NAG Library Installation

SPK Library relies on some of the components from NAG Library such as a re-entrant optimizer, matrix mutiplication routine and so on. The installation package comes as a simple compressed archive. The installer is expected to unpack the package and manually place headers and libraries in appropriate places. This tutorial explains these steps.

Table of Contents

Installation

1. As ordinary user, download the linux version of the Nag C library from whitechuck:

```
cd /tmp
scp whitechuck:/opt/download/lux06dbt.Z .
```

2. Become root,

```
su -
```

3. Unpack the archive

```
cd /tmp
tar xvzf lux06dbt.Z
```

4. Install the library and headers

```
cd cllux06db
cp libnagc* /usr/local/lib
cp include/* /usr/local/include
cd /usr/local/lib
ln -s libnagc.so.6 libnagc.so
```

5. Make sure `/usr/local/flexlm/licenses/license.dat` contains 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 lux06dbt.Z
rm -rf cllux06db
```

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>

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 );

  double pA[ A.size() ];
  for( int i=0; i<A.size(); i++ )
     pA[i] = A[i];
  double pB[ B.size() ];
  for( int i=0; i<B.size(); i++ )
     pB[i] = B[i];
  double pC[ nRowsA * nColsB ];

  NagDgemm(’n’, ’n’, nRowsA, nColsB, nColsA, 1.0, pA, pB, 0.0, pC);
  return C;
};

/*******************************************************************************
 *
 *                NagDgemm Implementation
 *
 * Cpp interface to Nag’s dgemm (f06yac)
 *
 * dgemm (f06yac) performs real matrix-matrix multiplication:
 *    C = (alpha * A B) + (beta * C)
 *
 * This function terminates the program when receives an invalid ar-
gument value.
 *
 *******************************************************************************/
# include <stdio.h>
# include <stdlib.h>

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

    dgemm(
        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 -static -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_r’ instead
   s09zzft.o(.text+0x4785): ‘sys_nerr’ is deprecated; use ‘strerror’ or ‘strerror_r’ instead
   ```

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!
   ```