Special Components: PDEDiscretizer and LESSolver

ORCAN Workshop

Erlangen, April, 26. - 27. 2005

Jan Treibig, IMMD10

PDEDiscretizer

§ Discretization of a partial differential equation

Interfaces:

§ Create

§ Solution

PDEDiscretizer: Create

```
bool Init( VolMeshRef mesh, LESSolverRef solver);
```

Initialize the object state.

```
bool ReserveMatrixEntries();
```

Allocate storage for the Matrix. Often crucial for performance with CRS formats.

```
bool MakeMatrixEntries();
```

Actually setup the linear equation system by FDM, FVM or FEM.

PDEDiscretizer: Create

```
bool UpdateMatrixEntries();
```

Only partial changing of the already setup matrix.

PDEDiscretizer: Solution

```
bool TransferToMesh();
```

Create a mesh attribute and store the solution vector in the mesh.

LESSolver

- Solving a linear equation system

Interfaces:

- § Reserve
- § Create
- § Solve
- § Query

LESSolver: Reserve

bool SetSize(const uint32& dim);

Set the dimension of the solution vector.

bool ReserveEntry(const uint32& row, const uint32& column) ;

Reserve a matrix entry for row and column.

bool ReserveRow(const uint32& row, const std::vector<uint32>& columns);

Reserve a number of entries at columns in row.

bool Finalize();

Setup the matrix with the reserved size.

LESSolver: Create

bool AddToEntry(const uint32 & row, const uint32 & column,const real64 & value);

Add a value to the matrix at row and column.

bool AddToEntries(const std::vector< uint32> & rows, const std::vector< uint32 > & columns, const std::vector< real64 > & values);

Add a submatrix to the matrix, the values vector has the dimension columns times rows.

bool SetEntry(const uint32 & row,const uint32 & column,const real64 & value)

Set a entry at row and column.

LESSolver: Create

```
bool SetEntries(const std::vector<uint32>& rows, const
     std::vector<uint32>& columns, const std::vector<real64>&
     values);
```

Set a submatrix in the matrix at rows and columns.

```
bool Reset();
```

Reset matrix and RHS entries to zero.

```
bool AddToRHS( const uint32 & index,const real64 & value );
```

Add a value to the RHS vector at index .

```
bool SetRHS(const uint32& index, const real64& value);
```

Set a entry in the RHS vector at index.

LESSolver: Create

```
bool SetSolution( const std::vector< real64 > & solution);
```

Set the Solution vector to an initial guess.

LESSolver: Solve

```
bool SolveLES();
```

Actually solve the linear equation system.

All parameter as the solver to use, the residuum to reach or the maximum number of iterations are set by Properties of the realization of the component.

LESSolver: Query

```
bool GetRHS( const uint32 & index,real64& value);
```

Get a single value of the RHS vector at index.

```
bool GetEntry(const uint32 & row,const uint32 & column,real64 &
    value);
```

Get a single value of the matrix at row and column.

```
bool GetSolution(std::vector< double > & in );
```

Get the solution vector.

```
uint32 GetSize();
```

Get the dimension of the linear equation system.

Getting realizations and interfaces

```
//Get a reference to a realization
ocs::LESSolverRef  mSolv=ocs::LESSolver::New();
if( !mSolv ) {
    OCERROR( "no Solver available");
    return false;
}

//Get InterfacePointer
ocs::LESSolver::Interfaces::QueryPtrType sq;
if( !(sq=mSolver.I.QueryPtr) ) {
    OCERROR("Solver Query interface required!");
    return false;
}
```

How it works together

```
std::cout<<"Initializing LES... \n";
//*********************************
SolverReserve->SetSize(mq->GetNumVertices());
PDECreate->Init(mVol, mSolv);
PDECreate->ReserveMatrixEntries();
SolverReserve->Finalize();
PDECreate->MakeMatrixEntries();
//*********************************
std::cout<<"Solving LES!"<<std::endl;
//*********************************
SolverSolve->SolveLES();
PDESolution->TransferToMesh();
//*********************************
```