“HODLR”

3.14

Generated by Doxygen 1.8.6

Sat Feb 1 2014 12:09:43

Contents

Chapter 1

HODLR Examples

Kepler

This is a test on real data from the NASA Kepler mission. The test is implemented in `./kepler_test.cpp` and the data are given in `./kepler-10-sc.csv`. The data are "short cadence" observations of the star Kepler 10 and the columns are time (measured in KBJD), relative flux (in arbitrary units), and the uncertainties on the fluxes. To run this example, after `make`-ing while still in the `build` directory, run "‘ bin/kepler $<$ ../examples/kepler-10-sc.csv "‘

Chapter 2

LICENSE

Chapter 3

README

#HODLR SOLVER: A fast direct solver for dense linear systems

This is an extension of the fast direct solver discussed in the article: "An O(N log (N)) Fast Direct Solver for Partial Hierarchically Semi-Separable Matrices". The solver has also been extended to matrices not necessarily arising out of kernels and also to higher dimensions. Further, the solver has been optimized and the running time of the solver is now massively (a few orders of magnitude) faster than the running times reported in the article. Low-rank approximation of the appropriate blocks are obtained using partial pivoted LU algorithm. The domain is sub-divided based on a KDTree. The solver is fairly general and works with minimal restrictions.

To give a rough idea of the running time, for a system size of 1 million the solver takes 23 seconds for a 1D problem and 86 seconds for a 2D problem (this is time taken from the time you press the return key on the keyboard to run your code and to get the final result). The matrix is of the form

```
    A   =   s^2 I + B
```

where B(i,j) depends on R(i,j), the distance between the points x(i) and x(j). The computed answer is accurate upto more than 10 digits. It is to be noted that even for higher dimensions (2D and 3D), the solver is still way faster than the conventional direct solvers, though the scaling may no longer be almost linear. The scaling depends on the smoothness of the kernel near the diagonal of the matrix.

Author

Sivaram Ambikasaran siva.1985@gmail.com

Citation

If you use the implementation or any part of the implementation in your work, kindly cite as follows:

∗∗∗Article∗∗∗

{ambikasaran2013fastdirect, author={{A}mbikasaran, {S}ivaram and {D}arve, {E}ric}, title={An ${O}(N N)$ Fast Direct Solver for Partial Hierarchically Semi-Separable Matrices}, journal={Journal of Scientific Computing}, year={2013}, volume={57}, number={3}, pages={477–501}, month={December}, publisher={Springer} }

∗∗∗Code∗∗∗

{ambikasaran2013HODLR, author = {{A}mbikasaran, {S}ivaram}, title = {A fast direct solver for dense linear systems}, howpublished = {https://github.com/sivaramambikasaran/HODLR_Solver}, year = {2013} }

Version 3.14

Date: November 14th, 2013

Developed by Sivaram Ambikasaran

License

this file, You can obtain one at http://mozilla.org/MPL/2.0/.

DIRECTORIES AND FILES

```
./examples/     :   Example input C++ codes; Needed to read input from user or from input file.
./src/          :   Source code in C++
./header/       :   Relevant header files
./exec/         :   Executables for HODLR
./README.md     :   This file
./LICENSE.md    :   License file
./makefile.mk   :   Makefile
```

Usage

DEPENDENCIES:

To run this package, you need to have **Eigen**. If you don't already have it, download and install Eigen following the instructions here.

BUILD USING CMAKE:

The easiest way to build this library is using CMake. In the project directory, run: "' mkdir build cd build cmake .. make make test [sudo] make install # optional "`this will build the static`hodlr' library and run a few tests. If your version of the Eigen headers is installed in a non-standard place, you can change the `cmake` line to: "' cmake .. -DEIGEN_INCLUDE_DIR_HINTS=/path/to/eigen "'

Your code should include `get_Matrix.hpp` and implement the function "' double get_Matrix_Entry (const unsigned i, const unsigned j) "'

CUSTOM MAKEFILE:

1. There is a sample input file named "HODLR_Test.cpp" in the directory './examples/'. This calls the features the code can handle.
2. Go to the directory where makefile is in, then key in the following command in the terminal:

   ```
   make -f makefile.mk
   ```

3. Once your run the make command, the executables are created in the directory named './exec/'. To run the code, go into the 'exec' directory and call './HODLR_Test'.
4. You can change the kernels in the makefile by changing KERNEL. More kernels can be added by editing the function

   ```
   double get_Matrix_Entry(const unsigned i, const unsigned j)
   ```

   in the file

   ```
     get_Matrix.cpp
   ```

5. The dimension of the problem can be changed by changing DIM in the makefile.
6. Read through the comments in all the files. Most of the function/class/method/variable names are self-explanatory. Read the file HODLR_Test.cpp to understand how to assemble, factor, solver a new HODLR system.

Chapter 4

Hierarchical Index

4.1 Class Hierarchy

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

Chapter 5

Data Structure Index

5.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 6

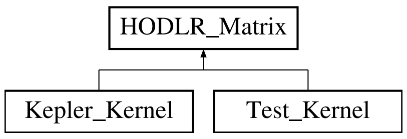
Data Structure Documentation

6.1 HODLR_Matrix Class Reference

This class is for HODLR matrix.

```
#include <HODLR_Matrix.hpp>
```

Inheritance diagram for HODLR_Matrix:



Public Member Functions

- virtual double get_Matrix_Entry (const unsigned i, const unsigned j)
- void get_Matrix (const unsigned start_Row, const unsigned start_Col, const unsigned n_Rows, const unsigned n_Cols, MatrixXd &A)
- void get_Matrix_Row (const unsigned start_Col, const unsigned n_Cols, const unsigned row_Index, VectorXd &v)
- void get_Matrix_Col (const unsigned start_Row, const unsigned n_Rows, const unsigned col_Index, Vector↩ Xd &v)
- unsigned max_Abs_Vector (const VectorXd &v, const vector< int > &not_Allowed_Indices, double &max)

6.1.1 Detailed Description

This class is for HODLR matrix.

Note

Author

$Dan Foreman-Mackey$

Version

Date

Date:

January 31st, 2014

6.1.2 Member Function Documentation

6.1.2.1 void HODLR_Matrix::get_Matrix (const unsigned *start_Row,* const unsigned *start_Col,* const unsigned *n_Rows,* const unsigned *n_Cols,* MatrixXd & *A*) `[inline]`

Allows access to the sub-matrix, i.e., returns a 'n_Rows' by 'n_Cols' sub-matrix starting at the index (start_Row, start_Column) of the HODLR matrix.

6.1.2.2 void HODLR_Matrix::get_Matrix_Col (const unsigned *start_Row,* const unsigned *n_Rows,* const unsigned *col_Index,* VectorXd & *v*) `[inline]`

Allows access to the columns of the HODLR matrix, i.e., returns the 'col_Index'th column with the row starting at 'start_Row' and ending at 'start_Row+n_Rows'.

6.1.2.3 virtual double HODLR_Matrix::get_Matrix_Entry (const unsigned *i,* const unsigned *j*) `[inline]`, `[virtual]`

Allows access to the matrix elements, i.e., returns the (i,j)th element of the HODLR matrix.

Reimplemented in Test_Kernel, and Kepler_Kernel.

6.1.2.4 void HODLR_Matrix::get_Matrix_Row (const unsigned *start_Col,* const unsigned *n_Cols,* const unsigned *row_Index,* VectorXd & *v*) `[inline]`

Allows access to the rows of the HODLR matrix, i.e., returns the 'row_Index'th row with the column starting at 'start_Col' and ending at 'start_Col+n_Cols'.

6.1.2.5 unsigned HODLR_Matrix::max_Abs_Vector (const VectorXd & *v,* const vector< int > & *not_Allowed_Indices,* double & *max*) `[inline]`

Returns the maximum absolute value and the index of maximum element of the vector.

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

- header/HODLR_Matrix.hpp

6.2 HODLR_Node< MatrixType > Class Template Reference

This class is for the node of a HODLR tree, i.e., a diagonal sub-matrix at the appropriate level in the tree.

`#include <HODLR_Node.hpp>`

Public Member Functions

- HODLR_Node (MatrixType ∗kernel, unsigned levelNumber, unsigned nodeNumber, unsigned nStart, unsigned nSize)
- void assemble_Matrices (double lowRankTolerance, VectorXd &diagonal)

- void matrix_Matrix_Product (MatrixXd &x, MatrixXd &b)
- void set_UV_Inversion ()
- void **compute_K** ()
- void compute_Inverse ()
- void apply_Inverse (MatrixXd &matrix, unsigned mStart)
- void compute_Determinant ()
- ∼HODLR_Node ()
- void partial_Piv_LU (const unsigned start_Row, const unsigned start_Col, const unsigned n_Rows, const unsigned n_Cols, const double tolerance, unsigned &computed_Rank, MatrixXd &U, MatrixXd &V)

  Partial pivoted LU to construct low-rank.

Data Fields

- HODLR_Node ∗ **parent**
- HODLR_Node ∗ **child** [2]
- unsigned levelNumber

  Level number of the node;.
- unsigned nodeNumber

  Node number is either 0 or 1;.
- unsigned nStart

  nStart is the starting index of node;
- unsigned nSize

  nSize is the size of the node;
- MatrixXd K

  At leaf level, stores the self interaction; At non-leaf stores the matrix [I, V1inverse∗U1inverse; V0inverse∗U0inverse, I];.
- FullPivLU< MatrixXd > Kinverse

  Stores Factorization of K;.
- double determinant

  Stores K.determinant();.
- unsigned nRank [2]

  nRank[0] rank of K01; nRank[1] rank of K10;
- MatrixXd U [2]

  Column basis of low-rank interaction of children at non-leaf;.
- MatrixXd V [2]

  Row basis of low-rank interaction ofchildren at non-leaf;.
- MatrixXd Uinverse [2]

  Column basis of low-rank interaction of children of inverse at non-leaf.
- MatrixXd Vinverse [2]

  Row basis of low-rank interaction of children of inverse at non-leaf.
- bool isLeaf

  If node is a leaf, it takes the value TRUE;.

6.2.1 Detailed Description

template<typename MatrixType>class HODLR_Node< MatrixType >

This class is for the node of a HODLR tree, i.e., a diagonal sub-matrix at the appropriate level in the tree.

Note

Author

$Sivaram Ambikasaran$

Version

Date

$November 8th, 2013$

Contact: siva.1985@gmail.com

6.2.2 Constructor & Destructor Documentation

6.2.2.1 template<typename MatrixType> **HODLR_Node**< MatrixType >::**HODLR_Node** (MatrixType ∗ *kernel*, unsigned *levelNumber*, unsigned *nodeNumber*, unsigned *nStart*, unsigned *nSize*) `[inline]`

Constructor for the class.

6.2.2.2 template<typename MatrixType> **HODLR_Node**< MatrixType >::∼**HODLR_Node** ( ) `[inline]`

Destructor.

6.2.3 Member Function Documentation

6.2.3.1 template<typename MatrixType> void **HODLR_Node**< MatrixType >::apply_Inverse (MatrixXd & *matrix*, unsigned *mStart*) `[inline]`

Applies the inverse.

6.2.3.2 template<typename MatrixType> void **HODLR_Node**< MatrixType >::assemble_Matrices (double *lowRankTolerance*, VectorXd & *diagonal*) `[inline]`

Assemble the relevant matrices.

6.2.3.3 template<typename MatrixType> void **HODLR_Node**< MatrixType >::compute_Determinant ( ) `[inline]`

Computes the determinant of the matrix.

6.2.3.4 template<typename MatrixType> void **HODLR_Node**< MatrixType >::compute_Inverse ( ) `[inline]`

Computes the inverse.

6.2.3.5 template<typename MatrixType> void **HODLR_Node**< MatrixType >::matrix_Matrix_Product (MatrixXd & *x*, MatrixXd & *b*) `[inline]`

Matrix matrix product.

6.2.3.6 template<typename MatrixType> void HODLR_Node< MatrixType >::partial_Piv_LU (const unsigned *start_Row,* const unsigned *start_Col,* const unsigned *n_Rows,* const unsigned *n_Cols,* const double *tolerance,* unsigned & *computed_Rank,* MatrixXd & *U,* MatrixXd & *V*) `[inline]`

Partial pivoted LU to construct low-rank.

Obtains the low-rank decomposition of the matrix to a desired tolerance using the partial pivoting LU algorithm, i.e., given a sub-matrix 'A' and tolerance 'epsilon', computes matrices 'U' and 'V' such that ||A-UV||_F < epsilon. The norm is Frobenius norm.

start_Row - Starting row of the sub-matrix. start_Col - Starting column of the sub-matrix. n_Rows - Number of rows of the sub-matrix. n_Cols - Number of columns of the sub-matrix. tolerance - Tolerance of low-rank approximation.

computed_Rank - Rank obtained for the given tolerance. U - Matrix forming the column basis. V - Matrix forming the row basis.

If the matrix is small enough, do not do anything

This stores the row indices, which have already been used.

This stores the column indices, which have already been used.

Stores the column basis.

Stores the row basis.

Initialize the matrix norm and the the first row index

Repeat till the desired tolerance is obtained

Generation of the row

Row of the residuum and the pivot column

This randomization is needed if in the middle of the algorithm the row happens to be exactly the linear combination of the previous rows.

Generation of the row

Row of the residuum and the pivot column

Normalizing constant

Generation of the column

Column of the residuum and the pivot row

This randomization is needed if in the middle of the algorithm the columns happens to be exactly the linear combination of the previous columns.

Generation of the column

Column of the residuum and the pivot row

New vectors

New approximation of matrix norm

If the computed_Rank is close to full-rank then return the trivial full-rank decomposition

6.2.3.7 template<typename MatrixType> void HODLR_Node< MatrixType >::set_UV_Inversion ( ) `[inline]`

Set UV Inversion.

6.2.4 Field Documentation

6.2.4.1 template<typename MatrixType> bool HODLR_Node< MatrixType >::isLeaf

If node is a leaf, it takes the value TRUE;.

Variables of interest at leaf;

6.2.4.2 template<typename MatrixType> unsigned HODLR_Node< MatrixType >::levelNumber

Level number of the node;.

Variables of interest for both leaf and non-leaf;

6.2.4.3 template<typename MatrixType> unsigned HODLR_Node< MatrixType >::nRank[2]

nRank[0] rank of K01; nRank[1] rank of K10;

Variables of interest for non-leaf;

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

- header/HODLR_Node.hpp

6.3 HODLR_Tree< MatrixType > Class Template Reference

This class is the main class for the HODLR tree.

`#include <HODLR_Tree.hpp>`

Public Member Functions

- HODLR_Tree (MatrixType ∗kernel, unsigned N, unsigned nLeaf)
- void assemble_Matrix (VectorXd &diagonal, double lowRankTolerance)
- void matMatProduct (MatrixXd &x, MatrixXd &b)
- void compute_Factor ()
- void solve (MatrixXd &b, MatrixXd &x)
- void compute_Determinant (double &determinant)

Friends

- class HODLR_Node< MatrixType >

6.3.1 Detailed Description

template<typename MatrixType>class HODLR_Tree< MatrixType >

This class is the main class for the HODLR tree.

Note

Author

$Sivaram Ambikasaran$

Version

Date

$November 8th, 2013$

Contact: `siva.1985@gmail.com`

6.3.2 Constructor & Destructor Documentation

6.3.2.1 template<typename MatrixType > HODLR_Tree< MatrixType >::HODLR_Tree (MatrixType * *kernel*, unsigned *N*, unsigned *nLeaf*) `[inline]`

Constructor for the HODLR tree.

6.3.3 Member Function Documentation

6.3.3.1 template<typename MatrixType > void HODLR_Tree< MatrixType >::assemble_Matrix (VectorXd & *diagonal*, double *lowRankTolerance*) `[inline]`

Assembles the matrix.

6.3.3.2 template<typename MatrixType > void HODLR_Tree< MatrixType >::compute_Determinant (double & *determinant*) `[inline]`

Computes the determinant.

6.3.3.3 template<typename MatrixType > void HODLR_Tree< MatrixType >::compute_Factor ( ) `[inline]`

Computes the factor of the HODLR matrix.

6.3.3.4 template<typename MatrixType > void HODLR_Tree< MatrixType >::matMatProduct (MatrixXd & *x*, MatrixXd & *b*) `[inline]`

Matrix matrix product.

6.3.3.5 template<typename MatrixType > void HODLR_Tree< MatrixType >::solve (MatrixXd & *b*, MatrixXd & *x*) `[inline]`

Solves the HODLR linear system.

6.3.4 Friends And Related Function Documentation

6.3.4.1 template<typename MatrixType > friend class HODLR_Node< MatrixType > `[friend]`

Variables of interest for the entire tree;

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

- header/HODLR_Tree.hpp

6.4 KDTree Class Reference

This class is for constructing a KDTree in any dimension.

`#include <KDTree.hpp>`

6.4.1 Detailed Description

This class is for constructing a KDTree in any dimension.

Note

Author

$Sivaram Ambikasaran$

Version

Date

$October 22nd, 2013$

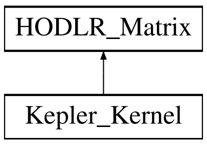
Contact: `siva.1985@gmail.com`

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

- header/KDTree.hpp

6.5 Kepler_Kernel Class Reference

Inheritance diagram for Kepler_Kernel:



Public Member Functions

- **Kepler_Kernel** (vector< double > theta, vector< double > time)
- double get_Matrix_Entry (const unsigned i, const unsigned j)

6.5.1 Member Function Documentation

6.5.1.1 double Kepler_Kernel::get_Matrix_Entry (const unsigned *i*, const unsigned *j*) `[inline]`, `[virtual]`

Allows access to the matrix elements, i.e., returns the (i,j)th element of the HODLR matrix.

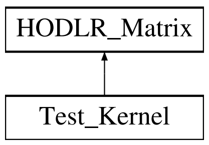
Reimplemented from HODLR_Matrix.

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

- examples/kepler_test.cpp

6.6 Test_Kernel Class Reference

Inheritance diagram for Test_Kernel:



Public Member Functions

- **Test_Kernel** (unsigned N)
- double get_Matrix_Entry (const unsigned i, const unsigned j)

6.6.1 Member Function Documentation

6.6.1.1 double Test_Kernel::get_Matrix_Entry (const unsigned *i*, const unsigned *j*) `[inline]`, `[virtual]`

Allows access to the matrix elements, i.e., returns the (i,j)th element of the HODLR matrix.

Reimplemented from HODLR_Matrix.

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

- examples/HODLR_Test.cpp