連立方程式の解法

連立方程式をエクセルを用いて解く方法は以下の2種類が考えられます。

1） エクセルの行列関数を用いる。

2） VBA でヤコビ法やガウスザイデル法を用いる。

ここでは両方について説明します。

1）エクセルの行列関数を用いる方法

エクセルは表計算ですから行と列に並んだ数値を扱うのは得意です。連立方程式は次のように行列を用いて表すことができます。

$$
\begin{aligned}
a_{11}x + a_{12}y + a_{13}z &= b_1 \\
a_{21}x + a_{22}y + a_{23}z &= b_2 \\
a_{31}x + a_{32}y + a_{33}z &= b_3
\end{aligned}
$$

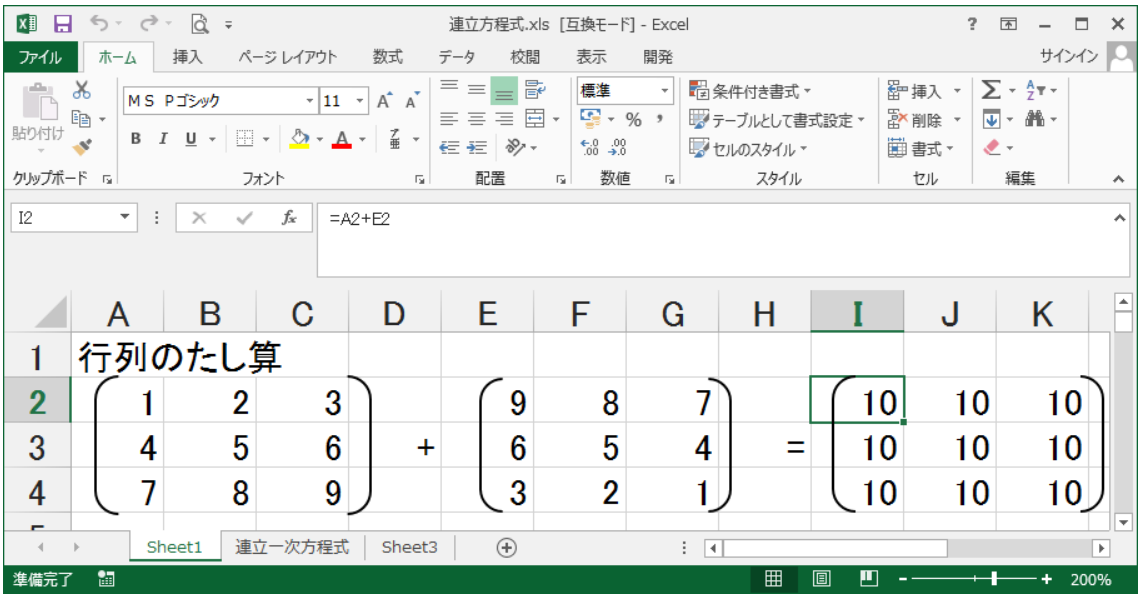
⇩

$$
\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}
\begin{pmatrix} x \\ y \\ z \end{pmatrix} =
\begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix}
$$

連立方程式が行列形式で表されることを考慮して解法を考えてみます。直接連立方程式を解く前に行列の計算をどの様にエクセルで取り扱うか考えてみます。

a)行列の加算

行列の加算は行列の各成分同士を足しあわせることにより計算できます。

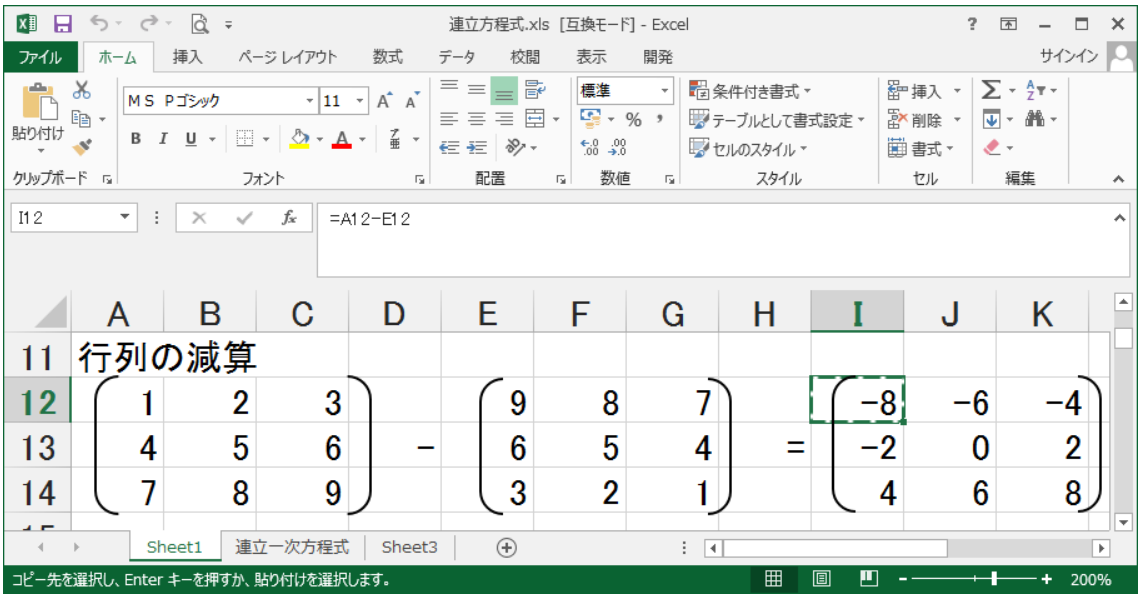


加える2つの行列を A2:C4 と E2:G4 に入力します。加えた結果を I2:K4 に代入するとします。I2 に=A2+E2 と入力し、このセルを残りのセルにコピー＆ペーストすると残りの成分も自動的に計算されます。

b)減算

行列の減算も加算と同様に行うことができます。

減算を行う2つの行列を A12:C14 と E12:G14 に入力します。結果を I12:K14 に代入するとします。I12 に=A12-E12 と入力し、このセルを残りのセルにコピー＆ペーストすると残りの成分も自動的に計算されます。

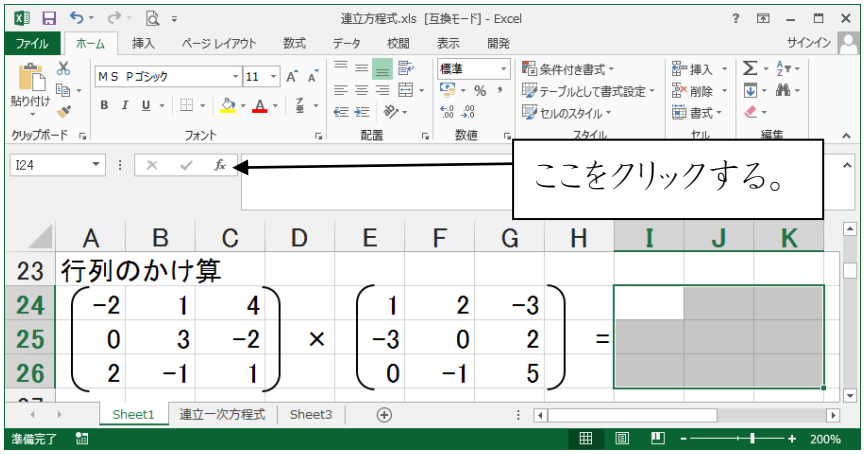


c)乗算

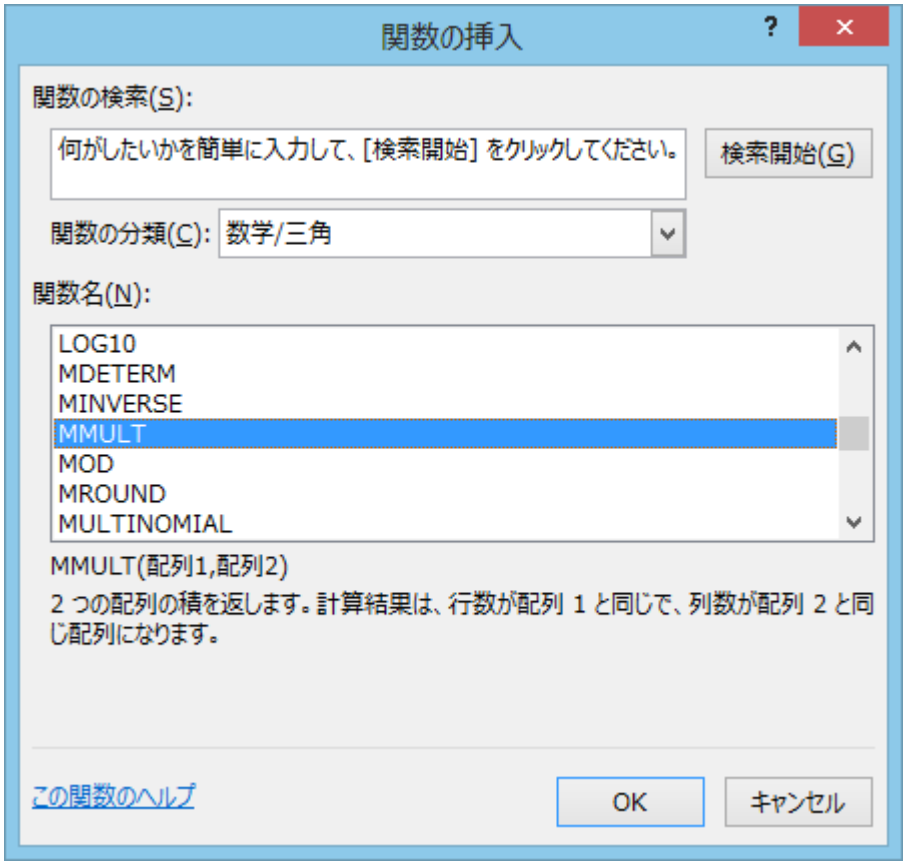
行列の乗算を加算や減算と同じように各セルに数式を入力することで計算することも可能です。しかし、エクセルには行列を扱う関数として MMULT が用意されていますのでこれを利用してみましょう。

まず、乗算を行う2つの行列を A24:C26 と E24:G26 に用意します。そして、乗算の答えを入力する部分をドラッグして選択します。

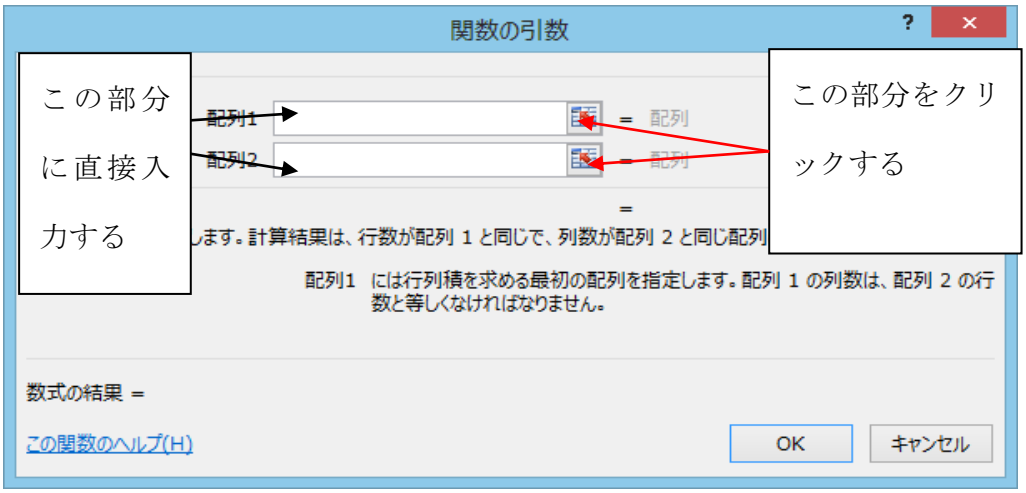
選択した状態で関数貼り付けのアイコン(*fx*)を押すか、メニューの



挿入→関数を選択します。すると関数の貼り付けのウィンドウが現れます。



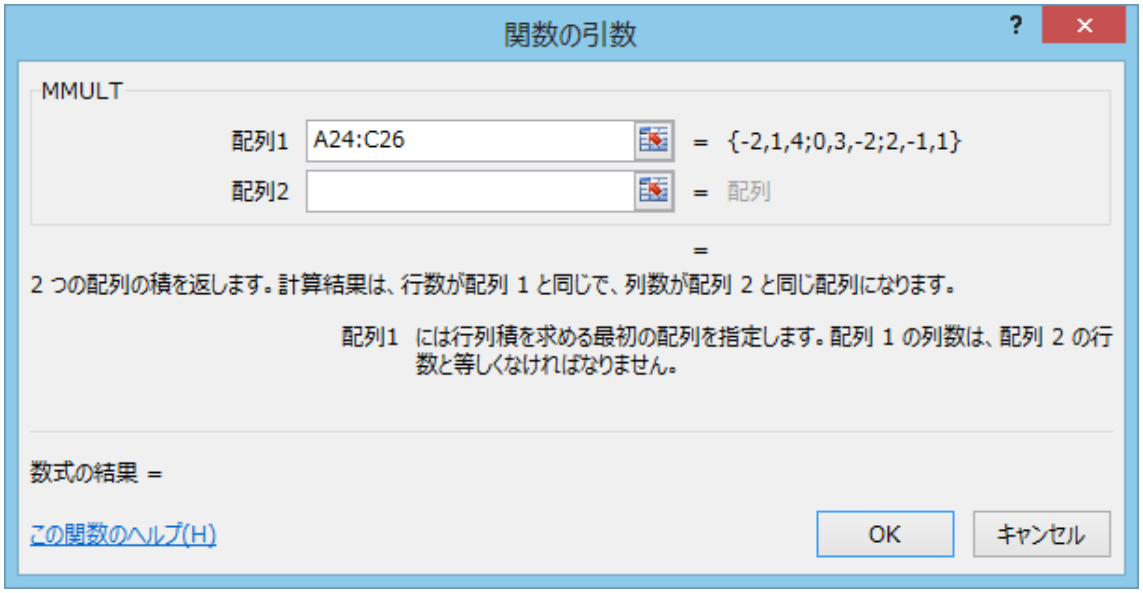
このウィンドウの数学／三角（関数の分類）、MMULT（関数名）を選択してOKを押してください。すると乗算に用いる2つの行列を指定するウィンドウが現れます。



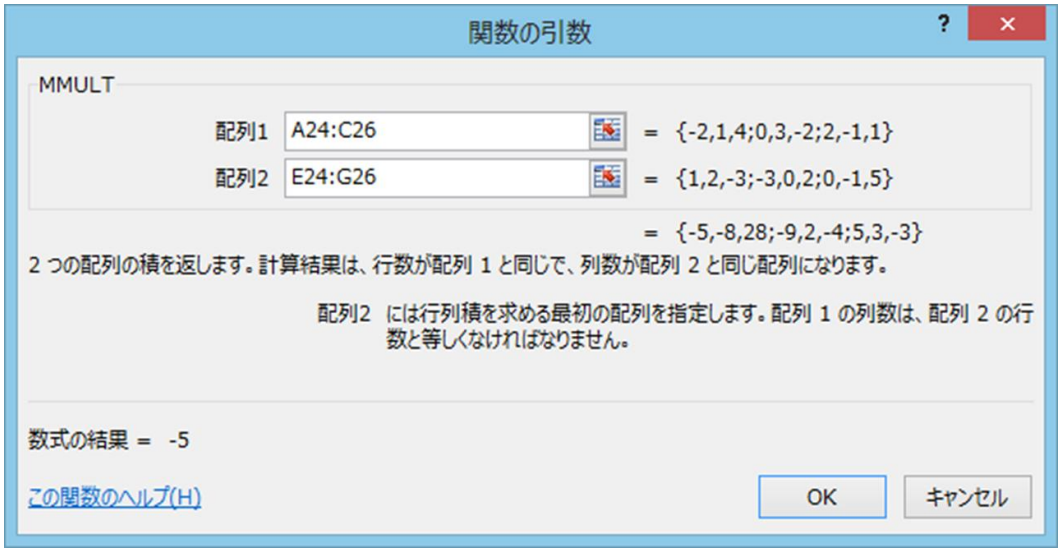
この画面で乗算に用いる2つの行列（配列）の指定を行います。指定の方法には2つの方法があります。1つは直接行列の範囲を入力する方法です。行列の左上のセルと右下のセルを使って指定します。例えば、今回の場合ではA24:C26とE24:G26という2つの行列を用意しています。このウィンドウの配列1と配列2のところに直接A24:C26とE24:G26と入力します。行列の計算は必ずしもA×B≠B×Aなので注意しましょう。

入力のもう一つの方法はマウスを使って行列を指定する方法です。まず、配

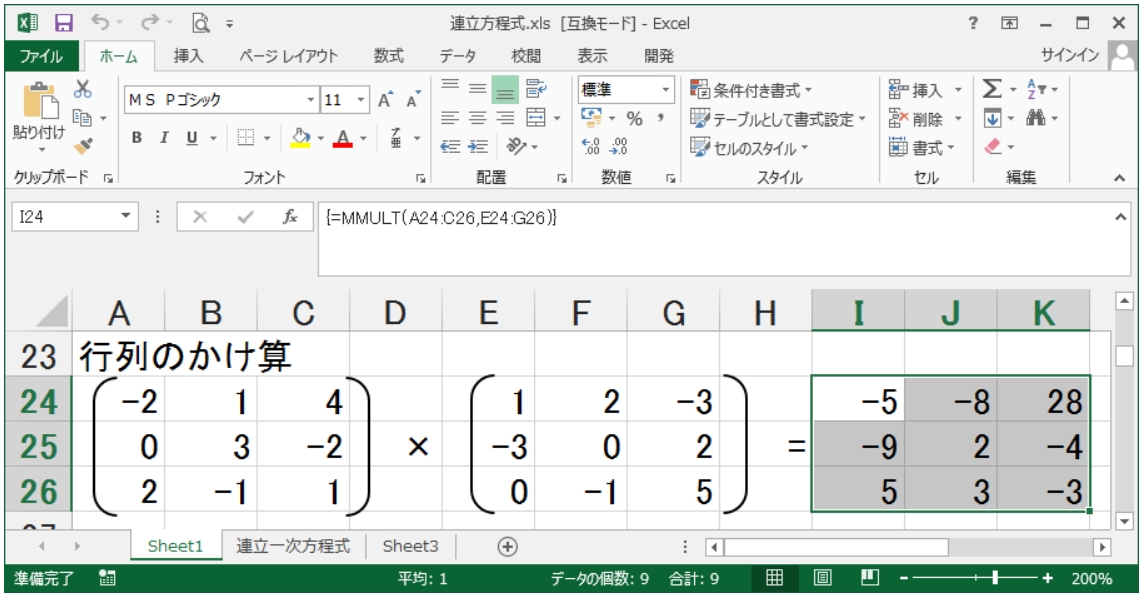
列1の入力ウィンドウの一番右端のアイコンをクリックしてください。
画面が元の表に戻りますから1番目の行列の範囲をマウスでドラッグしてください(上の図の点線で囲まれた部分)。指定し終えたらをクリックしてください。元の画面に戻ります。



これで一つめの行列(配列1)の入力が終了しました。2つめの行列(配列2)も同様に指定します。



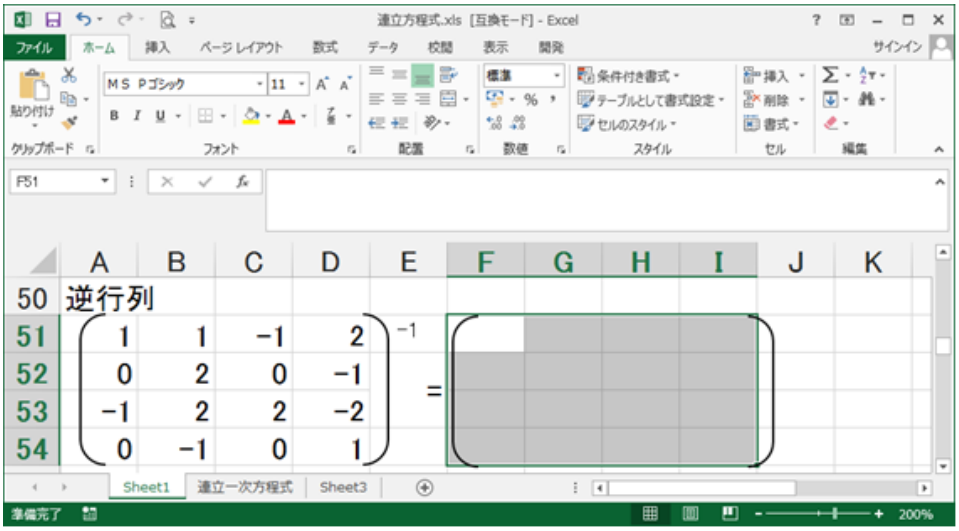
2つの行列の指定が終了したらただ OK を押すのではなく Ctrl キーと Shift キーを両方とも押した状態で OK をクリックします。すると行列の乗算の結果が示されます。

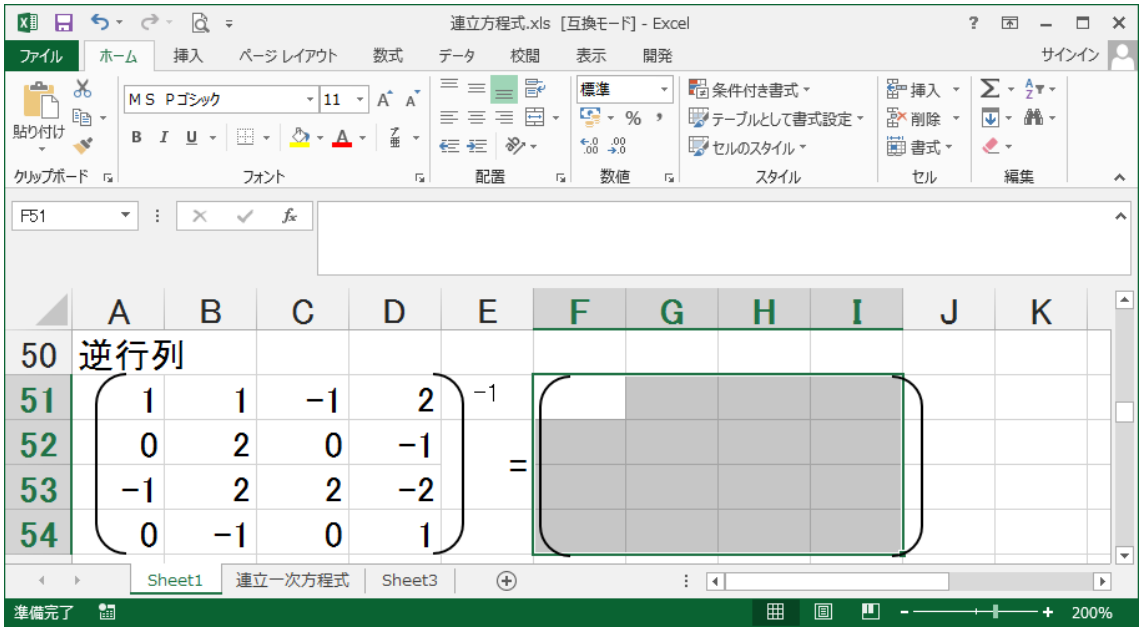


d)逆行列の計算

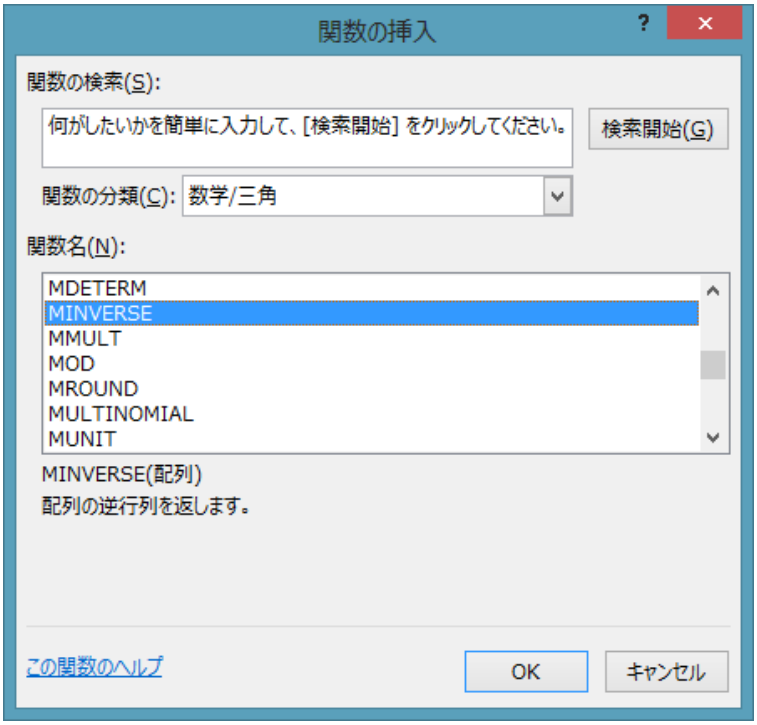
逆行列の計算も関数機能を用います。まず、逆行列を求める行列を入力します。

乗算と同様に逆行列の結果を表示する部分をドラッグして選択します。



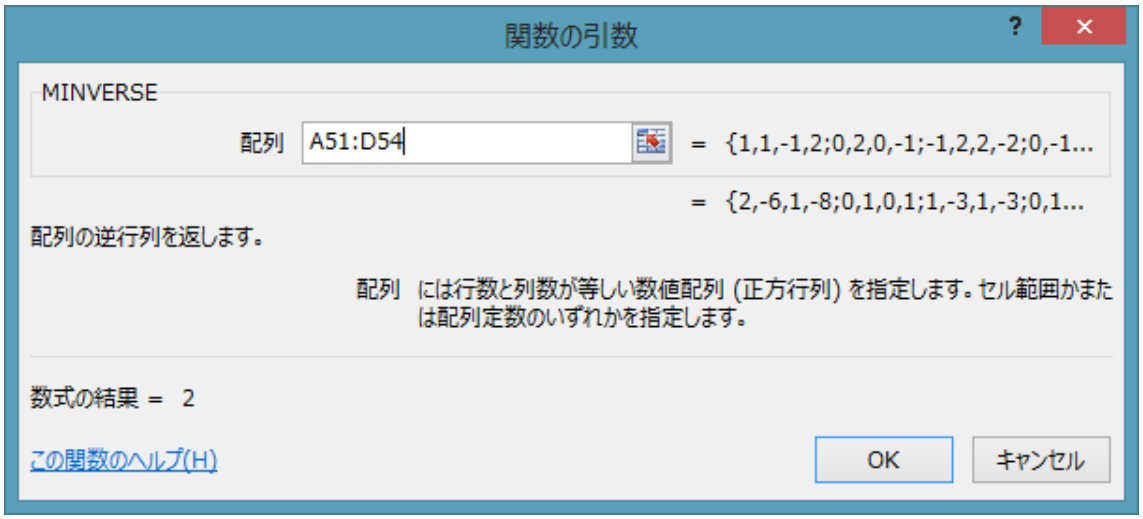


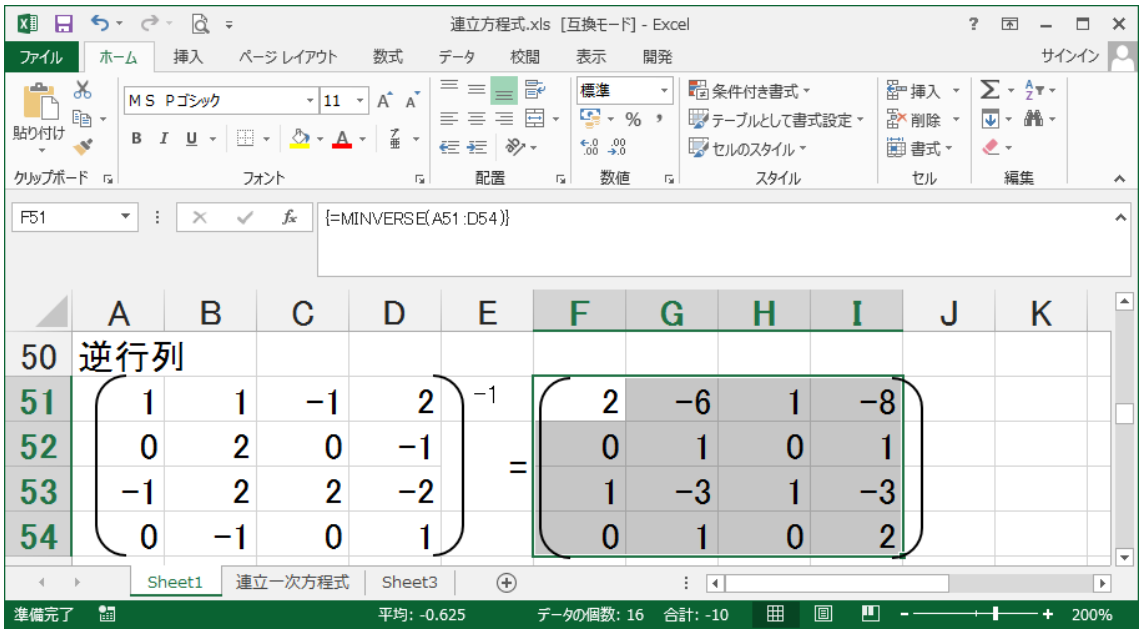
この状態で関数の貼り付けのアイコンをクリックします。乗算と同様に関数の選択画面が現れます。ここで数学／三角（関数の分類）、MINVERSE（関数名）を選択して OK をクリックします。

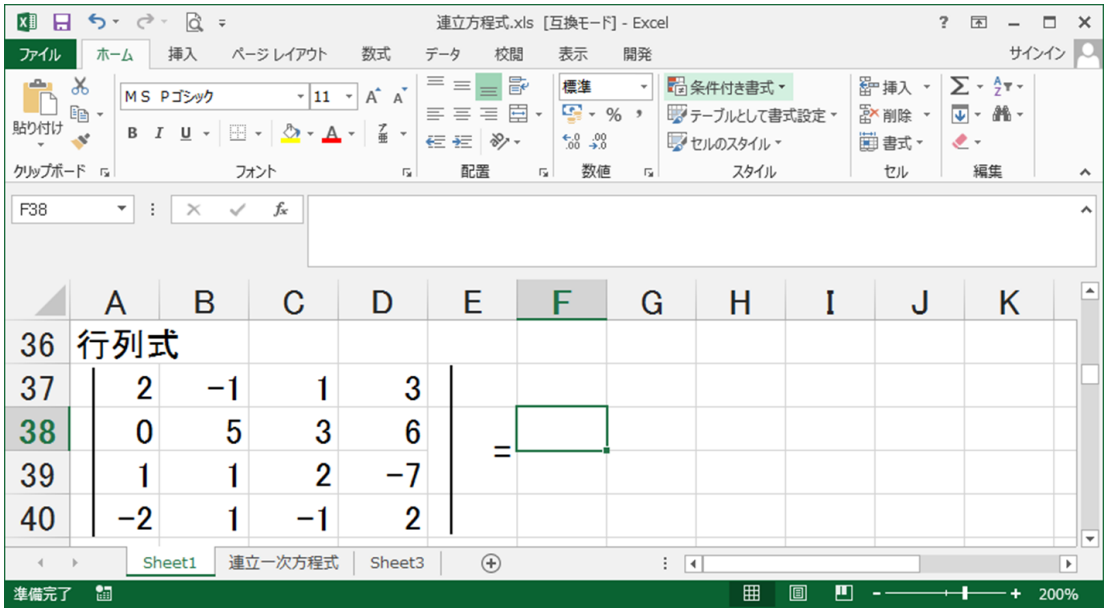


すると次に MINVERSE の条件設定の画面が現れます。逆行列を計算した

い行列を指定して Ctrl キーと Shift キーを押した状態で OK をクリックします。すると結果として表示されます。





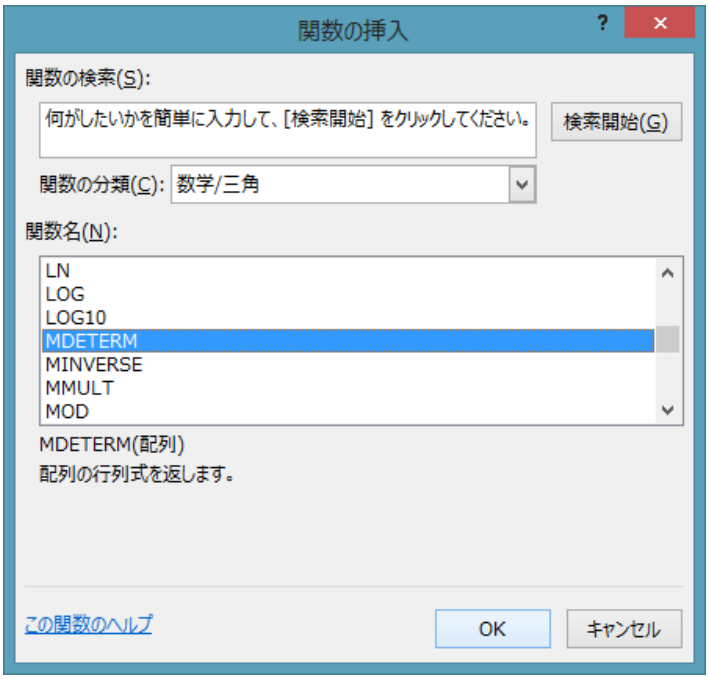


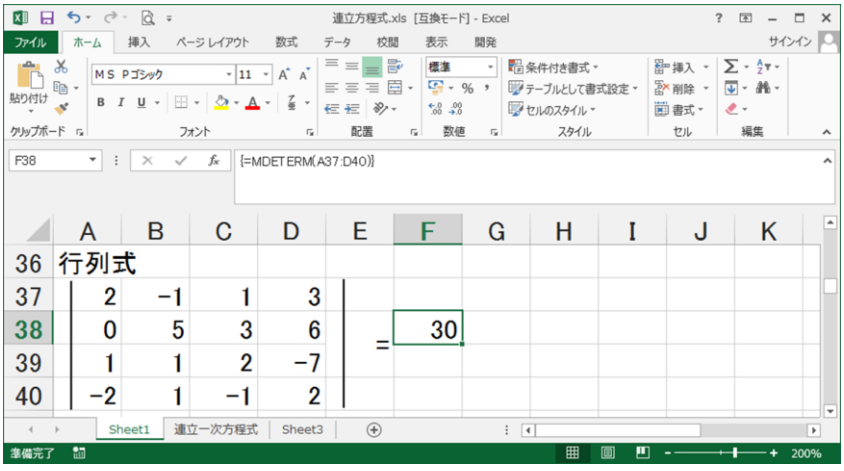
e)行列式の計算

行列式の計算も同じです。まず、計算すべき行列を入力します。

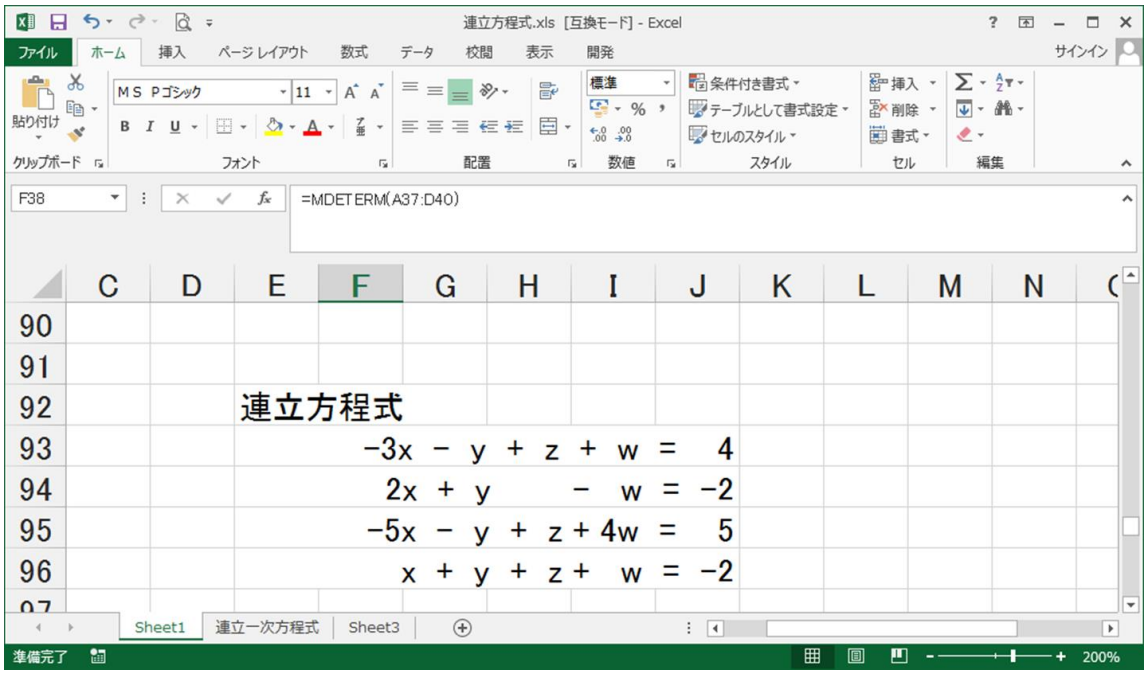
行列式の値を入力すべきセルを選択し(行列式の場合には値はスカラーになるので1つのセル)、関数の貼り付けをクリックします。そして数学／三角(関数の分類)、MDETERM(関数名)を選択します。

後は乗算や逆行列と同様に行列をします。そして、この場合は結果が数値なのでただ単にOK を押すことにより行列式が計算されます。

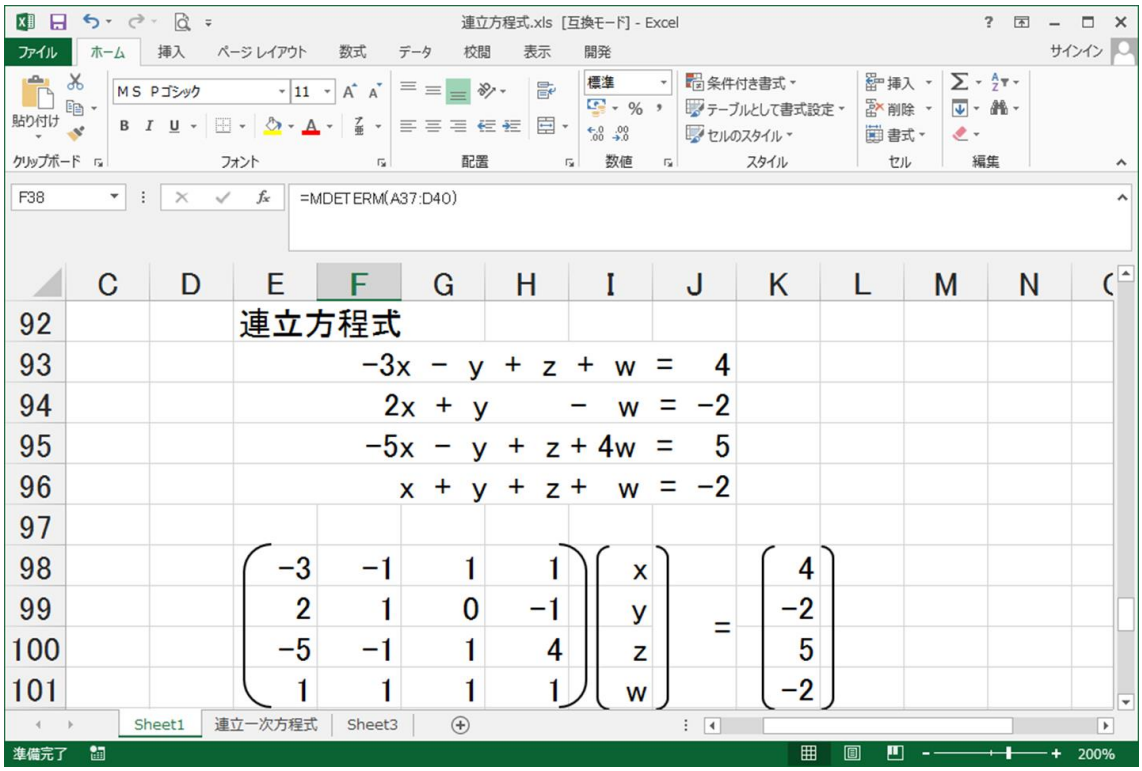




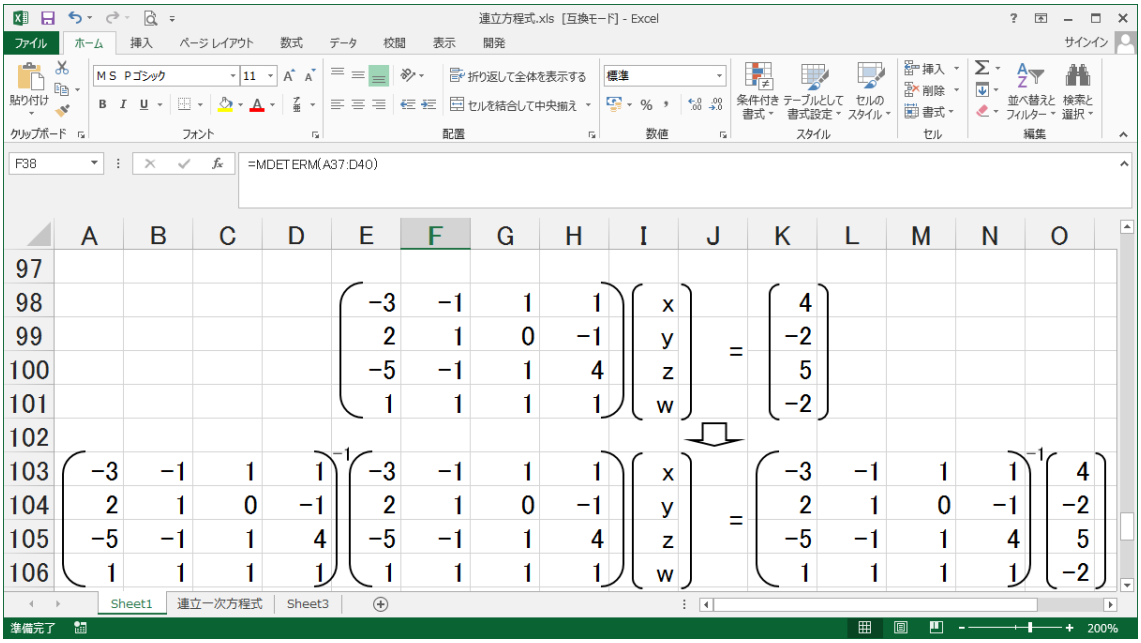
ここまでで行列の取り扱いが可能になりましたのでこれらを用いて連立方程式を解いてみます。



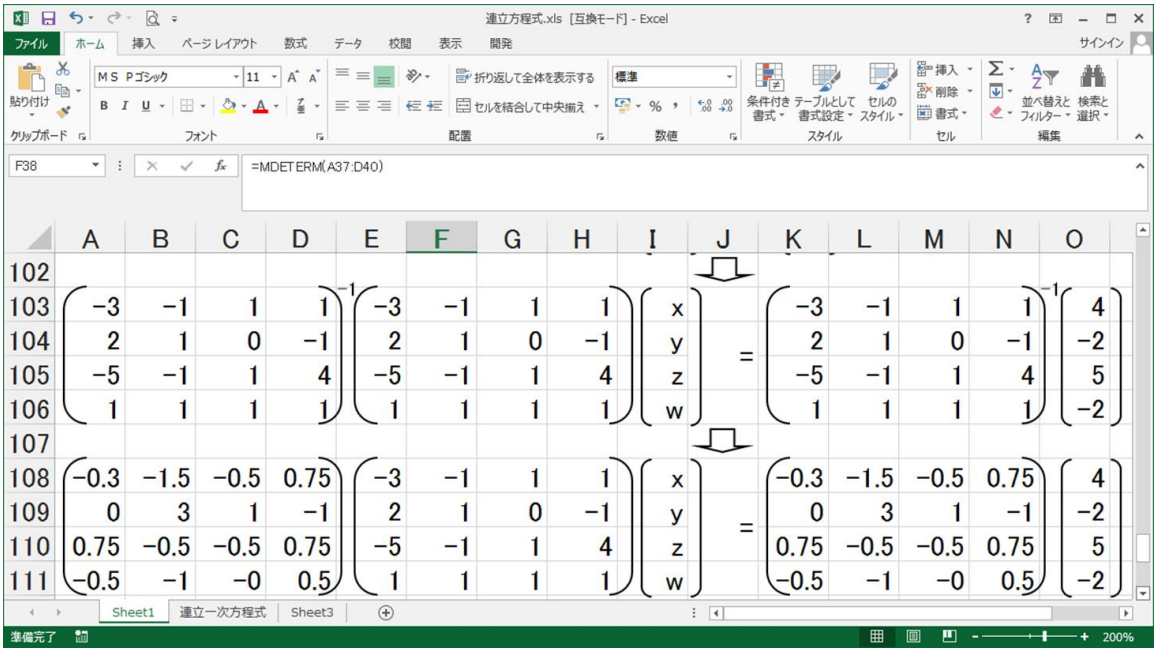
まず、連立方程式を入力します(これは必ずしも必要ではない)。
この連立方程式を行列形式で書き表します。



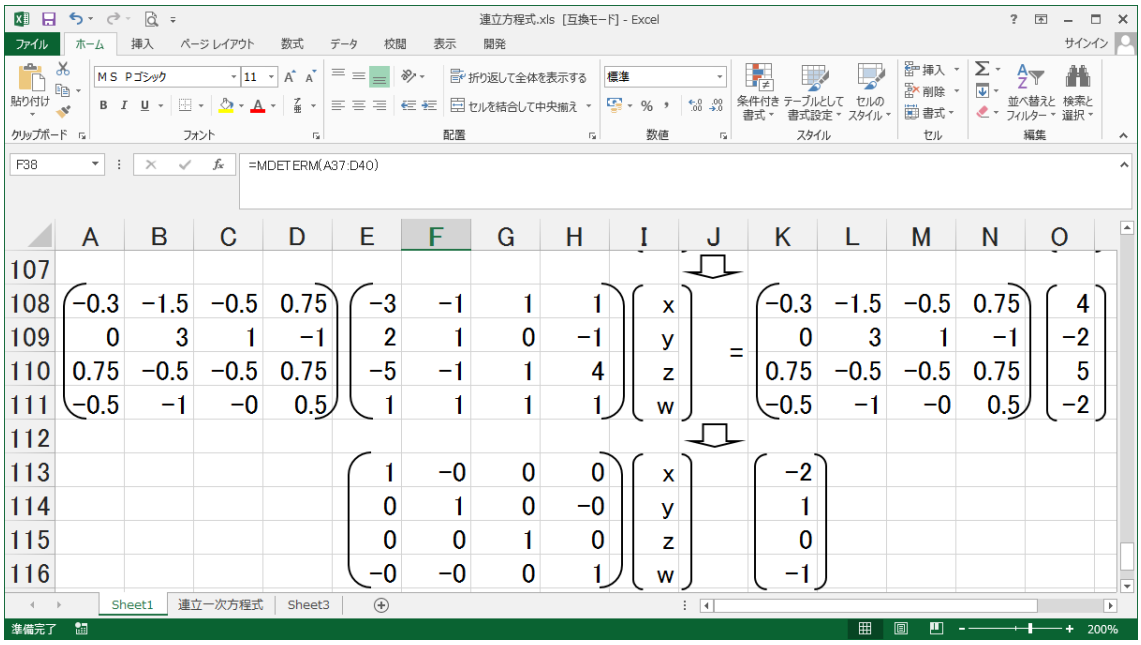
この行列形式の方程式に対して、要するに両辺から係数行列(この場合は4×4行列)の逆行列をかければ解が得られます。



この部分で逆行列の計算を行います(逆行列の計算参照)。その結果が下の図です。



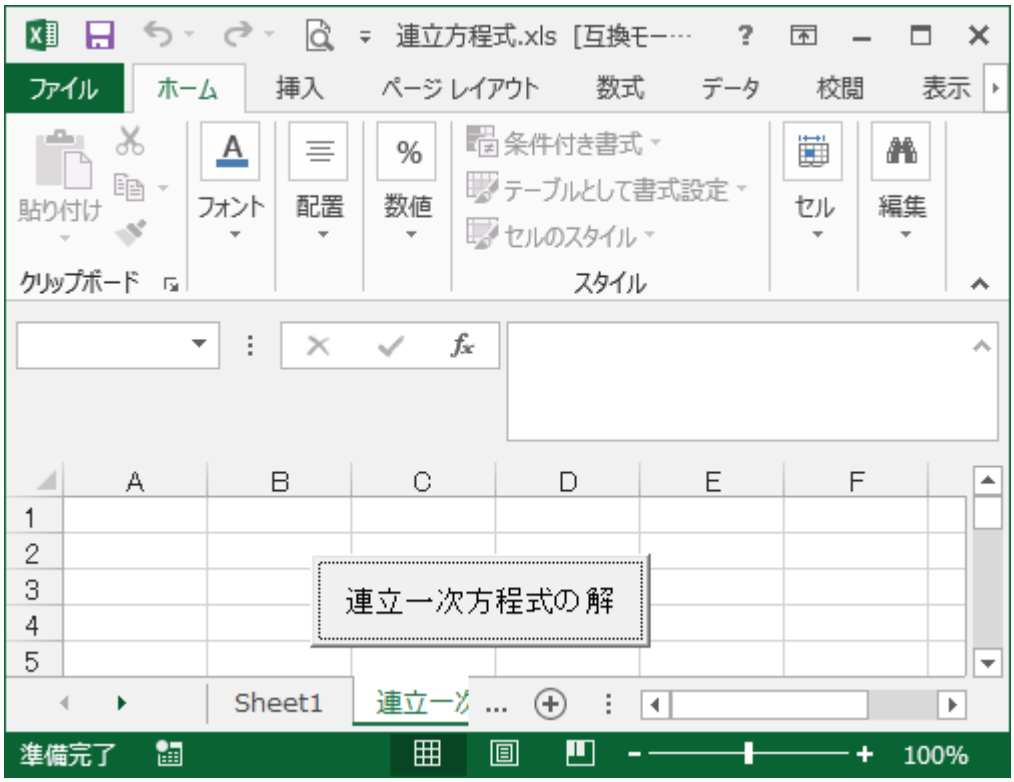
さらに、計算した逆行列と元の係数行列との計算を左辺と右辺で行います。



係数行列が対角行列で成分が1になっていることを確認してください。得られた右辺が連立方程式の解です。

2）VBA でヤコビ法やガウスザイデル法を用いる方法

ここではヤコビ法による VBA プログラムについて説明します。既に、VBA プログラムの作成については2回説明していますので細かいことについては省略します。まず、プログラムを起動するコマンドボタンを作りましょう。

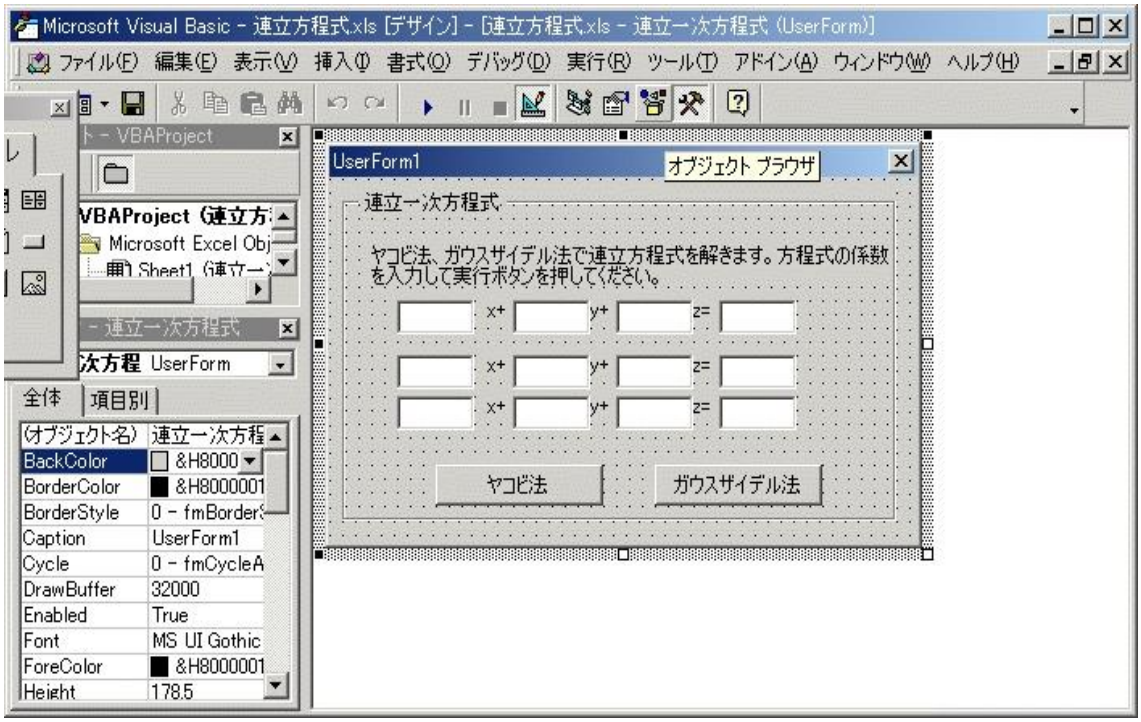


作成できたら、このコマンドボタンに下記のプログラムを入力します。

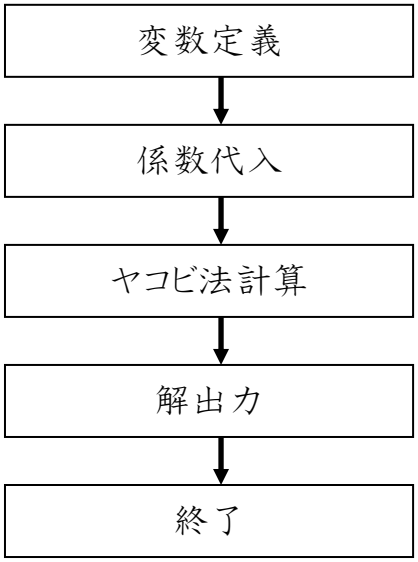
```
Private Sub CommandButton1_Click()
    連立一次方程式.Show
End Sub
```

次に、VBA の入力用のフォームを作成します。連立方程式としては三元連立方程式を想定し、その係数をすべて入力するフォームを作成します。そして、その係数を持つ方程式をヤコビ法を用いて解くこととします。下の図はその入力用のユーザーフォームです。この例ではガウスザイデル法も計算できるように設計してあります。テキストボックス、ラベル、コマンドボタンなどを用いて下図のようなユーザーフォームを作成してください。
ヤコビ法の計算は下記の式に基づいて行います。

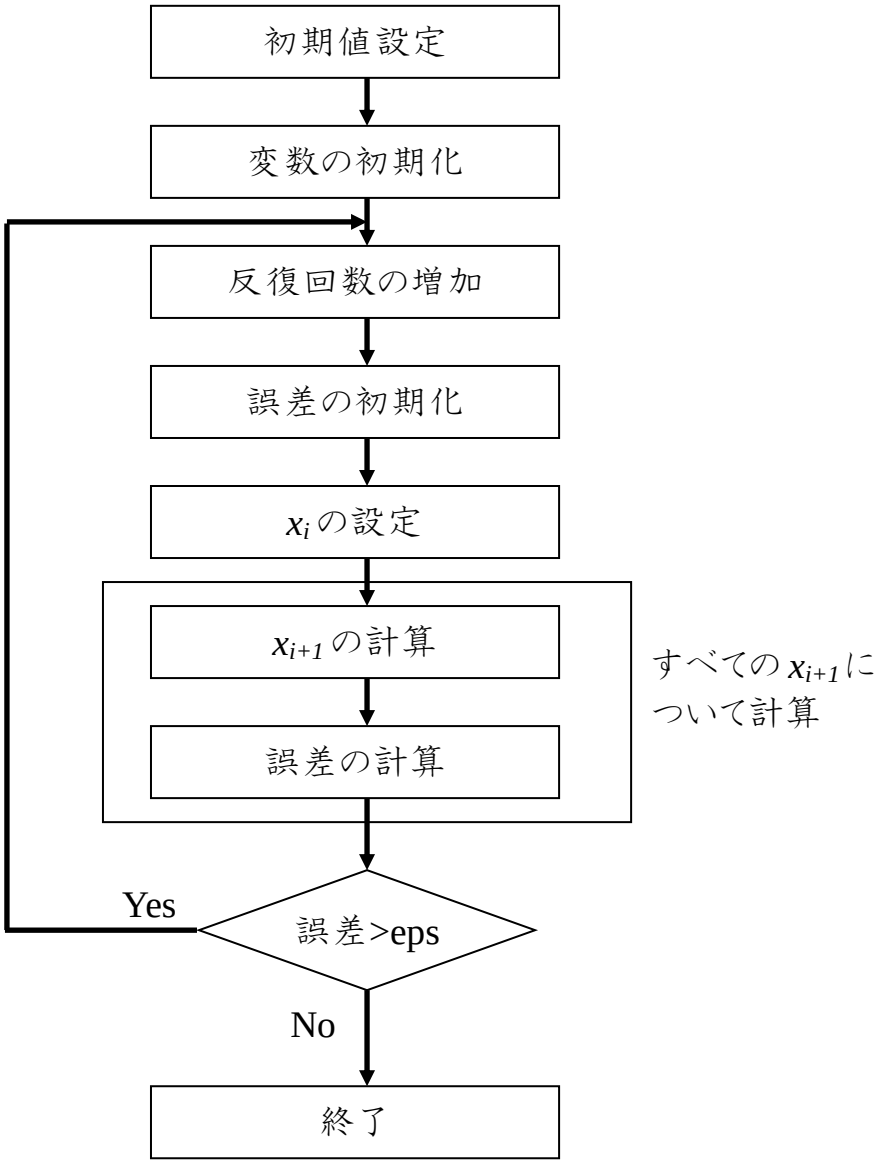
$$x_{i+1} = (b_1 - a_{12}y_i - a_{13}z_i)/a_{11}$$
$$y_{i+1} = (b_2 - a_{21}x_i - a_{23}z_i)/a_{22}$$
$$z_{i+1} = (b_3 - a_{31}x_i - a_{32}y_i)/a_{33}$$



このユーザーフォームのヤコビ法のコマンドボタンにプログラムを作り込んで行きます。プログラムを作る前にプログラムの流れを考える必要があります。今回は計算に必要な係数の入力、出力をコマンドボタンのプログラムに実行させ、ヤコビ法の部分はサブルーチン（別の部分にプログラムを書き、それを呼び出す形にする）で実行するようにします。これにより、サブルーチンはそれだけで開発することができ、別のプログラムを作るときに必要になればその部分だけ呼び出す形で利用することができます。さらに、そのようにサブルーチン化することにより、プログラムの流れを簡単にすることができます。コマンドボタンのプログラムの流れは次のようになります。



ヤコビ法の部分のプログラムの流れは次のようになります。



ヤコビ法の部分のプログラムはコマンドボタンの部分に記述するのではなく新たにサブルーチン(プロシージャ)を追加してそこに記述します。プロシージャの追加はメニューの挿入→プロシージャを選択します。するとプロシージャの追加のウィンドウが開きますのでここでは名前を YACOBIByRef として Sub プロシージャにチェックを入れて OK を押してください。するとこの名前のプロシージャが作成されます。

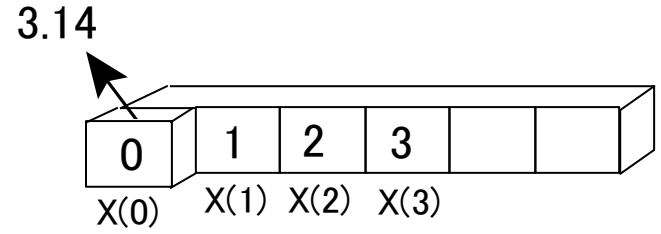
このフローチャートに対応するプログラムを次に示します。

```
Private Sub CommandButton1_Click()
    Dim Ma(5, 5) As Double
    Dim Mb(10) As Double                    ①
    Dim Mx(10) As Double
    Dim N As Integer
    Worksheets("連立一次方程式").Activate
    Set WS = Worksheets("連立一次方程式").Application
    N = 3
    Ma(0, 0) = TextBox1.Text
    Ma(0, 1) = TextBox2.Text
    Ma(0, 2) = TextBox3.Text
    Ma(1, 0) = TextBox5.Text
    Ma(1, 1) = TextBox6.Text
    Ma(1, 2) = TextBox7.Text
    Ma(2, 0) = TextBox9.Text
    Ma(2, 1) = TextBox10.Text
    Ma(2, 2) = TextBox11.Text
    Mb(0) = TextBox4.Text
    Mb(1) = TextBox8.Text
    Mb(2) = TextBox12.Text
    WS.Range("A6") = "ヤコビ法"
    WS.Range("B8") = "配列 Ma の内容"
    WS.Range("B9") = Ma(0, 0)
    WS.Range("D9") = Ma(0, 1)
    WS.Range("F9") = Ma(0, 2)
    WS.Range("B10") = Ma(1, 0)
    WS.Range("D10") = Ma(1, 1)
    WS.Range("F10") = Ma(1, 2)
    WS.Range("B11") = Ma(2, 0)
    WS.Range("D11") = Ma(2, 1)
    WS.Range("F11") = Ma(2, 2)
    WS.Range("H8") = "配列 Mb の内容"
    WS.Range("H9") = Mb(0)
    WS.Range("H10") = Mb(1)
    WS.Range("H11") = Mb(2)
    Call YACOBIByRef(Ma(), Mb(), Mx())      ←――――――――――  ②
    WS.Range("D14") = "繰り返し計算の回数"
    WS.Range("A13") = "ヤコビ法による方程式の解"
    WS.Cells(14, 5) = No
    WS.Cells(15, 4) = "Mx(0)="
    WS.Cells(16, 4) = "Mx(1)="
     WS.Cells(17, 4) = "Mx(2)="
    WS.Cells(15, 5) = Mx(0)
    WS.Cells(16, 5) = Mx(1)
    WS.Cells(17, 5) = Mx(2)
End Sub
```

```
Public Sub YACOBIByRef(Na() As Double, Nb() As Double, Nx() As Double)
    Dim AA As Double
    Dim eps As Double
    Dim error As Double
    Dim i As Integer
    Dim ii As Integer
    Dim j As Integer
    Dim k As Integer
    Dim Max As Integer
    Dim N As Integer
    Dim No As Integer
    Dim xx(10) As Double
    No = 0
    N = 3
    Max = 100
    eps = 0.00001
    For i = 0 To N - 1
        Nx(i) = 0
    Next
    For i = 0 To Max - 1                          ' ループi
        No = No + 1
        error = 0
        For ii = 0 To N - 1
            xx(ii) = Nx(ii)
        Next
        For j = 0 To N - 1                        ' ループj
            AA = Nb(j)
            For k = 0 To N - 1                    ' ループk
                If k <> j Then AA = AA - Na(j, k) * xx(k)
            Next
            Nx(j) = AA / Na(j, j)
            error = error + Abs(Nx(j) - xx(j))
        Next
        If error < eps Then Exit For
    Next
```

上記のプログラムはこれまで学んだプログラムの文法を用いるとほとんど理解できます。フローチャートに対応させて理解してみましょう。ここでは注意点と新しい事柄のみ説明します。

① 配列変数



このプログラムでは配列変数を使用しています。配列変数の概念は一連のデータを保存しておく引き出しでその引き出しに番号がついているものと考えてください（右図）。この配列変数は同じ属性を持つデータを入れることを前提としています。今回用いる行列のデータはその趣旨に非常に合います。たとえば行列の各成分を変数で表すときにこれまでのように一つ一つ宣言しているときりがありませんし、汎

用性のあるプログラムを作ることは不可能です。というのも行列の大きさが3×3とは決まっていませんし、どの様なサイズの行列を扱うか決まっていない場合も多々あります。そのような場合にはプログラムが書けないことになります。②で宣言されているのが配列です。例えば、

Dim X(10) as Double

配列の宣言はこれまで学んだ変数の宣言とほとんど同じです。異なる点は変数名に()が含まれていて、その中に数字が書かれている点です。この数字の意味は何個分のデータの箱を用意すればよいかということを意味します。この場合ですと10個の箱つまり、X(0)～X(9)まで用意されます。

配列はどのプログラム言語でも利用される非常に重要な事柄です。これなくしてプログラムは書けないことになります。これの利点は同じ性質を持つデータをひとまとめにして扱うことができるだけでなく、配列のインデックス(括弧内の数字)を変数としてあつかうことができる点です。例えば下のような命令です。

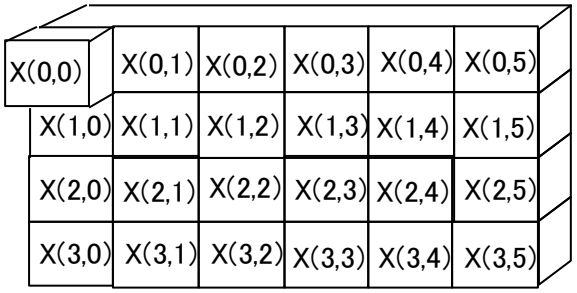
```
For i = 0 To N - 1
    Mx(i) = 0
Next
```

このプログラムは繰り返しの命令ですが、繰り返し変数はiです。この繰り返し変数は 0 から N-1 間で変化します。この繰り返し変数が配列のインデックスとして指定されています。例えば N=3 とすると次のように実行されます。

i	実行
0	Mx(0)=0
1	Mx(1)=0
2	Mx(2)=0

3回くらいの繰り返しなら、直接書いてもたいしたことはありませんが、100回、1000回となると直接書くことは現実的ではありません。配列の有効性が発揮されます。

配列は1次元だけでなく、多次元配列も用いることができます。多次元配列の定義は次の通りです。



Dim X(10,10) as Double

これは2次元配列の例です。2次元配列はこれこそ表のような表し方です。配列のインデックスは2つあり、右図のようなイメージを持ってください。これはまさに行列そのものです。行列の成分を示すときにインデックスでそのまま指

定することができます。2次元配列だけでなく3次元、4次元と次元を増やすことが可能です。ただし、次元数（最大60次元）が増えると急速に必要なメモリの量も増えます。

② サブプロシージャの呼び出し

ヤコビ法の部分は次のように呼び出されています。

Call YACOBIByRef(Ma(),Mb(),Mx())

この命令で、Call というのがサブプロシージャの呼び出しを行っています。YACOBIByRef はプロシージャの作成で指定した名前です。一方で呼び出されるプロシージャの先頭行は次のようになっています。

Public Sub YACOBIByRef(Na() As Double, Nb() As Double, Nx() As Double)

括弧の中は数値の受け渡しを行っています。それぞれの色に対応した部分がプロシージャ間で受け渡されます。

Call YACOBIByRef(Ma(),Mb(),Mx())

サブプロシージャ内では与えられた係数を用いてヤコビ法の計算を行います。ここでは前回勉強した繰り返しが多用されています。特に、ループ i、ループj、ループkは多重ループになっています。

```
For i = 0 To Max - 1
    For j = 0 To N - 1
        For k = 0 To N - 1
            If k <> j Then AA = AA - Na(j, k) * xx(k)
        Next
    Next
Next
```