Algorithms for the Vold-Kalman multiorder tracking filter

Jiří Tůma

VSB – Technical University of Ostrava
Ostrava, Czech republic
Vold-Kalman tracking filter?

Noise at Car Exhaust System

Harmonics of Engine Rotational Speed
Motivation: separation of crossing orders

Rotational speed of both rotating engine components are not synchronized, which causes the problem of how to analyze cross orders.

Rotational speed of a turbocharger

Rotational speed of an engine
Outline

- Principles of Kalman filtration
- Principles of Vold-Kalman filtering – algorithm
- Global solution
- Single / multi-components filtration
- Example of employing the Vold-Kalman order tracking filtration
Kalman filter

Process equation
\[ \mathbf{x}(n) = A \mathbf{x}(n-1) + \mathbf{v}_1(n) \]

Model of the time evolution of the measured signal

Measurement equation
\[ \mathbf{y}(n) = H \mathbf{x}(n) + \mathbf{v}_2(n) \]

Model of corruption of the measured signal by random noise (errors)

\[ \mathbf{y}(n) \] … measured signal
\[ \mathbf{x}(n) \] … state variable
\[ \mathbf{v}_1(n) \] … uncorrelated excitation vector of the process equation
\[ \mathbf{v}_2(n) \] … uncorrelated excitation vector vector of the measurement equation

Input parameters:  
• covariance matrices of \( \mathbf{v}_1(n) \) and \( \mathbf{v}_2(n) \)  
• matrices A, C, H
Principle of the Vold-Kalman filter

Data equation (equivalent to the measurement equation)
\[ y(n) = x(n) + \eta(n) \]
- measured signal
- error term 1
- harmonic signal of a specified frequency \( \omega(n) \)
- filter output

Matrix form of equations
\[ y - x = \eta \]

Square of the error vector norm
\[ \eta^T \eta = (y^T - x^T)(y - x) \]

Structural equation (equivalent to the process equation)
\[ x(n) - 2 \cos(\omega \Delta t) x(n - 1) + x(n - 2) = \epsilon(n) \]
- filter output
- error term 2
- matrix form of equations
\[ A x = \epsilon \]

Square of the error vector norm
\[ \epsilon^T \epsilon = x^T A^T A x \]

Objectives:
- a weighted sum of squares
\[ J = r^2 \epsilon^T \epsilon + \eta^T \eta \rightarrow \min \]
- a weighted balance between the error terms
- \( r \) – weighting coefficient
Data equation (equiv. to measurement equation)

**First generation**

\[ y(n) = x(n) + \eta(n) \]

\( n = 1,2, \ldots, N \)

- \( y(n) \) – measured signal
- \( \eta(n) \) – error term
- \( \omega(n) \) – angular frequency
- \( x(n) \) – filter output

**Matrix form of equations**

\[ y - x = \eta \]

Square of the error vector norm

\[ \eta^H \eta = (y^T - x^T)(y - x) \]

**Second generation**

\[ y(n) = x(n) \exp(j\Theta(n)) + \eta(n) \]

\[ \Theta(n) = \sum_{i=0}^{n} \omega(i)\Delta t \]

\( x(n) \) is a complex envelope

\[ C = \text{diag}\{\exp(j\Theta(1)), \ldots, \exp(j\Theta(N))\} \]

**Matrix form of equations**

\[ y - Cx = \eta \]

Square of the error vector norm

\[ \eta^H \eta = (y^T - x^H C^H)(y - Cx) \]
Structural equation (equiv. to process equation)

First generation

\[ x(n) - 2 \cos(\omega \Delta t)x(n-1) + x(n-2) = \varepsilon(n) \]

\( \omega \) – rotational speed (interpolated), \( c(n) = 2 \cos(\omega \Delta t) \)

\( x(n) \) – filtered signal, \( \varepsilon(n) \) – error term, \( N \) – sample number

Second generation

\[ x(n) - x(n-1) = \varepsilon(n) \]
\[ x(n) - 2x(n-1) + x(n-2) = \varepsilon(n) \]
\[ x(n) - 3x(n-1) + 3x(n-2) - x(n-3) = \varepsilon(n) \]

Matrix form of equations

\[ A \mathbf{x} = \varepsilon \]

Square of the error vector norm

\[ \varepsilon^T \varepsilon = \mathbf{x}^T A^T A \mathbf{x} \]
Global solution

Objectives:

\[ J = r^2 \epsilon^T \epsilon + \eta^T \eta \rightarrow \min \]

\( r \) – weighting coefficient

Solution:

First generation

\[
\frac{\partial J}{\partial x} = 2 r^2 A^T A x + 2 (x - y) = 0
\]

\[
x = (r^2 A^T A + E)^{-1} y
\]

Second generation

\[
\frac{\partial J}{\partial x^H} = (r^2 A^T A + E) x - C^H y = 0
\]

\[
x = (r^2 A^T A + E)^{-1} C^H y
\]

\[ B = r^2 A^T A + E \quad \ldots \quad \text{SPD} - \text{Symmetric Positive Definite matrix} \]
Multiorder tracking filter

Data equations for extraction of $P$ components

**First generation**

$$y(n) = \sum_{i=1}^{P} x_i(n) + \eta(n)$$

**Second generation**

$$y(n) = \sum_{i=1}^{P} x_i(n) \exp(j\Theta_i(n)) + \eta(n)$$

**Global solution**

$$\frac{\partial J}{\partial \mathbf{x}_i^H} = \mathbf{B}_i \mathbf{x}_i + \mathbf{C}_i^H \sum_{k=1}^{P} \mathbf{C}_k \mathbf{x}_k - \mathbf{C}_i^H \mathbf{y} = 0, \quad i = 1, ..., P$$

$P$ matrix blocks

$P \times P$ – block matrix

$$\begin{bmatrix}
\mathbf{B}_1 & \mathbf{C}_1^H \mathbf{C}_2 & \ldots & \mathbf{C}_1^H \mathbf{C}_P \\
\mathbf{C}_2^H \mathbf{C}_1 & \mathbf{B}_2 & \ldots & \mathbf{C}_2^H \mathbf{C}_P \\
\vdots & \vdots & \ddots & \vdots \\
\mathbf{C}_P^H \mathbf{C}_1 & \mathbf{C}_P^H \mathbf{C}_2 & \ldots & \mathbf{B}_P
\end{bmatrix} \times
\begin{bmatrix}
\mathbf{x}_1 \\
\mathbf{x}_2 \\
\vdots \\
\mathbf{x}_P
\end{bmatrix} =
\begin{bmatrix}
\mathbf{C}_1^H \mathbf{y} \\
\mathbf{C}_2^H \mathbf{y} \\
\vdots \\
\mathbf{C}_P^H \mathbf{y}
\end{bmatrix}$$
Large-scale system of linear equations

Iterative methods for sparse linear systems of Symmetric Positive Definite matrices

$$Bx = b$$

Preconditioned Conjugate Gradients method

$$BM^{-1}u = b, \quad x = M^{-1}u \quad (M \text{ precon}$$

Iterative method Direct method

MATLAB

```matlab
x = pcg(B,b)
x = pcg(B,b,tol)
x = pcg(B,b,tol,maxit)
x = pcg(B,b,tol,maxit,M)
x = pcg(B,b,tol,maxit,M1,M2)
x = pcg(B,b,tol,maxit,M1,M2,x0)
```

Preconditioner matrix

$$M = \text{diag}(B_1,B_2,\cdots,B_P)$$

$$M^{-1} = \text{diag}(B_1^{-1},B_2^{-1},\cdots,B_P^{-1})$$

Initial guess of the unknown variable

$$b = \begin{bmatrix} C_1^H y & C_2^H y & \cdots & C_P^H y \end{bmatrix}^T$$

$$x_0 = M^{-1}b.$$}

$$x_{0,i} = B_i^{-1}C_i^H y, \quad i = 1,\cdots,P.$$
**MyPCG1**

<table>
<thead>
<tr>
<th></th>
<th>(\text{Compute: } r_0 = b - Bx_0,\ z_0 = M^{-1}r_0,\ p_0 = z_0)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>(\text{For } m = 0, 1, ..., \text{ until convergence Do:})</td>
</tr>
<tr>
<td>3</td>
<td>(\alpha_m = \frac{(r_m, z_m)}{(Bp_m, p_m)})</td>
</tr>
<tr>
<td>4</td>
<td>(x_{m+1} = x_m + \alpha_m p_m)</td>
</tr>
<tr>
<td>5</td>
<td>(r_{m+1} = r_m - \alpha_m Bp_m)</td>
</tr>
<tr>
<td>6</td>
<td>(z_{m+1} = M^{-1}r_{m+1})</td>
</tr>
<tr>
<td>7</td>
<td>(\beta_m = \frac{(r_{m+1}, z_{m+1})}{(r_m, z_m)})</td>
</tr>
<tr>
<td>8</td>
<td>(p_{m+1} = z_{m+1} + \beta_m p_m)</td>
</tr>
<tr>
<td>9</td>
<td>(\text{EndDo})</td>
</tr>
</tbody>
</table>

**MyPCG2**

<table>
<thead>
<tr>
<th></th>
<th>(\text{Compute: } r_0 = b - Bx_0,\ \tilde{r}_0 = L^{-1}r_0,\ p_0 = L^{-T}\tilde{r}_0)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>(\text{For } m = 0, 1, ..., \text{ until convergence Do:})</td>
</tr>
<tr>
<td>3</td>
<td>(\alpha_m = \frac{(\tilde{r}_m, \tilde{r}_m)}{(Bp_m, p_m)})</td>
</tr>
<tr>
<td>4</td>
<td>(x_{m+1} = x_m + \alpha_m p_m)</td>
</tr>
<tr>
<td>5</td>
<td>(\tilde{r}_{m+1} = \tilde{r}_m - \alpha_m L^{-1}Bp_m)</td>
</tr>
<tr>
<td>6</td>
<td>(\beta_m = \frac{(\tilde{r}<em>{m+1}, \tilde{r}</em>{m+1})}{(\tilde{r}_m, \tilde{r}_m)})</td>
</tr>
<tr>
<td>7</td>
<td>(p_{m+1} = L^{-T}\tilde{r}_{m+1} + \beta_m p_m)</td>
</tr>
<tr>
<td>8</td>
<td>(\text{EndDo})</td>
</tr>
</tbody>
</table>
Example 1

The sum of two harmonic signals with amplitude equal to the unit

1000 samples * components = 2000 equations
Example 2

Time History

Multispectrum

Frequency
PULSE – Without Decoupling Orders

[V]

Vold-Kalman Order Filter Tacho1 (Response) (Magnitude)
Working: Input: Input: Time Capture Analyzer

Beats

Time

[V]

Vold-Kalman Order Filter Tacho2 (Response) (Magnitude)
Working: Input: Input: Time Capture Analyzer

Beats

Time

[Order] (Nominal Values)
Decoupling Orders

Pulse

MATLAB
Conclusion

Vold-Kalman order tracking filtration is a tool for conducting diagnostics of rotating machines

- **Advantages**
  - tracking an order without slew rate limitations
  - decoupling of close and crossing orders
  - stepwise changes of the RPM

- **Disadvantages**
  - non real time processing
  - longer calculation time
  - some prior knowledge about signal is required.
Thank you for your attention

http://homel.vsb.cz/~tum52
Software for Vold-Kalman Order Filtration

Second generation only

- Brüel & Kjær, LabShop PULSE, Software Type 7703
- MTS Systems Corporation, I-DEAS

First and Second generation

- VSB – Technical University of Ostrava
  - M-functions in MATLAB including crossing orders (open code)
  - Signal Analyzer (VB6 - without crossing orders)
- Axiom-EduTech Sweden & VSB – TU Ostrava
  - M-functions in MATLAB (open code)
MATLAB functions

First generation

function x = MyVoldKalman1(y,dt,f,r)

c  = 2*cos(2*pi*f*dt);
N  = max(size(y)); N2 = N-2;
e  = ones(N2,1);
A  = spdiags([e -2*c(1:N2) e],0:2,N2,N);
AA  = r*r*A'*A +speye(N,N);
x   = AA\y;

Second generation

function x = MyVoldKalman2(y,dt,f,r,filtord)

N  = max(size(y));
if filtord==1, NR = N-2; else NR = N-3; end;
e  = ones(NR,1);
if filtord==1,
    A = spdiags([e -2*e e],0:2,NR,N);
else
    A = spdiags([e -3*e 3*e -e],0:3,NR,N);
end;
AA = r*r*A'*A +speye(N); yy = exp(-j*2*pi*cumsum(f)*dt).*y;
x  = 2*AA\yy;

Sparse matrix functions

speye – identity matrix
spdiags – diagonal matrix
\ - left matrix divide