UTOPIA

Installation

```
git clone --recurse-submodules  \
https://bitbucket.org/zulianp/utopia.git

cd utopia/utopia && mkdir build && cd build
cmake .. -DCMAKE_INSTALL_PREFIX=<your install folder>
make && make install
```

Click HERE for more details!

Basics

Backends

Each backend is defined by their tensor types. We denote vector types with V, Matrix types with M, general tensor types with T.

- **Blas–Lapack**, node-level dense algebra

```
using M = BlasMatrixd;
using V = BlasVectord;
```

- **PETSc**, parallel sparse and dense algebra

```
using M = PetscMatrix;
using V = PetscVector;
```

- **Trilinos**, parallel sparse algebra

```
using M = TpetraMatrixd;
using V = TpetraVectord;
```

Traits

Traits are used to access data-types associated with a tensors T.

```
// The MPI Communicator
using Communicator = Traits<T>::Communicator;

// Double or single precision real number
using Scalar = Traits<T>::Scalar;

// 32 or 64 bits integer number used for global indexing
using SizeType = Traits<T>::SizeType;

// 32 or 64 bits integer number used for local indexing
using LocalSizeType = Traits<T>::LocalSizeType;

// Array of indices
using IndexArray = Traits<T>::IndexArray;

// Array of scalars
using ScalarArray = Traits<T>::ScalarArray;

// Vector layout type specific to the backend
using Layout = Traits<T>::Layout;

// Matrix layout type specific to the backend
using MatrixLayout = Traits<T>::MatrixLayout;
```

Communicator

```
// Get the serial communicator
auto self = Communicator::self();

// Get the world communicator
auto world = Communicator::world();

// Get the default communicator (typically world)
auto comm = Communicator::get_default();
```

Layouts

```
// Create a vector layout with n_local entries
// for each process and n entries in total
auto vl = layout(comm, n_local, n);

// Create a matrix layout from the vector layout
auto ml = square_matrix_layout(vl);

// Create a matrix layout with local and total
// number of rows and columns
auto ml = layout(comm, rows_local, cols_local, rows, cols);
```

Tensor

Vector

```
// Construct a vector with a layout vl
V vec(vl);

// Construct a vector with and set it a uniform value
V vec(vl, val);

// Construct an existing vector with zeros,
vec.zeros(vl);
// or with a uniform value
vec.values(vl, val);
```

Matrix

BLAS

Dense and sparse basic linear algebra subroutines

Level 1

Level 2

Level 3

Others

Linear solver

Nonlinear solver

Input-ouput

```
// Any tensor t can be inspected in the terminal
disp(t);
```

Conversions and interoperability

Host-side manipulations

Device-side manipulations

Acknowledgements

[ZKN+16]

References

[ZKN+16] Patrick Zulian, Alena Kopaničáková, Maria Chiara Giuseppina Nestola, Andreas Fink, Nur Fadel, Alessandro Rigazzi, Victor Magri, Teseo Schneider, Eric Botter, Jan Mankau, and Rolf Krause. Utopia: A C++ embedded domain specific language for scientific computing. Git repository. https://bitbucket.org/zulianp/utopia, 2016.