

# BaseMatrix & BaseVector

A way to store, centralize and manipulate mechanical data in solvers

---

- SOFA provides a powerful abstract interface to manipulate data in the solvers.
- Performances of this method are based on the fact that the data, stored in the mechanical state, is also manipulated in the state itself. The mathematical operations launched in the solvers are transmitted to the corresponding Object that compute the treatment locally.
- This method prevents data centralization and thus is mainly oriented to performances and distributed execution.

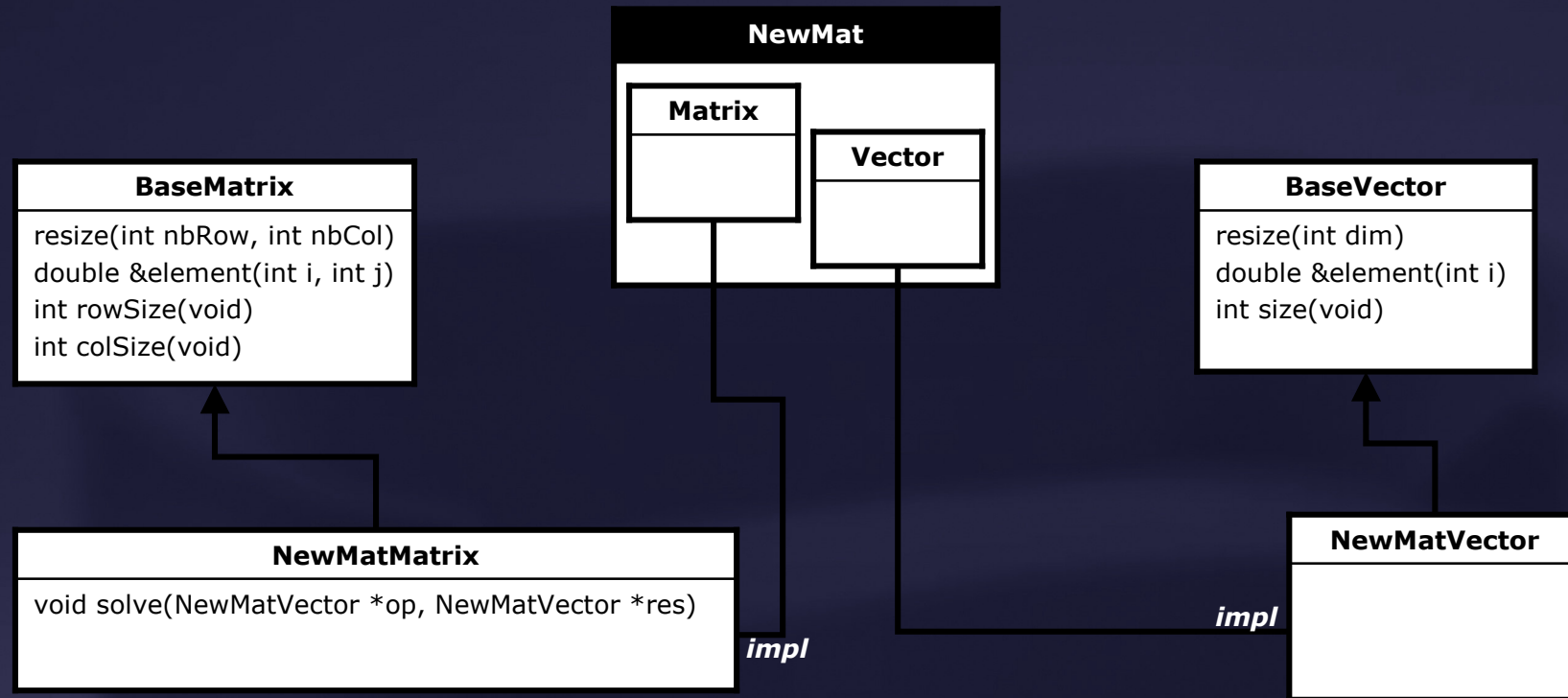
but...

- Mechanical simulation requires also Matrix / Vector systems resolutions.
- Solvers should be more « flexible » to allow a way to centralize data, and provide basic manipulations of raw data structures.
- To be efficient, the use of powerful Matrix libraries should also be possible in SOFA.

# BaseMatrix & BaseVector

A way to store, centralize and manipulate mechanical data in solvers

The current proposal : BaseMatrix & BaseVector Interface



```
NewMatMatrixSolver
addMBK_ToMatrix(BaseMatrix *A, double mFact, double bFact, double kFact, unsigned int offset)
addMBKdx_ToVector(BaseVector *V, VecId dx, double mFact, double bFact, double kFact, unsigned int offset)
getMatrixDimension(unsigned int * const, unsigned int * const)
multiVector2BaseVector(VecId src, BaseVector *dest, unsigned int offset)
multiVectorPeqBaseVector(VecId dest, BaseVector *src, unsigned int offset)
```

# BaseMatrix & BaseVector

A way to store, centralize and manipulate mechanical data in solvers

---

The current proposal : Associated Mechanical Actions and their corresponding SOFA base components interfaces

## MechanicalAddMBK\_ToMatrixAction

## MechanicalAddMBKdx\_ToVectorAction

### BaseMass

```
addMToMatrix(BaseMatrix * mat, double mFact, unsigned int offset)
addMDxToVector(BaseVector * resVect, const VecDeriv& dx, double mFact, unsigned int offset)
```

### BaseForceField

```
addKToMatrix(BaseMatrix * mat, double kFact, unsigned int offset)
addKDxToVector(BaseVector * resVect, const VecDeriv& dx, double kFact, unsigned int offset)
```

### BaseConstraint

```
applyConstraint(BaseMatrix *mat, unsigned int offset)
applyConstraint(defaulttype::BaseVector *vect, unsigned int offset)
```

# BaseMatrix & BaseVector

A way to store, centralize and manipulate mechanical data in solvers

---

The current proposal : Associated Mechanical Actions and their corresponding SOFA base components interfaces

**MechanicalMultiVector2BaseVectorAction**



**MechanicalMultiVectorPeqBaseVectorAction**



**MechanicalGetMatrixDimensionAction**



**BaseMechanicalState**

```
contributeToMatrixDimension(unsigned int *, unsigned int *)  
loadInBaseVector(BaseVector *dest, VecId src, unsigned int offset)  
addBaseVectorToState(VecId dest, BaseVector *src, unsigned int offset)  
setOffset(unsigned int &offset)
```

# BaseMatrix & BaseVector

A way to store, centralize and manipulate mechanical data in solvers

---

The current proposal : example of solver using matrix

```
MatrixStaticSolver::MatrixStaticSolver()
{
    mat = new NewMatMatrix();
    op = new NewMatVector();
    res = new NewMatVector();
}

void MatrixStaticSolver::solve(double dt)
{
    MultiVector f(group, VecId::force());
    MultiVector pos(group, VecId::position());

    computeForce(f);

    getMatrixDimension(&nbRow, &nbCol);

    mat->resize(nbRow, nbCol);
    op->resize(nbRow);
    res->resize(nbRow);

    addMBK_ToMatrix(mat,0,0,1);

    projectResponse(f);

    multiVector2BaseVector(f, op);

    mat->solve(op, res);

    multiVectorPeqBaseVector(pos, res);
}
```