CONTENTS

ビジュアルヘルプ – 分析（4） 2
疎行列 2
疎行列の概念 2
疎行列のフォーマット 2
COO 疎行列ストレージフォーマット 3
CSR 疎行列ストレージフォーマット 3
CSC 疎行列ストレージフォーマット 4
疎行列の例 5
MatrixSparse コマンドのリスト 5
MatrixSparse の入力 6
MatrixSparse コマンドのデータ形式 7
MatrixSparse インデックスのデータ形式 7
MatrixSparse 変換 7
オプションの疎行列に関する情報 8
MatrixSparse コマンド 8
MatrixSparse ADD 9
MatrixSparse ADD の例 9
MatrixSparse MM 10
MatrixSparse MM の例 10
MatrixSparse MV 10
MatrixSparse MV の例 11
MatrixSparse SMSM 11
MatrixSparse SMSM の例 11
MatrixSparse TOCOO 12
MatrixSparse TOCOO の例 12
MatrixSparse TOCSC 12
MatrixSparse TOCSC の例 13
MatrixSparse TOCSR 13
MatrixSparse TOCSR の例 13
MatrixSparse TODENSE 14
MatrixSparse TODENSE の例 14
MatrixSparse TRSV 14
MatrixSparse TRSV の例 15

ビジュアルヘルプ – 分析（4）

疎行列

一部のアプリケーションでは、要素の大部分が 0 である大規模な行列の操作が必要となります。
こうしたアプリケーションでは、疎行列を使うことでパフォーマンスが向上し、メモリ使用量が削減されます。

Igor は、Igor Pro 9.0 で追加された MatrixSparse コマンドを通じて疎行列をサポートします。
これは Intel Math Kernel Library Sparse BLAS ルーチンを使い、ライブラリの用語と規約を採用しています。

疎行列の概念

このセクションでは、Igor に適用される疎行列の基本概念について説明します。
疎行列に関する一般的な入門については、https://en.wikipedia.org/wiki/Sparse_matrix を参照してください。

Igor における通常の行列は 2D ウェーブであり、行列の各要素はメモリ内に規則的な行と列のパターンで格納されます。
疎行列の文脈では、通常の行列を指すために「密行列」という用語を使います。

Igor における疎行列は、行列の非ゼロ要素を定義する 3 つの 1D ウェーブの集合によって表現されます。
Igor は疎行列の表現のために、以下に説明する 3 つの形式をサポートしています。
これらの形式は COO（座標形式）、CSC（圧縮列形式）、CSR（圧縮行形式）と呼ばれます。
以下のセクションでは、「疎行列」という用語を、これらの形式のいずれかに従って 3 つの 1D ウェーブによって定義される行列を意味するものとして使います。

MatrixSparse の TOCOO、TOCSR、TOCSC 変換コマンドを使って、密行列に相当する疎行列を作成できます。
または、 3 つの 1D ウェーブを作成し、適切な値をそれらに格納することで直接作成することもできます。
TODENSE 変換コマンドを使って、疎行列に相当する密行列を作成できます。

MatrixSparse は、ADD（2 つの疎行列の加算）、MV（疎行列とベクトルの乗算）、SMSM（2 つの疎行列の乗算）、TRSV（連立一次方程式の解法）など、数多くの数学演算をサポートしています。

疎行列演算は、単精度および倍精度の実数および複素数データに対して動作します。
INF や NaN はサポートしていません。

疎行列のフォーマット

Igor における疎行列は、行列の非ゼロ要素を定義する 3 つの 1D ウェーブの集合で表現されます。
Igor は COO（座標形式）、CSR（圧縮行形式）、CSC（圧縮列形式）という 3 つの疎行列保存形式をサポートしています。

MatrixSparse は、形式間の変換を行う操作を除き、CSR 形式を使います。

これらの形式を説明するために、Wikipedia の疎行列ウェブページにある例示用の密行列を使います。

```
0  0  0  0
5  8  0  0
0  0  3  0
0  6  0  0
```

nnz は「非ゼロ値の数（number of non-zero values）」の略語であり、以下および MatrixSparse コマンドのドキュメントに記載されています。
この例では、nnz = 4 です。

COO 疎行列ストレージフォーマット

「COO」は「座標フォーマット（coordinate format）」の略称です。
概念的には最も単純なフォーマットであり、非ゼロの行列値を、対応するゼロベースの行と列のインデックスと共に格納します。

Igor の用語では、COO フォーマットは以下の 3 つのウェーブを使います。

W_COOValues :　　行列内の各非ゼロ値を格納します。

W_COORow　　行列内の各非ゼロ値に対応するゼロベースの行インデックスを格納します。

W_COOColumns :　　行列内の各非ゼロ値に対応するゼロベースの列インデックスを格納します。

例の行列は COO フォーマットで次のように表わされます。

```
W_COOValues:          5  8  3  6
W_COORows:            1  1  2  3
W_COOColumns:         0  1  2  1
```

これらの値は、順序付きトリプレットの集合として縦方向に読み取ることができます：(5,1,0)、(8,1,1)、(3,2,2)、(6,3,1)。
最後のものは、値 6 が 3 行目 1 列目の値であることを示しています。

ウェーブの名前 W_COOValues、W_COORows、W_COOColumns は、MatrixSparse が COO フォーマットで出力疎行列を作成する時に使われます。
入力疎行列を指定する場合、任意のウェーブの名前を使用できます。

CSR 疎行列ストレージフォーマット

「CSR」は「圧縮疎行列（compressed sparse row）」の略称です。
メモリ使用量と計算速度の面で COO よりも効率的であるため、広く利用されています。

MatrixSparse は、フォーマット間の変換を行う操作を除き、CSR フォーマットを使います。

CSR フォーマットでは、3 つの 1D ウェーブが非ゼロ値、ゼロベースの列インデックス、および各値がどの行に現れるかを決定するために使われる「ポインター」ベクトルを格納します。

Igor の用語では、CSR フォーマットは以下の 3 つのウェーブを使います。

W_CSRValues :　　行列内の各非ゼロ値を格納します。

W_CSRColumns :　　行列内の各非ゼロ値に対応するゼロベースの列インデックスを格納します。

W_CSRPointerB :　　特定の値がどの行に存在するかを決定するために使われる W_CSRValues へのインデックスを格納します。

例の行列は CSR フォーマットで以下のように表わされます。

```
W_CSRValues:          5  8  3  6
W_CSRColumns:         0  1  2  1
W_CSRPointerB:        0  2  3  4
```

COO フォーマットの場合とは異なり、これらの値は順序付きトリプレットの集合として縦方向に読み取ることができません。
CSR はもう少し複雑です。

これらのウェーブのうち最初の 2 つは順序対として読み取れます：(5,0)、(8,1)、(3,2)、(6,1)。
各順序対は値（例：6）と列（例：1）を指定しますが、その値がどの行に現れるかは示しません。

第 3 のウェーブ、W_CSRPointerB は、与えられた値がどの行に現れるかを決定するために使われます。
これは、表現される行列の各行に対する 1 つのインデックスと、行列内の非ゼロ値の数である nnz という追加のインデックスを含みます。
W_CSRPointerB[i] は、行 I の最初の非ゼロ値の W_CSRValues におけるゼロベースのインデックスです。

この例では、W_CSRPointerB の値を次のように解釈できます。

```
W_CSRPointerB[0] = 0        // 行 0 の最初の値は W_CSRValues のインデックス 0 に位置する *
W_CSRPointerB[1] = 0        // 行 1 の最初の値は W_CSRValues のインデックス 0 に位置する
W_CSRPointerB[2] = 2        // 行 2 の最初の値は W_CSRValues のインデックス 2 に位置する
W_CSRPointerB[3] = 3        // 行 3 の最初の値は W_CSRValues のインデックス 3 に位置する
W_CSRPointerB[4] = 4        // W_CSRValues 内の非ゼロ値の数は 4
```

* 行 0 には非ゼロ値が存在しないため、W_CSRPointerB[0] は W_CSRPointerB[1] と同じです。

MatrixSparse コマンドに CSR フォーマットの疎行列を指定する場合、nnz を指定する W_CSRPointerB の最後の要素はオプションであり、省略可能です。

ウェーブ名 W_CSRValues、W_CSRColumns、W_CSRPointerB は、MatrixSparse が CSR フォーマットで出力疎行列を作成する時に使われます。
入力疎行列を指定する場合、任意のウェーブ名を使用できます。

CSC 疎行列ストレージフォーマット

「CSC」は「圧縮疎列（compressed sparse column）」の略称です。
メモリ使用量と計算速度の面で、COO よりも効率的です。

Igor の用語では、CSC フォーマットは以下の 3 つのウェーブを使います。

W_CSCValues：	行列内の各非ゼロ値を格納します。
W_CSCRows：	行列内の各非ゼロ値に対応するゼロベースの行インデックスを格納します。
W_CSCPointerB：	特定の値がどの列に存在するかを決定するために使われる、W_CSCValues へのインデックスを格納します。

W_CSCPointerB ウェーブは、CSC において、CSR における W_CSRPointerB と同様の動作をします。

MatrixSparse コマンドに CSC フォーマットの疎行列を指定する場合、nnz を指定する W_CSCPointerB の最後の要素はオプションであり、省略可能です。

ウェーブの名前 W_CSCValues、W_CSCRows、W_CSCPointerB は、MatrixSparse が CSC フォーマットで出力疎行列を作成する時に使われます。
入力疎行列を指定する場合、任意のウェーブの名前を使用できます。

疎行列の例

MatrixSparse コマンドの動作を理解していただくために、MatrixSparse MV コマンドを使った疎行列とベクトルの乗算を示す簡単な例を以下に示します。

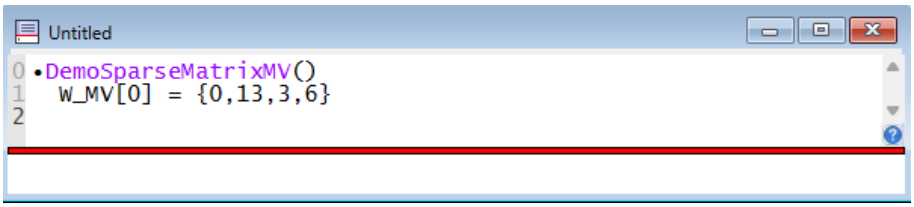
```
Function DemoSparseMatrixMV()
        // CSR フォーマットで Wikipedia の例の疎行列を定義
        Make/FREE/D values = {5, 8, 3, 6}       // 倍精度浮動小数点
        Make/FREE/L columns = {0, 1, 2, 1}      // 64-bit 符号あり整数
        Make/FREE/L ptrB = {0, 0, 2, 3, 4}      // 64-bit 符号あり整数

        // ベクトルを作成
        Make/FREE/D vector = {1, 1, 1, 1}      // 倍精度浮動小数点

        // 疎行列をベクトルで乗算
        MatrixSparse rowsA=4, colsA=4, csrA={values,columns,ptrB}, vectorX=vector, operation=MV

        // 出力疎行列用のウェーブ参照を作成
        WAVE W_MV      // MatrixSparse コマンド MV からの出力

        Print W_MV     // W_MV[0]= {0,13,3,6} を出力
End
```



先頭にある 3 つの Make コマンドは、自由ウェーブを使って CSR フォーマットの疎行列を定義します。
ウェーブの値は、単精度または倍精度浮動小数点、実数または複素数で、INF や NaN を含んではなりません。
この場合、インデックス、列、ptrB を含むウェーブは、64 ビットの符号付き整数でなければなりません。

次の Make コマンドは、values ウェーブと同じデータ型を持つベクトルを作成します。

疎入力行列は、rowsA、colsA、csrA キーワードによって定義されます。

入力ベクトルは vectorX キーワードで指定されます。

コマンドキーワードは実行する操作を指定します。
この場合は MV です。

この場合の出力を、現在のディレクトリに作成される W_MV という名前のウェーブとします。
これは values ウェーブと同じデータ型のベクトルです。

異なる MatrixSparse コマンドは、異なる入力を必要とし、異なる出力を生成します。

MatrixSparse コマンドのリスト

MatrixSparse がサポートするコマンドは以下の通りです。

コマンド	**動作**
ADD	2 つの疎行列を加算し、疎な出力行列を生成します。 詳細は MatrixSparse ADD のヘルプを参照してください。
MM	疎行列と密行列の積を計算し、密行列を出力行列として生成します。 詳細は MatrixSparse MM のヘルプを参照してください。

MV　　疎行列とベクトルの積を計算し、疎な出力行列を生成します。
詳細は MatrixSparse MV のヘルプを参照してください。

SMSM　　２つの疎行列の積を計算し、疎な出力行列を生成します。
詳細は MatrixSparse SMSM のヘルプを参照してください。

TOCOO　　入力行列（密行列、CSC フォーマット、または CSR フォーマット）に対応する COO フォーマットの疎な出力行列を生成します。
詳細は MatrixSparse TOCOO のヘルプを参照してください。

TOCSC　　入力行列（密行列、COO フォーマット、または CSR フォーマット）に対応する CSC フォーマットの疎な出力行列を生成します。
詳細は MatrixSparse TOCSC のヘルプを参照してください。

TOCSR　　入力行列（密行列、COO フォーマット、または CSC フォーマット）に対応する CSR フォーマットの疎な出力行列を生成します。
詳細は MatrixSparse TOCSR のヘルプを参照してください。

TODENSE　　疎な入力行列（COO、CSC、または CSR フォーマット）に相当する密な出力行列を生成します。
詳細は MatrixSparse TODENSE のヘルプを参照してください。

TRSV　　三角疎行列入力に対する連立一次方程式を解きます。
詳細は MatrixSparse TRSV のヘルプを参照してください。

MatrixSparse の入力

MatrixSparse コマンドへの入力は文書化されており、以下の概念的な行列、ベクトル、スカラー入力として理解できます。

Sparse matrix A は、rowsA と colsA キーワードと、cooA、csrA または cscA キーワードのいずれかによって定義されます。
上記の DemoSparseMatrixMV 例におけるこのコマンドでは、Sparse matrix A を指定するキーワードが鍵となります。

```
MatrixSparse rowsA=4, colsA=4, csrA={values,columns,ptrB}, vectorX=vector, operation=MV
```

Sparse matrix A は、１つ以上の疎行列入力を取るすべての MatrixSparse コマンドで使われます。
MatrixSparse 数学コマンド（ADD、MV、MM、SMSM、TRSV）では、入力疎行列が CSR フォーマットである必要があります。

Sparse matrix G は、rowsG と colsG キーワードと、cooG、csrG または cscG キーワードのいずれかによって定義されます。

Sparse matrix G は、２つの疎行列入力を取る MatrixSparse コマンド（ADD と SMSM）でのみ使われます。
これらのコマンドでは、入力疎行列が CSR フォーマットであることが必要です。

Matrix B は matrixB キーワードで定義され、１つ以上の密行列入力を取る MatrixSparse コマンド（現時点では MM コマンドのみ）で使われます。

Matrix C は matrixC キーワードで定義され、２つの密行列入力を取る MatrixSparse コマンド（現時点では MM コマンドのみ）で使われます。

Vector X は vectorX キーワードで定義され、１つ以上のベクトル入力を取る MatrixSparse コマンド（現時点では MM と TRSV コマンド）で使われます。

Vector Y は vector キーワードで定義され、 2 つのベクトル入力を取る MatrixSparse コマンド（現時点では MM コマンドのみ）で使われます。

Alpha と ***Alphai*** は、alpha と alphai キーワードで定義され、 1 つ以上のスカラー入力を取るすべての MatrixSparse コマンド（現時点では MM、MV、TRSV コマンド）で使われます。
Alphai は、複素数データに対する演算時にのみ使われます。

Beta と ***Betai*** は、beta と betai キーワードで定義され、 2 つのスカラー入力を取る MatrixSparse コマンド（現時点では MM と MV コマンド）で使われます。
Betai は、複素数データに対する演算時にのみ使われます。

MatrixSparse コマンドのデータ形式

コマンドのデータ形式は、MatrixSparse コマンドに使われるすべてのデータウェーブに必要なデータ形式です。
ここで「データウェーブ」とは、疎行列の表現における値ウェーブと、密行列を表す行列ウェーブを指します。

特定の MatrixSparse コマンドが sparse matrix A を入力として受け取る場合、ウェーブ（cooA、cscA、または csrA キーワードで指定される最初のウェーブ）が演算のデータ形式を決定します。

複数の入力データウェーブがある場合（例：2 つの疎行列を加算する ADD コマンド）、すべての入力データウェーブのデータ形式は同一でなければなりません。

コマンドのデータ形式は単精度または倍精度の浮動小数点型でなければならず、実数または複素数であることができます。
MatrixSparse は NaN または INF を含むウェーブをサポートしません。

出力ウェーブはコマンドのデータ形式を使って生成されます。

MatrixSparse の数学コマンド（ADD、MV、MM、SMSM、TRSV）では、入力疎行列が CSR フォーマットである必要があります。
疎行列を返す数学コマンド（ADD、SMSM）は、CSR フォーマットの出力疎行列を作成します。

変換コマンド（TOCOO、TOCSC、TOCSR、TODENSE）は、COO、CSC、CSR、または密行列形式での入力を受け付けます。

MatrixSparse インデックスのデータ形式

行または列のインデックス、あるいは値ウェーブへのインデックスを含むインデックスウェーブ（つまりポインターウェーブ）は、符号付き 64 ビット整数ウェーブである必要があり、通常は Make/L を使って作成されます。

MatrixSparse 変換

入力疎行列の変換済みバージョンに対して MatrixSparse を動作させるよう指定することも可能です。
利用可能な変換は、転置を表す T、エルミート変換を表す H、変換なし（デフォルト）を表す N で名付けられています。

opA キーワードは、MatrixSparse が Sparse matrix A の変換済みバージョンに対して操作を行うよう指示します。
例えば次のコマンド：

```
MatrixSparse rowsA=4, colsA=4, csrA={values,columns,ptrB}, opA=T, vectorX=vector, operation=MV
```

は、Sparse matrix A の転置済みバージョンに対して動作します。

opG キーワードは、MatrixSparse が Sparse matrix G の変換済みバージョンに対して操作を行うよう指示します。

オプションの疎行列に関する情報

MatrixSparse の sparseMatrixType キーワードを使うと、疎行列入力の特徴を表すオプション情報を指定できます。
疎行列入力の特性がわかっている場合は、sparseMatrixType を使ってこの情報を MatrixSparse に渡すことができます。
これにより、パフォーマンスが向上する場合があります。

sparseMatrixType キーワードの構文は次のとおりです：

```
sparseMatrixType={smType,smMode,smDiag}
```

すべてのパラメーターはキーワードです。

smType: GENERAL, SYMMETRIC, HERMITIAN, TRIANGULAR, DIAGONAL, BLOCK_TRIANGULAR, BLOCK_DIAGONAL

smMode: LOWER, UPPER

smDiag: DIAG, NON_DIAG

MatrixSparse コマンド

このセクションでは、MatrixSparse がサポートする各コマンドについて説明します。
疎行列に関する背景資料を読み、理解していることを前提とします。

以下のセクションでは、次の略語を使います：

シンボル	**意味**	**キーワードでの指定**
smA	Sparse matrix A	rowsA, colsA, csrA (1)
smG	Sparse matrix G	rowsG, colsG, csrG (1)
dmB	Dense matrix B	matrixB
dmC	Dense matrix C	matrixC
vX	Vector X	vectorX
vY	Vector Y	vectorY
alpha	スカラー値 alpha	alpha, 複素数入力に対しては alphai
beta	スカラー値 beta	beta, 複素数入力に対しては betai
smOut	出力疎行列	N/A (2)

(1) 行列形式変換コマンド TOCOO、TOCSC、TOCSR、TODENSE では、入力疎行列を COO フォーマットまたは CSC フォーマットで指定するために cooA および cscA キーワードも使用できます。

その他のすべてのコマンドでは、入力行列を CSR フォーマットで指定するために csrA および csrG を使う必要があります。

(2) 出力疎行列 smOut は、CSR フォーマットでウェーブ W_CSRValues、W_CSRColumns、W_CSRPointerB によって表現されます。

MatrixSparse ADD

ADD は Sparse matrix A と Sparse matrix G の和を計算します。
G は CSR フォーマットでなければなりません。
記号的には：

```
smOut = smA + smG
```

入力：CSR フォーマットの Sparse matrix A と Sparse matrix G

出力：CSR フォーマットの疎行列（W_CSRValues、W_CSRColumns、W_CSRPointerB で表現）

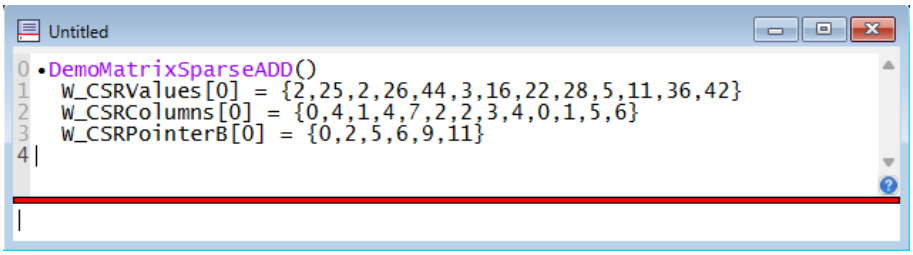
MatrixSparse ADD の例

```
Function DemoMatrixSparseADD()
        // CSR フォーマットで smA を作成
        Make/FREE/D/N=(11) valuesA = {1,25,26,44,16,22,28,5,11,36,42} // 倍精度浮動小数点
        Make/FREE/L/N=(11) columnsA = {0,4,4,7,2,3,4,0,1,5,6}         // 64-bit 符号付き整数
        Make/FREE/L/N=(6) ptrBA = {0,2,4,4,7,9}                       // 64-bit 符号付き整数

        // CSR フォーマットで smG を作成
        Make/FREE/D/N=(3) valuesG ={1,2,3}                            // 倍精度浮動小数点
        Make/FREE/L/N=(3) columnsG ={0,1,2}                           // 64-bit 符号付き整数
        Make/FREE/L/N=(4) ptrBG = {0,1,2,3,3,3,3,3,3}                 // 64-bit 符号付き整数

        // smA + smG を計算
        MatrixSparse rowsA=6, colsA=8, csrA={valuesA,columnsA,ptrBA}, rowsG=6, colsG=8,
csrG={valuesG,columnsG,ptrBG}, operation=ADD
        WAVE W_CSRValues, W_CSRColumns, W_CSRPointerB  // MatrixSparse コマンド ADD からの出力

        // 出力 CSR 疎行列を表す 1D ウェーブを出力
        Print W_CSRValues
        Print W_CSRColumns
        Print W_CSRPointerB
End
```



MatrixSparse MM

MM は疎行列と密行列の積を計算します。
記号的には：

```
M_MMOut = alpha*smA*dmB + beta*dmC
```

入力：alpha、CSR フォーマットの Sparse matrix A、Dense matrix B、オプションで beta と Dense matrix C
beta キーワードを省略して beta をデフォルト値 0 のままにすると、beta*dmC 項は計算されないため、dmC 入力を指定する必要はありません。

出力：密行列 M_MMOut

MatrixSparse MM の例

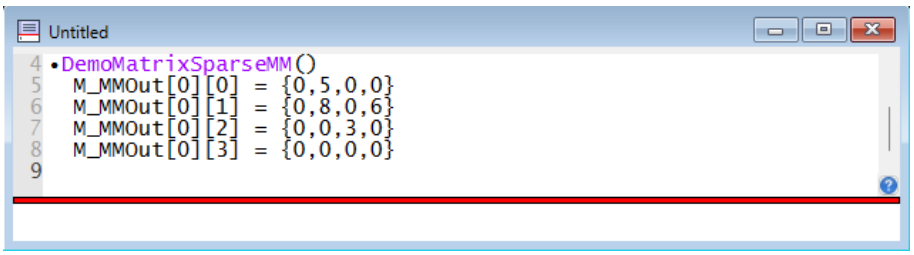
```
Function DemoMatrixSparseMM()
	// CSR フォーマットで疎行列を定義
	Make/FREE/D values = {5, 8, 3, 6}		// 倍精度浮動小数点
	Make/FREE/L columns = {0, 1, 2, 1}		// 64-bit 符号付き整数
	Make/FREE/L ptrB = {0, 0, 2, 3, 4}		// 64-bit 符号付き整数

	// 密行列を作成
	Make/FREE/D matrix = { {1,0,0,0}, {0,1,0,0}, {0,0,1,0}, {0,0,0,1} }	// 倍精度浮動小数点

	// 疎行列を密行列で乗算
	MatrixSparse rowsA=4, colsA=4, csrA={values,columns,ptrB}, matrixB=matrix, operation=MM

	// 出力密行列に対するウェーブ参照を作成
	WAVE M_MMOut 				// MatrixSparse コマンド MM からの出力

	Print M_MMOut
End
```



MatrixSparse MV

CSR フォーマットでなければならない疎行列とベクトルの積を計算し、出力ベクトルを生成します。
記号的には：

```
W_MV = alpha*smA*vX + beta*vY
```

入力：alpha、CSR フォーマットの Sparse matrix A、Vector X、オプションで beta と Vector Y
beta キーワードを省略して beta をデフォルト値 0 のままにすると、beta*vY 項は計算されないため、vY 入力を指定する必要はありません。

出力：ベクトル W_MV

MatrixSparse MV の例

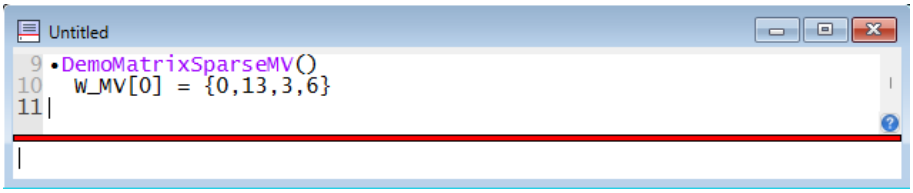
```
Function DemoMatrixSparseMV()
        // CSR フォーマットで疎行列を定義
        Make/FREE/D values = {5, 8, 3, 6}                    // 倍精度浮動小数点
        Make/FREE/L columns = {0, 1, 2, 1}                   // 64-bit 符号付き整数
        Make/FREE/L ptrB = {0, 0, 2, 3, 4}                   // 64-bit 符号付き整数

        // ベクトルを作成
        Make/FREE/D vector = {1, 1, 1, 1}                    // 倍精度浮動小数点

        // 疎行列をベクトルで乗算
        MatrixSparse rowsA=4, colsA=4, csrA={values,columns,ptrB}, vectorX=vector, operation=MV

        // 出力ベクトルに対するウェーブ参照を作成
        WAVE W_MV                                   // MatrixSparse コマンド MV からの出力

        Print W_MV
End
```



MatrixSparse SMSM

SMSM は 2 つの疎行列の積を計算します。
記号的には：

```
        smOut = smA * smG
```

入力：CSR フォーマットの Sparse matrix A、CSR フォーマットの Sparse matrix G

出力：W_CSRValues、W_CSRColumns、W_CSRPointerB で表される CSR フォーマットの疎行列

MatrixSparse SMSM の例

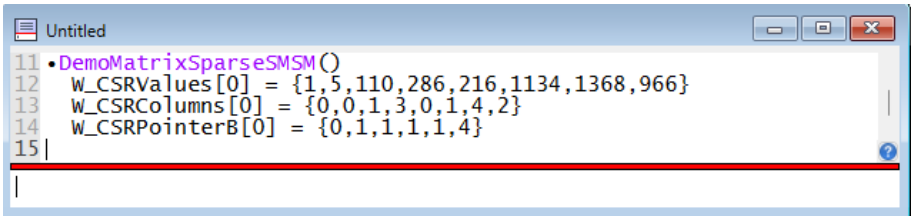
```
Function DemoMatrixSparseSMSM()
        // CSR フォーマットで sparse matrix A を作成
        Make/FREE/D/N=(11) valuesA = {1,25,26,44,16,22,28,5,11,36,42} // 倍精度浮動小数点
        Make/FREE/L/N=(11) columnsA = {0,4,4,7,2,3,4,0,1,5,6}         // 64-bit 符号付き整数
        Make/FREE/L/N=(6) ptrBA = {0,2,4,4,7,9}                       // 64-bit 符号付き整数

        // CSR フォーマットで Sparse matrix G を作成
        Make/FREE/D/N=(8) valuesG = {1,10,26,6,14,38,15,23}           // 倍精度浮動小数点
        Make/FREE/L/N=(8) columnsG = {0,1,3,0,1,4,1,2}                // 64-bit 符号付き整数
        Make/FREE/L/N=(8) ptrBG = {0,1,3,3,3,3,6,8}                   // 64-bit 符号付き整数

        // MatrixA x MatrixG を計算
        MatrixSparse rowsA=6, colsA=8, csrA={valuesA,columnsA,ptrBA}, rowsG=8, colsG=5,
csrG={valuesG,columnsG,ptrBG}, operation=SMSM
        WAVE W_CSRValues, W_CSRColumns, W_CSRPointerB  // MatrixSparse コマンド SMSM からの出力

        // 出力 CSR 疎行列を表す 1D ウェーブを出力
        Print W_CSRValues
```

```
        Print W_CSRColumns
        Print W_CSRPointerB
End
```



MatrixSparse TOCOO

TOCOO は、入力行列（密行列、CSC フォーマット、または CSR フォーマットのいずれか）に対応する COO フォーマットの疎な出力行列を生成します。

入力：matrixB キーワードで指定される密行列、または cscA または csrA キーワードで指定される疎行列

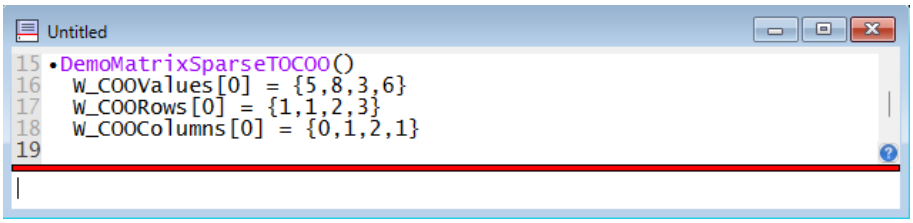
出力：W_COOValues、W_COORows、W_COOColumns で表される COO フォーマットの疎行列

MatrixSparse TOCOO の例

```
Function DemoMatrixSparseTOCOO()
        // CSR フォーマットで Wikipedia の例の 4x4 行列を作成
        Make/FREE/D values = {5, 8, 3, 6}              // 倍精度浮動小数点
        Make/FREE/L columns = {0, 1, 2, 1}             // 64-bit 符号付き整数
        Make/FREE/L ptrB = {0, 0, 2, 3, 4}             // 64-bit 符号付き整数

        // CSR 行列から COO フォーマットの疎行列を作成
        MatrixSparse rowsA=4, colsA=4, csrA={values,columns,ptrB}, operation=TOCOO
        WAVE W_COOValues, W_COORows, W_COOColumns      // MatrixSparse コマンド TOCOO からの出力

        // COO 疎行列を表す 1D ウェーブを出力
        Print W_COOValues
        Print W_COORows
        Print W_COOColumns
End
```



MatrixSparse TOCSC

TOCSC は、入力行列（密行列、COO フォーマット、または CSR フォーマットのいずれか）に相当する CSC フォーマットの疎な出力行列を生成します。

入力：matrixB キーワードで指定される密行列、または cooA または csrA キーワードで指定される疎行列

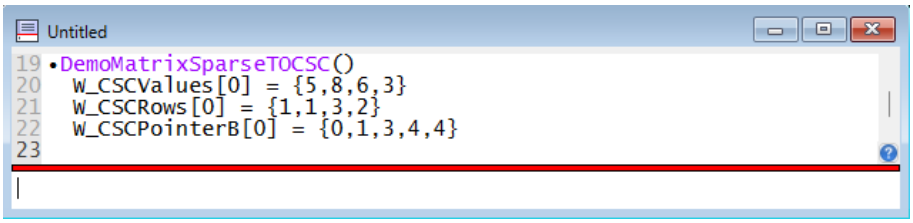
出力：W_CSCValues、W_CSCRows、W_CSCPointerB で表される CSC フォーマットの疎行列

MatrixSparse TOCSC の例

```
Function DemoMatrixSparseTOCSC()
        // 密なフォーマットで Wikipedia の例の 4x4 行列を作成
        Make/FREE/D/N=(4,4) dense                    // 倍精度浮動小数点
        dense[0][0] = {0,5,0,0}
        dense[0][1] = {0,8,0,6}
        dense[0][2] = {0,0,3,0}
        dense[0][3] = {0,0,0,0}

        // 密行列から CSC フォーマットの疎行列を作成
        // MatrixSparse は、このケースでは重要ではないにもかかわらず、rowsA と colsA を要求する
        MatrixSparse rowsA=4, colsA=4, matrixB=dense, operation=TOCSC
        WAVE W_CSCValues, W_CSCRows, W_CSCPointerB     // MatrixSparse コマンド TOCSC からの出力

        // CSC 疎行列を表す 1D ウェーブを出力
        Print W_CSCValues
        Print W_CSCRows
        Print W_CSCPointerB
End
```



MatrixSparse TOCSR

TOCSR は、入力行列（密行列、COO フォーマット、または CSC フォーマットのいずれか）に相当する CSR フォーマットの疎な出力行列を生成します。

入力：matrixB キーワードで指定される密行列、または cooA または cscA キーワードで指定される疎行列

出力：W_CSRValues、W_CSRColumns、W_CSRPointerB で表される CSR フォーマットの疎行列

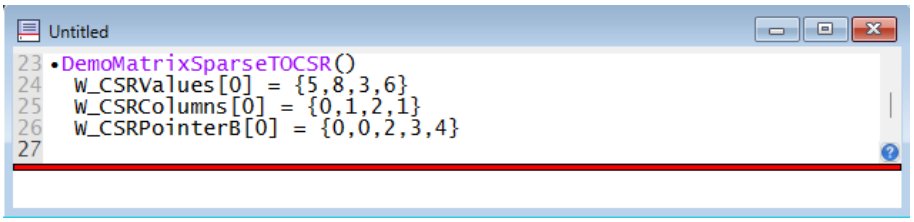
MatrixSparse TOCSR の例

```
Function DemoMatrixSparseTOCSR()
        // 密なフォーマットで Wikipedia の例の 4x4 行列を作成
        Make/FREE/D/N=(4,4) dense                    // 倍精度浮動小数点
        dense[0][0] = {0,5,0,0}
        dense[0][1] = {0,8,0,6}
        dense[0][2] = {0,0,3,0}
        dense[0][3] = {0,0,0,0}

        // 密行列から CSC フォーマットの疎行列を作成
        // MatrixSparse は、このケースでは重要ではないにもかかわらず、rowsA と colsA を要求する
        MatrixSparse rowsA=4, colsA=4, matrixB=dense, operation=TOCSR
        WAVE W_CSRValues, W_CSRColumns, W_CSRPointerB  // MatrixSparse コマンド TOCSR からの出力

        // CSR 疎行列を表す 1D ウェーブを出力
        Print W_CSRValues
```

```
        Print W_CSRColumns
        Print W_CSRPointerB
End
```



MatrixSparse TODENSE

TODENSE は、COO、CSC、または CSR フォーマットの疎入力行列に相当する密出力行列を生成します。

入力：cooA、cscA、または csrA キーワードで指定される疎行列

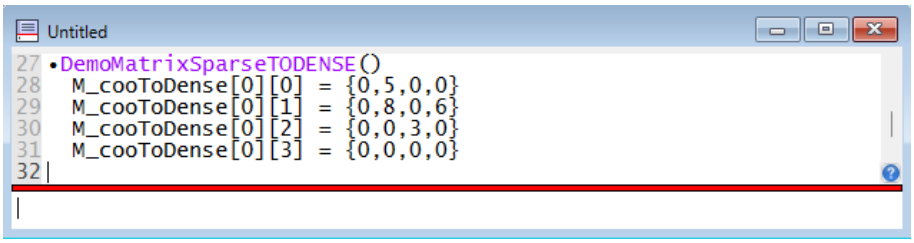
出力：密行列 M_cooToDense、M_cscToDense、または M_csrToDense

MatrixSparse TODENSE の例

```
Function DemoMatrixSparseTODENSE()
        // COO フォーマットで Wikipedia の例の 4x4 行列を作成
        Make/FREE/D values = {5, 8, 3, 6}              // 倍精度浮動小数点
        Make/FREE/L rows = {1, 1, 2, 3}                // 64-bit 符号付き整数
        Make/FREE/L columns = {0, 1, 2, 1}             // 64-bit 符号付き整数

        // 疎行列から密行列を作成
        MatrixSparse rowsA=4, colsA=4, cooA={values,rows,columns}, operation=TODENSE
        WAVE M_cooToDense                       // MatrixSparse コマンド TODENSE からの出力

        // 密な出力行列を出力
        Print M_cooToDense
End
```



MatrixSparse TRSV

TRSV は三角疎行列 A の連立一次方程式を解きます。
記号的には出力行列 M_TRSVOut を求めます。
ここでは：

```
        smA*M_TRSVOut = alpha*vX
```

入力：CSR フォーマットの Sparse matrix A、alpha、vector X

出力：密行列 M_TRSVOut

MatrixSparse TRSV の例

注記：2026 年 01 月時点のヘルプ内のコードでは {0, 0, 0} という結果となるため、一部編集が必要です。

```
// これは、Wikipedia ページの Row Reduction セクションにある例に基づいています。
//      https://en.wikipedia.org/wiki/System_of_linear_equations
// 連立方程式は次の通りです：
//              x + 3y - 2z = 5
//              3x + 5y + 6z = 7
//              2x + 47 + 3z = 8
// これにより、以下の拡張行列が得られます：
//              1       3       -2      5
//              3       5       6       7
//              2       4       3       8
// 解は： x=-15, y=8, z=2
// MatrixSparse の TRSV コマンドは三角化係数行列を必要とするため、
// Gauss-Jordan 消去法を使って得られた拡張行列の上三角版から開始します：
//              1       3       -2      5
//              0       1       -3      2
//              0       0       1       2
// MatrixSparse TOCSR を使って同等の疎行列を作成します。
// 次に、対応する解ベクトル {5, 5, 2} を作成します。
// 次に、MatrixSparse TRSV を呼び出し、解の集合{-15, 8, 2}を得ます。

Function DemoMatrixSparseTRSV()
        // 係数を表す密な上三角行列を作成
        Make/FREE/D/N=(3,3) utMat               // 倍精度浮動小数点
        utMat[0][0] = {1, 0, 0}                 // 列 0
        utMat[0][1] = {3, 1, 0}                 // 列 1
        utMat[0][2] = {-2, -3, 1}               // 列 2

        // CSR フォーマットで上三角行列の疎行列バージョンを作成
        MatrixSparse rowsA=3, colsA=3, matrixB=utMat, operation=TOCSR
        WAVE values = W_CSRValues
        WAVE columns = W_CSRColumns
        WAVE ptrB = W_CSRPointerB

        // 解ベクトルを作成
        Make/FREE/D/N=(3) vector = {5,2,2}      // 倍精度浮動小数点

        // 連立一次方程式を解く
        // sparseMatrixType={TRIANGULAR,UPPER,NON_DIAG}を追加すると、パフォーマンスが向上する可能性がある
        // この行を修正：MatrixSparse rowsA=3, colsA=3, csrA={values,columns,ptrB}, vectorX=vector,
        operation=TRSV
        MatrixSparse rowsA=3, colsA=3, csrA={values,columns,ptrB},
                sparseMatrixType={TRIANGULAR,UPPER,NON_DIAG}, vectorX=vector, operation=TRSV
        WAVE M_TRSVOut                          // MatrixSparse コマンド TRSV からの出力

        Print M_TRSVOut                         // {-15, 8, 2} となるはず
End
```