matrix.
in	*aLN*	The number of columns of the entire matrix.

Parameters

in	*aI*	The row index to the beginning of the sub-matrix.
in	*aJ*	The column index to the beginning of the sub-matrix.
in	*aM*	The number of rows of the sub-matrix.
in	*aN*	The number of columns of the sub-matrix.
in	*aP*	The number of rows of the 2D distribution grid.
in	*aQ*	The number of columns of the 2D distribution grid.
in	*aValidOOC*	Boolean refer to whether this descriptor can be created with OOC technology or not.

Returns

A pointer to the newly created CHAM_desc_t descriptor.

8.8.2.2 DestroyChameleonDescriptor()

```
template<typename T >
static int exageostat::dataunits::descriptor::ChameleonDescriptor< T >::DestroyChameleon←
Descriptor (
            void * apDescriptor )  [static]
```

destroys and finalize a descriptor

Parameters

in	*apDescriptor*	A pointer to the existing CHAM_desc_t descriptor.

Returns

An error code or success code.

The documentation for this class was generated from the following file:

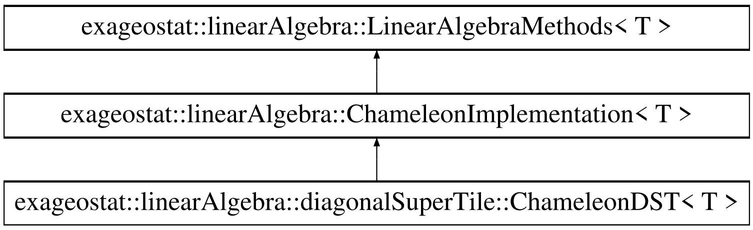
- ChameleonDescriptor.hpp

8.9 exageostat::linearAlgebra::diagonalSuperTile::ChameleonDST< T > Class Template Reference

ChameleonImplementationDST is a concrete implementation of LinearAlgebraMethods class for diagonal super tile matrices.

```
#include <ChameleonDST.hpp>
```

Inheritance diagram for exageostat::linearAlgebra::diagonalSuperTile::ChameleonDST< T >:



Public Member Functions

- ChameleonDST ()=default
  Default constructor.
- ∼ChameleonDST () override=default
  Virtual destructor to allow calls to the correct concrete destructor.
- void ExaGeoStatPotrfTile (const common::UpperLower &aUpperLower, void ∗apA, int aBand, void ∗apCD, void ∗apCrk, const int &aMaxRank, const int &aAcc) override
  Computes the Cholesky factorization of a symmetric positive definite or Symmetric positive definite matrix.
- void ExaGeoStatParallelPotrfDiagonal (const common::UpperLower &aUpperLower, void ∗apA, int aBand, void ∗apSequence, void ∗apRequest)
  Computes the parallel Cholesky factorization of a symmetric positive definite diagonal super tile matrix.
- int ExaGeoStatPotrfDiagonalTileAsync (const common::UpperLower &aUpperLower, void ∗apA, int aBand, void ∗apSequence, void ∗apRequest)
  Computes the Cholesky factorization of a symmetric positive definite diagonal super tile matrix.

8.9.1 Detailed Description

template<typename T>
class exageostat::linearAlgebra::diagonalSuperTile::ChameleonDST< T >

ChameleonImplementationDST is a concrete implementation of LinearAlgebraMethods class for diagonal super tile matrices.

Template Parameters

T	Data Type: float or double

8.9.2 Constructor & Destructor Documentation

8.9.2.1 ChameleonDST()

```
template<typename T >
exageostat::linearAlgebra::diagonalSuperTile::ChameleonDST< T >::ChameleonDST ( ) [explicit],
[default]
```

Default constructor.

8.9.2.2 ∼ChameleonDST()

```
template<typename T >
exageostat::linearAlgebra::diagonalSuperTile::ChameleonDST< T >::∼ChameleonDST ( ) [override],
[default]
```

Virtual destructor to allow calls to the correct concrete destructor.

8.9.3 Member Function Documentation

8.9.3.1 ExaGeoStatPotrfTile()

```
template<typename T >
void exageostat::linearAlgebra::diagonalSuperTile::ChameleonDST< T >::ExaGeoStatPotrfTile (
            const common::UpperLower & aUpperLower,
            void * apA,
            int aBand,
            void * apCD,
            void * apCrk,
            const int & aMaxRank,
            const int & aAcc ) [override], [virtual]
```

Computes the Cholesky factorization of a symmetric positive definite or Symmetric positive definite matrix.

Computes the Cholesky factorization of a symmetric positive definite or Symmetric positive definite matrix.

Parameters

in	*aUpperLower*	Whether upper or lower part of the matrix A.
in,out	*apA*	Symmetric matrix A.
in	*aBand*	Diagonal thickness parameter.
in	*apCD*	Additional matrix CD.
in	*apCrk*	Additional matrix Crk.
in	*aMaxRank*	Maximum rank parameter.
in	*aAcc*	Accuracy parameter.

Returns

void

Implements exageostat::linearAlgebra::LinearAlgebraMethods< T >.

8.9.3.2 ExaGeoStatParallelPotrfDiagonal()

```
template<typename T >
void exageostat::linearAlgebra::diagonalSuperTile::ChameleonDST< T >::ExaGeoStatParallel↩
PotrfDiagonal (
```

```
const common::UpperLower & aUpperLower,
void * apA,
int aBand,
void * apSequence,
void * apRequest )
```

Computes the parallel Cholesky factorization of a symmetric positive definite diagonal super tile matrix.

Parameters

in	*aUpperLower*	Whether upper or lower part of the matrix A
in	*apA*	Symmetric matrix A
in	*aBand*	diagonal thickness.
in	*apSequence*	The sequence structure to associate in the options.
in	*apRequest*	The request structure to associate in the options.

Returns

successful exit.

8.9.3.3 ExaGeoStatPotrfDiagonalTileAsync()

```
template<typename T >
int exageostat::linearAlgebra::diagonalSuperTile::ChameleonDST< T >::ExaGeoStatPotrfDiagonal↩
TileAsync (
            const common::UpperLower & aUpperLower,
            void * apA,
            int aBand,
            void * apSequence,
            void * apRequest )
```

Computes the Cholesky factorization of a symmetric positive definite diagonal super tile matrix.

Parameters

in	*aUpperLower*	Whether upper or lower part of the matrix A
in	*apA*	Symmetric matrix A
in	*aBand*	diagonal thickness.
in	*apSequence*	The sequence structure to associate in the options.
in	*apRequest*	The request structure to associate in the options.

Returns

successful exit.

The documentation for this class was generated from the following file:

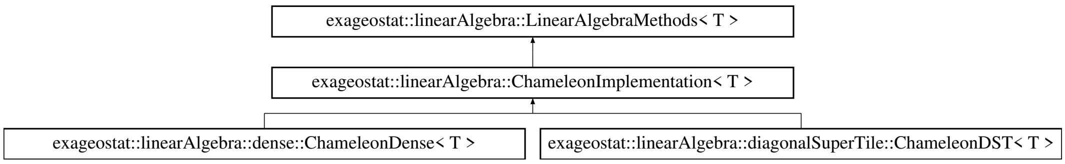
- ChameleonDST.hpp

8.10 exageostat::linearAlgebra::ChameleonImplementation< T > Class Template Reference

ChameleonImplementation is a concrete implementation of LinearAlgebraMethods class for dense or diagonal-super tile matrices.

```
#include <ChameleonImplementation.hpp>
```

Inheritance diagram for exageostat::linearAlgebra::ChameleonImplementation< T >:



Public Member Functions

- T ExaGeoStatMLETile (std::unique_ptr< ExaGeoStatData< T >> &aData, configurations::Configurations &aConfigurations, const double ∗theta, T ∗apMeasurementsMatrix, const kernels::Kernel< T > &aKernel) override

    Calculates the log likelihood value of a given value theta.
- void ExaGeoStatLapackCopyTile (const common::UpperLower &aUpperLower, void ∗apA, void ∗apB) override

    Copies a matrix in the tile layout from source to destination.
- void ExaGeoStatTrsmTile (const common::Side &aSide, const common::UpperLower &aUpperLower, const common::Trans &aTrans, const common::Diag &aDiag, const T &aAlpha, void ∗apA, void ∗apCD, void ∗ap↩Crk, void ∗apZ, const int &aMaxRank) override

    Solves one of the matrix equations op(A)∗X = alpha∗B, or X∗op(A) = alpha∗B.
- void ExaGeoStatSequenceWait (void ∗apSequence) override

    Wait for the completion of a sequence.
- void ExaGeoStatCreateSequence (void ∗apSequence) override

    Create CHAMELEON Sequence.

8.10.1 Detailed Description

template<typename T>
class exageostat::linearAlgebra::ChameleonImplementation< T >

ChameleonImplementation is a concrete implementation of LinearAlgebraMethods class for dense or diagonal-super tile matrices.

Template Parameters

T	Data Type: float or double

8.10.2 Member Function Documentation

8.10.2.1 ExaGeoStatMLETile()

```
template<typename T >
T exageostat::linearAlgebra::ChameleonImplementation< T >::ExaGeoStatMLETile (
            std::unique_ptr< ExaGeoStatData< T >> & aData,
            configurations::Configurations & aConfigurations,
            const double * theta,
            T * apMeasurementsMatrix,
            const kernels::Kernel< T > & aKernel )  [override], [virtual]
```

Calculates the log likelihood value of a given value theta.

Calculates the log likelihood value of a given value theta.

Parameters

in,out	*aData*	DescriptorData object to be populated with descriptors and data.
in	*aConfigurations*	Configurations object containing relevant settings.
in	*apTheta*	Optimization parameter used by NLOPT.
in	*apMeasurementsMatrix*	measurements matrix to be stored in DescZ.
in	*aKernel*	Reference to the kernel object to use.

Returns

log likelihood value

Implements exageostat::linearAlgebra::LinearAlgebraMethods< T >.

8.10.2.2 ExaGeoStatLapackCopyTile()

```
template<typename T >
void exageostat::linearAlgebra::ChameleonImplementation< T >::ExaGeoStatLapackCopyTile (
            const common::UpperLower & aUpperLower,
            void * apA,
            void * apB )  [override], [virtual]
```

Copies a matrix in the tile layout from source to destination.

Copies a matrix in the tile layout from source to destination.

Parameters

in	*aUpperLower*	Specifies the part of the matrix A to be copied to B.
in	*apA*	Source matrix A.
in,out	*apB*	Destination matrix B. On exit, B = A in the locations specified by Upper Lower.

Returns

void

Implements exageostat::linearAlgebra::LinearAlgebraMethods< T >.

8.10.2.3 ExaGeoStatTrsmTile()

```
template<typename T >
void exageostat::linearAlgebra::ChameleonImplementation< T >::ExaGeoStatTrsmTile (
            const common::Side & aSide,
            const common::UpperLower & aUpperLower,
            const common::Trans & aTrans,
            const common::Diag & aDiag,
            const T & aAlpha,
            void * apA,
            void * apCD,
            void * apCrk,
            void * apZ,
            const int & aMaxRank )  [override], [virtual]
```

Solves one of the matrix equations op(A)∗X = alpha∗B, or X∗op(A) = alpha∗B.

Solves one of the matrix equations op(A)∗X = alpha∗B, or X∗op(A) = alpha∗B.

Parameters

in	*aSide*	Specifies whether op(A) appears on the left or on the right of X.
in	*aUpperLower*	Specifies whether the matrix A is upper triangular or lower triangular.
in	*aTrans*	Specifies the form of op(A) to be used in the matrix multiplication.
in	*aDiag*	Specifies whether or not A is unit triangular.
in	*aAlpha*	Specifies the scalar alpha. When alpha is zero, A is not referenced and B need not be set before entry.
in	*apA*	The triangular matrix A.
in	*apCD*	Additional matrix CD.
in	*apCrk*	Additional matrix Crk.
in,out	*apZ*	The matrix B of dimension, on exit is overwritten by the solution matrix X.
in	*aMaxRank*	Maximum rank parameter.

Returns

void

Implements exageostat::linearAlgebra::LinearAlgebraMethods< T >.

8.10.2.4 ExaGeoStatSequenceWait()

```
template<typename T >
void exageostat::linearAlgebra::ChameleonImplementation< T >::ExaGeoStatSequenceWait (
            void * apSequence ) [override], [virtual]
```

Wait for the completion of a sequence.

Wait for the completion of a sequence.

Parameters

in	*apSequence*	apSequence A pointer to either CHAMELEON or HiCMA sequence.

Returns

void

Implements exageostat::linearAlgebra::LinearAlgebraMethods< T >.

8.10.2.5 ExaGeoStatCreateSequence()

```
template<typename T >
void exageostat::linearAlgebra::ChameleonImplementation< T >::ExaGeoStatCreateSequence (
            void * apSequence ) [override], [virtual]
```

Create CHAMELEON Sequence.

Create Sequence.

Parameters

out	*apSequence*	A pointer to either CHAMELEON or HiCMA sequence.

Returns

void

Implements exageostat::linearAlgebra::LinearAlgebraMethods< T >.

The documentation for this class was generated from the following file:

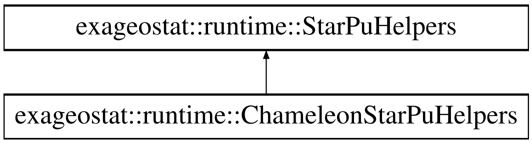
- ChameleonImplementation.hpp

8.11 exageostat::runtime::ChameleonStarPuHelpers Class Reference

ChameleonStarPuHelpers is a concrete implementation of StarPuHelpers interface for Chameleon library.

`#include <ChameleonStarPuHelpers.hpp>`

Inheritance diagram for exageostat::runtime::ChameleonStarPuHelpers:



Public Member Functions

- ChameleonStarPuHelpers ()=default

    Default constructor.
- ∼ChameleonStarPuHelpers ()=default

    Default destructor.
- void ExaGeoStatOptionsInit (void ∗apOptions, void ∗apSequence, void ∗apRequest) override

    Initialize the runtime option structure for CHAMELEON.
- void ExaGeoStatOptionsFree (void ∗apOptions) override

    Submit the release of the workspaces associated to the options structure.
- void ExaGeoStatOptionsFinalize (void ∗apOptions) override

    Finalize the runtime option structure for CHAMELEON.
- void ∗ ExaGeoStatDataGetAddr (void ∗apDescriptor, const int &aDescRow, const int &aDescCol) override

    Get the pointer to the data or the runtime handler associated to the piece of data (m, n) in desc.
- int GetMT (void ∗apDescriptor) override

    Get the number of tile rows of the sub-matrix.
- int GetM (void ∗apDescriptor) override

    Get the descriptor number of rows.
- int GetMB (void ∗apDescriptor) override

    Get the descriptor number of rows in a tile.
- void ∗ GetOptions () override

    Get the descriptor options.
- void DeleteOptions (void ∗apOptions) override

    Delete the options object.

8.11.1 Detailed Description

ChameleonStarPuHelpers is a concrete implementation of StarPuHelpers interface for Chameleon library.

8.11.2 Constructor & Destructor Documentation

8.11.2.1 ChameleonStarPuHelpers()

```
exageostat::runtime::ChameleonStarPuHelpers::ChameleonStarPuHelpers ( ) [default]
```

Default constructor.

8.11.2.2 ∼ChameleonStarPuHelpers()

```
exageostat::runtime::ChameleonStarPuHelpers::~ChameleonStarPuHelpers ( ) [default]
```

Default destructor.

8.11.3 Member Function Documentation

8.11.3.1 ExaGeoStatOptionsInit()

```
void exageostat::runtime::ChameleonStarPuHelpers::ExaGeoStatOptionsInit (
            void * apOptions,
            void * apSequence,
            void * apRequest ) [override], [virtual]
```

Initialize the runtime option structure for CHAMELEON.

Initialize the runtime option structure for either HiCMA or CHAMELEON.

Parameters

in,out	*apOptions*	The options structure that needs to be initialized.
in	*apSequence*	The sequence structure to associate in the options.
in	*apRequest*	The request structure to associate in the options.

Returns

void

Implements exageostat::runtime::StarPuHelpers.

8.11.3.2 ExaGeoStatOptionsFree()

```
void exageostat::runtime::ChameleonStarPuHelpers::ExaGeoStatOptionsFree (
            void * apOptions ) [override], [virtual]
```

Submit the release of the workspaces associated to the options structure.

Submit the release of the workspaces associated to the options structure.

Parameters

in,out	*apOptions*	The options structure for which to workspaces will be released

Returns

void

Implements exageostat::runtime::StarPuHelpers.

8.11.3.3 ExaGeoStatOptionsFinalize()

```
void exageostat::runtime::ChameleonStarPuHelpers::ExaGeoStatOptionsFinalize (
            void * apOptions )  [override], [virtual]
```

Finalize the runtime option structure for CHAMELEON.

Finalize the runtime option structure for either HiCMA or CHAMELEON.

Parameters

in,out	*apOptions*	The options structure that needs to be finalized.

Returns

void

Implements exageostat::runtime::StarPuHelpers.

8.11.3.4 ExaGeoStatDataGetAddr()

```
void* exageostat::runtime::ChameleonStarPuHelpers::ExaGeoStatDataGetAddr (
            void * apDescriptor,
            const int & aDescRow,
            const int & aDescCol )  [override], [virtual]
```

Get the pointer to the data or the runtime handler associated to the piece of data (m, n) in desc.

Get the pointer to the data or the runtime handler associated to the piece of data (m, n) in desc.

Parameters

in	*apDescriptor*	The descriptor to which belongs the piece of data
in	*aDescRow*	The row coordinate of the piece of data in the matrix
in	*aDescCol*	The column coordinate of the piece of data in the matrix

Returns

void

Implements exageostat::runtime::StarPuHelpers.

8.11.3.5 GetMT()

```
int exageostat::runtime::ChameleonStarPuHelpers::GetMT (
            void * apDescriptor )  [override], [virtual]
```

Get the number of tile rows of the sub-matrix.

Get the number of tile rows of the sub-matrix.

Parameters

in	*apDescriptor*	

Returns

int

Implements exageostat::runtime::StarPuHelpers.

8.11.3.6 GetM()

```
int exageostat::runtime::ChameleonStarPuHelpers::GetM (
            void * apDescriptor )  [override], [virtual]
```

Get the descriptor number of rows.

Get the descriptor number of rows.

Parameters

in	*apDescriptor*	

Returns

int

Implements exageostat::runtime::StarPuHelpers.

8.11.3.7 GetMB()

```
int exageostat::runtime::ChameleonStarPuHelpers::GetMB (
             void * apDescriptor )  [override], [virtual]
```

Get the descriptor number of rows in a tile.

Get the descriptor number of rows in a tile.

Parameters

in	*apDescriptor*	

Returns

int

Implements exageostat::runtime::StarPuHelpers.

8.11.3.8 GetOptions()

```
void* exageostat::runtime::ChameleonStarPuHelpers::GetOptions ( )  [override], [virtual]
```

Get the descriptor options.

Get the descriptor options.

Returns

void pointer to descriptor_option

void

Implements exageostat::runtime::StarPuHelpers.

8.11.3.9 DeleteOptions()

```
void exageostat::runtime::ChameleonStarPuHelpers::DeleteOptions (
             void * apOptions )  [override], [virtual]
```

Delete the options object.

Delete the options object.

Parameters

apOptions	

Returns

void

Implements exageostat::runtime::StarPuHelpers.

The documentation for this class was generated from the following file:

- ChameleonStarPuHelpers.hpp

8.12 exageostat::helpers::CommunicatorMPI Class Reference

A class for Communicating MPI rank.

```
#include <CommunicatorMPI.hpp>
```

Public Member Functions

- int GetRank () const

  Get the rank of the MPI process.
- void SetHardwareInitialization ()

  Set the hardware initialization flag.
- void RemoveHardwareInitialization ()

  Unset the hardware initialization flag.

Static Public Member Functions

- static CommunicatorMPI ∗ GetInstance ()

  Get a pointer to the singleton instance of the CommunicatorMPI class.

Private Member Functions

- CommunicatorMPI ()=default

  Prevent Class Instantiation for Communicator MPI Class.

Private Attributes

- bool mIsHardwareInitialized

  Used boolean to check if hardware is initialized.

Static Private Attributes

- static CommunicatorMPI ∗ mpInstance

  Pointer to the singleton instance of the CommunicatorMPI class.

8.12.1 Detailed Description

A class for Communicating MPI rank.

The CommunicatorMPI class provides functionality to communicate MPI rank information.

8.12.2 Constructor & Destructor Documentation

8.12.2.1 CommunicatorMPI()

```
exageostat::helpers::CommunicatorMPI::CommunicatorMPI ( )  [private], [default]
```

Prevent Class Instantiation for Communicator MPI Class.

8.12.3 Member Function Documentation

8.12.3.1 GetInstance()

```
static CommunicatorMPI* exageostat::helpers::CommunicatorMPI::GetInstance ( )  [static]
```

Get a pointer to the singleton instance of the CommunicatorMPI class.

Returns

A pointer to the instance of the CommunicatorMPI class.

8.12.3.2 GetRank()

```
int exageostat::helpers::CommunicatorMPI::GetRank ( ) const
```

Get the rank of the MPI process.

Returns

The rank of the MPI process.

8.12.3.3 SetHardwareInitialization()

```
void exageostat::helpers::CommunicatorMPI::SetHardwareInitialization ( )
```

Set the hardware initialization flag.

This function sets the flag to indicate that hardware has been initialized.

Returns

void

8.12.3.4 RemoveHardwareInitialization()

```
void exageostat::helpers::CommunicatorMPI::RemoveHardwareInitialization ( )
```

Unset the hardware initialization flag.

This function remove the flag to indicate that hardware has been initialized.

Returns

void

8.12.4 Member Data Documentation

8.12.4.1 mpInstance

```
CommunicatorMPI* exageostat::helpers::CommunicatorMPI::mpInstance  [static], [private]
```

Pointer to the singleton instance of the CommunicatorMPI class.

8.12.4.2 mIsHardwareInitialized

```
bool exageostat::helpers::CommunicatorMPI::mIsHardwareInitialized  [private]
```

Used boolean to check if hardware is initialized.

The documentation for this class was generated from the following file:

- CommunicatorMPI.hpp

8.13 exageostat::configurations::Configurations Class Reference

Contains methods to set and get.

```
#include <Configurations.hpp>
```

Public Member Functions

- Configurations ()

    Constructor initializing a Configuration object with default values.
- ∼Configurations ()

    destructor to allow calls to the correct concrete destructor.
- void InitializeArguments (const int &aArgC, char ∗∗apArgV, const bool &aEnableR=false)

    Initialize the module arguments.
- void InitializeAllTheta ()

    Initialize the all theta arguments.
- void InitializeDataGenerationArguments ()

    Initialize data generation arguments..
- void InitializeDataModelingArguments ()

    Initialize data Modeling arguments.
- void InitializeDataPredictionArguments ()

    Initialize data Prediction arguments.
- void SetTolerance (double aTolerance)
- void CheckKernelValue (const std::string &aKernel)

    Checks if the kernel value is valid.
- int CheckUnknownObservationsValue (const std::string &aValue)

    Checks the value of the unknown observations parameter.
- void PrintSummary (int aRank=0)

    print the summary of MLE inputs.
- int CalculateZObsNumber ()

    Calculates the number of observed measurements.
- void ParseDistanceMetric (const std::string &aDistanceMetric)

    parse user's input to distance metric.

Static Public Member Functions

- static void PrintUsage ()

    Print the usage and accepted Arguments.
- static int CheckNumericalValue (const std::string &aValue)

    Check if input value is numerical.
- static exageostat::common::Dimension CheckDimensionValue (const std::string &aDimension)

    Checks the value of the dimension parameter.
- static common::Computation CheckComputationValue (const std::string &aValue)

    Check input computation value.
- static common::Precision CheckPrecisionValue (const std::string &aValue)

    Check input precision value.
- static void InitTheta (std::vector< double > &aTheta, const int &aSize)

    Initialize a vector with a given size to contain zeros.
- static std::vector< double > ParseTheta (const std::string &aInputValues)

    Parses a string of theta values and returns an array of doubles.

Static Public Attributes

- static CREATE_SETTER_FUNCTION(ActualObservationsFilePath, const std::string &, aActual↩ObservationsFilePath, "ActualObservationsFilePath") static exageostat void SetVerbosity (const common::Verbose &aVerbose)

  Getter for the verbosity.

Static Private Member Functions

- static void ParseVerbose (const std::string &aVerbosity)

  Checks the run mode and sets the verbosity level.
- static bool IsCamelCase (const std::string &aString)

  Checks if a given string is in camel case format.

Private Attributes

- std::unordered_map< std::string, std::any > mDictionary

  Used Dictionary.
- int mArgC = 0

  Used Argument counter.
- char ** mpArgV = nullptr

  Used Argument vectors.

Static Private Attributes

- static exageostat::common::Verbose mVerbosity
- static bool mIsThetaInit
- static bool mHeapAllocated

8.13.1 Detailed Description

Contains methods to set and get.

8.13.2 Constructor & Destructor Documentation

8.13.2.1 Configurations()

```
exageostat::configurations::Configurations::Configurations ( )
```

Constructor initializing a Configuration object with default values.

8.13.2.2 ∼Configurations()

```
exageostat::configurations::Configurations::∼Configurations ( )
```

destructor to allow calls to the correct concrete destructor.

8.13.3 Member Function Documentation

8.13.3.1 InitializeArguments()

```
void exageostat::configurations::Configurations::InitializeArguments (
            const int & aArgC,
            char ** apArgV,
            const bool & aEnableR = false )
```

Initialize the module arguments.

Parameters

in	*aArgC*	The number of arguments being passed into the program from the command line.
in	*apArgV*	The array of arguments.
in	*aEnableR*	check if R is enabled

This method initializes the command line arguments and set default values for unused args.

Returns

void

8.13.3.2 InitializeAllTheta()

```
void exageostat::configurations::Configurations::InitializeAllTheta ( )
```

Initialize the all theta arguments.

Returns

void

8.13.3.3 InitializeDataGenerationArguments()

```
void exageostat::configurations::Configurations::InitializeDataGenerationArguments ( )
```

Initialize data generation arguments..

Returns

void

8.13.3.4 InitializeDataModelingArguments()

```
void exageostat::configurations::Configurations::InitializeDataModelingArguments ( )
```

Initialize data Modeling arguments.

Returns

void

8.13.3.5 InitializeDataPredictionArguments()

```
void exageostat::configurations::Configurations::InitializeDataPredictionArguments ( )
```

Initialize data Prediction arguments.

Returns

void

8.13.3.6 PrintUsage()

```
static void exageostat::configurations::Configurations::PrintUsage ( ) [static]
```

Print the usage and accepted Arguments.

Returns

void

8.13.3.7 SetTolerance()

```
void exageostat::configurations::Configurations::SetTolerance (
            double aTolerance )
```

END OF THE COMMON ARGUMENTS BETWEEN ALL MODULES. START OF THE DATA GENERATION MODULES. END OF THE DATA GENERATION MODULES. START OF THE DATA MODELING MODULES.

8.13.3.8 CheckNumericalValue()

```
static int exageostat::configurations::Configurations::CheckNumericalValue (
            const std::string & aValue )  [static]
```

Check if input value is numerical.

END OF THE DATA MODELING MODULES. START OF THE DATA PREDICTION MODULES. END OF THE DATA PREDICTION MODULES.

Parameters

in	*aValue*	The input from the user side.

Returns

The int casted value.

8.13.3.9 CheckDimensionValue()

```
static exageostat::common::Dimension exageostat::configurations::Configurations::CheckDimension↩
Value (
            const std::string & aDimension )  [static]
```

Checks the value of the dimension parameter.

Parameters

in	*aDimension*	A string represents the dimension.

Returns

The corresponding dimension value.

8.13.3.10 CheckKernelValue()

```
void exageostat::configurations::Configurations::CheckKernelValue (
            const std::string & aKernel )
```

Checks if the kernel value is valid.

Parameters

in	*aKernel*	The kernel to check.

Returns

void

8.13.3.11 CheckComputationValue()

```
static common::Computation exageostat::configurations::Configurations::CheckComputationValue (
            const std::string & aValue )  [static]
```

Check input computation value.

Parameters

in	*aValue*	The input from the user side.

Returns

Enum with the selected computation, Error if not exist.

8.13.3.12 CheckPrecisionValue()

```
static common::Precision exageostat::configurations::Configurations::CheckPrecisionValue (
            const std::string & aValue ) [static]
```

Check input precision value.

Parameters

in	*aValue*	The input from the user side.

Returns

Enum with the selected Precision, Error if not exist.

8.13.3.13 CheckUnknownObservationsValue()

```
int exageostat::configurations::Configurations::CheckUnknownObservationsValue (
            const std::string & aValue )
```

Checks the value of the unknown observations parameter.

Parameters

in	*aValue*	A string represents the number of unknown observations.

Returns

The corresponding integer value.

8.13.3.14 InitTheta()

```
static void exageostat::configurations::Configurations::InitTheta (
            std::vector< double > & aTheta,
            const int & aSize ) [static]
```

Initialize a vector with a given size to contain zeros.

Parameters

in,out	*aTheta*	A reference to the vector to initialize.
in	*aSize*	The size of the vector to initialize.

Returns

void.

8.13.3.15 PrintSummary()

```
void exageostat::configurations::Configurations::PrintSummary (
            int aRank = 0 ) [inline]
```

print the summary of MLE inputs.

Parameters

in	*aRank*	A MPI Rank variable

Returns

void

8.13.3.16 CalculateZObsNumber()

```
int exageostat::configurations::Configurations::CalculateZObsNumber ( )
```

Calculates the number of observed measurements.

Returns

number of observed measurements.

8.13.3.17 ParseTheta()

```
static std::vector<double> exageostat::configurations::Configurations::ParseTheta (
            const std::string & aInputValues ) [static]
```

Parses a string of theta values and returns an array of doubles.

Parameters

in	*aInputValues*	The input string of theta values.

Returns

A vector of parsed theta values.

8.13.3.18 ParseDistanceMetric()

```
void exageostat::configurations::Configurations::ParseDistanceMetric (
            const std::string & aDistanceMetric )
```

parse user's input to distance metric.

Parameters

in	*aDistanceMetric*	string specifying the used distance metric.

Returns

void

8.13.3.19 ParseVerbose()

```
static void exageostat::configurations::Configurations::ParseVerbose (
            const std::string & aVerbosity ) [static], [private]
```

Checks the run mode and sets the verbosity level.

Parameters

in	*aVerbosity*	A string represents the desired run mode ("verbose" or "standard").

Exceptions

std::range_error	if the input string is not "verbose" or "standard".

Returns

void

8.13.3.20 IsCamelCase()

```
static bool exageostat::configurations::Configurations::IsCamelCase (
            const std::string & aString ) [static], [private]
```

Checks if a given string is in camel case format.

Parameters

in	*aString*	The string to check.

Returns

true if the string is in camel case format, false otherwise.

8.13.4 Member Data Documentation

8.13.4.1 SetVerbosity

```
CREATE_SETTER_FUNCTION (ActualObservationsFilePath, const std::string &, aActualObservations←
FilePath, "ActualObservationsFilePath") static exageostat void exageostat::configurations::←
Configurations::SetVerbosity(const common::Verbose &aVerbose) [static]
```

Getter for the verbosity.

START OF THE COMMON ARGUMENTS BETWEEN ALL MODULES.

Returns

The verbosity mode.

8.13.4.2 mDictionary

```
std::unordered_map<std::string, std::any> exageostat::configurations::Configurations::m←
Dictionary [private]
```

Used Dictionary.

8.13.4.3 mArgC

```
int exageostat::configurations::Configurations::mArgC = 0 [private]
```

Used Argument counter.

8.13.4.4 mpArgV

```
char** exageostat::configurations::Configurations::mpArgV = nullptr  [private]
```

Used Argument vectors.

8.13.4.5 mVerbosity

```
exageostat::common::Verbose exageostat::configurations::Configurations::mVerbosity  [static],
[private]
```

8.13.4.6 mIsThetaInit

```
bool exageostat::configurations::Configurations::mIsThetaInit  [static], [private]
```

8.13.4.7 mHeapAllocated

```
bool exageostat::configurations::Configurations::mHeapAllocated  [static], [private]
```

The documentation for this class was generated from the following file:

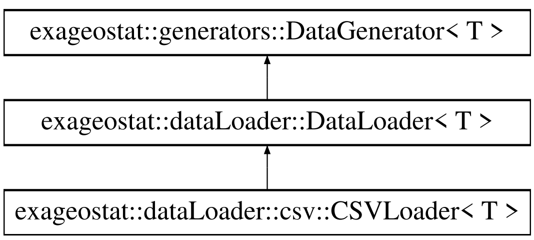
- Configurations.hpp

8.14 exageostat::dataLoader::csv::CSVLoader< T > Class Template Reference

A class for creating data by reading CSV files.

```
#include <CSVLoader.hpp>
```

Inheritance diagram for exageostat::dataLoader::csv::CSVLoader< T >:



Public Member Functions

- void ReadData (configurations::Configurations &aConfigurations, std::vector< T > &aMeasurementsMatrix, std::vector< T > &aXLocations, std::vector< T > &aYLocations, std::vector< T > &aZLocations, const int &aP) override

    Reads data from external sources into ExaGeoStat format.
- void WriteData (const T &aMatrixPointer, const int &aProblemSize, const int &aP, std::string &aLoggerPath, exageostat::dataunits::Locations< T > &aLocations) override

    Writes a matrix of vectors to disk.

Static Public Member Functions

- static CSVLoader< T > ∗ GetInstance ()

    Get a pointer to the singleton instance of the CSVLoader class.
- static void ReleaseInstance ()

    Release the singleton instance of the CSVLoader class.

Private Member Functions

- CSVLoader ()=default

    Constructor for the CSVLoader class.
- ∼CSVLoader () override=default

    Default destructor.

Static Private Attributes

- static CSVLoader< T > ∗ mpInstance

    Pointer to the singleton instance of the CSVLoader class.

Additional Inherited Members

8.14.1 Detailed Description

template<typename T>
class exageostat::dataLoader::csv::CSVLoader< T >

A class for creating data by reading CSV files.

Template Parameters

T	Data Type: float or double

8.14.2 Constructor & Destructor Documentation

8.14.2.1 CSVLoader()

```
template<typename T >
exageostat::dataLoader::csv::CSVLoader< T >::CSVLoader ( ) [private], [default]
```

Constructor for the CSVLoader class.

Returns

void

8.14.2.2 ∼CSVLoader()

```
template<typename T >
exageostat::dataLoader::csv::CSVLoader< T >::∼CSVLoader ( ) [override], [private], [default]
```

Default destructor.

8.14.3 Member Function Documentation

8.14.3.1 GetInstance()

```
template<typename T >
static CSVLoader<T>* exageostat::dataLoader::csv::CSVLoader< T >::GetInstance ( ) [static]
```

Get a pointer to the singleton instance of the CSVLoader class.

Returns

A pointer to the instance of the CSVLoader class.

8.14.3.2 ReadData()

```
template<typename T >
void exageostat::dataLoader::csv::CSVLoader< T >::ReadData (
            configurations::Configurations & aConfigurations,
            std::vector< T > & aMeasurementsMatrix,
            std::vector< T > & aXLocations,
            std::vector< T > & aYLocations,
            std::vector< T > & aZLocations,
            const int & aP ) [override], [virtual]
```

Reads data from external sources into ExaGeoStat format.

Reads data from external sources into ExaGeoStat format.

Parameters

aConfigurations	Configuration settings for data loading.
aMeasurementsMatrix	Vector to store measurement values.
aXLocations	Vector to store X coordinates of locations.
aYLocations	Vector to store Y coordinates of locations.
aZLocations	Vector to store Z coordinates of locations (if applicable).
aP	Partition index for distributed data loading.

Returns

void

Implements exageostat::dataLoader::DataLoader< T >.

8.14.3.3 WriteData()

```
template<typename T >
void exageostat::dataLoader::csv::CSVLoader< T >::WriteData (
            const T & aMatrixPointer,
            const int & aProblemSize,
            const int & aP,
            std::string & aLoggerPath,
            exageostat::dataunits::Locations< T > & aLocations ) [override], [virtual]
```

Writes a matrix of vectors to disk.

Writes a matrix of vectors to disk.

Parameters

in	*aMatrixPointer*	A Reference to the matrix data.
in	*aProblemSize*	The size of the problem.
in	*aP*	The number of processes.
in	*aLoggerPath*	The path to the logger file.
in	*aLocations*	A Reference to the Locations object.

Returns

void

Implements exageostat::dataLoader::DataLoader< T >.

8.14.3.4 ReleaseInstance()

```
template<typename T >
static void exageostat::dataLoader::csv::CSVLoader< T >::ReleaseInstance ( ) [static]
```

Release the singleton instance of the CSVLoader class.

Returns

void

8.14.4 Member Data Documentation

8.14.4.1 mpInstance

```
template<typename T >
CSVLoader<T>* exageostat::dataLoader::csv::CSVLoader< T >::mpInstance [static], [private]
```

Pointer to the singleton instance of the CSVLoader class.

The documentation for this class was generated from the following file:

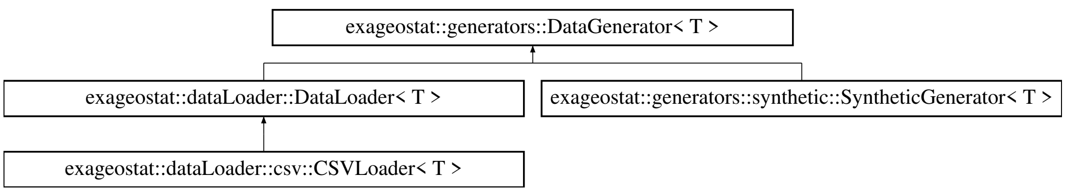
- CSVLoader.hpp

8.15 exageostat::generators::DataGenerator< T > Class Template Reference

Abstract base class for generating synthetic or real data.

```
#include <DataGenerator.hpp>
```

Inheritance diagram for exageostat::generators::DataGenerator< T >:



Public Member Functions

- virtual std::unique_ptr< ExaGeoStatData< T > > CreateData (configurations::Configurations &aConfigurations, exageostat::kernels::Kernel< T > &aKernel)=0

    Either generates synthetic data or reads data files.
- virtual ~DataGenerator ()

    Destructor for the data generator object.

Static Public Member Functions

- static std::unique_ptr< DataGenerator > CreateGenerator (configurations::Configurations &aConfigurations)

    Factory method for creating a data generator object.

Static Protected Attributes

- static common::DataSourceType aDataSourceType

    Used enum for data generators types.

8.15.1 Detailed Description

template<typename T>
class exageostat::generators::DataGenerator< T >

Abstract base class for generating synthetic or real data.

Template Parameters

T	Data Type: float or double

8.15.2 Constructor & Destructor Documentation

8.15.2.1 ~DataGenerator()

```
template<typename T >
virtual exageostat::generators::DataGenerator< T >::~DataGenerator ( )  [virtual]
```

Destructor for the data generator object.

This method frees the memory used by the data generator object.

8.15.3 Member Function Documentation

8.15.3.1 CreateData()

```
template<typename T >
virtual std::unique_ptr<ExaGeoStatData<T> > exageostat::generators::DataGenerator< T >↩
::CreateData (
            configurations::Configurations & aConfigurations,
            exageostat::kernels::Kernel< T > & aKernel )  [pure virtual]
```

Either generates synthetic data or reads data files.

This method generates the X, Y, and Z variables used to define the locations of the data points.

Parameters

in	*aConfigurations*	Reference to the data configurations.
in	*aKernel*	Reference to the used Kernel.

Returns

unique Pointer to a populated data.

Implemented in exageostat::dataLoader::DataLoader< T >, and exageostat::generators::synthetic::SyntheticGenerator< T >.

8.15.3.2 CreateGenerator()

```
template<typename T >
static std::unique_ptr<DataGenerator> exageostat::generators::DataGenerator< T >::Create↩
Generator (
            configurations::Configurations & aConfigurations )  [static]
```

Factory method for creating a data generator object.

This method creates a data generator object based on the specified configurations.

Parameters

in	*aConfigurations*	Reference to the data configurations.

Returns

A unique pointer to the created data generator object.

8.15.4 Member Data Documentation

8.15.4.1 aDataSourceType

```
template<typename T >
common::DataSourceType exageostat::generators::DataGenerator< T >::aDataSourceType  [static],
[protected]
```

Used enum for data generators types.

The documentation for this class was generated from the following file:

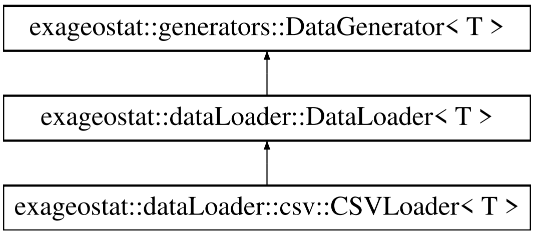
- DataGenerator.hpp

8.16 exageostat::dataLoader::DataLoader< T > Class Template Reference

Extends DataGenerator to include data loading functionalities.

```
#include <DataLoader.hpp>
```

Inheritance diagram for exageostat::dataLoader::DataLoader< T >:



Public Member Functions

- std::unique_ptr< ExaGeoStatData< T > > CreateData (configurations::Configurations &aConfigurations, kernels::Kernel< T > &aKernel) override

    Creates the data by synthetically generating it.
- virtual void ReadData (configurations::Configurations &aConfigurations, std::vector< T > &aMeasurements↩ Matrix, std::vector< T > &aXLocations, std::vector< T > &aYLocations, std::vector< T > &aZLocations, const int &aP)=0

    Reads data from external sources into ExaGeoStat format.
- virtual void WriteData (const T &aMatrixPointer, const int &aProblemSize, const int &aP, std::string &a↩ LoggerPath, exageostat::dataunits::Locations< T > &aLocations)=0

    Writes a matrix of vectors to disk.

Additional Inherited Members

8.16.1 Detailed Description

template<typename T>
class exageostat::dataLoader::DataLoader< T >

Extends DataGenerator to include data loading functionalities.

Template Parameters

T	Data Type: float or double

8.16.2 Member Function Documentation

8.16.2.1 CreateData()

```
template<typename T >
std::unique_ptr<ExaGeoStatData<T> > exageostat::dataLoader::DataLoader< T >::CreateData (
            configurations::Configurations & aConfigurations,
            kernels::Kernel< T > & aKernel )  [override], [virtual]
```

Creates the data by synthetically generating it.

Implements exageostat::generators::DataGenerator< T >.

8.16.2.2 ReadData()

```
template<typename T >
virtual void exageostat::dataLoader::DataLoader< T >::ReadData (
            configurations::Configurations & aConfigurations,
            std::vector< T > & aMeasurementsMatrix,
            std::vector< T > & aXLocations,
            std::vector< T > & aYLocations,
            std::vector< T > & aZLocations,
            const int & aP )  [pure virtual]
```

Reads data from external sources into ExaGeoStat format.

Parameters

aConfigurations	Configuration settings for data loading.
aMeasurementsMatrix	Vector to store measurement values.
aXLocations	Vector to store X coordinates of locations.
aYLocations	Vector to store Y coordinates of locations.
aZLocations	Vector to store Z coordinates of locations (if applicable).
aP	Partition index for distributed data loading.

Returns

void

Implemented in exageostat::dataLoader::csv::CSVLoader< T >.

8.16.2.3 WriteData()

```
template<typename T >
virtual void exageostat::dataLoader::DataLoader< T >::WriteData (
            const T & aMatrixPointer,
            const int & aProblemSize,
            const int & aP,
            std::string & aLoggerPath,
            exageostat::dataunits::Locations< T > & aLocations )  [pure virtual]
```

Writes a matrix of vectors to disk.

Parameters

in	*aMatrixPointer*	A Reference to the matrix data.
in	*aProblemSize*	The size of the problem.
in	*aP*	The number of processes.
in	*aLoggerPath*	The path to the logger file.
in	*aLocations*	A Reference to the Locations object.

Returns

void

Implemented in exageostat::dataLoader::csv::CSVLoader< T >.

The documentation for this class was generated from the following file:

- DataLoader.hpp

8.17 DCMG Class Reference

A class for starpu codelet dcmg.

8.17.1 Detailed Description

A class for starpu codelet dcmg.

Template Parameters

T	Data Type: float or double

This class encapsulates the struct cl_dcmg and its CPU functions.

The documentation for this class was generated from the following file:

- dcmg-codelet.hpp

8.18 exageostat::runtime::DCMGCodelet< T > Class Template Reference

```
#include <dcmg-codelet.hpp>
```

Public Member Functions

- DCMGCodelet ()=default

    Default constructor.
- ∼DCMGCodelet ()=default

    Default destructor.
- void InsertTask (void ∗apDescriptor, const int &aTriangularPart, dataunits::Locations< T > ∗apLocation1, dataunits::Locations< T > ∗apLocation2, dataunits::Locations< T > ∗apLocation3, T ∗apLocalTheta, const int &aDistanceMetric, const kernels::Kernel< T > ∗apKernel)

    Inserts a task for DCMG codelet processing.

Static Private Member Functions

- static void cl_dcmg_function (void ∗∗apBuffers, void ∗apCodeletArguments)

    CPU Function used by starpu_codelet struct.

Static Private Attributes

- static struct starpu_codelet cl_dcmg

    starpu_codelet struct

8.18.1 Constructor & Destructor Documentation

8.18.1.1 DCMGCodelet()

```
template<typename T >
exageostat::runtime::DCMGCodelet< T >::DCMGCodelet ( )  [default]
```

Default constructor.

8.18.1.2 ∼DCMGCodelet()

```
template<typename T >
exageostat::runtime::DCMGCodelet< T >::∼DCMGCodelet ( )  [default]
```

Default destructor.

8.18.2 Member Function Documentation

8.18.2.1 InsertTask()

```
template<typename T >
void exageostat::runtime::DCMGCodelet< T >::InsertTask (
            void * apDescriptor,
            const int & aTriangularPart,
            dataunits::Locations< T > * apLocation1,
            dataunits::Locations< T > * apLocation2,
            dataunits::Locations< T > * apLocation3,
            T * apLocalTheta,
            const int & aDistanceMetric,
            const kernels::Kernel< T > * apKernel )
```

Inserts a task for DCMG codelet processing.

Parameters

in,out	*apDescriptor*	A pointer to the descriptor containing task information.
in	*aTriangularPart*	An integer specifying the triangular part of the matrix (upper or lower).
in	*apLocation1*	A pointer to the first location object for the matrix elements.
in	*apLocation2*	A pointer to the second location object for the matrix elements.
in	*apLocation3*	A pointer to the third location object for the matrix elements.
in	*apLocalTheta*	A pointer to the local theta value.
in	*aDistanceMetric*	An integer specifying the distance metric to be used.
in	*apKernel*	A pointer to the kernel function to be applied during the task execution.

Returns

void

8.18.2.2 cl_dcmg_function()

```
template<typename T >
static void exageostat::runtime::DCMGCodelet< T >::cl_dcmg_function (
            void ** apBuffers,
            void * apCodeletArguments )  [static], [private]
```

CPU Function used by starpu_codelet struct.

Parameters

in	*apBuffers*	An array of pointers to the buffers containing the matrix data.
in	*apCodeletArguments*	A pointer to the codelet arguments structure

Returns

void

8.18.3 Member Data Documentation

8.18.3.1 cl_dcmg

```
template<typename T >
struct starpu_codelet exageostat::runtime::DCMGCodelet< T >::cl_dcmg [static], [private]
```

starpu_codelet struct

The documentation for this class was generated from the following file:

- dcmg-codelet.hpp

8.19 DDOTP Class Reference

A class for starpu codelet ddotp.

8.19.1 Detailed Description

A class for starpu codelet ddotp.

Template Parameters

T	Data Type: float or double

This class encapsulates the struct cl_ddotp and its CPU functions.

The documentation for this class was generated from the following file:

- ddotp-codelet.hpp

8.20 exageostat::runtime::DDOTPCodelet< T > Class Template Reference

```
#include <ddotp-codelet.hpp>
```

Public Member Functions

- DDOTPCodelet ()=default

  Default constructor.
- ∼DDOTPCodelet ()=default

  Default destructor.
- void InsertTask (void ∗apDescA, void ∗apDescProduct)

  Inserts a task for DDOTP codelet processing.

Static Private Member Functions

- static void cl_ddotp_function (void ∗∗apBuffers, void ∗apCodeletArguments)
  Executes the DDOTP codelet function for dot product calculation.

Static Private Attributes

- static struct starpu_codelet cl_ddotp
  starpu_codelet struct

8.20.1 Constructor & Destructor Documentation

8.20.1.1 DDOTPCodelet()

```
template<typename T >
exageostat::runtime::DDOTPCodelet< T >::DDOTPCodelet ( )  [default]
```

Default constructor.

8.20.1.2 ∼DDOTPCodelet()

```
template<typename T >
exageostat::runtime::DDOTPCodelet< T >::∼DDOTPCodelet ( )  [default]
```

Default destructor.

8.20.2 Member Function Documentation

8.20.2.1 InsertTask()

```
template<typename T >
void exageostat::runtime::DDOTPCodelet< T >::InsertTask (
            void ∗ apDescA,
            void ∗ apDescProduct )
```

Inserts a task for DDOTP codelet processing.

Parameters

in	*apDescA*	A pointer to the descriptor for the vector.
in,out	*apDescProduct*	A pointer to the descriptor for the dot product.

Returns

void

8.20.2.2 cl_ddotp_function()

```
template<typename T >
static void exageostat::runtime::DDOTPCodelet< T >::cl_ddotp_function (
            void ** apBuffers,
            void * apCodeletArguments )  [static], [private]
```

Executes the DDOTP codelet function for dot product calculation.

Parameters

in	*apBuffers*	An array of pointers to the buffers.
in	*apCodeletArguments*	A pointer to the codelet arguments structure, which includes the vector size (m) and the offset (m0).

Returns

void

8.20.3 Member Data Documentation

8.20.3.1 cl_ddotp

```
template<typename T >
struct starpu_codelet exageostat::runtime::DDOTPCodelet< T >::cl_ddotp  [static], [private]
```

starpu_codelet struct

The documentation for this class was generated from the following file:

- ddotp-codelet.hpp

8.21 exageostat::dataunits::DescriptorData< T > Class Template Reference

Manages geo-statistical descriptor data with functions for retrieving and manipulating descriptors.

`#include <DescriptorData.hpp>`

Public Member Functions

- DescriptorData ()=default
  Default Constructor for DescriptorData.
- ~DescriptorData ()
  Destructor for DescriptorData.
- BaseDescriptor GetDescriptor (const common::DescriptorType &aDescriptorType, const common::DescriptorName &aDescriptorName)
  Get the base descriptor.
- void ∗ GetSequence ()
  Get the sequence.
- void SetSequence (void ∗apSequence)
  Set the sequence.
- void ∗ GetRequest ()
  Get the request.
- void SetRequest (void ∗apRequest)
  Set the request.
- void SetDescriptor (const common::DescriptorType &aDescriptorType, const common::DescriptorName &a←DescriptorName, const bool &aIsOOC, void ∗apMatrix, const common::FloatPoint &aFloatPoint, const int &aMB, const int &aNB, const int &aSize, const int &aLM, const int &aLN, const int &aI, const int &aJ, const int &aM, const int &aN, const int &aP, const int &aQ, const bool &aValidOOC=true)
  Set the descriptor.
- T ∗ GetDescriptorMatrix (const common::DescriptorType &aDescriptorType, const common::DescriptorName &aDescriptorName)
  Getter for the Descriptor matrix.
- bool GetIsDescriptorInitiated ()
  Getter for the mIsDescriptorInitiated field.
- void SetIsDescriptorInitiated (bool aIsInitiated)
  Setter for mIsDescriptorInitiated field.

Private Member Functions

- std::string GetDescriptorName (const common::DescriptorName &aDescriptorName)
  Get the descriptor name.

Private Attributes

- std::unordered_map< std::string, void ∗ > mDictionary
- void ∗ mpSequence = nullptr
- void ∗ mpRequest = nullptr
- bool mIsDescriptorInitiated = false

8.21.1 Detailed Description

template<typename T>
class exageostat::dataunits::DescriptorData< T >

Manages geo-statistical descriptor data with functions for retrieving and manipulating descriptors.

@Class DescriptorData

Template Parameters

T	Data Type: float or double

8.21.2 Constructor & Destructor Documentation

8.21.2.1 DescriptorData()

```
template<typename T >
exageostat::dataunits::DescriptorData< T >::DescriptorData ( )  [explicit], [default]
```

Default Constructor for DescriptorData.

8.21.2.2 ∼DescriptorData()

```
template<typename T >
exageostat::dataunits::DescriptorData< T >::∼DescriptorData ( )
```

Destructor for DescriptorData.

8.21.3 Member Function Documentation

8.21.3.1 GetDescriptor()

```
template<typename T >
BaseDescriptor exageostat::dataunits::DescriptorData< T >::GetDescriptor (
            const common::DescriptorType & aDescriptorType,
            const common::DescriptorName & aDescriptorName )
```

Get the base descriptor.

Parameters

in	*aDescriptorType*	The type of the descriptor.
in	*aDescriptorName*	The name of the descriptor.

Returns

The base descriptor.

Exceptions

std::runtime_error	if the corresponding library is not enabled (USE_HICMA).

8.21.3.2 GetSequence()

```
template<typename T >
void* exageostat::dataunits::DescriptorData< T >::GetSequence ( )
```

Get the sequence.

Returns

Pointer to the sequence.

8.21.3.3 SetSequence()

```
template<typename T >
void exageostat::dataunits::DescriptorData< T >::SetSequence (
            void * apSequence )
```

Set the sequence.

Parameters

in	*apSequence*	Pointer to the sequence.

Returns

void

8.21.3.4 GetRequest()

```
template<typename T >
void* exageostat::dataunits::DescriptorData< T >::GetRequest ( )
```

Get the request.

Returns

Pointer to the request.

8.21.3.5 SetRequest()

```
template<typename T >
void exageostat::dataunits::DescriptorData< T >::SetRequest (
            void * apRequest )
```

Set the request.

Parameters

in	*apRequest*	Pointer to the request.

Returns

void

8.21.3.6 SetDescriptor()

```
template<typename T >
void exageostat::dataunits::DescriptorData< T >::SetDescriptor (
            const common::DescriptorType & aDescriptorType,
            const common::DescriptorName & aDescriptorName,
            const bool & aIsOOC,
            void * apMatrix,
            const common::FloatPoint & aFloatPoint,
            const int & aMB,
            const int & aNB,
            const int & aSize,
            const int & aLM,
            const int & aLN,
            const int & aI,
            const int & aJ,
            const int & aM,
            const int & aN,
            const int & aP,
            const int & aQ,
            const bool & aValidOOC = true )
```

Set the descriptor.

Parameters

in	*aDescriptorType*	The type of the descriptor.
in	*aDescriptorName*	The name of the descriptor.
in	*aIsOOC*	Boolean indicating if the descriptor is out-of-core.
in	*apMatrix*	Pointer to the matrix.
in	*aFloatPoint*	The floating-point precision.
in	*aMB*	The number of rows in a block.
in	*aNB*	The number of columns in a block.
in	*aSize*	The size of the matrix.
in	*aLM*	The leading dimension of the matrix.

Parameters

in	*aLN*	The trailing dimension of the matrix.
in	*aI*	The row index of the sub-matrix.
in	*aJ*	The column index of the sub-matrix.
in	*aM*	The number of rows in the sub-matrix.
in	*aN*	The number of columns in the sub-matrix.
in	*aP*	The number of rows in the complete matrix.
in	*aQ*	The number of columns in the complete matrix.
in	*aValidOOC*	Boolean refer to whether this descriptor can be created with OOC technology or not, default is true

Returns

void

Exceptions

std::runtime_error	if the corresponding library is not enabled (USE_HICMA).

8.21.3.7 GetDescriptorMatrix()

```
template<typename T >
T* exageostat::dataunits::DescriptorData< T >::GetDescriptorMatrix (
            const common::DescriptorType & aDescriptorType,
            const common::DescriptorName & aDescriptorName )
```

Getter for the Descriptor matrix.

Parameters

in	*aDescriptorType*	Type of the descriptor, whether it's CHAMELEON or HiCMA.
in	*aDescriptorName*	The name of the descriptor.

Returns

pointer to the Descriptor matrix.

Exceptions

std::runtime_error	if the corresponding library is not enabled (USE_HICMA).

8.21.3.8 GetIsDescriptorInitiated()

```
template<typename T >
bool exageostat::dataunits::DescriptorData< T >::GetIsDescriptorInitiated ( )
```

Getter for the mIsDescriptorInitiated field.

Returns

mIsDescriptorInitiated

8.21.3.9 SetIsDescriptorInitiated()

```
template<typename T >
void exageostat::dataunits::DescriptorData< T >::SetIsDescriptorInitiated (
            bool aIsInitiated )
```

Setter for mIsDescriptorInitiated field.

Parameters

aIsInitiated	Boolean for setting the mIsDescriptorInitiated field.

8.21.3.10 GetDescriptorName()

```
template<typename T >
std::string exageostat::dataunits::DescriptorData< T >::GetDescriptorName (
            const common::DescriptorName & aDescriptorName )  [private]
```

Get the descriptor name.

Parameters

in	*aDescriptorName*	The descriptor name.

Returns

The descriptor name as a string.

Exceptions

std::invalid_argument	if the provided descriptor name is not available.

8.21.4 Member Data Documentation

8.21.4.1 mDictionary

```
template<typename T >
std::unordered_map<std::string, void *> exageostat::dataunits::DescriptorData< T >::mDictionary
[private]
```

8.21.4.2 mpSequence

```
template<typename T >
void* exageostat::dataunits::DescriptorData< T >::mpSequence = nullptr  [private]
```

8.21.4.3 mpRequest

```
template<typename T >
void* exageostat::dataunits::DescriptorData< T >::mpRequest = nullptr  [private]
```

8.21.4.4 mIsDescriptorInitiated

```
template<typename T >
bool exageostat::dataunits::DescriptorData< T >::mIsDescriptorInitiated = false  [private]
```

The documentation for this class was generated from the following file:

- DescriptorData.hpp

8.22 exageostat::helpers::DistanceCalculationHelpers< T > Class Template Reference

Class to calculate the distance between two points.

```
#include <DistanceCalculationHelpers.hpp>
```

Static Public Member Functions

- static T CalculateDistance (exageostat::dataunits::Locations< T > &aLocations1, exageostat::dataunits::Locations< T > &aLocations2, const int &aIdxLocation1, const int &aIdxLocation2, const int &aDistanceMetric, const int &aFlagZ)

  Calculates the Euclidean distance between two points.
- static T DistanceEarth (T &aLatitude1, T &aLongitude1, T &aLatitude2, T &aLongitude2)

  Calculates the great-circle distance between two points on Earth using the Haversine formula.
- static T DegreeToRadian (T aDegree)

  Converts an angle from degrees to radians.

8.22.1 Detailed Description

template<typename T>
class exageostat::helpers::DistanceCalculationHelpers< T >

Class to calculate the distance between two points.

@Class DistanceCalculationHelpers

Template Parameters

T	Data Type: float or double.

8.22.2 Member Function Documentation

8.22.2.1 CalculateDistance()

```
template<typename T >
static T exageostat::helpers::DistanceCalculationHelpers< T >::CalculateDistance (
            exageostat::dataunits::Locations< T > & aLocations1,
            exageostat::dataunits::Locations< T > & aLocations2,
            const int & aIdxLocation1,
            const int & aIdxLocation2,
            const int & aDistanceMetric,
            const int & aFlagZ )  [static]
```

Calculates the Euclidean distance between two points.

Parameters

in	*aLocations1*	Reference to the first set of locations.
in	*aLocations2*	Reference to the second set of locations.
in	*aIdxLocation1*	Index of the first location in the first set.
in	*aIdxLocation2*	Index of the second location in the second set.
in	*aDistanceMetric*	Flag indicating the distance metric to use (1 for Manhattan distance, 2 for Euclidean distance).
in	*aFlagZ*	Flag indicating whether the points are in 2D or 3D space (0 for 2D, 1 for 3D).

Returns

The Euclidean distance between the two points.

8.22.2.2 DistanceEarth()

```
template<typename T >
static T exageostat::helpers::DistanceCalculationHelpers< T >::DistanceEarth (
            T & aLatitude1,
            T & aLongitude1,
            T & aLatitude2,
            T & aLongitude2 )  [static]
```

Calculates the great-circle distance between two points on Earth using the Haversine formula.

Parameters

in	*aLatitude1*	Latitude of the first point in degrees.
in	*aLongitude1*	Longitude of the first point in degrees.
in	*aLatitude2*	Latitude of the second point in degrees.
in	*aLongitude2*	Longitude of the second point in degrees.

Returns

The distance between the two points in kilometers.

8.22.2.3 DegreeToRadian()

```
template<typename T >
static T exageostat::helpers::DistanceCalculationHelpers< T >::DegreeToRadian (
            T aDegree )  [static]
```

Converts an angle from degrees to radians.

This function converts an angle from degrees to radians using the conversion factor /180.

Parameters

in	*aDegree*	The angle in degrees.

Returns

The angle converted to radians.

The documentation for this class was generated from the following file:

- DistanceCalculationHelpers.hpp

8.23 DMDET Class Reference

A class for starpu codelet dmdet.

8.23.1 Detailed Description

A class for starpu codelet dmdet.

Template Parameters

T	Data Type: float or double

This class encapsulates the struct cl_dmdet and its CPU functions.

The documentation for this class was generated from the following file:

- dmdet-codelet.hpp

8.24 exageostat::runtime::DMDETCodelet< T > Class Template Reference

```
#include <dmdet-codelet.hpp>
```

Public Member Functions

- DMDETCodelet ()=default

  Default constructor.
- ∼DMDETCodelet ()=default

  Default destructor.
- void InsertTask (const common::Computation &aComputation, void ∗apDescA, void ∗apDescDet, std↩ ::unique_ptr< StarPuHelpers > &aStarPuHelpers)

  Inserts a task for DMDET codelet processing.

Static Private Member Functions

- static void cl_dmdet_function (void ∗∗apBuffers, void ∗apCodeletArguments)

  Executes the DMDET codelet function for matrix determinant calculation.
- static T core_dmdet (const T ∗apDescriptor, const int &aSize)

  Calculates the determinant of a matrix.

Static Private Attributes

- static struct starpu_codelet cl_dmdet

  starpu_codelet struct

8.24.1 Constructor & Destructor Documentation

8.24.1.1 DMDETCodelet()

```
template<typename T >
exageostat::runtime::DMDETCodelet< T >::DMDETCodelet ( )  [default]
```

Default constructor.

8.24.1.2 ∼DMDETCodelet()

```
template<typename T >
exageostat::runtime::DMDETCodelet< T >::~DMDETCodelet ( )  [default]
```

Default destructor.

8.24.2 Member Function Documentation

8.24.2.1 InsertTask()

```
template<typename T >
void exageostat::runtime::DMDETCodelet< T >::InsertTask (
            const common::Computation & aComputation,
            void * apDescA,
            void * apDescDet,
            std::unique_ptr< StarPuHelpers > & aStarPuHelpers )
```

Inserts a task for DMDET codelet processing.

Parameters

in	*aComputation*	The type of computation to be performed, such as diagonal approximation or exact dense computation.
in	*apDescA*	A pointer to the descriptor for matrix A.
in,out	*apDescDet*	A pointer to the descriptor for the determinant.
in	*aStarPuHelpers*	A reference to a unique pointer of StarPuHelpers, used for accessing and managing data.

Returns

void

8.24.2.2 cl_dmdet_function()

```
template<typename T >
static void exageostat::runtime::DMDETCodelet< T >::cl_dmdet_function (
            void ** apBuffers,
            void * apCodeletArguments ) [static], [private]
```

Executes the DMDET codelet function for matrix determinant calculation.

Parameters

in	*apBuffers*	An array of pointers to the buffers containing the matrix data and the determinant.
in	*apCodeletArguments*	A pointer to the codelet arguments structure, which includes the matrix size.

Returns

void

8.24.2.3 core_dmdet()

```
template<typename T >
static T exageostat::runtime::DMDETCodelet< T >::core_dmdet (
            const T * apDescriptor,
            const int & aSize ) [static], [private]
```

Calculates the determinant of a matrix.

Parameters

in	*apDescriptor*	A pointer to the matrix data.
in	*aSize*	The size of the matrix (assumed to be square).

Returns

T The calculated determinant of the matrix.

8.24.3 Member Data Documentation

8.24.3.1 cl_dmdet

```
template<typename T >
struct starpu_codelet exageostat::runtime::DMDETCodelet< T >::cl_dmdet [static], [private]
```

starpu_codelet struct

The documentation for this class was generated from the following file:

- dmdet-codelet.hpp

8.25 Dmloe Class Reference

A class for starpu codelet dmloe-mmom.

8.25.1 Detailed Description

A class for starpu codelet dmloe-mmom.

Template Parameters

T	Data Type: float or double

This class encapsulates the struct cl_dmloe_mmom and its CPU functions.

The documentation for this class was generated from the following file:

- dmloe-mmom-codelet.hpp

8.26 exageostat::runtime::DmloeMmomCodelet< T > Class Template Reference

```
#include <dmloe-mmom-codelet.hpp>
```

Public Member Functions

- DmloeMmomCodelet ()=default

  Default constructor.
- ∼DmloeMmomCodelet ()=default

  Default destructor.
- void InsertTask (void ∗apDescExpr1, void ∗apDescExpr2, void ∗apDescExpr3, void ∗apDescMLOE, void ∗apDescMMOM)

  Inserts a task for DmloeMmom codelet processing.

Static Private Member Functions

- static void cl_dmloe_mmom_function (void ∗∗apBuffers, void ∗apCodeletArguments)

  Executes the DmloeMmom codelet function for MLOE and MMOM calculations.

Static Private Attributes

- static struct starpu_codelet cl_dmloe_mmom

  starpu_codelet struct

8.26.1 Constructor & Destructor Documentation

8.26.1.1 DmloeMmomCodelet()

```
template<typename T >
exageostat::runtime::DmloeMmomCodelet< T >::DmloeMmomCodelet ( ) [default]
```

Default constructor.

8.26.1.2 ∼DmloeMmomCodelet()

```
template<typename T >
exageostat::runtime::DmloeMmomCodelet< T >::~DmloeMmomCodelet ( ) [default]
```

Default destructor.

8.26.2 Member Function Documentation

8.26.2.1 InsertTask()

```
template<typename T >
void exageostat::runtime::DmloeMmomCodelet< T >::InsertTask (
            void * apDescExpr1,
            void * apDescExpr2,
            void * apDescExpr3,
            void * apDescMLOE,
            void * apDescMMOM )
```

Inserts a task for DmloeMmom codelet processing.

Parameters

in	*apDescExpr1*	A pointer to the descriptor for the first expression.
in	*apDescExpr2*	A pointer to the descriptor for the second expression.
in	*apDescExpr3*	A pointer to the descriptor for the third expression.
in,out	*apDescMLOE*	A pointer to the descriptor for the MLOE result.
in,out	*apDescMMOM*	A pointer to the descriptor for the MMOM result.

Returns

void

8.26.2.2 cl_dmloe_mmom_function()

```
template<typename T >
static void exageostat::runtime::DmloeMmomCodelet< T >::cl_dmloe_mmom_function (
            void ** apBuffers,
            void * apCodeletArguments )  [static], [private]
```

Executes the DmloeMmom codelet function for MLOE and MMOM calculations.

Parameters

in	*apBuffers*	An array of pointers to the buffers.
in	*apCodeletArguments*	A pointer to the codelet arguments structure, which includes the matrix dimensions and offsets.

Returns

void

8.26.3 Member Data Documentation

8.26.3.1 cl_dmloe_mmom

```
template<typename T >
struct starpu_codelet exageostat::runtime::DmloeMmomCodelet< T >::cl_dmloe_mmom  [static],
[private]
```

starpu_codelet struct

The documentation for this class was generated from the following file:

- dmloe-mmom-codelet.hpp

8.27 DMSE Class Reference

A class for starpu codelet dmse-bivariate.

8.27.1 Detailed Description

A class for starpu codelet dmse-bivariate.

A class for starpu codelet dmse.

Template Parameters

T	Data Type: float or double

This class encapsulates the struct cl_dmse_bivariate and its CPU functions.

Template Parameters

T	Data Type: float or double

This class encapsulates the struct cl_dmse and its CPU functions.

The documentation for this class was generated from the following files:

- dmse-bivariate-codelet.hpp
- dmse-codelet.hpp

8.28 exageostat::runtime::DMSEBivariateCodelet< T > Class Template Reference

`#include <dmse-bivariate-codelet.hpp>`

Public Member Functions

- DMSEBivariateCodelet ()=default

  Default constructor.
- ∼DMSEBivariateCodelet ()=default

  Default destructor.
- void InsertTask (void ∗apDescZMiss, void ∗apDescZPre, void ∗apDescsError, void ∗apDescsError1, void ∗apDescsError2)

  Inserts a task for DMSEBivariate codelet processing.

Static Private Member Functions

- static void cl_dmse_bivariate_function (void ∗∗apBuffers, void ∗apCodeletArguments)

  Executes the DMSEBivariate codelet function for bivariate error calculation.

Static Private Attributes

- static struct starpu_codelet cl_dmse_bivariate

  starpu_codelet struct

8.28.1 Constructor & Destructor Documentation

8.28.1.1 DMSEBivariateCodelet()

```
template<typename T >
exageostat::runtime::DMSEBivariateCodelet< T >::DMSEBivariateCodelet ( )  [default]
```

Default constructor.

8.28.1.2 ∼DMSEBivariateCodelet()

```
template<typename T >
exageostat::runtime::DMSEBivariateCodelet< T >::∼DMSEBivariateCodelet ( )  [default]
```

Default destructor.

8.28.2 Member Function Documentation

8.28.2.1 InsertTask()

```
template<typename T >
void exageostat::runtime::DMSEBivariateCodelet< T >::InsertTask (
            void * apDescZMiss,
            void * apDescZPre,
            void * apDescsError,
            void * apDescsError1,
            void * apDescsError2 )
```

Inserts a task for DMSEBivariate codelet processing.

Parameters

in	*apDescZMiss*	A pointer to the descriptor for the observed values.
in	*apDescZPre*	A pointer to the descriptor for the predicted values.
in,out	*apDescsError*	A pointer to the descriptor for the total error sum.
in,out	*apDescsError1*	A pointer to the descriptor for the error sum for the first variable.
in,out	*apDescsError2*	A pointer to the descriptor for the error sum for the second variable.

Returns

void

8.28.2.2 cl_dmse_bivariate_function()

```
template<typename T >
static void exageostat::runtime::DMSEBivariateCodelet< T >::cl_dmse_bivariate_function (
            void ** apBuffers,
            void * apCodeletArguments )  [static], [private]
```

Executes the DMSEBivariate codelet function for bivariate error calculation.

Parameters

in	*apBuffers*	An array of pointers to the buffers.
in	*apCodeletArguments*	A pointer to the codelet arguments structure, which includes the vector size and offset.

Returns

void

8.28.3 Member Data Documentation

8.28.3.1 cl_dmse_bivariate

```
template<typename T >
struct starpu_codelet exageostat::runtime::DMSEBivariateCodelet< T >::cl_dmse_bivariate  [static],
[private]
```

starpu_codelet struct

The documentation for this class was generated from the following file:

- dmse-bivariate-codelet.hpp

8.29 exageostat::runtime::DMSECodelet< T > Class Template Reference

```
#include <dmse-codelet.hpp>
```

Public Member Functions

- DMSECodelet ()=default

    Constructor for DMSE codelet.
- ∼DMSECodelet ()=default

    Default destructor.
- void InsertTask (void ∗apDescError, void ∗apDescZPredict, void ∗apDescZMiss)

    Inserts a task for DMSE codelet processing.

Static Private Member Functions

- static void cl_dmse_function (void ∗∗apBuffers, void ∗apCodeletArguments)

  Executes the DMSE codelet function for error calculation.

Static Private Attributes

- static struct starpu_codelet cl_dmse

  starpu_codelet struct

8.29.1 Constructor & Destructor Documentation

8.29.1.1 DMSECodelet()

```
template<typename T >
exageostat::runtime::DMSECodelet< T >::DMSECodelet ( )  [default]
```

Constructor for DMSE codelet.

8.29.1.2 ∼DMSECodelet()

```
template<typename T >
exageostat::runtime::DMSECodelet< T >::∼DMSECodelet ( )  [default]
```

Default destructor.

8.29.2 Member Function Documentation

8.29.2.1 InsertTask()

```
template<typename T >
void exageostat::runtime::DMSECodelet< T >::InsertTask (
            void * apDescError,
            void * apDescZPredict,
            void * apDescZMiss )
```

Inserts a task for DMSE codelet processing.

Parameters

in,out	*apDescError*	A pointer to the descriptor for the error sum.
in	*apDescZPredict*	A pointer to the descriptor for the predicted values.
in	*apDescZMiss*	A pointer to the descriptor for the observed values.

Returns

void

8.29.2.2 cl_dmse_function()

```
template<typename T >
static void exageostat::runtime::DMSECodelet< T >::cl_dmse_function (
            void ** apBuffers,
            void * apCodeletArguments )  [static], [private]
```

Executes the DMSE codelet function for error calculation.

Parameters

in	*apBuffers*	An array of pointers to the buffers.
in	*apCodeletArguments*	A pointer to the codelet arguments structure, which includes the vector size and offset. @retur void

8.29.3 Member Data Documentation

8.29.3.1 cl_dmse

```
template<typename T >
struct starpu_codelet exageostat::runtime::DMSECodelet< T >::cl_dmse  [static], [private]
```

starpu_codelet struct

The documentation for this class was generated from the following file:

- dmse-codelet.hpp

8.30 DTRACE Class Reference

A class for starpu codelet dtrace.

8.30.1 Detailed Description

A class for starpu codelet dtrace.

Template Parameters

T	Data Type: float or double

This class encapsulates the struct cl_dtrace and its CPU functions.

The documentation for this class was generated from the following file:

- dtrace-codelet.hpp

8.31 exageostat::runtime::DTRACECodelet< T > Class Template Reference

```
#include <dtrace-codelet.hpp>
```

Public Member Functions

- DTRACECodelet ()=default
  Default constructor.
- ∼DTRACECodelet ()=default
  Default destructor.
- void InsertTask (void ∗apDescA, void ∗apDescNum, void ∗apDescTrace)
  Inserts a task for DTRACE codelet processing.

Static Private Member Functions

- static void cl_dtrace_function (void ∗∗apBuffers, void ∗apCodeletArguments)
  Executes the DTRACE codelet function for matrix trace calculation.
- static double core_dtrace (const T ∗pDescriptor, const int &aSize, T ∗pTrace)
  Calculates the trace of a matrix.

Static Private Attributes

- static struct starpu_codelet cl_dtrace
  starpu_codelet struct

8.31.1 Constructor & Destructor Documentation

8.31.1.1 DTRACECodelet()

```
template<typename T >
exageostat::runtime::DTRACECodelet< T >::DTRACECodelet ( ) [default]
```

Default constructor.

8.31.1.2 ∼DTRACECodelet()

```
template<typename T >
exageostat::runtime::DTRACECodelet< T >::∼DTRACECodelet ( ) [default]
```

Default destructor.

8.31.2 Member Function Documentation

8.31.2.1 InsertTask()

```
template<typename T >
void exageostat::runtime::DTRACECodelet< T >::InsertTask (
            void * apDescA,
            void * apDescNum,
            void * apDescTrace )
```

Inserts a task for DTRACE codelet processing.

Parameters

in	*apDescA*	A pointer to the descriptor for the matrix.
in,out	*apDescNum*	A pointer to the descriptor for the sum.
in,out	*apDescTrace*	A pointer to the descriptor for the trace.

Returns

void

8.31.2.2 cl_dtrace_function()

```
template<typename T >
static void exageostat::runtime::DTRACECodelet< T >::cl_dtrace_function (
            void ** apBuffers,
            void * apCodeletArguments ) [static], [private]
```

Executes the DTRACE codelet function for matrix trace calculation.

Parameters

in	*apBuffers*	An array of pointers to the buffers.
in	*apCodeletArguments*	A pointer to the codelet arguments structure, which includes the matrix size.

Returns

void

8.31.2.3 core_dtrace()

```
template<typename T >
static double exageostat::runtime::DTRACECodelet< T >::core_dtrace (
            const T * pDescriptor,
            const int & aSize,
            T * pTrace )  [static], [private]
```

Calculates the trace of a matrix.

Parameters

in	*pDescriptor*	A pointer to the matrix data.
in	*aSize*	The size of the matrix (assumed to be square).
in,out	*pTrace*	A pointer to the buffer where the trace value will be stored.

Returns

The calculated trace of the matrix.

8.31.3 Member Data Documentation

8.31.3.1 cl_dtrace

```
template<typename T >
struct starpu_codelet exageostat::runtime::DTRACECodelet< T >::cl_dtrace  [static], [private]
```

starpu_codelet struct

The documentation for this class was generated from the following file:

- dtrace-codelet.hpp

8.32 DZCPY Class Reference

A class for starpu codelet dzcpy.

8.32.1 Detailed Description

A class for starpu codelet dzcpy.

Template Parameters

T	Data Type: float or double

This class encapsulates the struct cl_dzcpy and its CPU functions.

The documentation for this class was generated from the following file:

- dzcpy-codelet.hpp

8.33 exageostat::runtime::DZCPYCodelet< T > Class Template Reference

```
#include &lt;dzcpy-codelet.hpp&gt;
```

Public Member Functions

- DZCPYCodelet ()=default
  Default constructor.
- ∼DZCPYCodelet ()=default
  Default destructor.
- void InsertTask (void ∗apDescriptor, void ∗apDoubleVector)
  Inserts a task for DZCPY codelet processing.

Static Private Member Functions

- static void cl_dzcpy_function (void ∗∗apBuffers, void ∗apCodeletArguments)
  Executes the DZCPY codelet function for copying a double vector.

Static Private Attributes

- static struct starpu_codelet cl_dzcpy
  starpu_codelet struct

8.33.1 Constructor & Destructor Documentation

8.33.1.1 DZCPYCodelet()

```
template<typename T >
exageostat::runtime::DZCPYCodelet< T >::DZCPYCodelet ( ) [default]
```

Default constructor.

8.33.1.2 ~DZCPYCodelet()

```
template<typename T >
exageostat::runtime::DZCPYCodelet< T >::~DZCPYCodelet ( ) [default]
```

Default destructor.

8.33.2 Member Function Documentation

8.33.2.1 InsertTask()

```
template<typename T >
void exageostat::runtime::DZCPYCodelet< T >::InsertTask (
            void * apDescriptor,
            void * apDoubleVector )
```

Inserts a task for DZCPY codelet processing.

Parameters

in,out	*apDescriptor*	A pointer to the descriptor for the vector.
in	*apDoubleVector*	A pointer to the double vector to be copied.

Returns

void

8.33.2.2 cl_dzcpy_function()

```
template<typename T >
static void exageostat::runtime::DZCPYCodelet< T >::cl_dzcpy_function (
```

```
void ** apBuffers,
void * apCodeletArguments ) [static], [private]
```

Executes the DZCPY codelet function for copying a double vector.

Parameters

in	*apBuffers*	An array of pointers to the buffers.
in	*apCodeletArguments*	A pointer to the codelet arguments structure, which includes the vector size, offset, and the pointer to the destination vector.

Returns

void

8.33.3 Member Data Documentation

8.33.3.1 cl_dzcpy

```
template<typename T >
struct starpu_codelet exageostat::runtime::DZCPYCodelet< T >::cl_dzcpy [static], [private]
```

starpu_codelet struct

The documentation for this class was generated from the following file:

- dzcpy-codelet.hpp

8.34 exageostat::api::ExaGeoStat< T > Class Template Reference

High-Level Wrapper class containing the static API for ExaGeoStat operations.

```
#include <ExaGeoStat.hpp>
```

Static Public Member Functions

- static void ExaGeoStatLoadData (configurations::Configurations &aConfigurations, std::unique_ptr< ExaGeoStatData< T >> &aData)

  Generates Data whether it's synthetic data or real.
- static T ExaGeoStatDataModeling (configurations::Configurations &aConfigurations, std::unique_ptr< ExaGeoStatData< T >> &aData, T *apMeasurementsMatrix=nullptr)

  Models Data whether it's synthetic data or real.
- static double ExaGeoStatMLETileAPI (const std::vector< double > &aTheta, std::vector< double > &aGrad, void *apInfo)

  Objective function used in optimization, and following the NLOPT objective function format.
- static void ExaGeoStatPrediction (configurations::Configurations &aConfigurations, std::unique_ptr< ExaGeoStatData< T >> &aData, T *apMeasurementsMatrix=nullptr, dataunits::Locations< T > *apTrainLocations=nullptr, dataunits::Locations< T > *apTestLocations=nullptr)

  Predict missing measurements values.

8.34.1 Detailed Description

template<typename T>
class exageostat::api::ExaGeoStat< T >

High-Level Wrapper class containing the static API for ExaGeoStat operations.

Template Parameters

T	Data Type: float or double

8.34.2 Member Function Documentation

8.34.2.1 ExaGeoStatLoadData()

```
template<typename T >
static void exageostat::api::ExaGeoStat< T >::ExaGeoStatLoadData (
            configurations::Configurations & aConfigurations,
            std::unique_ptr< ExaGeoStatData< T >> & aData )  [static]
```

Generates Data whether it's synthetic data or real.

Parameters

in	*aConfigurations*	Reference to Configurations object containing user input data.
out	*aData*	Reference to an ExaGeoStatData<T> object where generated data will be stored.

Returns

void

8.34.2.2 ExaGeoStatDataModeling()

```
template<typename T >
static T exageostat::api::ExaGeoStat< T >::ExaGeoStatDataModeling (
            configurations::Configurations & aConfigurations,
            std::unique_ptr< ExaGeoStatData< T >> & aData,
            T * apMeasurementsMatrix = nullptr )  [static]
```

Models Data whether it's synthetic data or real.

Parameters

in	*aConfigurations*	Reference to Configurations object containing user input data.
in	*aData*	Reference to an ExaGeoStatData<T> object containing needed descriptors, and locations.
in	*apMeasurementsMatrix*	Pointer to the user input measurements matrix.

Returns

the last optimum value of MLE.

8.34.2.3 ExaGeoStatMLETileAPI()

```
template<typename T >
static double exageostat::api::ExaGeoStat< T >::ExaGeoStatMLETileAPI (
            const std::vector< double > & aTheta,
            std::vector< double > & aGrad,
            void * apInfo ) [static]
```

Objective function used in optimization, and following the NLOPT objective function format.

Parameters

in	*aTheta*	An array of length n containing the current point in the parameter space.
in	*aGrad*	An array of length n where you can optionally return the gradient of the objective function.
in	*apInfo*	pointer containing needed configurations and data.

Returns

double MLE results.

8.34.2.4 ExaGeoStatPrediction()

```
template<typename T >
static void exageostat::api::ExaGeoStat< T >::ExaGeoStatPrediction (
            configurations::Configurations & aConfigurations,
            std::unique_ptr< ExaGeoStatData< T >> & aData,
            T * apMeasurementsMatrix = nullptr,
            dataunits::Locations< T > * apTrainLocations = nullptr,
            dataunits::Locations< T > * apTestLocations = nullptr ) [static]
```

Predict missing measurements values.

Parameters

in	*aConfigurations*	Reference to Configurations object containing user input data.
in,out	*aData*	Reference to an ExaGeoStatData<T> object containing needed descriptors, and locations.
in	*apMeasurementsMatrix*	Pointer to the user input measurements matrix.
in	*apTrainLocations*	(Optional) Pointer to Locations represents training locations. these are used in training phase.
in	*apTestLocations*	(Optional) Pointer to Locations represents test locations. These are used in prediction phase.

Returns

void

The documentation for this class was generated from the following file:

- ExaGeoStat.hpp

8.35 ExaGeoStatData< T > Class Template Reference

Manages geo-statistical data with functions for location and descriptor manipulation.

`#include <ExaGeoStatData.hpp>`

Public Member Functions

- ExaGeoStatData (const int &aSize, const exageostat::common::Dimension &aDimension)
  Constructor for ExaGeoStatData.
- ExaGeoStatData (const int &aSize, const std::string &aDimension)
  Constructor for ExaGeoStatData.
- ExaGeoStatData ()=default
  Default constructor for ExaGeoStatData.
- ∼ExaGeoStatData ()
  Destructor for ExaGeoStatData.
- exageostat::dataunits::Locations< T > ∗ GetLocations ()
  Get the locations.
- void SetLocations (exageostat::dataunits::Locations< T > &aLocation)
  Set the locations.
- exageostat::dataunits::DescriptorData< T > ∗ GetDescriptorData ()
  Get the descriptor data.
- void SetMleIterations (const int &aMleIterations)
  Setter for the number of performed MLE iterations.
- int GetMleIterations ()
  Get the number of performed MLE iterations.
- void CalculateMedianLocations (const std::string &aKernelName, exageostat::dataunits::Locations< T > &aLocations)
  Calculates Median Locations.

Private Attributes

- exageostat::dataunits::DescriptorData< T > ∗ mpDescriptorData = nullptr
- exageostat::dataunits::Locations< T > ∗ mpLocations = nullptr
- int mMleIterations = 0

8.35.1 Detailed Description

template<typename T>
class ExaGeoStatData< T >

Manages geo-statistical data with functions for location and descriptor manipulation.

@Class ExaGeoStatData

Template Parameters

T	Data Type: float or double

8.35.2 Constructor & Destructor Documentation

8.35.2.1 ExaGeoStatData() [1/3]

```
template<typename T >
ExaGeoStatData< T >::ExaGeoStatData (
            const int & aSize,
            const exageostat::common::Dimension & aDimension )
```

Constructor for ExaGeoStatData.

Parameters

in	*aSize*	The size of the data.
in	*aDimension*	The dimension of the data.

8.35.2.2 ExaGeoStatData() [2/3]

```
template<typename T >
ExaGeoStatData< T >::ExaGeoStatData (
            const int & aSize,
            const std::string & aDimension )
```

Constructor for ExaGeoStatData.

Parameters

in	*aSize*	The size of the data.
in	*aDimension*	The dimension of the data.

8.35.2.3 ExaGeoStatData() [3/3]

```
template<typename T >
ExaGeoStatData< T >::ExaGeoStatData ( ) [default]
```

Default constructor for ExaGeoStatData.

8.35.2.4 ∼ExaGeoStatData()

```
template<typename T >
ExaGeoStatData< T >::~ExaGeoStatData ( )
```

Destructor for ExaGeoStatData.

8.35.3 Member Function Documentation

8.35.3.1 GetLocations()

```
template<typename T >
exageostat::dataunits::Locations<T>* ExaGeoStatData< T >::GetLocations ( )
```

Get the locations.

Returns

Pointer to the Locations object.

8.35.3.2 SetLocations()

```
template<typename T >
void ExaGeoStatData< T >::SetLocations (
            exageostat::dataunits::Locations< T > & aLocation )
```

Set the locations.

Parameters

in	*aLocation*	Pointer to the Locations object.

Returns

void

8.35.3.3 GetDescriptorData()

```
template<typename T >
exageostat::dataunits::DescriptorData<T>* ExaGeoStatData< T >::GetDescriptorData ( )
```

Get the descriptor data.

Returns

Pointer to the DescriptorData object.

8.35.3.4 SetMleIterations()

```
template<typename T >
void ExaGeoStatData< T >::SetMleIterations (
            const int & aMleIterations )
```

Setter for the number of performed MLE iterations.

Parameters

in	*aMleIterations*	number of performed MLE iterations.

Returns

void

8.35.3.5 GetMleIterations()

```
template<typename T >
int ExaGeoStatData< T >::GetMleIterations ( )
```

Get the number of performed MLE iterations.

Returns

Pointer to the DescriptorData object.

8.35.3.6 CalculateMedianLocations()

```
template<typename T >
void ExaGeoStatData< T >::CalculateMedianLocations (
            const std::string & aKernelName,
            exageostat::dataunits::Locations< T > & aLocations )
```

Calculates Median Locations.

Parameters

in	*aKernelName*	Name of the Kernel used.
out	*aLocations*	Location object to save medianLocations in.

Returns

void

8.35.4 Member Data Documentation

8.35.4.1 mpDescriptorData

```
template<typename T >
exageostat::dataunits::DescriptorData<T>* ExaGeoStatData< T >::mpDescriptorData = nullptr
[private]
```

8.35.4.2 mpLocations

```
template<typename T >
exageostat::dataunits::Locations<T>* ExaGeoStatData< T >::mpLocations = nullptr  [private]
```

8.35.4.3 mMleIterations

```
template<typename T >
int ExaGeoStatData< T >::mMleIterations = 0  [private]
```

The documentation for this class was generated from the following file:

- ExaGeoStatData.hpp

8.36 exageostat::dataunits::descriptor::ExaGeoStatDescriptor< T > Class Template Reference

ExaGeoStatDescriptor is a class for creating matrix descriptors used in CHAMELEON and HiCMA libraries.

```
#include <ExaGeoStatDescriptor.hpp>
```

Public Member Functions

- void ∗ CreateDescriptor (void ∗apDescriptor, const common::DescriptorType &aDescriptorType, const bool &aIsOOC, void ∗apMatrix, const common::FloatPoint &aFloatPoint, const int &aMB, const int &aNB, const int &aSize, const int &aLM, const int &aLN, const int &aI, const int &aJ, const int &aM, const int &aN, const int &aP, const int &aQ, const bool &aValidOOC)

    Create a descriptor for a matrix with the given parameters.
- int DestroyDescriptor (const common::DescriptorType &aDescriptorType, void ∗apDescriptor)

    destroys and finalize a descriptor

8.36.1 Detailed Description

template<typename T>
class exageostat::dataunits::descriptor::ExaGeoStatDescriptor< T >

ExaGeoStatDescriptor is a class for creating matrix descriptors used in CHAMELEON and HiCMA libraries.

Template Parameters

T	Data Type: float or double

8.36.2 Member Function Documentation

8.36.2.1 CreateDescriptor()

```
template<typename T >
void* exageostat::dataunits::descriptor::ExaGeoStatDescriptor< T >::CreateDescriptor (
            void * apDescriptor,
            const common::DescriptorType & aDescriptorType,
            const bool & aIsOOC,
            void * apMatrix,
            const common::FloatPoint & aFloatPoint,
            const int & aMB,
            const int & aNB,
            const int & aSize,
            const int & aLM,
            const int & aLN,
            const int & aI,
            const int & aJ,
            const int & aM,
            const int & aN,
            const int & aP,
            const int & aQ,
            const bool & aValidOOC )
```

Create a descriptor for a matrix with the given parameters.

Parameters

out	*apDescriptor*	A pointer to the existing to the descriptor. The new descriptor will be created based on this descriptor Type.
in	*aDescriptorType*	The type of the descriptor.
in	*aIsOOC*	A boolean value indicating whether the matrix is out-of-core or not.
in	*apMatrix*	A pointer to the beginning of the matrix.
in	*aFloatPoint*	The precision of the matrix.
in	*aMB*	The number of rows in a tile.
in	*aNB*	The number of columns in a tile.
in	*aSize*	The size of the matrix in elements including padding.
in	*aLM*	The number of rows of the entire matrix.
in	*aLN*	The number of columns of the entire matrix.
in	*aI*	The row index to the beginning of the sub-matrix.
in	*aJ*	The column index to the beginning of the sub-matrix.
in	*aM*	The number of rows of the sub-matrix.
in	*aN*	The number of columns of the sub-matrix.
in	*aP*	The number of rows of the 2D distribution grid.
in	*aQ*	The number of columns of the 2D distribution grid.
in	*aValidOOC*	Boolean refer to whether this descriptor can be created with OOC technology or not.

Returns

A pointer to the newly created descriptor.

8.36.2.2 DestroyDescriptor()

```
template<typename T >
int exageostat::dataunits::descriptor::ExaGeoStatDescriptor< T >::DestroyDescriptor (
            const common::DescriptorType & aDescriptorType,
            void * apDescriptor )
```

destroys and finalize a descriptor

Parameters

in	*aDescriptorType*	The type of the descriptor.
in	*apDescriptor*	A pointer to the existing descriptor.

Returns

An error code or success code.

The documentation for this class was generated from the following file:

- ExaGeoStatDescriptor.hpp

8.37 ExaGeoStatHardware Class Reference

Class represents the hardware configuration for the ExaGeoStat solver.

```
#include <ExaGeoStatHardware.hpp>
```

Public Member Functions

- ExaGeoStatHardware (const exageostat::common::Computation &aComputation, const int &aCoreNumber, const int &aGpuNumber, const int &aP=1, const int &aQ=1)

  Constructor for ExaGeoStatHardware.
- ExaGeoStatHardware (const std::string &aComputation, const int &aCoreNumber, const int &aGpuNumber, const int &aP=1, const int &aQ=1)

  Constructor for ExaGeoStatHardware.
- void FinalizeHardware ()

  A Finalize caller for Hardware.
- ∼ExaGeoStatHardware ()

  Destructor for ExaGeoStatHardware.

Static Public Member Functions

- static void InitHardware (const exageostat::common::Computation &aComputation, const int &aCoreNumber, const int &aGpuNumber, const int &aP, const int &aQ)

  Initializes hardware configuration.
- static void * GetChameleonContext ()

  Get the Chameleon hardware context.
- static void * GetHicmaContext ()

  Get the HiCMA hardware context.
- static void * GetContext (exageostat::common::Computation aComputation)

  Get the hardware context.
- static int GetPGrid ()

  Retrieves the P dimension of the grid.
- static int GetQGrid ()

  Retrieves the Q dimension of the grid.
- static void SetPGrid (int aP)

  Sets the P dimension of the grid.
- static void SetQGrid (int aQ)

  Sets the Q dimension of the grid.

Static Private Attributes

- static void * mpChameleonContext
- static void * mpHicmaContext
- static int mPGrid
- static int mQGrid

8.37.1 Detailed Description

Class represents the hardware configuration for the ExaGeoStat solver.

8.37.2 Constructor & Destructor Documentation

8.37.2.1 ExaGeoStatHardware() [1/2]

```
ExaGeoStatHardware::ExaGeoStatHardware (
            const exageostat::common::Computation & aComputation,
            const int & aCoreNumber,
            const int & aGpuNumber,
            const int & aP = 1,
            const int & aQ = 1 )  [explicit]
```

Constructor for ExaGeoStatHardware.

Parameters

in	*aComputation*	The computation mode for the solver.
in	*aCoreNumber*	The number of CPU cores to use for the solver.
in	*aGpuNumber*	The number of GPUs to use for the solver.
in	*aP*	The P grid dimension setting, default is 1.
in	*aQ*	The Q grid dimension setting, default is 1.

8.37.2.2 ExaGeoStatHardware() [2/2]

```
ExaGeoStatHardware::ExaGeoStatHardware (
            const std::string & aComputation,
            const int & aCoreNumber,
            const int & aGpuNumber,
            const int & aP = 1,
            const int & aQ = 1 )  [explicit]
```

Constructor for ExaGeoStatHardware.

Parameters

in	*aComputation*	The computation mode for the solver as a string.
in	*aCoreNumber*	The number of CPU cores to use for the solver.
in	*aGpuNumber*	The number of GPUs to use for the solver.
in	*aP*	The P grid dimension setting.
in	*aQ*	The Q grid dimension setting.

8.37.2.3 ∼ExaGeoStatHardware()

```
ExaGeoStatHardware::~ExaGeoStatHardware ( )
```

Destructor for ExaGeoStatHardware.

8.37.3 Member Function Documentation

8.37.3.1 FinalizeHardware()

```
void ExaGeoStatHardware::FinalizeHardware ( )
```

A Finalize caller for Hardware.

Returns

void.

8.37.3.2 InitHardware()

```
static void ExaGeoStatHardware::InitHardware (
            const exageostat::common::Computation & aComputation,
            const int & aCoreNumber,
            const int & aGpuNumber,
            const int & aP,
            const int & aQ )  [static]
```

Initializes hardware configuration.

Parameters

in	*aComputation*	The computation mode for the solver.
in	*aCoreNumber*	The number of CPU cores to use for the solver.
in	*aGpuNumber*	The number of GPUs to use for the solver.
in	*aP*	The P grid dimension setting.
in	*aQ*	The Q grid dimension setting.

Returns

void

8.37.3.3 GetChameleonContext()

```
static void* ExaGeoStatHardware::GetChameleonContext ( )  [static]
```

Get the Chameleon hardware context.

Returns

Pointer to the hardware context.

8.37.3.4 GetHicmaContext()

```
static void* ExaGeoStatHardware::GetHicmaContext ( )  [static]
```

Get the HiCMA hardware context.

Returns

Pointer to the hardware context.

8.37.3.5 GetContext()

```
static void* ExaGeoStatHardware::GetContext (
            exageostat::common::Computation aComputation )  [static]
```

Get the hardware context.

Parameters

in	*aComputation*	Used computation to decide whether to use Hicma or Chameleon context.

Returns

Pointer to the hardware context.

8.37.3.6 GetPGrid()

```
static int ExaGeoStatHardware::GetPGrid ( )  [static]
```

Retrieves the P dimension of the grid.

This function returns the current setting of the P dimension of the grid, which is part of the grid configuration used in various computational processes.

Returns

The current P dimension setting.

8.37.3.7 GetQGrid()

```
static int ExaGeoStatHardware::GetQGrid ( )  [static]
```

Retrieves the Q dimension of the grid.

This function returns the current setting of the Q dimension of the grid, which is part of the grid configuration used in various computational processes.

Returns

int The current Q dimension setting.

8.37.3.8 SetPGrid()

```
static void ExaGeoStatHardware::SetPGrid (
             int aP )  [static]
```

Sets the P dimension of the grid.

This function updates the P dimension setting of the grid. This dimension is critical in configuring the grid's layout for simulations or calculations.

Parameters

in	*aP*	The new value for the P dimension.

8.37.3.9 SetQGrid()

```
static void ExaGeoStatHardware::SetQGrid (
            int aQ )  [static]
```

Sets the Q dimension of the grid.

This function updates the Q dimension setting of the grid. This dimension is crucial in configuring the grid's layout for simulations or calculations.

Parameters

in	*aQ*	The new value for the Q dimension.

8.37.4 Member Data Documentation

8.37.4.1 mpChameleonContext

```
void* ExaGeoStatHardware::mpChameleonContext  [static], [private]
```

8.37.4.2 mpHicmaContext

```
void* ExaGeoStatHardware::mpHicmaContext  [static], [private]
```

8.37.4.3 mPGrid

```
int ExaGeoStatHardware::mPGrid  [static], [private]
```

8.37.4.4 mQGrid

`int ExaGeoStatHardware::mQGrid [static], [private]`

The documentation for this class was generated from the following file:

- ExaGeoStatHardware.hpp

8.38 Gaussian Class Reference

A class for starpu codelet gaussian-to-non.

8.38.1 Detailed Description

A class for starpu codelet gaussian-to-non.

Template Parameters

T	Data Type: float or double

This class encapsulates the struct cl_gaussian_to_non and its CPU functions.

The documentation for this class was generated from the following file:

- gaussian-to-non-codelet.hpp

8.39 exageostat::runtime::GaussianCodelet< T > Class Template Reference

`#include <gaussian-to-non-codelet.hpp>`

Public Member Functions

- GaussianCodelet ()=default
  Default constructor.
- ∼GaussianCodelet ()=default
  Default destructor.
- void InsertTask (void ∗apDesc, T ∗apTheta)
  Inserts a task for Gaussian to non-Gaussian conversion codelet processing.

Static Private Member Functions

- static void cl_gaussian_to_non_function (void ∗∗apBuffers, void ∗apCodeletArguments)
  Executes the Gaussian to non-Gaussian conversion codelet function.
- static void core_gaussian_to_non (T ∗apDescriptorZ, const T ∗apLocalTheta, const int &aSize)
  Transforms data from a Gaussian distribution to a non-Gaussian distribution.

Static Private Attributes

- static struct starpu_codelet cl_gaussian_to_non

  starpu_codelet struct

8.39.1 Constructor & Destructor Documentation

8.39.1.1 GaussianCodelet()

```
template<typename T >
exageostat::runtime::GaussianCodelet< T >::GaussianCodelet ( )  [default]
```

Default constructor.

8.39.1.2 ∼GaussianCodelet()

```
template<typename T >
exageostat::runtime::GaussianCodelet< T >::∼GaussianCodelet ( )  [default]
```

Default destructor.

8.39.2 Member Function Documentation

8.39.2.1 InsertTask()

```
template<typename T >
void exageostat::runtime::GaussianCodelet< T >::InsertTask (
            void * apDesc,
            T * apTheta )
```

Inserts a task for Gaussian to non-Gaussian conversion codelet processing.

Parameters

in,out	*apDesc*	A pointer to the descriptor for the matrix tile.
in	*apTheta*	A pointer to the transformation parameters.

Returns

void

8.39.2.2 cl_gaussian_to_non_function()

```
template<typename T >
static void exageostat::runtime::GaussianCodelet< T >::cl_gaussian_to_non_function (
            void ** apBuffers,
            void * apCodeletArguments ) [static], [private]
```

Executes the Gaussian to non-Gaussian conversion codelet function.

Parameters

in	*apBuffers*	An array of pointers to the buffers.
in	*apCodeletArguments*	A pointer to the codelet arguments structure, which includes the matrix size, offset, and the transformation parameters.

Returns

void

8.39.2.3 core_gaussian_to_non()

```
template<typename T >
static void exageostat::runtime::GaussianCodelet< T >::core_gaussian_to_non (
            T * apDescriptorZ,
            const T * apLocalTheta,
            const int & aSize ) [static], [private]
```

Transforms data from a Gaussian distribution to a non-Gaussian distribution.

Parameters

in,out	*apDescriptorZ*	A pointer to the array of data to be transformed. This array is modified in place.
in	*apLocalTheta*	A pointer to the array of transformation parameters. The first element is the mean (`xi`), the second element is the scale (`omega`), the third element is the skewness (`g`), and the fourth element is the kurtosis (`h`).
in	*aSize*	The size of the data array (`pZ`) and the transformation parameters array (`apLocalTheta`).

Exceptions

std::runtime_error	If the kurtosis parameter (`h`) is negative, indicating an invalid transformation parameter.

Returns

void

8.39.3 Member Data Documentation

8.39.3.1 cl_gaussian_to_non

```
template<typename T >
struct starpu_codelet exageostat::runtime::GaussianCodelet< T >::cl_gaussian_to_non [static],
[private]
```

starpu_codelet struct

The documentation for this class was generated from the following file:

- gaussian-to-non-codelet.hpp

8.40 exageostat::dataunits::descriptor::HicmaDescriptor< T > Class Template Reference

HicmaDescriptor is a class for creating matrix descriptors by HICMA library.

```
#include <HicmaDescriptor.hpp>
```

Static Public Member Functions

- static HICMA_desc_t * CreateHicmaDescriptor (void *apDescriptor, const bool &aIsOOC, void *apMatrix, const common::FloatPoint &aFloatPoint, const int &aMB, const int &aNB, const int &aSize, const int &aLM, const int &aLN, const int &aI, const int &aJ, const int &aM, const int &aN, const int &aP, const int &aQ, const bool &aValidOOC)

  Create a Hicma descriptor for a matrix with the given parameters.
- static int DestroyHicmaDescriptor (void *apDescriptor)

  destroys and finalize a descriptor

8.40.1 Detailed Description

template<typename T>
class exageostat::dataunits::descriptor::HicmaDescriptor< T >

HicmaDescriptor is a class for creating matrix descriptors by HICMA library.

Template Parameters

T	Data Type: float or double

8.40.2 Member Function Documentation

8.40.2.1 CreateHicmaDescriptor()

```
template<typename T >
static HICMA_desc_t* exageostat::dataunits::descriptor::HicmaDescriptor< T >::CreateHicma←-
Descriptor (
            void * apDescriptor,
            const bool & aIsOOC,
            void * apMatrix,
            const common::FloatPoint & aFloatPoint,
            const int & aMB,
            const int & aNB,
            const int & aSize,
            const int & aLM,
            const int & aLN,
            const int & aI,
            const int & aJ,
            const int & aM,
            const int & aN,
            const int & aP,
            const int & aQ,
            const bool & aValidOOC )  [static]
```

Create a Hicma descriptor for a matrix with the given parameters.

Parameters

in	*apDescriptor*	A pointer to the existing HICMA_desc_t descriptor. The new descriptor will be created based on this descriptor.
in	*aIsOOC*	A boolean value indicating whether the matrix is out-of-core or not.
in	*apMatrix*	A pointer to the beginning of the matrix.
in	*aFloatPoint*	The precision of the matrix.
in	*aMB*	The number of rows in a tile.
in	*aNB*	The number of columns in a tile.
in	*aSize*	The size of the matrix in elements including padding.
in	*aLM*	The number of rows of the entire matrix.
in	*aLN*	The number of columns of the entire matrix.
in	*aI*	The row index to the beginning of the sub-matrix.
in	*aJ*	The column index to the beginning of the sub-matrix.
in	*aM*	The number of rows of the sub-matrix.
in	*aN*	The number of columns of the sub-matrix.
in	*aP*	The number of rows of the 2D distribution grid.
in	*aQ*	The number of columns of the 2D distribution grid.
in	*aValidOOC*	Boolean refer to whether this descriptor can be created with OOC technology or not.

Returns

A pointer to the newly created HICMA_desc_t descriptor.

8.40.2.2 DestroyHicmaDescriptor()

```
template<typename T >
static int exageostat::dataunits::descriptor::HicmaDescriptor< T >::DestroyHicmaDescriptor (
            void * apDescriptor )  [static]
```

destroys and finalize a descriptor

Parameters

in	*apDescriptor*	A pointer to the existing HICMA_desc_t descriptor.

Returns

An error code or success code.

The documentation for this class was generated from the following file:

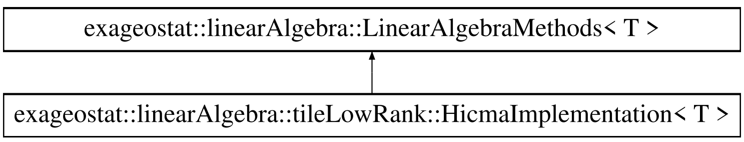
- HicmaDescriptor.hpp

8.41 exageostat::linearAlgebra::tileLowRank::HicmaImplementation< T > Class Template Reference

HicmaImplementation is a concrete implementation of LinearAlgebraMethods class for tile low-rank matrices.

```
#include <HicmaImplementation.hpp>
```

Inheritance diagram for exageostat::linearAlgebra::tileLowRank::HicmaImplementation< T >:



Public Member Functions

- HicmaImplementation ()=default

  Default constructor.
- ~HicmaImplementation () override=default

  Virtual destructor to allow calls to the correct concrete destructor.
- void SetModelingDescriptors (std::unique_ptr< ExaGeoStatData< T >> &aData, configurations::Configurations &aConfigurations, const int &aP)

  Set the modeling descriptors for HiCMA implementation.
- T ExaGeoStatMLETile (std::unique_ptr< ExaGeoStatData< T >> &aData, configurations::Configurations &aConfigurations, const double *theta, T *apMeasurementsMatrix, const kernels::Kernel< T > &aKernel) override

  Calculates the log likelihood value of a given value theta.

- void ExaGeoStatLapackCopyTile (const common::UpperLower &aUpperLower, void ∗apA, void ∗apB) override

  Copies a matrix in the tile layout from source to destination.
- void ExaGeoStatSequenceWait (void ∗apSequence) override

  Wait for the completion of a sequence.
- void ExaGeoStatCreateSequence (void ∗apSequence) override

  Create HiCMA Sequence.
- void ExaGeoStatPotrfTile (const common::UpperLower &aUpperLower, void ∗apA, int aBand, void ∗apCD, void ∗apCrk, const int &aMaxRank, const int &aAcc) override

  Computes the Cholesky factorization of a symmetric positive definite or Symmetric positive definite matrix.
- void ExaGeoStatTrsmTile (const common::Side &aSide, const common::UpperLower &aUpperLower, const common::Trans &aTrans, const common::Diag &aDiag, const T &aAlpha, void ∗apA, void ∗apCD, void ∗ap←Crk, void ∗apZ, const int &aMaxRank) override

  Solves one of the matrix equations op(A)∗X = alpha∗B, or X∗op(A) = alpha∗B.

8.41.1 Detailed Description

template<typename T>
class exageostat::linearAlgebra::tileLowRank::HicmaImplementation< T >

HicmaImplementation is a concrete implementation of LinearAlgebraMethods class for tile low-rank matrices.

Template Parameters

T	Data Type: float or double

8.41.2 Constructor & Destructor Documentation

8.41.2.1 HicmaImplementation()

```
template<typename T >
exageostat::linearAlgebra::tileLowRank::HicmaImplementation< T >::HicmaImplementation ( )
[explicit], [default]
```

Default constructor.

8.41.2.2 ∼HicmaImplementation()

```
template<typename T >
exageostat::linearAlgebra::tileLowRank::HicmaImplementation< T >::∼HicmaImplementation ( )
[override], [default]
```

Virtual destructor to allow calls to the correct concrete destructor.

8.41.3 Member Function Documentation

8.41.3.1 SetModelingDescriptors()

```
template<typename T >
void exageostat::linearAlgebra::tileLowRank::HicmaImplementation< T >::SetModelingDescriptors
(
            std::unique_ptr< ExaGeoStatData< T >> & aData,
            configurations::Configurations & aConfigurations,
            const int & aP )
```

Set the modeling descriptors for HiCMA implementation.

Parameters

in,out	*aData*	Reference to the ExaGeoStatData object.
in	*aConfigurations*	Reference to the Configurations object.
in	*aP*	the P value of the kernel multiplied by time slot.

8.41.3.2 ExaGeoStatMLETile()

```
template<typename T >
T exageostat::linearAlgebra::tileLowRank::HicmaImplementation< T >::ExaGeoStatMLETile (
            std::unique_ptr< ExaGeoStatData< T >> & aData,
            configurations::Configurations & aConfigurations,
            const double * theta,
            T * apMeasurementsMatrix,
            const kernels::Kernel< T > & aKernel )  [override], [virtual]
```

Calculates the log likelihood value of a given value theta.

Calculates the log likelihood value of a given value theta.

Parameters

in,out	*aData*	DescriptorData object to be populated with descriptors and data.
in	*aConfigurations*	Configurations object containing relevant settings.
in	*apTheta*	Optimization parameter used by NLOPT.
in	*apMeasurementsMatrix*	measurements matrix to be stored in DescZ.
in	*aKernel*	Reference to the kernel object to use.

Returns

log likelihood value

Implements exageostat::linearAlgebra::LinearAlgebraMethods< T >.

8.41.3.3 ExaGeoStatLapackCopyTile()

```
template<typename T >
void exageostat::linearAlgebra::tileLowRank::HicmaImplementation< T >::ExaGeoStatLapackCopy←
Tile (
            const common::UpperLower & aUpperLower,
            void * apA,
            void * apB ) [override], [virtual]
```

Copies a matrix in the tile layout from source to destination.

Copies a matrix in the tile layout from source to destination.

Parameters

in	*aUpperLower*	Specifies the part of the matrix A to be copied to B.
in	*apA*	Source matrix A.
in,out	*apB*	Destination matrix B. On exit, B = A in the locations specified by Upper Lower.

Returns

void

Implements exageostat::linearAlgebra::LinearAlgebraMethods< T >.

8.41.3.4 ExaGeoStatSequenceWait()

```
template<typename T >
void exageostat::linearAlgebra::tileLowRank::HicmaImplementation< T >::ExaGeoStatSequenceWait
(
            void * apSequence ) [override], [virtual]
```

Wait for the completion of a sequence.

Wait for the completion of a sequence.

Parameters

in	*apSequence*	apSequence A pointer to either CHAMELEON or HiCMA sequence.

Returns

void

Implements exageostat::linearAlgebra::LinearAlgebraMethods< T >.

8.41.3.5 ExaGeoStatCreateSequence()

```
template<typename T >
void exageostat::linearAlgebra::tileLowRank::HicmaImplementation< T >::ExaGeoStatCreate↩
Sequence (
            void * apSequence )  [override], [virtual]
```

Create HiCMA Sequence.

Create Sequence.

Parameters

out	*apSequence*	A pointer to either CHAMELEON or HiCMA sequence.

Returns

void

Implements exageostat::linearAlgebra::LinearAlgebraMethods< T >.

8.41.3.6 ExaGeoStatPotrfTile()

```
template<typename T >
void exageostat::linearAlgebra::tileLowRank::HicmaImplementation< T >::ExaGeoStatPotrfTile (
            const common::UpperLower & aUpperLower,
            void * apA,
            int aBand,
            void * apCD,
            void * apCrk,
            const int & aMaxRank,
            const int & aAcc )  [override], [virtual]
```

Computes the Cholesky factorization of a symmetric positive definite or Symmetric positive definite matrix.

Computes the Cholesky factorization of a symmetric positive definite or Symmetric positive definite matrix.

Parameters

in	*aUpperLower*	Whether upper or lower part of the matrix A.
in,out	*apA*	Symmetric matrix A.
in	*aBand*	Diagonal thickness parameter.
in	*apCD*	Additional matrix CD.
in	*apCrk*	Additional matrix Crk.
in	*aMaxRank*	Maximum rank parameter.
in	*aAcc*	Accuracy parameter.

Returns

void

Implements exageostat::linearAlgebra::LinearAlgebraMethods< T >.

8.41.3.7 ExaGeoStatTrsmTile()

```
template<typename T >
void exageostat::linearAlgebra::tileLowRank::HicmaImplementation< T >::ExaGeoStatTrsmTile (
            const common::Side & aSide,
            const common::UpperLower & aUpperLower,
            const common::Trans & aTrans,
            const common::Diag & aDiag,
            const T & aAlpha,
            void * apA,
            void * apCD,
            void * apCrk,
            void * apZ,
            const int & aMaxRank )  [override], [virtual]
```

Solves one of the matrix equations op(A)∗X = alpha∗B, or X∗op(A) = alpha∗B.

Solves one of the matrix equations op(A)∗X = alpha∗B, or X∗op(A) = alpha∗B.

Parameters

in	*aSide*	Specifies whether op(A) appears on the left or on the right of X.
in	*aUpperLower*	Specifies whether the matrix A is upper triangular or lower triangular.
in	*aTrans*	Specifies the form of op(A) to be used in the matrix multiplication.
in	*aDiag*	Specifies whether or not A is unit triangular.
in	*aAlpha*	Specifies the scalar alpha. When alpha is zero, A is not referenced and B need not be set before entry.
in	*apA*	The triangular matrix A.
in	*apCD*	Additional matrix CD.
in	*apCrk*	Additional matrix Crk.
in,out	*apZ*	The matrix B of dimension, on exit is overwritten by the solution matrix X.
in	*aMaxRank*	Maximum rank parameter.

Returns

void

Implements exageostat::linearAlgebra::LinearAlgebraMethods< T >.

The documentation for this class was generated from the following file:

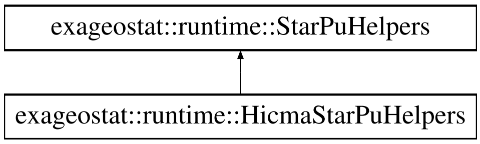
- HicmaImplementation.hpp

8.42 exageostat::runtime::HicmaStarPuHelpers Class Reference

HicmaStarPuHelpers is a concrete implementation of StarPuHelpers interface for Hicma library.

`#include <HicmaStarPuHelpers.hpp>`

Inheritance diagram for exageostat::runtime::HicmaStarPuHelpers:



Public Member Functions

- HicmaStarPuHelpers ()=default

    Default constructor.
- ∼HicmaStarPuHelpers ()=default

    Default destructor.
- void ExaGeoStatOptionsInit (void ∗apOptions, void ∗apSequence, void ∗apRequest) override

    Initialize the runtime option structure for HiCMA.
- void ExaGeoStatOptionsFree (void ∗apOptions) override

    Submit the release of the workspaces associated to the options structure.
- void ExaGeoStatOptionsFinalize (void ∗apOptions) override

    Finalize the runtime option structure for HiCMA.
- void ∗ ExaGeoStatDataGetAddr (void ∗apDescriptor, const int &aDescRow, const int &aDescCol) override

    Get the pointer to the data or the runtime handler associated to the piece of data (m, n) in desc.
- int GetMT (void ∗apDescriptor) override

    Get the number of tile rows of the sub-matrix.
- int GetM (void ∗apDescriptor) override

    Get the descriptor number of rows.
- int GetMB (void ∗apDescriptor) override

    Get the descriptor number of rows in a tile.
- void ∗ GetOptions () override

    Get the descriptor options.
- void DeleteOptions (void ∗apOptions) override

    Delete the options object.

8.42.1 Detailed Description

HicmaStarPuHelpers is a concrete implementation of StarPuHelpers interface for Hicma library.

8.42.2 Constructor & Destructor Documentation

8.42.2.1 HicmaStarPuHelpers()

```
exageostat::runtime::HicmaStarPuHelpers::HicmaStarPuHelpers ( ) [default]
```

Default constructor.

8.42.2.2 ∼HicmaStarPuHelpers()

```
exageostat::runtime::HicmaStarPuHelpers::~HicmaStarPuHelpers ( ) [default]
```

Default destructor.

8.42.3 Member Function Documentation

8.42.3.1 ExaGeoStatOptionsInit()

```
void exageostat::runtime::HicmaStarPuHelpers::ExaGeoStatOptionsInit (
            void * apOptions,
            void * apSequence,
            void * apRequest ) [override], [virtual]
```

Initialize the runtime option structure for HiCMA.

Initialize the runtime option structure for either HiCMA or CHAMELEON.

Parameters

in,out	*apOptions*	The options structure that needs to be initialized.
in	*apSequence*	The sequence structure to associate in the options.
in	*apRequest*	The request structure to associate in the options.

Returns

void

Implements exageostat::runtime::StarPuHelpers.

8.42.3.2 ExaGeoStatOptionsFree()

```
void exageostat::runtime::HicmaStarPuHelpers::ExaGeoStatOptionsFree (
            void * apOptions ) [override], [virtual]
```

Submit the release of the workspaces associated to the options structure.

Submit the release of the workspaces associated to the options structure.

Parameters

in,out	*apOptions*	The options structure for which to workspaces will be released

Returns

void

Implements exageostat::runtime::StarPuHelpers.

8.42.3.3 ExaGeoStatOptionsFinalize()

```
void exageostat::runtime::HicmaStarPuHelpers::ExaGeoStatOptionsFinalize (
            void * apOptions )  [override], [virtual]
```

Finalize the runtime option structure for HiCMA.

Finalize the runtime option structure for either HiCMA or CHAMELEON.

Parameters

in,out	*apOptions*	The options structure that needs to be finalized.

Returns

void

Implements exageostat::runtime::StarPuHelpers.

8.42.3.4 ExaGeoStatDataGetAddr()

```
void* exageostat::runtime::HicmaStarPuHelpers::ExaGeoStatDataGetAddr (
            void * apDescriptor,
            const int & aDescRow,
            const int & aDescCol )  [override], [virtual]
```

Get the pointer to the data or the runtime handler associated to the piece of data (m, n) in desc.

Get the pointer to the data or the runtime handler associated to the piece of data (m, n) in desc.

Parameters

in	*apDescriptor*	The descriptor to which belongs the piece of data
in	*aDescRow*	The row coordinate of the piece of data in the matrix
in	*aDescCol*	The column coordinate of the piece of data in the matrix

Returns

void

Implements exageostat::runtime::StarPuHelpers.

8.42.3.5 GetMT()

```
int exageostat::runtime::HicmaStarPuHelpers::GetMT (
            void * apDescriptor )  [override], [virtual]
```

Get the number of tile rows of the sub-matrix.

Get the number of tile rows of the sub-matrix.

Parameters

in	*apDescriptor*	

Returns

int

Implements exageostat::runtime::StarPuHelpers.

8.42.3.6 GetM()

```
int exageostat::runtime::HicmaStarPuHelpers::GetM (
            void * apDescriptor )  [override], [virtual]
```

Get the descriptor number of rows.

Get the descriptor number of rows.

Parameters

in	*apDescriptor*	

Returns

int

Implements exageostat::runtime::StarPuHelpers.

8.42.3.7 GetMB()

```
int exageostat::runtime::HicmaStarPuHelpers::GetMB (
            void * apDescriptor ) [override], [virtual]
```

Get the descriptor number of rows in a tile.

Get the descriptor number of rows in a tile.

Parameters

in	*apDescriptor*	

Returns

int

Implements exageostat::runtime::StarPuHelpers.

8.42.3.8 GetOptions()

```
void* exageostat::runtime::HicmaStarPuHelpers::GetOptions ( ) [override], [virtual]
```

Get the descriptor options.

Implements exageostat::runtime::StarPuHelpers.

8.42.3.9 DeleteOptions()

```
void exageostat::runtime::HicmaStarPuHelpers::DeleteOptions (
            void * apOptions ) [override], [virtual]
```

Delete the options object.

Delete the options object.

Parameters

apOptions	

Returns

void

Implements exageostat::runtime::StarPuHelpers.

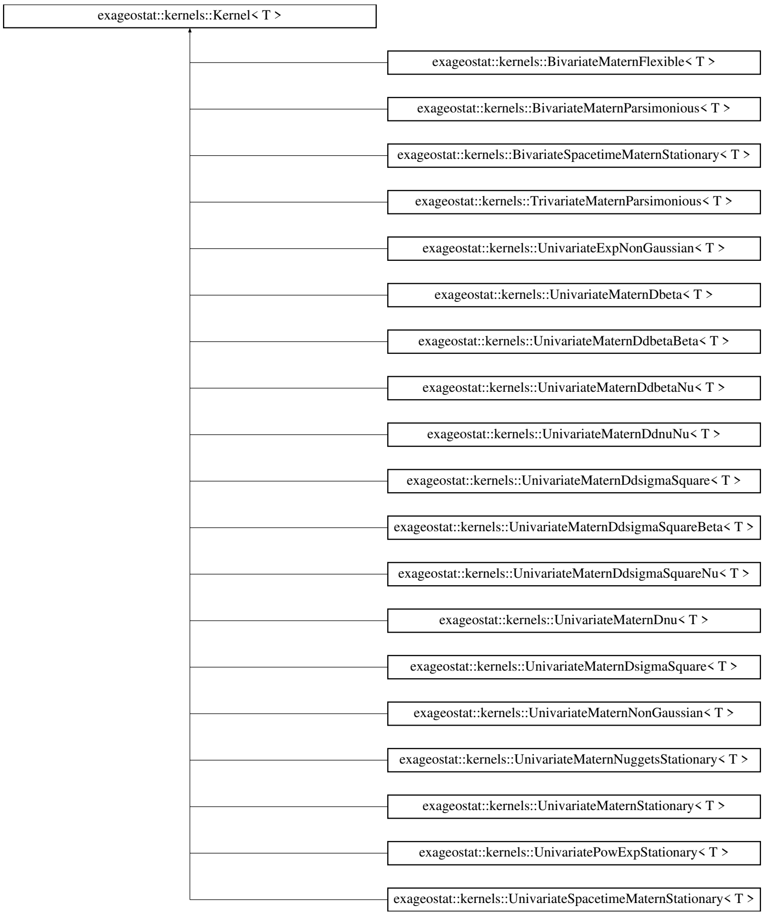
The documentation for this class was generated from the following file:

- HicmaStarPuHelpers.hpp

8.43 exageostat::kernels::Kernel< T > Class Template Reference

```
#include <Kernel.hpp>
```

Inheritance diagram for exageostat::kernels::Kernel< T >:



Public Member Functions

- virtual ∼Kernel ()=default
- virtual void GenerateCovarianceMatrix (T ∗apMatrixA, const int &aRowsNumber, const int &aColumnsNumber, const int &aRowOffset, const int &aColumnOffset, dataunits::Locations< T > &aLocation1, dataunits::Locations< T > &aLocation2, dataunits::Locations< T > &aLocation3, T ∗apLocalTheta, const int &aDistanceMetric)=0

    Generates a covariance matrix using a set of locations and kernel parameters.
- int GetVariablesNumber () const

    Returns the value of the parameter P used by the kernel function.
- void SetPValue (int aTimeSlot)

    Sets the value of the parameter P used by the kernel function.
- int GetParametersNumbers () const

    Returns the number of the parameters used by the kernel function.

Protected Attributes

- int mP = 1
- int mVariablesNumber = 1
- int mParametersNumber = 3

8.43.1 Constructor & Destructor Documentation

8.43.1.1 ∼Kernel()

```
template<typename T >
virtual exageostat::kernels::Kernel< T >::∼Kernel ( )  [virtual], [default]
```

Default virtual destructor to be overridden by the the suitable concrete kernel destructor.

8.43.2 Member Function Documentation

8.43.2.1 GenerateCovarianceMatrix()

```
template<typename T >
virtual void exageostat::kernels::Kernel< T >::GenerateCovarianceMatrix (
            T * apMatrixA,
            const int & aRowsNumber,
            const int & aColumnsNumber,
            const int & aRowOffset,
            const int & aColumnOffset,
            dataunits::Locations< T > & aLocation1,
            dataunits::Locations< T > & aLocation2,
            dataunits::Locations< T > & aLocation3,
            T * apLocalTheta,
            const int & aDistanceMetric )  [pure virtual]
```

Generates a covariance matrix using a set of locations and kernel parameters.

Parameters

out	*apMatrixA*	The output covariance matrix.
in	*aRowsNumber*	The number of rows in the output matrix.
in	*aColumnsNumber*	The number of columns in the output matrix.
in	*aRowOffset*	The row offset for the input locations.
in	*aColumnOffset*	The column offset for the input locations.
in	*apLocation1*	The set of input locations 1.
in	*apLocation2*	The set of input locations 2.
in	*apLocation3*	The set of input locations 3.
in	*aLocalTheta*	An array of kernel parameters.
in	*aDistanceMetric*	Distance metric to be used (1 = Euclidean, 2 = Manhattan, 3 = Minkowski).

Returns

void

Implemented in exageostat::kernels::UnivariateSpacetimeMaternStationary< T >, exageostat::kernels::UnivariatePowExpStationary< exageostat::kernels::UnivariateMaternStationary< T >, exageostat::kernels::UnivariateMaternNuggetsStationary< T >, exageostat::kernels::UnivariateMaternNonGaussian< T >, exageostat::kernels::UnivariateMaternDsigmaSquare< T >, exageostat::kernels::UnivariateMaternDnu< T >, exageostat::kernels::UnivariateMaternDdsigmaSquareNu< T >, exageostat::kernels::UnivariateMaternDdsigmaSquareBeta< T >, exageostat::kernels::UnivariateMaternDdsigmaSquare< T >, exageostat::kernels::UnivariateMaternDdnuNu< T >, exageostat::kernels::UnivariateMaternDdbetaNu< T >, exageostat::kernels::UnivariateMaternDdbetaBeta< T >, exageostat::kernels::UnivariateMaternDbeta< T >, exageostat::kernels::UnivariateExpNonGaussian< T >, exageostat::kernels::TrivariateMaternParsimonious< T >, exageostat::kernels::BivariateSpacetimeMaternStationary< T >, exageostat::kernels::BivariateMaternParsimonious< T >, and exageostat::kernels::BivariateMaternFlexible< T >.

8.43.2.2 GetVariablesNumber()

```
template<typename T >
int exageostat::kernels::Kernel< T >::GetVariablesNumber ( ) const
```

Returns the value of the parameter P used by the kernel function.

Returns

The value of P (Variables Number).

8.43.2.3 SetPValue()

```
template<typename T >
void exageostat::kernels::Kernel< T >::SetPValue (
            int aTimeSlot )
```

Sets the value of the parameter P used by the kernel function.

Parameters

in	*aTimeSlot*	Value to set `mP` with.

Returns

void

8.43.2.4 GetParametersNumbers()

```
template<typename T >
int exageostat::kernels::Kernel< T >::GetParametersNumbers ( ) const
```

Returns the number of the parameters used by the kernel function.

Returns

The value of ParametersNumber.

8.43.3 Member Data Documentation

8.43.3.1 mP

```
template<typename T >
int exageostat::kernels::Kernel< T >::mP = 1 [protected]
```

8.43.3.2 mVariablesNumber

```
template<typename T >
int exageostat::kernels::Kernel< T >::mVariablesNumber = 1 [protected]
```

8.43.3.3 mParametersNumber

```
template<typename T >
int exageostat::kernels::Kernel< T >::mParametersNumber = 3 [protected]
```

The documentation for this class was generated from the following file:

- Kernel.hpp

8.44 Kernels Class Reference

A base class for kernel functions.

```
#include <Kernel.hpp>
```

8.44.1 Detailed Description

A base class for kernel functions.

This class provides a base class for kernel functions and contains several utility functions for computing distance metrics and Bessel functions.

The documentation for this class was generated from the following file:

- Kernel.hpp

8.45 exageostat::kernels::KernelsConfigurations Struct Reference

```
#include <Kernel.hpp>
```

Static Public Member Functions

- static std::unordered_map< std::string, int > & GetParametersNumberKernelMap ()
  Returns the static map containing kernel parameter numbers.

8.45.1 Member Function Documentation

8.45.1.1 GetParametersNumberKernelMap()

```
static std::unordered_map<std::string, int>& exageostat::kernels::KernelsConfigurations::Get←
ParametersNumberKernelMap ( ) [inline], [static]
```

Returns the static map containing kernel parameter numbers.

Returns

Reference to the static map.

Static map containing kernel parameter numbers.

The map is initialized only once and retains its value across multiple function invocations.

The documentation for this struct was generated from the following file:

- Kernel.hpp

8.46 exageostat::linearAlgebra::LinearAlgebraFactory< T > Class Template Reference

A class that creates linear algebra solvers based on the input computation type.

```
#include <LinearAlgebraFactory.hpp>
```

Static Public Member Functions

- static std::unique_ptr< LinearAlgebraMethods< T > > CreateLinearAlgebraSolver (common::Computation aComputation)
  Creates a linear algebra solver based on the input computation type.

8.46.1 Detailed Description

template<typename T>
class exageostat::linearAlgebra::LinearAlgebraFactory< T >

A class that creates linear algebra solvers based on the input computation type.

Template Parameters

T	Data Type: float or double.

8.46.2 Member Function Documentation

8.46.2.1 CreateLinearAlgebraSolver()

```
template<typename T >
static std::unique_ptr<LinearAlgebraMethods<T> > exageostat::linearAlgebra::LinearAlgebraFactory<
T >::CreateLinearAlgebraSolver (
            common::Computation aComputation )  [static]
```

Creates a linear algebra solver based on the input computation type.

Parameters

in	*aComputation*	The computation type to create the solver for.

Returns

Pointer to the created linear algebra solver.

The documentation for this class was generated from the following file:

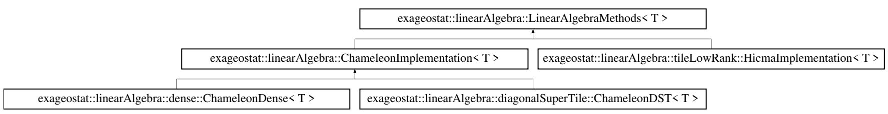
- LinearAlgebraFactory.hpp

8.47 exageostat::linearAlgebra::LinearAlgebraMethods< T > Class Template Reference

A class that defines the interface for linear algebra solvers.

```
#include <LinearAlgebraMethods.hpp>
```

Inheritance diagram for exageostat::linearAlgebra::LinearAlgebraMethods< T >:



Public Member Functions

- virtual ∼LinearAlgebraMethods ()=default

  Virtual destructor to allow calls to the correct concrete destructor.
- void InitiateDescriptors (configurations::Configurations &aConfigurations, dataunits::DescriptorData< T > &aDescriptorData, const int &aP, T ∗apMeasurementsMatrix=nullptr)

  Initializes the descriptors necessary for the linear algebra solver.
- void InitiateFisherDescriptors (configurations::Configurations &aConfigurations, dataunits::DescriptorData< T > &aDescriptorData)

  Initializes the descriptors necessary for the Fisher prediction function.
- void InitiatePredictionDescriptors (configurations::Configurations &aConfigurations, std::unique_ptr< ExaGeoStatData< T >> &aData)

  Initializes the descriptors necessary for the Prediction.
- void InitiateMLOEMMOMDescriptors (configurations::Configurations &aConfigurations, std::unique_ptr< ExaGeoStatData< T >> &aData, const int &aP)

  Initializes the descriptors necessary for the Prediction Auxiliary function MLE-MLOE-MMOM.
- void GenerateSyntheticData (configurations::Configurations &aConfigurations, std::unique_ptr< ExaGeoStatData< T >> &aData, const kernels::Kernel< T > &aKernel)

  Generates synthetic data.
- void GenerateObservationsVector (configurations::Configurations &aConfigurations, std::unique_ptr< ExaGeoStatData< T >> &aData, dataunits::Locations< T > ∗apLocation1, dataunits::Locations< T > ∗apLocation2, dataunits::Locations< T > ∗apLocation3, const int &aDistanceMetric, const kernels::Kernel< T > &aKernel)

  Generates the observations vector.
- virtual T ExaGeoStatMLETile (std::unique_ptr< ExaGeoStatData< T >> &aData, configurations::Configurations &aConfigurations, const double ∗apTheta, T ∗apMeasurementsMatrix, const kernels::Kernel< T > &a←Kernel)=0

  Calculates the log likelihood value of a given value theta.
- virtual void ExaGeoStatLapackCopyTile (const common::UpperLower &aUpperLower, void ∗apA, void ∗apB)=0

  Copies a matrix in the tile layout from source to destination.
- virtual void ExaGeoStatSequenceWait (void ∗apSequence)=0

  Wait for the completion of a sequence.
- virtual void ExaGeoStatCreateSequence (void ∗apSequence)=0

  Create Sequence.
- virtual void ExaGeoStatPotrfTile (const common::UpperLower &aUpperLower, void ∗apA, int aBand, void ∗apCD, void ∗apCrk, const int &aMaxRank, const int &aAcc)=0

  Computes the Cholesky factorization of a symmetric positive definite or Symmetric positive definite matrix.
- virtual void ExaGeoStatTrsmTile (const common::Side &aSide, const common::UpperLower &aUpperLower, const common::Trans &aTrans, const common::Diag &aDiag, const T &aAlpha, void ∗apA, void ∗apCD, void ∗apCrk, void ∗apZ, const int &aMaxRank)=0

  Solves one of the matrix equations op(A)∗X = alpha∗B, or X∗op(A) = alpha∗B.
- void ExaGeoStatPosvTile (const common::UpperLower &aUpperLower, void ∗apA, void ∗apB)

  Solve a positive definite linear system of equations AX = B using tiled algorithms.
- T ∗ ExaGeoStatMLEPredictTile (std::unique_ptr< ExaGeoStatData< T >> &aData, T ∗apTheta, const int &aZMissNumber, const int &aZObsNumber, T ∗apZObs, T ∗apZActual, T ∗apZMiss, configurations::Configurations &aConfiguration, exageostat::dataunits::Locations< T > &aMissLocations, exageostat::dataunits::Locations< T > &aObsLocations, const kernels::Kernel< T > &aKernel)

  Predict missing values base on a set of given values and covariance matrix/.
- T ∗ ExaGeoStatMLENonGaussianPredictTile (std::unique_ptr< ExaGeoStatData< T >> &aData, T ∗ap←Theta, const int &aZMissNumber, const int &aZObsNumber, T ∗apZObs, T ∗apZActual, T ∗apZMiss, configurations::Configurations &aConfiguration, exageostat::dataunits::Locations< T > &aMissLocations, exageostat::dataunits::Locations< T > &aObsLocations, const kernels::Kernel< T > &aKernel)

Predict missing values base on a set of given values and Non-Gaussian covariance matrix/.

- void ExaGeoStatLap2Desc (T ∗apA, const int &aLDA, void ∗apDescA, const common::UpperLower &a←UpperLower)

  Copy Lapack matrix to Descriptor Matrix.
- void ExaGeoStatDesc2Lap (T ∗apA, const int &aLDA, void ∗apDescA, const common::UpperLower &a←UpperLower)

  Copy Descriptor Matrix to Lapack matrix.
- void ExaGeoStatLaSetTile (const common::UpperLower &aUpperLower, T aAlpha, T aBeta, void ∗ap←Descriptor)

  Sets the values of all or part of a two-dimensional Tile.
- void ExaGeoStatGetZObs (configurations::Configurations &aConfigurations, T ∗apZ, const int &aSize, exageostat::dataunits::DescriptorData< T > &aDescData, T ∗apMeasurementsMatrix, const int &aP)

  Copy the Z matrix into a pointer.
- void ExaGeoStatMLETileMLOEMMOM (configurations::Configurations &aConfigurations, std::unique_ptr< ExaGeoStatData< T >> &aData, T ∗apTruthTheta, T ∗apEstimatedTheta, dataunits::Locations< T > &a←MissLocations, dataunits::Locations< T > &aObsLocations, const kernels::Kernel< T > &aKernel)

  Predict missing values based on a set of given values and covariance matrix.
- T ∗ ExaGeoStatFisherTile (configurations::Configurations &aConfigurations, std::unique_ptr< ExaGeoStatData< T >> &aData, T ∗apTheta, const kernels::Kernel< T > &aKernel)

  Maximum Likelihood Evaluation (MLE) Fisher method.
- void ExaGeoStatGeaddTile (const common::Trans &aTrans, const T &aAlpha, void ∗apDescA, const T &a←Beta, void ∗apDescB)

  Perform a matrix addition with scaling.
- void ExaGeoStatTrmmTile (const common::Side &aSide, const common::UpperLower &aUpperLower, const common::Trans &aTrans, const common::Diag &aDiag, const T &alpha, void ∗apDescA, void ∗apDescB)

  Perform a triangular matrix multiplication.
- bool Recover (char ∗apPath, const int &aIterationCount, T ∗apTheta, T ∗apLogLik, const int &aNumParams)

  Recovers theta and log-likelihood from a file.

8.47.1 Detailed Description

template< typename T >
class exageostat::linearAlgebra::LinearAlgebraMethods< T >

A class that defines the interface for linear algebra solvers.

Template Parameters

T	Data Type: float or double.

8.47.2 Constructor & Destructor Documentation

8.47.2.1 ∼LinearAlgebraMethods()

```
template<typename T >
virtual exageostat::linearAlgebra::LinearAlgebraMethods< T >::∼LinearAlgebraMethods ( )  [virtual],
[default]
```

Virtual destructor to allow calls to the correct concrete destructor.

8.47.3 Member Function Documentation

8.47.3.1 InitiateDescriptors()

```
template<typename T >
void exageostat::linearAlgebra::LinearAlgebraMethods< T >::InitiateDescriptors (
            configurations::Configurations & aConfigurations,
            dataunits::DescriptorData< T > & aDescriptorData,
            const int & aP,
            T * apMeasurementsMatrix = nullptr )
```

Initializes the descriptors necessary for the linear algebra solver.

This method initializes the descriptors necessary for the linear algebra solver.

Parameters

in	*aConfigurations*	Configurations object containing relevant settings.
in,out	*aDescriptorData*	Descriptor Data object to be populated with descriptors and data.
in	*aP*	the P value of the kernel multiplied by time slot.
in	*apMeasurementsMatrix*	Pointer to the measurement matrix.

Returns

void

8.47.3.2 InitiateFisherDescriptors()

```
template<typename T >
void exageostat::linearAlgebra::LinearAlgebraMethods< T >::InitiateFisherDescriptors (
            configurations::Configurations & aConfigurations,
            dataunits::DescriptorData< T > & aDescriptorData )
```

Initializes the descriptors necessary for the Fisher prediction function.

Parameters

in	*aConfigurations*	Configurations object containing relevant settings.
in,out	*aDescriptorData*	Descriptor Data object to be populated with descriptors and data.

Returns

void

8.47.3.3 InitiatePredictionDescriptors()

```
template<typename T >
void exageostat::linearAlgebra::LinearAlgebraMethods< T >::InitiatePredictionDescriptors (
            configurations::Configurations & aConfigurations,
            std::unique_ptr< ExaGeoStatData< T >> & aData )
```

Initializes the descriptors necessary for the Prediction.

This method initializes the descriptors necessary for the linear algebra solver.

Parameters

in	*aConfigurations*	Configurations object containing relevant settings.
in,out	*aData*	DescriptorData object to be populated with descriptors and data.

Returns

void

8.47.3.4 InitiateMLOEMMOMDescriptors()

```
template<typename T >
void exageostat::linearAlgebra::LinearAlgebraMethods< T >::InitiateMLOEMMOMDescriptors (
            configurations::Configurations & aConfigurations,
            std::unique_ptr< ExaGeoStatData< T >> & aData,
            const int & aP )
```

Initializes the descriptors necessary for the Prediction Auxiliary function MLE-MLOE-MMOM.

This method initializes the descriptors necessary for the linear algebra solver.

Parameters

in	*aConfigurations*	Configurations object containing relevant settings.
in,out	*aData*	DescriptorData object to be populated with descriptors and data.
in	*aP*	the P value of the kernel multiplied by time slot.

Returns

void

8.47.3.5 GenerateSyntheticData()

```
template<typename T >
void exageostat::linearAlgebra::LinearAlgebraMethods< T >::GenerateSyntheticData (
```

```
configurations::Configurations & aConfigurations,
std::unique_ptr< ExaGeoStatData< T >> & aData,
const kernels::Kernel< T > & aKernel )
```

Generates synthetic data.

Parameters

in	*aConfigurations*	The configurations object containing relevant settings.
in,out	*aData*	ExaGeoStatData object to be populated with synthetic data.
in	*aKernel*	Reference to the kernel object to use.

Returns

void.

8.47.3.6 GenerateObservationsVector()

```
template<typename T >
void exageostat::linearAlgebra::LinearAlgebraMethods< T >::GenerateObservationsVector (
        configurations::Configurations & aConfigurations,
        std::unique_ptr< ExaGeoStatData< T >> & aData,
        dataunits::Locations< T > * apLocation1,
        dataunits::Locations< T > * apLocation2,
        dataunits::Locations< T > * apLocation3,
        const int & aDistanceMetric,
        const kernels::Kernel< T > & aKernel )
```

Generates the observations vector.

Parameters

in	*aConfigurations*	Configurations object containing relevant settings.
in	*aDescriptorData*	pointer to the DescriptorData object holding descriptors and data.
in	*apLocation1*	Pointer to the first set of locations.
in	*apLocation2*	Pointer to the second set of locations.
in	*apLocation3*	Pointer to the third set of locations.
in	*aDistanceMetric*	Specifies the distance metric to use.
in	*aKernel*	Reference to the kernel object to use.

Returns

void

8.47.3.7 ExaGeoStatMLETile()

```
template<typename T >
virtual T exageostat::linearAlgebra::LinearAlgebraMethods< T >::ExaGeoStatMLETile (
            std::unique_ptr< ExaGeoStatData< T >> & aData,
            configurations::Configurations & aConfigurations,
            const double * apTheta,
            T * apMeasurementsMatrix,
            const kernels::Kernel< T > & aKernel ) [pure virtual]
```

Calculates the log likelihood value of a given value theta.

Parameters

in,out	*aData*	DescriptorData object to be populated with descriptors and data.
in	*aConfigurations*	Configurations object containing relevant settings.
in	*apTheta*	Optimization parameter used by NLOPT.
in	*apMeasurementsMatrix*	measurements matrix to be stored in DescZ.
in	*aKernel*	Reference to the kernel object to use.

Returns

log likelihood value

Implemented in exageostat::linearAlgebra::tileLowRank::HicmaImplementation< T >, and exageostat::linearAlgebra::ChameleonImpl

8.47.3.8 ExaGeoStatLapackCopyTile()

```
template<typename T >
virtual void exageostat::linearAlgebra::LinearAlgebraMethods< T >::ExaGeoStatLapackCopyTile (
            const common::UpperLower & aUpperLower,
            void * apA,
            void * apB ) [pure virtual]
```

Copies a matrix in the tile layout from source to destination.

Parameters

in	*aUpperLower*	Specifies the part of the matrix A to be copied to B.
in	*apA*	Source matrix A.
in,out	*apB*	Destination matrix B. On exit, B = A in the locations specified by Upper Lower.

Returns

void

Implemented in exageostat::linearAlgebra::tileLowRank::HicmaImplementation< T >, and exageostat::linearAlgebra::ChameleonImpl

8.47.3.9 ExaGeoStatSequenceWait()

```
template<typename T >
virtual void exageostat::linearAlgebra::LinearAlgebraMethods< T >::ExaGeoStatSequenceWait (
            void * apSequence ) [pure virtual]
```

Wait for the completion of a sequence.

Parameters

in	*apSequence*	apSequence A pointer to either CHAMELEON or HiCMA sequence.

Returns

void

Implemented in exageostat::linearAlgebra::tileLowRank::HicmaImplementation< T >, and exageostat::linearAlgebra::ChameleonImpl

8.47.3.10 ExaGeoStatCreateSequence()

```
template<typename T >
virtual void exageostat::linearAlgebra::LinearAlgebraMethods< T >::ExaGeoStatCreateSequence (
            void * apSequence ) [pure virtual]
```

Create Sequence.

Parameters

out	*apSequence*	A pointer to either CHAMELEON or HiCMA sequence.

Returns

void

Implemented in exageostat::linearAlgebra::tileLowRank::HicmaImplementation< T >, and exageostat::linearAlgebra::ChameleonImpl

8.47.3.11 ExaGeoStatPotrfTile()

```
template<typename T >
virtual void exageostat::linearAlgebra::LinearAlgebraMethods< T >::ExaGeoStatPotrfTile (
            const common::UpperLower & aUpperLower,
            void * apA,
            int aBand,
            void * apCD,
            void * apCrk,
            const int & aMaxRank,
            const int & aAcc ) [pure virtual]
```

Computes the Cholesky factorization of a symmetric positive definite or Symmetric positive definite matrix.

Parameters

in	*aUpperLower*	Whether upper or lower part of the matrix A.
in,out	*apA*	Symmetric matrix A.
in	*aBand*	Diagonal thickness parameter.
in	*apCD*	Additional matrix CD.
in	*apCrk*	Additional matrix Crk.
in	*aMaxRank*	Maximum rank parameter.
in	*aAcc*	Accuracy parameter.

Returns

void

Implemented in exageostat::linearAlgebra::tileLowRank::HicmaImplementation< T >, exageostat::linearAlgebra::diagonalSuperTile::
and exageostat::linearAlgebra::dense::ChameleonDense< T >.

8.47.3.12 ExaGeoStatTrsmTile()

```
template<typename T >
virtual void exageostat::linearAlgebra::LinearAlgebraMethods< T >::ExaGeoStatTrsmTile (
            const common::Side & aSide,
            const common::UpperLower & aUpperLower,
            const common::Trans & aTrans,
            const common::Diag & aDiag,
            const T & aAlpha,
            void * apA,
            void * apCD,
            void * apCrk,
            void * apZ,
            const int & aMaxRank )  [pure virtual]
```

Solves one of the matrix equations op(A)∗X = alpha∗B, or X∗op(A) = alpha∗B.

Parameters

in	*aSide*	Specifies whether op(A) appears on the left or on the right of X.
in	*aUpperLower*	Specifies whether the matrix A is upper triangular or lower triangular.
in	*aTrans*	Specifies the form of op(A) to be used in the matrix multiplication.
in	*aDiag*	Specifies whether or not A is unit triangular.
in	*aAlpha*	Specifies the scalar alpha. When alpha is zero, A is not referenced and B need not be set before entry.
in	*apA*	The triangular matrix A.
in	*apCD*	Additional matrix CD.
in	*apCrk*	Additional matrix Crk.
in,out	*apZ*	The matrix B of dimension, on exit is overwritten by the solution matrix X.
in	*aMaxRank*	Maximum rank parameter.

Returns

void

Implemented in exageostat::linearAlgebra::tileLowRank::HicmaImplementation< T >, and exageostat::linearAlgebra::ChameleonImp

8.47.3.13 ExaGeoStatPosvTile()

```
template<typename T >
void exageostat::linearAlgebra::LinearAlgebraMethods< T >::ExaGeoStatPosvTile (
            const common::UpperLower & aUpperLower,
            void * apA,
            void * apB )
```

Solve a positive definite linear system of equations AX = B using tiled algorithms.

Parameters

in	*aUpperLower*	Specifies whether the matrix A is upper triangular or lower triangular.
in	*apA*	coefficient matrix of the system of linear equations. This matrix is expected to be positive definite.
in	*apB*	Pointer to coefficient matrix of the system of linear equations. This matrix is expected to be positive definite.

Returns

void

8.47.3.14 ExaGeoStatMLEPredictTile()

```
template<typename T >
T* exageostat::linearAlgebra::LinearAlgebraMethods< T >::ExaGeoStatMLEPredictTile (
            std::unique_ptr< ExaGeoStatData< T >> & aData,
            T * apTheta,
            const int & aZMissNumber,
            const int & aZObsNumber,
            T * apZObs,
            T * apZActual,
            T * apZMiss,
            configurations::Configurations & aConfiguration,
            exageostat::dataunits::Locations< T > & aMissLocations,
            exageostat::dataunits::Locations< T > & aObsLocations,
            const kernels::Kernel< T > & aKernel )
```

Predict missing values base on a set of given values and covariance matrix/.

Parameters

in	*aData*	Reference to Data containing different MLE inputs.
in	*apTheta*	theta Vector with three parameter (Variance, Range, Smoothness) that is used to to generate the Covariance Matrix.
in	*aZMissNumber*	number of missing values (unknown observations).
in	*aZObsNumber*	number of observed values (known observations).
in	*apZObs*	observed values vector (known observations).
in	*apZActual*	actual missing values vector (in the case of testing MSPE).
in	*apZMiss*	missing values vector (unknown observations).
in	*aConfigurations*	Configurations object containing relevant settings.
in	*aMissLocations*	Reference to Locations object containing missed locations.
in	*aObsLocations*	Reference to Locations object containing observed locations.
in	*aKernel*	Reference to the kernel object to use.

Returns

the prediction Mean Square Error (MSPE).

8.47.3.15 ExaGeoStatMLENonGaussianPredictTile()

```
template<typename T >
T* exageostat::linearAlgebra::LinearAlgebraMethods< T >::ExaGeoStatMLENonGaussianPredictTile (
            std::unique_ptr< ExaGeoStatData< T >> & aData,
            T * apTheta,
            const int & aZMissNumber,
            const int & aZObsNumber,
            T * apZObs,
            T * apZActual,
            T * apZMiss,
            configurations::Configurations & aConfiguration,
            exageostat::dataunits::Locations< T > & aMissLocations,
            exageostat::dataunits::Locations< T > & aObsLocations,
            const kernels::Kernel< T > & aKernel )
```

Predict missing values base on a set of given values and Non-Gaussian covariance matrix/.

Parameters

in	*aData*	Reference to Data containing different MLE inputs.
in	*apTheta*	theta Vector with three parameter (Variance, Range, Smoothness) that is used to to generate the Covariance Matrix.
in	*aZMissNumber*	number of missing values (unknown observations).
in	*aZObsNumber*	number of observed values (known observations).
in	*apZObs*	observed values vector (known observations).
in	*apZActual*	actual missing values vector (in the case of testing MSPE).
in	*apZMiss*	missing values vector (unknown observations).
in	*aConfigurations*	Configurations object containing relevant settings.
in	*aMissLocations*	Reference to Locations object containing missed locations.
in	*aObsLocations*	Reference to Locations object containing observed locations.
in	*aKernel*	Reference to the kernel object to use.

Returns

the prediction Mean Square Error (MSPE).

8.47.3.16 ExaGeoStatLap2Desc()

```
template<typename T >
void exageostat::linearAlgebra::LinearAlgebraMethods< T >::ExaGeoStatLap2Desc (
            T * apA,
            const int & aLDA,
            void * apDescA,
            const common::UpperLower & aUpperLower )
```

Copy Lapack matrix to Descriptor Matrix.

Parameters

in	*apA*	Lapack Matrix.
in	*aLDA*	Size.
out	*apDescA*	Matrix Descriptor.
in	*aUpperLower*	Specifies Specifies whether the upper or lower triangular part of the covariance matrix is stored.

Returns

void

8.47.3.17 ExaGeoStatDesc2Lap()

```
template<typename T >
void exageostat::linearAlgebra::LinearAlgebraMethods< T >::ExaGeoStatDesc2Lap (
            T * apA,
            const int & aLDA,
            void * apDescA,
            const common::UpperLower & aUpperLower )
```

Copy Descriptor Matrix to Lapack matrix.

Parameters

out	*apA*	Lapack Matrix.
in	*aLDA*	Size.
in	*apDescA*	Matrix Descriptor
in	*aUpperLower*	Specifies whether the upper or lower triangular part of the covariance matrix is stored.

Returns

void

8.47.3.18 ExaGeoStatLaSetTile()

```
template<typename T >
void exageostat::linearAlgebra::LinearAlgebraMethods< T >::ExaGeoStatLaSetTile (
            const common::UpperLower & aUpperLower,
            T aAlpha,
            T aBeta,
            void * apDescriptor )
```

Sets the values of all or part of a two-dimensional Tile.

Parameters

in	*aUpperLower*	Specifies Specifies whether the upper or lower triangular part of the covariance matrix is stored.
in	*aAlpha*	All the off diagonal array elements are set to aAlpha.
in	*aBeta*	All the diagonal array elements are set to aBeta.
out	*apDescriptor*	Pointer to matrix descriptor to be set with aAlpha and aBeta.

Returns

void

8.47.3.19 ExaGeoStatGetZObs()

```
template<typename T >
void exageostat::linearAlgebra::LinearAlgebraMethods< T >::ExaGeoStatGetZObs (
            configurations::Configurations & aConfigurations,
            T * apZ,
            const int & aSize,
            exageostat::dataunits::DescriptorData< T > & aDescData,
            T * apMeasurementsMatrix,
            const int & aP )
```

Copy the Z matrix into a pointer.

Parameters

out	*apZ*	Pointer to an array to copy Z matrix into.
in	*aSize*	Size of the matrix.
in	*aDescData*	Descriptor data containing required Z matrix Descriptor.
in	*aP*	the P value of the kernel multiplied by time slot.

Returns

void

8.47.3.20 ExaGeoStatMLETileMLOEMMOM()

```
template<typename T >
void exageostat::linearAlgebra::LinearAlgebraMethods< T >::ExaGeoStatMLETileMLOEMMOM (
            configurations::Configurations & aConfigurations,
            std::unique_ptr< ExaGeoStatData< T >> & aData,
            T * apTruthTheta,
            T * apEstimatedTheta,
            dataunits::Locations< T > & aMissLocations,
            dataunits::Locations< T > & aObsLocations,
            const kernels::Kernel< T > & aKernel )
```

Predict missing values based on a set of given values and covariance matrix.

This function predicts missing values using the maximum likelihood estimation (MLE), maximum likelihood on the empirical orthogonal functions (MLOE), and method of moments (MMOM).

Parameters

in	*aConfigurations*	Configurations for the prediction.
in,out	*aData*	Data for prediction (input and output).
in	*apTruthTheta*	Pointer to the true theta values.
in	*apEstimatedTheta*	Pointer to the estimated theta values.
in	*aMissLocations*	Locations of missing values.
in	*aObsLocations*	Locations of observed values.
in	*aKernel*	Reference to the kernel object to use.

Returns

void

8.47.3.21 ExaGeoStatFisherTile()

```
template<typename T >
T* exageostat::linearAlgebra::LinearAlgebraMethods< T >::ExaGeoStatFisherTile (
            configurations::Configurations & aConfigurations,
            std::unique_ptr< ExaGeoStatData< T >> & aData,
            T * apTheta,
            const kernels::Kernel< T > & aKernel )
```

Maximum Likelihood Evaluation (MLE) Fisher method.

Parameters

in	*aConfigurations*	Configurations object containing relevant settings.
in,out	*aData*	Descriptor Data object to be populated with descriptors and data.
in	*apTheta*	Pointer containing three parameter (Variance, Range, Smoothness) that is used to to generate the Covariance Matrix.
in	*aKernel*	Reference to the kernel object to use.

Returns

Fisher Matrix

8.47.3.22 ExaGeoStatGeaddTile()

```
template<typename T >
void exageostat::linearAlgebra::LinearAlgebraMethods< T >::ExaGeoStatGeaddTile (
            const common::Trans & aTrans,
            const T & aAlpha,
            void * apDescA,
            const T & aBeta,
            void * apDescB )
```

Perform a matrix addition with scaling.

This function performs a matrix addition with scaling, given the matrices A and B.

Parameters

in	*aTrans*	Specifies whether to transpose matrix A.
in	*aAlpha*	Scaling factor for matrix A.
in	*apDescA*	Descriptor for matrix A.
in	*aBeta*	Scaling factor for matrix B.
in	*apDescB*	Descriptor for matrix B.

Returns

void

8.47.3.23 ExaGeoStatTrmmTile()

```
template<typename T >
void exageostat::linearAlgebra::LinearAlgebraMethods< T >::ExaGeoStatTrmmTile (
            const common::Side & aSide,
            const common::UpperLower & aUpperLower,
            const common::Trans & aTrans,
```

```
            const common::Diag & aDiag,
            const T & alpha,
            void * apDescA,
            void * apDescB )
```

Perform a triangular matrix multiplication.

This function performs triangular matrix multiplication on two matrices A and B.

Parameters

in	*aSide*	Specifies whether the multiplication is performed on the left or right side.
in	*aUpperLower*	Specifies whether the matrix is upper or lower triangular.
in	*aTrans*	Specifies whether to transpose the matrix.
in	*aDiag*	Specifies whether the diagonal elements are unitary or non-unitary.
in	*alpha*	Scaling factor for the multiplication.
in	*apDescA*	Descriptor for matrix A.
in	*apDescB*	Descriptor for matrix B.

8.47.3.24 Recover()

```
template<typename T >
bool exageostat::linearAlgebra::LinearAlgebraMethods< T >::Recover (
            char * apPath,
            const int & aIterationCount,
            T * apTheta,
            T * apLogLik,
            const int & aNumParams )
```

Recovers theta and log-likelihood from a file.

Parameters

in	*apPath*	A pointer to the path of the file from which to recover the data.
in	*aIterationCount*	The iteration count to look for in the file.
in,out	*apTheta*	A pointer to the array where the theta values will be stored.
in,out	*apLogLik*	A pointer to the variable where the log-likelihood value will be stored.
in	*aNumParams*	The number of parameters (elements) in the theta array.

Returns

bool `true` if the specified iteration count is found and successfully parsed, `false` otherwise.

The documentation for this class was generated from the following file:

- LinearAlgebraMethods.hpp

8.48 exageostat::generators::LocationGenerator< T > Class Template Reference

Generates spatial locations based on given parameters.

`#include <LocationGenerator.hpp>`

Static Public Member Functions

- static void GenerateLocations (const int &aN, const int &aTimeSlot, const common::Dimension &aDimension, dataunits::Locations< T > &aLocations)

  Generates the data locations.
- static T UniformDistribution (const T &aRangeLow, const T &aRangeHigh)

  Generate uniform distribution between rangeLow , rangeHigh.
- static void SortLocations (const int &aN, const common::Dimension &aDimension, dataunits::Locations< T > &aLocations)

  Sort locations in Morton order (input points must be in [0;1]x[0;1] square]).

8.48.1 Detailed Description

template<typename T>
class exageostat::generators::LocationGenerator< T >

Generates spatial locations based on given parameters.

Template Parameters

T	Data Type: float or double

8.48.2 Member Function Documentation

8.48.2.1 GenerateLocations()

```
template<typename T >
static void exageostat::generators::LocationGenerator< T >::GenerateLocations (
            const int & aN,
            const int & aTimeSlot,
            const common::Dimension & aDimension,
            dataunits::Locations< T > & aLocations )  [static]
```

Generates the data locations.

This method generates the X, Y, and Z variables used to define the locations of the data points.

Parameters

in	*aN*	The number of data points.
in	*aTimeSlot*	The time slot.
in	*aDimension*	The dimension of the locations.
out	*aLocations*	Reference to the Locations object where the generated data will be stored.

Returns

void

8.48.2.2 UniformDistribution()

```
template<typename T >
static T exageostat::generators::LocationGenerator< T >::UniformDistribution (
            const T & aRangeLow,
            const T & aRangeHigh )  [static]
```

Generate uniform distribution between rangeLow , rangeHigh.

Parameters

in	*aRangeLow*	The Lower range.
in	*aRangeHigh*	The Higher range.

Returns

The scaled uniform distribution between the two bounds.

8.48.2.3 SortLocations()

```
template<typename T >
static void exageostat::generators::LocationGenerator< T >::SortLocations (
            const int & aN,
            const common::Dimension & aDimension,
            dataunits::Locations< T > & aLocations )  [static]
```

Sort locations in Morton order (input points must be in [0;1]x[0;1] square]).

Parameters

in	*aN*	The problem size divided by P-Grid.
in	*aDimension*	Dimension of locations.
in,out	*aLocations*	Locations to be sorted.

Returns

void

The documentation for this class was generated from the following file:

- LocationGenerator.hpp

8.49 exageostat::dataunits::Locations< T > Class Template Reference

A class containing methods to set and get location data.

```
#include <Locations.hpp>
```

Public Member Functions

- Locations (const int &aSize, const exageostat::common::Dimension &aDimension)
  Constructor.
- Locations (const Locations< T > &aLocations)=default
  Default copy constructor.
- ∼Locations ()
  destructor for Locations.
- void SetLocationX (T &aLocationX, const int &aSize)
  Setter for LocationX.
- T ∗ GetLocationX ()
  Getter for LocationX.
- void SetLocationY (T &aLocationY, const int &aSize)
  Setter for LocationY.
- T ∗ GetLocationY ()
  Getter for LocationY.
- void SetLocationZ (T &aLocationZ, const int &aSize)
  Setter for LocationZ.
- T ∗ GetLocationZ ()
  Getter for LocationZ.
- void SetSize (const int &aSize)
  Setter for mSize.
- int GetSize ()
  Getter for mSize.
- void SetDimension (const common::Dimension &aDimension)
  Setter for Dimensions.
- common::Dimension GetDimension ()
  Getter for Dimension.

Private Attributes

- T ∗ mpLocationX = nullptr

  Pointer to X data.
- T ∗ mpLocationY = nullptr

  Pointer to Y data.
- T ∗ mpLocationZ = nullptr

  Pointer to Z data.
- int mSize = 1

  Size of each dimension.
- common::Dimension mDimension = common::Dimension2D

  Data dimensions.

8.49.1 Detailed Description

template<typename T>
class exageostat::dataunits::Locations< T >

A class containing methods to set and get location data.

Template Parameters

T	Data Type: float or double

8.49.2 Constructor & Destructor Documentation

8.49.2.1 Locations() [1/2]

```
template<typename T >
exageostat::dataunits::Locations< T >::Locations (
            const int & aSize,
            const exageostat::common::Dimension & aDimension )
```

Constructor.

Parameters

in	*aSize*	The number of data points.
in	*aDimension*	The dimensionality of the data points.

Returns

void

8.49.2.2 Locations() [2/2]

```
template<typename T >
exageostat::dataunits::Locations< T >::Locations (
            const Locations< T > & aLocations )  [default]
```

Default copy constructor.

Parameters

in	*aLocations*	Locations to be copied.

8.49.2.3 ∼Locations()

```
template<typename T >
exageostat::dataunits::Locations< T >::∼Locations ( )
```

destructor for Locations.

8.49.3 Member Function Documentation

8.49.3.1 SetLocationX()

```
template<typename T >
void exageostat::dataunits::Locations< T >::SetLocationX (
            T & aLocationX,
            const int & aSize )
```

Setter for LocationX.

Parameters

in	*aLocationX*	Reference to X data.

Returns

void

8.49.3.2 GetLocationX()

```
template<typename T >
T* exageostat::dataunits::Locations< T >::GetLocationX ( )
```

Getter for LocationX.

Returns

Pointer to X data.

8.49.3.3 SetLocationY()

```
template<typename T >
void exageostat::dataunits::Locations< T >::SetLocationY (
            T & aLocationY,
            const int & aSize )
```

Setter for LocationY.

Parameters

in	*aLocationY*	Reference to Y data.

Returns

void

8.49.3.4 GetLocationY()

```
template<typename T >
T* exageostat::dataunits::Locations< T >::GetLocationY ( )
```

Getter for LocationY.

Returns

Pointer to Y data.

8.49.3.5 SetLocationZ()

```
template<typename T >
void exageostat::dataunits::Locations< T >::SetLocationZ (
            T & aLocationZ,
            const int & aSize )
```

Setter for LocationZ.

Parameters

in	*aLocationZ*	Reference to Z data.

Returns

void

8.49.3.6 GetLocationZ()

```
template<typename T >
T* exageostat::dataunits::Locations< T >::GetLocationZ ( )
```

Getter for LocationZ.

Returns

Pointer to Z data.

8.49.3.7 SetSize()

```
template<typename T >
void exageostat::dataunits::Locations< T >::SetSize (
            const int & aSize )
```

Setter for mSize.

Parameters

in	*aSize.*	

Returns

void

8.49.3.8 GetSize()

```
template<typename T >
int exageostat::dataunits::Locations< T >::GetSize ( )
```

Getter for mSize.

Returns

Locations size.

8.49.3.9 SetDimension()

```
template<typename T >
void exageostat::dataunits::Locations< T >::SetDimension (
            const common::Dimension & aDimension )
```

Setter for Dimensions.

Parameters

in	*aDimension.*	

Returns

void

8.49.3.10 GetDimension()

```
template<typename T >
common::Dimension exageostat::dataunits::Locations< T >::GetDimension ( )
```

Getter for Dimension.

Returns

Locations dimension.

8.49.4 Member Data Documentation

8.49.4.1 mpLocationX

```
template<typename T >
T* exageostat::dataunits::Locations< T >::mpLocationX = nullptr [private]
```

Pointer to X data.

8.49.4.2 mpLocationY

```
template<typename T >
T* exageostat::dataunits::Locations< T >::mpLocationY = nullptr [private]
```

Pointer to Y data.

8.49.4.3 mpLocationZ

```
template<typename T >
T* exageostat::dataunits::Locations< T >::mpLocationZ = nullptr [private]
```

Pointer to Z data.

8.49.4.4 mSize

```
template<typename T >
int exageostat::dataunits::Locations< T >::mSize = 1 [private]
```

Size of each dimension.

8.49.4.5 mDimension

```
template<typename T >
common::Dimension exageostat::dataunits::Locations< T >::mDimension = common::Dimension2D
[private]
```

Data dimensions.

The documentation for this class was generated from the following file:

- Locations.hpp

8.50 exageostat::dataunits::mModelingData< T > Struct Template Reference

Struct containing all the data needed for modeling.

```
#include <ModelingDataHolders.hpp>
```

Public Member Functions

- mModelingData (std::unique_ptr< ExaGeoStatData< T >> &aData, configurations::Configurations &a↩ Configuration, T &aMatrix, const kernels::Kernel< T > &aKernel)

    Constructor.

Public Attributes

- std::unique_ptr< ExaGeoStatData< T > > * mpData

    ExaGeoStatData< T > object containing needed descriptors, and locations.
- configurations::Configurations * mpConfiguration

    Configurations object containing user input data.
- const kernels::Kernel< T > * mpKernel

    Used Kernel for ExaGeoStat Modeling Data.
- T * mpMeasurementsMatrix

    User Input Measurements Matrix.

8.50.1 Detailed Description

template<typename T>
struct exageostat::dataunits::mModelingData< T >

Struct containing all the data needed for modeling.

Template Parameters

T	The data type of the data.

8.50.2 Constructor & Destructor Documentation

8.50.2.1 mModelingData()

```
template<typename T >
exageostat::dataunits::mModelingData< T >::mModelingData (
            std::unique_ptr< ExaGeoStatData< T >> & aData,
            configurations::Configurations & aConfiguration,
            T & aMatrix,
            const kernels::Kernel< T > & aKernel )  [inline]
```

Constructor.

Parameters

aData	The ExaGeoStatData object.
aConfiguration	The Configurations object.
aKernel	The Kernel object.

8.50.3 Member Data Documentation

8.50.3.1 mpData

```
template<typename T >
std::unique_ptr<ExaGeoStatData<T> >* exageostat::dataunits::mModelingData< T >::mpData
```

ExaGeoStatData<T> object containing needed descriptors, and locations.

8.50.3.2 mpConfiguration

```
template<typename T >
configurations::Configurations* exageostat::dataunits::mModelingData< T >::mpConfiguration
```

Configurations object containing user input data.

8.50.3.3 mpKernel

```
template<typename T >
const kernels::Kernel<T>* exageostat::dataunits::mModelingData< T >::mpKernel
```

Used Kernel for ExaGeoStat Modeling Data.

8.50.3.4 mpMeasurementsMatrix

```
template<typename T >
T* exageostat::dataunits::mModelingData< T >::mpMeasurementsMatrix
```

User Input Measurements Matrix.

The documentation for this struct was generated from the following file:

- ModelingDataHolders.hpp

8.51 exageostat::runtime::NonGaussianLoglike< T > Class Template Reference

A class for starpu codelet non gaussian loglike.

```
#include <non-gaussian-loglike-codelet.hpp>
```

Public Member Functions

- NonGaussianLoglike ()=default

    Constructor for NonGaussianLoglike.
- ∼NonGaussianLoglike ()=default

    Default destructor.
- void InsertTask (void ∗apDescZ, void ∗apDescSum, const T ∗apTheta, std::unique_ptr< StarPuHelpers > &aStarPuHelper)

    Inserts a task for Non-Gaussian log-likelihood codelet processing.

Static Private Member Functions

- static void cl_non_gaussian_loglike_function (void ∗∗apBuffers, void ∗apCodeletArguments)

    Executes the Non-Gaussian log-likelihood codelet function.
- static double core_non_gaussian_loglike_helper (const T ∗apDescriptorZ, const T ∗apLocalTheta, const int &aSize)

    Helper function for calculating the log-likelihood under a non-Gaussian distribution.

Static Private Attributes

- static struct starpu_codelet cl_non_gaussian_loglike

    starpu_codelet struct

8.51.1 Detailed Description

template<typename T>
class exageostat::runtime::NonGaussianLoglike< T >

A class for starpu codelet non gaussian loglike.

Template Parameters

T	Data Type: float or double

This class encapsulates the struct cl_non_gaussian_loglike and its CPU functions.

8.51.2 Constructor & Destructor Documentation

8.51.2.1 NonGaussianLoglike()

```
template<typename T >
exageostat::runtime::NonGaussianLoglike< T >::NonGaussianLoglike ( ) [default]
```

Constructor for NonGaussianLoglike.

8.51.2.2 ∼NonGaussianLoglike()

```
template<typename T >
exageostat::runtime::NonGaussianLoglike< T >::∼NonGaussianLoglike ( ) [default]
```

Default destructor.

8.51.3 Member Function Documentation

8.51.3.1 InsertTask()

```
template<typename T >
void exageostat::runtime::NonGaussianLoglike< T >::InsertTask (
            void * apDescZ,
            void * apDescSum,
            const T * apTheta,
            std::unique_ptr< StarPuHelpers > & aStarPuHelper )
```

Inserts a task for Non-Gaussian log-likelihood codelet processing.

Parameters

in	*apDescZ*	A pointer to the descriptor for the dataset.
in,out	*apDescSum*	A pointer to the descriptor for the sum.
in	*apTheta*	A pointer to the transformation parameters.
in	*aStarPuHelpers*	A reference to a unique pointer of StarPuHelpers, used for accessing and managing data.

Returns

void

8.51.3.2 cl_non_gaussian_loglike_function()

```
template<typename T >
static void exageostat::runtime::NonGaussianLoglike< T >::cl_non_gaussian_loglike_function (
            void ** apBuffers,
            void * apCodeletArguments ) [static], [private]
```

Executes the Non-Gaussian log-likelihood codelet function.

Parameters

in	*apBuffers*	An array of pointers to the buffers.
in	*apCodeletArguments*	A pointer to the codelet arguments structure, which includes the dataset size, offset, and the transformation parameters.

Returns

void

8.51.3.3 core_non_gaussian_loglike_helper()

```
template<typename T >
static double exageostat::runtime::NonGaussianLoglike< T >::core_non_gaussian_loglike_helper (
            const T * apDescriptorZ,
            const T * apLocalTheta,
            const int & aSize )  [static], [private]
```

Helper function for calculating the log-likelihood under a non-Gaussian distribution.

Parameters

in	*apDescriptorZ*	A pointer to the dataset.
in	*apLocalTheta*	A pointer to the transformation parameters.
in	*aSize*	The size of the dataset.

Returns

T The calculated log-likelihood of the dataset under a non-Gaussian distribution.

8.51.4 Member Data Documentation

8.51.4.1 cl_non_gaussian_loglike

```
template<typename T >
struct starpu_codelet exageostat::runtime::NonGaussianLoglike< T >::cl_non_gaussian_loglike
[static], [private]
```

starpu_codelet struct

The documentation for this class was generated from the following file:

- non-gaussian-loglike-codelet.hpp

8.52 exageostat::runtime::NonGaussianTransform< T > Class Template Reference

A class for starpu codelet non gaussian transform.

`#include <non-gaussian-transform-codelet.hpp>`

Public Member Functions

- NonGaussianTransform ()=default

    Default constructor.
- ∼NonGaussianTransform ()=default

    Default destructor.
- void InsertTask (void ∗apDescZ, const T ∗apTheta, std::unique_ptr< StarPuHelpers > &apStarPuHelpers)

    Inserts a task for Non-Gaussian transformation codelet processing.

Static Private Member Functions

- static void cl_non_gaussian_transform_function (void ∗∗apBuffers, void ∗apCodeletArguments)

    Executes the Non-Gaussian transformation codelet function.
- static void core_non_gaussian_transform_helper (T ∗apDescripZ, const T ∗apLocalTheta, const int &aSize)

    Helper function for transforming a dataset to a non-Gaussian representation. It applies the Non-Gaussian transformation to each element of a dataset using the Newton-Raphson method for finding the root of a function.
- static double newton_raphson (T apDescriptorZ, T aTransLocation, T aTransScale, T aTransShape, T aTransKurtosis, T aEpsilon)

    Implements the Newton-Raphson method for finding the root of a function.
- static double tukeyGHTransfor (T aOriginalValue, T aCurrentValue, T aTransLocation, T aTransScale, T aTransShape, T aTransKurtosis)

    Calculates the Non-Gaussian transformation of a value.
- static double tukeyGHDiferencial (T aCurrentValue, T aTransScale, T aTransShape, T aTransKurtosis)

    Calculates the derivative of the Non-Gaussian transformation.

Static Private Attributes

- static struct starpu_codelet cl_non_gaussian_transform

    starpu_codelet struct

8.52.1 Detailed Description

template<typename T>
class exageostat::runtime::NonGaussianTransform< T >

A class for starpu codelet non gaussian transform.

Template Parameters

T	Data Type: float or double

This class encapsulates the struct cl_non_gaussian_transform and its CPU functions.

8.52.2 Constructor & Destructor Documentation

8.52.2.1 NonGaussianTransform()

```
template<typename T >
exageostat::runtime::NonGaussianTransform< T >::NonGaussianTransform ( )  [default]
```

Default constructor.

8.52.2.2 ∼NonGaussianTransform()

```
template<typename T >
exageostat::runtime::NonGaussianTransform< T >::∼NonGaussianTransform ( )  [default]
```

Default destructor.

8.52.3 Member Function Documentation

8.52.3.1 InsertTask()

```
template<typename T >
void exageostat::runtime::NonGaussianTransform< T >::InsertTask (
            void * apDescZ,
            const T * apTheta,
            std::unique_ptr< StarPuHelpers > & apStarPuHelpers )
```

Inserts a task for Non-Gaussian transformation codelet processing.

Parameters

in,out	*apDescZ*	A pointer to the descriptor for the dataset.
in	*apTheta*	A pointer to the transformation parameters.
in	*apStarPuHelpers*	A reference to a unique pointer of StarPuHelpers, used for accessing and managing data.

Returns

void

8.52.3.2 cl_non_gaussian_transform_function()

```
template<typename T >
static void exageostat::runtime::NonGaussianTransform< T >::cl_non_gaussian_transform_function
```

```
(
            void ** apBuffers,
            void * apCodeletArguments )   [static], [private]
```

Executes the Non-Gaussian transformation codelet function.

Parameters

in	*apBuffers*	An array of pointers to the buffers.
in	*apCodeletArguments*	A pointer to the codelet arguments structure, which includes the dataset size, offset, and the transformation parameters.

Returns

void

8.52.3.3 core_non_gaussian_transform_helper()

```
template<typename T >
static void exageostat::runtime::NonGaussianTransform< T >::core_non_gaussian_transform_helper
(
            T * apDescripZ,
            const T * apLocalTheta,
            const int & aSize )   [static], [private]
```

Helper function for transforming a dataset to a non-Gaussian representation. It applies the Non-Gaussian transformation to each element of a dataset using the Newton-Raphson method for finding the root of a function.

Parameters

in,out	*apDescripZ*	A pointer to the dataset to be transformed.
in	*apLocalTheta*	A pointer to the transformation parameters.
in	*aSize*	The size of the dataset.

Returns

void

8.52.3.4 newton_raphson()

```
template<typename T >
static double exageostat::runtime::NonGaussianTransform< T >::newton_raphson (
            T apDescriptorZ,
            T aTransLocation,
            T aTransScale,
            T aTransShape,
```

```
T aTransKurtosis,
T aEpsilon )  [static], [private]
```

Implements the Newton-Raphson method for finding the root of a function.

Parameters

in	*apDescriptorZ*	The initial guess for the root.
in	*aTransLocation*	The location parameter of the transformation.
in	*aTransScale*	The scale parameter of the transformation.
in	*aTransShape*	The shape parameter of the transformation.
in	*aTransKurtosis*	The kurtosis parameter of the transformation.
in	*aEpsilon*	The error threshold for the root-finding process.

Returns

T The calculated root of the function.

8.52.3.5 tukeyGHTransfor()

```
template<typename T >
static double exageostat::runtime::NonGaussianTransform< T >::tukeyGHTransfor (
            T aOriginalValue,
            T aCurrentValue,
            T aTransLocation,
            T aTransScale,
            T aTransShape,
            T aTransKurtosis )  [static], [private]
```

Calculates the Non-Gaussian transformation of a value.

Parameters

in	*aOriginalValue*	The original value to be transformed.
in	*aCurrentValue*	The current value of the transformation variable.
in	*aTransLocation*	The location parameter of the transformation.
in	*aTransScale*	The scale parameter of the transformation.
in	*aTransShape*	The shape parameter of the transformation.
in	*aTransKurtosis*	The kurtosis parameter of the transformation.

Returns

T The transformed value.

8.52.3.6 tukeyGHDiferencial()

```
template<typename T >
static double exageostat::runtime::NonGaussianTransform< T >::tukeyGHDiferencial (
            T aCurrentValue,
```

```
T aTransScale,
T aTransShape,
T aTransKurtosis ) [static], [private]
```

Calculates the derivative of the Non-Gaussian transformation.

Parameters

in	*aCurrentValue*	The current value of the transformation variable.
in	*aTransScale*	The scale parameter of the transformation.
in	*aTransShape*	The shape parameter of the transformation.
in	*aTransKurtosis*	The kurtosis parameter of the transformation.

Returns

T The derivative of the transformation at the given value.

8.52.4 Member Data Documentation

8.52.4.1 cl_non_gaussian_transform

```
template<typename T >
struct starpu_codelet exageostat::runtime::NonGaussianTransform< T >::cl_non_gaussian_transform
[static], [private]
```

starpu_codelet struct

The documentation for this class was generated from the following file:

- non-gaussian-transform-codelet.hpp

8.53 exageostat::plugins::PluginRegistry< T > Class Template Reference

Template class for registering and creating plugins.

`#include <PluginRegistry.hpp>`

Public Types

- typedef std::function< T *()> FactoryFunction

  Function type that returns a pointer to an instance of T.
- typedef std::unordered_map< std::string, FactoryFunction > FactoryMap

  Unordered map that maps plugin names to their corresponding factory functions.

Static Public Member Functions

- static bool Add (const std::string &name, FactoryFunction fac)

  Adds a factory function to the FactoryMap under the given plugin name.
- static T * Create (const std::string &aName, const int &aTimeSlot)

  Creates an instance of the plugin with the given name.

Static Private Member Functions

- static FactoryMap & GetFactoryMap ()

  Returns a reference to the FactoryMap singleton.

8.53.1 Detailed Description

template<typename T>
class exageostat::plugins::PluginRegistry< T >

Template class for registering and creating plugins.

Template Parameters

T	Data Type: float or double

8.53.2 Member Typedef Documentation

8.53.2.1 FactoryFunction

```
template<typename T >
typedef std::function<T *()> exageostat::plugins::PluginRegistry< T >::FactoryFunction
```

Function type that returns a pointer to an instance of T.

8.53.2.2 FactoryMap

```
template<typename T >
typedef std::unordered_map<std::string, FactoryFunction> exageostat::plugins::PluginRegistry<
T >::FactoryMap
```

Unordered map that maps plugin names to their corresponding factory functions.

8.53.3 Member Function Documentation

8.53.3.1 Add()

```
template<typename T >
static bool exageostat::plugins::PluginRegistry< T >::Add (
            const std::string & name,
            FactoryFunction fac ) [inline], [static]
```

Adds a factory function to the FactoryMap under the given plugin name.

Parameters

in	*name*	The name of the plugin to be added.
in	*fac*	The factory function to be added.

Returns

true if the factory function was successfully added, false otherwise.

8.53.3.2 Create()

```
template<typename T >
static T* exageostat::plugins::PluginRegistry< T >::Create (
            const std::string & aName,
            const int & aTimeSlot ) [inline], [static]
```

Creates an instance of the plugin with the given name.

Parameters

in	*name*	The name of the plugin to be created.

Returns

A pointer to the created plugin, or nullptr if the plugin could not be created.

8.53.3.3 GetFactoryMap()

```
template<typename T >
static FactoryMap& exageostat::plugins::PluginRegistry< T >::GetFactoryMap ( ) [inline],
[static], [private]
```

Returns a reference to the FactoryMap singleton.

Returns

A reference to the FactoryMap singleton.

The documentation for this class was generated from the following file:

- PluginRegistry.hpp

8.54 exageostat::prediction::Prediction< T > Class Template Reference

Class to handle different Prediction Module calls.

```
#include <Prediction.hpp>
```

Static Public Member Functions

- static void PredictMissingData (std::unique_ptr< ExaGeoStatData< T >> &aData, configurations::Configurations &aConfigurations, T *apMeasurementsMatrix, const kernels::Kernel< T > &aKernel, dataunits::Locations< T > *apTrainLocations=nullptr, dataunits::Locations< T > *apTestLocations=nullptr)

    Takes care of calling the MSPE function, and the appropriate auxiliary function.
- static void InitializePredictionArguments (configurations::Configurations &aConfigurations, std::unique_ptr< ExaGeoStatData< T >> &aData, std::unique_ptr< exageostat::linearAlgebra::LinearAlgebraMethods< T >> &aLinearAlgebraSolver, T *apZObs, T *apZActual, exageostat::dataunits::Locations< T > &aMissLocation, exageostat::dataunits::Locations< T > &aObsLocation, T *apMeasurementsMatrix, const int &aP, dataunits::Locations< T > *apTrainLocations, dataunits::Locations< T > *apTestLocations)

    Initializes needed pointers for prediction.

8.54.1 Detailed Description

template<typename T>
class exageostat::prediction::Prediction< T >

Class to handle different Prediction Module calls.

@Class Prediction

Template Parameters

T	Data Type: float or double.

8.54.2 Member Function Documentation

8.54.2.1 PredictMissingData()

```
template<typename T >
static void exageostat::prediction::Prediction< T >::PredictMissingData (
            std::unique_ptr< ExaGeoStatData< T >> & aData,
            configurations::Configurations & aConfigurations,
            T * apMeasurementsMatrix,
            const kernels::Kernel< T > & aKernel,
            dataunits::Locations< T > * apTrainLocations = nullptr,
            dataunits::Locations< T > * apTestLocations = nullptr ) [static]
```

Takes care of calling the MSPE function, and the appropriate auxiliary function.

Parameters

in,out	*aData*	Reference to an ExaGeoStatData<T> object containing needed descriptors, and locations.
in	*aConfigurations*	Reference to Configurations object containing user input data.
in	*apMeasurementsMatrix*	Pointer to the user input measurements matrix.
in	*aKernel*	Reference to the kernel object to use.
in	*apTrainLocations*	(Optional) Pointer to Locations represents training locations. these are used in training phase.
in	*apTestLocations*	(Optional) Pointer to Locations represents test locations. These are used in prediction phase.

Returns

void

8.54.2.2 InitializePredictionArguments()

```
template<typename T >
static void exageostat::prediction::Prediction< T >::InitializePredictionArguments (
            configurations::Configurations & aConfigurations,
            std::unique_ptr< ExaGeoStatData< T >> & aData,
            std::unique_ptr< exageostat::linearAlgebra::LinearAlgebraMethods< T >> & a←
LinearAlgebraSolver,
            T * apZObs,
            T * apZActual,
            exageostat::dataunits::Locations< T > & aMissLocation,
            exageostat::dataunits::Locations< T > & aObsLocation,
            T * apMeasurementsMatrix,
            const int & aP,
            dataunits::Locations< T > * apTrainLocations,
            dataunits::Locations< T > * apTestLocations ) [static]
```

Initializes needed pointers for prediction.

Parameters

in	*aConfigurations*	Reference to Configurations object containing user input data.

Parameters

in,out	*aData*	Reference to an ExaGeoStatData<T> object containing needed descriptors, and locations.
in	*aLinearAlgebraSolver*	linear algebra solver depending on implementation.
out	*apZObs*	Pointer to be filled with observation measurements
out	*apZActual*	Pointer to be filled with actual measurements
out	*aMissLocation*	Location object to be filled with missed locations.
out	*aObsLocation*	Location object to be filled with missed locations.
in	*apMeasurementsMatrix*	Pointer to the user input measurements matrix.
in	*aP*	the P value of the kernel multiplied by time slot.
in	*apTrainLocations*	(Optional) Pointer to Locations represents training locations. these are used in training phase.
in	*apTestLocations*	(Optional) Pointer to Locations represents test locations. These are used in prediction phase.

Returns

void

The documentation for this class was generated from the following file:

- Prediction.hpp

8.55 exageostat::prediction::PredictionAuxiliaryFunctions< T > Class Template Reference

Class to define and implement different Prediction Module Auxiliary Functions.

```
#include <PredictionAuxiliaryFunctions.hpp>
```

Static Public Member Functions

- static void PredictIDW (T ∗apZMiss, T ∗apZActual, T ∗apZObs, const int &aZMissNumber, const int &a←ZObsNumber, exageostat::dataunits::Locations< T > &aMissLocation, exageostat::dataunits::Locations< T > &aObsLocation, T ∗apMSPE)

  implements the Inverse Distance Weighting (IDW) interpolation method for predicting missing values based on available observed values.

8.55.1 Detailed Description

template<typename T>
class exageostat::prediction::PredictionAuxiliaryFunctions< T >

Class to define and implement different Prediction Module Auxiliary Functions.

@Class PredictionAuxiliaryFunctions

Template Parameters

T	Data Type: float or double.

8.55.2 Member Function Documentation

8.55.2.1 PredictIDW()

```
template<typename T >
static void exageostat::prediction::PredictionAuxiliaryFunctions< T >::PredictIDW (
            T * apZMiss,
            T * apZActual,
            T * apZObs,
            const int & aZMissNumber,
            const int & aZObsNumber,
            exageostat::dataunits::Locations< T > & aMissLocation,
            exageostat::dataunits::Locations< T > & aObsLocation,
            T * apMSPE )  [static]
```

implements the Inverse Distance Weighting (IDW) interpolation method for predicting missing values based on available observed values.

Parameters

in	*apZMiss*	Pointer to the missed measurements.
in	*apZActual*	Pointer to the actual measurements.
in	*apZObs*	Pointer to the observed measurements.
in	*aZMissNumber*	Number of missed measurements.
in	*aZObsNumber*	Number of observed measurements.
in	*aMissLocation*	Reference to the missed locations.
in	*aObsLocation*	Reference to the observed locations.
out	*apMSPE*	Pointer to be filled with MSPE value.

Returns

T Array provides insight into the accuracy of the IDW-interpolated predictions for missing values

The documentation for this class was generated from the following file:

- PredictionAuxiliaryFunctions.hpp

8.56 exageostat::prediction::PredictionHelpers< T > Class Template Reference

Class to define and implement different Prediction Module helpers functions.

`#include <PredictionHelpers.hpp>`

Static Public Member Functions

- static void PickRandomPoints (configurations::Configurations &aConfigurations, std::unique_ptr< ExaGeoStatData< T >> &aData, T ∗apZObs, T ∗apZActual, T ∗apZ, exageostat::dataunits::Locations< T > &aMissLocation, exageostat::dataunits::Locations< T > &aObsLocation, const int &aP)

  Pick random Z points for prediction depending on p.
- static void Shuffle (T ∗apArray, exageostat::dataunits::Locations< T > &aLocations, int aSize)

  Shuffle array.
- static void Shuffle (T ∗apArray1, T ∗apArray2, exageostat::dataunits::Locations< T > &aLocations, int aSize)

  Shuffle array.
- static void Shuffle (T ∗apArray1, T ∗apArray2, T ∗apArray3, exageostat::dataunits::Locations< T > &a↩ Locations, int aSize)

  Shuffle array.
- static void SortArray (uint32_t ∗aData, int aCount)

  Sorts the input data using the C++, the pre-defined standard library function sort().
- static int SortInplace (int aN, exageostat::dataunits::Locations< T > &aLocations, T ∗apZ)

  Sorts location data and corresponding observation values in-place based on Locations coordinates.

8.56.1 Detailed Description

template<typename T>
class exageostat::prediction::PredictionHelpers< T >

Class to define and implement different Prediction Module helpers functions.

@Class PredictionHelpers

Template Parameters

T	Data Type: float or double.

8.56.2 Member Function Documentation

8.56.2.1 PickRandomPoints()

```
template<typename T >
static void exageostat::prediction::PredictionHelpers< T >::PickRandomPoints (
            configurations::Configurations & aConfigurations,
            std::unique_ptr< ExaGeoStatData< T >> & aData,
            T * apZObs,
            T * apZActual,
            T * apZ,
            exageostat::dataunits::Locations< T > & aMissLocation,
            exageostat::dataunits::Locations< T > & aObsLocation,
            const int & aP )  [static]
```

Pick random Z points for prediction depending on p.

Parameters

in	*aConfigurations*	Configurations object containing relevant settings.
in,out	*aData*	Reference ExaGeoStatData object populated with locations and descriptor data.
out	*apZObs*	Pointer to be filled with observed measurements.
out	*apZActual*	Pointer to be filled with actual measurements.
in	*apZ*	Pointer to a copy of the measurements matrix.
out	*aMissLocation*	Location object to be filled with missed locations.
out	*aObsLocation*	Location object to be filled with missed locations.
in	*aP*	the P value of the kernel multiplied by time slot.

Returns

void

8.56.2.2 Shuffle() [1/3]

```
template<typename T >
static void exageostat::prediction::PredictionHelpers< T >::Shuffle (
            T * apArray,
            exageostat::dataunits::Locations< T > & aLocations,
            int aSize ) [static]
```

Shuffle array.

Parameters

in,out	*apArray*	Array to be shuffled.
in,out	*aLocations*	Locations to be shuffled.
out	*aSize*	Size of data.

Returns

void

8.56.2.3 Shuffle() [2/3]

```
template<typename T >
static void exageostat::prediction::PredictionHelpers< T >::Shuffle (
            T * apArray1,
            T * apArray2,
            exageostat::dataunits::Locations< T > & aLocations,
            int aSize ) [static]
```

Shuffle array.

Parameters

in,out	*apArray1*	First Array to be shuffled.
in,out	*apArray2*	Second Array to be shuffled.
in,out	*aLocations*	Locations to be shuffled.
out	*aSize*	Size of data.

Returns

void

8.56.2.4 Shuffle() [3/3]

```
template<typename T >
static void exageostat::prediction::PredictionHelpers< T >::Shuffle (
            T * apArray1,
            T * apArray2,
            T * apArray3,
            exageostat::dataunits::Locations< T > & aLocations,
            int aSize ) [static]
```

Shuffle array.

Parameters

in,out	*apArray1*	First Array to be shuffled.
in,out	*apArray2*	Second Array to be shuffled.
in,out	*apArray3*	Third Array to be shuffled.
in,out	*aLocations*	Locations to be shuffled.
out	*aSize*	Size of data.

Returns

void

8.56.2.5 SortArray()

```
template<typename T >
static void exageostat::prediction::PredictionHelpers< T >::SortArray (
            uint32_t * aData,
            int aCount ) [static]
```

Sorts the input data using the C++, the pre-defined standard library function sort().

Parameters

aData[in]	Pointer to the array of data to be sorted.
aCount[in]	Number of elements in the input array.
aDimension[in]	Dimension of the input data.

Returns

void

8.56.2.6 SortInplace()

```
template<typename T >
static int exageostat::prediction::PredictionHelpers< T >::SortInplace (
            int aN,
            exageostat::dataunits::Locations< T > & aLocations,
            T * apZ ) [static]
```

Sorts location data and corresponding observation values in-place based on Locations coordinates.

Parameters

in	*aN*	Number of data points (input).
in,out	*aLocations*	Reference to the Locations object containing X and Y coordinates (input/output).
in,out	*apZ*	Pointer to the array containing observation values (input/output).

Returns

0 if the sorting is successful.

The documentation for this class was generated from the following file:

- PredictionHelpers.hpp

8.57 exageostat::results::Results Class Reference

`#include <Results.hpp>`

Public Member Functions

- void SetIsSynthetic (bool aIsSynthetic)

  Set the flag indicating whether the results are synthetic or not.
- void SetGeneratedLocationsNumber (int aNumLocations)

  Set the number of generated locations.

- void SetIsLogger (bool aIsLogger)

  Set the flag indicating whether the logger is active or not.
- void SetLoggerPath (const std::string &aLoggerPath)

  Set the path for the logger.
- void SetTotalDataGenerationExecutionTime (double aTime)

  Set the Total Data Generation execution time.
- void SetTotalDataGenerationFlops (double aFlops)

  Set the Data Generation floating-point operations (FLOPs).
- void SetLogLikValue (double aLogLikValue)

  Set the log-likelihood value.
- void SetMLEIterations (int aIterationsNumber)

  Set the number of maximum likelihood estimation (MLE) iterations.
- void SetMaximumTheta (const std::vector< double > &aMaximumTheta)

  Set the vector of maximum theta values.
- void SetTotalModelingExecutionTime (double aTime)

  Set the total modeling execution time.
- double GetTotalModelingExecutionTime () const

  Get the total modeling execution time.
- double GetMLOE () const

  Get the MLOE.
- double GetMSPEError () const

  Get the MSPEError.
- std::vector< double > GetIDWError () const

  Get the IDW error.
- double GetMMOM () const

  Get the MMOM.
- std::vector< double > GetFisherMatrix () const

  Get the Fisher matrix elements.
- std::vector< double > GetPredictedMissedValues () const

  Get the Predicted Missed Z matrix elements.
- void SetTotalModelingFlops (double aTime)

  Set the total modeling FLOPs.
- double GetTotalModelingFlops () const

  Get the total modeling FLOPs.
- double GetAverageModelingExecutionTime () const

  Get the average modeling execution time.
- double GetAverageModelingFlops () const

  Get the average modeling FLOPs.
- void SetZMiss (int aZMiss)

  Set the value of ZMiss.
- void SetMSPEError (double aMSPEError)

  Set the value of MSPEError.
- void SetMSPEExecutionTime (double aTime)

  Set the MSPE execution time.
- void SetMSPEFlops (double aFlops)

  Set the MSPE number of floating-point operations (FLOPs).
- void SetIDWError (const std::vector< double > &aIDWError)

  Set the vector of IDW errors.
- void SetMLOE (double aMLOE)

  Set the value of MLOE.
- void SetMMOM (double aMMOM)

Set the value of MMOM.
- void SetExecutionTimeMLOEMMOM (double aTime)

  Set the MLOE-MMOM execution time.
- void SetMatrixGenerationTimeMLOEMMOM (double aTime)

  Set the MLOE-MMOM matrix generation time.
- void SetFactoTimeMLOEMMOM (double aTime)

  Set the MLOE-MMOM cholesky factorization time.
- void SetLoopTimeMLOEMMOM (double aTime)

  Set the MLOE-MMOM loop time.
- void SetFlopsMLOEMMOM (double aFlops)

  Set the MLOE-MMOM number of floating-point operations (FLOPs).
- void SetTotalFisherTime (double aTime)

  Set The total execution time of the fisher tile computation.
- void SetFisherMatrix (std::vector< double > aFisherMatrix)

  Set the elements of the fisher matrix.
- void SetPredictedMissedValues (std::vector< double > aPredictedValues)

  Set the elements of the Z missed matrix.
- void PrintEndSummary ()

  Print the end summary of the results.

Static Public Member Functions

- static Results * GetInstance ()

  Get a pointer to the singleton instance of the Results class.

Private Attributes

- bool mIsSynthetic = true

  Used is synthetic.
- int mGeneratedLocationsNumber = 0

  Used number of generated locations.
- bool mIsLogger = false

  Used is logger.
- std::string mLoggerPath

  Used logger path.
- double mExecutionTimeDataGeneration = 0

  Used Data Generation Execution time.
- double mFlopsDataGeneration = 0

  Used Data Generation flops.
- int mMLEIterations = 0

  Used MLE number of iterations.
- std::vector< double > mMaximumTheta

  Used MAX theta.
- double mLogLikValue = 0

  Used log likelihood value.
- int mZMiss = 0

  Used number of Z missed values.
- double mMSPEError = 0

  Used MSPE error.

- double mExecutionTimeMSPE

    Used Execution time.
- double mFlopsMSPE

    Used flops.
- std::vector< double > mIDWError

    Used IDW error.
- double mMLOE = 0

    Used MLOE.
- double mMMOM = 0

    USED MMOM.
- double mExecutionTimeMLOEMMOM = 0

    Used MLOE-MMOM Execution time.
- double mGenerationTimeMLOEMMOM = 0

    Used MLOE-MMOM Matrix Generation time.
- double mFactoTimeMLOEMMOM = 0

    Used MLOE-MMOM cholesky factorization time.
- double mLoopTimeMLOEMMOM = 0

    Used MLOE-MMOM loop time.
- double mFlopsMLOEMMOM = 0

    Used MLOE-MMOM flops.
- double mTotalModelingExecutionTime = 0

    Used Data Modeling Execution time.
- double mTotalModelingFlops = 0

    Used Data Modeling Number of Flops.
- double mTotalFisherTime = 0

    Used Total Fisher Time.
- std::vector< double > mFisherMatrix

    Fisher matrix.
- std::vector< double > mPredictedMissedValues

    Z miss values.

Static Private Attributes

- static Results * mpInstance

    Pointer to the singleton instance of the SyntheticGenerator class.

8.57.1 Member Function Documentation

8.57.1.1 GetInstance()

```
static Results* exageostat::results::Results::GetInstance ( )  [static]
```

Get a pointer to the singleton instance of the Results class.

Returns

A pointer to the instance of the Results class.

8.57.1.2 SetIsSynthetic()

```
void exageostat::results::Results::SetIsSynthetic (
            bool aIsSynthetic )
```

Set the flag indicating whether the results are synthetic or not.

Parameters

in	*aIsSynthetic*	True if the results are synthetic, false otherwise.

8.57.1.3 SetGeneratedLocationsNumber()

```
void exageostat::results::Results::SetGeneratedLocationsNumber (
            int aNumLocations )
```

Set the number of generated locations.

Parameters

in	*aNumLocations*	The number of generated locations.

8.57.1.4 SetIsLogger()

```
void exageostat::results::Results::SetIsLogger (
            bool aIsLogger )
```

Set the flag indicating whether the logger is active or not.

Parameters

in	*aIsLogger*	True if the logger is active, false otherwise.

8.57.1.5 SetLoggerPath()

```
void exageostat::results::Results::SetLoggerPath (
            const std::string & aLoggerPath )
```

Set the path for the logger.

Parameters

in	*aLoggerPath*	The path for the logger.

8.57.1.6 SetTotalDataGenerationExecutionTime()

```
void exageostat::results::Results::SetTotalDataGenerationExecutionTime (
            double aTime )
```

Set the Total Data Generation execution time.

Parameters

in	*aTime*	The execution time.

8.57.1.7 SetTotalDataGenerationFlops()

```
void exageostat::results::Results::SetTotalDataGenerationFlops (
            double aFlops )
```

Set the Data Generation floating-point operations (FLOPs).

Parameters

in	*aFlops*	The number of FLOPs.

8.57.1.8 SetLogLikValue()

```
void exageostat::results::Results::SetLogLikValue (
            double aLogLikValue )
```

Set the log-likelihood value.

Parameters

in	*aLogLikValue*	The log-likelihood value.

8.57.1.9 SetMLEIterations()

```
void exageostat::results::Results::SetMLEIterations (
            int aIterationsNumber )
```

Set the number of maximum likelihood estimation (MLE) iterations.

Parameters

in	*aIterationsNumber*	The number of MLE iterations.

8.57.1.10 SetMaximumTheta()

```
void exageostat::results::Results::SetMaximumTheta (
            const std::vector< double > & aMaximumTheta )
```

Set the vector of maximum theta values.

Parameters

in	*aMaximumTheta*	The vector of maximum theta values.

8.57.1.11 SetTotalModelingExecutionTime()

```
void exageostat::results::Results::SetTotalModelingExecutionTime (
            double aTime )
```

Set the total modeling execution time.

Parameters

in	*aTime*	The total execution time for data modeling.

8.57.1.12 GetTotalModelingExecutionTime()

```
double exageostat::results::Results::GetTotalModelingExecutionTime ( ) const
```

Get the total modeling execution time.

Returns

The total execution time for data modeling.

8.57.1.13 GetMLOE()

```
double exageostat::results::Results::GetMLOE ( ) const
```

Get the MLOE.

Returns

The MLOE.

8.57.1.14 GetMSPEError()

```
double exageostat::results::Results::GetMSPEError ( ) const
```

Get the MSPEError.

Returns

The MSPEError.

8.57.1.15 GetIDWError()

```
std::vector<double> exageostat::results::Results::GetIDWError ( ) const
```

Get the IDW error.

Returns

The the IDW error vector.

8.57.1.16 GetMMOM()

```
double exageostat::results::Results::GetMMOM ( ) const
```

Get the MMOM.

Returns

The MMOM.

8.57.1.17 GetFisherMatrix()

```
std::vector<double> exageostat::results::Results::GetFisherMatrix ( ) const
```

Get the Fisher matrix elements.

Returns

the Fisher matrix.

8.57.1.18 GetPredictedMissedValues()

```
std::vector<double> exageostat::results::Results::GetPredictedMissedValues ( ) const
```

Get the Predicted Missed Z matrix elements.

Returns

the Z Predicted matrix.

8.57.1.19 SetTotalModelingFlops()

```
void exageostat::results::Results::SetTotalModelingFlops (
            double aTime )
```

Set the total modeling FLOPs.

Parameters

in	*aTime*	The total number of FLOPs for data modeling.

8.57.1.20 GetTotalModelingFlops()

```
double exageostat::results::Results::GetTotalModelingFlops ( ) const
```

Get the total modeling FLOPs.

Returns

The total number of FLOPs for data modeling.

8.57.1.21 GetAverageModelingExecutionTime()

```
double exageostat::results::Results::GetAverageModelingExecutionTime ( ) const
```

Get the average modeling execution time.

Returns

The average execution time for data modeling.

8.57.1.22 GetAverageModelingFlops()

```
double exageostat::results::Results::GetAverageModelingFlops ( ) const
```

Get the average modeling FLOPs.

Returns

The average number of FLOPs for data modeling.

8.57.1.23 SetZMiss()

```
void exageostat::results::Results::SetZMiss (
            int aZMiss )
```

Set the value of ZMiss.

Parameters

in	*aZMiss*	The value of ZMiss.

8.57.1.24 SetMSPEError()

```
void exageostat::results::Results::SetMSPEError (
            double aMSPEError )
```

Set the value of MSPEError.

Parameters

in	*aMSPEError*	The value of MSPEError.

8.57.1.25 SetMSPEExecutionTime()

```
void exageostat::results::Results::SetMSPEExecutionTime (
            double aTime )
```

Set the MSPE execution time.

Parameters

in	*aTime*	The execution time.

8.57.1.26 SetMSPEFlops()

```
void exageostat::results::Results::SetMSPEFlops (
            double aFlops )
```

Set the MSPE number of floating-point operations (FLOPs).

Parameters

in	*aFlops*	The number of FLOPs.

8.57.1.27 SetIDWError()

```
void exageostat::results::Results::SetIDWError (
            const std::vector< double > & aIDWError )
```

Set the vector of IDW errors.

Parameters

in	*aIDWError*	The vector of IDW errors.

8.57.1.28 SetMLOE()

```
void exageostat::results::Results::SetMLOE (
            double aMLOE )
```

Set the value of MLOE.

Parameters

in	*aMLOE*	The value of MLOE.

8.57.1.29 SetMMOM()

```
void exageostat::results::Results::SetMMOM (
            double aMMOM )
```

Set the value of MMOM.

Parameters

in	*aMMOM*	The value of MMOM.

8.57.1.30 SetExecutionTimeMLOEMMOM()

```
void exageostat::results::Results::SetExecutionTimeMLOEMMOM (
            double aTime )
```

Set the MLOE-MMOM execution time.

Parameters

in	*aTime*	The execution time.

8.57.1.31 SetMatrixGenerationTimeMLOEMMOM()

```
void exageostat::results::Results::SetMatrixGenerationTimeMLOEMMOM (
            double aTime )
```

Set the MLOE-MMOM matrix generation time.

Parameters

in	*aTime*	The execution time.

8.57.1.32 SetFactoTimeMLOEMMOM()

```
void exageostat::results::Results::SetFactoTimeMLOEMMOM (
            double aTime )
```

Set the MLOE-MMOM cholesky factorization time.

Parameters

in	*aTime*	The execution time.

8.57.1.33 SetLoopTimeMLOEMMOM()

```
void exageostat::results::Results::SetLoopTimeMLOEMMOM (
            double aTime )
```

Set the MLOE-MMOM loop time.

Parameters

in	*aTime*	The execution time.

8.57.1.34 SetFlopsMLOEMMOM()

```
void exageostat::results::Results::SetFlopsMLOEMMOM (
            double aFlops )
```

Set the MLOE-MMOM number of floating-point operations (FLOPs).

Parameters

in	*aFlops*	The number of FLOPs.

8.57.1.35 SetTotalFisherTime()

```
void exageostat::results::Results::SetTotalFisherTime (
            double aTime )
```

Set The total execution time of the fisher tile computation.

Parameters

in	*aTime*	The total execution time for fisher tile computation.

8.57.1.36 SetFisherMatrix()

```
void exageostat::results::Results::SetFisherMatrix (
            std::vector< double > aFisherMatrix )
```

Set the elements of the fisher matrix.

Parameters

aFisherMatrix	Elements of the fisher matrix.

8.57.1.37 SetPredictedMissedValues()

```
void exageostat::results::Results::SetPredictedMissedValues (
            std::vector< double > aPredictedValues )
```

Set the elements of the Z missed matrix.

Parameters

aPredictedValues	Elements of the Predicted Z missed matrix.

8.57.1.38 PrintEndSummary()

```
void exageostat::results::Results::PrintEndSummary ( )
```

Print the end summary of the results.

8.57.2 Member Data Documentation

8.57.2.1 mpInstance

```
Results* exageostat::results::Results::mpInstance  [static], [private]
```

Pointer to the singleton instance of the SyntheticGenerator class.

8.57.2.2 mIsSynthetic

```
bool exageostat::results::Results::mIsSynthetic = true  [private]
```

Used is synthetic.

8.57.2.3 mGeneratedLocationsNumber

```
int exageostat::results::Results::mGeneratedLocationsNumber = 0  [private]
```

Used number of generated locations.

8.57.2.4 mIsLogger

```
bool exageostat::results::Results::mIsLogger = false  [private]
```

Used is logger.

8.57.2.5 mLoggerPath

```
std::string exageostat::results::Results::mLoggerPath  [private]
```

Used logger path.

8.57.2.6 mExecutionTimeDataGeneration

```
double exageostat::results::Results::mExecutionTimeDataGeneration = 0  [private]
```

Used Data Generation Execution time.

8.57.2.7 mFlopsDataGeneration

```
double exageostat::results::Results::mFlopsDataGeneration = 0  [private]
```

Used Data Generation flops.

8.57.2.8 mMLEIterations

```
int exageostat::results::Results::mMLEIterations = 0  [private]
```

Used MLE number of iterations.

8.57.2.9 mMaximumTheta

```
std::vector<double> exageostat::results::Results::mMaximumTheta  [private]
```

Used MAX theta.

8.57.2.10 mLogLikValue

```
double exageostat::results::Results::mLogLikValue = 0  [private]
```

Used log likelihood value.

8.57.2.11 mZMiss

```
int exageostat::results::Results::mZMiss = 0  [private]
```

Used number of Z missed values.

8.57.2.12 mMSPEError

```
double exageostat::results::Results::mMSPEError = 0  [private]
```

Used MSPE error.

8.57.2.13 mExecutionTimeMSPE

```
double exageostat::results::Results::mExecutionTimeMSPE  [private]
```

Used Execution time.

8.57.2.14 mFlopsMSPE

`double exageostat::results::Results::mFlopsMSPE [private]`

Used flops.

8.57.2.15 mIDWError

`std::vector<double> exageostat::results::Results::mIDWError [private]`

Used IDW error.

8.57.2.16 mMLOE

`double exageostat::results::Results::mMLOE = 0 [private]`

Used MLOE.

8.57.2.17 mMMOM

`double exageostat::results::Results::mMMOM = 0 [private]`

USED MMOM.

8.57.2.18 mExecutionTimeMLOEMMOM

`double exageostat::results::Results::mExecutionTimeMLOEMMOM = 0 [private]`

Used MLOE-MMOM Execution time.

8.57.2.19 mGenerationTimeMLOEMMOM

`double exageostat::results::Results::mGenerationTimeMLOEMMOM = 0 [private]`

Used MLOE-MMOM Matrix Generation time.

8.57.2.20 mFactoTimeMLOEMMOM

```
double exageostat::results::Results::mFactoTimeMLOEMMOM = 0  [private]
```

Used MLOE-MMOM cholesky factorization time.

8.57.2.21 mLoopTimeMLOEMMOM

```
double exageostat::results::Results::mLoopTimeMLOEMMOM = 0  [private]
```

Used MLOE-MMOM loop time.

8.57.2.22 mFlopsMLOEMMOM

```
double exageostat::results::Results::mFlopsMLOEMMOM = 0  [private]
```

Used MLOE-MMOM flops.

8.57.2.23 mTotalModelingExecutionTime

```
double exageostat::results::Results::mTotalModelingExecutionTime = 0  [private]
```

Used Data Modeling Execution time.

8.57.2.24 mTotalModelingFlops

```
double exageostat::results::Results::mTotalModelingFlops = 0  [private]
```

Used Data Modeling Number of Flops.

8.57.2.25 mTotalFisherTime

```
double exageostat::results::Results::mTotalFisherTime = 0  [private]
```

Used Total Fisher Time.

8.57.2.26 mFisherMatrix

```
std::vector<double> exageostat::results::Results::mFisherMatrix [private]
```

Fisher matrix.

8.57.2.27 mPredictedMissedValues

```
std::vector<double> exageostat::results::Results::mPredictedMissedValues [private]
```

Z miss values.

The documentation for this class was generated from the following file:

- Results.hpp

8.58 exageostat::runtime::RuntimeFunctions< T > Class Template Reference

A class that defines runtime static functions.

```
#include <RuntimeFunctions.hpp>
```

Static Public Member Functions

- static void CovarianceMatrix (dataunits::DescriptorData< T > &aDescriptorData, void *apDescriptor, const int &aTriangularPart, dataunits::Locations< T > *apLocation1, dataunits::Locations< T > *apLocation2, dataunits::Locations< T > *apLocation3, T *apLocalTheta, const int &aDistanceMetric, const kernels::Kernel< T > *apKernel)

  Computes the covariance matrix.
- static void ExaGeoStatMLETileAsyncMLOEMMOM (void *apDescExpr2, void *apDescExpr3, void *apDescExpr4, void *apDescMLOE, void *apDescMMOM, void *apSequence, void *apRequest)

  Perform an asynchronous computation of MLE, MLOE, and MMOM for a tile.
- static void ExaGeoStatMLEMSPETileAsync (void *apDescZPredict, void *apDescZMiss, void *apDescError, void *apSequence, void *apRequest)

  Calculate mean square prediction error (MSPE) scalar value of the prediction.
- static void CopyDescriptorZ (dataunits::DescriptorData< T > &aDescriptorData, void *apDescriptor, T *apDoubleVector)

  Copies the descriptor data to a double vector.
- static void ExaGeoStatGaussianToNonTileAsync (dataunits::DescriptorData< T > &aDescriptorData, void *apDesc, T *apTheta)

  Converts a Gaussian descriptor to a non-tiled descriptor.
- static void ExaGeoStaStrideVectorTileAsync (void *apDescA, void *apDescB, void *apDescC, void *apSequence, void *apRequest)

  copy Chameleon descriptor to vector float.*
- static void ExaGeoStaStrideVectorTileAsync (void *apDescA, void *apDescB, void *apDescC, void *apDescD, void *apSequence, void *apRequest)

Copy Chameleon descriptor to vector float∗.

- static void ExaGeoStatMeasureDetTileAsync (const common::Computation &aComputation, void ∗apDescA, void ∗apSequence, void ∗apRequest, void ∗apDescDet)

  Calculate determinant for triangular matrix.
- static void ExaGeoStatMLETraceTileAsync (void ∗apDescA, void ∗apSequence, void ∗apRequest, void ∗apDescNum, void ∗apDescTrace)

  Calculate determinant for triangular matrix.
- static void ExaGeoStatDoubleDotProduct (void ∗apDescA, void ∗apDescProduct, void ∗apSequence, void ∗apRequest)

  Computes dot product of A.A.
- static void ExaGeoStatMLEMSPEBivariateTileAsync (void ∗apDescZPre, void ∗apDescZMiss, void ∗apDescError1, void ∗apDescError2, void ∗apDescError, void ∗apSequence, void ∗apRequest)

  Calculate mean square error (MSE) scalar value for Bivariate kernels.
- static void ExaGeoStatNonGaussianLogLikeTileAsync (const common::Computation &aComputation, void ∗apDescZ, void ∗apDescSum, const T ∗apTheta, void ∗apSequence, void ∗apRequest)

  Calculate the log likelihood of non-Gaussian MLE.
- static void ExaGeoStatNonGaussianTransformTileAsync (const common::Computation &aComputation, void ∗apDescZ, const T ∗apTheta, void ∗apSequence, void ∗apRequest)

  Transform the measurements vector inside the non-Gaussian MLE function.

8.58.1 Detailed Description

template<typename T>
class exageostat::runtime::RuntimeFunctions< T >

A class that defines runtime static functions.

Template Parameters

T	Data Type: float or double.

8.58.2 Member Function Documentation

8.58.2.1 CovarianceMatrix()

```
template<typename T >
static void exageostat::runtime::RuntimeFunctions< T >::CovarianceMatrix (
            dataunits::DescriptorData< T > & aDescriptorData,
            void * apDescriptor,
            const int & aTriangularPart,
            dataunits::Locations< T > * apLocation1,
            dataunits::Locations< T > * apLocation2,
            dataunits::Locations< T > * apLocation3,
            T * apLocalTheta,
            const int & aDistanceMetric,
            const kernels::Kernel< T > * apKernel )  [static]
```

Computes the covariance matrix.

Parameters

in	*aDescriptorData*	pointer to the DescriptorData object holding descriptors and data.
out	*apDescriptor*	Pointer to the descriptor for the covariance matrix.
in	*aTriangularPart*	Specifies whether the upper or lower triangular part of the covariance matrix is stored.
in	*apLocation1*	Pointer to the first set of locations.
in	*apLocation2*	Pointer to the second set of locations.
in	*apLocation3*	Pointer to the third set of locations.
in	*apLocalTheta*	Pointer to the local theta values.
in	*aDistanceMetric*	Specifies the distance metric to use.
in	*apKernel*	Pointer to the kernel object to use.

Returns

void

8.58.2.2 ExaGeoStatMLETileAsyncMLOEMMOM()

```
template<typename T >
static void exageostat::runtime::RuntimeFunctions< T >::ExaGeoStatMLETileAsyncMLOEMMOM (
            void * apDescExpr2,
            void * apDescExpr3,
            void * apDescExpr4,
            void * apDescMLOE,
            void * apDescMMOM,
            void * apSequence,
            void * apRequest )  [static]
```

Perform an asynchronous computation of MLE, MLOE, and MMOM for a tile.

his function performs the computation of Maximum Likelihood Estimation (MLE), Maximum Likelihood on the Empirical Orthogonal Functions (MLOE), and Method of Moments (MMOM) for a tile asynchronously.

Parameters

in	*apDescExpr2*	Descriptor for expression 2.
in	*apDescExpr3*	Descriptor for expression 3.
in	*apDescExpr4*	Descriptor for expression 4.
in	*apDescMLOE*	Descriptor for MLOE.
in	*apDescMMOM*	Descriptor for MMOM.
in	*apSequence*	Sequence for the computation.
in	*apRequest*	Request for the computation.

Returns

void

8.58.2.3 ExaGeoStatMLEMSPETileAsync()

```
template<typename T >
static void exageostat::runtime::RuntimeFunctions< T >::ExaGeoStatMLEMSPETileAsync (
            void * apDescZPredict,
            void * apDescZMiss,
            void * apDescError,
            void * apSequence,
            void * apRequest )  [static]
```

Calculate mean square prediction error (MSPE) scalar value of the prediction.

Parameters

in	*apDescZPredict*	Observed measurements.
in	*apDescZMiss*	Missing measurements.
out	*apDescError*	Mean Square Prediction Error (MSPE).
in	*apSequence*	Identifies the sequence of function calls that this call belongs to.
out	*apRequest*	Identifies this function call (for exception handling purposes).

Returns

void

8.58.2.4 CopyDescriptorZ()

```
template<typename T >
static void exageostat::runtime::RuntimeFunctions< T >::CopyDescriptorZ (
            dataunits::DescriptorData< T > & aDescriptorData,
            void * apDescriptor,
            T * apDoubleVector )  [static]
```

Copies the descriptor data to a double vector.

Parameters

in	*aComputation*	computation used in configuration.
in	*aDescriptorData*	pointer to the DescriptorData object holding descriptors and data.
in	*apDescriptor*	Pointer to the descriptor data.
in,out	*apDoubleVector*	Pointer to the double vector to copy the descriptor data to.

Returns

void

8.58.2.5 ExaGeoStatGaussianToNonTileAsync()

```
template<typename T >
static void exageostat::runtime::RuntimeFunctions< T >::ExaGeoStatGaussianToNonTileAsync (
            datаunits::DescriptorData< T > & aDescriptorData,
            void * apDesc,
            T * apTheta )  [static]
```

Converts a Gaussian descriptor to a non-tiled descriptor.

Parameters

in	*aDescriptorData*	DescriptorData struct with the Gaussian descriptor.
in	*apDesc*	Pointer to the non-tiled descriptor.
in	*apTheta*	Theta vector.

Returns

void

8.58.2.6 ExaGeoStaStrideVectorTileAsync() [1/2]

```
template<typename T >
static void exageostat::runtime::RuntimeFunctions< T >::ExaGeoStaStrideVectorTileAsync (
            void * apDescA,
            void * apDescB,
            void * apDescC,
            void * apSequence,
            void * apRequest )  [static]
```

copy Chameleon descriptor to vector float*.

Parameters

in	*apDescA*	Exageostat descriptor A.
in	*apDescB*	Exageostat descriptor B.
in	*apDescC*	Exageostat descriptor C.
in	*apSequence*	Identifies the sequence of function calls that this call belongs to.
in	*apRequest*	Identifies this function call (for exception handling purposes).

Returns

void

8.58.2.7 ExaGeoStaStrideVectorTileAsync() [2/2]

```
template<typename T >
static void exageostat::runtime::RuntimeFunctions< T >::ExaGeoStaStrideVectorTileAsync (
            void * apDescA,
            void * apDescB,
            void * apDescC,
            void * apDescD,
            void * apSequence,
            void * apRequest ) [static]
```

Copy Chameleon descriptor to vector float*.

Parameters

in	*apDescA*	Exageostat descriptor A.
in	*apDescB*	Exageostat descriptor B.
in	*apDescC*	Exageostat descriptor C.
in	*apDescD*	Exageostat descriptor D.
in	*apSequence*	Identifies the sequence of function calls that this call belongs to.
in	*apRequest*	Identifies this function call (for exception handling purposes).

Returns

void

8.58.2.8 ExaGeoStatMeasureDetTileAsync()

```
template<typename T >
static void exageostat::runtime::RuntimeFunctions< T >::ExaGeoStatMeasureDetTileAsync (
            const common::Computation & aComputation,
            void * apDescA,
            void * apSequence,
            void * apRequest,
            void * apDescDet ) [static]
```

Calculate determinant for triangular matrix.

Parameters

in	*aComputation*	computation used in configuration.
in	*apDescA*	Exageostat descriptor.
in	*apSequence*	Identifies the sequence of function calls that this call belongs to.
in	*apRequest*	Identifies this function call (for exception handling purposes).
in	*apDescDet*	determinant value

Returns

void

8.58.2.9 ExaGeoStatMLETraceTileAsync()

```
template<typename T >
static void exageostat::runtime::RuntimeFunctions< T >::ExaGeoStatMLETraceTileAsync (
            void * apDescA,
            void * apSequence,
            void * apRequest,
            void * apDescNum,
            void * apDescTrace )  [static]
```

Calculate determinant for triangular matrix.

Parameters

in	*apDescA*	Pointer to the descriptor of the matrix 'descA'.
in	*apSequence*	Pointer to a sequence structure for managing asynchronous execution.
in	*apRequest*	Pointer to a request structure for tracking the operation's status.
out	*apDescNum*	Pointer to the descriptor of the matrix to store the sum of elements.
out	*apDescTrace*	Pointer to the descriptor of the matrix to store the trace.

Returns

void

8.58.2.10 ExaGeoStatDoubleDotProduct()

```
template<typename T >
static void exageostat::runtime::RuntimeFunctions< T >::ExaGeoStatDoubleDotProduct (
            void * apDescA,
            void * apDescProduct,
            void * apSequence,
            void * apRequest )  [static]
```

Computes dot product of A.A.

Parameters

in	*apDescA*	A Descriptor
out	*apDescProduct*	Stores the result of A.A.
in	*apSequence*	Identifies the sequence of function calls that this call belongs to.
in	*apRequest*	Identifies this function call (for exception handling purposes).

Returns

void

8.58.2.11 ExaGeoStatMLEMSPEBivariateTileAsync()

```
template<typename T >
static void exageostat::runtime::RuntimeFunctions< T >::ExaGeoStatMLEMSPEBivariateTileAsync (
            void * apDescZPre,
            void * apDescZMiss,
            void * apDescError1,
            void * apDescError2,
            void * apDescError,
            void * apSequence,
            void * apRequest ) [static]
```

Calculate mean square error (MSE) scalar value for Bivariate kernels.

Parameters

in	*apDescZPre*	Observed measurements descZpre.
in	*apDescZMiss*	Missing measurements descZpre
out	*apDescError1*	Mean Square Error (MSE) 1.
out	*apDescError2*	Mean Square Error (MSE) 2.
out	*apDescError*	Mean Square Error (MSE).
in	*apSequence*	Sequence for the computation.
in	*apRequest*	Request for the computation.

Returns

void

8.58.2.12 ExaGeoStatNonGaussianLogLikeTileAsync()

```
template<typename T >
static void exageostat::runtime::RuntimeFunctions< T >::ExaGeoStatNonGaussianLogLikeTileAsync
(
            const common::Computation & aComputation,
            void * apDescZ,
            void * apDescSum,
            const T * apTheta,
            void * apSequence,
            void * apRequest ) [static]
```

Calculate the log likelihood of non-Gaussian MLE.

Parameters

in	*aComputation*	computation used in configuration.
in	*apDescZ*	pointer to the Observed Measurements descriptor.
in	*apDescSum*	The log-likelihood Sum of descriptor Z.
in	*apTheta*	Pointer to Model parameters.
in	*apSequence*	Identifies the sequence of function calls that this call belongs to.
out	*apRequest*	Identifies this function call (for exception handling purposes).

Returns

void

8.58.2.13 ExaGeoStatNonGaussianTransformTileAsync()

```
template<typename T >
static void exageostat::runtime::RuntimeFunctions< T >::ExaGeoStatNonGaussianTransformTile←
Async (
            const common::Computation & aComputation,
            void * apDescZ,
            const T * apTheta,
            void * apSequence,
            void * apRequest )  [static]
```

Transform the measurements vector inside the non-Gaussian MLE function.

Parameters

in	*aComputation*	computation used in configuration.
in	*apDescZ*	pointer to the Observed Measurements descriptor.
in	*apTheta*	Pointer to Model parameters.
in	*apSequence*	Identifies the sequence of function calls that this call belongs to.
in	*apRequest*	Identifies this function call (for exception handling purposes).

Returns

void

The documentation for this class was generated from the following file:

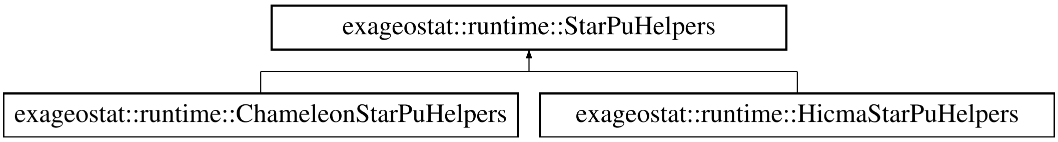
- RuntimeFunctions.hpp

8.59 exageostat::runtime::StarPuHelpers Class Reference

A class that defines the interface for StarPu helpers.

```
#include <StarPuHelpers.hpp>
```

Inheritance diagram for exageostat::runtime::StarPuHelpers:



Public Member Functions

- virtual void ExaGeoStatOptionsInit (void ∗apOptions, void ∗apSequence, void ∗apRequest)=0

  Initialize the runtime option structure for either HiCMA or CHAMELEON.
- virtual void ExaGeoStatOptionsFree (void ∗apOptions)=0

  Submit the release of the workspaces associated to the options structure.
- virtual void ExaGeoStatOptionsFinalize (void ∗apOptions)=0

  Finalize the runtime option structure for either HiCMA or CHAMELEON.
- virtual void ∗ ExaGeoStatDataGetAddr (void ∗apDescriptor, const int &aDescRow, const int &aDescCol)=0

  Get the pointer to the data or the runtime handler associated to the piece of data (m, n) in desc.
- virtual int GetMT (void ∗apDescriptor)=0

  Get the number of tile rows of the sub-matrix.
- virtual int GetM (void ∗apDescriptor)=0

  Get the descriptor number of rows.
- virtual int GetMB (void ∗apDescriptor)=0

  Get the descriptor number of rows in a tile.
- virtual void ∗ GetOptions ()=0

  Get the descriptor options.
- virtual void DeleteOptions (void ∗apOptions)=0

  Delete the options object.

8.59.1 Detailed Description

A class that defines the interface for StarPu helpers.

Template Parameters

T	Data Type: float or double.

8.59.2 Member Function Documentation

8.59.2.1 ExaGeoStatOptionsInit()

```
virtual void exageostat::runtime::StarPuHelpers::ExaGeoStatOptionsInit (
            void * apOptions,
```

```
            void * apSequence,
            void * apRequest )  [pure virtual]
```

Initialize the runtime option structure for either HiCMA or CHAMELEON.

Parameters

in,out	*apOptions*	The options structure that needs to be initialized.
in	*apSequence*	The sequence structure to associate in the options.
in	*apRequest*	The request structure to associate in the options.

Returns

void

Implemented in exageostat::runtime::HicmaStarPuHelpers, and exageostat::runtime::ChameleonStarPuHelpers.

8.59.2.2 ExaGeoStatOptionsFree()

```
virtual void exageostat::runtime::StarPuHelpers::ExaGeoStatOptionsFree (
            void * apOptions )  [pure virtual]
```

Submit the release of the workspaces associated to the options structure.

Parameters

in,out	*apOptions*	The options structure for which to workspaces will be released

Returns

void

Implemented in exageostat::runtime::HicmaStarPuHelpers, and exageostat::runtime::ChameleonStarPuHelpers.

8.59.2.3 ExaGeoStatOptionsFinalize()

```
virtual void exageostat::runtime::StarPuHelpers::ExaGeoStatOptionsFinalize (
            void * apOptions )  [pure virtual]
```

Finalize the runtime option structure for either HiCMA or CHAMELEON.

Parameters

in,out	*apOptions*	The options structure that needs to be finalized.

Returns

void

Implemented in exageostat::runtime::HicmaStarPuHelpers, and exageostat::runtime::ChameleonStarPuHelpers.

8.59.2.4 ExaGeoStatDataGetAddr()

```
virtual void* exageostat::runtime::StarPuHelpers::ExaGeoStatDataGetAddr (
            void * apDescriptor,
            const int & aDescRow,
            const int & aDescCol )  [pure virtual]
```

Get the pointer to the data or the runtime handler associated to the piece of data (m, n) in desc.

Parameters

in	*apDescriptor*	The descriptor to which belongs the piece of data
in	*aDescRow*	The row coordinate of the piece of data in the matrix
in	*aDescCol*	The column coordinate of the piece of data in the matrix

Returns

void

Implemented in exageostat::runtime::HicmaStarPuHelpers, and exageostat::runtime::ChameleonStarPuHelpers.

8.59.2.5 GetMT()

```
virtual int exageostat::runtime::StarPuHelpers::GetMT (
            void * apDescriptor )  [pure virtual]
```

Get the number of tile rows of the sub-matrix.

Parameters

in	*apDescriptor*	

Returns

int

Implemented in exageostat::runtime::HicmaStarPuHelpers, and exageostat::runtime::ChameleonStarPuHelpers.

8.59.2.6 GetM()

```
virtual int exageostat::runtime::StarPuHelpers::GetM (
            void * apDescriptor )  [pure virtual]
```

Get the descriptor number of rows.

Parameters

in	*apDescriptor*	

Returns

int

Implemented in exageostat::runtime::HicmaStarPuHelpers, and exageostat::runtime::ChameleonStarPuHelpers.

8.59.2.7 GetMB()

```
virtual int exageostat::runtime::StarPuHelpers::GetMB (
            void * apDescriptor )  [pure virtual]
```

Get the descriptor number of rows in a tile.

Parameters

in	*apDescriptor*	

Returns

int

Implemented in exageostat::runtime::HicmaStarPuHelpers, and exageostat::runtime::ChameleonStarPuHelpers.

8.59.2.8 GetOptions()

```
virtual void* exageostat::runtime::StarPuHelpers::GetOptions ( )  [pure virtual]
```

Get the descriptor options.

Returns

void pointer to descriptor_option

void

Implemented in exageostat::runtime::HicmaStarPuHelpers, and exageostat::runtime::ChameleonStarPuHelpers.

8.59.2.9 DeleteOptions()

```
virtual void exageostat::runtime::StarPuHelpers::DeleteOptions (
            void * apOptions )  [pure virtual]
```

Delete the options object.

Parameters

apOptions	

Returns

void

Implemented in exageostat::runtime::HicmaStarPuHelpers, and exageostat::runtime::ChameleonStarPuHelpers.

The documentation for this class was generated from the following file:

- StarPuHelpers.hpp

8.60 exageostat::runtime::StarPuHelpersFactory Class Reference

A class that creates StarPu helpers based on the input computation type.

`#include <StarPuHelpersFactory.hpp>`

Static Public Member Functions

- static std::unique_ptr< StarPuHelpers > CreateStarPuHelper (const common::Computation &aComputation)
  Creates a StarPu helper.

8.60.1 Detailed Description

A class that creates StarPu helpers based on the input computation type.

8.60.2 Member Function Documentation

8.60.2.1 CreateStarPuHelper()

```
static std::unique_ptr<StarPuHelpers> exageostat::runtime::StarPuHelpersFactory::CreateStar←
PuHelper (
            const common::Computation & aComputation )  [static]
```

Creates a StarPu helper.

Parameters

in	*aComputation*	The computation type to create the solver for.

Returns

Unique pointer to the created StarPu helper.

The documentation for this class was generated from the following file:

- StarPuHelpersFactory.hpp

8.61 stride Class Reference

A class for starpu codelet stride-vec.

8.61.1 Detailed Description

A class for starpu codelet stride-vec.

Template Parameters

T	Data Type: float or double

This class encapsulates the struct cl_stride_vec and its CPU functions.

The documentation for this class was generated from the following file:

- stride-vec-codelet.hpp

8.62 exageostat::runtime::STRIDEVECCodelet< T > Class Template Reference

```
#include <stride-vec-codelet.hpp>
```

Public Member Functions

- STRIDEVECCodelet ()=default
  Default constructor.
- ∼STRIDEVECCodelet ()=default
  Default destructor.
- void InsertTask (const void ∗apDescA, void ∗apDescB, void ∗apDescC)
  Inserts a task for STRIDE vector operation codelet processing.

Static Private Member Functions

- static void cl_stride_vec_function (void **apBuffers, void *apCodeletArguments)

  Executes the STRIDE vector operation codelet function.

Static Private Attributes

- static struct starpu_codelet cl_stride_vec

  starpu_codelet struct

8.62.1 Constructor & Destructor Documentation

8.62.1.1 STRIDEVECCodelet()

```
template<typename T >
exageostat::runtime::STRIDEVECCodelet< T >::STRIDEVECCodelet ( )  [default]
```

Default constructor.

8.62.1.2 ~STRIDEVECCodelet()

```
template<typename T >
exageostat::runtime::STRIDEVECCodelet< T >::~STRIDEVECCodelet ( )  [default]
```

Default destructor.

8.62.2 Member Function Documentation

8.62.2.1 InsertTask()

```
template<typename T >
void exageostat::runtime::STRIDEVECCodelet< T >::InsertTask (
            const void * apDescA,
            void * apDescB,
            void * apDescC )
```

Inserts a task for STRIDE vector operation codelet processing.

Parameters

in	*apDescA*	A pointer to the descriptor for the source vector.
in,out	*apDescB*	A pointer to the descriptor for the first destination vector.
in,out	*apDescC*	A pointer to the descriptor for the second destination vector.

Returns

void

8.62.2.2 cl_stride_vec_function()

```
template<typename T >
static void exageostat::runtime::STRIDEVECCodelet< T >::cl_stride_vec_function (
            void ** apBuffers,
            void * apCodeletArguments )  [static], [private]
```

Executes the STRIDE vector operation codelet function.

Parameters

in	*apBuffers*	An array of pointers to the buffers.
in	*apCodeletArguments*	A pointer to the codelet arguments structure, which includes the vector size, offset, and the stride factor.

Returns

void

8.62.3 Member Data Documentation

8.62.3.1 cl_stride_vec

```
template<typename T >
struct starpu_codelet exageostat::runtime::STRIDEVECCodelet< T >::cl_stride_vec  [static],
[private]
```

starpu_codelet struct

The documentation for this class was generated from the following file:

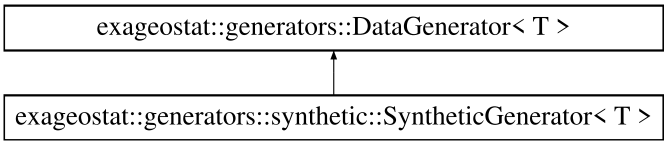
- stride-vec-codelet.hpp

8.63 exageostat::generators::synthetic::SyntheticGenerator< T > Class Template Reference

A class for generating synthetic data.

```
#include <SyntheticGenerator.hpp>
```

Inheritance diagram for exageostat::generators::synthetic::SyntheticGenerator< T >:



Public Member Functions

- std::unique_ptr< ExaGeoStatData< T > > CreateData (configurations::Configurations &aConfigurations, exageostat::kernels::Kernel< T > &aKernel) override

    Creates the data by synthetically generating it.

Static Public Member Functions

- static SyntheticGenerator< T > * GetInstance ()

    Get a pointer to the singleton instance of the SyntheticGenerator class.
- static void ReleaseInstance ()

    Release the singleton instance of the SyntheticGenerator class.

Private Member Functions

- SyntheticGenerator ()=default

    Constructor for the SyntheticGenerator class.
- ~SyntheticGenerator () override=default

    Default destructor.

Static Private Attributes

- static SyntheticGenerator< T > * mpInstance

    Pointer to the singleton instance of the SyntheticGenerator class.

Additional Inherited Members

8.63.1 Detailed Description

template< typename T >
class exageostat::generators::synthetic::SyntheticGenerator< T >

A class for generating synthetic data.

Template Parameters

T	Data Type: float or double

This class generates synthetic data for use in testing machine learning models.

8.63.2 Constructor & Destructor Documentation

8.63.2.1 SyntheticGenerator()

```
template<typename T >
exageostat::generators::synthetic::SyntheticGenerator< T >::SyntheticGenerator ( ) [private],
[default]
```

Constructor for the SyntheticGenerator class.

Returns

void

8.63.2.2 ~SyntheticGenerator()

```
template<typename T >
exageostat::generators::synthetic::SyntheticGenerator< T >::~SyntheticGenerator ( ) [override],
[private], [default]
```

Default destructor.

8.63.3 Member Function Documentation

8.63.3.1 GetInstance()

```
template<typename T >
static SyntheticGenerator<T>* exageostat::generators::synthetic::SyntheticGenerator< T >↩
::GetInstance ( ) [static]
```

Get a pointer to the singleton instance of the SyntheticGenerator class.

Returns

A pointer to the instance of the SyntheticGenerator class.

8.63.3.2 CreateData()

```
template<typename T >
std::unique_ptr<ExaGeoStatData<T> > exageostat::generators::synthetic::SyntheticGenerator< T
>::CreateData (
            configurations::Configurations & aConfigurations,
            exageostat::kernels::Kernel< T > & aKernel ) [override], [virtual]
```

Creates the data by synthetically generating it.

Either generates synthetic data or reads data files. This method generates the X, Y, and Z variables used to define the locations of the data points.

Parameters

in	*aConfigurations*	Reference to the data configurations.
in	*aKernel*	Reference to the used Kernel.

Returns

unique Pointer to a populated data.

Implements exageostat::generators::DataGenerator< T >.

8.63.3.3 ReleaseInstance()

```
template<typename T >
static void exageostat::generators::synthetic::SyntheticGenerator< T >::ReleaseInstance ( )
[static]
```

Release the singleton instance of the SyntheticGenerator class.

Returns

void

8.63.4 Member Data Documentation

8.63.4.1 mpInstance

```
template<typename T >
SyntheticGenerator<T>* exageostat::generators::synthetic::SyntheticGenerator< T >::mpInstance
[static], [private]
```

Pointer to the singleton instance of the SyntheticGenerator class.

The documentation for this class was generated from the following file:

- SyntheticGenerator.hpp

8.64 exageostat::runtime::TriStrideVecCodelet< T > Class Template Reference

A class for starpu codelet tri_stride_vec.

`#include <tri-stride-vec-codelet.hpp>`

Public Member Functions

- TriStrideVecCodelet ()=default
  Default constructor.
- ~TriStrideVecCodelet ()=default
  Default destructor.
- void InsertTask (const void *apDescA, void *apDescB, void *apDescC, void *apDescD)
  Inserts a task for TriStride vector operation codelet processing.

Static Private Member Functions

- static void cl_tri_stride_vec_function (void **apBuffers, void *apCodeletArguments)
  Executes the TriStride vector operation codelet function.

Static Private Attributes

- static struct starpu_codelet cl_tri_stride_vec
  starpu_codelet struct

8.64.1 Detailed Description

template<typename T>
class exageostat::runtime::TriStrideVecCodelet< T >

A class for starpu codelet tri_stride_vec.

Template Parameters

T	Data Type: float or double

This class encapsulates the struct cl_tri_stride_vec and its CPU functions.

8.64.2 Constructor & Destructor Documentation

8.64.2.1 TriStrideVecCodelet()

```
template<typename T >
exageostat::runtime::TriStrideVecCodelet< T >::TriStrideVecCodelet ( ) [default]
```

Default constructor.

8.64.2.2 ∼TriStrideVecCodelet()

```
template<typename T >
exageostat::runtime::TriStrideVecCodelet< T >::~TriStrideVecCodelet ( ) [default]
```

Default destructor.

8.64.3 Member Function Documentation

8.64.3.1 InsertTask()

```
template<typename T >
void exageostat::runtime::TriStrideVecCodelet< T >::InsertTask (
            const void * apDescA,
            void * apDescB,
            void * apDescC,
            void * apDescD )
```

Inserts a task for TriStride vector operation codelet processing.

Parameters

in	*apDescA*	A pointer to the descriptor for the source vector.
in,out	*apDescB*	A pointer to the descriptor for the first destination vector.
in,out	*apDescC*	A pointer to the descriptor for the second destination vector.
in,out	*apDescD*	A pointer to the descriptor for the third destination vector.

Returns

void

8.64.3.2 cl_tri_stride_vec_function()

```
template<typename T >
static void exageostat::runtime::TriStrideVecCodelet< T >::cl_tri_stride_vec_function (
```

```
void ** apBuffers,
void * apCodeletArguments ) [static], [private]
```

Executes the TriStride vector operation codelet function.

Parameters

in	*apBuffers*	An array of pointers to the buffers.
in	*apCodeletArguments*	A pointer to the codelet arguments structure, which includes the vector size, offset, and the stride factor.

Returns

void

8.64.4 Member Data Documentation

8.64.4.1 cl_tri_stride_vec

```
template<typename T >
struct starpu_codelet exageostat::runtime::TriStrideVecCodelet< T >::cl_tri_stride_vec [static],
[private]
```

starpu_codelet struct

The documentation for this class was generated from the following file:

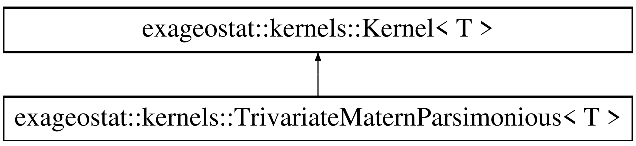
- tri-stride-vec-codelet.hpp

8.65 exageostat::kernels::TrivariateMaternParsimonious< T > Class Template Reference

A class represents a Trivariate Matern Parsimonious kernel.

```
#include <TrivariateMaternParsimonious.hpp>
```

Inheritance diagram for exageostat::kernels::TrivariateMaternParsimonious< T >:



Public Member Functions

- TrivariateMaternParsimonious ()

    Constructs a new TrivariateMaternParsimonious object.
- ∼TrivariateMaternParsimonious () override=default

    Virtual destructor to allow calls to the correct concrete destructor.
- void GenerateCovarianceMatrix (T ∗apMatrixA, const int &aRowsNumber, const int &aColumnsNumber, const int &aRowOffset, const int &aColumnOffset, dataunits::Locations< T > &aLocation1, dataunits::Locations< T > &aLocation2, dataunits::Locations< T > &aLocation3, T ∗apLocalTheta, const int &aDistanceMetric) override

    Generates a covariance matrix using a set of locations and kernel parameters.

Static Public Member Functions

- static Kernel< T > ∗ Create ()

    Creates a new TrivariateMaternParsimonious object.

Static Private Attributes

- static bool plugin_name

Additional Inherited Members

8.65.1 Detailed Description

template<typename T>
class exageostat::kernels::TrivariateMaternParsimonious< T >

A class represents a Trivariate Matern Parsimonious kernel.

This class represents a Trivariate Matern Parsimonious, which is a subclass of the Kernel class. It provides a method for generating a covariance matrix using a set of input locations and kernel parameters.

8.65.2 Constructor & Destructor Documentation

8.65.2.1 TrivariateMaternParsimonious()

```
template<typename T >
exageostat::kernels::TrivariateMaternParsimonious< T >::TrivariateMaternParsimonious ( )
```

Constructs a new TrivariateMaternParsimonious object.

Initializes a new TrivariateMaternParsimonious object with default values.

8.65.2.2 ∼TrivariateMaternParsimonious()

```
template<typename T >
exageostat::kernels::TrivariateMaternParsimonious< T >::∼TrivariateMaternParsimonious ( )
[override], [default]
```

Virtual destructor to allow calls to the correct concrete destructor.

8.65.3 Member Function Documentation

8.65.3.1 GenerateCovarianceMatrix()

```
template<typename T >
void exageostat::kernels::TrivariateMaternParsimonious< T >::GenerateCovarianceMatrix (
            T * apMatrixA,
            const int & aRowsNumber,
            const int & aColumnsNumber,
            const int & aRowOffset,
            const int & aColumnOffset,
            dataunits::Locations< T > & aLocation1,
            dataunits::Locations< T > & aLocation2,
            dataunits::Locations< T > & aLocation3,
            T * apLocalTheta,
            const int & aDistanceMetric )  [override], [virtual]
```

Generates a covariance matrix using a set of locations and kernel parameters.

Generates a covariance matrix using a set of locations and kernel parameters.

Parameters

out	*apMatrixA*	The output covariance matrix.
in	*aRowsNumber*	The number of rows in the output matrix.
in	*aColumnsNumber*	The number of columns in the output matrix.
in	*aRowOffset*	The row offset for the input locations.
in	*aColumnOffset*	The column offset for the input locations.
in	*apLocation1*	The set of input locations 1.
in	*apLocation2*	The set of input locations 2.
in	*apLocation3*	The set of input locations 3.
in	*aLocalTheta*	An array of kernel parameters.
in	*aDistanceMetric*	Distance metric to be used (1 = Euclidean, 2 = Manhattan, 3 = Minkowski).

Returns

void

Implements exageostat::kernels::Kernel< T >.

8.65.3.2 Create()

```
template<typename T >
static Kernel<T>* exageostat::kernels::TrivariateMaternParsimonious< T >::Create ( ) [static]
```

Creates a new TrivariateMaternParsimonious object.

This method creates a new TrivariateMaternParsimonious object and returns a pointer to it.

Returns

A pointer to the new TrivariateMaternParsimonious object.

8.65.4 Member Data Documentation

8.65.4.1 plugin_name

```
template<typename T >
bool exageostat::kernels::TrivariateMaternParsimonious< T >::plugin_name [static], [private]
```

The documentation for this class was generated from the following file:

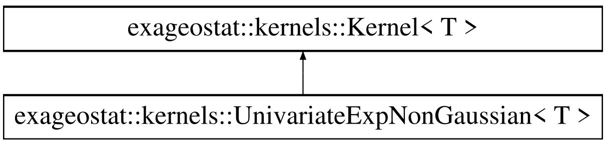
- TrivariateMaternParsimonious.hpp

8.66 exageostat::kernels::UnivariateExpNonGaussian< T > Class Template Reference

A class represents a Univariate Exp Non Gaussian kernel.

```
#include <UnivariateExpNonGaussian.hpp>
```

Inheritance diagram for exageostat::kernels::UnivariateExpNonGaussian< T >:



Public Member Functions

- UnivariateExpNonGaussian ()

  Constructs a new UnivariateExpNonGaussian object.
- ∼UnivariateExpNonGaussian () override=default

  Virtual destructor to allow calls to the correct concrete destructor.
- void GenerateCovarianceMatrix (T ∗apMatrixA, const int &aRowsNumber, const int &aColumns←Number, const int &aRowOffset, const int &aColumnOffset, dataunits::Locations< T > &aLocation1, dataunits::Locations< T > &aLocation2, dataunits::Locations< T > &aLocation3, T ∗apLocalTheta, const int &aDistanceMetric) override

  Generates a covariance matrix using a set of locations and kernel parameters.

Static Public Member Functions

- static Kernel< T > * Create ()

  Creates a new UnivariateExpNonGaussian object.

Static Private Attributes

- static bool plugin_name

Additional Inherited Members

8.66.1 Detailed Description

template<typename T>
class exageostat::kernels::UnivariateExpNonGaussian< T >

A class represents a Univariate Exp Non Gaussian kernel.

This class represents a Univariate Exp Non Gaussian, which is a subclass of the Kernel class. It provides a method for generating a covariance matrix using a set of input locations and kernel parameters.

8.66.2 Constructor & Destructor Documentation

8.66.2.1 UnivariateExpNonGaussian()

```
template<typename T >
exageostat::kernels::UnivariateExpNonGaussian< T >::UnivariateExpNonGaussian ( )
```

Constructs a new UnivariateExpNonGaussian object.

Initializes a new UnivariateExpNonGaussian object with default values.

8.66.2.2 ∼UnivariateExpNonGaussian()

```
template<typename T >
exageostat::kernels::UnivariateExpNonGaussian< T >::~UnivariateExpNonGaussian ( ) [override],
[default]
```

Virtual destructor to allow calls to the correct concrete destructor.

8.66.3 Member Function Documentation

8.66.3.1 GenerateCovarianceMatrix()

```
template<typename T >
void exageostat::kernels::UnivariateExpNonGaussian< T >::GenerateCovarianceMatrix (
            T * apMatrixA,
            const int & aRowsNumber,
            const int & aColumnsNumber,
            const int & aRowOffset,
            const int & aColumnOffset,
            dataunits::Locations< T > & aLocation1,
            dataunits::Locations< T > & aLocation2,
            dataunits::Locations< T > & aLocation3,
            T * apLocalTheta,
            const int & aDistanceMetric )  [override], [virtual]
```

Generates a covariance matrix using a set of locations and kernel parameters.

Generates a covariance matrix using a set of locations and kernel parameters.

Parameters

out	*apMatrixA*	The output covariance matrix.
in	*aRowsNumber*	The number of rows in the output matrix.
in	*aColumnsNumber*	The number of columns in the output matrix.
in	*aRowOffset*	The row offset for the input locations.
in	*aColumnOffset*	The column offset for the input locations.
in	*apLocation1*	The set of input locations 1.
in	*apLocation2*	The set of input locations 2.
in	*apLocation3*	The set of input locations 3.
in	*aLocalTheta*	An array of kernel parameters.
in	*aDistanceMetric*	Distance metric to be used (1 = Euclidean, 2 = Manhattan, 3 = Minkowski).

Returns

void

Implements exageostat::kernels::Kernel< T >.

8.66.3.2 Create()

```
template<typename T >
static Kernel<T>* exageostat::kernels::UnivariateExpNonGaussian< T >::Create ( )  [static]
```

Creates a new UnivariateExpNonGaussian object.

This method creates a new UnivariateExpNonGaussian object and returns a pointer to it.

Returns

A pointer to the new UnivariateExpNonGaussian object.

8.66.4 Member Data Documentation

8.66.4.1 plugin_name

```
template<typename T >
bool exageostat::kernels::UnivariateExpNonGaussian< T >::plugin_name [static], [private]
```

The documentation for this class was generated from the following file:

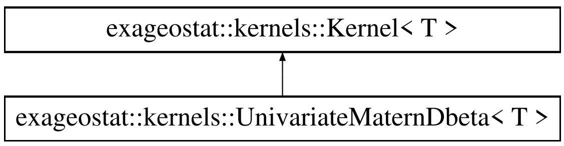
- UnivariateExpNonGaussian.hpp

8.67 exageostat::kernels::UnivariateMaternDbeta< T > Class Template Reference

A class represents a Univariate Matern Dbeta kernel.

```
#include <UnivariateMaternDbeta.hpp>
```

Inheritance diagram for exageostat::kernels::UnivariateMaternDbeta< T >:



Public Member Functions

- UnivariateMaternDbeta ()

  Constructs a new UnivariateMaternDbeta object.
- ∼UnivariateMaternDbeta () override=default

  Virtual destructor to allow calls to the correct concrete destructor.
- void GenerateCovarianceMatrix (T ∗apMatrixA, const int &aRowsNumber, const int &aColumns←Number, const int &aRowOffset, const int &aColumnOffset, dataunits::Locations< T > &aLocation1, dataunits::Locations< T > &aLocation2, dataunits::Locations< T > &aLocation3, T ∗apLocalTheta, const int &aDistanceMetric) override

  Generates a covariance matrix using a set of locations and kernel parameters.

Static Public Member Functions

- static Kernel< T > ∗ Create ()

  Creates a new UnivariateMaternDbeta object.

Static Private Attributes

- static bool plugin_name

Additional Inherited Members

8.67.1 Detailed Description

template<typename T>
class exageostat::kernels::UnivariateMaternDbeta< T >

A class represents a Univariate Matern Dbeta kernel.

This class represents a Univariate Matern Dbeta, which is a subclass of the Kernel class. It provides a method for generating a covariance matrix using a set of input locations and kernel parameters.

8.67.2 Constructor & Destructor Documentation

8.67.2.1 UnivariateMaternDbeta()

```
template<typename T >
exageostat::kernels::UnivariateMaternDbeta< T >::UnivariateMaternDbeta ( )
```

Constructs a new UnivariateMaternDbeta object.

Initializes a new UnivariateMaternDbeta object with default values.

8.67.2.2 ∼UnivariateMaternDbeta()

```
template<typename T >
exageostat::kernels::UnivariateMaternDbeta< T >::∼UnivariateMaternDbeta ( ) [override],
[default]
```

Virtual destructor to allow calls to the correct concrete destructor.

8.67.3 Member Function Documentation

8.67.3.1 GenerateCovarianceMatrix()

```
template<typename T >
void exageostat::kernels::UnivariateMaternDbeta< T >::GenerateCovarianceMatrix (
            T * apMatrixA,
            const int & aRowsNumber,
            const int & aColumnsNumber,
            const int & aRowOffset,
            const int & aColumnOffset,
            dataunits::Locations< T > & aLocation1,
            dataunits::Locations< T > & aLocation2,
            dataunits::Locations< T > & aLocation3,
            T * apLocalTheta,
            const int & aDistanceMetric ) [override], [virtual]
```

Generates a covariance matrix using a set of locations and kernel parameters.

Generates a covariance matrix using a set of locations and kernel parameters.

Parameters

out	*apMatrixA*	The output covariance matrix.
in	*aRowsNumber*	The number of rows in the output matrix.
in	*aColumnsNumber*	The number of columns in the output matrix.
in	*aRowOffset*	The row offset for the input locations.
in	*aColumnOffset*	The column offset for the input locations.
in	*apLocation1*	The set of input locations 1.
in	*apLocation2*	The set of input locations 2.
in	*apLocation3*	The set of input locations 3.
in	*aLocalTheta*	An array of kernel parameters.
in	*aDistanceMetric*	Distance metric to be used (1 = Euclidean, 2 = Manhattan, 3 = Minkowski).

Returns

void

Implements exageostat::kernels::Kernel< T >.

8.67.3.2 Create()

```
template<typename T >
static Kernel<T>* exageostat::kernels::UnivariateMaternDbeta< T >::Create ( )  [static]
```

Creates a new UnivariateMaternDbeta object.

This method creates a new UnivariateMaternDbeta object and returns a pointer to it.

Returns

A pointer to the new UnivariateMaternDbeta object.

8.67.4 Member Data Documentation

8.67.4.1 plugin_name

```
template<typename T >
bool exageostat::kernels::UnivariateMaternDbeta< T >::plugin_name  [static], [private]
```

The documentation for this class was generated from the following file:

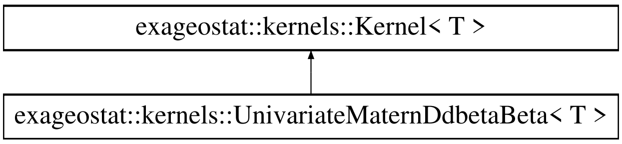
- UnivariateMaternDbeta.hpp

8.68 exageostat::kernels::UnivariateMaternDdbetaBeta< T > Class Template Reference

A class represents a Univariate Matern Ddbeta Beta kernel.

```
#include <UnivariateMaternDdbetaBeta.hpp>
```

Inheritance diagram for exageostat::kernels::UnivariateMaternDdbetaBeta< T >:



Public Member Functions

- UnivariateMaternDdbetaBeta ()
  Constructs a new UnivariateMaternDdbetaBeta object.
- ∼UnivariateMaternDdbetaBeta () override=default
  Virtual destructor to allow calls to the correct concrete destructor.
- void GenerateCovarianceMatrix (T ∗apMatrixA, const int &aRowsNumber, const int &aColumnsNumber, const int &aRowOffset, const int &aColumnOffset, dataunits::Locations< T > &aLocation1, dataunits::Locations< T > &aLocation2, dataunits::Locations< T > &aLocation3, T ∗apLocalTheta, const int &aDistanceMetric) override
  Generates a covariance matrix using a set of locations and kernel parameters.

Static Public Member Functions

- static Kernel< T > ∗ Create ()
  Creates a new UnivariateMaternDdbetaBeta object.

Static Private Attributes

- static bool plugin_name

Additional Inherited Members

8.68.1 Detailed Description

template<typename T>
class exageostat::kernels::UnivariateMaternDdbetaBeta< T >

A class represents a Univariate Matern Ddbeta Beta kernel.

This class represents a Univariate Matern Ddbeta Beta, which is a subclass of the Kernel class. It provides a method for generating a covariance matrix using a set of input locations and kernel parameters.

8.68.2 Constructor & Destructor Documentation

8.68.2.1 UnivariateMaternDdbetaBeta()

```
template<typename T >
exageostat::kernels::UnivariateMaternDdbetaBeta< T >::UnivariateMaternDdbetaBeta ( )
```

Constructs a new UnivariateMaternDdbetaBeta object.

Initializes a new UnivariateMaternDdbetaBeta object with default values.

8.68.2.2 ∼UnivariateMaternDdbetaBeta()

```
template<typename T >
exageostat::kernels::UnivariateMaternDdbetaBeta< T >::∼UnivariateMaternDdbetaBeta ( ) [override],
[default]
```

Virtual destructor to allow calls to the correct concrete destructor.

8.68.3 Member Function Documentation

8.68.3.1 GenerateCovarianceMatrix()

```
template<typename T >
void exageostat::kernels::UnivariateMaternDdbetaBeta< T >::GenerateCovarianceMatrix (
            T * apMatrixA,
            const int & aRowsNumber,
            const int & aColumnsNumber,
            const int & aRowOffset,
            const int & aColumnOffset,
            dataunits::Locations< T > & aLocation1,
            dataunits::Locations< T > & aLocation2,
            dataunits::Locations< T > & aLocation3,
            T * apLocalTheta,
            const int & aDistanceMetric ) [override], [virtual]
```

Generates a covariance matrix using a set of locations and kernel parameters.

Generates a covariance matrix using a set of locations and kernel parameters.

Parameters

out	*apMatrixA*	The output covariance matrix.
in	*aRowsNumber*	The number of rows in the output matrix.
in	*aColumnsNumber*	The number of columns in the output matrix.
in	*aRowOffset*	The row offset for the input locations.
in	*aColumnOffset*	The column offset for the input locations.
in	*apLocation1*	The set of input locations 1.
in	*apLocation2*	The set of input locations 2.
in	*apLocation3*	The set of input locations 3.

Returns

void

Implements exageostat::kernels::Kernel< T >.

8.68.3.2 Create()

```
template<typename T >
static Kernel<T>* exageostat::kernels::UnivariateMaternDdbetaBeta< T >::Create ( ) [static]
```

Creates a new UnivariateMaternDdbetaBeta object.

This method creates a new UnivariateMaternDdbetaBeta object and returns a pointer to it.

Returns

A pointer to the new UnivariateMaternDdbetaBeta object.

8.68.4 Member Data Documentation

8.68.4.1 plugin_name

```
template<typename T >
bool exageostat::kernels::UnivariateMaternDdbetaBeta< T >::plugin_name [static], [private]
```

The documentation for this class was generated from the following file:

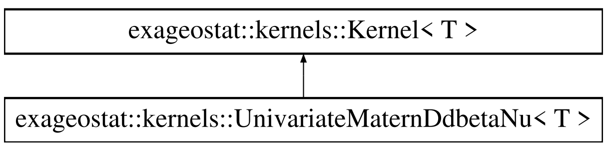
- UnivariateMaternDdbetaBeta.hpp

8.69 exageostat::kernels::UnivariateMaternDdbetaNu< T > Class Template Reference

A class represents a Univariate Matern Ddbeta Nu kernel.

`#include <UnivariateMaternDdbetaNu.hpp>`

Inheritance diagram for exageostat::kernels::UnivariateMaternDdbetaNu< T >:



Public Member Functions

- UnivariateMaternDdbetaNu ()

  Constructs a new UnivariateMaternDdbetaNu object.
- ∼UnivariateMaternDdbetaNu () override=default

  Virtual destructor to allow calls to the correct concrete destructor.
- void GenerateCovarianceMatrix (T ∗apMatrixA, const int &aRowsNumber, const int &aColumns←Number, const int &aRowOffset, const int &aColumnOffset, dataunits::Locations< T > &aLocation1, dataunits::Locations< T > &aLocation2, dataunits::Locations< T > &aLocation3, T ∗apLocalTheta, const int &aDistanceMetric) override

  Generates a covariance matrix using a set of locations and kernel parameters.

Static Public Member Functions

- static Kernel< T > ∗ Create ()

  Creates a new UnivariateMaternDdbetaNu object.

Static Private Attributes

- static bool plugin_name

Additional Inherited Members

8.69.1 Detailed Description

template<typename T>
class exageostat::kernels::UnivariateMaternDdbetaNu< T >

A class represents a Univariate Matern Ddbeta Nu kernel.

This class represents a Univariate Matern Ddbeta Nu, which is a subclass of the Kernel class. It provides a method for generating a covariance matrix using a set of input locations and kernel parameters.

8.69.2 Constructor & Destructor Documentation

8.69.2.1 UnivariateMaternDdbetaNu()

```
template<typename T >
exageostat::kernels::UnivariateMaternDdbetaNu< T >::UnivariateMaternDdbetaNu ( )
```

Constructs a new UnivariateMaternDdbetaNu object.

Initializes a new UnivariateMaternDdbetaNu object with default values.

8.69.2.2 ∼UnivariateMaternDdbetaNu()

```
template<typename T >
exageostat::kernels::UnivariateMaternDdbetaNu< T >::~UnivariateMaternDdbetaNu ( ) [override],
[default]
```

Virtual destructor to allow calls to the correct concrete destructor.

8.69.3 Member Function Documentation

8.69.3.1 GenerateCovarianceMatrix()

```
template<typename T >
void exageostat::kernels::UnivariateMaternDdbetaNu< T >::GenerateCovarianceMatrix (
            T * apMatrixA,
            const int & aRowsNumber,
            const int & aColumnsNumber,
            const int & aRowOffset,
            const int & aColumnOffset,
            dataunits::Locations< T > & aLocation1,
            dataunits::Locations< T > & aLocation2,
            dataunits::Locations< T > & aLocation3,
            T * apLocalTheta,
            const int & aDistanceMetric ) [override], [virtual]
```

Generates a covariance matrix using a set of locations and kernel parameters.

Generates a covariance matrix using a set of locations and kernel parameters.

Parameters

out	*apMatrixA*	The output covariance matrix.
in	*aRowsNumber*	The number of rows in the output matrix.
in	*aColumnsNumber*	The number of columns in the output matrix.
in	*aRowOffset*	The row offset for the input locations.
in	*aColumnOffset*	The column offset for the input locations.
in	*apLocation1*	The set of input locations 1.
in	*apLocation2*	The set of input locations 2.
in	*apLocation3*	The set of input locations 3.
in	*aLocalTheta*	An array of kernel parameters.
in	*aDistanceMetric*	Distance metric to be used (1 = Euclidean, 2 = Manhattan, 3 = Minkowski).

Returns

void

Implements exageostat::kernels::Kernel< T >.

8.69.3.2 Create()

```
template<typename T >
static Kernel<T>* exageostat::kernels::UnivariateMaternDdbetaNu< T >::Create ( )  [static]
```

Creates a new UnivariateMaternDdbetaNu object.

This method creates a new UnivariateMaternDdbetaNu object and returns a pointer to it.

Returns

A pointer to the new UnivariateMaternDdbetaNu object.

8.69.4 Member Data Documentation

8.69.4.1 plugin_name

```
template<typename T >
bool exageostat::kernels::UnivariateMaternDdbetaNu< T >::plugin_name  [static], [private]
```

The documentation for this class was generated from the following file:

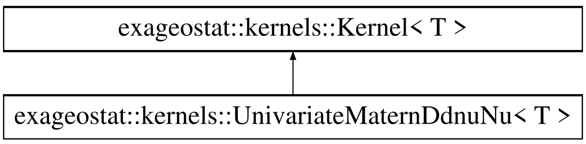
- UnivariateMaternDdbetaNu.hpp

8.70 exageostat::kernels::UnivariateMaternDdnuNu< T > Class Template Reference

A class represents a Univariate Matern Ddnu Nu kernel.

```
#include <UnivariateMaternDdnuNu.hpp>
```

Inheritance diagram for exageostat::kernels::UnivariateMaternDdnuNu< T >:



Public Member Functions

- UnivariateMaternDdnuNu ()

  Constructs a new UnivariateMaternDdnuNu object.
- ∼UnivariateMaternDdnuNu () override=default

  Virtual destructor to allow calls to the correct concrete destructor.
- void GenerateCovarianceMatrix (T ∗apMatrixA, const int &aRowsNumber, const int &aColumnsNumber, const int &aRowOffset, const int &aColumnOffset, dataunits::Locations< T > &aLocation1, dataunits::Locations< T > &aLocation2, dataunits::Locations< T > &aLocation3, T ∗apLocalTheta, const int &aDistanceMetric) override

  Generates a covariance matrix using a set of locations and kernel parameters.

Static Public Member Functions

- static Kernel< T > * Create ()

  Creates a new UnivariateMaternDdnuNu object.

Static Private Attributes

- static bool plugin_name

Additional Inherited Members

8.70.1 Detailed Description

template<typename T>
class exageostat::kernels::UnivariateMaternDdnuNu< T >

A class represents a Univariate Matern Ddnu Nu kernel.

This class represents a Univariate Matern Ddnu Nu , which is a subclass of the Kernel class. It provides a method for generating a covariance matrix using a set of input locations and kernel parameters.

8.70.2 Constructor & Destructor Documentation

8.70.2.1 UnivariateMaternDdnuNu()

```
template<typename T >
exageostat::kernels::UnivariateMaternDdnuNu< T >::UnivariateMaternDdnuNu ( )
```

Constructs a new UnivariateMaternDdnuNu object.

Initializes a new UnivariateMaternDdnuNu object with default values.

8.70.2.2 ∼UnivariateMaternDdnuNu()

```
template<typename T >
exageostat::kernels::UnivariateMaternDdnuNu< T >::~UnivariateMaternDdnuNu ( ) [override],
[default]
```

Virtual destructor to allow calls to the correct concrete destructor.

8.70.3 Member Function Documentation

8.70.3.1 GenerateCovarianceMatrix()

```
template<typename T >
void exageostat::kernels::UnivariateMaternDdnuNu< T >::GenerateCovarianceMatrix (
            T * apMatrixA,
            const int & aRowsNumber,
            const int & aColumnsNumber,
            const int & aRowOffset,
            const int & aColumnOffset,
            dataunits::Locations< T > & aLocation1,
            dataunits::Locations< T > & aLocation2,
            dataunits::Locations< T > & aLocation3,
            T * apLocalTheta,
            const int & aDistanceMetric )  [override], [virtual]
```

Generates a covariance matrix using a set of locations and kernel parameters.

Generates a covariance matrix using a set of locations and kernel parameters.

Parameters

out	*apMatrixA*	The output covariance matrix.
in	*aRowsNumber*	The number of rows in the output matrix.
in	*aColumnsNumber*	The number of columns in the output matrix.
in	*aRowOffset*	The row offset for the input locations.
in	*aColumnOffset*	The column offset for the input locations.
in	*apLocation1*	The set of input locations 1.
in	*apLocation2*	The set of input locations 2.
in	*apLocation3*	The set of input locations 3.
in	*aLocalTheta*	An array of kernel parameters.
in	*aDistanceMetric*	Distance metric to be used (1 = Euclidean, 2 = Manhattan, 3 = Minkowski).

Returns

void

Implements exageostat::kernels::Kernel< T >.

8.70.3.2 Create()

```
template<typename T >
static Kernel<T>* exageostat::kernels::UnivariateMaternDdnuNu< T >::Create ( )  [static]
```

Creates a new UnivariateMaternDdnuNu object.

This method creates a new UnivariateMaternDdnuNu object and returns a pointer to it.

Returns

A pointer to the new UnivariateMaternDdnuNu object.

8.70.4 Member Data Documentation

8.70.4.1 plugin_name

```
template<typename T >
bool exageostat::kernels::UnivariateMaternDdnuNu< T >::plugin_name [static], [private]
```

The documentation for this class was generated from the following file:

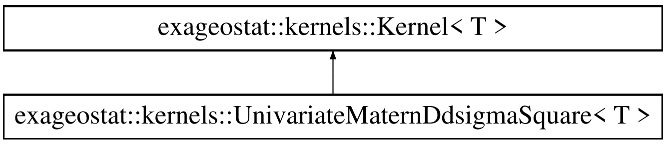
- UnivariateMaternDdnuNu.hpp

8.71 exageostat::kernels::UnivariateMaternDdsigmaSquare< T > Class Template Reference

A class represents a Univariate Matern Ddsigma Square kernel.

```
#include <UnivariateMaternDdsigmaSquare.hpp>
```

Inheritance diagram for exageostat::kernels::UnivariateMaternDdsigmaSquare< T >:



Public Member Functions

- UnivariateMaternDdsigmaSquare ()

    Constructs a new UnivariateMaternDdsigmaSquare object.
- ∼UnivariateMaternDdsigmaSquare () override=default

    Virtual destructor to allow calls to the correct concrete destructor.
- void GenerateCovarianceMatrix (T ∗apMatrixA, const int &aRowsNumber, const int &aColumnsNumber, const int &aRowOffset, const int &aColumnOffset, dataunits::Locations< T > &aLocation1, dataunits::Locations< T > &aLocation2, dataunits::Locations< T > &aLocation3, T ∗apLocalTheta, const int &aDistanceMetric) override

    Generates a covariance matrix using a set of locations and kernel parameters.

Static Public Member Functions

- static Kernel< T > ∗ Create ()

    Creates a new UnivariateMaternDdsigmaSquare object.

Static Private Attributes

- static bool plugin_name

Additional Inherited Members

8.71.1 Detailed Description

template<typename T>
class exageostat::kernels::UnivariateMaternDdsigmaSquare< T >

A class represents a Univariate Matern Ddsigma Square kernel.

This class represents a Univariate Matern Ddsigma Square, which is a subclass of the Kernel class. It provides a method for generating a covariance matrix using a set of input locations and kernel parameters.

8.71.2 Constructor & Destructor Documentation

8.71.2.1 UnivariateMaternDdsigmaSquare()

```
template<typename T >
exageostat::kernels::UnivariateMaternDdsigmaSquare< T >::UnivariateMaternDdsigmaSquare ( )
```

Constructs a new UnivariateMaternDdsigmaSquare object.

Initializes a new UnivariateMaternDdsigmaSquare object with default values.

8.71.2.2 ∼UnivariateMaternDdsigmaSquare()

```
template<typename T >
exageostat::kernels::UnivariateMaternDdsigmaSquare< T >::∼UnivariateMaternDdsigmaSquare ( )
[override], [default]
```

Virtual destructor to allow calls to the correct concrete destructor.

8.71.3 Member Function Documentation

8.71.3.1 GenerateCovarianceMatrix()

```
template<typename T >
void exageostat::kernels::UnivariateMaternDdsigmaSquare< T >::GenerateCovarianceMatrix (
            T * apMatrixA,
            const int & aRowsNumber,
            const int & aColumnsNumber,
            const int & aRowOffset,
            const int & aColumnOffset,
            dataunits::Locations< T > & aLocation1,
            dataunits::Locations< T > & aLocation2,
            dataunits::Locations< T > & aLocation3,
            T * apLocalTheta,
            const int & aDistanceMetric )  [override], [virtual]
```

Generates a covariance matrix using a set of locations and kernel parameters.

Generates a covariance matrix using a set of locations and kernel parameters.

Parameters

out	*apMatrixA*	The output covariance matrix.
in	*aRowsNumber*	The number of rows in the output matrix.
in	*aColumnsNumber*	The number of columns in the output matrix.
in	*aRowOffset*	The row offset for the input locations.
in	*aColumnOffset*	The column offset for the input locations.
in	*apLocation1*	The set of input locations 1.
in	*apLocation2*	The set of input locations 2.
in	*apLocation3*	The set of input locations 3.
in	*aLocalTheta*	An array of kernel parameters.
in	*aDistanceMetric*	Distance metric to be used (1 = Euclidean, 2 = Manhattan, 3 = Minkowski).

Returns

void

Implements exageostat::kernels::Kernel< T >.

8.71.3.2 Create()

```
template<typename T >
static Kernel<T>* exageostat::kernels::UnivariateMaternDdsigmaSquare< T >::Create ( )  [static]
```

Creates a new UnivariateMaternDdsigmaSquare object.

This method creates a new UnivariateMaternDdsigmaSquare object and returns a pointer to it.

Returns

A pointer to the new UnivariateMaternDdsigmaSquare object.

8.71.4 Member Data Documentation

8.71.4.1 plugin_name

```
template<typename T >
bool exageostat::kernels::UnivariateMaternDdsigmaSquare< T >::plugin_name  [static], [private]
```

The documentation for this class was generated from the following file:

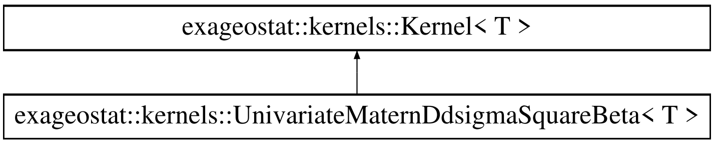
- UnivariateMaternDdsigmaSquare.hpp

8.72 exageostat::kernels::UnivariateMaternDdsigmaSquareBeta< T > Class Template Reference

A class represents a Univariate Matern Ddsigma Square Beta kernel.

```
#include <UnivariateMaternDdsigmaSquareBeta.hpp>
```

Inheritance diagram for exageostat::kernels::UnivariateMaternDdsigmaSquareBeta< T >:



Public Member Functions

- UnivariateMaternDdsigmaSquareBeta ()

    Constructs a new UnivariateMaternDdsigmaSquareBeta object.
- ∼UnivariateMaternDdsigmaSquareBeta () override=default

    Virtual destructor to allow calls to the correct concrete destructor.
- void GenerateCovarianceMatrix (T ∗apMatrixA, const int &aRowsNumber, const int &aColumnsNumber, const int &aRowOffset, const int &aColumnOffset, dataunits::Locations< T > &aLocation1, dataunits::Locations< T > &aLocation2, dataunits::Locations< T > &aLocation3, T ∗apLocalTheta, const int &aDistanceMetric) override

    Generates a covariance matrix using a set of locations and kernel parameters.

Static Public Member Functions

- static Kernel< T > ∗ Create ()

    Creates a new UnivariateMaternDdsigmaSquareBeta object.

Static Private Attributes

- static bool plugin_name

Additional Inherited Members

8.72.1 Detailed Description

template<typename T>
class exageostat::kernels::UnivariateMaternDdsigmaSquareBeta< T >

A class represents a Univariate Matern Ddsigma Square Beta kernel.

This class represents a Univariate Matern Ddsigma Square Beta, which is a subclass of the Kernel class. It provides a method for generating a covariance matrix using a set of input locations and kernel parameters.

8.72.2 Constructor & Destructor Documentation

8.72.2.1 UnivariateMaternDdsigmaSquareBeta()

```
template<typename T >
exageostat::kernels::UnivariateMaternDdsigmaSquareBeta< T >::UnivariateMaternDdsigmaSquareBeta
( )
```

Constructs a new UnivariateMaternDdsigmaSquareBeta object.

Initializes a new UnivariateMaternDdsigmaSquareBeta object with default values.

8.72.2.2 ∼UnivariateMaternDdsigmaSquareBeta()

```
template<typename T >
exageostat::kernels::UnivariateMaternDdsigmaSquareBeta< T >::~UnivariateMaternDdsigmaSquareBeta
( ) [override], [default]
```

Virtual destructor to allow calls to the correct concrete destructor.

8.72.3 Member Function Documentation

8.72.3.1 GenerateCovarianceMatrix()

```
template<typename T >
void exageostat::kernels::UnivariateMaternDdsigmaSquareBeta< T >::GenerateCovarianceMatrix (
            T * apMatrixA,
            const int & aRowsNumber,
            const int & aColumnsNumber,
            const int & aRowOffset,
            const int & aColumnOffset,
            dataunits::Locations< T > & aLocation1,
            dataunits::Locations< T > & aLocation2,
            dataunits::Locations< T > & aLocation3,
            T * apLocalTheta,
            const int & aDistanceMetric ) [override], [virtual]
```

Generates a covariance matrix using a set of locations and kernel parameters.

Generates a covariance matrix using a set of locations and kernel parameters.

Parameters

out	*apMatrixA*	The output covariance matrix.
in	*aRowsNumber*	The number of rows in the output matrix.
in	*aColumnsNumber*	The number of columns in the output matrix.
in	*aRowOffset*	The row offset for the input locations.
in	*aColumnOffset*	The column offset for the input locations.
in	*apLocation1*	The set of input locations 1.
in	*apLocation2*	The set of input locations 2.

Returns

void

Implements exageostat::kernels::Kernel< T >.

8.72.3.2 Create()

```
template<typename T >
static Kernel<T>* exageostat::kernels::UnivariateMaternDdsigmaSquareBeta< T >::Create ( )
[static]
```

Creates a new UnivariateMaternDdsigmaSquareBeta object.

This method creates a new UnivariateMaternDdsigmaSquareBeta object and returns a pointer to it.

Returns

A pointer to the new UnivariateMaternDdsigmaSquareBeta object.

8.72.4 Member Data Documentation

8.72.4.1 plugin_name

```
template<typename T >
bool exageostat::kernels::UnivariateMaternDdsigmaSquareBeta< T >::plugin_name [static], [private]
```

The documentation for this class was generated from the following file:

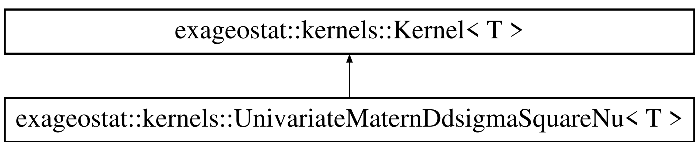
- UnivariateMaternDdsigmaSquareBeta.hpp

8.73 exageostat::kernels::UnivariateMaternDdsigmaSquareNu< T > Class Template Reference

A class represents a Univariate Matern Ddsigma Square Nu kernel.

```
#include <UnivariateMaternDdsigmaSquareNu.hpp>
```

Inheritance diagram for exageostat::kernels::UnivariateMaternDdsigmaSquareNu< T >:



Public Member Functions

- UnivariateMaternDdsigmaSquareNu ()

  Constructs a new UnivariateMaternDdsigmaSquareNu object.
- ∼UnivariateMaternDdsigmaSquareNu () override=default

  Virtual destructor to allow calls to the correct concrete destructor.
- void GenerateCovarianceMatrix (T ∗apMatrixA, const int &aRowsNumber, const int &aColumns↩Number, const int &aRowOffset, const int &aColumnOffset, dataunits::Locations< T > &aLocation1, dataunits::Locations< T > &aLocation2, dataunits::Locations< T > &aLocation3, T ∗apLocalTheta, const int &aDistanceMetric) override

  Generates a covariance matrix using a set of locations and kernel parameters.

Static Public Member Functions

- static Kernel< T > ∗ Create ()

  Creates a new UnivariateMaternDdsigmaSquareNu object.

Static Private Attributes

- static bool plugin_name

Additional Inherited Members

8.73.1 Detailed Description

template<typename T>
class exageostat::kernels::UnivariateMaternDdsigmaSquareNu< T >

A class represents a Univariate Matern Ddsigma Square Nu kernel.

This class represents a Univariate Matern Ddsigma Square Nu, which is a subclass of the Kernel class. It provides a method for generating a covariance matrix using a set of input locations and kernel parameters.

8.73.2 Constructor & Destructor Documentation

8.73.2.1 UnivariateMaternDdsigmaSquareNu()

```
template<typename T >
exageostat::kernels::UnivariateMaternDdsigmaSquareNu< T >::UnivariateMaternDdsigmaSquareNu ( )
```

Constructs a new UnivariateMaternDdsigmaSquareNu object.

Initializes a new UnivariateMaternDdsigmaSquareNu object with default values.

8.73.2.2 ∼UnivariateMaternDdsigmaSquareNu()

```
template<typename T >
exageostat::kernels::UnivariateMaternDdsigmaSquareNu< T >::∼UnivariateMaternDdsigmaSquareNu (
) [override], [default]
```

Virtual destructor to allow calls to the correct concrete destructor.

8.73.3 Member Function Documentation

8.73.3.1 GenerateCovarianceMatrix()

```
template<typename T >
void exageostat::kernels::UnivariateMaternDdsigmaSquareNu< T >::GenerateCovarianceMatrix (
            T * apMatrixA,
            const int & aRowsNumber,
            const int & aColumnsNumber,
            const int & aRowOffset,
            const int & aColumnOffset,
            dataunits::Locations< T > & aLocation1,
            dataunits::Locations< T > & aLocation2,
            dataunits::Locations< T > & aLocation3,
            T * apLocalTheta,
            const int & aDistanceMetric ) [override], [virtual]
```

Generates a covariance matrix using a set of locations and kernel parameters.

Generates a covariance matrix using a set of locations and kernel parameters.

Parameters

out	*apMatrixA*	The output covariance matrix.
in	*aRowsNumber*	The number of rows in the output matrix.
in	*aColumnsNumber*	The number of columns in the output matrix.
in	*aRowOffset*	The row offset for the input locations.
in	*aColumnOffset*	The column offset for the input locations.
in	*apLocation1*	The set of input locations 1.
in	*apLocation2*	The set of input locations 2.
in	*apLocation3*	The set of input locations 3.
in	*aLocalTheta*	An array of kernel parameters.
in	*aDistanceMetric*	Distance metric to be used (1 = Euclidean, 2 = Manhattan, 3 = Minkowski).

Returns

void

Implements exageostat::kernels::Kernel< T >.

8.73.3.2 Create()

```
template<typename T >
static Kernel<T>* exageostat::kernels::UnivariateMaternDdsigmaSquareNu< T >::Create ( ) [static]
```

Creates a new UnivariateMaternDdsigmaSquareNu object.

This method creates a new UnivariateMaternDdsigmaSquareNu object and returns a pointer to it.

Returns

A pointer to the new UnivariateMaternDdsigmaSquareNu object.

8.73.4 Member Data Documentation

8.73.4.1 plugin_name

```
template<typename T >
bool exageostat::kernels::UnivariateMaternDdsigmaSquareNu< T >::plugin_name [static], [private]
```

The documentation for this class was generated from the following file:

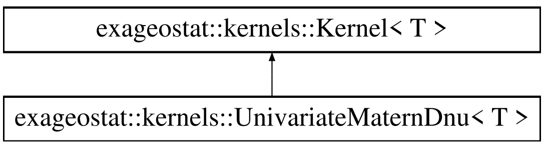
- UnivariateMaternDdsigmaSquareNu.hpp

8.74 exageostat::kernels::UnivariateMaternDnu< T > Class Template Reference

A class represents a Univariate Matern Dnu kernel.

`#include <UnivariateMaternDnu.hpp>`

Inheritance diagram for exageostat::kernels::UnivariateMaternDnu< T >:



Public Member Functions

- UnivariateMaternDnu ()

  Constructs a new UnivariateMaternDnu object.
- ~UnivariateMaternDnu () override=default

  Virtual destructor to allow calls to the correct concrete destructor.
- void GenerateCovarianceMatrix (T *apMatrixA, const int &aRowsNumber, const int &aColumns←Number, const int &aRowOffset, const int &aColumnOffset, datunits::Locations< T > &aLocation1, datunits::Locations< T > &aLocation2, datunits::Locations< T > &aLocation3, T *apLocalTheta, const int &aDistanceMetric) override

  Generates a covariance matrix using a set of locations and kernel parameters.

Static Public Member Functions

- static Kernel< T > * Create ()

  Creates a new UnivariateMaternDnu object.

Static Private Attributes

- static bool plugin_name

Additional Inherited Members

8.74.1 Detailed Description

template<typename T>
class exageostat::kernels::UnivariateMaternDnu< T >

A class represents a Univariate Matern Dnu kernel.

This class represents a Univariate Matern Dnu, which is a subclass of the Kernel class. It provides a method for generating a covariance matrix using a set of input locations and kernel parameters.

8.74.2 Constructor & Destructor Documentation

8.74.2.1 UnivariateMaternDnu()

```
template<typename T >
exageostat::kernels::UnivariateMaternDnu< T >::UnivariateMaternDnu ( )
```

Constructs a new UnivariateMaternDnu object.

Initializes a new UnivariateMaternDnu object with default values.

8.74.2.2 ∼UnivariateMaternDnu()

```
template<typename T >
exageostat::kernels::UnivariateMaternDnu< T >::~UnivariateMaternDnu ( )  [override], [default]
```

Virtual destructor to allow calls to the correct concrete destructor.

8.74.3 Member Function Documentation

8.74.3.1 GenerateCovarianceMatrix()

```
template<typename T >
void exageostat::kernels::UnivariateMaternDnu< T >::GenerateCovarianceMatrix (
            T * apMatrixA,
            const int & aRowsNumber,
            const int & aColumnsNumber,
            const int & aRowOffset,
            const int & aColumnOffset,
            dataunits::Locations< T > & aLocation1,
            dataunits::Locations< T > & aLocation2,
            dataunits::Locations< T > & aLocation3,
            T * apLocalTheta,
            const int & aDistanceMetric )  [override], [virtual]
```

Generates a covariance matrix using a set of locations and kernel parameters.

Generates a covariance matrix using a set of locations and kernel parameters.

Parameters

out	*apMatrixA*	The output covariance matrix.
in	*aRowsNumber*	The number of rows in the output matrix.
in	*aColumnsNumber*	The number of columns in the output matrix.
in	*aRowOffset*	The row offset for the input locations.
in	*aColumnOffset*	The column offset for the input locations.
in	*apLocation1*	The set of input locations 1.
in	*apLocation2*	The set of input locations 2.
in	*apLocation3*	The set of input locations 3.
in	*aLocalTheta*	An array of kernel parameters.
in	*aDistanceMetric*	Distance metric to be used (1 = Euclidean, 2 = Manhattan, 3 = Minkowski).

Returns

void

Implements exageostat::kernels::Kernel< T >.

8.74.3.2 Create()

```
template<typename T >
static Kernel<T>* exageostat::kernels::UnivariateMaternDnu< T >::Create ( )  [static]
```

Creates a new UnivariateMaternDnu object.

This method creates a new UnivariateMaternDnu object and returns a pointer to it.

Returns

A pointer to the new UnivariateMaternDnu object.

8.74.4 Member Data Documentation

8.74.4.1 plugin_name

```
template<typename T >
bool exageostat::kernels::UnivariateMaternDnu< T >::plugin_name [static], [private]
```

The documentation for this class was generated from the following file:

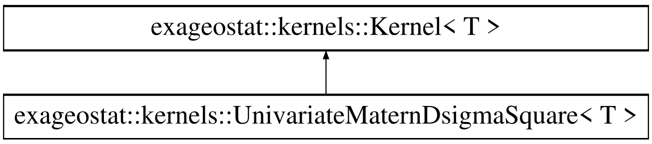
- UnivariateMaternDnu.hpp

8.75 exageostat::kernels::UnivariateMaternDsigmaSquare< T > Class Template Reference

A class represents a Univariate Matern Dsigma Square kernel.

```
#include <UnivariateMaternDsigmaSquare.hpp>
```

Inheritance diagram for exageostat::kernels::UnivariateMaternDsigmaSquare< T >:



Public Member Functions

- UnivariateMaternDsigmaSquare ()

  Constructs a new UnivariateMaternDsigmaSquare object.
- ∼UnivariateMaternDsigmaSquare () override=default

  Virtual destructor to allow calls to the correct concrete destructor.
- void GenerateCovarianceMatrix (T ∗apMatrixA, const int &aRowsNumber, const int &aColumns←Number, const int &aRowOffset, const int &aColumnOffset, dataunits::Locations< T > &aLocation1, dataunits::Locations< T > &aLocation2, dataunits::Locations< T > &aLocation3, T ∗apLocalTheta, const int &aDistanceMetric) override

  Generates a covariance matrix using a set of locations and kernel parameters.

Static Public Member Functions

- static Kernel< T > ∗ Create ()

  Creates a new UnivariateMaternDsigmaSquare object.

Static Private Attributes

- static bool plugin_name

Additional Inherited Members

8.75.1 Detailed Description

template<typename T>
class exageostat::kernels::UnivariateMaternDsigmaSquare< T >

A class represents a Univariate Matern Dsigma Square kernel.

This class represents a Univariate Matern Dsigma Square, which is a subclass of the Kernel class. It provides a method for generating a covariance matrix using a set of input locations and kernel parameters.

8.75.2 Constructor & Destructor Documentation

8.75.2.1 UnivariateMaternDsigmaSquare()

```
template<typename T >
exageostat::kernels::UnivariateMaternDsigmaSquare< T >::UnivariateMaternDsigmaSquare ( )
```

Constructs a new UnivariateMaternDsigmaSquare object.

Initializes a new UnivariateMaternDsigmaSquare object with default values.

8.75.2.2 ∼UnivariateMaternDsigmaSquare()

```
template<typename T >
exageostat::kernels::UnivariateMaternDsigmaSquare< T >::~UnivariateMaternDsigmaSquare ( )
[override], [default]
```

Virtual destructor to allow calls to the correct concrete destructor.

8.75.3 Member Function Documentation

8.75.3.1 GenerateCovarianceMatrix()

```
template<typename T >
void exageostat::kernels::UnivariateMaternDsigmaSquare< T >::GenerateCovarianceMatrix (
            T * apMatrixA,
            const int & aRowsNumber,
            const int & aColumnsNumber,
            const int & aRowOffset,
            const int & aColumnOffset,
            dataunits::Locations< T > & aLocation1,
            dataunits::Locations< T > & aLocation2,
            dataunits::Locations< T > & aLocation3,
            T * apLocalTheta,
            const int & aDistanceMetric )  [override], [virtual]
```

Generates a covariance matrix using a set of locations and kernel parameters.

Generates a covariance matrix using a set of locations and kernel parameters.

Parameters

out	*apMatrixA*	The output covariance matrix.
in	*aRowsNumber*	The number of rows in the output matrix.
in	*aColumnsNumber*	The number of columns in the output matrix.
in	*aRowOffset*	The row offset for the input locations.
in	*aColumnOffset*	The column offset for the input locations.
in	*apLocation1*	The set of input locations 1.
in	*apLocation2*	The set of input locations 2.
in	*apLocation3*	The set of input locations 3.
in	*aLocalTheta*	An array of kernel parameters.
in	*aDistanceMetric*	Distance metric to be used (1 = Euclidean, 2 = Manhattan, 3 = Minkowski).

Returns

void

Implements exageostat::kernels::Kernel< T >.

8.75.3.2 Create()

```
template<typename T >
static Kernel<T>* exageostat::kernels::UnivariateMaternDsigmaSquare< T >::Create ( )  [static]
```

Creates a new UnivariateMaternDsigmaSquare object.

This method creates a new UnivariateMaternDsigmaSquare object and returns a pointer to it.

Returns

A pointer to the new UnivariateMaternDsigmaSquare object.

8.75.4 Member Data Documentation

8.75.4.1 plugin_name

```
template<typename T >
bool exageostat::kernels::UnivariateMaternDsigmaSquare< T >::plugin_name  [static], [private]
```

The documentation for this class was generated from the following file:

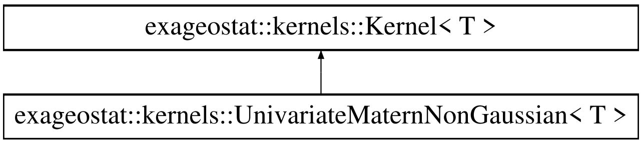
- UnivariateMaternDsigmaSquare.hpp

8.76 exageostat::kernels::UnivariateMaternNonGaussian< T > Class Template Reference

A class represents a Univariate Matern Non Gaussian kernel.

```
#include <UnivariateMaternNonGaussian.hpp>
```

Inheritance diagram for exageostat::kernels::UnivariateMaternNonGaussian< T >:



Public Member Functions

- UnivariateMaternNonGaussian ()

  Constructs a new UnivariateMaternNonGaussian object.
- ∼UnivariateMaternNonGaussian () override=default

  Virtual destructor to allow calls to the correct concrete destructor.
- void GenerateCovarianceMatrix (T ∗apMatrixA, const int &aRowsNumber, const int &aColumns←Number, const int &aRowOffset, const int &aColumnOffset, dataunits::Locations< T > &aLocation1, dataunits::Locations< T > &aLocation2, dataunits::Locations< T > &aLocation3, T ∗apLocalTheta, const int &aDistanceMetric) override

  Generates a covariance matrix using a set of locations and kernel parameters.

Static Public Member Functions

- static Kernel< T > ∗ Create ()

  Creates a new UnivariateMaternNonGaussian object.

Static Private Attributes

- static bool plugin_name

Additional Inherited Members

8.76.1 Detailed Description

template<typename T>
class exageostat::kernels::UnivariateMaternNonGaussian< T >

A class represents a Univariate Matern Non Gaussian kernel.

This class represents a Univariate Matern Non Gaussian, which is a subclass of the Kernel class. It provides a method for generating a covariance matrix using a set of input locations and kernel parameters.

8.76.2 Constructor & Destructor Documentation

8.76.2.1 UnivariateMaternNonGaussian()

```
template<typename T >
exageostat::kernels::UnivariateMaternNonGaussian< T >::UnivariateMaternNonGaussian ( )
```

Constructs a new UnivariateMaternNonGaussian object.

Initializes a new UnivariateMaternNonGaussian object with default values.

8.76.2.2 ∼UnivariateMaternNonGaussian()

```
template<typename T >
exageostat::kernels::UnivariateMaternNonGaussian< T >::∼UnivariateMaternNonGaussian ( )  [override],
[default]
```

Virtual destructor to allow calls to the correct concrete destructor.

8.76.3 Member Function Documentation

8.76.3.1 GenerateCovarianceMatrix()

```
template<typename T >
void exageostat::kernels::UnivariateMaternNonGaussian< T >::GenerateCovarianceMatrix (
            T * apMatrixA,
            const int & aRowsNumber,
            const int & aColumnsNumber,
            const int & aRowOffset,
            const int & aColumnOffset,
            dataunits::Locations< T > & aLocation1,
            dataunits::Locations< T > & aLocation2,
            dataunits::Locations< T > & aLocation3,
            T * apLocalTheta,
            const int & aDistanceMetric )  [override], [virtual]
```

Generates a covariance matrix using a set of locations and kernel parameters.

Generates a covariance matrix using a set of locations and kernel parameters.

Parameters

out	*apMatrixA*	The output covariance matrix.
in	*aRowsNumber*	The number of rows in the output matrix.
in	*aColumnsNumber*	The number of columns in the output matrix.
in	*aRowOffset*	The row offset for the input locations.
in	*aColumnOffset*	The column offset for the input locations.
in	*apLocation1*	The set of input locations 1.
in	*apLocation2*	The set of input locations 2.
in	*apLocation3*	The set of input locations 3.

Returns

void

Implements exageostat::kernels::Kernel< T >.

8.76.3.2 Create()

```
template<typename T >
static Kernel<T>* exageostat::kernels::UnivariateMaternNonGaussian< T >::Create ( )  [static]
```

Creates a new UnivariateMaternNonGaussian object.

This method creates a new UnivariateMaternNonGaussian object and returns a pointer to it.

Returns

A pointer to the new UnivariateMaternNonGaussian object.

8.76.4 Member Data Documentation

8.76.4.1 plugin_name

```
template<typename T >
bool exageostat::kernels::UnivariateMaternNonGaussian< T >::plugin_name  [static], [private]
```

The documentation for this class was generated from the following file:

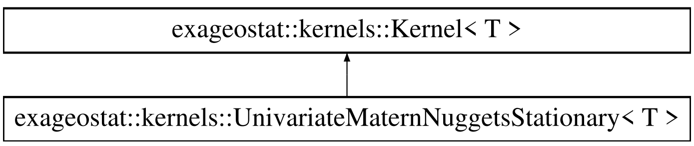
- UnivariateMaternNonGaussian.hpp

8.77 exageostat::kernels::UnivariateMaternNuggetsStationary< T > Class Template Reference

A class represents a Univariate Matern Nuggets Stationary kernel.

```
#include <UnivariateMaternNuggetsStationary.hpp>
```

Inheritance diagram for exageostat::kernels::UnivariateMaternNuggetsStationary< T >:



Public Member Functions

- UnivariateMaternNuggetsStationary ()

  Constructs a new UnivariateMaternNuggetsStationary object.
- ∼UnivariateMaternNuggetsStationary () override=default

  Virtual destructor to allow calls to the correct concrete destructor.
- void GenerateCovarianceMatrix (T ∗apMatrixA, const int &aRowsNumber, const int &aColumns←Number, const int &aRowOffset, const int &aColumnOffset, dataunits::Locations< T > &aLocation1, dataunits::Locations< T > &aLocation2, dataunits::Locations< T > &aLocation3, T ∗apLocalTheta, const int &aDistanceMetric) override

  Generates a covariance matrix using a set of locations and kernel parameters.

Static Public Member Functions

- static Kernel< T > ∗ Create ()

  Creates a new UnivariateMaternNuggetsStationary object.

Static Private Attributes

- static bool plugin_name

Additional Inherited Members

8.77.1 Detailed Description

template<typename T>
class exageostat::kernels::UnivariateMaternNuggetsStationary< T >

A class represents a Univariate Matern Nuggets Stationary kernel.

This class represents a Univariate Matern Nuggets Stationary, which is a subclass of the Kernel class. It provides a method for generating a covariance matrix using a set of input locations and kernel parameters.

8.77.2 Constructor & Destructor Documentation

8.77.2.1 UnivariateMaternNuggetsStationary()

```
template<typename T >
exageostat::kernels::UnivariateMaternNuggetsStationary< T >::UnivariateMaternNuggetsStationary
( )
```

Constructs a new UnivariateMaternNuggetsStationary object.

Initializes a new UnivariateMaternNuggetsStationary object with default values.

8.77.2.2 ∼UnivariateMaternNuggetsStationary()

```
template<typename T >
exageostat::kernels::UnivariateMaternNuggetsStationary< T >::∼UnivariateMaternNuggetsStationary
( ) [override], [default]
```

Virtual destructor to allow calls to the correct concrete destructor.

8.77.3 Member Function Documentation

8.77.3.1 GenerateCovarianceMatrix()

```
template<typename T >
void exageostat::kernels::UnivariateMaternNuggetsStationary< T >::GenerateCovarianceMatrix (
            T * apMatrixA,
            const int & aRowsNumber,
            const int & aColumnsNumber,
            const int & aRowOffset,
            const int & aColumnOffset,
            dataunits::Locations< T > & aLocation1,
            dataunits::Locations< T > & aLocation2,
            dataunits::Locations< T > & aLocation3,
            T * apLocalTheta,
            const int & aDistanceMetric ) [override], [virtual]
```

Generates a covariance matrix using a set of locations and kernel parameters.

Generates a covariance matrix using a set of locations and kernel parameters.

Parameters

out	*apMatrixA*	The output covariance matrix.
in	*aRowsNumber*	The number of rows in the output matrix.
in	*aColumnsNumber*	The number of columns in the output matrix.
in	*aRowOffset*	The row offset for the input locations.
in	*aColumnOffset*	The column offset for the input locations.
in	*apLocation1*	The set of input locations 1.
in	*apLocation2*	The set of input locations 2.
in	*apLocation3*	The set of input locations 3.
in	*aLocalTheta*	An array of kernel parameters.
in	*aDistanceMetric*	Distance metric to be used (1 = Euclidean, 2 = Manhattan, 3 = Minkowski).

Returns

void

Implements exageostat::kernels::Kernel< T >.

8.77.3.2 Create()

```
template<typename T >
static Kernel<T>* exageostat::kernels::UnivariateMaternNuggetsStationary< T >::Create ( )
[static]
```

Creates a new UnivariateMaternNuggetsStationary object.

This method creates a new UnivariateMaternNuggetsStationary object and returns a pointer to it.

Returns

A pointer to the new UnivariateMaternNuggetsStationary object.

8.77.4 Member Data Documentation

8.77.4.1 plugin_name

```
template<typename T >
bool exageostat::kernels::UnivariateMaternNuggetsStationary< T >::plugin_name [static], [private]
```

The documentation for this class was generated from the following file:

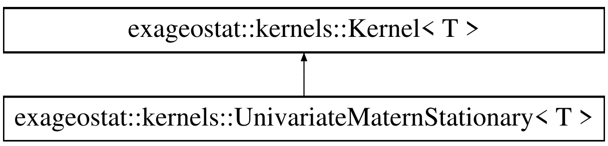
- UnivariateMaternNuggetsStationary.hpp

8.78 exageostat::kernels::UnivariateMaternStationary< T > Class Template Reference

A class represents a Univariate Matern Stationary kernel.

```
#include <UnivariateMaternStationary.hpp>
```

Inheritance diagram for exageostat::kernels::UnivariateMaternStationary< T >:



Public Member Functions

- UnivariateMaternStationary ()

  Constructs a new UnivariateMaternStationary object.
- ∼UnivariateMaternStationary () override=default

  Virtual destructor to allow calls to the correct concrete destructor.
- void GenerateCovarianceMatrix (T ∗apMatrixA, const int &aRowsNumber, const int &aColumns↩ Number, const int &aRowOffset, const int &aColumnOffset, dataunits::Locations< T > &aLocation1, dataunits::Locations< T > &aLocation2, dataunits::Locations< T > &aLocation3, T ∗apLocalTheta, const int &aDistanceMetric) override

  Generates a covariance matrix using a set of locations and kernel parameters.

Static Public Member Functions

- static Kernel< T > ∗ Create ()

    Creates a new UnivariateMaternStationary object.

Static Private Attributes

- static bool plugin_name

Additional Inherited Members

8.78.1 Detailed Description

template<typename T>
class exageostat::kernels::UnivariateMaternStationary< T >

A class represents a Univariate Matern Stationary kernel.

This class represents a Univariate Matern Stationary, which is a subclass of the Kernel class. It provides a method for generating a covariance matrix using a set of input locations and kernel parameters.

8.78.2 Constructor & Destructor Documentation

8.78.2.1 UnivariateMaternStationary()

```
template<typename T >
exageostat::kernels::UnivariateMaternStationary< T >::UnivariateMaternStationary ( )
```

Constructs a new UnivariateMaternStationary object.

Initializes a new UnivariateMaternStationary object with default values.

8.78.2.2 ∼UnivariateMaternStationary()

```
template<typename T >
exageostat::kernels::UnivariateMaternStationary< T >::~UnivariateMaternStationary ( ) [override],
[default]
```

Virtual destructor to allow calls to the correct concrete destructor.

8.78.3 Member Function Documentation

8.78.3.1 GenerateCovarianceMatrix()

```
template<typename T >
void exageostat::kernels::UnivariateMaternStationary< T >::GenerateCovarianceMatrix (
            T * apMatrixA,
            const int & aRowsNumber,
            const int & aColumnsNumber,
            const int & aRowOffset,
            const int & aColumnOffset,
            dataunits::Locations< T > & aLocation1,
            dataunits::Locations< T > & aLocation2,
            dataunits::Locations< T > & aLocation3,
            T * apLocalTheta,
            const int & aDistanceMetric )  [override], [virtual]
```

Generates a covariance matrix using a set of locations and kernel parameters.

Generates a covariance matrix using a set of locations and kernel parameters.

Parameters

out	*apMatrixA*	The output covariance matrix.
in	*aRowsNumber*	The number of rows in the output matrix.
in	*aColumnsNumber*	The number of columns in the output matrix.
in	*aRowOffset*	The row offset for the input locations.
in	*aColumnOffset*	The column offset for the input locations.
in	*apLocation1*	The set of input locations 1.
in	*apLocation2*	The set of input locations 2.
in	*apLocation3*	The set of input locations 3.
in	*aLocalTheta*	An array of kernel parameters.
in	*aDistanceMetric*	Distance metric to be used (1 = Euclidean, 2 = Manhattan, 3 = Minkowski).

Returns

void

Implements exageostat::kernels::Kernel< T >.

8.78.3.2 Create()

```
template<typename T >
static Kernel<T>* exageostat::kernels::UnivariateMaternStationary< T >::Create ( )  [static]
```

Creates a new UnivariateMaternStationary object.

This method creates a new UnivariateMaternStationary object and returns a pointer to it.

Returns

A pointer to the new UnivariateMaternStationary object.

8.78.4 Member Data Documentation

8.78.4.1 plugin_name

```
template<typename T >
bool exageostat::kernels::UnivariateMaternStationary< T >::plugin_name [static], [private]
```

The documentation for this class was generated from the following file:

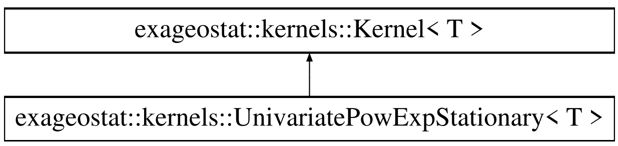
- UnivariateMaternStationary.hpp

8.79 exageostat::kernels::UnivariatePowExpStationary< T > Class Template Reference

A class represents a Univariate PowExp Stationary kernel.

```
#include <UnivariatePowExpStationary.hpp>
```

Inheritance diagram for exageostat::kernels::UnivariatePowExpStationary< T >:



Public Member Functions

- UnivariatePowExpStationary ()

  Constructs a new UnivariatePowExpStationary object.
- ∼UnivariatePowExpStationary () override=default

  Virtual destructor to allow calls to the correct concrete destructor.
- void GenerateCovarianceMatrix (T ∗apMatrixA, const int &aRowsNumber, const int &aColumns←Number, const int &aRowOffset, const int &aColumnOffset, dataunits::Locations< T > &aLocation1, dataunits::Locations< T > &aLocation2, dataunits::Locations< T > &aLocation3, T ∗apLocalTheta, const int &aDistanceMetric) override

  Generates a covariance matrix using a set of locations and kernel parameters.

Static Public Member Functions

- static Kernel< T > ∗ Create ()

  Creates a new UnivariatePowExpStationary object.

Static Private Attributes

- static bool plugin_name

Additional Inherited Members

8.79.1 Detailed Description

template<typename T>
class exageostat::kernels::UnivariatePowExpStationary< T >

A class represents a Univariate PowExp Stationary kernel.

This class represents a Univariate PowExp Stationary, which is a subclass of the Kernel class. It provides a method for generating a covariance matrix using a set of input locations and kernel parameters.

8.79.2 Constructor & Destructor Documentation

8.79.2.1 UnivariatePowExpStationary()

```
template<typename T >
exageostat::kernels::UnivariatePowExpStationary< T >::UnivariatePowExpStationary ( )
```

Constructs a new UnivariatePowExpStationary object.

Initializes a new UnivariatePowExpStationary object with default values.

8.79.2.2 ∼UnivariatePowExpStationary()

```
template<typename T >
exageostat::kernels::UnivariatePowExpStationary< T >::∼UnivariatePowExpStationary ( ) [override],
[default]
```

Virtual destructor to allow calls to the correct concrete destructor.

8.79.3 Member Function Documentation

8.79.3.1 GenerateCovarianceMatrix()

```
template<typename T >
void exageostat::kernels::UnivariatePowExpStationary< T >::GenerateCovarianceMatrix (
            T * apMatrixA,
            const int & aRowsNumber,
            const int & aColumnsNumber,
            const int & aRowOffset,
            const int & aColumnOffset,
            dataunits::Locations< T > & aLocation1,
            dataunits::Locations< T > & aLocation2,
            dataunits::Locations< T > & aLocation3,
            T * apLocalTheta,
            const int & aDistanceMetric )  [override], [virtual]
```

Generates a covariance matrix using a set of locations and kernel parameters.

Generates a covariance matrix using a set of locations and kernel parameters.

Parameters

out	*apMatrixA*	The output covariance matrix.
in	*aRowsNumber*	The number of rows in the output matrix.
in	*aColumnsNumber*	The number of columns in the output matrix.
in	*aRowOffset*	The row offset for the input locations.
in	*aColumnOffset*	The column offset for the input locations.
in	*apLocation1*	The set of input locations 1.
in	*apLocation2*	The set of input locations 2.
in	*apLocation3*	The set of input locations 3.
in	*aLocalTheta*	An array of kernel parameters.
in	*aDistanceMetric*	Distance metric to be used (1 = Euclidean, 2 = Manhattan, 3 = Minkowski).

Returns

void

Implements exageostat::kernels::Kernel< T >.

8.79.3.2 Create()

```
template<typename T >
static Kernel<T>* exageostat::kernels::UnivariatePowExpStationary< T >::Create ( ) [static]
```

Creates a new UnivariatePowExpStationary object.

This method creates a new UnivariatePowExpStationary object and returns a pointer to it.

Returns

A pointer to the new UnivariatePowExpStationary object.

8.79.4 Member Data Documentation

8.79.4.1 plugin_name

```
template<typename T >
bool exageostat::kernels::UnivariatePowExpStationary< T >::plugin_name [static], [private]
```

The documentation for this class was generated from the following file:

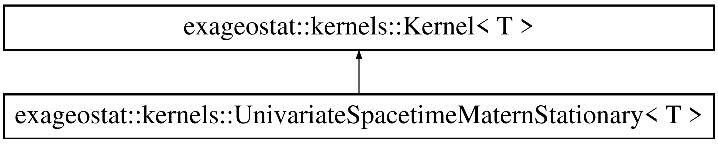
- UnivariatePowExpStationary.hpp

8.80 exageostat::kernels::UnivariateSpacetimeMaternStationary< T > Class Template Reference

A class represents a Univariate Spacetime Matern Stationary kernel.

```
#include <UnivariateSpacetimeMaternStationary.hpp>
```

Inheritance diagram for exageostat::kernels::UnivariateSpacetimeMaternStationary< T >:



Public Member Functions

- UnivariateSpacetimeMaternStationary ()

  Constructs a new UnivariateSpacetimeMaternStationary object.
- ∼UnivariateSpacetimeMaternStationary () override=default

  Virtual destructor to allow calls to the correct concrete destructor.
- void GenerateCovarianceMatrix (T ∗apMatrixA, const int &aRowsNumber, const int &aColumnsNumber, const int &aRowOffset, const int &aColumnOffset, dataunits::Locations< T > &aLocation1, dataunits::Locations< T > &aLocation2, dataunits::Locations< T > &aLocation3, T ∗apLocalTheta, const int &aDistanceMetric) override

  Generates a covariance matrix using a set of locations and kernel parameters.

Static Public Member Functions

- static Kernel< T > ∗ Create ()

  Creates a new UnivariateSpacetimeMaternStationary object.

Static Private Attributes

- static bool plugin_name

Additional Inherited Members

8.80.1 Detailed Description

template<typename T>
class exageostat::kernels::UnivariateSpacetimeMaternStationary< T >

A class represents a Univariate Spacetime Matern Stationary kernel.

This class represents a Univariate Spacetime Matern Stationary, which is a subclass of the Kernel class. It provides a method for generating a covariance matrix using a set of input locations and kernel parameters.

8.80.2 Constructor & Destructor Documentation

8.80.2.1 UnivariateSpacetimeMaternStationary()

```
template<typename T >
exageostat::kernels::UnivariateSpacetimeMaternStationary< T >::UnivariateSpacetimeMaternStationary
( )
```

Constructs a new UnivariateSpacetimeMaternStationary object.

Initializes a new UnivariateSpacetimeMaternStationary object with default values.

8.80.2.2 ∼UnivariateSpacetimeMaternStationary()

```
template<typename T >
exageostat::kernels::UnivariateSpacetimeMaternStationary< T >::∼UnivariateSpacetimeMaternStationary
( ) [override], [default]
```

Virtual destructor to allow calls to the correct concrete destructor.

8.80.3 Member Function Documentation

8.80.3.1 GenerateCovarianceMatrix()

```
template<typename T >
void exageostat::kernels::UnivariateSpacetimeMaternStationary< T >::GenerateCovarianceMatrix (
            T * apMatrixA,
            const int & aRowsNumber,
            const int & aColumnsNumber,
            const int & aRowOffset,
            const int & aColumnOffset,
            dataunits::Locations< T > & aLocation1,
            dataunits::Locations< T > & aLocation2,
            dataunits::Locations< T > & aLocation3,
            T * apLocalTheta,
            const int & aDistanceMetric ) [override], [virtual]
```

Generates a covariance matrix using a set of locations and kernel parameters.

Generates a covariance matrix using a set of locations and kernel parameters.

Parameters

out	*apMatrixA*	The output covariance matrix.
in	*aRowsNumber*	The number of rows in the output matrix.
in	*aColumnsNumber*	The number of columns in the output matrix.
in	*aRowOffset*	The row offset for the input locations.
in	*aColumnOffset*	The column offset for the input locations.
in	*apLocation1*	The set of input locations 1.
in	*apLocation2*	The set of input locations 2.

Returns

void

Implements exageostat::kernels::Kernel< T >.

8.80.3.2 Create()

```
template<typename T >
static Kernel<T>* exageostat::kernels::UnivariateSpacetimeMaternStationary< T >::Create ( )
[static]
```

Creates a new UnivariateSpacetimeMaternStationary object.

This method creates a new UnivariateSpacetimeMaternStationary object and returns a pointer to it.

Returns

A pointer to the new UnivariateSpacetimeMaternStationary object.

8.80.4 Member Data Documentation

8.80.4.1 plugin_name

```
template<typename T >
bool exageostat::kernels::UnivariateSpacetimeMaternStationary< T >::plugin_name [static],
[private]
```

The documentation for this class was generated from the following file:

- UnivariateSpacetimeMaternStationary.hpp

Chapter 9

File Documentation

9.1 BasselFunction.hpp File Reference

This file contains the BasselFunction class which provides methods for computing derivatives of the modified Bessel function of the second kind. These functions are crucial in statistical and mathematical computations, especially in fields such as geostatistics and spatial analysis.

```
#include <common/Definitions.hpp>
```

Classes

- class exageostat::helpers::BasselFunction< T >

  The BasselFunction class provides methods for computing various derivatives of the modified Bessel function of the second kind, (K_{\nu}). This class is templated to support both float and double data types, enabling precision-based computations as required by different applications.

Namespaces

- exageostat
- exageostat::helpers

9.1.1 Detailed Description

This file contains the BasselFunction class which provides methods for computing derivatives of the modified Bessel function of the second kind. These functions are crucial in statistical and mathematical computations, especially in fields such as geostatistics and spatial analysis.

Version

1.1.0

Author

Mahmoud ElKarargy

Sameh Abdulah

Date

2023-01-24

9.2 BivariateMaternFlexible.hpp File Reference

Defines the BivariateMaternFlexible class, a Bivariate Matern Flexible kernel.

```
#include <kernels/Kernel.hpp>
```

Classes

- class exageostat::kernels::BivariateMaternFlexible< T >
  A class represents a Bivariate Matern Flexible kernel.

Namespaces

- exageostat
- exageostat::kernels

9.2.1 Detailed Description

Defines the BivariateMaternFlexible class, a Bivariate Matern Flexible kernel.

Version

1.1.0

Author

Mahmoud ElKarargy

Sameh Abdulah

Suhas Shankar

Mary Lai Salvana

Date

2023-04-14

9.3 BivariateMaternParsimonious.hpp File Reference

Defines the BivariateMaternParsimonious class, a Bivariate Matern Parsimonious kernel.

```
#include <kernels/Kernel.hpp>
```

Classes

- class exageostat::kernels::BivariateMaternParsimonious< T >
  A class represents a Bivariate Matern Parsimonious kernel.

Namespaces

- exageostat
- exageostat::kernels

9.3.1 Detailed Description

Defines the BivariateMaternParsimonious class, a Bivariate Matern Parsimonious kernel.

Version

1.1.0

Author

Mahmoud ElKarargy

Sameh Abdulah

Suhas Shankar

Mary Lai Salvana

Date

2023-04-14

9.4 BivariateSpacetimeMaternStationary.hpp File Reference

Defines the BivariateSpacetimeMaternStationary class, a Bivariate Spacetime Matern Stationary kernel.

```
#include <kernels/Kernel.hpp>
```

Classes

- class exageostat::kernels::BivariateSpacetimeMaternStationary< T >

  A class represents a Bivariate Spacetime Matern Stationary kernel.

Namespaces

- exageostat
- exageostat::kernels

9.4.1 Detailed Description

Defines the BivariateSpacetimeMaternStationary class, a Bivariate Spacetime Matern Stationary kernel.

Version

1.1.0

Author

Mahmoud ElKarargy

Sameh Abdulah

Suhas Shankar

Mary Lai Salvana

Date

2023-04-14

9.5 ByteHandler.hpp File Reference

Implementation of byte manipulation functions for ExaGeoStat.

```
#include <memory>
```

Namespaces

- exageostat
- exageostat::helpers

Functions

- uint64_t exageostat::helpers::SpreadBits (uint64_t aInputByte)

    Spread bits by three spaces.
- uint64_t exageostat::helpers::ReverseSpreadBits (uint64_t aInputByte)

    Reverse Spread bits operation.
- bool exageostat::helpers::CompareUint64 (const uint64_t &aFirstValue, const uint64_t &aSecondValue)

    Compares two Unit64 values.

9.5.1 Detailed Description

Implementation of byte manipulation functions for ExaGeoStat.

Version

1.1.0

Author

Mahmoud ElKarargy

Sameh Abdulah

Date

2024-01-24

9.6 ChameleonDense.hpp File Reference

```
#include <linear-algebra-solvers/concrete/chameleon/ChameleonImplementation.↩
hpp>
```

Classes

- class exageostat::linearAlgebra::dense::ChameleonDense< T >

  ChameleonImplementationDense is a concrete implementation for dense matrices using Chameleon..

Namespaces

- exageostat
- exageostat::linearAlgebra
- exageostat::linearAlgebra::dense

9.7 ChameleonDescriptor.hpp File Reference

Defines the ChameleonDescriptor class for creating matrix descriptors using the CHAMELEON library.

```
#include <linear-algebra-solvers/concrete/ChameleonHeaders.hpp>
#include <common/Definitions.hpp>
```

Classes

- class exageostat::dataunits::descriptor::ChameleonDescriptor< T >

  ChameleonDescriptor is a class for creating matrix descriptors by CHAMELEON library.

Namespaces

- exageostat
- exageostat::dataunits
- exageostat::dataunits::descriptor

9.7.1 Detailed Description

Defines the ChameleonDescriptor class for creating matrix descriptors using the CHAMELEON library.

Version

1.1.0

Author

Mahmoud ElKarargy

Sameh Abdulah

Date

2023-08-15

9.8 ChameleonDST.hpp File Reference

```
#include <linear-algebra-solvers/concrete/chameleon/ChameleonImplementation.↵
hpp>
```

Classes

- class exageostat::linearAlgebra::diagonalSuperTile::ChameleonDST< T >
  ChameleonImplementationDST is a concrete implementation of LinearAlgebraMethods class for diagonal super tile matrices.

Namespaces

- exageostat
- exageostat::linearAlgebra
- exageostat::linearAlgebra::diagonalSuperTile

9.9 ChameleonHeaders.hpp File Reference

This file contains the necessary includes for using the Chameleon library.

```
#include <chameleon.h>
#include <control/async.h>
#include <control/descriptor.h>
#include <control/context.h>
#include <include/chameleon/flops.h>
#include <coreblas.h>
```

9.9.1 Detailed Description

This file contains the necessary includes for using the Chameleon library.

Version

1.1.0

Author

Mahmoud ElKarargy

Date

2023-08-24

9.10 ChameleonImplementation.hpp File Reference

This file contains the declaration of ChameleonImplementation class.

```
#include <linear-algebra-solvers/LinearAlgebraMethods.hpp>
```

Classes

- class exageostat::linearAlgebra::ChameleonImplementation$< T >$

  ChameleonImplementation is a concrete implementation of LinearAlgebraMethods class for dense or diagonal-super tile matrices.

Namespaces

- exageostat
- exageostat::linearAlgebra

9.10.1 Detailed Description

This file contains the declaration of ChameleonImplementation class.

ChameleonImplementation is a concrete implementation of the LinearAlgebraMethods class for the common functionality implementation shared between dense and diagonal-super tile matrices.

Version

1.1.0

Author

Mahmoud ElKarargy

Sameh Abdulah

Date

2023-03-20

9.11 ChameleonStarPuHelpers.hpp File Reference

A class for Chameleon implementation of StarPu helpers interface StarPuHelpers.hpp.

```
#include <runtime/starpu/helpers/StarPuHelpers.hpp>
```

Classes

- class exageostat::runtime::ChameleonStarPuHelpers
  ChameleonStarPuHelpers is a concrete implementation of StarPuHelpers interface for Chameleon library.

Namespaces

- exageostat
- exageostat::runtime

9.11.1 Detailed Description

A class for Chameleon implementation of StarPu helpers interface StarPuHelpers.hpp.

Version

1.1.0

Author

Mahmoud ElKarargy

Date

2024-02-25

9.12 CommunicatorMPI.hpp File Reference

Defines the CommunicatorMPI class for MPI rank communication.

Classes

- class exageostat::helpers::CommunicatorMPI
  A class for Communicating MPI rank.

Namespaces

- exageostat
- exageostat::helpers

9.12.1 Detailed Description

Defines the CommunicatorMPI class for MPI rank communication.

Version

1.1.0

Author

Sameh Abdulah

Date

2023-11-10

9.13 Configurations.hpp File Reference

Contains the declaration of the Configurations class and its member functions.

```
#include <vector>
#include <unordered_map>
#include <any>
#include <common/Definitions.hpp>
```

Classes

- class exageostat::configurations::Configurations

  Contains methods to set and get.

Namespaces

- exageostat
- exageostat::configurations

Macros

- #define CREATE_SETTER_FUNCTION(name, type, argument_name, dictionary_name)

  Macro that generates a setter function for a member variable.
- #define CREATE_GETTER_FUNCTION(name, type, dictionary_name)

  Macro that generates a getter function for a member variable.

9.13.1 Detailed Description

Contains the declaration of the Configurations class and its member functions.

Version

1.1.0

Author

Mahmoud ElKarargy

Sameh Abdulah

Date

2024-02-04

9.13.2 Macro Definition Documentation

9.13.2.1 CREATE_SETTER_FUNCTION

```
#define CREATE_SETTER_FUNCTION(
            name,
            type,
            argument_name,
            dictionary_name )
```

Value:

```
void Set##name(type argument_name)                                  \
{                                                                   \
    mDictionary[dictionary_name] = argument_name;                   \
}
```

Macro that generates a setter function for a member variable.

This macro generates a function named Set##name that takes an argument of the specified type and sets the member variable with the specified name to the value of the argument. The name of the member variable is used as the key to set the corresponding value in the specified dictionary.

Parameters

in	*name*	The name of the member variable to be set.
in	*type*	The data type of the member variable.
in	*argument_name*	The name of the argument to the generated function.
in	*dictionary_name*	The name of the dictionary to set the value in.

9.13.2.2 CREATE_GETTER_FUNCTION

```
#define CREATE_GETTER_FUNCTION(
              name,
              type,
              dictionary_name )
```

Value:

```
type Get##name()
      \
{
      \
    if (mDictionary.find(dictionary_name) == mDictionary.end()) {
      \
       throw std::range_error(std::string("Argument ").append(dictionary_name).append(" is not set!"));
      \
    }
      \
    return std::any_cast<type>(mDictionary[dictionary_name]);
      \
}
```

Macro that generates a getter function for a member variable.

This macro generates a function named Get##name that returns the value of the member variable with the specified name from the specified dictionary.

Parameters

in	*name*	The name of the member variable to be retrieved.
in	*type*	The data type of the member variable.
in	*dictionary_name*	The name of the dictionary to retrieve the value from.

9.14 CSVLoader.hpp File Reference

A class for generating synthetic data.

```
#include <data-loader/DataLoader.hpp>
```

Classes

- class exageostat::dataLoader::csv::CSVLoader< T >

  A class for creating data by reading CSV files.

Namespaces

- exageostat
- exageostat::dataLoader
- exageostat::dataLoader::csv

9.14.1 Detailed Description

A class for generating synthetic data.

Version

1.1.0

Author

Mahmoud ElKarargy

Sameh Abdulah

Date

2024-02-04

9.15 DataGenerator.hpp File Reference

Contains definition for abstract Data Generator Class.

```
#include <linear-algebra-solvers/LinearAlgebraFactory.hpp>
#include <linear-algebra-solvers/LinearAlgebraMethods.hpp>
```

Classes

- class exageostat::generators::DataGenerator< T >

  Abstract base class for generating synthetic or real data.

Namespaces

- exageostat
- exageostat::generators

9.15.1 Detailed Description

Contains definition for abstract Data Generator Class.

Version

1.1.0

Author

Mahmoud ElKarargy

Date

2023-02-14

9.16 DataLoader.hpp File Reference

Manages data loading operations for ExaGeoStat.

```
#include <data-generators/DataGenerator.hpp>
```

Classes

- class exageostat::dataLoader::DataLoader< T >
  Extends DataGenerator to include data loading functionalities.

Namespaces

- exageostat
- exageostat::dataLoader

9.16.1 Detailed Description

Manages data loading operations for ExaGeoStat.

Version

1.1.0

Author

Mahmoud ElKarargy

Sameh Abdulah

Date

2024-02-04

9.17 dcmg-codelet.hpp File Reference

A class for starpu codelet dcmg.

```
#include <kernels/Kernel.hpp>
```

Classes

- class exageostat::runtime::DCMGCodelet< T >

Namespaces

- exageostat
- exageostat::runtime

9.17.1 Detailed Description

A class for starpu codelet dcmg.

Version

1.1.0

Author

Mahmoud ElKarargy

Sameh Abdulah

Date

2024-02-19

9.18 ddotp-codelet.hpp File Reference

A class for starpu codelet ddotp.

`#include <common/Definitions.hpp>`

Classes

- class exageostat::runtime::DDOTPCodelet< T >

Namespaces

- exageostat
- exageostat::runtime

9.18.1 Detailed Description

A class for starpu codelet ddotp.

Version

1.1.0

Author

Mahmoud ElKarargy

Sameh Abdulah

Date

2024-02-25

9.19 Definitions.hpp File Reference

This file contains common definitions used in ExaGeoStat software package.

```
#include <set>
#include <filesystem>
```

Namespaces

- exageostat
- exageostat::common

Macros

- #define EXAGEOSTAT_INSTANTIATE_CLASS(TEMPLATE_CLASS)

  Macro definition to instantiate the EXAGEOSTAT template classes with supported types.
- #define SIZE_OF_FLOAT 4
- #define SIZE_OF_DOUBLE 8
- #define PI (3.141592653589793)
- #define EARTH_RADIUS 6371.0
- #define Q_NORM 1.959964
- #define KERNELS_PATH PROJECT_SOURCE_DIR "/inst/include/kernels/concrete/"
- #define LOG_PATH PROJECT_SOURCE_DIR "/synthetic_ds/"

Typedefs

- typedef enum exageostat::common::TileStorage exageostat::common::ExaGeoStatTileStorage

Enumerations

- enum exageostat::common::Verbose { exageostat::common::QUIET_MODE = 0 , exageostat::common::STANDARD_MODE = 1 , exageostat::common::DETAILED_MODE = 2 }
- enum exageostat::common::Dimension { exageostat::common::Dimension2D = 0 , exageostat::common::Dimension3D = 1 , exageostat::common::DimensionST = 2 }

  Enum denoting the dimension of generated data.
- enum exageostat::common::Side { exageostat::common::EXAGEOSTAT_LEFT = 141 , exageostat::common::EXAGEOSTAT_RI = 142 }

  Enum denoting the side on which the matrix appears in an equation.
- enum exageostat::common::Trans { exageostat::common::EXAGEOSTAT_NO_TRANS = 111 , exageostat::common::EXAGEOS = 112 , exageostat::common::EXAGEOSTAT_CONJ_TRANS = 113 }

  Enum denoting whether or not to transpose a matrix.
- enum exageostat::common::Diag { exageostat::common::EXAGEOSTAT_NON_UNIT = 131 , exageostat::common::EXAGEOST = 132 }

  Enum denoting whether the diagonal is unitary.
- enum exageostat::common::DistanceMetric { exageostat::common::EUCLIDEAN_DISTANCE = 0 , exageostat::common::GREAT_CIRCLE_DISTANCE = 1 }
- enum exageostat::common::DescriptorType { exageostat::common::CHAMELEON_DESCRIPTOR = 0 , exageostat::common::HICMA_DESCRIPTOR = 1 }

- enum exageostat::common::DataSourceType { exageostat::common::SYNTHETIC = 0 , exageostat::common::CSV_FILE = 1 }
- enum exageostat::common::DescriptorName : int {
exageostat::common::DESCRIPTOR_C = 0 , exageostat::common::DESCRIPTOR_Z = 1 , exageostat::common::DESCRIPTOF = 2 , exageostat::common::DESCRIPTOR_PRODUCT = 3 ,
exageostat::common::DESCRIPTOR_DETERMINANT = 4 , exageostat::common::DESCRIPTOR_CD = 5 , exageostat::common::DESCRIPTOR_CUV = 6 , exageostat::common::DESCRIPTOR_CRK = 7 , exageostat::common::DESCRIPTOR_Z_OBSERVATIONS = 8 , exageostat::common::DESCRIPTOR_Z_Actual = 9 , exageostat::common::DESCRIPTOR_Z_MISS = 10 , exageostat::common::DESCRIPTOR_MSPE = 11 ,
exageostat::common::DESCRIPTOR_Z_1 = 12 , exageostat::common::DESCRIPTOR_Z_2 = 13 , exageostat::common::DESCRIPTOR_Z_3 = 14 , exageostat::common::DESCRIPTOR_PRODUCT_1 = 15 ,
exageostat::common::DESCRIPTOR_PRODUCT_2 = 16 , exageostat::common::DESCRIPTOR_PRODUCT_3 = 17 , exageostat::common::DESCRIPTOR_C11 = 18 , exageostat::common::DESCRIPTOR_C12 = 19 , exageostat::common::DESCRIPTOR_C22 = 20 , exageostat::common::DESCRIPTOR_C12D = 21 , exageostat::common::DESCRIPTOR_C12UV = 22 , exageostat::common::DESCRIPTOR_C12RK = 23 ,
exageostat::common::DESCRIPTOR_C22D = 24 , exageostat::common::DESCRIPTOR_C22UV = 25 , exageostat::common::DESCRIPTOR_C22RK = 26 , exageostat::common::DESCRIPTOR_MSPE_1 = 27 , exageostat::common::DESCRIPTOR_MSPE_2 = 28 , exageostat::common::DESCRIPTOR_k_T = 29 , exageostat::common::DESCRIPTOR_k_A = 30 , exageostat::common::DESCRIPTOR_k_A_TMP = 31 , exageostat::common::DESCRIPTOR_k_T_TMP = 32 , exageostat::common::DESCRIPTOR_K_T = 33 , exageostat::common::DESCRIPTOR_K_T_TMP = 34 , exageostat::common::DESCRIPTOR_K_A = 35 , exageostat::common::DESCRIPTOR_EXPR_1 = 36 , exageostat::common::DESCRIPTOR_EXPR_2 = 37 , exageostat::common::DESCRIPTOR_EXPR_3 = 38 , exageostat::common::DESCRIPTOR_EXPR_4 = 39 , exageostat::common::DESCRIPTOR_MLOE = 40 , exageostat::common::DESCRIPTOR_MMOM = 41 , exageostat::common::DESCRIPTOR_MLOE_MMOM = 42 , exageostat::common::DESCRIPTOR_ALPHA = 43 ,
exageostat::common::DESCRIPTOR_TRUTH_ALPHA = 44 , exageostat::common::DESCRIPTOR_TIMATED_ALPHA = 45 , exageostat::common::DESCRIPTOR_CK = 46 , exageostat::common::DESCRIPTOR_CJ = 47 , exageostat::common::DESCRIPTOR_C_TRACE = 48 , exageostat::common::DESCRIPTOR_C_DIAG = 49 , exageostat::common::DESCRIPTOR_A = 50 , exageostat::common::DESCRIPTOR_RESULTS = 51 , exageostat::common::DESCRIPTOR_SUM = 52 , exageostat::common::DESCRIPTOR_R = 53 , exageostat::common::DESCRIPTOR_R_COPY = 54 }
- enum exageostat::common::TileStorage {
exageostat::common::EXAGOSTAT_CM = 101 , exageostat::common::EXAGOSTAT_RM = 102 , exageostat::common::EXAGOSTAT_CCRB = 103 , exageostat::common::EXAGOSTAT_CRRB = 104 , exageostat::common::EXAGOSTAT_RCRB = 105 , exageostat::common::EXAGOSTAT_RRRB = 106 }
- enum exageostat::common::Computation { exageostat::common::EXACT_DENSE = 0 , exageostat::common::DIAGONAL_APP = 1 , exageostat::common::TILE_LOW_RANK = 2 }

  Enum denoting the types of computations that can be requested, to use the required Linear Algebra solver library.
- enum exageostat::common::Precision { exageostat::common::SINGLE = 0 , exageostat::common::DOUBLE = 1 , exageostat::common::MIXED = 2 }

  Enum denoting the precision of operations that are supported to be done on the matrix.
- enum exageostat::common::FloatPoint : int {
exageostat::common::EXAGEOSTAT_BYTE = 0 , exageostat::common::EXAGEOSTAT_INTEGER = 1 , exageostat::common::EXAGEOSTAT_REAL_FLOAT = 2 , exageostat::common::EXAGEOSTAT_REAL_DOUBLE = 3 ,
exageostat::common::EXAGEOSTAT_COMPLEX_FLOAT = 4 , exageostat::common::EXAGEOSTAT_COMPLEX_DOUBLE = 5 }

  Enum denoting the floating point arithmetic of the matrix.
- enum exageostat::common::UpperLower : int { exageostat::common::EXAGEOSTAT_UPPER = 121 , exageostat::common::EXAGEOSTAT_LOWER = 122 , exageostat::common::EXAGEOSTAT_UPPER_LOWER = 123 }

  Enum denoting the Upper/Lower part.

- enum exageostat::common::CopyDirection : int { exageostat::common::CHAMELEON_TO_HICMA = 0 , exageostat::common::HICMA_TO_CHAMELEON = 1 }

  Enum denoting the copy descriptors flow.

Variables

- static const std::set< std::string > exageostat::common::availableKernels

  Set denoting the available kernels supported in matrix generation.

9.19.1 Detailed Description

This file contains common definitions used in ExaGeoStat software package.

Version

1.1.0

These definitions include enums for dimension, computation, precision, and floating point arithmetic; A macro for instantiating template classes with supported types; and a set of available kernels.

Author

Mahmoud ElKarargy

Sameh Abdulah

Date

2023-03-21

9.19.2 Macro Definition Documentation

9.19.2.1 EXAGEOSTAT_INSTANTIATE_CLASS

```
#define EXAGEOSTAT_INSTANTIATE_CLASS(
            TEMPLATE_CLASS )
```

Value:

```
template class TEMPLATE_CLASS<float>;  \
template class TEMPLATE_CLASS<double>;
```

Macro definition to instantiate the EXAGEOSTAT template classes with supported types.

9.19.2.2 SIZE_OF_FLOAT

```
#define SIZE_OF_FLOAT 4
```

9.19.2.3 SIZE_OF_DOUBLE

```
#define SIZE_OF_DOUBLE 8
```

9.19.2.4 PI

```
#define PI (3.141592653589793)
```

Pi value.

9.19.2.5 EARTH_RADIUS

```
#define EARTH_RADIUS 6371.0
```

Earth Radius value.

9.19.2.6 Q_NORM

```
#define Q_NORM 1.959964
```

Q Norm value.

9.19.2.7 KERNELS_PATH

```
#define KERNELS_PATH PROJECT_SOURCE_DIR "/inst/include/kernels/concrete/"
```

Kernel Files Path Definition

9.19.2.8 LOG_PATH

```
#define LOG_PATH PROJECT_SOURCE_DIR "/synthetic_ds/"
```

Logging Path Definition

9.20 DescriptorData.hpp File Reference

Contains the definition of the DescriptorData class.

```
#include <unordered_map>
#include <data-units/descriptor/ExaGeoStatDescriptor.hpp>
#include <hardware/ExaGeoStatHardware.hpp>
```

Classes

- union exageostat::dataunits::BaseDescriptor

  Union represents the base descriptor.
- class exageostat::dataunits::DescriptorData< T >

  Manages geo-statistical descriptor data with functions for retrieving and manipulating descriptors.

Namespaces

- exageostat
- exageostat::dataunits

9.20.1 Detailed Description

Contains the definition of the DescriptorData class.

Version

1.1.0

Author

Mahmoud ElKarargy

Sameh Abdulah

Date

2023-07-18

9.21 DistanceCalculationHelpers.hpp File Reference

Contains the definition of the DistanceCalculationHelpers class.

```
#include <data-units/Locations.hpp>
```

Classes

- class exageostat::helpers::DistanceCalculationHelpers< T >

  Class to calculate the distance between two points.

Namespaces

- exageostat
- exageostat::helpers

9.21.1 Detailed Description

Contains the definition of the DistanceCalculationHelpers class.

Version

1.1.0

Author

Mahmoud ElKarargy

Sameh Abdulah

Date

2023-06-08

9.22 dmdet-codelet.hpp File Reference

A class for starpu codelet dmdet.

```
#include <runtime/starpu/helpers/StarPuHelpers.hpp>
```

Classes

- class exageostat::runtime::DMDETCodelet< T >

Namespaces

- exageostat
- exageostat::runtime

9.22.1 Detailed Description

A class for starpu codelet dmdet.

Version

1.1.0

Author

Mahmoud ElKarargy

Sameh Abdulah

Date

2024-02-21

9.23 dmloe-mmom-codelet.hpp File Reference

A class for starpu codelet dmloe-mmom.

`#include <common/Definitions.hpp>`

Classes

- class exageostat::runtime::DmloeMmomCodelet< T >

Namespaces

- exageostat
- exageostat::runtime

9.23.1 Detailed Description

A class for starpu codelet dmloe-mmom.

Version

1.1.0

Author

Mahmoud ElKarargy

Sameh Abdulah

Date

2024-02-19

9.24 dmse-bivariate-codelet.hpp File Reference

A class for starpu codelet dmse-bivariate.

```
#include <common/Definitions.hpp>
```

Classes

- class exageostat::runtime::DMSEBivariateCodelet< T >

Namespaces

- exageostat
- exageostat::runtime

9.24.1 Detailed Description

A class for starpu codelet dmse-bivariate.

Version

1.1.0

Author

Mahmoud ElKarargy

Sameh Abdulah

Date

2024-02-25

9.25 dmse-codelet.hpp File Reference

A class for starpu codelet dmse.

```
#include <common/Definitions.hpp>
```

Classes

- class exageostat::runtime::DMSECodelet< T >

Namespaces

- exageostat
- exageostat::runtime

9.25.1 Detailed Description

A class for starpu codelet dmse.

Version

1.1.0

Author

Mahmoud ElKarargy
Sameh Abdulah

Date

2024-02-21

9.26 dtrace-codelet.hpp File Reference

A class for starpu codelet dtrace.

```
#include <common/Definitions.hpp>
```

Classes

- class exageostat::runtime::DTRACECodelet< T >

Namespaces

- exageostat
- exageostat::runtime

9.26.1 Detailed Description

A class for starpu codelet dtrace.

Version

1.1.0

Author

Mahmoud ElKarargy
Sameh Abdulah

Date

2024-02-25

9.27 dzcpy-codelet.hpp File Reference

A class for starpu codelet dzcpy.

```
#include <common/Definitions.hpp>
```

Classes

- class exageostat::runtime::DZCPYCodelet< T >

Namespaces

- exageostat
- exageostat::runtime

9.27.1 Detailed Description

A class for starpu codelet dzcpy.

Version

1.1.0

Author

Mahmoud ElKarargy

Sameh Abdulah

Date

2024-02-25

9.28 EnumStringParser.hpp File Reference

Provides utility functions for parsing enumeration values from strings.

```
#include <algorithm>
#include <utilities/ErrorHandler.hpp>
#include <common/Definitions.hpp>
```

Functions

- exageostat::common::Computation GetInputComputation (std::string aComputation)

  Convert a string representation of computation mode to its corresponding enum value.
- exageostat::common::Dimension GetInputDimension (std::string aDimension)

  Converts string to dimension enum.

9.28.1 Detailed Description

Provides utility functions for parsing enumeration values from strings.

Version

1.1.0

Author

Mahmoud ElKarargy

Date

2024-01-20

9.28.2 Function Documentation

9.28.2.1 GetInputComputation()

```
exageostat::common::Computation GetInputComputation (
            std::string aComputation )  [inline]
```

Convert a string representation of computation mode to its corresponding enum value.

Parameters

in	*aComputation*	String representation of computation mode.

Returns

Computation enum value.

9.28.2.2 GetInputDimension()

```
exageostat::common::Dimension GetInputDimension (
            std::string aDimension )  [inline]
```

Converts string to dimension enum.

Parameters

in	*aDimension*	Dimension as a string.

Returns

Dimension as an enum.

9.29 ErrorHandler.hpp File Reference

Provides error handling functionalities.

Classes

- class APIException
  Custom exception class for handling API errors and warnings.

Macros

- #define API_EXCEPTION(MESSAGE, ERROR_TYPE) APIException(MESSAGE, ERROR_TYPE)
  EXAGEOSTAT API Exceptions Macro to use for Errors and Warnings.

Enumerations

- enum ErrorType : int { RUNTIME_ERROR = 0 , RANGE_ERROR = 1 , INVALID_ARGUMENT_ERROR = 2 , WARNING = 3 }
  Enumeration for error types.

9.29.1 Detailed Description

Provides error handling functionalities.

Version

1.1.0

Defines macros and functions for handling errors and warnings.

Author

Mahmoud ElKarargy

David Helmy

Date

2024-01-20

9.29.2 Macro Definition Documentation

9.29.2.1 API_EXCEPTION

```
#define API_EXCEPTION(
              MESSAGE,
              ERROR_TYPE )  APIException(MESSAGE, ERROR_TYPE)
```

EXAGEOSTAT API Exceptions Macro to use for Errors and Warnings.

9.29.3 Enumeration Type Documentation

9.29.3.1 ErrorType

```
enum ErrorType : int
```

Enumeration for error types.

Enumerator

RUNTIME_ERROR	
RANGE_ERROR	
INVALID_ARGUMENT_ERROR	
WARNING	

9.30 ExaGeoStat.hpp File Reference

High-Level Wrapper class containing the static API for ExaGeoStat operations.

```
#include <nlopt.hpp>
#include <configurations/Configurations.hpp>
#include <data-units/ExaGeoStatData.hpp>
```

Classes

- class exageostat::api::ExaGeoStat< T >
  High-Level Wrapper class containing the static API for ExaGeoStat operations.

Namespaces

- exageostat
- exageostat::api

9.30.1 Detailed Description

High-Level Wrapper class containing the static API for ExaGeoStat operations.

Version

1.1.0

Author

Mahmoud ElKarargy

Date

2024-02-04

9.31 ExaGeoStatData.hpp File Reference

Contains the definition of the ExaGeoStatData class.

```
#include <data-units/DescriptorData.hpp>
#include <data-units/Locations.hpp>
```

Classes

- class ExaGeoStatData< T >

  Manages geo-statistical data with functions for location and descriptor manipulation.

9.31.1 Detailed Description

Contains the definition of the ExaGeoStatData class.

Version

1.1.0

Author

Mahmoud ElKarargy

Sameh Abdulah

Date

2024-02-04

9.32 ExaGeoStatDescriptor.hpp File Reference

Class for creating matrix descriptors used in CHAMELEON and HiCMA libraries.

```
#include <linear-algebra-solvers/concrete/ChameleonHeaders.hpp>
#include <linear-algebra-solvers/concrete/HicmaHeaders.hpp>
#include <common/Definitions.hpp>
```

Classes

- class exageostat::dataunits::descriptor::ExaGeoStatDescriptor< T >
  ExaGeoStatDescriptor is a class for creating matrix descriptors used in CHAMELEON and HiCMA libraries.

Namespaces

- exageostat
- exageostat::dataunits
- exageostat::dataunits::descriptor

9.32.1 Detailed Description

Class for creating matrix descriptors used in CHAMELEON and HiCMA libraries.

Version

1.1.0

Author

Mahmoud ElKarargy

Sameh Abdulah

Date

2023-07-16

9.33 ExaGeoStatHardware.hpp File Reference

Contains the definition of the ExaGeoStatHardware class.

```
#include <common/Definitions.hpp>
```

Classes

- class ExaGeoStatHardware
  Class represents the hardware configuration for the ExaGeoStat solver.

9.33.1 Detailed Description

Contains the definition of the ExaGeoStatHardware class.

Version

1.1.0

Author

Mahmoud ElKarargy

Sameh Abdulah

Date

2024-01-24

9.34 FunctionsAdapter.hpp File Reference

```
#include <Rcpp.h>
#include <configurations/Configurations.hpp>
#include <data-units/ExaGeoStatData.hpp>
```

Namespaces

- exageostat
- exageostat::adapters

Functions

- std::vector< std::vector< double > > exageostat::adapters::R_GetLocations (ExaGeoStatData< double > *apData)

  Retrieves locations from ExaGeoStat data.
- Rcpp::NumericVector exageostat::adapters::R_GetDescZValues (ExaGeoStatData< double > *apData, const std::string &aType)

  Retrieves descriptive Z values from ExaGeoStat data based on type.
- ExaGeoStatData< double > * exageostat::adapters::R_ExaGeoStatLoadData (const std::string &aKernel←Name, const std::vector< double > &aInitialTheta, const std::string &aDistanceMatrix, const int &aProblem←Size, const int &aSeed, const int &aDenseTileSize, const int &aLowTileSize, const std::string &aDimension, const std::string &aLogPath, const std::string &aDataPath, const std::string &aRecoveryFilePath, const std←::string &aObservationsFilePath)

  Function to load ExaGeoStat data.
- std::vector< double > exageostat::adapters::R_ExaGeoStatModelData (const std::string &aComputation, const std::string &aKernelName, const std::string &aDistanceMatrix, const std::vector< double > &aLower←Bound, const std::vector< double > &aUpperBound, const int &aTolerance, const int &aMleIterations, const int &aDenseTileSize, const int &aLowTileSize, const std::string &aDimension, const int &aBand, const int &aMaxRank, SEXP apData, Rcpp::Nullable< Rcpp::NumericVector > aMeasurementsVector=R_NilValue, Rcpp::Nullable< Rcpp::NumericVector > aLocationsX=R_NilValue, Rcpp::Nullable< Rcpp::NumericVector > aLocationsY=R_NilValue, Rcpp::Nullable< Rcpp::NumericVector > aLocationsZ=R_NilValue)

Models ExaGeoStat data using specified arguments.

- std::vector< double > exageostat::adapters::R_ExaGeoStatPredictData (const std::string &aKernelName, const std::string &aDistanceMatrix, const std::vector< double > &aEstimatedTheta, const int &aDenseTileSize, const int &aLowTileSize, const std::string &aDimension, std::vector< std::vector< double >> &aTrainData, std::vector< std::vector< double >> &aTestData)

  Predicts outcomes using ExaGeoStat data and configurations.

- std::vector< double > exageostat::adapters::R_ExaGeoStatMLOE_MMOM (const std::string &aKernelName, const std::string &aDistanceMatrix, const std::vector< double > &aEstimatedTheta, const std::vector< double > &aTrueTheta, const int &aDenseTileSize, const int &aLowTileSize, const std::string &aDimension, std::vector< std::vector< double >> &aTrainData, std::vector< std::vector< double >> &aTestData)

  Calculates the Mean Logarithmic Error (MLOE) and the Mean Measure of Model Output (MMOM) for ExaGeoStat predictions.

- std::vector< double > exageostat::adapters::R_ExaGeoStatFisher (const std::string &aKernelName, const std::string &aDistanceMatrix, const std::vector< double > &aEstimatedTheta, const int &aDenseTileSize, const int &aLowTileSize, const std::string &aDimension, std::vector< std::vector< double >> &aTrainData, std::vector< std::vector< double >> &aTestData)

  Computes the Fisher information matrix for ExaGeoStat models.

- std::vector< double > exageostat::adapters::R_ExaGeoStatIDW (const std::string &aKernelName, const std::string &aDistanceMatrix, const std::vector< double > &aEstimatedTheta, const int &aDenseTileSize, const int &aLowTileSize, const std::string &aDimension, std::vector< std::vector< double >> &aTrainData, std::vector< std::vector< double >> &aTestData, std::vector< double > &aTestMeasurementsValues)

  Applies Inverse Distance Weighting (IDW) for spatial interpolation using ExaGeoStat data.

- double * exageostat::adapters::GetDataFromArguments (Rcpp::Nullable< Rcpp::NumericVector > aMeasurementsVector, Rcpp::Nullable< Rcpp::NumericVector > aLocationsX, Rcpp::Nullable< Rcpp::NumericVector > aLocationsY, Rcpp::Nullable< Rcpp::NumericVector > aLocationsZ, std::unique_ptr< ExaGeoStatData< double >> &aData, configurations::Configurations &aConfigurations, const std::string &aKernelName, const std::string &aDistanceMatrix, const int &aDenseTileSize, const int &aLowTileSize, const std::string &aDimension, const common::Computation &aComputation)

  Extracts and prepares data from given arguments for ExaGeoStat operations.

- void exageostat::adapters::ValidateDataDimensions (const std::vector< std::vector< double >> &aData, const std::string &aDataType)

  Validates the dimensions of input data.

- void exageostat::adapters::PredictionSetupHelper (configurations::Configurations &aConfigurations, const std::string &aKernelName, const std::string &aDistanceMatrix, const int &aDenseTileSize, const int &aLowTileSize, const std::string &aDimension, std::vector< std::vector< double >> &aTrainData, std::vector< std::vector< double >> &aTestData, const std::vector< double > &aEstimatedTheta, const std::vector< double > &aTestMeasurementsValues)

  Sets up the prediction environment.

9.35 gaussian-to-non-codelet.hpp File Reference

A class for starpu codelet gaussian-to-non.

```
#include <common/Definitions.hpp>
```

Classes

- class exageostat::runtime::GaussianCodelet< T >

Namespaces

- exageostat
- exageostat::runtime

9.35.1 Detailed Description

A class for starpu codelet gaussian-to-non.

Version

1.1.0

Author

Mahmoud ElKarargy

Sameh Abdulah

Date

2024-02-25

9.36 HicmaDescriptor.hpp File Reference

Defines the Hicma Descriptor class for creating matrix descriptors using the HICMA library.

```
#include <linear-algebra-solvers/concrete/HicmaHeaders.hpp>
#include <common/Definitions.hpp>
```

Classes

- class exageostat::dataunits::descriptor::HicmaDescriptor< T >

  HicmaDescriptor is a class for creating matrix descriptors by HICMA library.

Namespaces

- exageostat
- exageostat::dataunits
- exageostat::dataunits::descriptor

9.36.1 Detailed Description

Defines the Hicma Descriptor class for creating matrix descriptors using the HICMA library.

Version

1.1.0

Author

Mahmoud ElKarargy

Sameh Abdulah

Date

2023-08-15

9.37 HicmaHeaders.hpp File Reference

This file contains the necessary includes for using the Chameleon library.

9.37.1 Detailed Description

This file contains the necessary includes for using the Chameleon library.

Version

1.1.0

Author

Mahmoud ElKarargy

Date

2023-08-24

9.38 HicmaImplementation.hpp File Reference

This file contains the declaration of HicmaImplementation class.

```
#include <linear-algebra-solvers/LinearAlgebraMethods.hpp>
```

Classes

- class exageostat::linearAlgebra::tileLowRank::HicmaImplementation< T >

    HicmaImplementation is a concrete implementation of LinearAlgebraMethods class for tile low-rank matrices.

Namespaces

- exageostat
- exageostat::linearAlgebra
- exageostat::linearAlgebra::tileLowRank

9.38.1 Detailed Description

This file contains the declaration of HicmaImplementation class.

HicmaImplementation is a concrete implementation of LinearAlgebraMethods class for tile low-rank matrices.

Version

1.1.0

Author

Mahmoud ElKarargy

Sameh Abdulah

Date

2023-03-26

9.39 HicmaStarPuHelpers.hpp File Reference

A class for Hicma implementation of StarPu helpers interface StarPuHelpers.hpp.

```
#include <runtime/starpu/helpers/StarPuHelpers.hpp>
```

Classes

- class exageostat::runtime::HicmaStarPuHelpers

    HicmaStarPuHelpers is a concrete implementation of StarPuHelpers interface for Hicma library.

Namespaces

- exageostat
- exageostat::runtime

9.39.1 Detailed Description

A class for Hicma implementation of StarPu helpers interface StarPuHelpers.hpp.

Version

1.1.0

Author

Mahmoud ElKarargy

Date

2024-02-25

9.40 Kernel.hpp File Reference

```
#include <cmath>
#include <starpu.h>
#include <gsl/gsl_sf_bessel.h>
#include <gsl/gsl_sf_psi.h>
#include <common/PluginRegistry.hpp>
#include <data-units/Locations.hpp>
#include <helpers/DistanceCalculationHelpers.hpp>
#include <helpers/BasselFunction.hpp>
```

Classes

- struct exageostat::kernels::KernelsConfigurations
- class exageostat::kernels::Kernel< T >

Namespaces

- exageostat
- exageostat::kernels

Macros

- #define EARTH_RADIUS 6371.0

  The radius of the Earth in kilometers.

9.40.1 Macro Definition Documentation

9.40.1.1 EARTH_RADIUS

```
#define EARTH_RADIUS 6371.0
```

The radius of the Earth in kilometers.

This macro defines the radius of the Earth in kilometers, which is used by the Great Circle Distance (GCD) function.

9.41 LinearAlgebraFactory.hpp File Reference

Header file for the LinearAlgebraFactory class, which creates linear algebra solvers based on the input computation type.

```
#include <memory>
#include <common/Definitions.hpp>
#include <data-units/DescriptorData.hpp>
#include <linear-algebra-solvers/LinearAlgebraMethods.hpp>
```

Classes

- class exageostat::linearAlgebra::LinearAlgebraFactory< T >

  A class that creates linear algebra solvers based on the input computation type.

Namespaces

- exageostat
- exageostat::linearAlgebra

9.41.1 Detailed Description

Header file for the LinearAlgebraFactory class, which creates linear algebra solvers based on the input computation type.

Version

1.1.0

Author

Mahmoud ElKarargy

Date

2023-03-20

9.42 LinearAlgebraMethods.hpp File Reference

Header file for the LinearAlgebraMethods class, which defines the interface for linear algebra solvers.

```
#include <vector>
#include <gsl/gsl_errno.h>
#include <utilities/Logger.hpp>
#include <results/Results.hpp>
#include <data-units/ExaGeoStatData.hpp>
#include <runtime/RuntimeFunctions.hpp>
```

Classes

- class exageostat::linearAlgebra::LinearAlgebraMethods< T >

    A class that defines the interface for linear algebra solvers.

Namespaces

- exageostat
- exageostat::linearAlgebra

9.42.1 Detailed Description

Header file for the LinearAlgebraMethods class, which defines the interface for linear algebra solvers.

Version

1.1.0

Author

Mahmoud ElKarargy

Sameh Abdulah

Date

2024-02-25

This header file defines the abstract class LinearAlgebraMethods, which provides an interface for linear algebra solvers. The purpose of this interface is to allow different concrete linear algebra solvers to be interchangeable, so that they can be used interchangeably by other parts of the software system that rely on linear algebra.

9.43 LocationGenerator.hpp File Reference

Generates and manages spatial locations for ExaGeoStat.

```
#include <data-units/Locations.hpp>
```

Classes

- class exageostat::generators::LocationGenerator< T >

  Generates spatial locations based on given parameters.

Namespaces

- exageostat
- exageostat::generators

9.43.1 Detailed Description

Generates and manages spatial locations for ExaGeoStat.

Version

1.1.0

Author

Mahmoud ElKarargy

Date

2024-02-04

9.44 Locations.hpp File Reference

Header file for the Locations class, which contains methods to set and get location data.

```
#include <common/Definitions.hpp>
```

Classes

- class exageostat::dataunits::Locations< T >

  A class containing methods to set and get location data.

Namespaces

- exageostat
- exageostat::dataunits

9.44.1 Detailed Description

Header file for the Locations class, which contains methods to set and get location data.

Version

1.1.0

Author

Mahmoud ElKarargy

Sameh Abdulah

Date

2023-02-27

9.45 Logger.hpp File Reference

Provides logging and timing macros for debugging and profiling.

```
#include <iostream>
#include <string>
#include <sys/time.h>
#include <common/Definitions.hpp>
#include <configurations/Configurations.hpp>
#include <helpers/CommunicatorMPI.hpp>
```

Macros

- #define DEFAULT_PRECISION 6

    The value of the default C++ std::std::cout number of precision.
- #define VERBOSE(msg)

    Verbose macro for logging and debugging mode.
- #define LOGGER_1(msg)

    LOGGER_1 macro for logging outputs with double taps and new line at the end.
- #define LOGGER_2(msg, A)

    LOGGER_2 macro for logging outputs with double taps and without new line at the end.
- #define LOGGER_CONTROL(x, A, B, FUNC, ...) FUNC

    LOGGER_CONTROL is The internal macro that simply strips the excess and ends up with the required macro.
- #define LOGGER(...)

    LOGGER macro that's called, Used to logging outputs.
- #define LOGGER_PRECISION_1(msg, precision)

    LOGGER_PRECISION_1 macro for logging outputs without any taps, without new line at the end, and with customized precision.
- #define LOGGER_PRECISION_2(msg)

    LOGGER_PRECISION_2 macro for logging outputs without any taps, without new line at the end, and with default C++ precision.
- #define LOGGER_PRECISION_CONTROL(x, A, B, FUNC, ...) FUNC

    is The internal macro that simply strips the excess and ends up with the required macro
- #define LOGGER_PRECISION(...)

    LOGGER_PRECISION macro that's called, Used for logging outputs with precision.
- #define START_TIMING(t) auto t##_start = std::chrono::high_resolution_clock::now()

    Timing macro to start timing.
- #define STOP_TIMING(t)

    Timing macro to stop timing.

9.45.1 Detailed Description

Provides logging and timing macros for debugging and profiling.

Defines macros for verbose logging, various levels of logging, and timing.

Version

1.1.0

Author

Mahmoud ElKarargy

Sameh Abdulah

Date

2024-02-04

9.45.2 Macro Definition Documentation

9.45.2.1 DEFAULT_PRECISION

```
#define DEFAULT_PRECISION 6
```

The value of the default C++ std::std::cout number of precision.

9.45.2.2 VERBOSE

```
#define VERBOSE(
            msg )
```

Value:

```
if(exageostat::configurations::Configurations::GetVerbosity() ==
   exageostat::common::Verbose::DETAILED_MODE && \
   !exageostat::helpers::CommunicatorMPI::GetInstance()->GetRank()) { \
    std::ostringstream oss; \
    oss « "\t\t\t " « msg « std::endl; \
    std::cout « oss.str(); \
}
```

Verbose macro for logging and debugging mode.

9.45.2.3 LOGGER_1

```
#define LOGGER_1(
                msg )
```

Value:

```
    if(!(exageostat::configurations::Configurations::GetVerbosity() ==
       exageostat::common::Verbose::QUIET_MODE) &&
       !exageostat::helpers::CommunicatorMPI::GetInstance()->GetRank()){ \
        std::ostringstream oss; \
        oss « "\t\t " « std::fixed « std::setprecision(DEFAULT_PRECISION) « msg « std::endl; \
        std::cout « oss.str();
                                                                \
    }
```

LOGGER_1 macro for logging outputs with double taps and new line at the end.

9.45.2.4 LOGGER_2

```
#define LOGGER_2(
                msg,
                A )
```

Value:

```
    if(!(exageostat::configurations::Configurations::GetVerbosity() ==
       exageostat::common::Verbose::QUIET_MODE) &&
       !exageostat::helpers::CommunicatorMPI::GetInstance()->GetRank()){ \
        std::ostringstream oss;
                                                                        \
        oss « "\t\t " « std::fixed « std::setprecision(DEFAULT_PRECISION) « msg;                  \
        std::cout « oss.str(); \
}
```

LOGGER_2 macro for logging outputs with double taps and without new line at the end.

9.45.2.5 LOGGER_CONTROL

```
#define LOGGER_CONTROL(
                x,
                A,
                B,
                FUNC,
                ... ) FUNC
```

LOGGER_CONTROL is The internal macro that simply strips the excess and ends up with the required macro.

9.45.2.6 LOGGER

```
#define LOGGER(
                ... )
```

Value:

```
                        LOGGER_CONTROL(,##__VA_ARGS__, \
                        LOGGER_2(__VA_ARGS__),         \
                        LOGGER_1(__VA_ARGS__),         \
                                        )
```

LOGGER macro that's called, Used to logging outputs.

9.45.2.7 LOGGER_PRECISION_1

```
#define LOGGER_PRECISION_1(
                msg,
                precision )
```

Value:

```
if(!(exageostat::configurations::Configurations::GetVerbosity() ==
   exageostat::common::Verbose::QUIET_MODE) &&
   !exageostat::helpers::CommunicatorMPI::GetInstance()->GetRank()){ \
    std::ostringstream oss;
                                                                          \
    oss « std::fixed « std::setprecision(precision) « msg;                                  \
    std::cout « oss.str(); \
}
```

LOGGER_PRECISION_1 macro for logging outputs without any taps, without new line at the end, and with customized precision.

9.45.2.8 LOGGER_PRECISION_2

```
#define LOGGER_PRECISION_2(
                msg )
```

Value:

```
if(!(exageostat::configurations::Configurations::GetVerbosity() ==
   exageostat::common::Verbose::QUIET_MODE) &&
   !exageostat::helpers::CommunicatorMPI::GetInstance()->GetRank()) {\
    std::ostringstream oss;
                                                                          \
    oss « std::fixed « std::setprecision(DEFAULT_PRECISION) « msg;                               \
    std::cout « oss.str(); \
}
```

LOGGER_PRECISION_2 macro for logging outputs without any taps, without new line at the end, and with default C++ precision.

9.45.2.9 LOGGER_PRECISION_CONTROL

```
#define LOGGER_PRECISION_CONTROL(
                x,
                A,
                B,
                FUNC,
                ... ) FUNC
```

is The internal macro that simply strips the excess and ends up with the required macro

9.45.2.10 LOGGER_PRECISION

```
#define LOGGER_PRECISION(
              ... )
```

Value:

```
LOGGER_PRECISION_CONTROL(,##__VA_ARGS__, \
LOGGER_PRECISION_1(__VA_ARGS__),          \
LOGGER_PRECISION_2(__VA_ARGS__),          \
              )
```

LOGGER_PRECISION macro that's called, Used for logging outputs with precision.

9.45.2.11 START_TIMING

```
#define START_TIMING(
              t ) auto t##_start = std::chrono::high_resolution_clock::now()
```

Timing macro to start timing.

9.45.2.12 STOP_TIMING

```
#define STOP_TIMING(
              t )
```

Value:

```
auto t##_end = std::chrono::high_resolution_clock::now(); \
t = std::chrono::duration_cast<std::chrono::duration<double»(t##_end -
t##_start).count()
```

Timing macro to stop timing.

9.46 ModelingDataHolders.hpp File Reference

This file contains the definition of the mModelingData struct, which contains all the data needed for modeling.

Classes

- struct exageostat::dataunits::mModelingData< T >

    Struct containing all the data needed for modeling.

Namespaces

- exageostat
- exageostat::dataunits

9.46.1 Detailed Description

This file contains the definition of the mModelingData struct, which contains all the data needed for modeling.

Version

1.1.0

Author

Mahmoud ElKarargy

Date

2023-08-24

9.47 non-gaussian-loglike-codelet.hpp File Reference

A class for starpu codelet non-gaussian-loglike.

```
#include <runtime/starpu/helpers/StarPuHelpers.hpp>
```

Classes

- class exageostat::runtime::NonGaussianLoglike< T >
  A class for starpu codelet non gaussian loglike.

Namespaces

- exageostat
- exageostat::runtime

9.47.1 Detailed Description

A class for starpu codelet non-gaussian-loglike.

Version

1.1.0

Author

Mahmoud ElKarargy

Sameh Abdulah

Date

2024-02-26

9.48 non-gaussian-transform-codelet.hpp File Reference

A class for starpu codelet non-gaussian-transform.

```
#include <runtime/starpu/helpers/StarPuHelpers.hpp>
```

Classes

- class exageostat::runtime::NonGaussianTransform< T >

  A class for starpu codelet non gaussian transform.

Namespaces

- exageostat
- exageostat::runtime

9.48.1 Detailed Description

A class for starpu codelet non-gaussian-transform.

Version

1.1.0

Author

Mahmoud ElKarargy

Sameh Abdulah

Date

2024-02-26

9.49 PluginRegistry.hpp File Reference

Defines a template class for registering and creating plugins.

```
#include <functional>
#include <configurations/Configurations.hpp>
```

Classes

- class exageostat::plugins::PluginRegistry< T >

  Template class for registering and creating plugins.

Namespaces

- exageostat
- exageostat::plugins

9.49.1 Detailed Description

Defines a template class for registering and creating plugins.

Version

1.1.0

Author

Mahmoud ElKarargy

Date

2023-04-30

9.50 Prediction.hpp File Reference

Contains the definition of the Prediction class.

```
#include <linear-algebra-solvers/LinearAlgebraMethods.hpp>
```

Classes

- class exageostat::prediction::Prediction< T >

  Class to handle different Prediction Module calls.

Namespaces

- exageostat
- exageostat::prediction

9.50.1 Detailed Description

Contains the definition of the Prediction class.

Version

1.1.0

Author

Mahmoud ElKarargy

Sameh Abdulah

Date

2023-06-08

9.51 PredictionAuxiliaryFunctions.hpp File Reference

Contains the definition of the PredictionAuxiliaryFunctions.hpp class.

```
#include <data-units/Locations.hpp>
```

Classes

- class exageostat::prediction::PredictionAuxiliaryFunctions< T >
  Class to define and implement different Prediction Module Auxiliary Functions.

Namespaces

- exageostat
- exageostat::prediction

9.51.1 Detailed Description

Contains the definition of the PredictionAuxiliaryFunctions.hpp class.

Version

1.1.0

Author

Mahmoud ElKarargy

Sameh Abdulah

Date

2023-06-08

9.52 PredictionHelpers.hpp File Reference

Contains the definition of the PredictionHelpers.hpp class.

```
#include <data-units/Locations.hpp>
#include <data-units/ExaGeoStatData.hpp>
#include <configurations/Configurations.hpp>
```

Classes

- class exageostat::prediction::PredictionHelpers< T >
  Class to define and implement different Prediction Module helpers functions.

Namespaces

- exageostat
- exageostat::prediction

9.52.1 Detailed Description

Contains the definition of the PredictionHelpers.hpp class.

Version

1.1.0

Author

Mahmoud ElKarargy

Sameh Abdulah

Date

2023-06-08

9.53 README.md File Reference

9.54 README.md File Reference

9.55 Results.hpp File Reference

Defines the Results class for storing and accessing result data.

```
#include <iostream>
#include <vector>
```

Classes

- class exageostat::results::Results

Namespaces

- exageostat
- exageostat::results

9.55.1 Detailed Description

Defines the Results class for storing and accessing result data.

Version

1.1.0

Author

Mahmoud ElKarargy

Date

2023-09-14

9.56 RuntimeFunctions.hpp File Reference

A class for runtime static functions.

```
#include <kernels/Kernel.hpp>
#include <data-units/ExaGeoStatData.hpp>
```

Classes

- class exageostat::runtime::RuntimeFunctions< T >
    A class that defines runtime static functions.

Namespaces

- exageostat
- exageostat::runtime

9.56.1 Detailed Description

A class for runtime static functions.

Version

1.1.0

Author

Mahmoud ElKarargy

Date

2024-03-10

9.57 StarPuCodeletsHeaders.hpp File Reference

```
#include <runtime/starpu/concrete/dcmg-codelet.hpp>
#include <runtime/starpu/concrete/ddotp-codelet.hpp>
#include <runtime/starpu/concrete/dmdet-codelet.hpp>
#include <runtime/starpu/concrete/dmloe-mmom-codelet.hpp>
#include <runtime/starpu/concrete/dmse-bivariate-codelet.hpp>
#include <runtime/starpu/concrete/dmse-codelet.hpp>
#include <runtime/starpu/concrete/dtrace-codelet.hpp>
#include <runtime/starpu/concrete/dzcpy-codelet.hpp>
#include <runtime/starpu/concrete/gaussian-to-non-codelet.hpp>
#include <runtime/starpu/concrete/non-gaussian-loglike-codelet.hpp>
#include <runtime/starpu/concrete/non-gaussian-transform-codelet.hpp>
#include <runtime/starpu/concrete/stride-vec-codelet.hpp>
#include <runtime/starpu/concrete/tri-stride-vec-codelet.hpp>
```

9.58 StarPuHelpers.hpp File Reference

An interface for StarPu helpers.

```
#include <complex>
#include <common/Definitions.hpp>
#include <hardware/ExaGeoStatHardware.hpp>
```

Classes

- class exageostat::runtime::StarPuHelpers

    A class that defines the interface for StarPu helpers.

Namespaces

- exageostat
- exageostat::runtime

9.58.1 Detailed Description

An interface for StarPu helpers.

Version

1.1.0

Author

Mahmoud ElKarargy

Date

2024-02-25

9.59 StarPuHelpersFactory.hpp File Reference

Factory for StarPu helpers.

```
#include <runtime/starpu/helpers/StarPuHelpers.hpp>
```

Classes

- class exageostat::runtime::StarPuHelpersFactory

  A class that creates StarPu helpers based on the input computation type.

Namespaces

- exageostat
- exageostat::runtime

9.59.1 Detailed Description

Factory for StarPu helpers.

Version

1.1.0

Author

Mahmoud ElKarargy

Date

2024-02-25

9.60 stride-vec-codelet.hpp File Reference

A class for starpu codelet stride-vec.

```
#include <common/Definitions.hpp>
```

Classes

- class exageostat::runtime::STRIDEVECCodelet< T >

Namespaces

- exageostat
- exageostat::runtime

9.60.1 Detailed Description

A class for starpu codelet stride-vec.

Version

1.1.0

Author

Mahmoud ElKarargy

Sameh Abdulah

Date

2024-02-25

9.61 SyntheticGenerator.hpp File Reference

A class for generating synthetic data.

```
#include <data-generators/DataGenerator.hpp>
```

Classes

- class exageostat::generators::synthetic::SyntheticGenerator< T >

  A class for generating synthetic data.

Namespaces

- exageostat
- exageostat::generators
- exageostat::generators::synthetic

9.61.1 Detailed Description

A class for generating synthetic data.

Version

1.1.0

Author

Mahmoud ElKarargy

Sameh Abdulah

Date

2023-02-14

9.62 tri-stride-vec-codelet.hpp File Reference

A class for starpu codelet tri-stride-vec.

```
#include <common/Definitions.hpp>
```

Classes

- class exageostat::runtime::TriStrideVecCodelet< T >
  A class for starpu codelet tri_stride_vec.

Namespaces

- exageostat
- exageostat::runtime

9.62.1 Detailed Description

A class for starpu codelet tri-stride-vec.

Version

1.1.0

Author

Mahmoud ElKarargy

Sameh Abdulah

Date

2024-02-25

9.63 TrivariateMaternParsimonious.hpp File Reference

Defines the TrivariateMaternParsimonious class, a Trivariate Matern Parsimonious kernel.

```
#include <kernels/Kernel.hpp>
```

Classes

- class exageostat::kernels::TrivariateMaternParsimonious< T >

  A class represents a Trivariate Matern Parsimonious kernel.

Namespaces

- exageostat
- exageostat::kernels

9.63.1 Detailed Description

Defines the TrivariateMaternParsimonious class, a Trivariate Matern Parsimonious kernel.

Version

1.1.0

Author

Mahmoud ElKarargy

Sameh Abdulah

Suhas Shankar

Mary Lai Salvana

Date

2023-04-14

9.64 UnivariateExpNonGaussian.hpp File Reference

Defines the UnivariateExpNonGaussian class, a Univariate Exp Non Gaussian kernel.

```
#include <kernels/Kernel.hpp>
```

Classes

- class exageostat::kernels::UnivariateExpNonGaussian< T >

  A class represents a Univariate Exp Non Gaussian kernel.

Namespaces

- exageostat
- exageostat::kernels

9.64.1 Detailed Description

Defines the UnivariateExpNonGaussian class, a Univariate Exp Non Gaussian kernel.

Version

1.1.0

Author

Mahmoud ElKarargy

Sameh Abdulah

Suhas Shankar

Mary Lai Salvana

Date

2023-04-14

9.65 UnivariateMaternDbeta.hpp File Reference

Defines the UnivariateMaternDbeta class, a Univariate Matern Dbeta kernel.

```
#include <kernels/Kernel.hpp>
```

Classes

- class exageostat::kernels::UnivariateMaternDbeta< T >

    A class represents a Univariate Matern Dbeta kernel.

Namespaces

- exageostat
- exageostat::kernels

9.65.1 Detailed Description

Defines the UnivariateMaternDbeta class, a Univariate Matern Dbeta kernel.

Version

1.1.0

Author

Mahmoud ElKarargy

Sameh Abdulah

Suhas Shankar

Mary Lai Salvana

Date

2023-04-14

9.66 UnivariateMaternDdbetaBeta.hpp File Reference

Defines the UnivariateMaternDdbetaBeta class, a Univariate Matern Ddbeta Beta kernel.

```
#include <kernels/Kernel.hpp>
```

Classes

- class exageostat::kernels::UnivariateMaternDdbetaBeta< T >

  A class represents a Univariate Matern Ddbeta Beta kernel.

Namespaces

- exageostat
- exageostat::kernels

9.66.1 Detailed Description

Defines the UnivariateMaternDdbetaBeta class, a Univariate Matern Ddbeta Beta kernel.

Version

1.1.0

Author

Mahmoud ElKarargy

Sameh Abdulah

Suhas Shankar

Mary Lai Salvana

Date

2023-04-14

9.67 UnivariateMaternDdbetaNu.hpp File Reference

Defines the UnivariateMaternDdbetaNu class, a Univariate Matern Ddbeta Nu kernel.

```
#include <kernels/Kernel.hpp>
```

Classes

- class exageostat::kernels::UnivariateMaternDdbetaNu< T >

  A class represents a Univariate Matern Ddbeta Nu kernel.

Namespaces

- exageostat
- exageostat::kernels

9.67.1 Detailed Description

Defines the UnivariateMaternDdbetaNu class, a Univariate Matern Ddbeta Nu kernel.

Version

1.1.0

Author

Mahmoud ElKarargy

Sameh Abdulah

Suhas Shankar

Mary Lai Salvana

Date

2023-04-14

9.68 UnivariateMaternDdnuNu.hpp File Reference

Defines the UnivariateMaternDdnuNu class, a Univariate Matern Ddnu Nu kernel.

```
#include <kernels/Kernel.hpp>
```

Classes

- class exageostat::kernels::UnivariateMaternDdnuNu< T >

  A class represents a Univariate Matern Ddnu Nu kernel.

Namespaces

- exageostat
- exageostat::kernels

9.68.1 Detailed Description

Defines the UnivariateMaternDdnuNu class, a Univariate Matern Ddnu Nu kernel.

Version

1.1.0

Author

Mahmoud ElKarargy

Sameh Abdulah

Suhas Shankar

Mary Lai Salvana

Date

2023-04-14

9.69 UnivariateMaternDdsigmaSquare.hpp File Reference

Defines the UnivariateMaternDdsigmaSquare class, a univariate stationary Matern kernel.

`#include <kernels/Kernel.hpp>`

Classes

- class exageostat::kernels::UnivariateMaternDdsigmaSquare< T >

  A class represents a Univariate Matern Ddsigma Square kernel.

Namespaces

- exageostat
- exageostat::kernels

9.69.1 Detailed Description

Defines the UnivariateMaternDdsigmaSquare class, a univariate stationary Matern kernel.

Version

1.1.0

Author

Mahmoud ElKarargy

Sameh Abdulah

Suhas Shankar

Mary Lai Salvana

Date

2023-04-12

This file provides the declaration of the UnivariateMaternDdsigmaSquare class, which is a subclass of the Kernel class and represents a univariate stationary Matern kernel. It provides a method for generating a covariance matrix using a set of input locations and kernel parameters.

9.70 UnivariateMaternDdsigmaSquareBeta.hpp File Reference

Defines the UnivariateMaternDdsigmaSquareBeta class, a Univariate Matern Ddsigma Square Beta kernel.

```
#include <kernels/Kernel.hpp>
```

Classes

- class exageostat::kernels::UnivariateMaternDdsigmaSquareBeta< T >

  A class represents a Univariate Matern Ddsigma Square Beta kernel.

Namespaces

- exageostat
- exageostat::kernels

9.70.1 Detailed Description

Defines the UnivariateMaternDdsigmaSquareBeta class, a Univariate Matern Ddsigma Square Beta kernel.

Version

1.1.0

Author

Mahmoud ElKarargy

Sameh Abdulah

Suhas Shankar

Mary Lai Salvana

Date

2023-04-14

9.71 UnivariateMaternDdsigmaSquareNu.hpp File Reference

Defines the UnivariateMaternDdsigmaSquareNu class, a Univariate Matern Ddsigma Square Nu kernel.

```
#include <kernels/Kernel.hpp>
```

Classes

- class exageostat::kernels::UnivariateMaternDdsigmaSquareNu< T >
  A class represents a Univariate Matern Ddsigma Square Nu kernel.

Namespaces

- exageostat
- exageostat::kernels

9.71.1 Detailed Description

Defines the UnivariateMaternDdsigmaSquareNu class, a Univariate Matern Ddsigma Square Nu kernel.

Version

1.1.0

Author

Mahmoud ElKarargy

Sameh Abdulah

Suhas Shankar

Mary Lai Salvana

Date

2023-04-14

9.72 UnivariateMaternDnu.hpp File Reference

Defines the UnivariateMaternDnu class, a Univariate Matern Dnu kernel.

```
#include <kernels/Kernel.hpp>
```

Classes

- class exageostat::kernels::UnivariateMaternDnu< T >

  A class represents a Univariate Matern Dnu kernel.

Namespaces

- exageostat
- exageostat::kernels

9.72.1 Detailed Description

Defines the UnivariateMaternDnu class, a Univariate Matern Dnu kernel.

Version

1.1.0

Author

Mahmoud ElKarargy

Sameh Abdulah

Suhas Shankar

Mary Lai Salvana

Date

2023-04-14

9.73 UnivariateMaternDsigmaSquare.hpp File Reference

Defines the UnivariateMaternDsigmaSquare class, a Univariate Matern Dsigma Square kernel.

```
#include <kernels/Kernel.hpp>
```

Classes

- class exageostat::kernels::UnivariateMaternDsigmaSquare< T >

  A class represents a Univariate Matern Dsigma Square kernel.

Namespaces

- exageostat
- exageostat::kernels

9.73.1 Detailed Description

Defines the UnivariateMaternDsigmaSquare class, a Univariate Matern Dsigma Square kernel.

Version

1.1.0

Author

Mahmoud ElKarargy

Sameh Abdulah

Suhas Shankar

Mary Lai Salvana

Date

2023-04-14

9.74 UnivariateMaternNonGaussian.hpp File Reference

Defines the UnivariateMaternNonGaussian class, a Univariate Matern Non Gaussian kernel.

```
#include <kernels/Kernel.hpp>
```

Classes

- class exageostat::kernels::UnivariateMaternNonGaussian< T >

    A class represents a Univariate Matern Non Gaussian kernel.

Namespaces

- exageostat
- exageostat::kernels

9.74.1 Detailed Description

Defines the UnivariateMaternNonGaussian class, a Univariate Matern Non Gaussian kernel.

Version

1.1.0

Author

Mahmoud ElKarargy
Sameh Abdulah
Suhas Shankar
Mary Lai Salvana

Date

2023-04-14

9.75 UnivariateMaternNuggetsStationary.hpp File Reference

Defines the UnivariateMaternNuggetsStationary class, a Univariate Matern Nuggets Stationary kernel.

```
#include <kernels/Kernel.hpp>
```

Classes

- class exageostat::kernels::UnivariateMaternNuggetsStationary< T >
  A class represents a Univariate Matern Nuggets Stationary kernel.

Namespaces

- exageostat
- exageostat::kernels

9.75.1 Detailed Description

Defines the UnivariateMaternNuggetsStationary class, a Univariate Matern Nuggets Stationary kernel.

Version

1.1.0

Author

Mahmoud ElKarargy
Sameh Abdulah
Suhas Shankar
Mary Lai Salvana

Date

2023-04-14

9.76 UnivariateMaternStationary.hpp File Reference

Defines the UnivariateMaternStationary class, a univariate stationary Matern kernel.

```
#include <kernels/Kernel.hpp>
```

Classes

- class exageostat::kernels::UnivariateMaternStationary< T >

  A class represents a Univariate Matern Stationary kernel.

Namespaces

- exageostat
- exageostat::kernels

9.76.1 Detailed Description

Defines the UnivariateMaternStationary class, a univariate stationary Matern kernel.

Version

1.1.0

Author

Mahmoud ElKarargy

Sameh Abdulah

Suhas Shankar

Mary Lai Salvana

Date

2023-04-12

This file provides the declaration of the UnivariateMaternStationary class, which is a subclass of the Kernel class and represents a univariate stationary Matern kernel. It provides a method for generating a covariance matrix using a set of input locations and kernel parameters.

9.77 UnivariatePowExpStationary.hpp File Reference

Defines the UnivariatePowExpStationary class, a univariate stationary PowExp kernel.

```
#include <kernels/Kernel.hpp>
```

Classes

- class exageostat::kernels::UnivariatePowExpStationary< T >

  A class represents a Univariate PowExp Stationary kernel.

Namespaces

- exageostat
- exageostat::kernels

9.77.1 Detailed Description

Defines the UnivariatePowExpStationary class, a univariate stationary PowExp kernel.

Version

1.1.0

Author

Mahmoud ElKarargy

Sameh Abdulah

Suhas Shankar

Mary Lai Salvana

Date

2024-11-22

This file provides the declaration of the UnivariatePowExpStationary class, which is a subclass of the Kernel class and represents a univariate stationary PowExp kernel. It provides a method for generating a covariance matrix using a set of input locations and kernel parameters.

9.78 UnivariateSpacetimeMaternStationary.hpp File Reference

Defines the UnivariateSpacetimeMaternStationary class, a Univariate Spacetime Matern Stationary kernel.

```
#include <kernels/Kernel.hpp>
```

Classes

- class exageostat::kernels::UnivariateSpacetimeMaternStationary< T >

  A class represents a Univariate Spacetime Matern Stationary kernel.

Namespaces

- exageostat
- exageostat::kernels

9.78.1 Detailed Description

Defines the UnivariateSpacetimeMaternStationary class, a Univariate Spacetime Matern Stationary kernel.

Version

1.1.0

Author

Mahmoud ElKarargy

Sameh Abdulah

Suhas Shankar

Mary Lai Salvana

Date

2023-04-14