Overview

We present new numerical library designed to profit from C++ templates. It supports computations on GPU via Nvidia CUDA. In near future, we plan to release version 0.9 which targets to solution of non-linear partial differential equations. To solve the non-linear problems efficiently, fast assembling of the linear systems in each time step is necessary. Therefore our library provides uniform interface for several formats for sparse matrices which is accessible even from the ODE kernels. It significantly simplifies development of the non-linear solvers.

Aims of TNL project

We want our library to be F.E.A.R. - Flexible, Effective, Reliable and Easy to use.

1. Flexible - researchers in numerical analysis and computer simulations often try many different approaches and methods before they find the successful one. A library, which meets these requirements, must be flexible and it must allow to do large changes with small effort.

2. Effective - computer simulations can be very time consuming. To speed things up, TNL supports computations on GPUs. C++ templates help to generate highly optimized code.

3. Reliable - numerical simulations must be, of course, correct. To eliminate as many bugs in our code as possible, we write number of tests and compare our implementation with other libraries.

4. Easy to use - even though TNL combines technologies like C++ templates and GPU computations, it should be available even to people without deep knowledge of C++ and GPU.

Vectors

Vectors are represented by a template class:

```cpp
tnlVector< Real = double, Device = tnlHost, Index = int >
```


The following formats are supported on both CPU and GPU:

- Dense
- Triangular
- Multidiagonal
- Ellpack
- Skewed Ellpack (published as Row-grouped CSR format)
- Channeled Ellpack (published as Improved Row-grouped CSR format)
- CSR

To initialize the sparse matrix formats like (Sliced/Chunked) Ellpack or CSR, one must specify the number of the nonzero elements in each row. We refer to the following formats for the supported CPU and GPU:

```cpp
tfnlSMatrix<Real> = dense, Device = tnnlHost, Index = int>
```

The following formats are supported on both CPU and GPU:

- Dense
- Triangular
- Multidiagonal
- Ellpack
- Skewed Ellpack (published as Row-grouped CSR format)
- Channeled Ellpack (published as Improved Row-grouped CSR format)
- CSR

To initialize the sparse matrix formats like (Sliced/Chunked) Ellpack or CSR, one must specify the number of the nonzero elements in each row. We refer to the following formats for the supported CPU and GPU:

```cpp
tfnlSMatrix<Real> = dense, Device = tnnlHost, Index = int>
```

To set the nonzero elements of the matrix, we first transfer the matrix object instance to the GPU (we transfer only metadata not the matrix elements - they are already on the GPU):

```cpp
Matrix<double> = device = Matrix<double> = device = Matrix<double> = device = Matrix<double>
```

---

References


---

TNL will be available soon at https://code.google.com/p/tnl