Quasi triangular matrices multiplication in Kalman filter estimation

Chapter 1

Problem setting

We want to optimize the matrix multiplication used in the estimate of the Kalman filter.

In particular we want to speed up the computation of the following steps:

$$\begin{aligned} &Ta_{t-1|t-1} \\ &TP_{t-1|t-1}T' \end{aligned} \tag{1.1}$$

where

T is upper quasi-triangular matrix;

P is symmetric (with diagonal non negative diagonal elements);

a is a vector.

Chapter 2

Strategy

The main idea is to replace right multiplication $*T'$ with transpose, where T is a lower quasi-triangular matrix. Given the property of the matrices T and P and since we need to compute TST' we propose the following strategy.

We first write (see QT2Ld.f90 and QT2T.f90)

$$T = T_1 + L_d \tag{2.1}$$

as the sum of a upper triangular matrix $T_1$and a lower sub diagonal matrix

$$L_d = \begin{pmatrix} 0 & 0 & \dots & 0 & 0 \\ * & 0 & \dots & 0 & 0 \\ \vdots & * & \dots & 0 & 0 \\ & \ddots & * & 0 & 0 \\ 0 & 0 & \dots & * & 0 \end{pmatrix}$$

where only elements $L_d(i,i-1)$ may be different from zeros.

We also write the symmetric matrix (see S2U.f90 and S2D.f90)

$$P = U + D + U' \tag{2.2}$$

where

$$U = \begin{pmatrix} 0 & P_{12} & \dots & & P_{1n} \\ 0 & \ddots & & & \\ \vdots & & & & \\ & & & & P_{n-1n} \\ 0 & \dots & & 0 & 0 \end{pmatrix}$$

and

$$D = \begin{pmatrix} P_{11} & 0 & \dots & 0 \\ & P_{22} & & \\ \vdots & & \ddots & \\ & & & \\ 0 & & \dots & P_{nn} \end{pmatrix}$$

.

2.1 Ta

According to decomposition (2.1) we write

$$Ta = T_1 a + L_d a.$$

It is easy to see that $L_d a = (0, T_{2,1}a_2, \dots, T_{n,n-1}a_n)'$ (see LdV.f90). The multiplication (see TV.f90) $T_1 a$ consists in $\frac{N*N*(N+1)}{2}$ operations.

2.2 TPT'

According to decomposition (2.1) and (2.2) we write

$$\begin{aligned} TPT' =& (T_1 + L_d)P(T_1' + L_d') \\ =& T_1 PT' + T_1 PL_d' + L_d PT_1' + L_d PL_d'. \end{aligned} \tag{2.3}$$

We observe that $L_d M$, where M is an arbitrary matrix, is equal to the matrix whose only non zero rows are those corresponding to the non zero elements of L_d (see LdM.f90).

We can use the symmetricitity of P and consider as a whole $T_1 PL_d' + L_d PT_1'$. We first compute $X = T_1 P$ (see TM.f90), we have

$$\begin{aligned} T_1 SL_d' + L_d ST_1' =& XL_d + L_d X' \\ =& L_d X' + (L_d X')'. \end{aligned}$$

In this way we have replaced right multiplication by T' and, instead, we use left multiplication by L_d (which we know it is very low computational demanding) and a matrix transpose.

The computation of L_dSL_d' is quite straightforward (see LdSLd.f90), since it ends out top be equal to the matrix whose elements are given by

$$L_d(i,i-1)S(i-1,j-i)L_d(j,j-1),$$

i.e. the only non zero terms are those corresponding to the non zeros elements of L_d.

To compute T_1PT_1' , we propose two different approaches:

direct computation: we directly compute T_1PT_1' using properties of triangular and symmetric matrices (see TSTt.f90)

decomposition of symmetric matrices: if we may use multiple processors, we may think of computing T_1PT_1' by applying (2.2). We write

$$\begin{aligned} T_1PT_1' =& T_1(U+D+U')T_1' \\ =& T_1UT_1' + T_1DT_1' + T_1U'T_1'. \end{aligned} \tag{2.4}$$

We observe that the diagonal elements of S are all non negative. We then define $\sqrt{D}$as the diagonal matrix whose elements are the square roots of the diagonal elements of S (which is D in (2.2)).

Let $Y_U = T_1U$ (see TU.f90) and $Y_D = T_1\sqrt{D}$ (see TD.f90).

Equation (2.4) becomes

$$\begin{aligned} T_1UT_1' + T_1DT_1' + T_1U'T_1' =& Y_UT_1' + Y_DY_D' + T_1Y_U \\ =& T_1Y_U + (T_1Y_U)' + Y_DY_D'. \end{aligned}$$

Once again instead of right multiplication by a lower triangular matrix (T_1'), we need to compute matrix transposes.

Setting $X_1 = L_dX' = L_d(T_1P)'$ (see LdM.f90), $X_2 = T_1Y_U' = T_1(T_1U)'$ (see TM.f90) and $X_3 = Y_D$, we have

$$\begin{aligned} TPT' =& X_1 + X_1' + L_dPL_d' \\ &+ X_2 + X_2' + X_3X_3', \end{aligned}$$

which can be also thought in terms of parallel routines (if multiple processors are available).

Chapter 3

Routines

In order to apply the proposed strategies, we have the following FORTRAN 90 routines:

- QT2Ld.f90: extracts the lower diagonal part L_d from an upper quasi-triangular matrix QT
- QT2T.f90: extracts the upper triangular part T from an upper quasi-triangular matrix QT
- S2D.f90: extracts the square diagonal part $\sqrt{D}$ from a symmetric matrix S
- S2U.f90: extracts the over diagonal part U from a symmetric matrix S
- TV.f90: $T * V$ where T is upper triangular and V is a vector
- LdV.f90: $L_d * V$ where L_d is lower diagonal and V is a vector
- LdSLd.f90: $L_d * S * L_d'$ where L_d is lower diagonal and S is symmetric
- LdM.f90: $L_d * M$ where L_d is lower diagonal and M is arbitrary
- TD.f90: $T * V$ where T is upper triangular and D is a diagonal
- TM.f90: $T * M$ where T is upper triangular and M is arbitrary
- TU.f90: $T * U$ where T is upper triangular and U is over diagonal
- TUt.f90: $T * U'$ where T is upper triangular and U is over diagonal
- TT.f90: $T_1 * T_2$where both $T_1$and $T_2$are upper triangular matrices
- TSTt.f90: $T * S * T'$ where T is upper triangular and S is a symmetric

Chapter 4

Sequence of routine's calls

Given an upper quasi-triangular matrix T and a symmetric matrix P, whose dimension is equal to n and a vector a, we do the following steps:

Ta**:**

1. $T_1 = QT2T(QT,n)$ and $L_d = QT2L_d(QT,n)$;
2. $Ta = LdV(L_d,a,n) + TV(T_1,a,n)$.

TPT'**:**

Case 1:

1. $T_1 = QT2T(QT,n)$ and $L_d = QT2L_d(QT,n)$;
2. $X = TM(T_1,S,n)$ and $X_2 = LdSLd(L_d,P,n)$;
3. $X_1 = LdM(L_d,X',n)$, $X_3 = TSTt(T_1,P,n)$;
4. $TPT' = X_1 + X_1' + X_2 + X_3$

Case 2:

1. $T_1 = QT2T(QT,n)$ and $L_d = QT2L_d(QT,n)$;
2. $U = S2U(P,n)$ and $\sqrt{D} = S2D(S,n)$;
3. $X = TM(T_1,S,n)$, $Y_U = TU(T_1,U,n)$ and $Y_D = TD(T_1,D,n)$;
4. $X_1 = LdM(L_d,X',n)$, $X_2 = TM(T_1,Y_U,n)$, $X_3 = TM(Y_D,Y_D',n)$ and $X_4 = LdSLd(L_d,P,n)$;
5. $TPT' = X_1 + X_1' + X_4 + X_2 + X_2' + X_3$.