The Influence Diagram Form Implementation of Multivariate Gaussian Distributions

C. Robert Kenley
Purdue University
315 N. Grant St.
West Lafayette, IN, United States 47907
kenley@purdue.edu

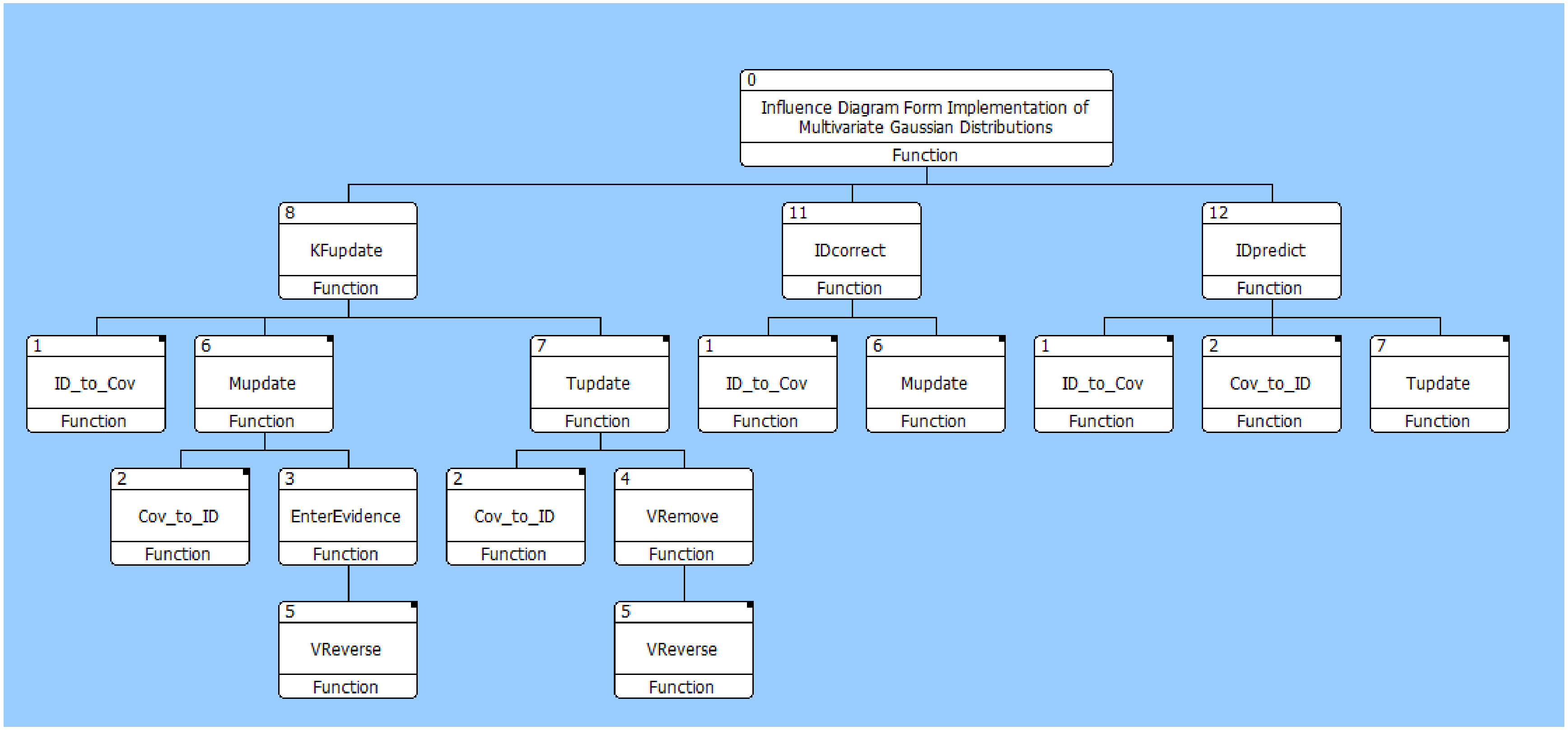

Figure 1.Hierarchy of Algorithms (the numbers refer to the tables in this document)

1 DIFFERENT FORMS FOR REPRESENTING MULTIVARIATE GAUSSIAN DISTRIBUTIONS

1.1 COVARIANCE FORM

The covariance form is represented by the collection of objects [$\boldsymbol{\mu}$, $\boldsymbol{\Sigma}$] as follows:
$\boldsymbol{\mu}$ = an $n \times 1$ vector; and
$\boldsymbol{\Sigma}$ = is an $n \times n$ positive-semidefinite symmetric matrix.

1.2 INFLUENCE DIAGRAM

The influence diagram (ID) is represented by the collection of objects [$\boldsymbol{\mu}$, $\mathbf{B}$, $\mathbf{v}$] as follows:
$\boldsymbol{\mu}$ is an $n \times 1$ vector;
$\mathbf{B}$ = is an $n \times n$ matrix, which is strictly upper triangular; and
$\mathbf{v}$ is an $n \times 1$ vector with entries that are non-negative (including infinity).

2 CONVERSIONS BETWEEN THE FORMS

2.1 CONVERTING INFLUENCE DIAGRAM FORM TO COVARIANCE FORM

Let $\mathbf{x}$ be a continuous random variable of dimension n with ID form given by vector $\boldsymbol{\mu}$, a strictly upper triangular matrix $\mathbf{B}$, and vector $\mathbf{v}$. The covariance form values of $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ are calculated using the algorithm in Table 1.

Table 1 Algorithm for Converting from Influence Diagram Form to Covariance Form

```
Procedure ID_to_Cov (B, v, Σ)
Σ(1,1) = v(1)
Do i = 2, …, n
        Do j = 1, …, i-1
                Σ(i, j) = 0
                Do k = 1,…, i -1
                        If Σ( j, k) ≠ ∞ then
                                Σ( i, j) = Σ( i, j) + Σ(j,k) B(k, i)
                        End
                End; end of k loop
                Σ( j, i) = Σ( i, j)
        End; end of j loop
        If v(i) =∞ then
                Σ( i, i) = ∞
        Else
```

$$\Sigma(i,\mathrm{i}) = \mathbf{v}(i) + \begin{pmatrix} \Sigma(i,1) \\ \vdots \\ \Sigma(i,\mathrm{i}-1) \end{pmatrix} \cdot \begin{pmatrix} B(1,\mathrm{i}) \\ \vdots \\ B(i-1,\mathrm{i}) \end{pmatrix}$$

```
        End If
End; end of i loop
```

2.2 CONVERTING COVARIANCE FORM TO INFLUENCE DIAGRAM FORM

Let **x** be a continuous random variable of dimension *n* with covariance form **μ**, **Σ**. The values for **μ**, **B**, and **v** are determined using the algorithm in Table 2.

Table 2 Algorithm for Converting Covariance Form to Influence Diagram Form

Procedure Cov_to_ID (**Σ, B**, **v**, **P**)
Initialize **P, B,** and **v** as all 0's using dimension of **Σ**
B(1,1) = 0
$v(1) = \Sigma(1,1)$
If (v(1) = 0) or (v(1) = ∞), then
    P(1,1) = 0
Else
    P(1,1) = *1* / v(1)
End
If n >=2 then
    Do *j* = 2, n
        B(*j*,*j*) = 0

$$\begin{pmatrix} B(j,1) \\ \vdots \\ B(j,\mathrm{j}-1) \end{pmatrix} = \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix}$$

$$\begin{pmatrix} B(1,j) \\ \vdots \\ B(j-1,\mathrm{j}) \end{pmatrix} = \begin{bmatrix} \boldsymbol{P}(1,1) & \cdots & \boldsymbol{P}(1,\mathrm{j}-1) \\ \vdots & \ddots & \vdots \\ \boldsymbol{P}(\mathrm{j}-1,1) & \cdots & \boldsymbol{P}(\mathrm{j}-1,\mathrm{j}-1) \end{bmatrix} \begin{bmatrix} \Sigma(1,\mathrm{j}) \\ \vdots \\ \Sigma(j-1,\mathrm{j}) \end{bmatrix}$$

        If **Σ**(*j*,*j*) = ∞
            v(*j*) = ∞
        Else

$$v(\mathrm{j}) = \Sigma(j,\mathrm{j}) - \begin{pmatrix} \Sigma(j,1) \\ \vdots \\ \Sigma(j,\mathrm{j}-1) \end{pmatrix} \cdot \begin{pmatrix} B(1,\mathrm{j}) \\ \vdots \\ B(j-1,\mathrm{j}) \end{pmatrix}$$

            v(*j*) = max{ **v**(*j*), 0}
        End If
        If (v(*j*) = 0) or (v(*j*) = ∞), then
            P(*j*,*j*) = 0

$$\begin{pmatrix} \boldsymbol{P}(j,1) \\ \vdots \\ \boldsymbol{P}(j,\mathrm{j}-1) \end{pmatrix} = \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix}$$

$$\begin{pmatrix} \boldsymbol{P}(1,j) \\ \vdots \\ \boldsymbol{P}(j-1,\mathrm{j}) \end{pmatrix} = \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix}$$

        Else
            P(*j*,*j*) = *1* / v(*j*)
            Do *k* = *1*, …, *j-1*

temp = **P**(j,j) **B**(k,j)
If k ≠ 1

$$\begin{pmatrix} \boldsymbol{P}(k,1) \\ \vdots \\ \boldsymbol{P}(k,\mathrm{k}-1) \end{pmatrix} = \begin{pmatrix} \boldsymbol{P}(k,1) \\ \vdots \\ \boldsymbol{P}(k,\mathrm{k}-1) \end{pmatrix} + temp \begin{pmatrix} B(1,\mathrm{j}) \\ \vdots \\ B(\mathrm{k}-1,\mathrm{j}) \end{pmatrix}$$

$$\begin{pmatrix} \boldsymbol{P}(1,\mathrm{k}) \\ \vdots \\ \boldsymbol{P}(k-1,\mathrm{k}) \end{pmatrix} = \begin{pmatrix} \boldsymbol{P}(k,1) \\ \vdots \\ \boldsymbol{P}(k,\mathrm{k}-1) \end{pmatrix}$$

End If
P(k,k) = P(k,k) + temp * **B**(k, j)
End; end of k loop

$$\begin{pmatrix} \boldsymbol{P}(j,1) \\ \vdots \\ \boldsymbol{P}(j,\mathrm{j}-1) \end{pmatrix} = -\boldsymbol{P}(j,\mathrm{j}) \begin{pmatrix} B(1,\mathrm{j}) \\ \vdots \\ B(j-1,\mathrm{j}) \end{pmatrix}$$

$$\begin{pmatrix} \boldsymbol{P}(1,\mathrm{j}) \\ \vdots \\ \boldsymbol{P}(j-1,\mathrm{j}) \end{pmatrix} = \begin{pmatrix} \boldsymbol{P}(j,1) \\ \vdots \\ \boldsymbol{P}(j,\mathrm{j}-1) \end{pmatrix}$$

End if
End do; end of j loop
End If ; end of n>= 2

3 BASIC OPERATIONS ON THE INFLUENCE DIAGRAM FORM

3.1 ENTERING EVIDENCE (INSTANTIATION)

Let $\mathbf{x}_1$ be a vector of n_1 values with evidence in a multivariate Gaussian with ID form ($\boldsymbol{\mu}$, **B**, **v**) with domain index set $\{1,\ldots,n\}$ that is ordered so that **B** is upper triangular. Let $\mathbf{x}_0$ be a vector of n_0 variables that are the predecessors of $\mathbf{x}_1$**;** and $\mathbf{x}_2$, a vector of n_2 variables that are the successors of $\mathbf{x}_1$.

Partition the domain using the index subset from $\{1,\ldots n\}$ into three ordered subsets so that

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{x_0} \\ \boldsymbol{x_1} \\ \boldsymbol{x_2} \end{bmatrix},\ \boldsymbol{x_0} = \begin{bmatrix} \boldsymbol{x_0}(1) \\ \vdots \\ \boldsymbol{x_0}(n_0) \end{bmatrix},\ \boldsymbol{x_1} = \begin{bmatrix} \boldsymbol{x_1}(1) \\ \vdots \\ \boldsymbol{x_1}(n_1) \end{bmatrix},\ \text{and } \boldsymbol{x_2} = \begin{bmatrix} \boldsymbol{x_2}(1) \\ \vdots \\ \boldsymbol{x_2}(n_2) \end{bmatrix}$$

and the partitioned ID form representation is

$$\boldsymbol{\mu} = \begin{bmatrix} \boldsymbol{\mu_0} \\ \boldsymbol{\mu_1} \\ \boldsymbol{\mu_2} \end{bmatrix},\ \boldsymbol{v} = \begin{bmatrix} \boldsymbol{v_0} \\ \boldsymbol{v_1} \\ \boldsymbol{v_2} \end{bmatrix}, \text{and } \boldsymbol{B} = \begin{bmatrix} \boldsymbol{B_{00}} & \boldsymbol{B_{01}} & \boldsymbol{B_{02}} \\ \boldsymbol{0} & \boldsymbol{B_{11}} & \boldsymbol{B_{12}} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{B_{22}} \end{bmatrix}.$$

After entering evidence for $\boldsymbol{x_1} = \begin{bmatrix} \boldsymbol{x_1}(1) \\ \vdots \\ \boldsymbol{x_1}(n_1) \end{bmatrix}$**,** the ID form representation is updated with the data for the observed variables set to zero as appears in the form below.

$$\boldsymbol{\mu} = \begin{bmatrix} \boldsymbol{\mu_0} \\ \boldsymbol{0} \\ \boldsymbol{\mu_2} \end{bmatrix},\ \boldsymbol{v} = \begin{bmatrix} \boldsymbol{v_0} \\ \boldsymbol{0} \\ \boldsymbol{v_2} \end{bmatrix}, \text{and } \boldsymbol{B} = \begin{bmatrix} \boldsymbol{B_{00}} & \boldsymbol{0} & \boldsymbol{B_{02}} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{B_{22}} \end{bmatrix}$$

The algorithm for entering evidence is given in

Table 3. The algorithm modifies the scalar evidence update description from Shachter and Kenley (1989, 531-532) and Kenley (1986, 66) by extending the evidence set to a vector.

Table 3 Algorithm to Enter Evidence in Influence Diagram

Procedure EnterEvidence($\mathbf{x}_1$,n_0,n_1,n_2,$\boldsymbol{\mu}$,$\boldsymbol{v}$,$\boldsymbol{B}$)

Do $j = 1,\ldots n_1$; loop through the observed variables in forward order

First, perform the Arc Reversal Between Vector Nodes (see section 3.3) with algorithm input parameters (n_1, n_2, n_3, n_4) set to (0, n_0 + j-1, 1, n_1 - j + n_2) to reverse the arcs from $\mathbf{x}_0$ and from $\mathbf{x}_1(1)$,, $\mathbf{x}_1$(j-1) to $\mathbf{x}_1$(j)

Call $V_{reverse}(\mu, \boldsymbol{B}, \boldsymbol{v}, 0, n_0 + j - 1, 1, n_1 - j + n_2)$

$\Delta\mu(n_0 + j) = \mathbf{x}_1(j) - \mu(n_0 + j)$

$$\begin{pmatrix} \boldsymbol{\Delta\mu}(1) \\ \vdots \\ \boldsymbol{\Delta\mu}(n_0) \end{pmatrix} = \boldsymbol{\Delta\mu}(n_0 + \mathrm{j}) \begin{pmatrix} B(n_0 + j, 1) \\ \vdots \\ B(n_0 + j, n_0) \end{pmatrix}$$
; calculate the direct effect of the observed variable on the mean of $\mathbf{x}_0$

If $n_0 + n_1 + n_2 >= n_0 + \mathrm{j} + 1$

$$\begin{pmatrix} \boldsymbol{\Delta\mu}(n_0 + \mathrm{j} + 1) \\ \vdots \\ \boldsymbol{\Delta\mu}(n_0 + n_1 + n_2) \end{pmatrix} = \boldsymbol{\Delta\mu}(n_0 + \mathrm{j}) \begin{pmatrix} B(n_0 + j, n_0 + \mathrm{j} + 1) \\ \vdots \\ B(n_0 + j, n_0 + n_1 + n_2) \end{pmatrix}$$
; calculate the direct effect of the observed variable on the mean of $\mathbf{x}_1$(j+1),…, $\mathbf{x}_1(n_1)$ and on the mean of $\mathbf{x}_2$

End If

If $n_0 >= 2$

Do $k = 2,\ldots,n_0$; calculate the indirect effect of the observed variable on the mean of $\mathbf{x}_0(2),\ldots, \mathbf{x}_0(n_0)$

$$\Delta\mu(k) = \Delta\mu(k) + \begin{pmatrix} B(1, k) \\ \vdots \\ B(\mathrm{k} - 1, \mathrm{k}) \end{pmatrix} \cdot \begin{pmatrix} \boldsymbol{\Delta\mu}(1) \\ \vdots \\ \boldsymbol{\Delta\mu}(k - 1) \end{pmatrix}$$

End do

End If

$$\begin{pmatrix} \boldsymbol{\mu}(1) \\ \vdots \\ \boldsymbol{\mu}(n_0) \end{pmatrix} = \begin{pmatrix} \boldsymbol{\mu}(1) \\ \vdots \\ \boldsymbol{\mu}(n_0) \end{pmatrix} + \begin{pmatrix} \boldsymbol{\Delta\mu}(1) \\ \vdots \\ \boldsymbol{\Delta\mu}(n_0) \end{pmatrix}$$
; update mean of $\mathbf{x}_0$

If $n_1 + n_2 >=$ j+1

Do $k = n_0$ + j+1,…,$n_0 + n_1 + n_2$; calculate the indirect effect of the observed variable on the mean of $\mathbf{x}_1$(j+1),…, $\mathbf{x}_1(n_1)$ and on the mean of $\mathbf{x}_2$

$$\Delta\mu(k) = \Delta\mu(k) + \begin{pmatrix} B(1, k) \\ \vdots \\ B(n_0, \mathrm{k}) \end{pmatrix} \cdot \begin{pmatrix} \boldsymbol{\Delta\mu}(1) \\ \vdots \\ \boldsymbol{\Delta\mu}(n_0) \end{pmatrix}$$

$$\Delta\mu(k) = \Delta\mu(k) + \begin{pmatrix} B(n_0 + \mathrm{j} + 1, \mathrm{k}) \\ \vdots \\ B(n_0 + n_1 + n_2, k) \end{pmatrix} \cdot \begin{pmatrix} \boldsymbol{\Delta\mu}(n_0 + \mathrm{j} + 1) \\ \vdots \\ \boldsymbol{\Delta\mu}(n_0 + n_1 + n_2) \end{pmatrix}$$

$\mu(k) = \mu(k) + \Delta\mu(k)$

End do

End If

Zero the terms for the observed variable
$\mu(n_0 + j) = 0$
$v(n_0 + j) = 0$
$$\begin{pmatrix} B(n_0 + \mathrm{j}, 1) \\ \vdots \\ B(n_0 + \mathrm{j}, n_0 + n_1 + n_2) \end{pmatrix} = \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix}$$
End do; end of j loop

3.2 NODE REMOVAL FOR VECTOR NODES

Let $\mathbf{x} = (x_1,\ldots,x_n)^T$ be a multivariate Gaussian random variable with ID form representation ($\boldsymbol{\mu}$, $\mathbf{B}$, $\mathbf{v}$). Partition the domain using the index subset from $\{1,\ldots n\}$ into three ordered subsets so that
$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{x_0} \\ \boldsymbol{x_1} \\ \boldsymbol{x_2} \end{bmatrix},\ \boldsymbol{y_0} = \begin{bmatrix} \boldsymbol{x_0}(1) \\ \vdots \\ \boldsymbol{x_0}(n_0) \end{bmatrix},\ \boldsymbol{x_1} = \begin{bmatrix} \boldsymbol{x_1}(1) \\ \vdots \\ \boldsymbol{x}(n_1) \end{bmatrix},\text{ and } \boldsymbol{x_2} = \begin{bmatrix} \boldsymbol{x_2}(1) \\ \vdots \\ \boldsymbol{x_2}(n_2) \end{bmatrix}$$
and the partitioned ID form representation is
$$\boldsymbol{\mu} = \begin{bmatrix} \boldsymbol{\mu_0} \\ \boldsymbol{\mu_1} \\ \boldsymbol{\mu_2} \end{bmatrix},\ \boldsymbol{v} = \begin{bmatrix} \boldsymbol{v_0} \\ \boldsymbol{v_1} \\ \boldsymbol{v_2} \end{bmatrix},\text{ and } \boldsymbol{B} = \begin{bmatrix} \boldsymbol{B_{00}} & \boldsymbol{B_{01}} & \boldsymbol{B_{02}} \\ \boldsymbol{0} & \boldsymbol{B_{11}} & \boldsymbol{B_{12}} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{B_{22}} \end{bmatrix}.$$
After removal of $\boldsymbol{x_1} = \begin{bmatrix} \boldsymbol{x_1}(1) \\ \vdots \\ \boldsymbol{x_1}(n_1) \end{bmatrix}$**,** the ID form representation is updated with the data for the removed variables set to zero as appears in the form below.
$$\boldsymbol{\mu} = \begin{bmatrix} \boldsymbol{\mu_0} \\ \boldsymbol{0} \\ \boldsymbol{\mu_2} \end{bmatrix},\ \boldsymbol{v} = \begin{bmatrix} \boldsymbol{v_0} \\ \boldsymbol{0} \\ \boldsymbol{v_2} \end{bmatrix},\text{ and } \boldsymbol{B} = \begin{bmatrix} \boldsymbol{B_{00}} & \boldsymbol{0} & \boldsymbol{B_{02}} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{B_{22}} \end{bmatrix}$$
.

The procedure for influence diagram removal of $\mathbf{x_1}$ into $\mathbf{x_2}$ is shown in Table 4**.** It is based on Kenley (1986, 100-101).

Table 4 Algorithm for Node Removal of Vector Nodes in Gaussian Influence Diagrams

Procedure Vremove($\mu, \boldsymbol{B}, \boldsymbol{v}, n_0, n_1, n_2$)

First $\mathbf{x}_1$ is reversed with all of $\mathbf{x}_2(1)$**,…,**, $\mathbf{x}_2(n_2\text{-}1)$ *i*.e. the reversal algorithm (see section 3.3) is first done with parameters (n_0, n_1, n_2, n_3) set to (n_0, n_1, n_2-1, 0) and then removing each component of $\mathbf{x}_1$ into $\mathbf{x}_2(n_2)$.

If $n_2 > 1$
  Call $V_{reverse}(\mu, \boldsymbol{B}, \boldsymbol{v}, n_0, n_1, n_2 - 1, 0)$
End If
N = $n_0 + n_1 + n_2$; index of $x_2(n_2)$
Do i = $n_0 + n_1$, $n_0 + 1$; in reverse order, remove each component of $\mathbf{x}_1$ into $\mathbf{x}_2(n_2)$
  If $n_0 >= 1$

$$\begin{pmatrix} B(1,\mathrm{N}) \\ \vdots \\ B(n_0,\mathrm{N}) \end{pmatrix} = \begin{pmatrix} B(1,\mathrm{N}) \\ \vdots \\ B(n_0,\mathrm{N}) \end{pmatrix} + B(\mathrm{i},\mathrm{N}) \begin{pmatrix} B(1,\mathrm{i}) \\ \vdots \\ B(n_0,\mathrm{i}) \end{pmatrix}$$; update arcs from x_0 to $x_2(n_2)$ into $\mathbf{x}_2(n_2)$

End If

If i -1 >= $n_0 + 1$

$$\begin{pmatrix} B(n_0+1,\mathrm{N}) \\ \vdots \\ B(i-1,\mathrm{N}) \end{pmatrix} = \begin{pmatrix} B(n_0+1,\mathrm{N}) \\ \vdots \\ B(i-1,\mathrm{N}) \end{pmatrix} + B(\mathrm{i},\mathrm{N}) \begin{pmatrix} B(n_0+1,\mathrm{i}) \\ \vdots \\ B(\mathrm{i}-1,\mathrm{i}) \end{pmatrix}$$; update arcs from $x_1(1)$ to $x_1(i - n_0 - 1)$ into $\mathbf{x}_2(n_2)$

End If

If N -1 >= $n_0 + n_1 + 1$

$$\begin{pmatrix} B(n_0+n_1+1,\mathrm{N}) \\ \vdots \\ B(N-1,\mathrm{N}) \end{pmatrix} = \begin{pmatrix} B(n_0+n_1+1,\mathrm{N}) \\ \vdots \\ B(N-1,\mathrm{N}) \end{pmatrix} +$$

$$B(\mathrm{i},\mathrm{N}) \begin{pmatrix} B(n_0+n_1+1,\mathrm{i}) \\ \vdots \\ B(N-1,\mathrm{i}) \end{pmatrix}$$; update arcs from $x_2(1)$ to $x_2(n_2 - 1)$ into $\mathbf{x}_2(n_2)$

End If

If ($\mathbf{v}$(i) ≠ 0)

If ((v(i) ≠ ∞) and (v(N) ≠ ∞)); update conditional variance of $\mathbf{x}_2(n_2)$

v(N) = v(N) + B(i,N) B(i,N) v(i)

Else

v(N) = ∞

End If

End If

End Do

$$\begin{pmatrix} \mu(n_0+1) \\ \vdots \\ \mu(n_0+n_1) \end{pmatrix} = \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix}$$; zero all entries for $\boldsymbol{\mu}_1$

$$\begin{pmatrix} v(n_0+1) \\ \vdots \\ v(n_0+n_1) \end{pmatrix} = \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix}$$; zero all entries for $\mathbf{v}_1$

$$\begin{bmatrix} \boldsymbol{B_{01}} \\ \boldsymbol{B_{11}} \end{bmatrix} = \begin{bmatrix} B(1,n_0+1) & \cdots & B(1,n_0+n_1) \\ \vdots & \ddots & \vdots \\ B(n_0+n_1,n_0+1) & \cdots & B(n_0+n_1,n_0+n_1) \end{bmatrix} = \begin{bmatrix} 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{bmatrix}$$; zero all entries for $\boldsymbol{B_{01}}$ and $\boldsymbol{B_{11}}$

$$\boldsymbol{B_{12}} = \begin{bmatrix} B(n_0+1,n_0+n_1+1) & \cdots & B(n_0+1,n_0+n_1+n_2) \\ \vdots & \ddots & \vdots \\ B(n_0+n_1,n_0+n_1+1) & \cdots & B(n_0+n_1,n_0+n_1+n_2) \end{bmatrix} = \begin{bmatrix} 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{bmatrix}$$; zero all entries for $\boldsymbol{B_{12}}$

$$\boldsymbol{B_{21}} = \begin{bmatrix} B(n_0+n_1+1, n_0+1) & \cdots & B(n_0+n_1+1, n_0+n_1) \\ \vdots & \ddots & \vdots \\ B(\mathrm{N}, n_0+1) & \cdots & B(\mathrm{N}, n_0+n_1) \end{bmatrix} = \begin{bmatrix} 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{bmatrix}$$

; zero all entries for $\boldsymbol{B_{21}}$ below the diagonal result from reversal

3.3 ARC REVERSAL BETWEEN VECTOR NODES

This algorithm is taken from Kenley (1986, 97-99) with minor modifications.

Let $\mathbf{x} = (x_1,\ldots,x_n)^T$ be a multivariate Gaussian random variable with ID form representation ($\boldsymbol{\mu}$, $\mathbf{B}$, $\mathbf{v}$). Partition the domain using the index subset from $\{1,\ldots n\}$ into four ordered subsets

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{x_0} \\ \boldsymbol{x_1} \\ \boldsymbol{x_2} \\ \boldsymbol{x_3} \end{bmatrix},\ \boldsymbol{x_0} = \begin{bmatrix} \boldsymbol{x_0}(1) \\ \vdots \\ \boldsymbol{x_0}(n_0) \end{bmatrix},\ \boldsymbol{x_1} = \begin{bmatrix} \boldsymbol{x_1}(1) \\ \vdots \\ \boldsymbol{x_1}(n_1) \end{bmatrix},\ \boldsymbol{x_2} = \begin{bmatrix} \boldsymbol{x_2}(1) \\ \vdots \\ \boldsymbol{x_2}(n_2) \end{bmatrix}, \text{and } \boldsymbol{x_3} = \begin{bmatrix} \boldsymbol{x_2}(1) \\ \vdots \\ \boldsymbol{x_3}(n_3) \end{bmatrix}$$

and the partitioned ID form representation is

$$\boldsymbol{\mu} = \begin{bmatrix} \boldsymbol{\mu_0} \\ \boldsymbol{\mu_1} \\ \boldsymbol{\mu_2} \\ \boldsymbol{\mu_3} \end{bmatrix},\ \boldsymbol{v} = \begin{bmatrix} \boldsymbol{v_0} \\ \boldsymbol{v_1} \\ \boldsymbol{v_2} \\ \boldsymbol{v_3} \end{bmatrix}, \text{and } \boldsymbol{B} = \begin{bmatrix} \boldsymbol{B_{00}} & \boldsymbol{B_{01}} & \boldsymbol{B_{02}} & \boldsymbol{B_{03}} \\ \mathbf{0} & \boldsymbol{B_{11}} & \boldsymbol{B_{12}} & \boldsymbol{B_{13}} \\ \mathbf{0} & \mathbf{0} & \boldsymbol{B_{22}} & \boldsymbol{B_{23}} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \boldsymbol{B_{33}} \end{bmatrix}$$

After reversal of $\mathbf{x}_1$ with $\mathbf{x}_2$, the ID form representation is updated in the form below.

$$\boldsymbol{\mu} = \begin{bmatrix} \boldsymbol{\mu_0} \\ \boldsymbol{\mu_1} \\ \boldsymbol{\mu_2} \\ \boldsymbol{\mu_3} \end{bmatrix},\ \boldsymbol{v} = \begin{bmatrix} \boldsymbol{v_0} \\ \boldsymbol{v_1} \\ \boldsymbol{v_2} \\ \boldsymbol{v_3} \end{bmatrix}, \text{and } \boldsymbol{B} = \begin{bmatrix} \boldsymbol{B_{00}} & \boldsymbol{B_{01}} & \boldsymbol{B_{02}} & \boldsymbol{B_{03}} \\ \mathbf{0} & \boldsymbol{B_{11}} & \mathbf{0} & \boldsymbol{B_{13}} \\ \mathbf{0} & \boldsymbol{B_{21}} & \boldsymbol{B_{22}} & \boldsymbol{B_{23}} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \boldsymbol{B_{33}} \end{bmatrix}$$

The procedure for influence diagram reversal of $\mathbf{x}_1$ with $\mathbf{x}_2$ is shown in Table 5.Table 5 Algorithm for Arc Reversal Between Vector Nodes in Gaussian Influence Diagrams

Procedure Vreverse($\mu, \boldsymbol{B}, \boldsymbol{v}, n_0, n_1, n_2, n_3$)

Do $i = n_0 + n_1, \ldots, n_0+1$; loop through x_1 in reverse order

Do $j = n_0 + n_1 + 1, n_0 + n_1 + n_2$; reverse with $\{\mathbf{x_2}(1),\ldots,\mathbf{x_2}(n_2)\}$ in forward order

If ($B(i,j) \neq 0$)

If $n_0 >= 1$

$$\begin{pmatrix} B(1,\mathrm{j}) \\ \vdots \\ B(n_0,j) \end{pmatrix} = \begin{pmatrix} B(1,\mathrm{j}) \\ \vdots \\ B(n_0,j) \end{pmatrix} + B(\mathrm{i},\mathrm{j}) \begin{pmatrix} B(1,\mathrm{i}) \\ \vdots \\ B(n_0,\mathrm{i}) \end{pmatrix}$$

; update arcs from x_0 to x_1 $\{x_1(1),\ldots, x_1(i-1-n_0)\}$ to $x_2(j-(n_0 + n_1))$

End If

If i -1 >= $n_0 + 1$

$$\begin{pmatrix} B(n_0+1,\mathrm{j}) \\ \vdots \\ B(\mathrm{i}-1,\mathrm{j}) \end{pmatrix} = \begin{pmatrix} B(n_0+1,\mathrm{j}) \\ \vdots \\ B(\mathrm{i}-1,\mathrm{j}) \end{pmatrix} + B(\mathrm{i},\mathrm{j}) \begin{pmatrix} B(n_0+1,\mathrm{i}) \\ \vdots \\ B(\mathrm{i}-1,\mathrm{i}) \end{pmatrix}$$

; update arcs from $\{x_1(1),\ldots, x_1(i-1-n_0)\}$ to $x_2(j-(n_0 + n_1))$

End If

If j-1 >= $n_0 + n_1 + 1$

$$\begin{pmatrix} B(n_0+n_1+1,\mathrm{j}) \\ \vdots \\ B(\mathrm{j}-1,\mathrm{j}) \end{pmatrix} = \begin{pmatrix} B(n_0+n_1+1,\mathrm{j}) \\ \vdots \\ B(\mathrm{j}-1,\mathrm{j}) \end{pmatrix} + B(\mathrm{i},\mathrm{j}) \begin{pmatrix} B(n_0+n_1+1,\mathrm{i}) \\ \vdots \\ B(\mathrm{j}-1,\mathrm{i}) \end{pmatrix};$$

update arcs from $\{x_2(1),\ldots, x_2(j-1-(n_0+n_1))\}$ to $x_1(i-n_0)$

End If

If (v(i) = 0)

B(j,i) = 0

Else

If ((v(i) ≠ ∞) and (v(j) ≠ ∞)) ; standard distributions

If (v(j) = 0) ; j is deterministic

v(j) = B(i,j)B(i,j)v(i)

v(i) = 0

B(j,i) = *1* / B(i,j)

Else ; both nodes probabilistic

vj_{old} = v(j)

v(j) = v(j) + B(i,j)B(i,j)v(i)

vratio = v(i) / v(j)

v(i) = vj_{old} * v_{ratio}

B(j,i) = B(i,j) * v_{ratio}

End If

Else ; non informative distributions

If (v(j) ≠ ∞)

B(j,i) = *1* / B(i,j)

Else

B(j,i) = 0

End If

If ((v(i) = ∞) and (v(j) ≠ ∞))

v(i) = v(j)B(j,i)B(j,i)

End If

v(j) = ∞

End If

End If

B(i,j) = 0

If n_0 >= 1

$$\begin{pmatrix} B(1,\mathrm{i}) \\ \vdots \\ B(n_0,i) \end{pmatrix} = \begin{pmatrix} B(1,\mathrm{i}) \\ \vdots \\ B(n_0,i) \end{pmatrix} - B(j,\mathrm{i}) \begin{pmatrix} B(1,\mathrm{j}) \\ \vdots \\ B(n_0,\mathrm{j}) \end{pmatrix};$$ update arcs from x_0 to x_1

End If

If i -1 >= $n_0 + 1$

$$\begin{pmatrix} B(n_0+1,\mathrm{i}) \\ \vdots \\ B(\mathrm{i}-1,\mathrm{i}) \end{pmatrix} = \begin{pmatrix} B(n_0+1,\mathrm{i}) \\ \vdots \\ B(\mathrm{i}-1,\mathrm{i}) \end{pmatrix} - B(j,\mathrm{i}) \begin{pmatrix} B(n_0+1,\mathrm{j}) \\ \vdots \\ B(\mathrm{i}-1,\mathrm{j}) \end{pmatrix};$$ update arcs from x_1 and x_2

End If

If j-1 >= $n_0 + n_1 + 1$

$$\begin{pmatrix} B(n_0+n_1+1,\mathrm{i}) \\ \vdots \\ B(\mathrm{j}-1,\mathrm{i}) \end{pmatrix} = \begin{pmatrix} B(n_0+n_1+1,\mathrm{i}) \\ \vdots \\ B(\mathrm{j}-1,\mathrm{i}) \end{pmatrix} - B(j,\mathrm{i}) \begin{pmatrix} B(n_0+n_1+1,\mathrm{j}) \\ \vdots \\ B(\mathrm{j}-1,\mathrm{j}) \end{pmatrix};$$

update arcs from {$x_2(1)$,..., $x_2(j-1-(n0+n1))$} to $x_1(i-n_0)$

End If

End If; end $B(i,j) \neq 0$

End Do; j loop

End Do; i loop

4 IMPLEMENTATION OF KALMAN FILTERING USING THE INFLUENCE DIAGRAM FORM

4.1 MATHEMATICAL MODEL

The mathematical model for discrete-time Kalman filtering follows Bryson and Ho (1975, 360):

Dynamic Process

$x(k+1) = \Phi(k)x(k) + \Gamma(k)w(k)\ for\ k = 0,\ldots,N.$

Measurement Process

$z(k) = \mathrm{H}(k)x(k) + v(k)\ for\ k = 0,\ldots,N.$

Probabilistic Structure

$E[x(0)] = \mu_0$

$Cov[x(0)] = \mathrm{P}_0$

$E[w(k)] = 0\ for\ k = 0,\ldots,N.$

$Cov[w(j),w(k)] = \delta_{jk}\mathrm{Q}_k\ for\ k = 0,\ldots,N..$

$Cov[x(0),w(0)] = 0.$

$E[v(k)] = 0\ for\ k = 0,\ldots,N.$

$Cov[v(j),v(k)] = \delta_{jk}\mathrm{R}_k\ for\ k = 0,\ldots,N.\ \mathrm{R}_k\ are\ diagonal\ for\ k = 0,\ldots,N.$

$Cov[w(j),v(k)] = 0\ for\ j = 1,\ldots,N\ and\ for\ k = 0,\ldots,N.$

$Cov[x(0),v(k)] = 0\ for\ k = 0,\ldots,N.$

Dimensions of Vectors

$x(k)\epsilon R^n, w(k)\epsilon R^r, z(k)\epsilon R^p, v(k)\epsilon R^p$

4.2 MEASUREMENT UPDATE FOR MEASUREMENT Z(K)

Let $\mathbf{z} = \mathbf{z}(k)$ be a measurement vector of p values.
Let $\mathbf{x}(k)$ be the state vector of n values. If k = 0, assume that the probability distribution of $\mathbf{x}(0)$ is in covariance form with $[\boldsymbol{\mu}, \boldsymbol{\Sigma}] = [\boldsymbol{\mu}_0, \mathbf{P}_0]$. If k > 0, assume that the probability distribution of $\mathbf{x}(k)$ is in ID form $(\boldsymbol{\mu}, \mathbf{B}, \mathbf{v}) = (\boldsymbol{\mu}(k), \mathbf{B}(k), \mathbf{v}(k))$ and is ordered so that $\mathbf{B}$ is upper triangular.
Let $\mathbf{v}(k)$ be the measurement noise vector of p values with p-dimensional diagonal covariance matrix $\mathbf{R} = \mathbf{R}_k$.
Let $\mathbf{H} = \mathbf{H}(k)$ be the pxn measurement matrix.

Table 6 Algorithm for Measurement Update

Procedure Mupdate(k, z, $\boldsymbol{\mu}, \boldsymbol{B\_or\_Sigma}, \boldsymbol{v}, \boldsymbol{R}, \boldsymbol{H}$)
 	
If k = 0; Convert to ID Form if k = 0
 Call Cov_to_ID ($\boldsymbol{B\_or\_Sigma}, \boldsymbol{B}, \boldsymbol{v}, \boldsymbol{P}$); do the conversion
Else
 B = B_or_Sigma
End If
$\boldsymbol{x_1} = \boldsymbol{z}$, $n_0 = n$, $n_1 = p$, $n_2 = 0$, $\boldsymbol{\mu}' = \begin{bmatrix} \boldsymbol{\mu} \\ \boldsymbol{H\mu} \end{bmatrix}$, $\boldsymbol{v}' = \begin{bmatrix} \boldsymbol{v} \\ \boldsymbol{R}(1,1) \\ \vdots \\ \boldsymbol{R}(p,p) \end{bmatrix}$, and $\boldsymbol{B}' = \begin{bmatrix} \boldsymbol{B} & \boldsymbol{H}^t \\ \mathbf{0}_{pxn} & \mathbf{0}_{pxp} \end{bmatrix}$.
 	
EnterEvidence($\mathbf{x}_1, n_0, n_1, n_2, \boldsymbol{\mu}', \boldsymbol{v}', \boldsymbol{B}'$).
The output of Enter Evidence is
$\boldsymbol{\mu}' = \begin{bmatrix} \boldsymbol{\mu}_1 \\ \mathbf{0}_p \end{bmatrix}$, $\boldsymbol{v}' = \begin{bmatrix} \boldsymbol{v}_1 \\ \mathbf{0}_p \end{bmatrix}$, and $\boldsymbol{B}' = \begin{bmatrix} \boldsymbol{B}_{11} & \mathbf{0}_{nxp} \\ \mathbf{0}_{pxn} & \mathbf{0}_{pxp} \end{bmatrix}$
This means the probability distribution of $\mathbf{x}(k)$ in ID form is $(\boldsymbol{\mu}, \mathbf{B}, \mathbf{v}) = (\boldsymbol{\mu}_1, \mathbf{B}_{11}, \mathbf{v}_1)$

4.3 TIME UPDATE FROM X(K) TO X(K+1)

Let $\mathbf{x}(k)$ be the state vector of n values and assume the probability distribution of $\mathbf{x}(k)$ is in ID form $(\boldsymbol{\mu}, \mathbf{B}, \mathbf{v}) = (\boldsymbol{\mu}(k), \mathbf{B}(k), \mathbf{v}(k))$ and is ordered so that $\mathbf{B}$ is upper triangular.
Let $\boldsymbol{\Phi} = \boldsymbol{\Phi}(k)$ be the state transition matrix from $\mathbf{x}(k)$ to $\mathbf{x}(k+1)$.
Let $\mathbf{w}(k)$ be the process noise vector of r values with r-dimensional diagonal covariance matrix $\mathbf{Q} = \mathbf{Q}_k$.
Let $\boldsymbol{\Gamma} = \boldsymbol{\Gamma}(k)$ be the nxr process noise matrix.

Table 7 Algorithm for Time Update from x(k) to x(k+1)

Procedure Tupdate($\boldsymbol{\mu}, \boldsymbol{B}, \boldsymbol{v}, \boldsymbol{\Phi}, \boldsymbol{Q}, \boldsymbol{\Gamma}$) Call Cov_to_ID ($\boldsymbol{Q}, \boldsymbol{B_q}, \boldsymbol{v_q}, \boldsymbol{P_q}$); convert the process noise to ID form $\boldsymbol{\mu}' = \begin{bmatrix} \mathbf{0}_r \\ \boldsymbol{\mu} \\ \boldsymbol{\Phi\mu} \end{bmatrix}$, $\boldsymbol{v}' = \begin{bmatrix} \boldsymbol{v_q} \\ \boldsymbol{v} \\ \mathbf{0}_n \end{bmatrix}$, $\boldsymbol{B}' = \begin{bmatrix} \boldsymbol{B_q} & \mathbf{0}_{rxn} & \boldsymbol{\Gamma}^t \\ \mathbf{0}_{nxr} & \boldsymbol{B} & \boldsymbol{\Phi}^t \\ \mathbf{0}_{nxr} & \mathbf{0}_{nxn} & \mathbf{0}_{nxn} \end{bmatrix}$, $n_0 = 0$, $n_1 = n+r$, and $n_2 = n$ Vremove($\boldsymbol{\mu}', \boldsymbol{B}', \boldsymbol{v}', n_0, n_1, n_2$). The output of VRemove is $\boldsymbol{\mu}' = \begin{bmatrix} \mathbf{0}_r \\ \mathbf{0}_n \\ \boldsymbol{\mu}'_2 \end{bmatrix}$, $\boldsymbol{v}' = \begin{bmatrix} \mathbf{0}_r \\ \mathbf{0}_n \\ \boldsymbol{v}'_2 \end{bmatrix}$, and $\boldsymbol{B}' = \begin{bmatrix} \mathbf{0}_{rxr} & \mathbf{0}_{rxn} & \mathbf{0}_{rxn} \\ \mathbf{0}_{nxr} & \mathbf{0}_{nxn} & \mathbf{0}_{nxn} \\ \mathbf{0}_{nxr} & \mathbf{0}_{nxn} & \boldsymbol{B}'_{22} \end{bmatrix}$ This means the probability distribution of $\mathbf{x}(k+1)$ in ID form is $(\boldsymbol{\mu}, \mathbf{B}, \mathbf{v}) = (\boldsymbol{\mu}'_2, \mathbf{B}'_{22}, \mathbf{v}'_2)$

4.4 KALMAN FILTER UPDATE AT TIME STEP K

Let **z** = **z(**k**)** be a measurement vector of p values.
Let **x(**k**)** be the state vector of n values. If k = 0, assume that the probability distribution of **x(**0**)** is in covariance form with [**μ, Σ**] = [$\boldsymbol{\mu}_0$, $\mathbf{P}_0$]. If k > 0, assume that the probability distribution of **x**(k**)** is in ID form (**μ**, **B**, **v**) = (**μ**(k), **B**(k), **v**(k)) and is ordered so that **B** is upper triangular.
Let **v(**k**)** be the measurement noise vector of p values with p-dimensional diagonal covariance matrix **R** = $\mathbf{R}_k$.
Let **H** = **H(**k**)** be the pxn measurement matrix.
Let **Φ** = **Φ(**k**)** be the state transition matrix from **x**(k) to **x**(k+1).
Let **w**(k) be the process noise vector of r values with r-dimensional diagonal covariance matrix **Q** = $\mathbf{Q}_k$.
Let **Γ** = **Γ(**k**)** be the nxr process noise matrix.
Set *convert* = 0 if the result is to be reported in ID form. Otherwise the result will be reported to covariance form.

Table 8 Algorithm for Kalman Filter Update at Time Step k

Procedure KFupdate(k, **z**, $\boldsymbol{\mu}, \boldsymbol{B\_or\_Sigma}, \boldsymbol{v}, \boldsymbol{R}, \boldsymbol{H}, \boldsymbol{\Phi}, \boldsymbol{Q}, \boldsymbol{\Gamma}, convert$)

Mupdate(k, **z**, $\boldsymbol{\mu}, \boldsymbol{B\_or\_Sigma}, \boldsymbol{v}, \boldsymbol{R}, \boldsymbol{H}$); Perform measurement update

The output of Mupdate is
$\boldsymbol{\mu}' = \begin{bmatrix} \boldsymbol{\mu}_1 \\ \mathbf{0}_p \end{bmatrix}$, $\boldsymbol{v}' = \begin{bmatrix} \boldsymbol{v}_1 \\ \mathbf{0}_p \end{bmatrix}$, and $\boldsymbol{B}' = \begin{bmatrix} \boldsymbol{B}_{11} & \mathbf{0}_{nxp} \\ \mathbf{0}_{pxn} & \mathbf{0}_{pxp} \end{bmatrix}$

Tupdate($\boldsymbol{\mu_1}, \boldsymbol{B_{11}}, \boldsymbol{v_1}, \boldsymbol{\Phi}, \boldsymbol{Q}, \boldsymbol{\Gamma}$) ; Perform time update

The output of Tupdate is
$\boldsymbol{\mu}' = \begin{bmatrix} \mathbf{0}_r \\ \mathbf{0}_n \\ \boldsymbol{\mu}'_{\mathbf{2}} \end{bmatrix}$, $\boldsymbol{v}' = \begin{bmatrix} \mathbf{0}_r \\ \mathbf{0}_n \\ \boldsymbol{v}'_{\mathbf{2}} \end{bmatrix}$, and $\boldsymbol{B}' = \begin{bmatrix} \mathbf{0}_{rxr} & \mathbf{0}_{rxn} & \mathbf{0}_{rxn} \\ \mathbf{0}_{nxr} & \mathbf{0}_{nxn} & \mathbf{0}_{nxn} \\ \mathbf{0}_{nxr} & \mathbf{0}_{nxn} & \boldsymbol{B}'_{\mathbf{22}} \end{bmatrix}$
This means the probability distribution of **x(**k+1**)** in ID form is (**μ**, **B**, **v**) = ($\boldsymbol{\mu}'_2$, $\mathbf{B}'_{22}$, $\mathbf{v}'_2$).
If *convert* = 0
  The output is ($\boldsymbol{\mu}, \boldsymbol{B\_or\_Sigma}, \boldsymbol{v}$) = (**μ**, **B**, **v**)
  This means the probability distribution of **x(**k+1**)** in influence diagram form has mean $\boldsymbol{\mu}$ and influence diagram arc coefficient matrix $\boldsymbol{B\_or\_Sigma}$ and conditional variance vector $\boldsymbol{v}$
Else
    Convert back to covariance form
    ID_to_Cov ($\mathbf{B}'_{22}$, $\mathbf{v}'_2$, **Σ**)
    The output is ($\boldsymbol{\mu}, \boldsymbol{B\_or\_Sigma}, \boldsymbol{v}$) = ($\boldsymbol{\mu}'_2$, **Σ**, $\mathbf{0}_{\boldsymbol{n}}$).
    This means the probability distribution of **x(**k+1**)** with mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{B\_or\_Sigma}$
End if

5 INTERFACING MATLAB SENSOR FUSION TRACKING TOOLBOX KALMAN FILTER TO INFLUENCE DIAGRAM

5.1 MATHEMATICAL MODELS

Table 9. Mapping from Bryson and Ho to MATLAB trackingKF object and correct predict functions

Bryson and Ho (1975, 360)	MATLAB trackingKF equations (trackingKF 2020)	Bryson and Ho (1975, 360)	MATLAB trackingKF properties (trackingKF 2020)	MATLAB correct and predict variables (correct 2020, predict 2020)
Dynamic Process				
$x(k+1) = \Phi(k)x(k) + \Gamma(k)w(k)\ for\ k = 0,\dots,N.$	$x_{k+1} = F_k x_k + G_k u_k + v_k$	$x(k)$	x_k	
		$\Phi(k)$	StateTransitionModel = F_k	
		$null$	ControlModel = G_k	
		$\Gamma(k)w(k)$	v_k	
Measurement Process				
$z(k) = \mathrm{H}(k)x(k) + v(k)\ for\ k = 0,\dots,N.$	$z_k = H_k x_k + w_k$	$z(k)$	z_k	zmeas
		$\mathrm{H}(k)$	MeasurementModel $= H_k$	
		$x(k)$	x_k	
		$v(k)$	w_k	
Probabilistic Structure				
$E[x(0)] = \mu_0$				
$Cov[x(0)] = \mathrm{P}_0$				
$E[w(k)] = 0\ for\ k = 0,\dots,N.$	*Implicit*			
$Cov[w(j),w(k)] = \delta_{jk}\mathrm{Q}_k\ for\ k = 0,\dots,N.$	*Implicit*	Q	ProcessNoise = $Q_{trackingKF} =$	$\boldsymbol{Q} = Q_{trackingKF}$

Bryson and Ho (1975, 360)	MATLAB trackingKF equations (trackingKF 2020)	Bryson and Ho (1975, 360)	MATLAB trackingKF properties (trackingKF 2020)	MATLAB correct and predict variables (correct 2020, predict 2020)
			$\Gamma(k)Q_k\Gamma^t(k)$	
$Cov[x(0), w(0)] = 0.$	*Implicit*			
$E[v(k)] = 0 \; for \; k = 0, \ldots, N.$	*Implicit*			
$Cov[v(j), v(k)] = \delta_{jk}\mathrm{R}_k \; for \; k = 0, \ldots, N.$ R_k $are \; diagonal \; for \; k = 0, \ldots, N.$	*Implicit*	*R*	MeasurementNoise = *R*	zcov
$Cov[w(j), v(k)] = 0 \; for \; j = 1, \ldots, N \; and \; for \; k = 0, \ldots, N.$	*Implicit*			
$Cov[x(0), v(k)] = 0 \; for \; k = 0, \ldots, N.$	*Implicit*			

Table 10. Dimensions of Vectors for Bryson and Ho and for MATLAB trackingKF object

Bryson and Ho (1975, 360)	MATLAB trackingKF equations (trackingKF 2020)
$x(k) \epsilon R^n, w(k) \epsilon R^r, z(k) \epsilon R^p, v(k) \epsilon R^p$	$x_k \epsilon R^M, u_k \epsilon R^L, v_k \epsilon R^M, z_k \epsilon R^N, w_k \epsilon R^N$

5.2 INFLUENCE DIAGRAM IMPLEMENTATION OF MATLAB CORRECT FUNCTION

Table 11 Algorithm for Influence Diagram Implementation of MATLAB correct function when filter is a trackingKF object.

Let filter = trackingKF(F, H, 'State', $\boldsymbol{\mu}$, ‘StateCovariance’, $\boldsymbol{P}$, ‘ProcessNoise’, $\boldsymbol{Q}_{trackingKF}$) Additional assumptions: (1) zmeas can only be a column vector of size N, or an Nx1 matrix. The function to implement in MATLAB is [xcorr,Pcorr] = IDcorrect(filter, zmeas, zcov) This is done by assigning using Mupdate as follows: Mupdate(0, zmeas, $\boldsymbol{\mu}, \boldsymbol{P}, \boldsymbol{0}_{M\times 1}$, zcov, H) The output of Mupdate is in influence diagram form (**μ**, **B**, **v**) and must be converted to covariance form. Call ID_to_Cov (**B**, **v**, **Σ**) The output of IDcorrect is [xcorr, Pcorr] = [**μ**, **Σ**]

5.3 INFLUENCE DIAGRAM IMPLEMENTATION OF MATLAB PREDICT FUNCTION

Table 12 Algorithm for Influence Diagram Implementation of MATLAB predict function when filter is a trackingKF object.

Let filter = trackingKF(F, H, 'State', $\boldsymbol{\mu}$, ‘StateCovariance’, $\boldsymbol{P}$, ‘ProcessNoise’, $\boldsymbol{Q}_{trackingKF}$) Additional assumptions: (1) There is no control model, G, or control input, u. (2) If using the model from Bryson and Ho (1975, 360), $\boldsymbol{Q}_{trackingKF}$ must be calculated using matrix multiplication: $\boldsymbol{Q}_{trackingKF} = \Gamma(k)Q_k\Gamma^t(k)$, where $w(k)$ is the process noise at time step k, Q_k is the covariance of the process noise, and $\Gamma(k)$ is the mapping of the process noise to the state space in the model from Bryson and Ho. The function to implement in MATLAB is [xpred,Ppred] = IDpredict(filter) This is done by using TUpdate as follows: The input of IDpredict is in covariance diagram form and must be converted to influence diagram form. Let **Σ** = filter.StateCovariance Call Cov_to_ID (**Σ, B**, **v**, **P**)

$(\boldsymbol{\mu}'_2, \mathbf{B}'_{22}, \mathbf{v}'_2) = \text{Tupdate}(\boldsymbol{\mu}, \mathbf{B}, \mathbf{v}, \text{F}, \boldsymbol{Q}_{trackingKF}, \boldsymbol{I}_{M\times M})$

The output of Mupdate is in influence diagram form ($\boldsymbol{\mu}'_2$, $\mathbf{B}'_{22}$, $\mathbf{v}'_2$) and must be converted to covariance form.

Call ID_to_Cov ($\mathbf{B}'_{22}$, $\mathbf{v}'_2$, $\boldsymbol{\Sigma}$)

The output of IDpredict is [xpred, Ppred] = [$\boldsymbol{\mu}'_2$, $\boldsymbol{\Sigma}$]

6 REFERENCES

Bryson, Arthur E., and Yui-Chi Ho. 1975. *Applied Optimal Control: Optimization, Estimation and Control.* Washington, DC: Hemisphere Publishing.

Kenley, C. Robert. 1986. *Influence Diagram Models with Continuous Variables.* Phd Thesis, Engineering-Economic Systems, Stanford, CA: Stanford University.

Mathworks. 2020. "correct." Accessed July 7, 2020. https://www.mathworks.com/help/fusion/ref/trackingekf.correct.html.

—. 2020. "predict." Accessed July 7, 2020. https://www.mathworks.com/help/fusion/ref/trackingekf.predict.html.

—. 2020. "trackingKF." *Mathworks Help Center.* Accessed July 7, 2020. https://www.mathworks.com/help/fusion/ref/trackingkf.html.

Shachter, Ross D., and C. Robert Kenley. 1989. "Gaussian Influence Diagrams." *Management Science* 35 (5): 527-550. http://www.jstor.org/stable/2632102.