Systems of Nonlinear Equations
Part II

L. T. Biegler
Chemical Engineering Department
Carnegie Mellon University
Pittsburgh, PA 15213
Biegler@cmu.edu
http://dynopt.cheme.cmu.edu

February 24, 2000

Obtaining Derivatives

Broyden vs. Newton Method

First Order Methods

  Dominant Eigenvalue

  Wegstein

Recycle Tearing Algorithms

Obtaining Derivatives

Symbolic Differentiation

Tools like Maple and Mathematica develop symbolic differentiation methods that give the functional derivatives in closed form. These can also be ported to FORTRAN or C calls but tend to be extremely lengthy for complicated functions and should be avoided for all but the smallest problems.

Finite Differences

A simple alternative to an exact calculation of the derivatives is to use a finite difference approximation, given by:

$$\left(\frac{\partial f}{\partial x_j}\right)_{x^k} = \frac{f\left(x^k + he_j\right) - f(x^k)}{h}$$

where e_j is given by:

$$(e_j)_i = \begin{cases} 0 & i \neq j \\ 1 & i = j \end{cases}$$

and h is a scalar normally chosen from 10^{-6} to 10^{-3}. This approach requires an additional n function evaluations per iteration.

Grouping Finite Differences

Consider a tridiagonal system:

```
x  x
x  x  x
   x  x  x
      x  x  x
         x  x  x
            x  x  x
               x  x  x
                  x  x  x
                     ..............
```

We can perturb variables {1, 4, 7, 10,...} simultaneously as well as variables {2, 5, 8, 11,...} and {3, 6, 9, 12,...} because they do not interact. As a result, the complete Jacobian can be obtained with only three perturbations.

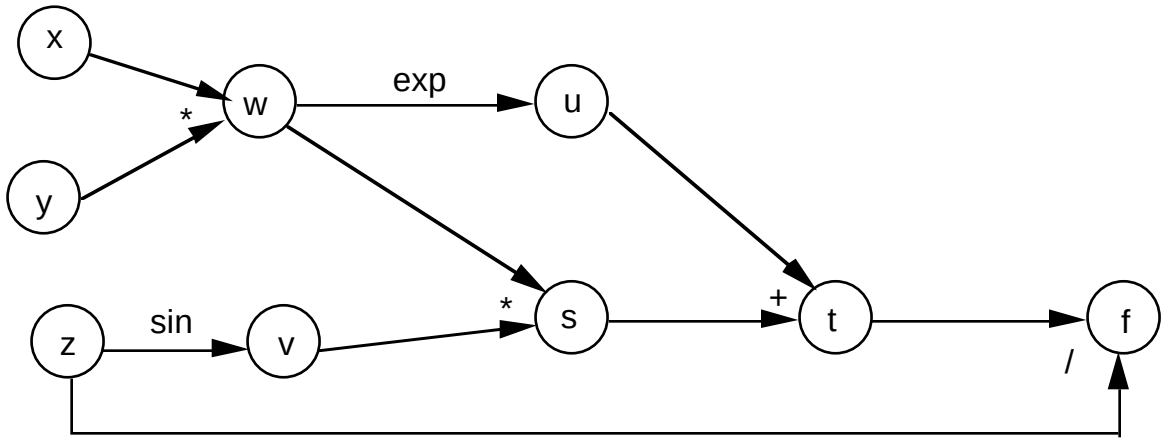
Sparse systems in general can be grouped using a 'graph coloring' algorithm that determines noninteracting variables. Two algorithms in this class are Coleman-More and Curtis, Powell, Reid (see Harwell library).

Automatic Differentiation

Consider the problem

$f(x, y, z) = (x\ y \sin z + \exp(x\ y))/z$

and let's break it down to intermediate values (nodes) and operations (arcs) as follows:



For differentiation, each operation can be replaced by the corresponding derivative operation, e.g.:

$$w = x*y \Longrightarrow \partial w/\partial x = y,\ \partial w/\partial y = x$$

$$v = \sin(z) \Longrightarrow \partial v/\partial z = \cos(z), \text{ etc.}$$

and the operations can be chain-ruled forward or backward in order to get $\partial f/\partial x$, $\partial f/\partial y$, $\partial f/\partial z$.

This is done using the numerical values, not the symbolic values and it can be applied to existing FORTRAN and C codes.

Available Codes for Automatic Differentiation

ADIFOR

- translator of FORTRAN code to create 'derivative' code that is compiled and run along with function evaluations
- runs in reverse and forward mode
- 'commercial' product, free for research purposes
- maintained at Argonne National Lab.

ADIC

- translator of C code to create 'derivative' code that is compiled and run along with function evaluations
- runs in reverse and forward mode
- 'commercial' product, free for research purposes
- maintained at Argonne National Lab.

ADOL-C

- C++ code overload function operations for 'derivative' operations that are run simultaneously with same code
- runs in reverse and forward mode
- freely available
- obtained from Prof. Griewank, Dresden, Germany

JAKE-F

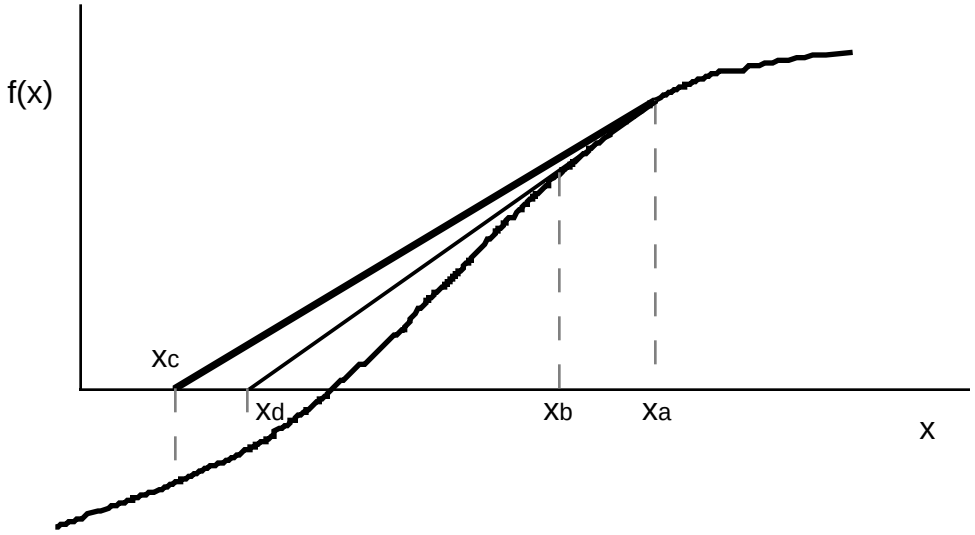
- translator of limiteed FORTRAN code to create 'derivative' code - obsolete.
- runs in reverse mode
- freely available on netlib.org

Broyden's Method

Consider the class of *Quasi-Newton methods*

- Avoid evaluation and decomposition of Jacobian
- Jacobian is approximated based on differences in x and f(x),

The basis for this derivation can be seen by considering a single equation with a single variable:



Comparison of Newton and Secant Methods for Single Equation

Newton's method to the system starting from x^a --> x_c

Newton step: $x_c = x_a - f(x^a)/f'(x^a)$

where f'(x) is the slope.

If f'(x) is not readily available, we can approximate this term by a difference between two points, say x_a and x_b. From the thin line the next point is given by x_d from a secant that is drawn between x_a and x_b.

The secant formula to obtain x_d is given by:

$$\text{Secant step: } x_d = x_a - f(x^a) \left[\frac{x_b - x_a}{f(x_b) - f(x_a)} \right]$$

Moreover, we can define a *secant relation* so that for some scalar, ***B***, we have:

$$\boldsymbol{B}\,(x_b - x_a) = f(x_b) - f(x_a) \qquad x_d = x_a - \boldsymbol{B^{-1}}\, f(x^a)$$

For the multivariable case, need to consider additional conditions to obtain a secant step.

Define a B that substitutes for ∇f^T so that

$$B^{k+1}\,(x^{k+1} - x^k) = f(x^{k+1}) - f(x^k)$$

As in Newton's method, B^k can be substituted to calculate the change in x:

$$x^{k+1} = x^k - (B^k)^{-1}\, f(x^k)$$

Secant relation is not enough to define B.

- Calculate the least change to B^{k+1} from B^k that satisfies the secant formula.

- Constrained minimization problem written as:

$$\text{Min } \left\| B^{k+1} - B^k \right\|_F$$
$$\text{s.t. } B^{k+1} s = y$$

where $y = f(x^{k+1}) - f(x^k)$
$s = x^{k+1} - x^k$
$\|B\|_F = [\Sigma_i \Sigma_j B_{ij}^2]^{1/2}$.

Solution of this convex NLP leads to *Broyden's formula:*

$$B^{k+1} = B^k + (y - B^k s)s^T / s^T s$$

With this relation we can calculate the new search direction by solving directly:

$$B^{k+1} p^{k+1} = - f(x^{k+1})$$

We can also calculate p^{k+1} *explicitly*

Update *inverse* of B^{k+1} through a modification of Broyden's formula.

Apply the Sherman Morrison Woodbury formula for a square matrix A with an update using vectors x and v:

$$(A + xv^T)^{-1} = A^{-1} - \frac{A^{-1}xv^TA^{-1}}{1+v^TA^{-1}x}$$

$(A + xv^T)$ is a *rank one update* to A.

Assign:

$A \Rightarrow B^k$
$A + xv^T \Rightarrow B^{k+1}$
$x \Rightarrow (y - B^ks)/s^Ts$
$v = s$

after simplifying, we have for $H^k = (B^k)^{-1}$

$$H^{k+1} = H^k + \frac{\left(s - H^ky\right)s^TH^k}{s^TH^ky}$$

<u>The Broyden algorithm can now be stated as follows:</u>

1. Guess x^0, B^0 (e.g. J^0 or I) or calculate H^0 (e.g. $(J^0)^{-1}$)
2. If k = 0, go to 3, otherwise calculate $f(x^k)$, $y = f(x^k) - f(x^{k-1})$, $s = x^k - x^{k-1}$ and H^k or B^k from the Broyden formula
3. Calculate the search direction by $p^k = -H^k f(x^k)$ or by solving $B^kp^k = -f(x^k)$.
4. If $||p^k|| \le \varepsilon_1$, and $||f(x^k)|| \le \varepsilon_2$ stop. Else, find the stepsize α and update the variables: $x^{k+1} = x^k + \alpha p^k$.
5. Set k = k+1, go to 2.

Characteristics of Broyden method:

- widely used in process simulation, when the number of equations is small (< 100).
- used for inside-out flash calculations and for recycle convergence in flowsheets.
- rank one update formulas for Broyden's method that approximate the Jacobian ensure superlinear convergence defined by:

$$\lim_{k\to\infty} \frac{\left\| x^{k+1} - x^{*} \right\|}{\left\| x^{k} - x^{*} \right\|} \to 0$$

Some Drawbacks:

- B and H are dense matrices although Broyden's method can be applied directly to L U factors
- There is no guarantee of a descent direction for the line search and this may fail.

Notes:

- B* does not in general converge to J*
- If $B^0 = J^0$, elements from linear equations do not change (linear subsets property) and hence linear equations are always satisfied.
- Often $B^0 = I$ is used for flowsheet convergence and this behaves less well.

Example

Use Broyden's method with an Armijo line search, solve the following system of equations and compare to Newton's method:

$$f_1 = 2x_1^2 + x_2^2 - 6 = 0$$
$$f_2 = x_1 + 2x_2 - 3.5 = 0$$

The Newton and Broyden iterations are given by:

$$x^{k+1} = x^k - (J^k)^{-1} f(x^k)$$
$$x^{k+1} = x^k - H^k f(x^k)$$

and the Jacobian matrix and its inverse are given as:

$$J^k = \begin{bmatrix} 4x_1 & 2x_2 \\ 1 & 2 \end{bmatrix} \qquad J^{-1} = (8x_1 - 2x_2)^{-1} \begin{bmatrix} 2 & -2x_2 \\ -1 & 4x_1 \end{bmatrix}$$

Starting from $x^0 = [2., 1.]^T$, we obtain the following values for x^k and we see that the constraint violations quickly reduce with full steps. The problem is essentially converged after three iterations with a solution of $x_1^* = 1.59586$ and $x_2^* = 0.95206$. Note that because we start reasonably close to the solution, $\alpha^k = 1$ for all of the steps.

k	x_1^k	x_2^k	$\|f(x^k)\|^2$	α^k
0	2.00000	1.00000	4.6250	1.0000
1	1.64285	0.92857	3.3853E-02	1.0000
2	1.59674	0.95162	1.1444E-05	1.0000
3	1.59586	0.95206	1.5194E-12	1.0000

On the other hand, with Broyden's method starting with $B^0 = J(x^0)$, we have:

k	x_1^k	x_2^k	$\|f(x^k)\|^2$	α_k
0	2.00000	1.00000	4.6250	1.0000
1	1.64285	0.92857	3.3853E-2	1.0000
2	1.60155	0.94922	9.5852E-4	1.0000
3	1.59606	0.95196	1.1934E-6	1.0000
4	1.59586	0.95206	6.2967E-10	1.0000
5	1.59586	0.95206	2.8435E-13	1.0000

While the equations are solved exactly, the Broyden matrix is actually quite different:

$$B(x^*) = \begin{bmatrix} 6.5493 & 2.2349 \\ 1.0 & 2.0 \end{bmatrix}$$

$$J(x^*) = \begin{bmatrix} 6.3834 & 1.9041 \\ 1.0 & 2.0 \end{bmatrix}$$

First Order Methods

These methods:

- do not evaluate or approximate the Jacobian matrix and are much simpler in structure.

- convergence is only at a linear rate, and this can be very slow.

- main application is for flowsheet convergence (ASPEN/Plus)

Consider fixed point form: $x = g(x)$

x and g(x) are vectors of n stream variables, e.g., x represents a guessed tear stream and g(x) is the calculated value after executing the units around the flowsheet.

Direct Substitution Methods

- simplest fixed point method
- define $x^{k+1} = g(x^k)$ with an initial guess x^0.
- convergence properties for the n dimensional case can be derived from the contraction mapping theorem (Dennis and Schnabel, 1983; p. 93).

For the fixed point function, consider the Taylor series expansion:

$$g(x_k) = g(x_{k-1}) + \left(\frac{\partial g}{\partial x}\right)^T_{x^{k-1}} (x_k - x_{k-1}) + \ldots$$

and if we assume that $\partial g/\partial x$ doesn't vanish, it is the dominant term near the solution, x*. Then

$$x^{k+1} - x^k = g(x^k) - g(x^{k-1}) = \left(\frac{\partial g}{\partial x}\right)^T_{x^{k-1}} (x^k - x^{k-1})$$

and for

$$x^{k+1} - x^k = \Delta x^{k+1} = \Gamma\, \Delta x^k \quad \text{with} \quad \Gamma = \left(\frac{\partial g}{\partial x}\right)^T$$

we can write:

$$|| \Delta x^{k+1}|| \leq \ ||\Gamma||\ || \Delta x^k||.$$

From this expression we can show a linear convergence rate, but the speed of these iterations is related to $||\Gamma||$. If we use the Euclidean norm, then $||\Gamma|| = |\lambda|^{max}$,

This leads to:

$$|| \Delta x^k|| \leq \ (|\lambda|^{max})^k\ || \Delta x^0||.$$

and a necessary and sufficient condition for convergence is that $|\lambda|^{max} < 1$. This relation is known as a *contraction mapping* if $|\lambda|^{max} < 1$.

Speed of convergence depends on how close $|\lambda|^{max}$ is to zero.

We can estimate the number of iterations (n_{iter}) to reach $||\Delta x^n|| \leq \delta$ (some zero tolerance), from the relation:

$$n_{iter} \geq \ln[\delta/|| \Delta x^0||]/\ln |\lambda|^{max}$$

For example, if $\delta = 10^{-4}$ and $\left\|\Delta x^0\right\| = 1$, we have the following iteration counts, for:

$$|\lambda|^{max} = 0.1, \quad n = 4$$
$$|\lambda|^{max} = 0.5, \quad n = 14$$
$$|\lambda|^{max} = 0.99, \quad n = 916$$

Relaxation (Acceleration) Methods

When $|\lambda|^{max}$ is close to one, direct substitution is limited and converges slowly.

Instead, alter the fixed point function g(x) to reduce $|\lambda|^{max}$.

$$x^{k+1} = h(x^k) \equiv \omega\, g(x^k) + (1 - \omega)\, x^k$$

ω is chosen adaptively depending on the changes in x and g(x).

Two common fixed point methods for recycle convergence are:

- *dominant eigenvalue* method (Orbach & Crowe, 1971)
- *Wegstein (1958) iteration* method

Dominant eigenvalue method (DEM)

- Obtain an estimate of $|\lambda|^{max}$ by monitoring the ratio:

$$|\lambda|^{max} \approx \frac{\left\|\Delta x^{k}\right\|}{\left\|\Delta x^{k-1}\right\|}$$

  after, say, 5 iterations.

- Choose the relaxation factor ω to minimize $|\lambda|^{max}$ of the modified fixed point form.

- To choose an optimum value for ω, assume that $\lambda^{min} > 0$ and that $\lambda^{min} \approx \lambda^{max}$ ==> $\omega^* = 1/(1 - |\lambda|^{max})$

<u>Notes</u>

- if this assumption is violated and the minimum and maximum eigenvalues of Φ are far apart, DEM may not converge.
- extended to the *Generalized Dominant Eigenvalue Method* (GDEM) (Crowe and Nishio, 1975) where several eigenvalues are estimated and are used to determine the next step - overcomes the assumption that $\lambda^{min} \approx \lambda^{max}$.

Wegstein method

Obtains relaxation factor ω, by applying a secant method independently to each component of x. From above we have for component x_i:

$$x_i^{k+1} = x_i^k - f_i(x^k)\,[x_i^k - x_i^{k-1}]/[f_i(x^k) - f_i(x^{k-1})]$$

Define

$$f_i(x^k) = x_i^k - g_i(x^k)$$
$$s_i = [g_i(x^k) - g_i(x^{k-1})]/[x_i^k - x_i^{k-1}]$$

we have:

$$x_i^{k+1} = x_i^k - f_i(x^k)\,[x_i^k - x_i^{k-1}]/[f_i(x^k) - f_i(x^{k-1})]$$

$$= x_i^k - \frac{\{x_i^k - g_i(x_i^k)\}\,[x_i^k - x_i^{k-1}]}{[x_i^k - g_i(x^k) - x_i^{k-1} + g_i(x^{k-1})]}$$

$$= x_i^k - \frac{\{x_i^k - g_i(x_i^k)\}\,[x_i^k - x_i^{k-1}]}{[x_i^k - x_i^{k-1} + g_i(x^{k-1}) - g_i(x^k)\,]}$$

$$= x_i^k - \{x_i^k - g_i(x^k)\}/[1 - s_i\,]$$

$$= \omega_i\, g(x^k)_i + (1 - \omega_i)\, x_i^k$$

where $\omega_i = 1/[1 - s_i\,]$.

Approach works well on flowsheets where the components do not interact strongly (e.g., single recycle without reactors).

To ensure stable performance:

- Relaxation factors for both DEM and the Wegstein method are normally bounded

- Algorithm for fixed point methods can be summarized by:

    1) Start with x^0 and $g(x^0)$

    2) Execute a fixed number of direct substitution iterations (usually 2 to 5) and check convergence at each iteration.

    3) *Dominant Eigenvalue Method:* Apply this acceleration with a bounded value of ω to find the next point and go to 2.

    Wegstein: Apply this acceleration with a bounded value of ω_i to find the next point. Iterate until convergence.

Example

Solve the fixed point problem given by:

$$x_1 = 1 - 0.5 \exp(0.7(1 - x_2) - 1)$$
$$x_2 = 2 - 0.3 \exp(0.5(x_1 + x_2))$$

using a direct substitution method, starting from $x_1 = 0.8$, and $x_2 = 0.8$. Estimate the maximum eigenvalue based on the sequence of iterates.

Using direct substitution, $x^{k+1} = g(x^k)$, we obtain the following iterates:

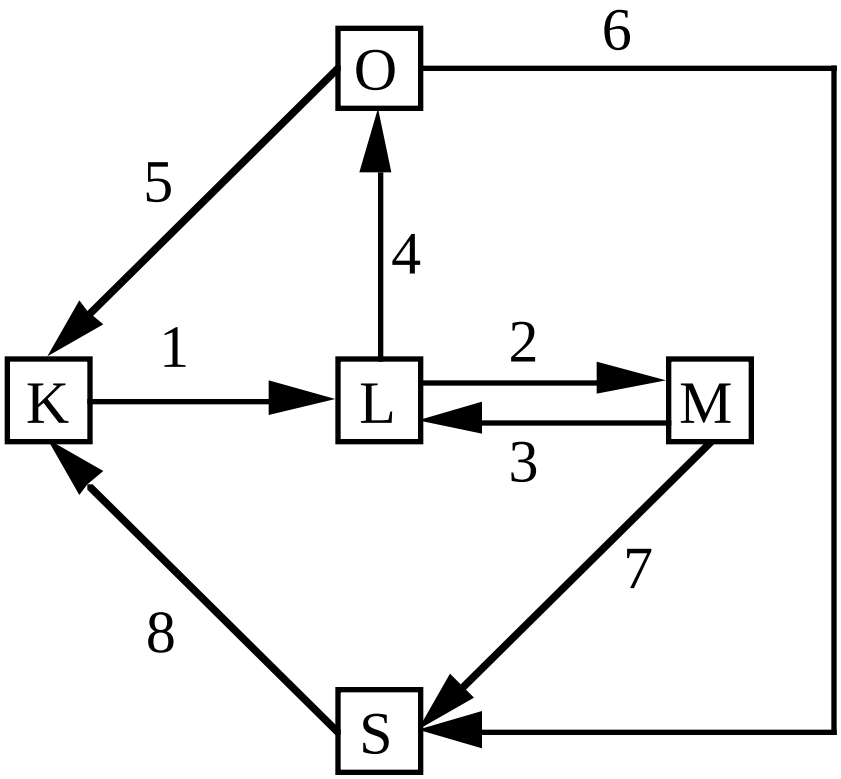
k	x_1^k	x_2^k
0	0.8	0.8
1	0.7884	1.3323
2	0.8542	1.3376
3	0.8325	1.1894
4	0.8389	1.1755
5	0.8373	1.1786
6	0.8376	1.1780

From these iterates, we can estimate the maximum eigenvalue from:

$$|\lambda|\max = \left\|x^5 - x^4\right\| / \left\|x^4 - x^3\right\| = 0.226$$

Recycle Tearing Algorithms

Consider the following flowsheet and recycle system:



- Find a set of tear streams that breaks all recycle loops
- For small problems a good tear set can be seen by inspection.
- For larger problems a systematic procedure needs to be applied.
- The choice of tear set greatly affects the performance of fixed point algorithms.

General tearing approach

- treat tear set selection as an integer program with binary (0/1) variables (Pho and Lapidus, 1973)
- can weight choice of tear streams
- loop finding constraints leading to a *set covering problem*

Loop Finding

Start with any unit in the partition, for example unit K.

```
K -(1)->L-(2)->M-(3)->L
```

We note that unit L repeats and the two streams, 2 and 3, which connect the two appearances of unit L are placed on a list of loops:

List: {2,3}

We then start with the unit just before the repeated one and trace any alternate paths from it.

```
K -(1)->L-(2)->M-(3)->L
                |
                 -(7)->S-(8)->K
```

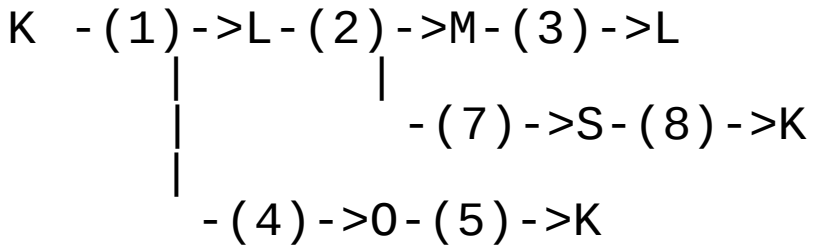
Now K repeats and we place streams {1,2,7,8} on the list of loops.

List: {2,3}, {1,2,7,8}

Back up to S and look for an alternate path leaving from it, we find there is none.

Back up to unit M, we again there is no additional unexplored paths.

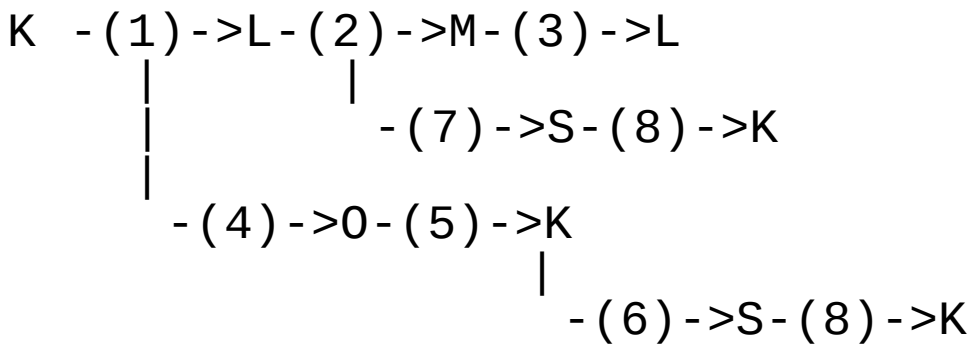
Back up to L we find another path and this is given by:.



K repeats and we place {1,4,5} on the list of loops.

List : {2,3}, {1,2,7,8}, {1,4,5}

Back up to unit O on the last branch we can identify alternate paths which include:



Again K repeats and we place {1,4,6,8} on the final list of loops.

List : {2,3}, {1,2,7,8}, {1,4,5}, {1,4,6,8}

There appear to be no additional loops.

Loop incidence array for partition

Strm	1	2	3	4	5	6	7	8
Loop								
1		x	x					
2	x	x					x	x
3	x			x	x			
4	x			x		x		x

Initialize a loop matrix, A, with elements:

a_{ij} = 1 if stream j is in loop i
0 otherwise

The structure of this matrix is identical to the loop incidence array. We define the selection of tear streams through an integer variable, y_j, for each stream j: optimal values of these variables determine:

y_j = 1 if stream j is a tear stream
0 otherwise

To ensure that each recycle loop is broken 'optimally' at least once by the tear streams, we write the set covering problem is given by:

$$\underset{y_j}{\text{Min}} \quad \sum_{j=1}^{n} w_j y_j$$

$$\text{s.t.} \quad \sum_{j=1}^{n} a_{ij} y_j \geq 1 \qquad i = 1, L$$

$$y_j = \{0, 1\}$$

The weight w_j to the cost of tearing stream j.

Popular choices for weights are:

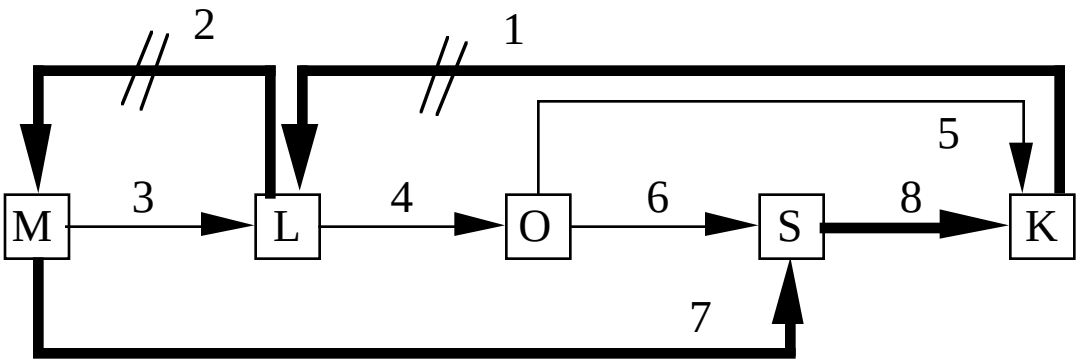
- Choose $w_j = 1$, Barkeley and Motard (1972).
- Choose $w_j = n_j$ where n_j is the number of variables in the jth tear stream, Christensen and Rudd (1969).
- Choose $w_j = \Sigma_i a_{ij}$ - number of loops that are broken by the tear stream j. Breaking a loop more than once causes a 'delay' in the tear variable iteration for the fixed point algorithms and much poorer performance, Upadhye and Grens (1975) and Westerberg and Motard (1981).

Solution to this integer problem is combinatorial (NP Hard) and an upper bound on the number of alternatives is 2^n cases.

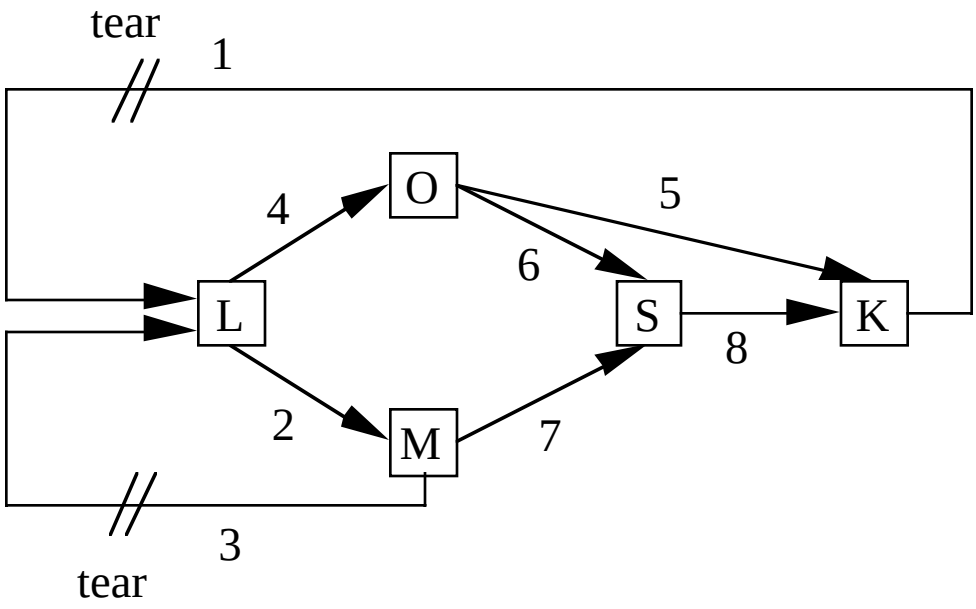
Simple reduction rules can make this problem and the resulting solution effort much smaller.

To facilitate the solution, the most common approach is a branch and bound search.

For solution procedure, see Chapter 8, Biegler et al. (1997)



Solution where all weights are equal



Minimize the number of times the loops are torn

These solutions are also nonunique and similarly converging tear sets can be derived.

Effect of Tearing Strategies on Newton-type Methods

What if a Newton or quasi-Newton algorithm is applied to converge a modular flowsheet (e.g., SQP, Broyden, etc., in Aspen/Plus)?

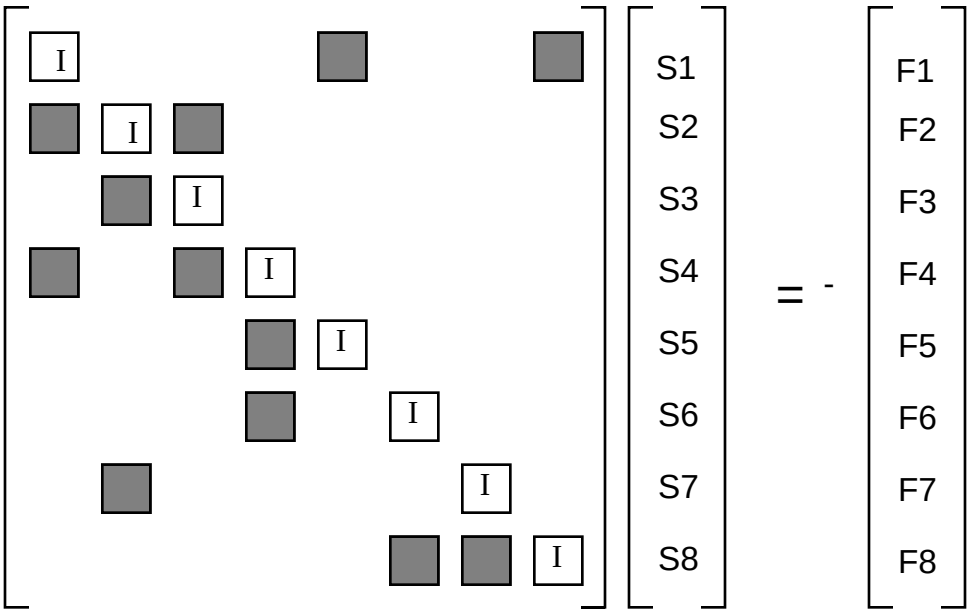
Rewrite these equations as:

$$x = g(x) \quad \text{or} \quad f(x) = x - g(x) = 0$$

x are tear streams and g(x) refers to the calculated value.

Solve using Newton-Raphson or Broyden iterations applied to f(x) = 0.

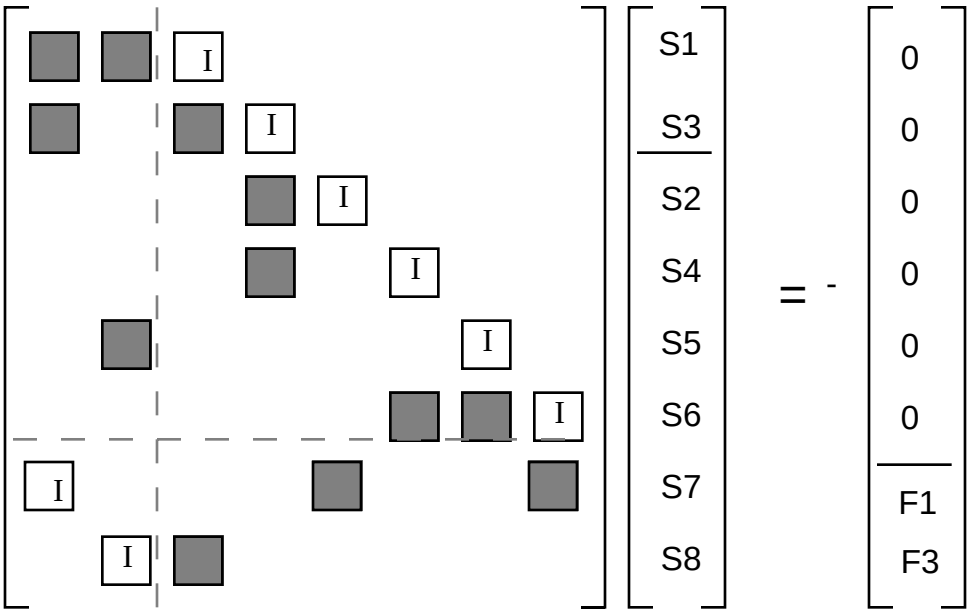
Solving the recycle equations by tearing *all of the streams* in the recycle loops. This leads to:



Linearized equations for flowsheet

Instead, permute the remaining stream variables and equations to block lower triangular form.

Choose S1 and S3 as tear streams, and hold these fixed, it is easy to see that the diagonal streams can be calculated directly from streams that are determined from S1 and S3.



Linearized equations with S1 and S3 as tear streams

Therefore, we see that for Newton or Broyden methods, it is desirable to choose the minimum number of *stream variables* that breaks all recycle loops (fewer equations)

As long as all of the recycle loops are torn, the choice of tear streams has little effect on the convergence rate of either the Broyden or Newton methods.

If all equations are linear, any tear set selection requires only one iteration with Newton (or appropriate Broyden) method.