CalcLib Example of Calc::Mtx

```
Calc::Mtx Example
Objective: set coefficients for a test polynomial. Evaluate incremental data points.   Build supporting
matrices for least-squares analyses. Then, take least-squares of data points, recover coefficients.
Compare for errors.
References:
   - Press, William H, et al, Numerical Recipes, 2nd Ed, Cambridge Press, 1992.
   - Spiegel, MR, et al, Schaum's Mathematical Handbook, 5th Ed, McGraw-Hill, 2018.
Author: Everett George Copyright (C) 2023
   https://www.jefgeorge.com
   07/06/2023
```

Output:

```
calc::Mtx Class Example Application

Initial Coefficients:          -9.3000,9.3000  1.1000,-1.1000  -5.1000,5.1000  0.0000,0.0000
Least-Squares Coefficients:    -9.1875,9.4062  1.1030,-1.1406  -5.1006,5.1008  7.6294e-06,1.6093e-06
```

C++ Source:

```
#include "Cpx.h"                                    // complex variable definitions
#include "Mtx.h"                                    // matrix definitions

using namespace std;                                // standard C++ library namespace
using namespace calc;                               // CalcLib namespace

typedef Cpx<float> CPXf;                            // alias for float type complex numbers
typedef Mtx<CPXf > MTXc;                            // alias for complex matrices

// results print function
void results(ostream& ostr, const char *cTle, const CPXf *a, int n)
   {ostr<<"   "<<cTle<<"  ";
      for(int i=0; i<n; i++) ostr<<"  "<<a[i]; ostr<<endl;}

// least-squares curve-fit function
void leastSquares(CPXf *c, const CPXf *u, const CPXf *v, int npts, int nvar)
{
   // convert data arrays into matrix objects
   MTXc mU(u,npts,nvar);
   MTXc mV(v,npts,1);

   // set up augmented matrix for Ax=B
   MTXc mA=(~mU)*mU;                                // build square matrix: transpose(U)*U = A
   MTXc mB=(~mU)*mV;                                // build coefficient vector: transpose(U)*V = B

   // solve matrix with inverse of matrix A times B
   MTXc mC=(!mA)*mB;
   mC.copyArray(c);                                 // best-fit coefficients in C as solution
}

int main(void)
{
   // define matrix size
   const int NPTS=40;                               // number of data points to curve-fit
   const int NVAR=4;                                // number of variables to use in curve-fit

   // set coefficients for check
   const CPXf aC0[NVAR]=                            // least-squares method recovers these values
      {CPXf(-9.3f, 9.3f),CPXf( 1.1f,-1.1f),
       CPXf(-5.1f, 5.1f),CPXf( 0.0f, 0.0f)};

   // declare solution variables
   CPXf aU[NPTS][NVAR];                             // independent variable data array
   CPXf aV[NPTS];                                   // dependent variable data vector
   CPXf aC[NVAR];                                   // least-squares coefficient array

   // build data arrays
   for(int i=0; i<NPTS; i++) {
      const CPXf x((float)(i),(float)(-i));
      aU[i][0]=1.0f;                                // matrix defines best-fit equation structure
      aU[i][1]=x;                                   // equation is 3rd degree polynomial
      aU[i][2]=x*x;
      aU[i][3]=x*x*x;
      aV[i]=aC0[0]+aC0[1]*x+aC0[2]*x*x+aC0[3]*x*x*x;  // dependent data vector
   }                                                // usually generated from outside source

   // curve-fit aV data to 3rd degree polynomial in aU
   leastSquares(aC,&aU[0][0],aV,NPTS,NVAR);
```

```
   // print out coefficients
   cout.precision(5);
   cout.setf(ios::showpoint,ios::showpoint);
   cout<<endl<<"   calc::Mtx Class Example Application"<<endl<<endl;
   results(cout,"Initial Coefficients:       ",aC0,NVAR);
   results(cout,"Least-Squares Coefficients: ",aC ,NVAR);
   cout<<endl<<flush;

   return 0;
}
```