Implementing Triangular Solve

Due March 10, 2011

The purpose of this exercise is to implement the lower and upper triangular solves that can be used to solve \( Ax = b \) when \( A \) has been factored into an LU factorization: \( A = LU \) where \( L \) is lower triangular and \( U \) is upper triangular.

1 Lower triangular solve

Write a routine, SLAP_Ltrsv_n_unb_var1,

\[
\text{function } \text{[bout]} = \text{SLAP}_n\text{Ltrsv}_n\text{unb}_\text{var1}(\text{diag}, L, b)
\]

That assumes that \( L \) is stored in the lower triangular part of array \( L \) and overwrites \( b \) with the solution of \( Lx = b \), returning the result in \( \text{bout} \) (meaning that when you use the Spark tool, you will want to make \( b \) input/output.

The parameter \( \text{diag} \) can take on the values SLAP_NON_UNIT_DIAG or SLAP_UNIT_DIAG. In the case of the former, the elements on the diagonal of \( L \) are simply those stored in the array \( L \). In the case of the latter, the elemental on the diagonal of \( L \) are not stored, but are assumed to be one (as happens when you overwrite matrix \( A \) with its LU factorization).

Optionally, also implement a second variant. One would access \( L \) by rows, the other by columns.

2 Upper triangular solve

Write a routine, SLAP_n_Utrsv_unb_var1,

\[
\text{function } \text{[bout]} = \text{SLAP}_n\text{Utrsv}_n\text{unb}_\text{var1}(\text{diag}, U, b)
\]

That assumes that \( U \) is stored in the upper triangular part of array \( U \) and overwrites \( b \) with the solution of \( Ux = b \), returning the result in \( \text{bout} \) (meaning that when you use the Spark tool, you will want to make \( b \) input/output.

The parameter \( \text{diag} \) can take on the values SLAP_NON_UNIT_DIAG or SLAP_UNIT_DIAG. In the case of the former, the elements on the diagonal of \( U \) are simply those stored in the array \( U \). In the case of the latter, the elemental on the diagonal of \( U \) are not stored, but are assumed to be one (as happens when you overwrite matrix \( A \) with its LU factorization).

Optionally, also implement a second variant. One would access \( U \) by rows, the other by columns.

**Warning:** To solve \( Ux = b \) you have to run backwards through the matrix!

3 Wrapper routine

Write a routine, SLA_Trsv,

\[
\text{function } \text{[bout]} = \text{SLA}_\text{Trsv}(\text{uplo}, \text{trans}, \text{diag}, A, b)
\]

As before, \( b \) can be either a row or a column vector. Depending on parameter \( \text{uplo} \) you call SLAP_Ltrsv_n_unb_var1 or SLAP_Utrsv_n_unb_var1. You may ignore the \( \text{trans} \) parameter. But if you are ambitious, you will make everything work (You now notice that the \( \_n \) in the function names stand for *no transpose* so that one can also write routines for the transpose case.)