

Changes to the VSIPL++ Specification¹

CodeSourcery, LLC

4th April 2005

¹This specification developed under subcontract 601-02-S-0109 under U.S. Government contract F30602-00-D-0221.

© 2004 by Georgia Tech Research Corporation. All rights reserved

A non-exclusive, non-royalty bearing license is hereby granted to all persons to copy, distribute, and produce derivative works for any purpose, provided that this copyright notice and following disclaimer appear on all copies:

THIS LICENSE INCLUDES NO WARRANTIES, EXPRESSED OR IMPLIED, WHETHER ORAL OR WRITTEN, WITH RESPECT TO THE SOFTWARE OR OTHER MATERIAL INCLUDING, BUT NOT LIMITED TO, ANY IMPLIED WARRANTIES OR MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE, OR ARISING FROM A COURSE OF PERFORMANCE OR DEALING, OR FROM USAGE OR TRADE, OR OF NON-INFRINGEMENT OF ANY PATENTS OF THIRD PARTIES. THE INFORMATION IN THIS DOCUMENT SHOULD NOT BE CONSTRUED AS A COMMITMENT OF DEVELOPMENT BY ANY OF THE ABOVE PARTIES.

The U.S. Government has a license under these copyrights, and this material may be reproduced by or for the U.S. Government.

Published 2004

Printed in the United States of America

11 10 09 08 07 06 05 04

5 4 3 2 1

-
- 1 This document contains short descriptions of changes to the VSIPL++ specification, both serial and parallel portions. Minor changes, e.g., spelling mistakes, incorrect words, are not listed.

2003 Sep 02

- 2 Change the names of `Block` and `Cyclic` to `BlockDistribution` and `CyclicDistribution` in the parallel [view].
- 3 Add a `vsipl` constructor taking command-line arguments to [initfin].

2003 Sep 08

- 4 Add references to VSIPL mathematical descriptions of various functions and classes, e.g., `atan2`, complex exponential, FFTs, IIR filters, etc.

2003 Sep 12

- 5 Define subsubblocks in [view.distribute].

2003 Sep 18

- 6 In [selgen.clip.clip] and [selgen.clip.invclip], add `Clip` and `InverseClip` accessors.
- 7 In [selgen.clip.clip] and [selgen.clip.invclip], make the operators `const` functions.
- 8 In [selgen.manipulation], the parameters cannot be `const` since these views' values are swapped.
- 9 In [view.dense], [view.vector], [view.matrix], and [view.tensor], split the `Dense`, `Vector`, `Matrix`, and `Tensor` constructors. One constructor requires a value used to initialize all the values. The other causes the block or view to have unspecified initial values. The user is responsible for ensuring any particular value is set before reading.

2003 Sep 19

- 10 The `mat_op_t` enumeration was moved earlier in `<vsip/solvers.hpp>`. There is no change to the user interface.
- 11 In `[math.solvers.qr]`, the enumerated values of `storage_t` were changed from `NONE`, `SKINNY`, and `FULL` to `QRD_NOSAVEQ`, `QRD_SAVEQ1`, and `QRD_SAVEQ`, reflecting the `VSIPL` names. The enumerated values of `product_side_t` were changed from `LEFT` and `RIGHT` to `MAT_LSIDE` and `MAT_RSIDE`.
- 12 In `[math.solvers.svd]`, the enumerated values of `storage_t` were changed from `NONE`, `SKINNY`, and `FULL` to `SVD_UVNOS`, `SVD_UVPART`, and `SVD_UVFULL`. The enumerated values of `product_side` were changed from `LEFT` and `RIGHT` to `MAT_LSIDE` and `MAT_RSIDE`.

2003 Sep 22

- 13 In `[conventions.requirements]`: The contents of the `impl` namespace is unspecified, not implementation defined.

2003 Oct 02

- 14 In `[domains.domain.comparison]`: Fixed the variables used in the definition to make it correct.

2003 Oct 07

- 15 In `[domains.definitions.conformant]`: Remove the example.
- 16 In `[domains.domain.comparison]`: Correct the definition of `element_conformant`.
- 17 In `[initfin]`: Clarify the effect of calling `vsip1` and `~vsip1` when other such objects exist or do not exist.

2003 Nov 10

- 18 In [conventions.defs], refer to VSIPL 1.1, not VSIPL 1.10.
- 19 Move definition of correspondence from [domains.definitions.conformant] to [domains.index.correspond]. In [domains.domain.accessors], clarify the result of `size()`. In [domains.domain.equality], add an example of equality.
- 20 In [view.block], add a note how a map's dimensions beyond a block's will be ignored. In [view.view], add a note that a subview's domain is not necessarily a subset of the view. In [view.dense.constructors], clarify the returned block's index ordering. In [view.dense.accessors], specify one-dimensional `get` for multi-dimensional blocks and add `increment_count`. In [view.vector.accessors], [view.vector.assign], and [view.matrix.assign], note the corresponding VSIPL functions. In [view.vector.valaccess], [view.matrix.valaccess], and [view.tensor.valaccess], the accessor index must be strictly less than, not less than or equal to, the corresponding dimension.
- 21 In [view.vector], [view.matrix], and [view.tensor], revise the constructors for the one-line-create-it-all-at-once specializations to either take an initial value xor not. Some of the more general constructors may have also been changed.
- 22 Specify transposition enumerated values in a new subclause [math.solvers.enum].
- 23 In [math.solvers.qr], revise to use enumerated values with names similar to VSIPL enumerations. In [math.solvers.qr], revise the constructors' exception specifications so only a constructor, not a copy or assignment, can throw an exception. `decompose` can modify its argument.
- 24 In [math.solvers.svd], revise the enumerations to use VSIPL-like names.
- 25 [signal.fft.enumer] was renamed [signal.fft.enum] to reflect other similarly named sections. Similarly, [signal.convolver.enumer] was renamed [signal.convolver.enum].

2003 Nov 19

- 26 Add missing paragraph numbers to this document.
- 27 Modify header file synopses [views.vector], [views.matrix], and [views.tensor] to include and use internal type definitions for `Vector`'s, `Matrix`'s, and `Tensor`'s subviews. Modify [views.vector.subview_types], [views.matrix.subview_types],

and [views.tensor.subview_types] to specify these internal type definitions. Modify return types in [views.vector.subviews], [views.matrix.subviews], and [views.tensor.subviews] to use these internal type definitions.

2003 Nov 20

- 28 `Vector<T,Block>::length`, a synonym for `Vector<T,Block>::size()`, is added to [views.vector.accessors].
- 29 Clarify the exposition of instantiation restrictions in [selgen.generate.gather] and [selgen.generate.scatter].

2003 Nov 21

- 30 In [math.matvec] and [math.solvers], fix typos in the block and type template parameters. Add notes that the `Block` template parameters may differ from the returned views' blocks. Do the same in [selgen.generate], [selgen.clip.clip], and [selgen.clip.invclip].

2003 Nov 24

- 31 In [view.vector.assign] and [view.matrix.assign], the instantiation restrictions for assignments with view operands were corrected to only require support for `scalar_i`, `scalar_f`, and `cscalar_f`. In [view.tensor.assign], no assignment operators must be instantiated. These restrictions reflect the C VSIP interface.

2003 Nov 25

- 32 In [view.vector.subviews], [view.matrix.subviews], and [view.tensor.subviews], change the throw specifications from `VSIP_NOTHROW` to `VSIP_THROW((std::bad_alloc))` since memory allocation might be necessary. An exception is thrown if the allocation fails.
- 33 In [view.tensor.assign], change the return types from a mistaken `Vector` to `Tensor`.

2003 Dec 14

- 34 In [view.block], two additional block requirements to yield the block's size were added. `b.size()` must yield the total number of values accessible from the block. `b.size(dimension_t d)` yields the number of values in the specified dimension. Views must implement similar accessors, probably just by calling these accessors. [view.dense] and [view.dense.accessors] were similarly updated.
- 35 [view.dense.accessors] was split into [view.dense.valaccess] specifying value accessors, e.g., `get` and `put`, and [view.dense.accessors] specifying other accessors.
- 36 In [view.view], the return type of `v.size()` and `v.size(dimension_t d)` was corrected to be `length`, not `index_t`. [view.vector], [view.matrix], and [view.tensor] already specified a return type of `length`.

2003 Dec 15

- 37 In [math.solvers], enumeration constants `TRANSPOSE` and `NO_TRANSPOSE` were changed to `MAT_TRANS` and `MAT_NTRANS`. This should have been changed Nov 10.
- 38 In [math.solvers.qr], correct the throw specifications for `BY_REFERENCE` member functions. These functions use a return value, not exceptions, to indicate success or failure.
- 39 In [view.vector.constructors], [view.matrix.constructors], and [view.tensor.constructors], add constructors taking views with different block types, e.g., `template <typename T0, typename Block0> Vector(const Vector<T0, Block0>&)`. Prior to this change, only copying a view with the same value type and block type was supported.

These added constructors' semantics are more similar to those for `operator=s` than the copy constructors. The constructed object must be modifiable and will be element conformant to the operand. Its values will equal corresponding values in the operand.

The constructed object is not a subview and will not use reference semantics. Thus, changing a value in the constructed view will not change the corresponding value in the operand given for its construction.

-
- 40 Specify allocatable blocks in [view.block.alloc]. Some view constructors allocate their underlying blocks. These blocks must satisfy the interface presented in this clause.

In [view.vector.constructors], [view.matrix.constructors], and [view.tensor.constructors], for constructors not taking a block argument, require allocatable blocks and remove all throw specifications. Since the allocatable block can throw any exception when constructed, the view constructors cannot have throw restrictions.

2003 Dec 16

- 41 Remove the initial view name from subview internal type definitions to shorten the type definitions. For example, `vector_subview_type` was changed to `subview_t` since `Vector` is already known. [views.vector.subview_types], [views.matrix.subview_types], [views.tensor.subview_types], [views.vector.subviews], [views.matrix.subviews], and [views.tensor.subviews] were modified.

2004 Feb 09

- 42 In [math.fns.reductions], revise the required template specializations for `sumval` and `sumsqval` to reflect VSIPL functions.
- 43 In [math.fns.reductions], revise the operand types for `maxmgsqval`, `maxmgval`, `minmgsqval`, and `minmgval` to reflect VSIPL functions.
- 44 In [support.types], require `cscalar_i` to be equivalent to `complex<scalar_i>`.
- 45 In [view.view], an assignment operator should not invoke `increment_block` nor `decrement_block`.
- 46 In [view.dense.constructors], the constructed `Domains` are specified.

2004 Feb 10

- 47 In [math.solvers], change non-const view parameters in `solve` and other functions from reference parameters to call-by-value parameters. View copying se-

antics are pass-by-reference; using a reference parameter prevents using these semantics.

- 48 In [selgen.selection.first], remove the throw specification since the given function object may throw an exception.
- 49 In [selgen.selection.scatter], add a `View` template parameter to `scatter`.

2004 Feb 14

- 50 In [view.tensor.constructors], copy constructors and assignment operators need not be have any specializations, i.e., they need not be supported. This is because the VSIPL 1.1 API need not support copying tensors.
- 51 The parameter types in the declaration of `jmul` in [math.fn] differed from the restriction in [math.fns.scalar]. These have been changed to both take parameters with template types `T1` and `T2`. If these are not complex, the implementation can specify any definition consist with the requirement to return “the product of `a` with the conjugate of `b`.” In other words, `jmul` will operate like `mul` when the second argument is not complex.
- 52 In [math.fns.elementwise], the declaration of `cmplx` was corrected to permit various view and view value types. The declaration in [math.fns] was correct.
- 53 In [math.fns.scalarview], synonymous functions listed in Table 8.1 are added. For example, `cadd` and `cdiv` were added.
- 54 In [math.fns.reductions], the definition of accumulation operator was rewritten to avoid unsupported syntax.
- 55 In [math.fns.operators], correct the `operator~` and the `operator^` restriction. Also, the operators in Table 8.6 must also obey [math.fns.operators]/1, i.e., they must have element-wise extensions.
- 56 In [math.matvec.dot] and [math.matvec.hermitian], correct the specialization syntax. For example, `cvjdot<scalar_f, scalar_f>` is not correct since the function template requires four template arguments.
- 57 In [math.matvec.hermitian], [math.matvec.kron], [math.matvec.outer], [math.matvec.product], and [math.matvec.vmmul] remove the incorrect requirements about blocks.

-
- 58 In `[math.matvec.product]`, `prod4` views must have four, not three, rows or columns.
- 59 In `[math.solvers.covsol]`, `[math.solvers.llsqsol]`, and `[math.solvers.toeplitz]`, `non-const` parameters should be passed by value without a reference because views already implement pass-by-reference semantics. These views' blocks must be modifiable.
- 60 In `[math.solvers.lu]`, removed the unnecessary template parameter for the constructor. `decompose` takes a modifiable `non-const` parameter.
- 61 In `[math.solvers.cholesky]`, `decompose` takes a modifiable `non-const` parameter and must decompose according to `this->get_uplo()`.
- 62 In `[math.solvers.qr]`, removed the unnecessary template parameter for the constructor. `decompose` takes a modifiable `non-const` parameter. The `BY_REFERENCE` version of `lsqsol` returns a `bool`, not a `Matrix`.
- 63 In `[math.solvers.svd]`, removed the unnecessary template parameter for the constructor. `decompose`'s first parameter must be modifiable.
- 64 In `[signal.fft.operators]`, `unit stride` is now required only if `InputType` and `OutputType` differ.
- 65 In `[signal.correl.accessor]`, correct the exposition for `reference_size()`. In `[signal.correl.operators]` and `[signal.iir.operators]`, remove the reference in the return type.
- 66 In `[signal.windows]`, the returned `Vectors`' types are unspecified, not implementation-defined.

2004 Feb 17

- 67 In `[view.view]` add a requirement that a D-dimensional view uses a D-dimensional block.
- 68 In `[view.block]/5`, typographical errors were fixed. Also, `b.size(d)` was changed to `b.size<X>(d)`. A block can value accesses for various numbers of `index_ts`. For example, a block may be a 1,3-dimensional block. When asking for the size of a particular dimension, the answer depends on how one views the block (pun intended), e.g., as a 1-dimensional block or as a 3-dimensional block.

The template argument specifies the dimension, e.g., 1 or 3. When asking for the total number of values in the block, the desired dimensionality is not important.

Corresponding changes were also made to [view.dense], [view.dense.template], and [view.dense.accessors].

- 69 [math.enum] replaced [math.solvers.enum] because the same enumeration was needed in both [math.matvec] and [math.solvers]. Also, MAT_HERM and MAT_CONJ were added to support `gemp` and `gems`.
- 70 [math.matvec.gem] specifies a generalized matrix product function and a generalized matrix sum function.
- 71 [math.fns.userelt] was completely revised.
- 72 In [view.map], `subblock_iterator` and `processor_iterator` must be constant iterators, i.e., they do not support assigning to their values. This is because the VSIP++ library, not its user, should control changing these values.
- 73 *Subdomain* was defined in [domains.definition.subdomain]. In [domains.definition.overlap], the definitions were made more precise.
- 74 In [view.dense.valaccess], [view.vector.valaccess], [view.matrix.valaccess], and [view.tensor.valaccess], added notes indicating that `get` and `put` must obey the block and view requirements on them.
- 75 In [view.view], the description of subview reference semantics was changed. For a view `v` and a subview `subv`, the values are the same `Index` are not necessarily the same. For example, in a matrix transpose subview, the value at `(3, 5)` in the transpose and the source view are not necessarily the same. Instead, the value in the transpose at `(3, 5)` is the same as the value at `(5, 3)` in the source view.
- 76 In [view.vector.subviews], [view.matrix.subviews], and [view.tensor.subviews], subviews' domains are specified. These have zero-based and unit stride `Indexes`. For example, a `Vector` subview `s` of `Vector v` having every other value will have `Indexes` `0, 1, ..., v.size()/2` corresponding to `v` `Indexes` `0, 2, ..., 2 * (v.size()/2)`.

2004 Feb 19

- 77 Some compilers had difficulty parsing member function templates so the block `size<Dim>(dim)` member function template was changed to a non-template

member function `size(Dim, dim)`. Changes were made to `[view.block]/5` and `[dense.accessors]`.

2004 Apr 21

78 In `[view.dense.accessors]`, clarify which `get` and `put` accessors must be supported.

2004 Apr 28

79 In `[math.matvec.outer]`, correct the return value for complex operands to reflect the VSIPPL standard. If both `Vector` operands contain `cscalar_f` values, then Hermitian conjugate is used. Otherwise, transpose is used.

2004 Jun 08

80 In `[math.fns]`, add declarations of `band`, `bnot`, and `bor`. In `[math.fns.elementwise]`, add definitions of `band`, `bnot`, `bor`, and `bxor`. Consequently, in `[math.fns.scalar]`, revise specifications for `land`, `band`, `lnot`, `bnot`, `lor`, `bor`, `lxor`, and `bxor`.

In `[math.fns.reductions]`, revise `alltrue` and `anytrue` to use `land` or `band` and `lor` or `bor`, respectively.

In `[math.fns.operators]`, remove `land`, `lnot`, and `lor` from list of overloaded logical operators with multiple overloaded symbolic operators. Change underlying overloaded assignment operators for views to use the bitwise, not logical, versions.

This resolves issue 8-6.

81 In `[domain.arithmetic.shift]`, add `operator-`. In `[domain.arithmetic.scale]`, add `operator/`. This resolves issue 5-8.

82 In `[view.matrix.assign]`, correct the return type for assignment operators with a scalar operand to `Matrix`.

83 In `[math.fns.scalar]`, correct `fmod`'s return statement to implement the corresponding VSIPPL specification.

In `[math.fns.operators]`, remove overloading of `%` for `fmod`. Such an overloading is problematic because C++'s `%` operates only on integers while `fmod` need only

operate on floating-point values. In [view.vector.assign], [view.matrix.assign], and [view.tensor.assign], remove the overloading of % for fmod.

This resolves issue 8-3.

84 In [math.fns], add bold, introductory comments before functions to ease finding functions. This resolves issue 8-4.

85 In [math.fns], remove cadd, cdiv, cexp, clog, cmag, cmagsq, cmul, cneg, csqrt, and csub. [math.fns.scalar] and [math.fns.scalarview], similar changes were made. This resolves issue 8-8.

86 In [domains.arithmetic.shift], eliminate the reference to a production implementation. This resolves issue 5-9.

2004 Jun 09

87 Add summary tables of contents.

2004 Jun 11

88 In [signal.fft], change “folding frequency” to “Nyquist frequency.” This resolves issue 11-5.

89 In [math.solvers.cholesky], correct the spelling of Hermitian. This resolves issue 8-21.

90 In [signal], ensure the template parameters for FIR and IIR have the same names. This resolves issue 11-1.

91 In [math.enum], revise MAT_TRANS to indicate only matrix transpose. MAT_HERM is used for Hermitian transpose. In [math.solvers.lu], [math.solvers.qr], and [math.solvers.svd], require MAT_HERM, not MAT_TRANS, given complex values. This partially resolves issue 8-2. See also 2004Jun28.

92 In [conventions.struct.requirements], note undefined behavior if required constraints are violated. This resolves issue 1-1.

93 In [selgen.generate], note VSIPL’s fill functions are not needed. This resolves issue 9-1.

-
- 94 In [random.rand.constructors], change “generator” to “generator object”. This resolves issue 10-1.
- 95 In [conventions.struct.spec], remove “Complexity” element and description. Such requirements are removed from [domain], [math], [random], [selgen], [signal], [support], and [view].
- 96 Throughout the document, add spaces between C++ code and English-language punctuation. This addresses issue 10-2.
- 97 In [complex], `recttopolar` and `polartorect` were added. These convert between rectangular and polar representations. The name is slightly misleading because VSIPL++ does not specify the underlying complex representation. `cmplx` was removed from [math.fns.elementwise] because `polartorect` has the same functionality. This addresses issue 7-1. (See also 2004 Jun 18.)
- 98 In [support.exceptions], add a note concerning VSIPL development and production modes. In [intro.compliance], add a requirement that memory-allocation errors must be reported. These address issues 1-2 and 1-3.
- 99 In [random.rand.constructors], correct “implementation dependent” to “implementation defined.” In [support.exceptions], note that exceptions can indicate errors. This resolves issue 2-1.
- 100 In [math.matvec.gem], remove the `prod` notation. This resolves issue 8-14.

2004 Jun 14

- 101 In [math.solvers.lu], [math.solvers.cholesky], [math.solvers.qr], and [math.solvers.svd], change “most recent decomposition” to “most recent decomposition for this object.” This resolves issue 8-19.
- 102 In [math.solvers.cholesky], change `get_half` to `uplo`. This resolves issue 8-20.
- 103 In [math.solvers.svd], change ρ to p . This resolves issue 8-23.
- 104 In [selgen.clip], replace the `Clip` and `InverseClip` classes by `clip` and `invclip` functions. This resolves issue 9-2.
- 105 In [signal], replace “mimic” by “implement.” This resolves issue 11-7.
- 106 In [signal.fir.enumer] and [signal.iir], note that filters support continuous filtering. This resolves issue 11-2.

-
- 107 In [domain.index], change “location in a Domain” to an “element in a Domain. In [domain.domainone.position], add more text explaining positions. These resolve issue 5-6.
- 108 In [signal.fft], change `cscalar_f` and `scalar_f` to `complex<T>` and `T`. This resolves issue 11-3.
- 109 In [conventions.defs], define “undefined.”
- 110 Format the headers and footers to avoid long headers.
- 111 In [math.solvers.qr], note undefined behavior for `QRD_NOSAVEQ`. This resolves issue 8-22.

2004 Jun 17

- 112 In [support.types.domain], change `index_t`'s definition from `unsigned long int` to `size_t`. This resolves issue 3-1.

2004 Jun 18

- 113 In [support.types.dimorder], declare `tuple`, `ROW1`, `ROW2`, ..., `COL1`, `COL2`, ... to add support for dimension ordering, e.g., row-major ordering. In [view.dense.template], specify the `Dense Major` template parameter and specify the meaning of the tuple, i.e., the underlying storage ordering. This resolves issue 6-2.
- 114 In [support.exceptions], add a note that implementations that do not support exceptions must provide an implementation-defined way to report memory allocation errors to the user. This also appears in [intro.compliance]. This supplements the issue 1-3 resolution.
- 115 In [math.fns.elementwise], add `cmplx`, which should not have been removed 2004 Jun 11. This corrects issue 7-1.
- 116 In [view.block], restrict blocks to store only one type of value. This resolves issue 6-4.
- 117 In [view], note “block” and “view” are interfaces, not classes. This resolves issue 6-7.

-
- 118 In [view.matrix.subviews], add notes that `transpose` has reference semantics so most implementations will implement by reordering indices. This resolves issue 6-14.

2004 Jun 21

- 119 In [view.matrix.subviews], add notes to the transpose implementations in [math.matvec.transpose]. This augments the resolution of issue 6-14.
- 120 In [view.block], add a note pointing to [view.map]. This resolves issue 6-6. (This was further supplemented 2004Jun28.)
- 121 In [view.vector.constructors], [view.matrix.constructors], and [view.tensor.constructors], specify the sizes of views constructed from blocks.
- 122 In [math.solvers.cholesky], change a constructor's throw specification to permit throwing `std::bad_alloc`.
- 123 In [math.solvers.qr], change `decompose`'s exception specification from `std::bad_alloc` to forbid throwing exceptions.

2004 Jun 22

- 124 Rename [math.matvec.hermitian] to [math.matvec.transpose]. Add the `trans` transposition function. This resolves issue 8-13 and also a 19May2004 Forum decision.

2004 Jun 23

- 125 In [math.matvec.transpose], add a cross reference to an alternate transposition implementation in [view.matrix.subviews].
- 126 Thoroughly revise [signal.fft], splitting into [signal.fft] and [signal.fftm]. [signal.fft] specifies FFTs applied single views. [signal.fftm] specifies multiple FFTs. Template parameters were changed. FFT criteria tables were revised, with more lines added. In-place, by-reference operators were added. VSIPL restrictions on unit stride were removed. This resolves issues 11-4 and 11-6.

-
- 127 In [support], add [support.constants], specifying ROW, COL, DIM0, DIM1, This resolves issue 8-17.
- 128 In [support] and [support.types.domain], `index_difference_t` should be a signed version of `index`, not a specific type. This was suggested by Randy Judd. This supplements issue 3-1.
- 129 In [view.block], specify the interaction of the two `get` functions for an x,y -dimensional block. This partially addresses issue 6-13.

2004 Jun 28

- 130 In [math.enum], correct description of MAT_HERM. This resolves issue 8-2.
- 131 In [view], add notes defining a map and uniprocessor program use of maps. This resolves issue 6-6, but also see the 2004Jun21 changes.

2004 Jul 06

- 132 In [random.rand.constructors], add a note explaining the `numproc` and `id` parameters.

2004 Jul 07

- 133 Enumerations were moved out of classes into the `vsip` namespace. Some were moved into [math.enum]. Other modifications occurred in [math.solvers.cholesky], [math.solvers.qr], [math.solvers.svd], and [signal.correl].
- 134 [signal.fir.enumer] was renamed [signal.fir.enum] to improve consistency.
- 135 The `<vsip/signal.hpp>` in [signal]/2 was corrected to be in `vsip` namespace.

2004 Jul 08

- 136 In [math.solvers.cholesky], note that `mat_uplo` applies to symmetric or Hermitian matrices.

137 In [math.matvec.dot], add a note that, for `cvjdot`, using a `scalar_f` template argument yields a complex Vector. In [math.matvec.transpose], similarly modify `herm`. In [math.matvec.product], similarly modify `prodh` and `prodj`. This resolves issue 8-10.

138 In [view.block], [view.block.alloc], and [view.view], clarify block reference counting. An explicitly created block uses reference counting but obeys the usual C++ / C rules for creation. It is destroyed when it leaves scope or when it is explicitly deallocated, e.g., via `delete`, despite its reference count. An implicitly-created block created by a view constructor will be heap-allocated and should be destroyed by VSIPL++ when its net reference count equals negative one. In [view.dense], specify `~Dense`. This resolves Issue 6-9.

2004 Jul 09

139 For [math.solvers.cholesky] reference the upper or lower matrix, not store it.

2004 Jul 12

140 In [view.view], add a note explaining subviews, copies, and assignments. This resolves issue 6-12.

2004 Jul 15

141 In [view.vector.vectormap], [view.matrix.vectormap], and [view.tensor.vectormap], correct the specification to support constructors not having an initial value. Modify constructors to use `const_View`, not `const View&`.

142 Dense blocks with user-specified were added to [view.dense]. Two Dense constructors taking pointers were added to [view.dense.constructors]. [view.dense.userdata] specifies related definitions, `admit`, `release`, `find`, and `rebind`. Predicates `admitted` and `single_pointer` were added to [view.dense.accessors]. The term *complex type* is defined in [conventions.defs]. This resolves Issue 6-1.

2004 Jul 19

- 143 Modify the definitions of modifiable and non-modifiable views in [view.views]. Change [view.vector] to specify both `const_Vector` and `Vector`. Make similar to changes to [view.matrix] and [view.tensor]. Add [view.vector.convert], [view.matrix.convert], and [view.tensor.convert] to define class `ViewConversion` to convert between modifiable and non-modifiable views, e.g., relating `const_Vector` and `Vector`. (See also a 2004 Aug 25 entry for `Tensor` corrections.)

Modify [complex], [math], [random], [selgen], and [signal], modifying input views to be constant views. In [complex], modify output views to use pass-by-value. This resolves issue 6-5.

- 144 In [math.solvers.lu], [math.solvers.cholesky], [math.solvers.qr], and [math.solvers.svd], correct all solver constructors to throw `std::bad_alloc` upon failure.

- 145 In [math.solvers.lu], the “Effects” requirement was corrected to use `m`, not `answer`.

2004 Jul 20

- 146 In [domains.definitions.conformant], add one more sentence to the definition of product conformant. This resolves Issue 5-3.

old name	new name
<code>get_distribution</code>	<code>distribution</code>
<code>get_length</code>	<code>length</code>
<code>get_stride</code>	<code>stride</code>
<code>get_support</code>	<code>support</code>
<code>get_symmetry</code>	<code>symmetry</code>

`_t` suffixes were added to `ViewConversion` internal types.

This resolves Issue 5-7.

- 148 In `[random.rand]`, specify `random.hpp` as the header file name, add and use internal view type definitions, and correct the constructor throw specifications.

2004 Jul 23

- 149 In `[view.vector.subview_types]`, `[view.vector.subview]`, `[view.matrix.subview_types]`, `[view.matrix.subview]`, `[view.tensor.subview_types]`, and `[view.tensor.subview]`, specify real and imaginary subviews of *Vector*, *Matrix*, and *Tensor*. These subviews permit either read only or read and write access to a portion of another view. No copies occur.

- 150 Change “addition conformant” to “element conformant.” This resolves Issue 5-1.

2004 Jul 25

- 151 In `[view.view]`, define exactly overlapping views. In `[view.view.assign]`, define order-dependent and order-independent assignments.

In `[math.solvers.covsol]` and `[math.solvers.llsqsol]`, forbid operand overlaps. These prohibitions are not required by VSIPL.

In `[math.solvers.lu]`, forbid operand overlaps for the `BY_REFERENCE solve`. This prohibition is not required by VSIPL. In `[math.solvers.cholesky]`, forbid operand overlaps for the `BY_REFERENCE solve`. In `[math.solvers.qr]`, forbid operand overlaps for the `BY_REFERENCE prodq`, `rsol`, `covsol`, and `lsqsol`.

In `[math.solvers.svd]`, forbid overlaps for the `BY_REFERENCE decompose`. This prohibition is not required by VSIPL. In `[math.solvers.svd]`, forbid overlaps for the `BY_REFERENCE produ` and `prodv`.

In [selgen.selection.scatter], forbid overlaps of the constant views with the output non-constant view. It is not clear if VSIPL has these restrictions.

This resolves Issue 5-2.

- 152 Require `arg`, `atan`, and `atan2` to meet the VSIPL specification. Those specifications are subject to modification after outstanding issues are resolved. This resolves Issue 8-7.

2004 Jul 27

- 153 In [random.rand], specify a default template argument. In [random.rand.generate], add requirements for positive lengths.
- 154 In [view.vector.constructors], add an instantiation restriction for assignment from a `T0` constant. This restriction mimics the restriction for the similar function assigning from a `T`.

2004 Jul 28

- 155 In [view.tensor], correct the `real` and `imag` subview accessors for `const_Tensor` and `Tensor`. Constant `real` and `imag` accessors are available for both `const_Tensor` and `Tensor`. Non-constant `real` and `imag` are available for `Tensor`. [view.tensor.subview_types] was completely rewritten to be analogous to [view.matrix.subview_types].

In [views.vector.subview_types] and [views.matrix.subview_types], correct the explanation of `const_realview_t` and `const_imagview_t`. In [views.matrix.subview_types], prepend `const_` for some types and improve the placement of “non-modifiable.”

In [views.vector.constructors], [views.matrix.constructors], and [views.tensor.constructors], correct the Effects specifications for copying from a view with a different block.

Most of these changes are related to Issue 6-5 and the introduction of real and imaginary subviews (2004Jul23).

2004 Jul 29

- 156 In [math.matvec.gem] and for `gemv`, correct the size requirements for `Matrix C`.

2004 Aug 02

- 157 In [view.dense.accessors], the Dense accessor `user_storage()` was added. This accessor yields `true` if and only if the block has user-specified storage. In [view.dense.constructors], the constructor postconditions were updated to indicate this accessor's result on each constructed block.
- 158 In [random.rand.view.types], correct the actual view types from `Vector` and `Matrix` to `const_Vector` and `const_Matrix`.
- 159 In [signal.iir.template], restrict the required specializations of `T` to `scalar_f`.

2004 Aug 04

- 160 In [signal.convolve], change the name of the compile-time constants `symmetry` and `support` to `symmtry` and `supprt`. This avoids a name conflict with the accessors. In [signal.correl], change the name of the compile-time constants `support` to `supprt`. This avoids a name conflict with the accessors.
- 161 In [signal.histo], add default template parameters.

2004 Aug 10

- 162 Ensure the table of contents lines can be broken.
- 163 Ensure blank pages at the end of a chapter have no headers.
- 164 In [math.fns.scalar], remove the extraneous description of `T1` and `T2` for `land`. For the note in `sqrt` and `sub`, correct the corresponding VSIPL functions.
- 165 In [math.fns.reductions] and [math.fns.reductidx], remove “scalar versions of” in the notes describing corresponding VSIPL functions.

2004 Aug 11

166 In [view.dense.accessors], correct `decrement_count` to deallocate the block if the count becomes negative, not zero. This is because [view.block.alloc]/7 permits a use count of zero for implicitly-created blocks.

In [view.matrix.constructors] and [view.tensor.constructors], explicitly note that the block's use count is decremented. This is required by [view.view] so no new requirements were added; existing requirements are repeated.

167 In [views.matrix]'s `<vsip/matrix.hpp>`, correct `realview_type` and `imagview_type` to `realview_t` and `imagview_t`. These should have been changed 2004Jul21.

168 In the 2004Jul21 entry in this file, add conversions from `col_type`, `row_type`, and `diag_type` to `col_t`, `row_t`, and `diag_t`.

2004 Aug 12

169 In [convention.defs], define a complex type's underlying type. This is used for defining Dense blocks with user-specified storage.

[view.dense.uservocab], defining terms related to user-specified storage, was added before constructor restrictions. These definitions were moved from [view.dense.userdata].

In [view.dense.constructors], the four constructors were reordered to better reflect their symmetries and their order in the header file. In the note for the constructor taking only a `Domain` and a map, shorten the note, removing language about assigning before use. Added an initial note indicating that all user-defined storage definitions are presented in [view.dense.userdata]. The new constructor taking one `uT` pointer supports a complex interleaved format. The last constructor was modified to require two pointers to a complex number's underlying type `uT`, not `T`.

In [view.dense.userdata], the third and fourth items were merged to define three explicit formats: array format, interleaved format, and split format. Non-complex types must be represented using the array format, which is just a set of contiguous values. Complex types can be represented in any of the three formats. For complex numbers, the array format contains contiguous complex values. The interleaved format interleaves real and imaginary portions of complex numbers in a contiguous array. These two formats differ because the C++ and the VSIPL++ specification do not require any particular representation of complex numbers, but the interleaved

format does. The split format uses two arrays to store separately real and imaginary portions of complex numbers.

enum `user_storage_t` differentiates these three formats. The first enumerated value `NO_USER_STORAGE` equals `false` so that the result of the `this->user_storage()` accessor can be used in a boolean statement. The other values are `ARRAY_FORMAT`, `INTERLEAVED_FORMAT`, and `SPLIT_FORMAT`. In `[views.dense.accessors]`, the `user_storage` accessor returning a `user_storage_t` value was defined, obviating the `single_pointer` accessor.

The definition of an admit-release sequence was simplified by removing the “subsequence” terminology. Such terminology was no longer needed after resolving the underlying VSIPL functionality of releasing an already released block. Furthermore, “subsequence” is overloaded to mean both a contiguous sequence of values within a sequence and non-contiguous within a sequence. This ambiguity makes “subsequence” much less useful.

The incorrect uses of “undefined” values were changed to “unspecified” values. The values are defined; it’s just not specified exactly what they are.

For the `release` restrictions, the terminology regarding `this->map()` and distributed blocks was removed. The restrictions were changed to differentiate the block’s state using `this->user_storage()`. A fourth `release` taking a `uT` pointer was added to support `INTERLEAVED_FORMAT`. The `release` variant taking two pointers was changed to take two `uT` pointer references, not two `T` pointer references.

In the `find` restrictions, removed the notes about not using the function to determine whether the block has user-specified storage. One can now use the `user_storage` accessor. A third `find(uT*&)` function joined the two existing `find` functions. This function supports the `INTERLEAVED_FORMAT`. The two-pointer `find` version was changed to require `uT` pointers, not `T` pointers.

Revise the `rebind` restrictions analogously to `release` and `find`. Add a third `rebind` taking a `uT` pointer to support `INTERLEAVED_FORMAT`. Revise the other function restrictions to use `this->user_storage`.

- 170 In `[views.block]`, correct the subscripts to run from 1 to the end, not from 0 to the end less one. The difficulty is that using variable names to indicate subscripts and also indicating subtraction is difficult.

2004 Aug 12

- 171 Remove the draft status from the title pages. Remove the summary tables of contents. Write forwards to both the main and parallel sections.

2004 Aug 16

- 172 On the reverse of the title pages, a copyright notice was added. Remove the rights notice from [intro.ack].

2004 Aug 18

- 173 In [conventions.struct.spec]/3, move “Returns” before “Postconditions” to reflect use throughout the specification.

2004 Aug 19

- 174 In [domains.arithmetic.shift], add a requirement for operator- that $dm.first() + (-difference) + dm.stride() * (dm.length() - 1) \geq 0$.
- 175 In [view.dense] and [view.dense.accessors], change `Dense<...>::size`'s return type from `index_t` to `length_t`. This reflects the block requirements in [view.block]. (VSIPL++ users will probably not experience any difficulties because `index_t` and `length_t` are usually implemented using the same underlying type.)
- 176 In [view.block], correct the references to parallel sections, i.e., from [view.matrix.vectormap] and [view.tensor.vectormap] to [view.matrix.matrixmap] and [view.tensor.tensormap].
- 177 In [view.dense.template], further restrict the supported types for `Dense`'s `T` template parameter that must be supported. For `Dense<1, ...>`, the set of required instantiations did not change. For dimensions `D` greater than one, the set of required instantiations for `T` contains at least `scalar_f`, `scalar_i`, `cscalar_f`, `cscalar_i`, and `bool`. This change is because `Dense` must support `get` and `put`. VSIPL blocks do not directly support `get` and `put` so an implementation based on VSIPL will use a VSIPL matrix or tensor.

2004 Aug 23

- 178 In [domains.domainone.comparison], add `product_conformant`. This ensures the `Domain<1>` interface is a superset of the `Domain<D>` interface.
- 179 Move VSIP++ scalar functions without corresponding VSIP scalar functions from [math.fns.scalar] to a new section [math.fns.elements]. These functions are not required to be present in a VSIP++ implementation. These functions are `am`, `land`, `band`, `eq`, `euler`, `expoavg`, `ge`, `gt`, `le`, `lt`, `ma`, `maxmg`, `maxmgsq`, `minmg`, `minmgsq`, `msb`, `ne`, `lnot`, `bnot`, `lor`, `bor`, `sbm`, `sq`, `lxor`, and `bxor`.
- For some VSIP++ scalar functions, reduce the set of required instantiations to those corresponding to VSIP scalar functions. Other instantiations are moved to a new section [math.fns.elements]. These functions are `add`, `div`, `mul`, `neg`, `recip`, and `sub`.
- `jmul` was changed so both its scalar and element-wise versions accept non-complex arguments.
- Create a new subclass [math.fns.elements] containing scalar functions that are not required to be in the `vsip` namespace but are used when specifying element-wise extensions or other uses.
- In [math.fns.elementwise]/1 and [math.fns.scalarwise]/1, update the reference to scalar functions to include the new section [math.fns.elements].
- In [math.fns.operators], remove the example describing which overloads are required, which is now too complicated to describe succinctly.
- 180 Improving wording in the main specification's forward per James Lebak's comments. Basically, the fourth paragraph's last sentence listing distributed terms was removed.
- 181 In [view.view] and [view.block], change "view's contents" and "block's contents" to "view's data" and "block's data."
- 182 In [view.block]/8, add an open parenthesis after `==` and before `j_1`.

2004 Aug 25

- 183 In [view.view.assign], indicate that function calls on the right-hand side of an assignment must complete before a collective assignment occurs. Element-wise

functions are excepted so that their loops can be fused together if the VSIPL++ implementation wishes to do so. For example, element-wise addition, element-wise multiplication, and `ramp` can all be fused together. Invoking an FFT, a convolution, or a matrix multiplication must complete before the rest of an assignment occurs.

184 In `[math.fns.elementwise]`, remove incorrect requirements concerning overlapping views and blocks. These requirements are not needed. For example, consider the operation $v + w$. The result is a view that need have nothing to do with v and w . There are no read-write or write-read dependences because there is no assignment. Also remove incorrect requirements in `[math.matvec.transpose]` and `[math.matvec.vmmul]`.

185 [*Note:* Order-independence refers only to assignments, whether explicit or implicit, not to ordinary operations. Thus, this concept is specified in `[view.view.assign]`, not in the descriptions of element-wise operations. Implicit assignments, e.g., `gemp(alpha, A, B, beta, C)` require the output block does not overlap the input blocks.

For an (explicit) order-independent assignment, requiring the left-hand side's block and the right-hand side's block to either exactly overlap or not overlap at all is sufficient.]

186 In `[view.matrix.constructors]`, remove the incorrect note that the *Matrix* constructor taking a block yields a modifiable view depending on the block's modifiability. Actually, a *Matrix* is modifiable, and a `const_Matrix` is not modifiable. A similar change was made in `[view.tensor.constructors]`. The *Vector* specification was already correct.

In `[view.vector.constructors]`, `[view.matrix.constructors]`, and `[view.tensor.constructors]`, replace the view constructors taking blocks. For each view, add `View(Block)` and `const_View(Block)` constructors. The former, creating a modifiable view, accepts only modifiable blocks. The latter accepts modifiable and non-modifiable blocks but the resulting view will be non-modifiable. These changes explicitly prohibit constructing a modifiable view using a non-modifiable block. Using such a combination is problematic. For example, the modifiable view's `put` operator might be invoked but the underlying block has no corresponding operator to implement the operation.

2004 Sep 01

187 The math function declarations at the beginning of `[math.fns]` were revised.

-
- All overloaded operators automatically provided by C++ were removed. These scalar functions include `operator&&`, `operator&`, `operator==`, `operator>=`, `operator>=`, `operator>`, `operator<=`, `operator<`, `operator!=`, `operator!`, `operator~`, `operator|`, `operator|`, `operator-`, and `operator^`.
 - One scalar function declaration should have removed when the function specification was moved to `[math.fns.element]`. These was `euler`.
 - Added math function declarations for operator overload synonyms of element-wise functions with scalar and view operands. Added declarations included `operator+` for `add`, `operator/` for `div`, `operator*` for `mul`, and `operator-` for `sub`.

- 188 In `[math.fns.elements]`, note that implementers may wish to add `eq` for complex arguments since `==` is already present.
- 189 In `[math.fns.elementwise]`, remove the incorrect requirement comparing input and output view blocks. There are no output views in element-wise extensions. Correct the return type of `cmplx` from `View` to `const_View`.
- 190 In `[math.fns.scalarview]`, correct `am`'s extension. Prior to the change, it incorrectly described an extension to a scalar and a view. After the change, it describes an extension to a view, a scalar, and a view.
- 191 In `[math.fns.operators]`, add synonym requirements for `land (&&)`, `band (&)`, `lnot (!)`, `bnot (~)`, `lor (| |)`, and `bor (|)`. This change is connected with the next change.
- 192 The last table in `[math.fns.operators]` was removed. This table specified overloaded operators for the bitwise variants of assignment operators. This section should have been removed when splitting logical and bitwise functions into separate functions. These functions are already declared in a previous table in this section.
- 193 Throughout `[view]`, change `this` to `*this`. The meaning does not change. In both cases it refers to the object, but the latter is used throughout the document.
- 194 In `[view.tensor]`, add a paragraph indicating the meaning of *Tensor*, i.e., either `Tensor` or `const_Tensor`. Correct `Tensor<complex<T>, Block>::realview_t` to `Tensor<Tp, unspecified>` and `Tensor<complex<T>, Block>::const_realview_t` to `const_Tensor<Tp, unspecified>`. Make similar changes to `imagview_t` and `const_imagview_t`.

In [view.tensor.template], change `Tensor` to *Tensor*.

In [view.tensor.subview_types], make similar corrections for `submatrix_t` and `transpose_t`.

In [view.tensor.constructors], make these same corrections to `Tensor(length_t z_length, length_t y_length, length_t x_length, const T& value)`, `Tensor(length_t z_length, length_t y_length, length_t x_length)`, `Tensor(const Tensor<T,Block>& t)`, and `~Tensor`. Correct the specification of `Tensor(const Tensor<T0, Block0>& t)`, removing an erroneous value and requiring `Block`, but not `*this`, to be modifiable. `*this` is already modifiable because it is a `Tensor`, not a `const_Tensor`. For [view.tensor.constructors]'s `operator=s`, remove a requirement that `*this` must be modifiable since these operations are present only for the modifiable `Tensor`, not the non-modifiable `const_Tensor`.

In [view.tensor.assign], add an introductory sentence that `Tensor`, not `const_Tensor`, has assignment operators. For every function, remove the extraneous statement that `*this` must be modifiable.

In [view.tensor.transpose], change `Tensor` to *Tensor*. Change the one specification of `transpose` to separate specifications for `const` and non-`const` versions.

In [view.tensor.subviews], for the `const` functions except `real` and `imag`, change `Tensor` to *Tensor* and omit the type of the resulting subview object. For the non-`const` functions, add requirements the objects must have type `Tensor`, not `const_Tensor`.

In [view.tensor.transpose], correct the mapping between dimensions z , y , x and 0, 1, and 2.

2004 Sep 14

195 Add a royalty-free license statement to the copyright statements.

2004 Oct 11

196 Correct the spelling of the title of the “Foreword” section.

2005 Jan 5

- 197 In [math.fns.reductions] for the `alltrue` reduction, change the definition of the base value when `T` is a `bool` from `~(T())` to `!(T())`

2005 Feb 24

- 198 Split restriction environment into separate prototype and nrestriction environments to improve formatting of function prototypes.
- 199 In [signal.correl] fix correlation operators to use `const_View` template parameter (operator() was referring to 'View')
- 200 [conventions.requirements] new paragraph to reserve class member names beginning with `impl_` for implementation
- 201 In [support.macros]/1 clarify `VSIP_MAX_DIMENSION` note: change from “may not” to “must not”, and add description that value must be “positive integral literal value”.
- 202 Removed [view.view.assign]/5 unnecessary requirement on view assignment.
- 203 In [math.fns.elements] replace “Implementers” with “Implementors”. In [view.view]/15 replace “New C++ programmers” with “Users who are new to C++”.
- 204 Change names of correlation, convolution classes to be consistent with other `VSIP++` class names.
[signal.convol] change class name “convolution” to “Convolution”.
[signal.correl] change class name “correlation” to “Correlation”.
[random] change class name “rand” to “Rand”.
- 205 In [signal.window] change function prototype for Cheby to be consistent with other window functions.
- 206 In [view.block] change relationship between x-dimensionality and y-dimensionality of an x,y-dimension block to be dependent on block’s dimensionality.
- 207 [math.fns.reductions] Add `meansqval` prototype for complex to scalar case.
[math.fns.reductions] Add `sumsqval` prototype for complex to scalar case.

208 [view.dense.uservocab]/7, include the empty sequence of admit and release calls in the definition of a released block.

209 Reference counting changes:

In [view.dense.accessors], change blocks to deallocate when count goes to zero.

In [view.block.alloc]/6, for blocks allocated by a view constructor, change effects of creation to include the effect of invoking `increment_count`.

In [view.vector.constructors], [view.matrix.constructors], and [view.tensor.constructors] add note: Blocks are created with the effect of `increment_count`, views do not invoke `increment_count` again for blocks it allocates.

210 In [conventions.requirements], add new paragraph to requirement header include-guards.

211 In [math.fns.scalarview], add requirement for `add(const_View, T)`, `mul(const_View, T)`, and `sub(const_View, T)`.

2005 Mar 7

212 Update reference to parallel specification in forward.

213 In [view.vector]/2: change `const`Vector::block()` return type to `”block`t const&”`

In [view.vector.accessor]: add `”const block`t&”` return form of `block()`.

In [view.matrix]/2: change `const`Matrix::block()` return type to `”block`t const&”`

In [view.matrix.accessor]: add `”const block`t&”` return form of `block()`.

In [view.tensor]/2: change `const`Matrix::block()` return type to `”block`t const&”`

In [view.tensor.accessor]: add `”const block`t&”` return form of `block()`.

2005 Mar 8

214 in [support]/1: change `processor_t` and `subblock_t` typedefs from `unsigned long int` to implementation-defined.

2005 Mar 15

- 215 Changed typename suffix from “t” to “type”.
- 216 In [view.view.assign]/4, clarify that it is the user’s responsibility to ensure that collective assignments are order-independent.

2005 Apr 4

- 217 Several changes for spelling mistakes and incorrect words.