Documentation

Accelerate / Creating a sparse matrix from coordinate format arrays

Article

Creating a sparse matrix from coordinate format arrays

Use separate coordinate format arrays to create sparse matrices.

Overview

In some cases — for example, if you're reading matrix values from a file — you may find it easier to create sparse matrix objects from coordinate format arrays. This approach requires three separate arrays: one that contains the column indexes, a second that contains the row indexes, and a third that contains the matrix values. Each array contains the same number of elements.

Create the sparse matrix

The following is an example of a symmetric sparse matrix:

$$\begin{pmatrix} 10.0 & 1.0 & & 2.5 \\ 1.0 & 12.0 & -0.3 & 1.1 \\ & -0.3 & 9.5 & \\ 2.5 & 1.1 & & 6.0 \end{pmatrix}$$

Because this sparse matrix is symmetric, define it with the arrays below that describe its lower triangle. For example, the value `9.5` is in row 2, column 2.

Swift   Objective-C

```
row =      [ 0,   1,   3,    1,    2,   3,   2,   3]
column =   [ 0,   0,   0,    1,    1,   1,   2,   3]
values =   [10.0, 1.0, 2.5, 12.0, -0.3, 1.1, 9.5, 6.0]
```

Use the `attributes` parameter to specify that the matrix is symmetric and the items in the values array derive from the lower triangle.

The following code defines the attributes and creates the sparse matrix instance:

Swift Objective-C

```
var attributes = SparseAttributes_t()
attributes.triangle = SparseLowerTriangle
attributes.kind = SparseSymmetric

var row: [Int32] =      [ 0,   1,   3,    1,    2,   3,   2,   3]
var column: [Int32] =   [ 0,   0,   0,    1,    1,   1,   2,   3]
var values =            [10.0, 1.0, 2.5, 12.0, -0.3, 1.1, 9.5, 6.0]

let blockCount = 8
let blockSize = 1

let A = SparseConvertFromCoordinate(4, 4,
                                    blockCount, UInt8(blockSize),
                                    attributes,
                                    &row, &column,
                                    &values)
```

Invalid and duplicate entries

The system ignores the block element and doesn't include it in the returned matrix if the coordinates `(row[i], column[i])` are invalid, meaning either of the following is true:

- They lie outside the ranges `0..<rowCount` or `0..<columnCount`, respectively.
- `kind` is `SparseTriangular` or `SparseUnitTriangular`, and the coordinates lie in the wrong triangle.

If `kind` is `SparseSymmetric`, the system transposes any entries in the wrong triangle and sums them into the block at `(column[i], row[i])`, if one is present.

The system sums elements with duplicate coordinates and replaces them with a single entry.

The coordinate-conversion functions support block matrices, that is, those with a `blockSize` greater than 1. The described matrix has `rowCount * blockSize` rows and `columnCount * blockSize` columns. For each `i in 0..<blockCount`, there's a structurally nonzero block at block position `(row[i], column[i])` with numerical values `data[i * blockSize *`

`blockSize:(i + 1) * blockSize * blockSize - 1]`. The system interprets the block’s values as the elements of a dense column-major matrix with `blockSize` rows and columns.

Supply a user-defined workspace

There are two variants of each converter. The following functions allocate their own workspace internally and allocate space for the matrices that they return.

- `SparseConvertFromCoordinate(_:_:_:_:_:_:_:_:)` for double-precision, floating-point values
- `SparseConvertFromCoordinate(_:_:_:_:_:_:_:_:)` for single-precision, floating-point values

The following functions require preallocated storage for the matrices that they return and a separate workspace for precise control over allocations:

- `SparseConvertFromCoordinate(_:_:_:_:_:_:_:_:_:_:)` for double-precision, floating-point values
- `SparseConvertFromCoordinate(_:_:_:_:_:_:_:_:_:_:)` for single-precision, floating-point values

See Also

Sparse Matrices

Creating sparse matrices

Create sparse matrices for factorization and solving systems.

Solving systems using direct methods

Use direct methods to solve systems of equations where the coefficient matrix is sparse.

Solving systems using iterative methods

Use iterative methods to solve systems of equations where the coefficient matrix is sparse.

Sparse Solvers

Solve systems of equations where the coefficient matrix is sparse.