The aim of this code is use Levenberg–Marquardt Method to solve Bundle Adjustment. It can be summarized as followed:

cost = 0, lastcost = 0
v = 2, iter = 0
while iter < MaxIter && cost < lastcost
    Q_down = 0
    H = $J^T J$, b = -$J^T f(x)$
    Solve H*x_new = b
solveProblem(const int iteration)
    cost_new = $f(x\_new)^T f(n\_new)$
    Q = (cost – cost_new) / Q_down
    If Q > 0
        x = x_new
        mu *= max$\{\frac{1}{3}, 1-(2Q-1)^3\}$
        v = 2
    else
        mu *= v
        v *= 2
computeUpdate(clMat*, double, double, int)

I use seven steps to solve Hx = b.

$$\begin{bmatrix} H_{11} & H_{12} \\ {H_{12}}^T & H_{22} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$$

$$\left(H_{11} - H_{12}{H_{22}}^{-1}{H_{12}}^T\right)x_1 = b_1 - H_{12}{H_{22}}^{-1}b_2$$

$$H_{22}x_2 = b_2 - {H_{12}}^T x_1$$

Step1: calculating the ${H_{22}}^{-1}$

```
clMat* getH22Inverse(const clMat* H22);
```

Step2: $\mathrm{T} = H_{12}{H_{22}}^{-1}$

```
clMat* getMatrixT(const clMat* H12, const clMat* H22inverse);
```

Step3: $b_1 = b_1 - Tb_2$

```
void getNewb1(const clMat* T, const clMat* b2, clMat* b1);
```

Step4: $\mathrm{A} = \mathrm{T}\,{H_{12}}^{T}$

```
clMat* getMatrixA(const clMat* T, const clMat* H12);
```

Step5: $(H_{11} - A)x_1 = b_1$

```
clMat* getMatrix_x1(const clMat* A, const clMat* H11, const clMat* b1);
```

Step6: $b_2 = b_2 - {H_{12}}^{T}x_1$

```
void getNewb2(const clMat* H12, const clMat* x1, clMat* b2);
```

Step7: $x_2 = {H_{22}}^{-1}b_2$

```
clMat* getMatrix_x2(const clMat* H22inverse, const clMat* b2);
```