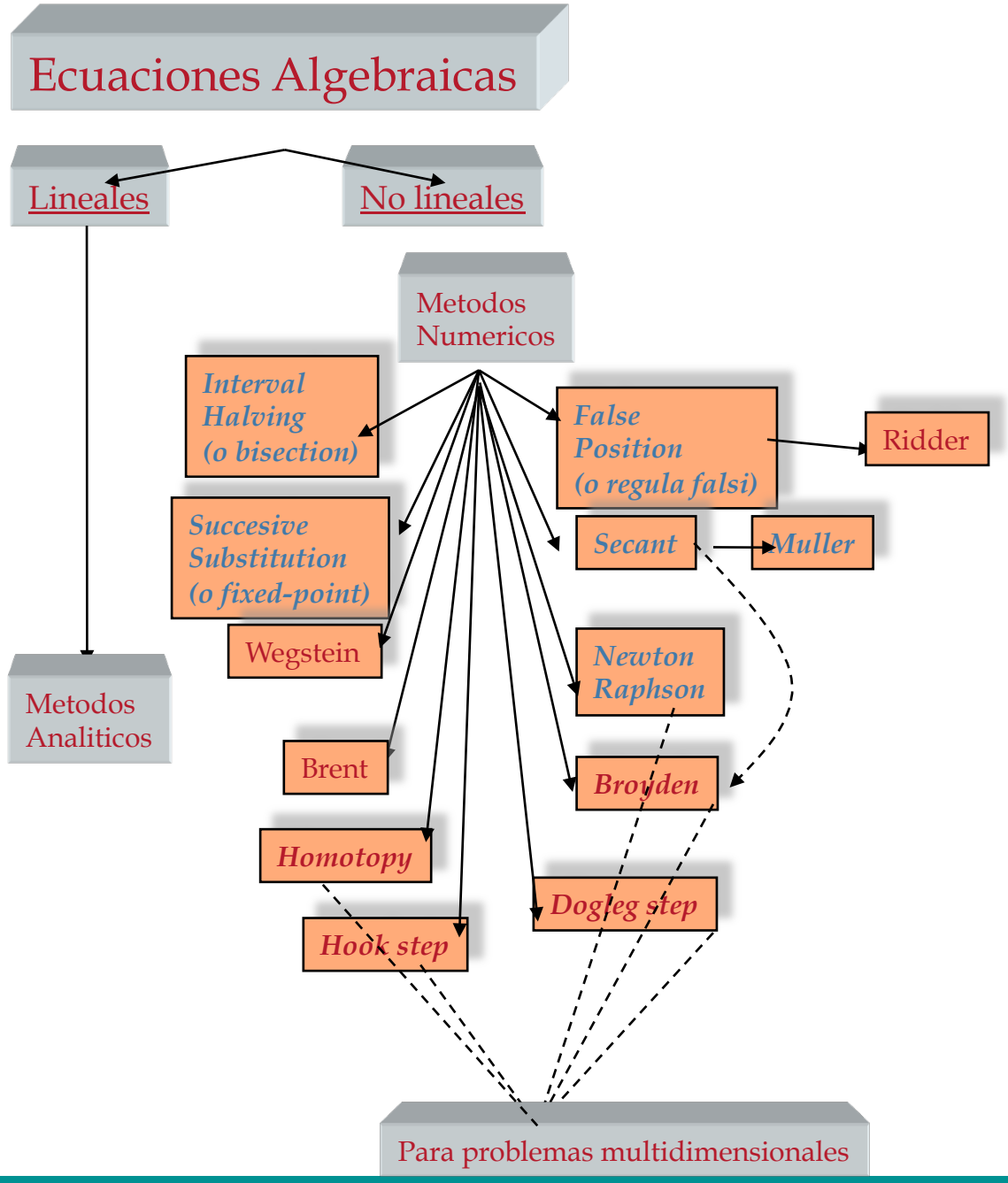
3. Métodos de resolución

Ecuaciones algebraicas lineales

Ecuaciones algebraicas no lineales

- Métodos para una variable
- Métodos para multivariable


Ecuaciones Algebraicas
Lineales
No lineales
Metodos Numericos
Metodos Analiticos
Interval Halving (o bisection)
False Position (o regula falsi)
Ridder
Succesive Substitution (o fixed-point)
Secant
Muller
Wegstein
Newton Raphson
Brent
Broyden
Homotopy
Dogleg step
Hook step
Para problemas multidimensionales

Ecuaciones algebraicas no lineales

Objetivo

Sea f(x) una función ***no lineal*** en x. Hallar el valor de x, x^*, tal que se cumple $f(x^*)=0$.

x^* se suele denominar el cero o raíz de f(x)

x^* se puede determinar por medios analíticos (solución exacta) o por medios numéricos (solución aproximada)

La elección del método numérico depende del problema a resolver (estructura del problema, tipo de ecuaciones, precisión requerida, rápidez del cálculo,....).

Por tanto no existe un mejor método universalmente aplicable.

Tipos de métodos

Métodos acotados (bracketing methods)

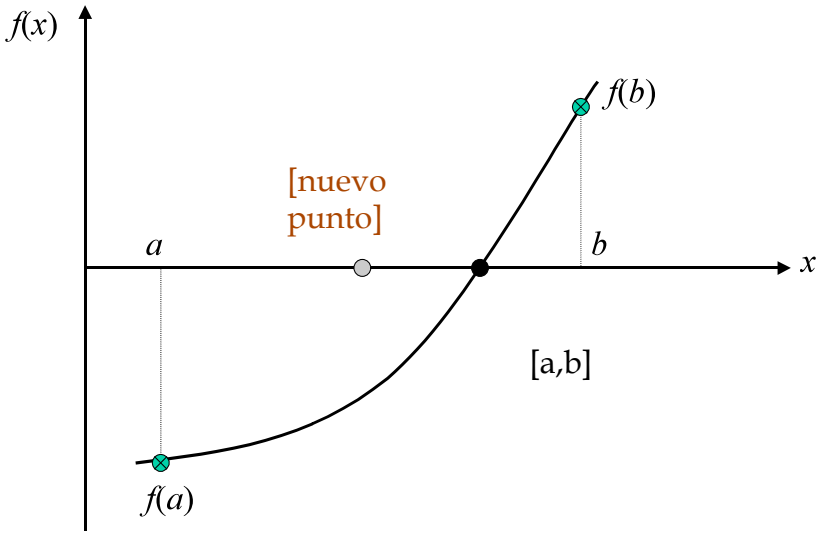
Métodos abiertos (open methods)

Métodos acotados

Base: Una función cambia de signo en la proximidad de una raíz

•Una raíz está acotada en el intervalo [a,b] si el signo de f(a) es diferente al signo de f(b)

Método de la bisección (o intervalo medio)



Algoritmo

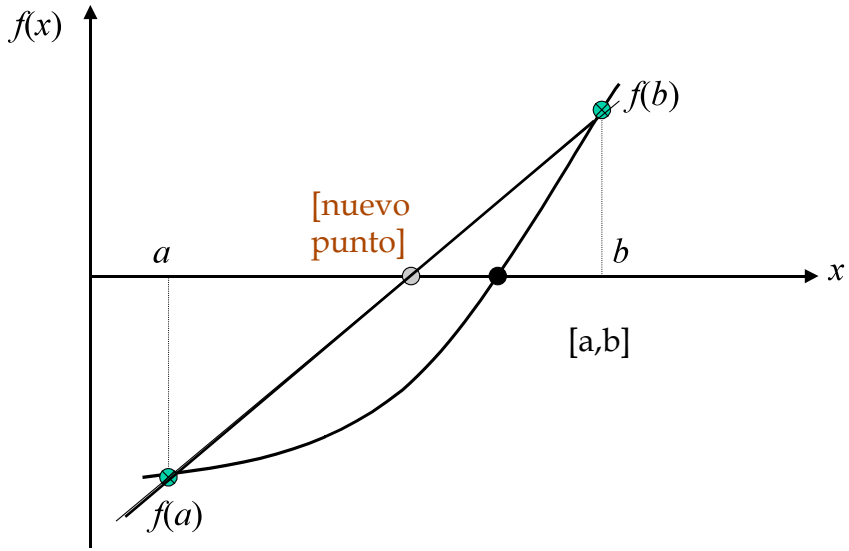
1. Selecciona un intervalo [a,b] donde halla un cero
2. Calcula el punto medio como nuevo punto
3. Comprueba si hay cambio de signo en [a,p] o en [p,b]. Comprobación: f(a)*f(p).
4. Si el producto es cero, entonces p es una **raíz**. Si no es cero volver al punto 2.

Ejemplo Método bisección (Intervalo medio)

$$x^2 - 4 = 0$$

$$[0,6] \rightarrow [3,6]$$

Método de la posición falsa



Algoritmo

1. Selecciona un intervalo [a,b] donde halla un cero
2. Calcula un punto intersección como nuevo punto
$$\frac{f(a)}{m-a}=\frac{f(b)}{m-b}\Rightarrow m=b-\frac{f(b)[a-b]}{f(a)-f(b)}$$
3. Comprueba si hay cambio de signo en [a,p] o en [p,b]. Comprobación: f(a)*f(p).
4. Si el producto es cero, entonces p es una **raíz**. Si no es cero volver al punto 2.

Ejemplo método de la posición falsa (Regula Falsi)

$$x^2 - 4 = 0$$

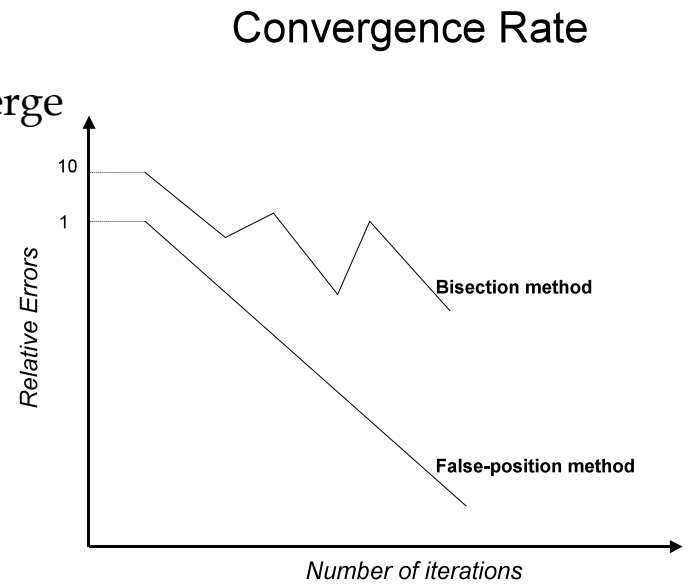
Comparación entre ambos métodos.

Similaridades:

- Ambos métodos necesitan *DOS valores iniciales*
- Requieren un procedimiento para *determinar el cambio de signo.*
- *Acaban convergiendo* a la raíz con cierta tolerancia

Diferencias:

- El cálculo del nuevo punto estimado se hace con *diferentes estrategias*
- *En general* el método de la posición falsa converge más rápido que el de la bisección.



Métodos abiertos

- Emplean una aproximación funcional para obtener el nuevo valor estimado de la raíz (línea recta, cuadrática, polinomio)
- Métodos:
  - Punto-fijo (sustitución sucesiva o directa)
  - Newton-Raphson (línea recta empleando información del gradiente)
  - Secante (línea recta empleando dos puntos)
  - Muller (aprox. cuadrática empleando tres puntos)

Metodos acotados vs. Métodos abiertos

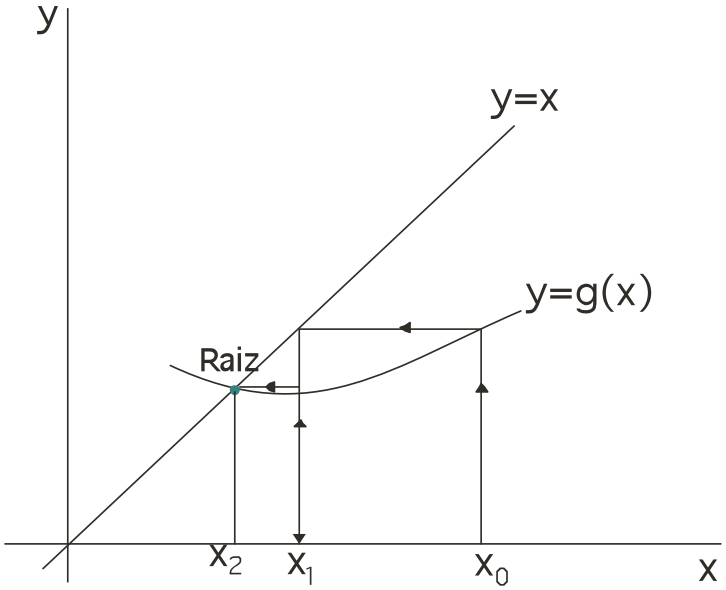
Métodos acotados

La raíz está situada en un intervalo (necesita dos puntos). Acaba convergiendo dentro de una tolerancia.

Métodos abiertos

Sólo emplean un punto inicial (o dos puntos que no tienen por qué contener a la raíz) y una fórmula para encontrar la raíz. No siempre convergen, pero cuando lo hacen son mucho más rápidos que los métodos acotados.

Sustitución sucesiva



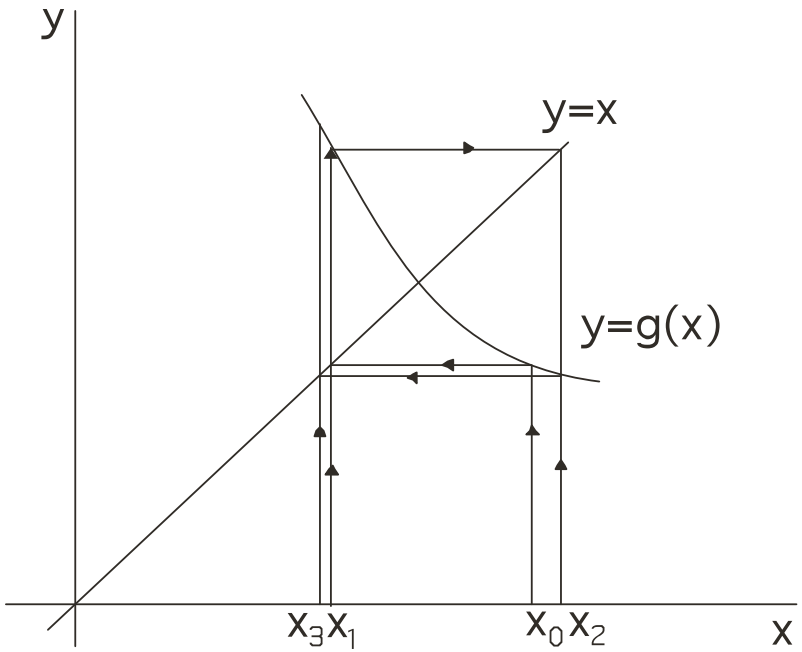
Problema f(x)=0

1. Transformar a x=g(x)
2. Seleccionar un punto inicial x_0
3. Calcular nuevo valor $x_{i+1}=g(x_i)$
4. Repetir hasta llegar a la tolerancia requerida

Si:

$|g'(x)|<1$ El algoritmo *converge* linealmente

$|g'(x)|>=1$ El algoritmo *diverge*



Ejemplo Método punto fijo (Sustitución directa/sucesiva)

$$x^2 - 4 = 0$$

$x = x - 4 + x^2$

$x = x(x - 3)$

$g(x) = x(x - 3)^2$

$g'(x) = 3x - 3 = 3 - 3 = 0 < 1$

$x_{k+1} = x_k(x_k - 3)^2$ ¿Converge?

$x_0 = 1$

$x_1 = 1(1 - 3)^2 = -2$

$x_2 = -2((-2) - 3)^2 = -2$

$x_0 = 3$

$g'(x) = 3x - 3 = 3 * 9 - 3 = 24 > 1$

$x_1 = 3(3 - 3)^2 = 18$ ¿Converge?

$x_2 = 18(18 - 3)^2 = 5778$

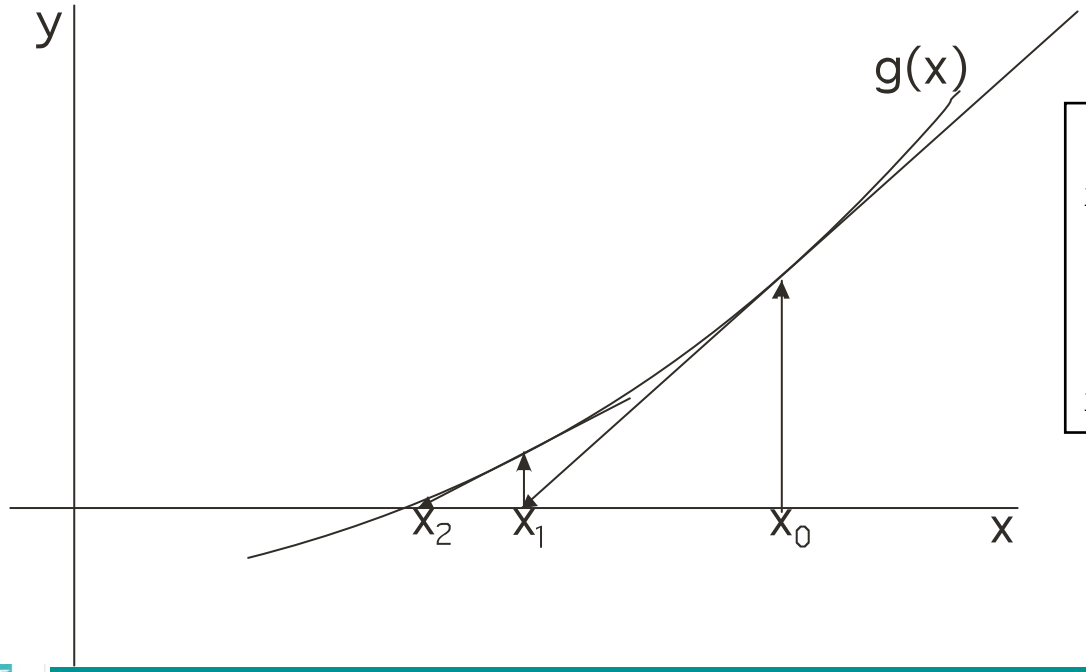
Newton Raphson

Problema g(x)=0

1. Seleccionar un punto inicial x_0
2. Calcular $g(x_i)$ y $g'(x_i)$
3. Aplicar la tangente en ese punto y en el corte con el eje de abcisas tenemos el nuevo punto estimado

$$x_{i+1}=x_i-\frac{g(x_i)}{g'(x_i)}$$

4. Repetir hasta llegar a la tolerancia requerida



- Necesita conocer la derivada de la función
- Convergencia cuadrática (rápida)
- Puede no converger (depende de la función y de la **estimación inicial**)

Ejemplo Método de Newton

$$x^2 - 4 = 0$$

$X_0=1$ $\qquad$ $X_0=3$

$$x_1 = x_0 - \frac{(x_0^{\,2} - 4)}{2x_0} \qquad x_1 = x_0 - \frac{(x_0^{\,2} - 4)}{2x_0}$$

$$x_1 = 1 - \frac{(1^2 - 4)}{2*1} = 1.5 \qquad x_1 = 3 - \frac{(3^2 - 4)}{2*3} = 2.1666$$

$$x_2 = 1.5 - \frac{(1.5^2 - 4)}{2*1.5} = \frac{23}{12} \qquad x_2 = 2.1666 - \frac{(2.1666^2 - 4)}{2*2.1666} = 2,006$$

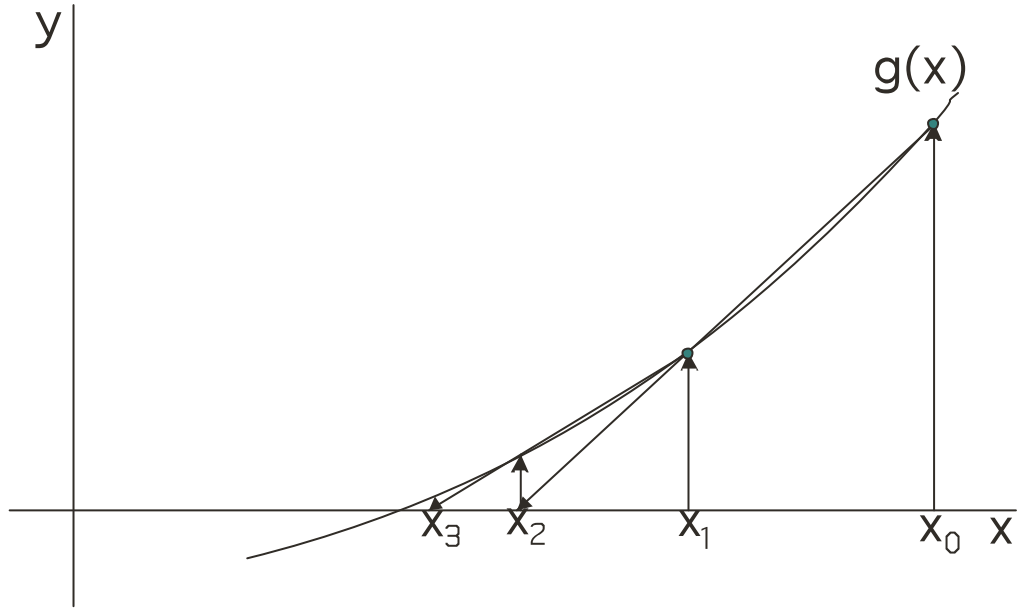
Secante

Problema g(x)=0

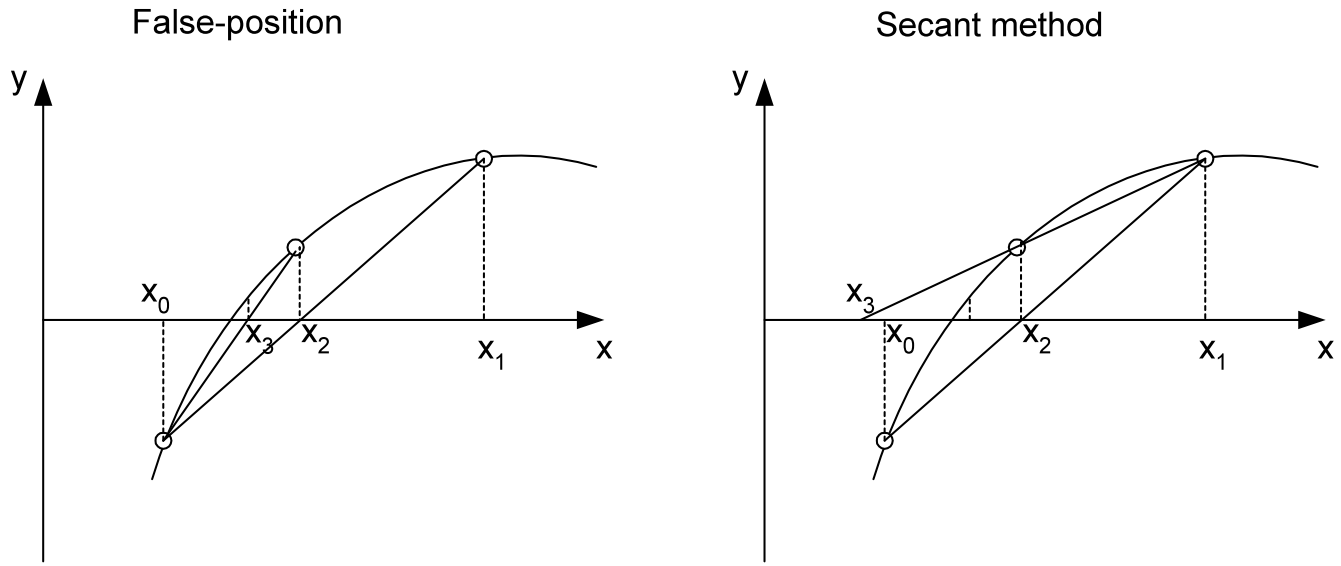
1. Seleccionar dos puntos iniciales x_0, x_1
2. Calcular la recta que pasa por esos puntos
3. El corte con el eje de abcisas da el nuevo punto estimado. Volver a calcular la recta.

$$x_{i+1}=x_i-\frac{x_{i+1}-x_i}{g(x_{i+1})-g(x_i)}\,g(x_{i+1})$$

4. Repetir hasta llegar a la tolerancia requerida

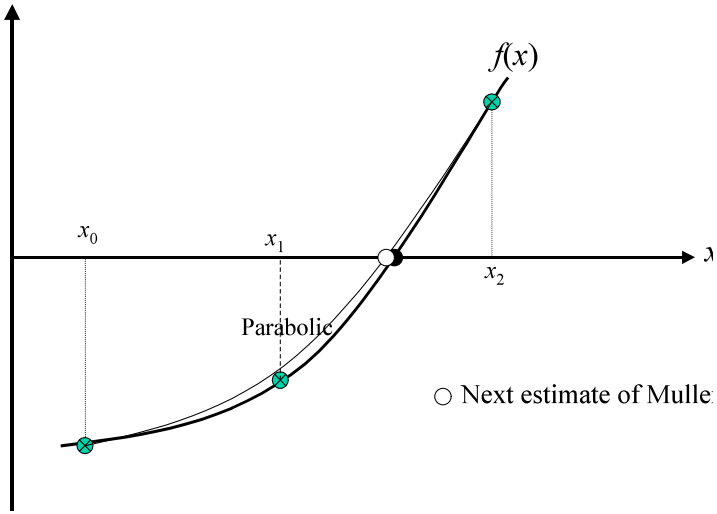


- No Necesita conocer la derivada de la función (la aproxima).
- Necesita dos puntos iniciales.
- Puede no converger.



La primera iteración da el mismo resultado, luego cada uno obtiene un nuevo punto estimado diferente

Muller's Method



Muller's Algorithm

1. Select three points $[x_o, f(x_o)]$; $[x_1, f(x_1)]$ and $[x_2, f(x_2)]$
2. Compute the coefficients *a*, *b* and *c* *
3. Apply a quadratic formula to calculate the *next estimate* of the zero

$$x_3 = x_2 - \frac{2c}{b \pm \sqrt{b^2 - 4ac}}$$

Real and complex root Can be located

4. Repeat step 2-3 until satisfying the convergent criteria

Which point is discarded in Muller Algorithm

- Two general strategies are typically used:
  - If only real roots are being located, we choose the two original points that are nearest the new root estimate, x_3
  - If both real and complex roots are being evaluated, a sequential approach is employed. That is just like the secant method, x_1, x_2 and x_3 take place of x_0, x_1 and x_2.

* Se computan obligando a que g(x) pase por los 3 puntos seleccionados.

$$g(x) = ax^2 + bx + c \qquad i$$

Sistemas de ecuaciones algebraicas no lineales

$$
\begin{aligned}
&f_1(x_1, x_2, x_3, \ldots, x_n) = 0 \\
&f_2(x_1, x_2, x_3, \ldots, x_n) = 0 \\
&f_3(x_1, x_2, x_3, \ldots, x_n) = 0 \\
&\vdots \\
&f_n(x_1, x_2, x_3, \ldots, x_n) = 0
\end{aligned}
$$

$$F(X) = 0$$

Sustitución sucesiva acelerada

La sobrerelajación es una técnica para acelerar la convergencia de métodos iterativos en la solución de ecuaciones lineales. La idea es aplicarlo al método de sustitución sucesiva.

Se dan pesos a los valores anteriores y a los previos con el fin de dar pasos "mayores" hacia la solución.

$$X^{k+1}=qX^{k}+(1-q)g(X^{k})$$

- q=0 sustitución sucesiva
- q<0 aceleración de la convergencia
- 0<q<1 estabilización de la convergencia por amortiguamiento

WEGSTEIN'S METHOD

Idea: treat each variable with the one dimensional secant method by driving it with a uniquely associated function. This implies that interactions with other variables are neglected.

Given: values of argument and function from two successive substitution steps (i.e., $\mathbf{x}^k = \mathbf{g}(\mathbf{x}^{k-1})$):

$$\left\{\mathbf{x}^{k-1}, \mathbf{g}(\mathbf{x}^{k-1})\right\}$$
$$\left\{\mathbf{x}^{k}, \mathbf{g}(\mathbf{x}^{k})\right\}$$

Calculate (for each variable j):

a) $$s_j = \frac{x_j^k - x_j^{k-1}}{g_j(\mathbf{x}^k) - g_j(\mathbf{x}^{k-1})}$$ (inverse of the slope)

b) $$q_j = \frac{1}{1 - s_j}$$

c) $$x_j^{k+1} = q_j x_j^{k-1} + (1 - q_j) x_j^k$$

This is also an overrelaxation method, but an individual overrelaxation factor is derived for each variable.

If Wegstein is applied to every step, it is also likely to suffer from instability (if q_j large and negative).

En los problemas de diagrama de flujo las variables de iteración no están todas muy acopladas ni todas desacopladas, nos podemos encontrar:

1) *Todas las especies débilmente acopladas a través de equilibrio L-V (no ideal)*

2) *Especies muy acopladas si participan en una reacción*

3) *Si hay más de una corriente de rasgado, hay un fuerte acoplamiento entre los flujos de cada corriente de rasgado.*

Cuando las variables están desacopladas o débilmente acopladas es necesario un parámetro de aceleración diferente para cada variable.

BOUNDED WEGSTEIN'S METHOD

In order to control stability, the Bounded Wegstein Method is used in most process simulators:

a) compute q_j as above

b) bound q_j in the interval $[-n,0]$.

Upper bound ensures the method defaults to successive substitution.

Lower bound $-n$ in the range -2 to -5 to control stability.

Alternatively, relax the lower bound, but apply Wegstein acceleration only every few successive substitution steps. ASPENPLUS has options to control all these parameters.

El método de Wegstein es muy bueno para particiones en las que hay una única corriente de rasgado. O cuando hay reciclo sin estar los componentes muy acoplados (presencia de reacción)

Newton Raphson

- En lugar de la derivada emplea el jacobiano (matriz de derivadas parciales)
- La estimación del nuevo conjunto de raíces se computa mediante la siguiente ecuación:

$$X_{i+1} = X_i - J^{-1}(X_i)F(X_i)$$

Jacobiano

$$J(X_i) = \begin{pmatrix} \left.\frac{\partial f_1}{\partial x_1}\right|_{x_{1,i}} & \left.\frac{\partial f_1}{\partial x_2}\right|_{x_{2,i}} & \cdots & \left.\frac{\partial f_1}{\partial x_n}\right|_{x_{n,i}} \\ \left.\frac{\partial f_2}{\partial x_1}\right|_{x_{1,i}} & \left.\frac{\partial f_2}{\partial x_2}\right|_{x_{2,i}} & \cdots & \left.\frac{\partial f_2}{\partial x_n}\right|_{x_{n,i}} \\ \vdots & \vdots & \ddots & \vdots \\ \left.\frac{\partial f_n}{\partial x_1}\right|_{x_{1,i}} & \left.\frac{\partial f_n}{\partial x_2}\right|_{x_{2,i}} & \cdots & \left.\frac{\partial f_n}{\partial x_n}\right|_{x_{n,i}} \end{pmatrix}$$

¿Cómo resolverías la ecuación anterior sin tener que calcular la inversa de la matriz jacobiana?

Newton Raphson- procedimiento de resolución

$$X_{i+1} = X_i - J^{-1}(X_i)F(X_i)$$

¡Resuelve un sistema de ecuaciones lineales!

$$J(X_i)\Delta(X) = -F(X_i)$$

$AXb=$

$$\Delta X = (X_{i+1} - X_i)$$
$$X_{i+1} = X_i + \Delta X$$

Actualiza el valor hasta que ΔX es tan pequeño como se haya requerido

Ventajas:
a) Buena convergencia (cuadrática)
b) Bueno para diagramas de flujo con mucha interacción, ya que esta interacción se tiene en cuenta en el Jacobiano.
Desventajas:
a) Requiere unas estimaciones iniciales buenas
b) Como las funciones no se conocen explícitamente, el Jacobiano se aproxima de forma numérica.

Método de Broyden

- No calcula el jacobiano, lo aproxima empleando valores previos de x y f(x).
- W es la aproximación a la negativa de la inversa del jacobiano.
- Es una extensión del método de la secante (o método quasi-Newton)

$$X_{i+1} = X_i - W^i F(X_i)$$

$$p^i = X^{i+1} - X^i$$

$$y^i = f(X^{i+1}) - f(X^i)$$

$$\min \left\| W^{i+1} - W^i \right\|$$

$$W^i p^i = y^i$$

$$W^{i+1} = W^i - \frac{(p^i + W^i y^i) p^{i^T} W^i}{p^{i^T} W^i y^i}$$

Ventajas:

- *Sólo requiere una "pasada" por el diagrama de flujo en cada*
- *Tiene en cuenta de forma aproximada la interacción entre variables. Bueno para diagramas de flujo con alta interacción.*

Desventajas:

- *Convergencia más lenta que Newton. Necesita más pasos que el método de Newton pero el costo computacional de cada método es*

Muy utilizado si el número de ecuaciones no es muy grande (<100)
Utilizado para convergencia de reciclos en diagramas de flujo.

• Para los métodos de Newton o Broyden es deseable escoger el mínimo número de variables de corriente que rasgan todos los lazos (es decir buscar el menor número de ecuaciones).
• Si todos los bucles están rasgados la elección de las corrientes de rasgado no influye mucho en la convergencia de estos dos métodos.

MATLAB Built in Function (1)

- The built in **fzero** function (*Optimization Toolbox*) is a hybri method that combines bisection, secant and reverse quadr interpolation to a find the root of $f(x) = 0$
  - Syntax:

    x = **fzero**('fun',x0)

    X0 can be scalar or a two element vector
    - If x0 is a scalar, **fzero** tries to create its own bracket
    - If x0 is a two element vector, **fzero** uses the vector as a bracket

MATLAB Built in Function (2)

- The built in **fsolve** function (*Optimization Toolbox*) is an M-file function to solve a system of nonlinear equations:

$$\begin{aligned} f_1(x_1, x_2, \ldots, x_n) &= 0 \\ f_2(x_1, x_2, \ldots, x_n) &= 0 \\ &\vdots \\ f_n(x_1, x_2, \ldots, x_n) &= 0 \end{aligned}$$

  - Syntax:

    x = **fsolve**('fun',x0)

    x0 is a vector of initial estimate of the roots

fsolve

Solve a system of nonlinear equations

$$F(x) = 0$$

for *x*, where *x* is a vector and *F(x)* is a function that returns a vector value.

Syntax

```
x = fsolve(fun,x0) x = fsolve(fun,x0,options)
x = fsolve(fun,x0,options,P1,P2, ... )
[x,fval] = fsolve(...)
[x,fval,exitflag] = fsolve(...)
[x,fval,exitflag,output] = fsolve(...)
[x,fval,exitflag,output,jacobian] = fsolve(...)
```

Métodos de resolución y Aspen Plus

Specifying Default Methods

You can specify the numerical methods to be used by the system-generated convergence blocks. See Convergence Methods for information on the numerical methods.

To specify the numerical methods to be used by the system-generated convergence blocks:

1 From the Data menu, point to Convergence, the Conv Options.

2 Select the Default Methods sheet.

3 You can specify the numerical methods to be used by the convergence blocks.

The following parameters are available on the Default Methods sheet:

Field	Default	To specify the convergence method for system-generated
Tears	Wegstein	Tear convergence blocks The other methods available are Direct, Broyden, and Newton.
Single Design Spec	Secant	Single design-spec convergence blocks The other methods available are Broyden and Newton.
Multiple Design Specs	Broyden	Multiple design-spec convergence blocks The other method available is Newton.
Tears & Design Specs	Broyden	Combined tears and design-specs convergence blocks The other method available is Newton.
Optimization	SQP	Optimization convergence blocks

Use this method	To converge
BROYDEN or NEWTON	Tear streams; two or more design specifications; or tear streams and design specifications simultaneously. Use when the recycle loops and/or design specifications are highly interrelated. Use Newton when Broyden is unable to converge.
COMPLEX	Optimization with inequality constraints
DIRECT	Tear streams by simple direct substitution. Convergence may be slow, but sure.
SECANT	Single design specifications. Recommended for design specification convergence blocks.
SQP	Sequential quadratic programming. Optimization with any combination of tear streams, equality constraints, and inequality constraints.
WEGSTEIN	Tear streams. You can apply Wegstein to any number of streams simultaneously. Recommended tear stream convergence method.

WEGSTEIN Method

The classical bounded Wegstein method is usually the quickest and most reliable method for tear stream convergence. It is an extrapolation of Direct substitution iteration. Interactions between variables are ignored; therefore, it does not work well when variables are strongly coupled.

Wegstein method can only be used for Tear streams. It is the default method for Aspen Plus tear stream convergence. Apply it to any number of streams simultaneously. You can control the Wegstein bounds and the frequency of acceleration.

Normally, you should use an Upper Bound of the Wegstein acceleration parameter of 0. If iterations move the variables slowly toward convergence, smaller values of the lower bound of the Wegstein acceleration parameter (perhaps -25 or -50) may give better results. If oscillation occurs with direct substitution, values of the lower and upper bounds between 0 and 1 may help.

DIRECT Method

With direct substitution, convergence is slow but sure. It is available for those rare cases where other methods may be unstable. Direct substitution can also make it easy to identify convergence problems, such as component build-up in the system. Direct substitution is equivalent to Wegstein with lower bound=upper bound=0.

Secant Method

Secant is the secant linear approximation method, with higher order enhancements. You can select a bracketing/interval halving option. Select this option whenever the function is discontinuous, non-monotonic, or flat over a region. Bracketing will eliminate the flat region and switch back to Secant method if possible.

You can use Secant for converging single design specifications. Secant is the default method for design specification convergence, and is recommended for user-generated convergence blocks.

BROYDEN Method

The Broyden method is a modification of Broyden's quasi-Newton method. The Broyden method is similar to the Newton method, but it uses approximate linearization. This approximation makes Broyden faster, but occasionally not as reliable, as the Newton method.

Use Broyden to converge tear streams, two or more design specifications, or tear streams and design specifications simultaneously. Broyden is useful for multiple tear streams and/or design specifications, tear variables that are highly interdependent, or recycle loops and design specifications so interrelated that nesting is impractical. When converging both tear streams and design specifications, you can specify that tear streams be converged or partially converged first. The simultaneous convergence of both tear streams and design specifications then follows.

NEWTON Method

NEWTON is an implementation of the modified Newton method for simultaneous nonlinear equations. Derivatives are calculated only when the rate of convergence is not satisfactory. The implementation allows bounds on the variables, and includes a line search for improved stability. NEWTON is useful when the recycle loops and/or design specifications are highly interrelated, but convergence is not achieved using the Broyden method. Numerical derivatives are calculated frequently. Use NEWTON for tear streams only when the number of components is small or when convergence cannot be achieved by the other methods. When converging both tear streams and design specifications, you can specify that tear streams be converged or partially converged first. The simultaneous convergence of both tear streams and design specifications then follows.

Strategies for Flowsheet Convergence

Often a flowsheet can be converged without changing any convergence parameters.

Some general guidelines are:

- Start small. Make sure that individual blocks and elements of a flowsheet behave as expected, before slowly combining them into a larger simulation. A sensitivity block is useful for determining the results of other blocks under a range of conditions.
- Start with the simplest blocks possible. For example, converge the flowsheet with a simple HeatX before switching it to a rigorous HeatX.
- Give good initial guesses. Make sure the flowsheet starts converging from a reasonable point. Aspen Plus gives tear streams a default value of zero, which can cause problems. If possible, select a tear stream that remains relatively constant.
- Check physical properties. Make sure they are calculated correctly in the entire operating range of the simulation.
- Know how your flowsheet responds. Check the behavior of blocks and design specifications using sensitivity analysis. Look for discontinuities and flat regions that could cause convergence difficulties.

Some other general strategies for tear stream convergence are:

- Provide a good initial guess for the Tear stream on the Stream form.
- Select a Tear stream that will not vary a great deal. For example, the outlet stream of a Heater block is generally a better choice for a tear stream than the outlet stream from a Reactor block.
- Disconnect the recycle stream to get a good initial estimate and to examine the sensitivity.
- Try to simplify the problem. It may be possible to do one or more of the following to reduce the complexity of the problem:
  - Add a Mixer block to reduce the number of tear streams
  - Replace a HeatX block with an MHeatX to reduce the number of tear streams
  - Define and use a Component Group to reduce the number of variables
  - Choose a Tear stream that has fewer components present
  - Choose a Tear stream from a block that sets an outlet temperature
- Reinitialize the simulation. Try to converge the simulation using a Wegstein acceleration parameter equal to 0 (set the upper bound and lower bound to 0). This is equivalent to direct substitution. Look for a continuing buildup of one or more components as the iterations proceed.
- Try using a different convergence method such as Broyden or Newton rather than the default Wegstein method.
- Confirm that the sequence for the simulation (either Aspen Plus defined or user defined) is reasonable. See Specifying the Calculation Sequence.

Using Initial Guesses

For many simulations with recycle streams, initial guesses for the tear streams will help convergence. This is especially true for recycle systems with closed loops or recirculating solvent loops. You can often provide a reasonable initial guess from your knowledge of the process or through a simple mass-balance calculation.

The sequence is displayed in the left pane of the Control Panel. If the left pane of the Control Panel is empty, select Step from the Run menu.

Enter initial compositions and flow rates for the tear streams on Streams Specification sheets, and run the simulation. Or select your own tear streams using the Tear sheet, and provide initial estimates for them.

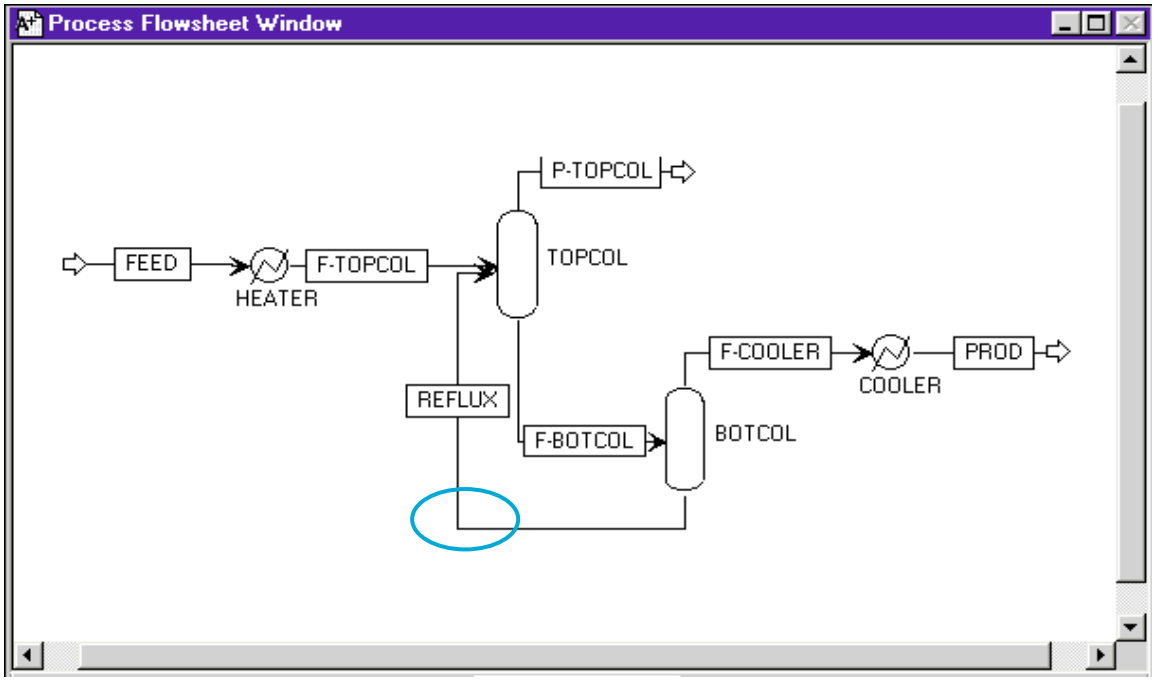
Specifying Convergence Order

You can specify the calculation order of convergence blocks you define if you use more than one user-defined convergence block.

Specifying the Calculation Sequence

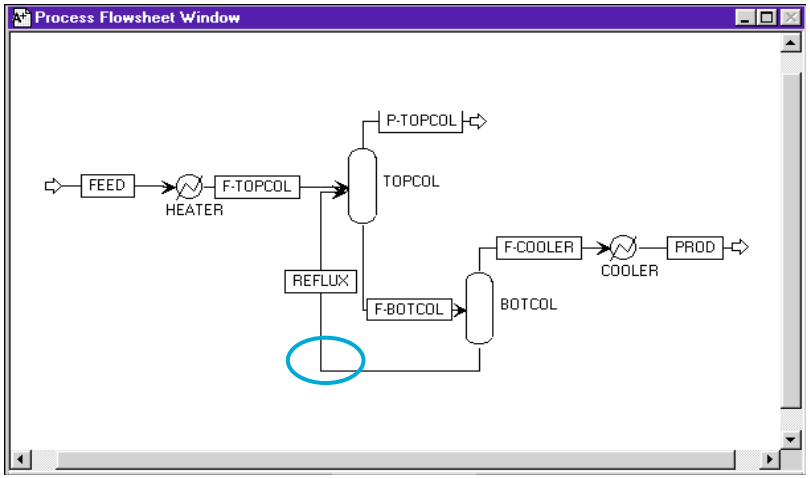
You can define the calculation order for all or part of the flowsheet. You supply an ID for each sequence.

ejemplo



The mass flow of stream REFLUX, the inter-reflux stream from BOTCOL to TOPCOL, is manipulated to meet a purity specification of component THF in stream PROD. PROD is a product stream from BOTCOL/COOLER, in design specification THF. PSPEC is the convergence block defined to converge THF.

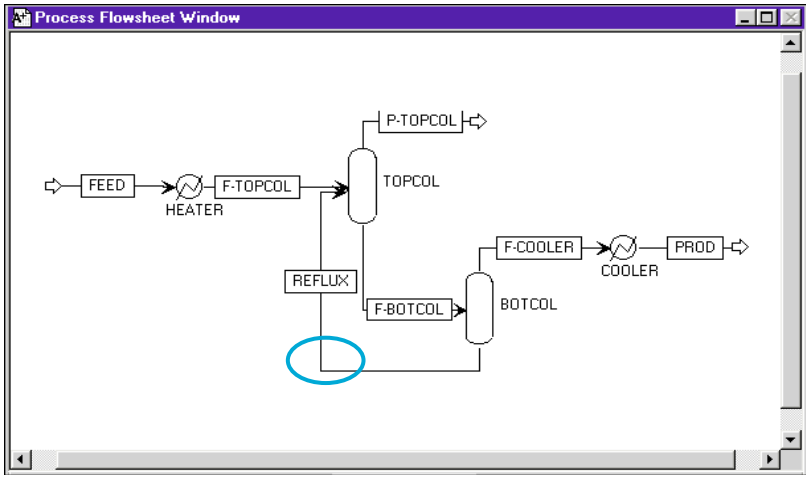
When distillation columns appear in a recycle loop, it is often necessary to give initial estimates for the tear stream. Aspen Plus makes this easy. Simply supply data for a column feed or other stream in the loop on Streams forms, just as you would for a feed stream, and Aspen Plus will preferentially select the stream as a tear stream (your stream may not be selected if another stream is a better choice by the tearing criteria).



From initial estimates for the tear stream, the Aspen Plus sequencing algorithm determines the following computation sequence:

HEATER
$OLVER01 TOPCOL
| PSPEC BOTCOL COOLER
| (RETURN PSPEC)
(RETURN $OLVER01)

$OLVER01 is defined to converge stream REFLUX, the inter-connecting stream, with initial data provided. However, with this sequence the PSPEC and $OLVER01 convergence blocks fail to converge, because the design specification is nested inside the column recycle loop. The design specification THF does not converge, because the purity specification is determined primarily by the inter-reflux between the two columns (not the top product rate of the BOTCOL alone).



The inter-reflux between the columns should be converged before evaluation of the design specification. The design specification should be nested outside the column recycle loop. You can alter the nesting order of the convergence loops by either:

- Specifying Design Spec Nesting as Outside on the Convergence ConvOptions Defaults Sequencing sheet, or
- Specifying PSPEC on the Convergence ConvOrder Specifications sheet.

Either specification would cause the sequencing algorithm to determine the following computation sequence, which converges:

```
HEATER
PSPEC
|   $OLVER01 TOPCOL BOTCOL
|   (RETURN $OLVER01
|   COOLER
(RETURN PSPEC)
```

Fin tema 3