NemoMath - Enhancement #623
Allow inversion rules inside MathVector<T>

10/12/2011 03:37 PM - M. Rolf

Status:	Closed	**Start date:**	10/12/2011
Priority:	Low	**Due date:**	
Assignee:	M. Rolf	**% Done:**	0%
Category:		**Estimated time:**	0.00 hour
Target version:	NemoMath 0.4		

Description

==> Do not have to specify full inversion rules like

```
leftAddInverse<MathVector<TYPE> >
```

every time a new TYPE is used inside MathVector

History

#1 - 10/19/2011 03:58 PM - M. Rolf

- Target version set to NemoMath 0.2

#2 - 10/27/2011 01:05 PM - M. Rolf

- Target version changed from NemoMath 0.2 to NemoMath 0.4

#3 - 11/08/2011 04:15 PM - M. Rolf

Reduce compilation time by NOT instanciating Vectors/Matrices over several types T before absolutely necessary.

Possible solution:
Call with typename-inference leftAddInverse(MathVector<T>) and provide implementations inside MathVector.
The compiler should find them as 'specializations' by ADL, similar to the swap(ReadWriteRef) implementation in MathVector.

#4 - 02/05/2013 02:27 PM - M. Rolf

- Status changed from New to Closed

Something like this inside MathVector<T> was intended

```
friend MathVector<T> leftAddInverse(const MathVector<T> &value){return -value;}
```

Yet, this particular piece of code doesn't, because it is not recognized as *template specialization* of leftAddInverse<D>.

This version tries to do that specialization, but doesn't compile inside MathVector:

```
template<> MathVector<T> rightAddInverse<MathVector<T> >(const MathVector<T> &value){return -value;}
```

GCC says:

> error: explicit specialization in non-namespace scope ‘class nemo::MathVector<T>’

You cannot specialize in a different namespace (here class-namespace) that the generic declaration (::nemo space).
C++03 specification:

> An explicit specialization shall be declared in the namespace of which the template is a member, or, for member templates, in the namespace of which the enclosing class or enclosing class template is a member.

Doesn't seem to be possible.

#5 - 02/05/2013 02:40 PM - M. Rolf

In order to reduce the compilation overhead, now only two types inside MathVector come with ready-made inversion rules:

1. int
2. double

These are also the two types that have ready-made typedefs IntVector and RealVector.

All other integer types can still be conveniently supplied with inversion rules (for addition/subtraction) by stating

> _nemo_vector_inversion_(TYPE)

in the client code.
The same holds for floating point types (addition/subtraction/multiplication/division) with

> _nemo_vector_cwise_inversion_(TYPE)