```
- init
{
        [super init];
        ......... // some code here

        mmaProcessor = [[MiscMMAProcessor alloc] init];

        // we let mmaProcessor open a link and initialize a Mathlink environment
        // this is a "handy" code block
        if(![mmaProcessor openLink]){
                .......   // error
        }
        .......
        return self;
}


- (void)invertMatrix:(const double *)matrix size:(int)n in:(double *)result
{
        ..........                  // some code here
        tempBuffer = result;       // suppose tempBuffer is an instance variable

        MLINK lp = [mmaProcessor link];

        MLPutFunction(lp, "Inverse", 1);
        MLPutFunction(lp, "List", n);
                for(i = 0; i < n; i++){
                        MLPutFunction(lp, "List", n);
                        for(j = 0; j < n;j++){
                        MLPutReal(lp, matrix[i * n + j]);
                        }
                }
        MLEndPacket(lp);

        // we let mmaProcessor read packets for us
        // this is another "handy" code block
        // note here that we provide ourselves to read the RETURNPKT packet
        // which interests us (the inverted matrix)
        // all other packets are read by mmaProcessor and prepared for the
        // returning NSDictionary
        // we have to read the matrix in the method below, which gets called
        // when mmaProcessor encounters the desired packet

        [mmaProcessor readReturnWithReader:self];

}

- (void)readReturnPacketWithProcessor:(MiscMMAProcessor *)aProcessor // MiscMMAReturnPacketReading protocol
{
                double *mmaReturnList;
        long i,j,mmaReturnCount, len;

        ....... // some code

        MLINK lp = [mmaProcessor link];

        MLCheckFunction(lp,"List",&len);

        for(i = 0; i < len; i++){
                if(!MLGetRealList(lp,&mmaReturnList,&mmaReturnCount)){
                        ..... // error
                }

                for(j = 0; j < mmaReturnCount; j++)
                        tempBuffer[i * mmaReturnCount + j] = mmaReturnList[j];

                MLDisownRealList(lp,mmaReturnList,mmaReturnCount);
        }
}
```