Process control. Review of concepts

1. Introduction

There are several aspects, why many chemical process companies are using NIR instruments for process control. The cost savings of NIR measurements are related to improved control and product quality. The NIR method produces analysis of certain points of the process. The NIR instrument can provide results within few minutes even in few seconds, which is often significantly faster compared to traditional laboratory analysis. In batch processes the quality of the final product the quality can be obtained several times within a manufacturing cycle, instead of analyzing only the quality of end batch. On-line process data can also reveal problems early allowing early corrective actions. Also safety aspects can be seen as one of the advantages due to intrinsically safe measurement probes and fiber optics. On-line measurements are also demanding with respect to instrument and sample handling system design. The non-contact characteristics of NIR measurements are well suited to coping with these variations, and recent improvements in optical design have led to devices that are virtually insensitive to both product pass height variation and presentation. The NIR instrument provides a lot of data, which is nowadays cheap to store, and to send onwards. However, the growth of NIR technology has always been dependent on a synergism of technologies, which together provide this powerful analytical tool. The lack of development of these technologies may also limit the use of NIR or other spectral technologies. The mathematical data handing should be included to this list of these vital technologies.

Looking at the linear least squares methods can show the problems involved. Suppose the response variable, $\mathbf{y}$, represents some quality measure. The linear least squares method is looking for a solution, $\mathbf{b}$, such that the measure $|\mathbf{Xb}-\mathbf{y}|^2$ is minimized. The exact solution is given by $\mathbf{b}=(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$. For NIR data the matrix $\mathbf{X}^T\mathbf{X}$ will be 1050×1050, but the solution should be based on a four or five dimensions. If the number of samples, N, is sufficiently large, we may be able to compute this solution, possible using extended precision. But the solution will have 1050 values, many of which are large, and it will be useless for prediction purposes.

Authors have developed and suggested a strategy called COVPROC (*COVariance PROCedures*), which has been proven to lead stable solutions, when reliable predictions based on dynamic process data are needed. The approach chosen here is to use the H-principle of mathematical modelling (Höskuldsson (1996)). The basic idea is to carry out the modeling in steps and at each step compute a rank one approximation to the solution. This rank one solution is based on optimizing the balance between the improvement in fit and the associated precision that can be obtained by such an improvement in the solution. Thus, each of the rank one part is a result of optimization task involving fit and precision, such that all parts are in certain sense optimal at the respective step of the analysis.

2. Linear mathematical models

Suppose that there are given values of the instrumental variables that have been collected in a matrix $\mathbf{X}$. A common assumption in standard linear regression is to assume that the response variable can be derived linearly from $\mathbf{X}$ apart from small random values that are assumed normally distributed. We write it as $\mathbf{y}=\mathbf{X}\boldsymbol{\beta}+\boldsymbol{\varepsilon}$, or $\mathbf{y}\sim N(\mathbf{X}\boldsymbol{\beta},\sigma^2\mathbf{I})$. This indicates that the residuals have the same variance, σ^2. The linear least squares method is concerned finding the value of $\boldsymbol{\beta}$ that minimizes the measure of fit, $|\mathbf{X}\boldsymbol{\beta}-\mathbf{y}|^2 \rightarrow$min. The exact solution, $\mathbf{b}_1$, to this task is given by

$\mathbf{b}_1=(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$. Sometimes there is a requirement that the solution vector in some sense should be as small as possible. This can be included in the optimization task as minimizing the sum $\boldsymbol{\beta}^T\mathbf{U}\boldsymbol{\beta} + |\mathbf{X}\boldsymbol{\beta}\text{-}\mathbf{y}|^2 \rightarrow \min$. The exact solutions are $\mathbf{b}_2=(\mathbf{X}^T\mathbf{X}+\mathbf{U})^{-1}\mathbf{X}^T\mathbf{y}$. The matrix $\mathbf{U}$ can be the unity matrix, $\mathbf{I}$, a constant times the unity matrix, k$\mathbf{I}$, the covariance matrix for the $\mathbf{b}$'s or some other positive definite matrix. A common choice for $\mathbf{U}$ is k$\mathbf{I}$, where the constant k is chosen by some external condition, e.g., the value that gives the smallest leave-one-out predictions. This is the popular Ridge Regression method.

There are occasions, where a good value of $\boldsymbol{\beta}$, $\boldsymbol{\beta}_0$, is known, and we want to include it in the model. One possibility is $(\boldsymbol{\beta}\text{-}\boldsymbol{\beta}_0)^T\mathbf{U}(\boldsymbol{\beta}\text{-}\boldsymbol{\beta}_0) + |\mathbf{X}\boldsymbol{\beta}\text{-}\mathbf{y}|^2 \rightarrow \min$. It has the exact solution $\mathbf{b}_3= (\mathbf{X}^T\mathbf{X}+\mathbf{U})^{-1}(\mathbf{X}^T\mathbf{y}+\mathbf{U}\boldsymbol{\beta}_0)$. There are many other extensions of the basic linear least squares method that may be useful to work with, which fit into the framework considered here. But this is not considered closer.

When working with dynamic systems we are interested in the changes in time in the solution vector. We shall look closer at the Kalman filter approach in finding the solutions. Let $\mathbf{X}_t$ be the instrumental data up to time t, $\mathbf{Y}_t$ the response values up to time t, $(\mathbf{x}_t,y_t)$ the sample values at time t, and $\mathbf{S}_t= \mathbf{X}_t^T\mathbf{X}_t+\mathbf{U}$. Then the solution at time t, $\mathbf{b}_{2,t}$, can be written as

$$\begin{aligned}\mathbf{b}_{2,t} &= (\mathbf{X}_{t-1}^T\mathbf{X}_{t-1}+\mathbf{U}+\mathbf{x}_t\,\mathbf{x}_t^T)^{-1}(\mathbf{X}_{t-1}^T\mathbf{Y}_{t-1} + \mathbf{x}_t\,y_t)\\ &= \mathbf{b}_{2,t-1} + \mathbf{k}_t(y_t - \mathbf{x}_t^T\,\mathbf{b}_{2,t-1}),\end{aligned}$$

with $\mathbf{k}_t= \mathbf{S}_{t-1}^{-1}\mathbf{x}_t/g_t$, and $g_t=1+\mathbf{x}_t^T\,\mathbf{S}_{t-1}^{-1}\mathbf{x}_t$.

This follows from the rewriting $\mathbf{X}_t^T\mathbf{X}_t= \mathbf{X}_{t-1}^T\mathbf{X}_{t-1}+ \mathbf{x}_t\,\mathbf{x}_t^T$, $\mathbf{X}_t^T\mathbf{Y}_t = \mathbf{X}_{t-1}^T\mathbf{Y}_{t-1} + \mathbf{x}_t\,y_t$, and the application of the matrix inversion lemma. This leads to *the Kalman filter equations*

1. Sample variance: $g_t=1+\mathbf{x}_t^T\,\mathbf{S}_{t-1}^{-1}\mathbf{x}_t$.
2. Kalman gain: $\mathbf{k}_t= \mathbf{S}_{t-1}^{-1}\mathbf{x}_t/g_t$.
3. Update the solution: $\mathbf{b}_{2,t} = \mathbf{b}_{2,t-1} + \mathbf{k}_t(y_t - \mathbf{x}_t^T\,\mathbf{b}_{2,t-1})$.
4. Update the inverse: $\mathbf{S}_t^{-1} = \mathbf{S}_{t-1}^{-1} - g_t\,\mathbf{k}_t\,\mathbf{k}_t^T$.

In these equations at time zero $\mathbf{S}_0=\mathbf{U}$. Otherwise the matrix $\mathbf{U}$ does not enter the equations. Apart from these equations there may be some further ones on requirements to the solution vector. When there are many variables the recursive updating equations tend to give unstable solutions. E.g., in the case of NIR instruments $\mathbf{S}$ would be 1050×1050. Even if we start with a diagonal $\mathbf{U}$, the updating becomes unstable because the difference matrix $\mathbf{S}$-$\mathbf{U}$ is typically of practical rank 3-6 for NIR data.

The present approach is concerned with finding stable solution in the case the data show low rank like in the case of NIR data. The algorithm proposed is independent of $\mathbf{U}$. Thus, $\mathbf{U}$ can be zero or any other prior choice. The solution is based on the H-principle of mathematical modeling that we shall consider closer.

3. The basic algorithm

H-principle is a recommendation of how we should carry out the modelling procedure for any mathematical model:

1) Carry out the modelling in steps. You specify how you want to look at the data at this step by formulating how the weights are computed.
2) At each step compute expressions for i) improvement in fit, ΔFit, and ii) the associated prediction, ΔPrecision
3) Compute the solution that maximizes the product ΔFit × ΔPrecision
4) In case the computed solution improves the prediction abilities of the model, the solution is accepted. If the solution does not provide this improvement, it stops.
5) The data is adjusted for what has been selected and start again at 1).

The main motivation for this approach is the prediction aspect of the model. When a new sample, $\mathbf{x}_0$, is available, the response values $\mathbf{Y}(\mathbf{x}_0)$ are estimated from the regression equation. The prediction variance for the estimated response values for a standard regression model is,

$$\mathrm{Var}(\mathbf{Y}(\mathbf{x}_0)) = [\mathbf{Y}^T(\mathbf{I}-\mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T)\mathbf{Y}]\times\mathbf{x}_0^T(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{x}_0/(N-K)$$

Assuming normal distribution, $(\mathbf{X}^T\mathbf{X})^{-1}$ and $[\mathbf{Y}^T(\mathbf{I}-\mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T)\mathbf{Y}]$ are stochastically independent, hence both must to be modelled, if we want to control the prediction variance.

The H-principle suggests that we should find a *weight vector* **w** that gives us a solution of step 3 (Höskuldsson (1996)). In the case of linear regression it is shown that the solution is given by the eigen vector of the leading eigen value to the eigen value problem,

$$\mathbf{X}^T\mathbf{Y}\mathbf{Y}^T\mathbf{X}\mathbf{w} = \lambda\mathbf{w}$$

In case there is only one response variable, $\mathbf{Y}=\mathbf{y}$, there is a closed form expression for **w**

$$\mathbf{w} = \mathbf{X}^T\mathbf{y}/|\mathbf{X}^T\mathbf{y}|.$$

The next task is to compute the *loading vector*, **p**, as $\mathbf{p}=\mathbf{S}\mathbf{w}$. The *score vector*, **t**, is defined as $\mathbf{t}=\mathbf{X}\mathbf{w}$. Besides these vectors we need one type more, the *transformation or causal vector* **v**. It is defined such that $\mathbf{p}=\mathbf{S}\mathbf{v}$. Finally a scaling constant is needed, d, where $d=1/\mathbf{w}^T\mathbf{S}\mathbf{w}$. These computations are carried out at each step. At the end of the computations the data is adjusted for what has been selected.

The algorithm is as follows:

```
0. Initialize variables. $\mathbf{X}_0=\mathbf{X}$, $\mathbf{S}_0=\mathbf{S}$, $\mathbf{Y}_0=\mathbf{Y}$, $\mathbf{E}_0=\mathbf{I}_K$, $\mathbf{B}=\mathbf{0}$.
For a=1,2, … , K,
1. Find the weight vector $\mathbf{w}_a$:
   $\mathbf{w}_a$ is the left eigen vector of $(\mathbf{X}^T\mathbf{Y})$ associated with the largest singular value.
2. Compute
   loading vector $\mathbf{p}_a$: $\mathbf{p}_a=\mathbf{S}_{a-1}\mathbf{w}_a$,
   score vector $\mathbf{t}_a=\mathbf{X}_{a-1}\mathbf{w}_a$,
   scaling constant $d_a$: $d_a=1/\mathbf{w}_a^T\mathbf{S}_{a-1}\mathbf{w}_a$.
3. The loading weight vectors $\mathbf{v}_a$:
   $\mathbf{v}_a=\mathbf{E}_{a-1}\mathbf{w}_a$,
   Adjust transformation matrix: $\mathbf{E}_a=\mathbf{E}_{a-1}-d_a\mathbf{v}_a\mathbf{p}_a^T$
4. Compute new solution coefficients $\mathbf{B}$:
   $\mathbf{q}_a=(\mathbf{X}^T\mathbf{Y})^T\mathbf{v}_a$.
   $\mathbf{B}_a = \mathbf{B}_{a-1} + d_a\mathbf{v}_a\mathbf{q}_a^T$,
5. Adjust $\mathbf{S}$: $\mathbf{S}_a=\mathbf{S}_{a-1} - d_a\,\mathbf{p}_a\mathbf{p}_a^T$.
6. Adjust covariance: $(\mathbf{X}^T\mathbf{Y})=(\mathbf{X}^T\mathbf{Y}) - d_a\,\mathbf{p}_a\mathbf{q}_a^T$.
7. Adjust $\mathbf{X}$: $\mathbf{X}_a=\mathbf{X}_{a-1}- d_a\,\mathbf{t}_a\mathbf{p}_a^T$.
8. Adjust $\mathbf{Y}$: $\mathbf{Y}_a=\mathbf{Y}_{a-1}- d_a\,\mathbf{t}_a\mathbf{q}_a^T$.
9. Check if this step has improved the prediction aspect of the model, and if $\lambda_a$ or $d_a$ are not too
   small. If it pays to continue, start a new iteration at 1.
```

Note that the steps 7. and 8. are not necessary. The score vectors can be computed afterwards as $\mathbf{T}=\mathbf{X}\mathbf{V}$. The covariance $(\mathbf{X}^T\mathbf{Y})$ need not be equal to $\mathbf{X}_{a-1}^T\mathbf{Y}_{a-1}$. The covariance is the right hand terms in the set of linear equations. See the MATLAB program in Appendix for the implementation of the algorithm.

The results of this algorithm are expansions of the matrices as follows:

$$\begin{aligned}
\mathbf{X} &= d_1\,\mathbf{t}_1\,\mathbf{p}_1^T + d_2\,\mathbf{t}_2\,\mathbf{p}_2^T + \ldots + d_A\mathbf{t}_A\mathbf{p}_A^T + \ldots + d_K\,\mathbf{t}_K\mathbf{p}_K^T &&= \mathbf{TDP}^T.\\
\mathbf{S} &= d_1\,\mathbf{p}_1\,\mathbf{p}_1^T + d_2\,\mathbf{p}_2\,\mathbf{p}_2^T + \ldots + d_A\,\mathbf{p}_A\mathbf{p}_A^T + \ldots + d_K\,\mathbf{p}_K\mathbf{p}_K^T &&= \mathbf{PDP}^T.\\
\mathbf{S}^{-1} &= d_1\,\mathbf{v}_1\,\mathbf{v}_1^T + d_2\,\mathbf{v}_2\,\mathbf{v}_2^T + \ldots + d_A\,\mathbf{v}_A\mathbf{v}_A^T + \ldots + d_K\,\mathbf{v}_K\mathbf{v}_K^T &&= \mathbf{VDV}^T.\\
\mathbf{B} &= d_1\,\mathbf{v}_1\,\mathbf{q}_1^T + d_2\,\mathbf{v}_2\,\mathbf{q}_2^T + \ldots + d_A\,\mathbf{v}_A\mathbf{q}_A^T + \ldots + d_K\,\mathbf{v}_K\mathbf{q}_K^T &&= \mathbf{VDQ}^T.
\end{aligned}$$

Here the vectors are collected in a matrix, e.g., $\mathbf{T}=(\mathbf{t}_1,\mathbf{t}_2,\ldots,\mathbf{t}_K)$. $\mathbf{D}$ is a diagonal matrix with d_a's in the diagonal. The decomposition of $\mathbf{S}$ is a rank one reduction, meaning that the rank of say, $\mathbf{S}_a$ is one less than that of $\mathbf{S}_{a-1}$. (Follows from $\mathbf{S}_a\mathbf{w}_a=\mathbf{0}$). Thus $\mathbf{S}_K$ will be the zero matrix. The matrix $\mathbf{V}$ satisfies $\mathbf{V}^T\mathbf{P}=\mathbf{D}^{-1}$, or $\mathbf{v}_i^T\mathbf{p}_j=\delta_{ij}/d_i$, where $\delta_{ij}=0$ for $i\neq j$ and 1 for $i=j$. It can also be written as $(\mathbf{VD}^{½})^T(\mathbf{PD}^{½})=\mathbf{I}$ or $\mathbf{VDP}^T=\mathbf{I}$. The score vectors ($\mathbf{t}_a$) are not orthogonal, $\mathbf{t}_i^T\mathbf{t}_j\neq 0$ for $i\neq j$.

The idea of the algorithm is simultaneously to decompose $\mathbf{S}$ and approximate $\mathbf{S}^{-1}$ with the aim to optimize the quality of the predictions derived from the model.

This algorithm is carried out for each time point *t*. Note, that if $\mathbf{U}=\mathbf{0}$, $\mathbf{S}=\mathbf{X}^T\mathbf{X}$ and the algorithm reduces to PLS regression. In that case the score vectors become orthogonal, $\mathbf{T}^T\mathbf{T}=\mathbf{D}^{-1}$.

We can view the algorithm as an approximation,

$$\mathbf{B} = \mathbf{S}^{-1}\,\mathbf{X}^T\mathbf{Y} = (d_1\,\mathbf{v}_1\,\mathbf{v}_1^T + \ldots)(\,d_1\,\mathbf{p}_1\,\mathbf{t}_1^T + \ldots)\mathbf{Y} = (d_1\,\mathbf{v}_1\,\mathbf{t}_1^T + \ldots)\mathbf{Y} = (d_1\,\mathbf{v}_1\,\mathbf{q}_1^T + \ldots).$$

Note that only *A* terms in the expansions are used. The choice of the weight vector $\mathbf{w}$ at each step reflects the covariance that is left. The expansion stops, when there is no covariance left, $\mathbf{X}_a^T\mathbf{Y}_a\cong\mathbf{0}$. When there are many variables, it is often necessary to be careful in finding the weight vector $\mathbf{w}$. A collection of methods has been developed that optimise the choice of $\mathbf{w}$ (Reinikainen et al. (2003)).

When the algorithm is applied, it is important to auto-scale the data, i.e., to centre data and scale to unit variance. The reason is that in step 3) in the recommendation of the H-principle, the measure used for improved fit and the associated precision should have the same unit of scales. If $\mathbf{C}_1$ is a diagonal matrix used for scaling the instrumental data, $\mathbf{S}_1=\mathbf{SC}_1$ and $\mathbf{X}_1=\mathbf{XC}_1$, and $\mathbf{C}_2$ is a diagonal matrix for the response data, $\mathbf{Y}_1=\mathbf{YC}_2$, the exact solution is derived as

$$\mathbf{B}_1=\mathbf{S}_1^{-1}\mathbf{X}_1^T\mathbf{Y}_1 = \mathbf{C}_1^{-1}\,[\mathbf{S}^{-1}\mathbf{X}^T\mathbf{Y}]\,\mathbf{C}_2 = \mathbf{C}_1^{-1}\,\mathbf{B}\,\mathbf{C}_2 \qquad \text{or} \quad \mathbf{B} = \mathbf{C}_1\,\mathbf{B}_1\,\mathbf{C}_2^{-1}$$

For the approximated solution the same scaling is used, $\mathbf{B}_A = \mathbf{C}_1\,\mathbf{B}_{1,A}\,\mathbf{C}_2^{-1}$. Note that the validation of the model, determination of the dimension *A*, etc is carried out for the scaled data.

The choice of $\mathbf{U}$ should take into consideration the choice of $\mathbf{C}_1$ and $\mathbf{C}_2$. The important issue is that the term $\boldsymbol{\beta}^T\mathbf{U}\boldsymbol{\beta}$ should have the same unit as the term $|\mathbf{X}\boldsymbol{\beta}-\mathbf{y}|^2$ for the scaled data.

The algorithm is numerically very stable. The reason is that the vectors $\mathbf{p}_a$, $\mathbf{t}_a$ and $\mathbf{v}_a$ are the range of a unit vector. Even though the steps 5. and 6. may turn out to be unstable for arbitrary data, they will not cause any numerical problems here, because the data is auto-scaled and the terms are well computed.

The algorithm works with four sets of vectors. The weight vectors $\mathbf{W}=(\mathbf{w}_1,\mathbf{w}_2,\ldots,\mathbf{w}_A)$ are found such that at each step the prediction of the model is optimized. The score vectors $\mathbf{T}=(\mathbf{t}_1,\mathbf{t}_2,\ldots,\mathbf{t}_A)$ show the variation in $\mathbf{X}$ that is used for the solution found. The loading vectors $\mathbf{P}=(\mathbf{p}_1,\mathbf{p}_2,\ldots,\mathbf{p}_A)$ show how $\mathbf{S}$ has been decomposed. Finally, the loading weight vectors $\mathbf{V}=(\mathbf{v}_1,\mathbf{v}_2,\ldots,\mathbf{v}_A)$ show how the loading vectors are derived from $\mathbf{S}$, $\mathbf{P}=\mathbf{SV}$. For further properties of the vectors see the appendix.

The MATLAB code for the basic algorithm.

```
% ----------- dynam -----------
% Data are assumed appropriately centred and scaled
%
%
% INPUT
% X      The instrumental data
% Y      The response values
% XY     Right hand side of a set of linear equations
%        Note that it need not be equal X^TY
% S      A positive semi-definite matrix
% A      Desired dimension. If A=0, A is set to K
% eps    Numerical precision
%
% OUTPUT
% Out.B  The solution vector/matrix
% Out.W  The weight vectors
% Out.P  The loading vectors
% Out.T  The score vectors
% Out.V  The loading vector vectors
% Out.Y  The reduced response matrix
% Out.X  The reduced instrumental matrix
% Out.Q  The matrix (XY)^TV
% Out.S  The reduced positive matrix
% Out.Si The estimated inverse of input S
% Out.D  The scaling constants
% Out.L  The singular values of XY at each step
% Out.Ns The dimension actually computed
%
% USAGE
%         [Out]=dynam(X, Y, XY, S, A, eps);
%
function [Out]=dynam(X, Y, XY, S, A, eps);
%
% ---------------------------------------------------------%
Initialisation of matrices
Out=struct('B',[],'W',[],'P',[],'T',[],'V',[],'Y',[], …
   'X',[], 'Q',[], 'S',[], 'Si',[], 'D',[], 'L',[],Ns',[]);
[K,K1]=size(S);
[K2,M]=size(XY);
if A<=0
    A=K;
end
B=zeros(K,M);
Si=zeros(K,K);
E=eye(K);
W=[];P=[];T=[];V=[];Q=[];D=[];L=[];
Ns=0;
%
% ---------------------------------------------------------%
for i=1:A
    [U1,S1,V1] = svd(XY,0);
    w=U1(:,1);    % The weight vector
    lambda=S1(1,1);% The singular values
%             Other types of weight vector can be computed
%                  without changing the other code
    p=S*w;        % The loading vector
    d=w'*p;       % =1/w’Sw
    if d<eps      % May not be too small
       break;     % Stop analysis
    else
       d=1/d;     % The scaling constant
    end
    v=E*w;        % Transformation vector
    t=X*w;        % The score vector
    q=XY'*v;      % Inner product of XY and v
%  ------------    All necessary vectors have been computed
    Ns=Ns+1;      % Count the dimension
    E=E-d*v*p';   % Adjust transformation matrix E
    B=B+d*v*q';   % Solution vector at this step
    X=X-d*t*p';   % Reduce X
    S=S-d*p*p';   % Reduce S
    Y=Y-d*t*q';   % Reduce Y
    XY=XY-d*p*q'; % Reduce the covariance XY
    Si=Si+d*v*v'; % Approximate the inverse
%  ------------   All necessary matrices updated
    W=[W w];      % Store w
    P=[P p];      % Store p
    T=[T t];      % Store t
    V=[V v];      % Store v
    Q=[Q q];      % Store q
    D=[D d];      % Store d
    L=[L lambda]; % Store lambda
end
%  -----------            Store results
Out.B=B;
Out.W=W;
Out.P=P;
Out.T=T;
Out.V=V;
Out.Y=Y;
Out.X=X;
Out.Q=Q;
Out.S=S;
Out.Si=Si;
Out.D=D;
Out.L=L;
Out.Ns=Ns;
%
%
% -------------   Numerical results can be checked
%           by removing the comment sign % in column 1:
%D1=inv(diag(D));
%U1=W'*P-D1       % Lower triangular with zeros in diagonal
%O1=V'*P-D1       % The zero matrix
%I1=W'*W          % Identity matrix
% If all dimensions are selected:
% X               % is zeros
% S               % is zeros
% Si              % The inverse of S at entry
% E               % is zeros
% O2=T-(T*diag(D)*P’)*V % is zeros
% X0=T*diag(D)*P’ % Equal to X at entry
```