RESOLVER

7/1/2012

Technical Documentation

A detailed description of the algorithms and analysis used within the Resolver data reconciliation engine.

Resolver

A DETAILED DESCRIPTION OF THE ALGORITHMS AND ANALYSIS USED WITHIN THE RESOLVER RECONCILIATION ENGINE

Contents

THE DATA RECONCILIATION PROBLEM

The objective of data reconciliation is to produce an estimate of the variables such that all constraints are met and the estimates are 'close' to the provided measurements.

Typical constraints are that the total material mass flow in and out of a node within a flowsheet must be zero, since mass cannot be created or destroyed.

Objective Function

'Closeness' of the estimates to the provided measurements has a variety of interpretations. The one most widely adopted is based on the assumption that the measurements differ from the 'true' value of the variable by a normally distributed random error. It can thus be shown, under these assumptions, that the 'best' estimates are found when the following objective function is minimized:

$$Objective\ function\ J = \sum_{i=1}^{n} \frac{(x_i - y_i)^2}{q_i}$$

Where

i is an index to the measurement or variable of the problem
x_i is the estimate that is sought for the i-th variable
y_i is the measurement of the i-th variable
q_i is the variance (sometimes called the tolerance) of the measurement error of the i-th variable.

However we need to consider the case when the estimates are cross-correlated. This is not usually the case with 'raw' measurements, however as we progress through the solution we will see that the estimates are indeed correlated. Thus let's use a matrix notation:

$$Objective\ function\ J = (\boldsymbol{x} - \boldsymbol{y})^T . Q^{-1} . (\boldsymbol{x} - \boldsymbol{y})$$

Where

x is the vector of i estimates of the variables
y is the vector of i measurements of the variables
Q is the covariance matrix.

Constraints

The constraints that typically arise when formulating data reconciliation problems are of the form:

A linear equality constraint:	$x_1 + x_2 - x_3 = 0$,
A nonlinear equality constraint:	$x_1 * x_4 - x_5 * x_6 = x_7$
A linear inequality constraint:	$x_1 + x_2 - x_3 > 0$
A nonlinear inequality constraint:	$x_1 * x_4 - x_5 * x_6 <= x_7$

For now we will ignore the inequality constraints, because as we will see later these become equality constraints under certain circumstances.

Thus we can write these constraints in the vector form, where g is a vector function:

$$\boldsymbol{g}(\boldsymbol{x}) = \mathbf{0}$$

Lagrangian

We can now formulate the data reconciliation problem (see Appendix: Lagrangian Formulation):

$$Lagrangian, L = (\boldsymbol{x} - \boldsymbol{y})^T Q^{-1} (\boldsymbol{x} - \boldsymbol{y}) + \boldsymbol{l}^T \boldsymbol{g}(\boldsymbol{x})$$

If we minimize the Lagrangian L with-respect-to the estimates **x**, then we will have also met the constraints.

Before doing so it makes sense to use the Taylor series expansion of the constraint function **g**(**x**):

$$\boldsymbol{g}(\boldsymbol{x}) = \boldsymbol{g_0} + \frac{\partial \boldsymbol{g}(\boldsymbol{x})}{\partial \boldsymbol{x}} . \boldsymbol{x} + \boldsymbol{x^T} . \frac{\partial^2 \boldsymbol{g}(\boldsymbol{x})}{\partial \boldsymbol{x^2}} . \boldsymbol{x} + \cdots$$

We will be using this expansion frequently so let's rewrite it as:

$$\boldsymbol{g}(\boldsymbol{x}) \cong \boldsymbol{g_0} + A . \boldsymbol{x}$$

Thus differentiating the Lagrangian L with respect t x, and l we have these two equations:

$$\frac{\partial L}{\partial \boldsymbol{x}} = 2 . Q^{-1} . (\boldsymbol{x} - \boldsymbol{y}) + A^T . \boldsymbol{l} = \mathbf{0}$$

$$\frac{\partial L}{\partial \boldsymbol{l}} = A . \boldsymbol{x} = -\boldsymbol{g_0}$$

These can be rewritten in the familiar form as:

$$\begin{bmatrix} 2 . Q^{-1} & A^T \\ A & 0 \end{bmatrix} . \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{l} \end{bmatrix} = \begin{bmatrix} 2 . Q^{-1} . \boldsymbol{y} \\ -\boldsymbol{g_0} \end{bmatrix}$$

The solution for x and l can then be expressed as:

$$\begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{l} \end{bmatrix} = \begin{bmatrix} 2 . Q^{-1} & A^T \\ A & 0 \end{bmatrix}^{-1} \begin{bmatrix} 2 . Q^{-1} . \boldsymbol{y} \\ -\boldsymbol{g_0} \end{bmatrix}$$

Using the Matrix Inversion Lemma we can rewrite the right-hand-side as:

$$\begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{l} \end{bmatrix} = \begin{bmatrix} \frac{Q}{2}(I - A^T (AQA^T)^{-1} AQ) & QA^T (AQA^T)^{-1} \\ (AQA^T)^{-1} AQ & -2(AQA^T)^{-1} \end{bmatrix} \begin{bmatrix} 2 . Q^{-1} . \boldsymbol{y} \\ -\boldsymbol{g_0} \end{bmatrix}$$

We can then multiply out the right-hand-side, and with a bit of rearrangement we get:

$$\begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{l} \end{bmatrix} = \begin{bmatrix} Q(I - A^T(AQA^T)^{-1}AQ)Q^{-1}.\boldsymbol{y} - QA^T(AQA^T)^{-1}.\boldsymbol{g_0} \\ 2(AQA^T)^{-1}(A.\boldsymbol{y} + \boldsymbol{g_0}) \end{bmatrix}$$

$$\begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{l} \end{bmatrix} = \begin{bmatrix} \boldsymbol{y} - QA^T(AQA^T)^{-1}(A.\boldsymbol{y} + \boldsymbol{g_0}) \\ 2(AQA^T)^{-1}(A.\boldsymbol{y} + \boldsymbol{g_0}) \end{bmatrix}$$

Kalman Formulation

Kalman recognized the iterative formulation of this equation, if one assumes that y is the 'initial' estimate, and x becomes the updated estimate. However since we have introduced a recursive formulation, let's also change the notation:

- Let the initial estimate **y** be renamed $\mathbf{x_{n-1}}$, then the updated estimate becomes x_n
- Let the initial estimate of the covariance be Q_{n-1}
- Let the constant part $\mathbf{g}_0$ of the Taylor expansion of the constraint vector function be $\mathbf{g}_0(\mathbf{x}_{n-1})$ highlighting that it is the constant part of Taylor series expansion based on the best prior estimate $\mathbf{x}_{n-1}$ of the function

Thus the covariance update equation becomes:

$$Q_n = Q_{n-1}(I - A^T(AQ_{n-1}A^T)^{-1}AQ_{n-1})$$

And the estimate update equation becomes:

$$\boldsymbol{x_n} = \boldsymbol{x_{n-1}} - Q_{n-1}A^T(AQ_{n-1}A^T)^{-1}(A.\boldsymbol{x_{n-1}} + \boldsymbol{g_0}(\boldsymbol{x_{n-1}}))$$

To make this equation easier to manage we can introduce some terms:

Alpha:

$$\alpha = AQ_{n-1}A^T + \rho$$

Convergence Control

Note that ρ originates from the full Kalman filter and corresponds to a measure of the process noise. In the case of linear equality and inequality constraints ρ is set to 0.0 and convergence will be the same as Newtonian descent, alternatively known as quadratic convergence. However in the case of nonlinear constraints ρ can be used to control the rate of descent. If ρ is zero descent proceeds at a Newtonian pace, attempting to jump directly to the minimum. As ρ increases, the descent rotates to the steepest descent direction, and the step size becomes smaller. It has been shown that this guarantees convergence, albeit at an infinitesimally slow pace.

Kalman gain:

$$\boldsymbol{k} = Q_{n-1}A^T\alpha^{-1}$$

Update covariance:

$$Q_n = Q_{n-1} - \boldsymbol{k}.A^T Q_{n-1}$$

Innovation:

$$\boldsymbol{\varepsilon} = \boldsymbol{g_0} + A.\boldsymbol{x_{n-1}}$$

Estimate update:

$$\boldsymbol{x_n} = \boldsymbol{x_{n-1}} - \boldsymbol{k}.\boldsymbol{\varepsilon}$$

The above then becomes the well known Kalman updating algorithm, but modified for the data reconciliation problem.

Active Set Inequality Constraints

Throughout the above problem formulation we have ignored the inequality constraints. Although there are a variety of techniques to handle inequality constraints, the Active Set approach is best aligned with the recursive constraint approach of Resolver.

The Active Set algorithm works as follows:

1. Evaluate each inequality constraints to see if any are 'active'.
2. Determine which of the active inequality constraint is the 'most' violated (most active).
3. Apply this most active inequality constraint to the solution thus updating the estimates.
4. The process steps 1-3 are repeated until there are no further inequality constraints that are active.

Once an active constraint is identified, it is no longer an 'inequality' constraint, but an equality constraint:

$C(\mathbf{x_n}) > 0$ added to the set of active constraints, thus treated as $C(\mathbf{x_n}) = 0$

Thus there needs to be a way to identify the 'most active' constraint. That constraint is the one that is the greatest distance from the current.

The measure used is as follows, where i is the i-th inactive inequality.

$$\boldsymbol{k_i} = \frac{\boldsymbol{\varepsilon}_i}{\alpha_i}$$

The most active constraint is the one with the greatest k value.

SOLUTION ALGORITHM

Efficient and Stable Kalman Updating Algorithm

Despite its simplicity and elegance, the Kalman formulation has significant numerical stability problems that were studied extensively in the 1970's and 1980's, particularly by Bierman in his paper 'Filtering and Error Analysis via the UDU^T covariance factorization'.[1] The algorithm as implemented in Resolver is documented below:

Observation Model

Definition of the model used as the basis of the UDU^T algorithm that follows.

```
Variable(n,s)                = Expected[ Variable(s) ]

Innovation(n)                = -Constraint(n, Variable(n, s))

or

Innovation(n)                = Measurement(n) - MeasurementEquation(n,Variable(n,s))

r(n)                         = Expected[ Innovation(n)**2 ]
```

Kalman Equations for covariance update

Conventional Kalman updating algorithm

```
Gradient(n,j)                = Derivative[ Innovation(n) wrt Variable(j) ]

InnovationCovariance(n)      = Gradient(n, i) * P(n, i, j) * Gradient(n, j) + r(n)

KalmanGain(n, j)             = P(n, j, i) * Gradient(n, i) / InnovationCovariance(n)

P(n + 1, i, j)               = P(n, i, j) - KalmanGain(n, i) * Gradient(n, k) * P(n, k, j)

Variable(n + 1, j)           = Variable(n, j) + KalmanGain(n, j) * Innovation(n)
```

Efficient Kalman Equations but with numerical stability (UDU Factorization)

Efficient and stable Kalman updating algorithm.

```
NormalizedKalmanGain(n, j)   = P(n, j, i) * Gradient(n, i)

InnovationCovariance(n)      = Gradient(n, i) * NormalizedKalmanGain(n, i) + r(n)

KalmanGain(n, j)             = NormalizedKalmanGain(n, j) / InnovationCovariance(n)

P(n + 1, i, j)               = P(n, i, j) - KalmanGain(n, i) * NormalizedKalmanGain(n, j)

Variable(n + 1, j)           = Variable(n, j) + KalmanGain(n, j) * Innovation(n)
```

U-D Factorization

U-D factorization converts the covariance matrix P into the product of an upper diagonal U, and diagonal matrix D as P = U.D.U^T. This results in much greater stability and a reduction in calculations.

```
Transformation P(n,i,j)-> U(n,i,k) * D(n,k) * U(n,j,k)
```

```
F(n, j)                         = U(n, i, j) * Gradient(n, i)

B(n, j)                         = D(n, j) * F(n, j)

Alpha(0)                        = r(n)

Iterate j = 1, Number[ Variable(s) ]

        Alpha(j)            = Alpha(j-1) + F(n,j) * B(n,j)

        D(n+1,j)            = ( Alpha(j-1) / Alpha(j) ) * D(n,j)

        NormalizedKalmanGain(n,j) = B(n,j)

        Lambda                  = - F(n,j) / Alpha(j-1)

        Iterate i = 1, j-1

                U(n+1,i,j)      = U(n,i,j) + NormalizedKalmanGain(n,i) * Lambda

                NormalizedKalmanGain(n,i)= NormalizedKalmanGain(n,i) +
                                U(n,i,j)*NormalizedKalmanGain(n,j)

        Next i

Next j

Variable(n + 1, j)              = Variable(n, j) + KalmanGain(n, j) * Innovation(n)
```

Constraint Sorting and Convergence Control

In theory the Resolver algorithm can iterate through all of the constraints in any order. However accelerated convergence is achieved when the constraints are sorted and applied as follows:

Constraint Sorting By Category

Initialize estimate to measurements
Repeat
Initialize covariance to measurement tolerance
Apply all linear equality constraints
Apply all nonlinear equality constraints
Iterate
Most active linear or nonlinear inequality constraint
Until no further active inequality constraints
Retain estimate
Until converged

Constraint Sorting By Complexity

Furthermore it is even possible to sort the constraints according to the 'complexity' of the constraint. There is no universal definition of complexity; however the measure used in Resolver is the number of nonlinear operators plus the number of variables. It has been found that convergence is improved if constraints are applied in decreasing order of complexity.

Convergence Control

In the calculation of α a term ρ was introduced as follows:

$$\alpha = AQ_{n-1}A^T + \rho$$

ρ originates from the full Kalman filter and corresponds to a measure of the process noise. In the case of linear equality and inequality constraints ρ is set to 0.0 and convergence will be the same as Newtonian descent, alternatively known as quadratic convergence. However in the case of nonlinear constraints ρ can be used to control the rate of descent. If ρ is zero descent proceeds at a Newtonian pace, attempting to jump directly to the minimum. As ρ increases, the descent rotates to the steepest descent direction, and the step size becomes smaller. It has been shown that this guarantees convergence, albeit at an infinitesimally slow pace.

Thus convergence can be controlled by adjusting ρ.

Program Variable Mapping

The program variables are mapped to the problem formulation as follows:

Name	**Program variable**	**Formula**
Covariance	dUT	**Q**
Estimates	Results	**x**
Kalman Gain	dCp	**QA^T**

ERROR DIAGNOSIS MEASURES

Gross Error Detection

The Global (or Gross) Error detection is a single metric whose role is to indicate if the overall problem has errors that are invalidating the original assumption that:

- All measurements are approximately correct
- All constraints are accurately defined.

The global critical value is calculated as follows:

$$\text{Global Critical Value} = Inv - \chi^2(confidence, redundancy\ degre)$$

Where

Confidence is the confidence level used when assigning tolerances, such as 95% or 0.95

Redundancy degree is the number of measurements which have more than one way of estimation. (????)

Measurement Error Detection

The measurement critical value is used as a threshold value. If any measurement differs by more than this amount then the measurement error is unlikely to be explained by random errors alone. As an example, if this were to arise in a material balance problem then it is possible that the measurement was recorded incorrectly oir there is a significant calibration error.

The measurement critical value is calculated as follows:

$$\beta = (1 - (1 - confidence)^{\frac{1}{n}})$$

$$\text{Measurement Critical Value} = Erf(1 - \frac{\beta}{2})$$

Where

Confidence is the confidence level used when assigning tolerances, such as 95% or 0.95

n is the number of distinct measurements

Constraint Error Detection

The constraint critical value is used as a threshold value. If any constraint, when evaluated using the initial measurements, differs by more than this amount then the constraint error is unlikely to be explained by random errors alone. As an example, if this were to arise in a material balance problem then it is possible that there are additional material flows.

The constraint critical value is calculated as follows:

$$\beta = (1 - (1 - confidence)^{\frac{1}{n}})$$

$$\text{Measurement Critical Value} = Erf(1 - \frac{\beta}{2})$$

Where

Confidence is the confidence level used when assigning tolerances, such as 95% or 0.95

n is the number of distinct constraints

Solution Sensitivity

Another aspect of any solution that is helpful in diagnosis is the sensitivity of the results with respect to the measurements. There are two variants of sensitivity:

- How much influence does a measurement have on an estimate?
- How much influence does a measurement have on a constraint?

Estimate Sensitivity

The solution at convergence can be written as

$$\boldsymbol{x} = \boldsymbol{y} - QA^T(AQA^T)^{-1}(A.\boldsymbol{y} + \boldsymbol{g_0})$$

The sensitivity of the estimates, x, with respect to the measurements, y can be expressed as:

$$\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{y}} = I - QA^T(AQA^T)^{-1}A$$

However the updated covariance is given as follows:

$$Q_n = Q_0 - Q_0A^T(A\,Q_0A^T)^{-1}A\,Q_0$$

Thus post-multiplying by Q_0 we get the required expression for the sensitivity:

$$Q_nQ_0^{-1} = I - Q_0A^T(A\,Q_0A^T)^{-1}A = \frac{\partial \boldsymbol{x}}{\partial \boldsymbol{y}}$$

Or more succinctly:

$$\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{y}} = Q_nQ_0^{-1}$$

In words, the sensitivity is the estimated covariance normalized by the initial covariance.

Constraint Sensitivity

The constraints in the linear case are written as

$$\boldsymbol{g}(\boldsymbol{x}) \cong A\boldsymbol{x} = 0$$

Thus the sensitivity of the constraints with respect to the measurements, y, is:

$$\frac{\partial}{\partial \boldsymbol{y}} \boldsymbol{g}(\boldsymbol{x}) \cong \frac{\partial}{\partial \boldsymbol{y}} A\boldsymbol{x} = A \frac{\partial \boldsymbol{x}}{\partial \boldsymbol{y}}$$

Since the sensitivity of the estimate with respect to the measurement has already been evaluated we can write this as:

$$\frac{\partial}{\partial \boldsymbol{y}} g(\boldsymbol{x}) = A\, Q_n Q_0^{-1}$$

APPENDIX: LAGRANGIAN FORMULATION

Consider the two-dimensional problem introduced above:

maximize $f(x, y)$
subject to $g(x, y) = c.$

We can visualize contours of *f* given by

$$f(x, y) = d$$

for various values of d, and the contour of g given by $g(x, y) = c$.

Suppose we walk along the contour line with $g = c$. In general the contour lines of f and g may be distinct, so following the contour line for $g = c$ one could intersect with or cross the contour lines of f. This is equivalent to saying that while moving along the contour line for $g = c$ the value of f can vary. Only when the contour line for $g = c$ meets contour lines of f tangentially, do we not increase or decrease the value of f - that is, when the contour lines touch but do not cross.

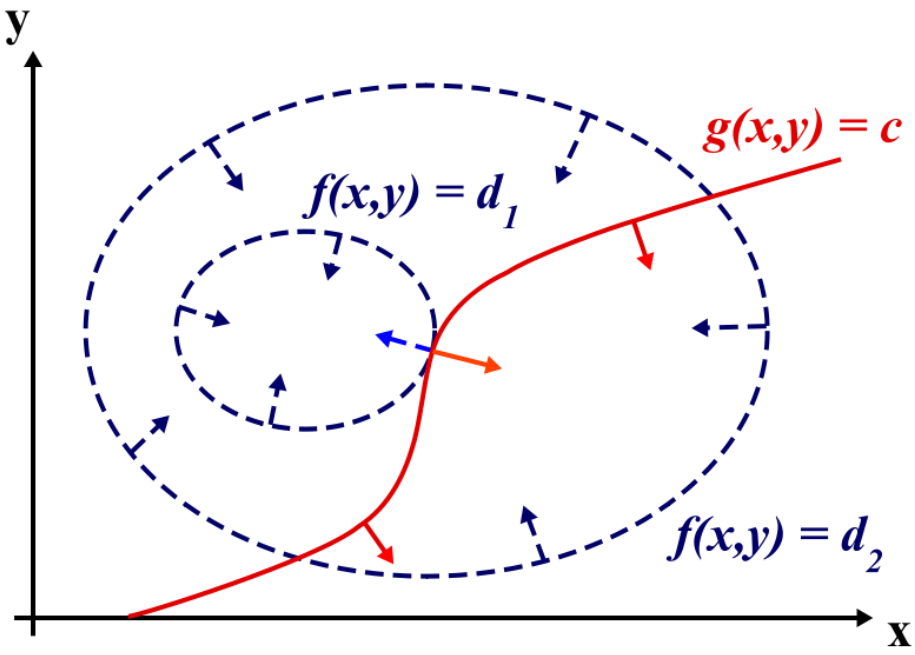

FIGURE 1: FIND X AND Y TO MAXIMIZE F(X,Y) SUBJECT TO A CONSTRAINT (SHOWN IN RED) G(X,Y)=C.

The contour lines of *f* and *g* touch when the tangent vectors of the contour lines are parallel. Since the gradient of a function is perpendicular to the contour lines, this is the same as saying that the gradients of *f* and *g* are parallel.

Thus we want points (x, y) where $g(x, y) = c$ and

$$\nabla_{x,y} f = -\lambda \nabla_{x,y} g,$$

where

$$\nabla_{x,y} f = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y} \right)$$

and

$$\nabla_{x,y} g = \left(\frac{\partial g}{\partial x}, \frac{\partial g}{\partial y} \right)$$

are the respective gradients. The constant λ is required because although the two gradient vectors are parallel, the magnitudes of the gradient vectors are generally not equal.

To incorporate these conditions into one equation, we introduce an auxiliary function

$$\Lambda(x, y, \lambda) = f(x, y) + \lambda \cdot \Big(g(x, y) - c \Big),$$

and solve

$$\nabla_{x,y,\lambda} \Lambda(x, y, \lambda) = 0.$$

This is the method of Lagrange multipliers. Note that $\nabla_{\lambda} \Lambda(x, y, \lambda) = 0$ implies $g(x, y) = c$.

APPENDIX: MATRIX INVERSION LEMMA

The Matrix Inversion Lemma is one of those handy rewrites of a matrix. Believe it or not the right-hand-side below is often easier to handle than the left-hand-side. This occurs particularly when W is a vector.

$$\begin{bmatrix} X & W^T \\ W & 0 \end{bmatrix}^{-1} = \begin{bmatrix} X^{-1}(I - W^T(WX^{-1}W^T)^{-1}WX^{-1} & X^{-1}W^T(WX^{-1}W^T)^{-1} \\ (WX^{-1}W^T)^{-1}WX^{-1} & -(WX^{-1}W^T)^{-1} \end{bmatrix}$$

A slightly more general version is:

$$\begin{bmatrix} X & Y \\ W & 0 \end{bmatrix}^{-1} = \begin{bmatrix} X^{-1}(I - Y(WX^{-1}Y)^{-1}WX^{-1} & X^{-1}Y(WX^{-1}Y)^{-1} \\ (WX^{-1}Y)^{-1}WX^{-1} & -(WX^{-1}Y)^{-1} \end{bmatrix}$$

REFERENCES

[1] 'Filtering and Error Analysis via the UDU^T covariance factorization', Thornton, C.L., and Bierman, G.J. IEEE Trans AC-23, October 1978, pp901-907.