NAG Library Installation

SPK Library is dependent on some of the basic linear algebra routines offered by the NAG C Library. The NAG shared library must be installed on each machine in order build and/or execute SPK applications. This document focuses on installation of components minimally required for building the SPK library and its applications since the library is about to be replaced by and open-source equivalent(s) soon.

Table of Contents

Installation

1. As an ordinary user, download the Nag C Mark 7 library from whitechuck to /tmp on your machine:

```
cd /tmp
scp whitechuck:/opt/download/cllux07ddl.tgz .
```

2. From the /tmp directory, type the following to uncompress and unpack the tar ball:

```
tar xvzf cllux07ddl.tgz
```

3. To install the NAG libraries (ie. static and shared versions) and headers into proper locations, you must first become root. Note that it has no "-".

```
su
```

   You should be in the directory, /tmp/cllux07ddl/.

4. Install the library and headers in /usr/local:

```
cp libnagc.a /usr/local/lib
cp libnagc.so.7 /usr/local/lib
cp include/* /usr/local/include
cd /usr/local/lib
ln -s libnagc.so.7 libnagc.so
```

   If your linker failed to find a nag libarary, make sure /usr/local/lib is in the linker's search path. Add /usr/local/lib in LD_LIBRARY_PATH environment variable.

5. Create or update `/usr/local/flexlm/licenses/license.dat` with the following license key entries:

```
FEATURE NAG_CL NAG 9.900 01-aug-2004 0 BC5E444363B0CF819FC9 "ANY" DEMO
FEATURE NAG_FD NAG 9.900 01-aug-2004 0 CCDE34737BB5C767BCC6 "ANY" DEMO
FEATURE NAG_FL NAG 29.900 01-aug-2004 0 AC9E44335FBDCF5F94E2 "ANY" DEMO
FEATURE NAG_FN NAG 9.900 01-aug-2004 0 BC3E3473619FD15D8CE0 "ANY" DEMO
FEATURE IRIS_Explorer NAG 9.900 01-aug-2004 0 1CEEA44345A79C7B1394 "ANY" DEMO
INCREMENT NAGWaref95 NAG 9.900 01-aug-2004 0 FCEE4483B7ADA5C0CA49 "ANY" DEMO
FEATURE NAGWareFtools NAG 9.900 01-aug-2004 0 ECDEB423F87A659A99EB "ANY" DEMO
```

6. Clean up

```
cd /tmp
rm cllux07ddl.tgz
rm -rf cllux07ddl
```

Verification

1. Copy and past the following code into a new file and save the file in somewhere as `nag_test.cpp`.

```
#include <iostream>
#include <valarray>
#include <cassert>
#include <string>

extern "C"{
# include "nag.h"
# include "nag_types.h"
# include "nag_names.h"
# include "nagf06.h"
}

using namespace std;
static void NagDgemm(
    char transa,            // Is matrix A transpose (At) or non (An)?
    char transb,            // Is matrix B transpose (Bt) or non (Bn)?
    int m,                  // (An) A->nr, (At) A->nc
    int n,                  // (Bn) B->nc, (Bt) B->nr
    int k,                  // (An) A->nc, (Bn) B->nr, (At) A->nr, (Bt) B->nc
    double alpha,           // scalar
    const double A[],       // column major matrix
    const double B[],       // column major matrix
    double beta,            // scaler
    double C[]              // column major matrix
);

static const valarray<double> multiply( const valarray<double> & A, int nColsA,
     const valarray<double> & B, int nColsB );

void printInMatrix( const valarray<double> & A, int cols );

int main()
{
  //      /         \
  //      |  1  2  |
  // A = |         |
  //      |  3  4  |
  //      \         /
  //
  valarray<double> A(4);
  for( int i=0; i<4; i++ )
    A[i] = i+1;

  //
  //      /      \
  //      |  2  |
  // B = |      |
  //      |  2  |
  //     \     /
  //
  valarray<double> B(2.0, 2);

  valarray<double> C = multiply( A, 2, B, 1 );
  valarray<double> correctC( 2 );
  correctC[0] = 6.0;
  correctC[1] = 14.0;

  cout << "A = " << endl;
  printInMatrix( A, 2 );
  cout << endl;

  cout << "B = " << endl;
  printInMatrix( B, 1 );
  cout << endl;

  cout << "C = " << endl;
  printInMatrix( C, 1 );
  cout << endl;

  if( C[0] == correctC[0] && C[1] == correctC[1] )
    cout << "Test passed successfully!\n" << endl;
  else
    cout << "Test failed!\n" << endl;
  return 0;
}

void printInMatrix( const valarray<double>& A, int cols )
{
  int rows = A.size() / cols;
  assert( rows * cols * A.size() );

  for( int i=0; i < rows; i++ )
    {
      cout << "[ ";
      for( int j=0; j < cols; j++ )
 {
   cout << A[j + i * cols] << " ";
 }
      cout << "]" << endl;
    }
}
const valarray<double> multiply( const valarray<double>& A, int nColsA,
                                 const valarray<double>& B, int nColsB )
{
  using namespace std;

  if( A.size() == 0 || B.size() == 0 )
          return valarray<double>(0);

  int nRowsA = A.size() / nColsA;
  assert( nRowsA * nColsA == A.size() );

  int nRowsB = B.size() / nColsB;
  assert( nRowsB * nColsB == B.size() );

  assert( nColsA == nRowsB );

  const double *pA = &(A[0]);
  const double *pB = &(B[0]);

  valarray<double> C( nRowsA * nColsB );
  double *pC = &(C[0]);

  /***************************************************************
```

```
   *
   * dgemm (f06yac) performs real matrix-matrix multiplication:
   *    C = (alpha * A B) + (beta * C)
   *
   ************************************************************/
  NagDgemm(’n’, ’n’, nRowsA, nColsB, nColsA, 1.0, pA, pB, 0.0, pC);
  return C;
};

# include <stdio.h>
# include <stdlib.h>


static void NagDgemm(
    char transa,            // Is matrix A transpose (At) or non (An)?
    char transb,            // Is matrix B transpose (Bt) or non (Bn)?
    int m,                  // (An) A->nr, (At) A->nc
    int n,                  // (Bn) B->nc, (Bt) B->nr
    int k,                  // (An) A->nc, (Bn) B->nr, (At) A->nr, (Bt) B->nc
    double alpha,           // scalar
    const double A[],       // row major matrix
    const double B[],       // row major matrix
    double beta,            // scaler
    double C[]              // row major matrix
){
    const char* errmsg =
    "\nNagDgemm received an invalid value in the %s argument (%c)...terminating pr


    MatrixTranspose transA;
    MatrixTranspose transB;

    if( transa == ’n’ || transa == ’N’ ){
        transA = NoTranspose;
    }
    else if( transa == ’t’ || transa == ’T’ ){
        transA = Transpose;
    }
    else{
        fprintf(stderr, errmsg, "1st", transa);
        exit(-1);
    }

    // for matrix B
    if( transb == ’n’ || transb == ’N’){
        transB = NoTranspose;
    }
    else if( transb == ’t’ || transb == ’T’ ){
        transB = Transpose;
    }
    else{
        fprintf(stderr, errmsg, "2nd", transb);
        exit(-1);
    }

    f06yac(
        transA,
        transB,
        (Integer) m,
        (Integer) n,
        (Integer) k,
        alpha,
        A,
        (Integer) k,
        B,
        (Integer) n,
        beta,
        C,
        (Integer) n
    );
}
```

2. Compile `nag_test.cpp` and statically link to the NAG library which requires two other libraries: `pthreadlib` and `mlib`:

   ```
   g++ nag_test.cpp -lnagc -lpthread -lm -o test
   ```

   The following warning messages will be displayed. Please ignore.

   ```
   s09zzft.o(.text+0x479b): ‘sys_errlist’ is deprecated; use ‘strerror’ or ‘strerror_
   s09zzft.o(.text+0x4785): ‘sys_nerr’ is deprecated; use ‘strerror’ or ‘strerror_r’
   ```

3. If the above compilation completed sucessfully, you should be able to run the built executable `test`:

   ```
   ./test
   ```

   and see the following output on screen.

   ```
   [Honda@pasta myTemp]$ ./test
   A =
   [ 1 2 ]
   [ 3 4 ]

   B =
   [ 2 ]
   [ 2 ]

   C =
   [ 6 ]
   [ 14 ]

   Test passed successfully!
   ```