Gram-Schmidt process on GPU

Generated by Doxygen 1.9.2

1 Class Index **1**

1.1 Class List 1

2 Class Documentation **3**

2.1 GPUGramSchmidt Class Reference 3

2.1.1 Detailed Description 4

2.1.2 Constructor & Destructor Documentation 4

2.1.2.1 GPUGramSchmidt() 4

2.1.2.2 ∼GPUGramSchmidt() 4

2.1.3 Member Function Documentation 5

2.1.3.1 run() 5

2.1.4 Member Data Documentation 5

2.1.4.1 shader_folder 5

Index **7**

Chapter 1

Class Index

1.1 Class List

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

GPUGramSchmidt
Tools to execute Gram-Schmidt process on GPU 3

Chapter 2

Class Documentation

2.1 GPUGramSchmidt Class Reference

Tools to execute Gram-Schmidt process on GPU.

```
#include <vulkan-gram-schmidt.hpp>
```

Public Types

- using **Matrix** = std::vector< std::vector< double > >

Public Member Functions

Constructors & destructors

- GPUGramSchmidt (bool const enable_debug=false)
  Creates a new solver.
- ~GPUGramSchmidt (void)
  Destroys the solver.

Computations

- void run (GPUGramSchmidt::Matrix &matrix, bool const vectors_as_columns=false)
  Run Gram-Schmidt process on GPU.

Static Public Attributes

Static parameters

- static std::string shader_folder = "."

2.1.1 Detailed Description

Tools to execute Gram-Schmidt process on GPU.

This class provides interface for calculation of orthonormal basis on GPU given the initial set of n linearly independent vectors from $\mathbb{R}^n$.

The following requirements are needed to be explicitly satisfied by the end user:

- GPU is requitred to be able to perform compute operations.
- GPU is required to have a host coherent part of memory.
- Vulkan 1.2 (or newer) is required to be supported by the GPU driver.
- Matrices passed to the GPUGramSchmidt::run function are required to be non-singular; otherwise, no guarantees are given about the behaviour of the program.

Instances of this class are generally expected to be thread-secure, however, this was not heavily tested.

2.1.2 Constructor & Destructor Documentation

2.1.2.1 GPUGramSchmidt()

```
GPUGramSchmidt::GPUGramSchmidt (
            bool const enable_debug = false )
```

Creates a new solver.

Sets up a Vulkan communication environment with the GPU.

Parameters

enable_debug	Send Vulkan debug information to the output.

Warning

`enable_debug = true` will require the presence of the `VK_LAYER_KHRONOS_validation` Vulkan layer and the `VK_EXT_debug_utils` Vulkan extension.

2.1.2.2 ∼GPUGramSchmidt()

```
GPUGramSchmidt::∼GPUGramSchmidt (
            void )
```

Destroys the solver.

Destructs the communication environment with the GPU.

2.1.3 Member Function Documentation

2.1.3.1 run()

```
void GPUGramSchmidt::run (
            GPUGramSchmidt::Matrix & matrix,
            bool const vectors_as_columns = false )
```

Run Gram-Schmidt process on GPU.

Perform orthonormalisation of vectors with the help of GPU.

Parameters

matrix	Square matrix with the coordinates of the original vectors.
vectors_as_columns	Indicates whether vectors are packed into `matrix` as columns or as rows.

Warning

Keep in mind, that the non-singularity of `matrix` must be guaranteed by you.

Returns

Nothing; the answer is written directly into `matrix`. If `vectors_as_columns == true`, the answer will also be written in columns.

2.1.4 Member Data Documentation

2.1.4.1 shader_folder

```
std::string GPUGramSchmidt::shader_folder = "." [static]
```

Path to a folder containing "vulkan-gram-schmidt.spv"

Index

∼GPUGramSchmidt
    GPUGramSchmidt, 4

GPUGramSchmidt, 3
    ∼GPUGramSchmidt, 4
    GPUGramSchmidt, 4
    run, 5
    shader_folder, 5

run
    GPUGramSchmidt, 5

shader_folder
    GPUGramSchmidt, 5