方程式解法ソフトの利用

今回から6回の予定で化学工業における数値計算について事例を中心に紹介する。一口に化学工業における数値計算といっても，その内容はたいへん広範であり[1]，とてもその全体をカバーすることはできない。そこで本連載ではこれらの中から，いわゆる方程式解法ソフトで取り扱いうる問題に焦点を絞って解説してみることにする。

方程式解法ソフト

「方程式解法ソフト」という名称は必ずしも定着したものではないが，方程式をそのまま入力して(方程式を解く手続きを与えるのでなく)，直接その数値解を得ることのできるソフトウェアを総称している。このようなソフトウェアは大型計算機では以前から開発され利用されてきたが，最近ではパーソナルコンピュータ用のものも市販されている。国内で入手可能なものとしてたとえば，代数方程式や初等関数を含む超越方程式を扱うものとしては，TK! Solver[注1)]，GINO[注2)]，常微分方程式を扱うものとしてACSL-PC[注3)]があり，EQUATRAN-M[注4)]はその両方の取り扱いが可能となっている。本連載ではこれらのソフトウェアによる計算例とともに，FORTRANによる実用的なプログラミング例も併せて紹介したいと考えている。

今回は非線形連立方程式解法の代表例としてフラッシュ計算を取り上げ，Regula-falsi法を用いたプログラミング例を示したのち，同じ問題を方程式解法ソフトで解いた例を示す。

フラッシュ計算 part 1

気液平衡状態にある多成分系流れについて，液相と蒸気相の流量と組成を決定するフラッシュ計算は，化学工学では基本的な計算の一つである。化学プロセスの中で気液の分離のためにしばしば登場するフラッシュ分離器まわりの物質収支計算は，フラッシュ計算そのものである。**図1**に示すフラッシュ分離器において，気液混相の全フィード量をF〔kgmol/h〕，各成分のモル分率を zi (i＝1, 2…… N：Nは成分数) とし，液および蒸気の流量をLおよびV〔kgmol/h〕，それぞれのモル分率を x_i および y_i とすると，物質収支から次式が成り立つ。

$$L + V = F \qquad \cdots\cdots (1)$$

$$Lx_i + Vy_i = Fz_i \quad (i = 1, 2 \cdots N) \qquad \cdots\cdots (2)$$

また気液平衡関係から

$$y_i = K_i x_i \quad (i = 1, 2 \cdots N) \qquad \cdots\cdots (3)$$

ここに，K_i は平衡比である。また組成に関する条件から

$$\sum_{i=1}^{N} x_i - 1 = 0 \qquad \cdots\cdots (4)$$

注1) TK! Solver：Software Arts社(米)が開発，日本ユニバック情報システム(株)より日本語版が発売されている。
注2) GINO：住商コンピュータサービス(株)より発売。
注3) ACSL-PC：Mitchell & Gauthier Associate社(米)が開発したACSLをサイバネットシステム(株)がパソコンに移植して発売している。
注4) EQUATRAN-M(イコートラン　エム)：三井東圧化学(株)が開発，販売。

図1　フラッシュ分離器

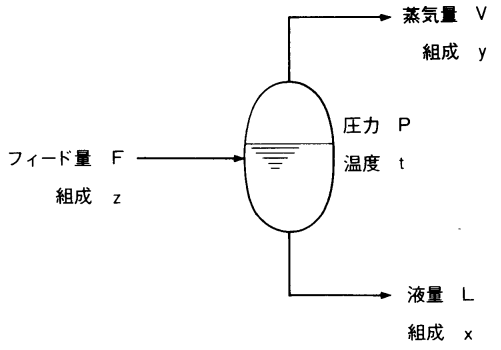

表1　ベンゼン－トルエン－キシレン系のフラッシュ計算

Antoine 式の係数[3]

	A	B	C
ベンゼン	6.90565	1211.03	220.790
トルエン	6.95464	1344.80	219.482
キシレン	6.99052	1453.43	215.307

計算条件

フィード量	100 kgmol/h
フィード組成	(0.4, 0.4, 0.2)
圧　力	1140 mmHg
温　度	118 ℃

または

$$\sum_{i=1}^{N} y_i - 1 = 0 \qquad \cdots\cdots(4')$$

のいずれかが必要である。

平衡比 K_i は一般に系の温度 t〔°C〕，圧力 P〔mmHg〕，液組成 **x** および蒸気組成 **y** の関数となる。

$$K_i = f_i(t, P, \mathbf{x}, \mathbf{y}) \quad (i = 1, 2 \cdots N) \qquad \cdots\cdots(5)$$

以上の連立方程式を，F, zi, t, P を与えて解けばよい。

具体例として，ベンゼン－トルエン－キシレンの3成分系の計算をする。この系では通常平衡比は温度と圧力のみの関数と考えてよく次式で与えられる。

$$K_i = p_i/P \qquad (i = 1, 2 \cdots N) \qquad \cdots\cdots(6)$$

P_i は各成分の飽和蒸気圧であり，次の Antoine の式で与えられるものとする。

$$\log_{10}(P_i) = A_i - B_i/(t + C_i) \quad (i = 1, 2 \cdots N) \qquad \cdots\cdots(7)$$

A_i, B_i, C_i は各成分に固有の係数である。**表1**に計算条件とともにその値を示した。

以上の式には非線形な項が含まれるので，数値的な繰り返し収束計算によって解く必要がある。計算の手順としては，L（または V）の値を仮定すれば，(1) 式から V（または L）を求めた後 (2)，(3) 式から x_i および y_i が計算できる。L の値は (4)（または (4′)）式が成立するように，収束計算によって計算すればよいように思われる。しかし，式をよく見ると，L = F とすれば (4) 式は恒等的に成り立つことがわかる。同様に L = 0 とすれば (4′) 式が恒等的に成り立つ。すなわち (4) 式，または (4′) 式を使用すると物理的に意味のないニセの解に収束する恐れがあることがわかる。その対策としていくつかの考え方が提案されているが[2]，(4) 式の代わりに

$$\sum_{i=1}^{N} (x_i - y_i) = 0 \qquad \cdots\cdots(8)$$

を用いると，このニセの解を避けることができる。

Regula-falsi法によるプログラム

本例のような一変数の求解の手法は単純なようであるが，収束が確実でかつ高速（試行の回数が少なくてすむ）なものにするのは，さほど簡単ではない。一変数の探索は試行回数が多少増えても計算時間に大きな影響はなさそうに思われるが，プログラムの規模が大きくなると2重，3重にネスティングされて使われることが多いのでバカにしてはいけない。

Regula-falsi 法は代表的な手法の一つである。方程式 $f(x) = 0$ の根を求めるとき，根をはさむ2点を x_1, x_2 とするとき（すなわち $f(x_1) \cdot f(x_2) < 0$），新たな x の推定値 x^* を2点 $(x_1, f(x_1))$, $(x_2, f(x_2))$ を結ぶ直線と x 軸との交点として求める。

すなわち，

図 2　Regula falsi 法が収束しにくい関数

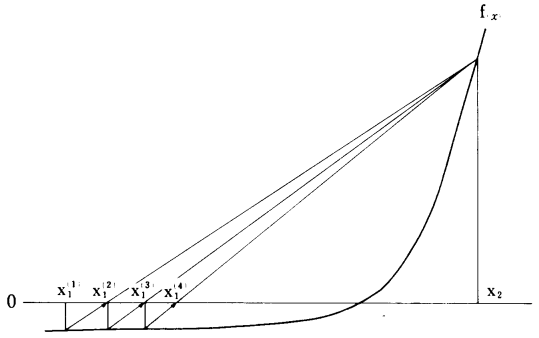

図 4　FORTRAN によるフラッシュ計算（メインプログラムと計算結果）

Microsoft FORTRAN 使用

```
C MAIN FOR FLASH TEST
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION X(3),Y(3),Z(3),AK(3),PO(3)
     *         ,A(3),B(3),C(3),Q(10)
      DATA A/ 6.90565, 6.95464, 6.99052/
     *    ,B/ 1211.03, 1344.80, 1453.43/
     *    ,C/ 220.790, 219.482, 215.307/
C  GET INPUT DATA
      WRITE(*,101)
 101  FORMAT(' T,P='¥)
      READ(*,*) T,P
      WRITE(*,102)
 102  FORMAT(' F='¥)
      READ(*,*) F
      WRITE(*,103)
 103  FORMAT(' Z='¥)
      READ(*,*) Z
C  CALC. K-VALUE
      DO 10 I=1,3
         PO(I) = 10.0D0**(A(I)-B(I)/(T+C(I)))
         AK(I) = PO(I)/P
 10   CONTINUE
C
C  SEARCH AL(L) THAT SATISFY S = 0 ( SUM(X(I)-Y(I)) = 0 )
C
      AL = 0.5*F
      IS = 0
      DO 60 LOOP=1,30
         V = F-AL
         S = 0.D0
         DO 30 I=1,3
            X(I) = F*Z(I)/(AK(I)*V+AL)
            Y(I) = X(I)*AK(I)
            S = S+X(I)-Y(I)
 30      CONTINUE
         WRITE(*,609) LOOP,IS,AL,S
C
         CALL REGL( AL, S, 0.3D0, 0.000001D0, Q, IS )
C
         IF( IS.EQ.0 ) GOTO 61
 60   CONTINUE
 61   WRITE(*,600) T,P
      DO 70 I=1,3
        WRITE(*,601) I,Z(I),X(I),Y(I),AK(I)
 70   CONTINUE
      WRITE(*,602) F,AL,V
      STOP
 600  FORMAT(//12X,'T = ',F12.3/12X,'P = ',F12.3
     *       //12X,'FEED',8X,'LIQUID',6X,'VAPOUR',9X,'K')
 601  FORMAT(1X,I5,4F12.5)
 602  FORMAT(1X,'TOTAL',3F12.3)
 609  FORMAT(1X,'LOOP=',I3,'  IS=',I2,'  L=',F12.5,'  S=',F12.5)
      END
```

```
T,P=118 1140
F=100
Z=0.5  0.3  0.2
LOOP=  1  IS= 0  L=     50.00000  S=      -.06342
LOOP=  2  IS= 1  L=     65.00000  S=      -.11791
LOOP=  3  IS= 1  L=     35.00000  S=      -.00707
LOOP=  4  IS= 1  L=     24.50000  S=       .03613
LOOP=  5  IS= 3  L=     33.28140  S=      -.00029
LOOP=  6  IS= 3  L=     33.21234  S=      -.00001
LOOP=  7  IS= 3  L=     33.20938  S=       .00000


           T =      118.000
           P =     1140.000

           FEED        LIQUID       VAPOUR          K
    1      .50000       .31490       .59203      1.88006
    2      .30000       .34143       .27940       .81833
    3      .20000       .34367       .12856       .37409
TOTAL    100.000       33.209       66.791
```

図 3　Regula falsi 法の FORTRAN サブルーチン “REGL”

```
      SUBROUTINE REGL(XC,YC,DX,CHECK,Q,IS)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION Q(10)
C
C  CONVERGENCE METHOD BY MODIFIED REGULA-FALSI
C   BY H.NAKAMOTO & G.OGUCHI(1986)
C   SEARCH  X FOR Y(X)=0
C       XC    ..... CURRENT VALUE OF X [INPUT(INITIAL)/OUTPUT]
C       YC    ..... CURRENT VALUE OF Y                 [INPUT]
C       DX    ..... MOVE FOR X BEFORE CATCH            [INPUT]
C                    MOVES +-ABS(X*DX) WHEN DX>0
C                          +-ABS(DX)   WHEN DX<0
C       CHECK .... CONVERGENCE ERROR ALLOWANCE         [INPUT]
C                  CHECKED BY ABS(Y)<=CHECK
C       Q     ..... WORK AREA FOR LOGIC                [OUTPUT]
C                  KEEP 80 BYTES FOR EACH CALL
C       IS    ..... CONTROL              [INPUT(INITIAL)/OUTPUT]
C                  SET 0 ON INITIAL CALL ONLY (DON'T FORGET!!!)
C                  RETURNS  0  WHEN CONVERGED
C                           1  BEFORE CATCH (Y<YO)
C                           2  BEFORE CATCH (Y>YO)
C                           3  AFTER CATCH
C
      X = XC
      Y = YC
      IF( IS.EQ.0 ) THEN
         IF( Y.EQ.0.D0 ) RETURN
         Q(2) = X
         Q(6) = Y
         Q(9) = DABS(DX)
         Q(10) = 0.0D0
      ELSE
         IF( Q(10).EQ.0.0D0 ) THEN
            IF( (Q(6)-Y)*(Q(2)-X).LT.0.0D0 ) THEN
               Q(9) = -Q(9)
               X = Q(2)
               Y = Q(6)
            ENDIF
            Q(10) = 1.0D0
         ENDIF
      ENDIF
      IF( IS.EQ.3 ) THEN
         IF( Q(6).EQ.Y ) THEN
            GRD = 10.0D0
         ELSE
            GRD = DABS( (Q(7)-Y)/(Q(3)-X)*(Q(2)-X)/(Q(6)-Y) )
         ENDIF
      ELSE
         GRD = 1.0D0
      ENDIF
      IF( Y.LT.0.D0 ) THEN
         Q(6) = Y
         Q(4) = Q(2)
         Q(2) = X
         IF( IS.EQ.2 ) IS = 3
         IF( IS.EQ.0 ) IS = 1
         PM = 1.0D0
      ELSE
         Q(7) = Y
         Q(5) = Q(3)
         Q(3) = X
         IF( IS.EQ.1 ) IS = 3
         IF( IS.EQ.0 ) IS = 2
         PM = -1.0D0
      ENDIF
      IF( DABS(Y).LE.CHECK ) IS = 0
      IF( IS.EQ.0 ) RETURN
      IF( IS.NE.3 ) THEN
         IF( DX.GT.0.0D0 ) THEN
            Q(1) = X+PM*Q(9)*DABS(X)
         ELSE
            Q(1) = X+PM*Q(9)
         ENDIF
      ELSE
         IF( GRD.GT.2.0D0 .OR. GRD.LT.0.5D0 ) THEN
            Q(1) = (Q(2)+Q(3))*0.5D0
         ELSE
            IF( Y*Q(8).LE.0.0D0 ) THEN
               Q(1) = Q(2)-Q(6)*(Q(3)-Q(2))/(Q(7)-Q(6))
            ELSE
               IF( Y.LE.0.D0 ) THEN
                  Q(1) = Q(2)-Q(6)*(Q(4)-Q(2))/(Q(8)-Q(6))
               ELSE
                  Q(1) = Q(3)-Q(7)*(Q(5)-Q(3))/(Q(8)-Q(7))
               ENDIF
               IF( (Q(1)-Q(3))*(Q(1)-Q(2)).GE.0.0D0 ) THEN
                  Q(1) = Q(2)-Q(6)*(Q(3)-Q(2))/(Q(7)-Q(6))
               ENDIF
            ENDIF
         ENDIF
      ENDIF
      Q(8) = Y
      XC = Q(1)
      RETURN
      END
```

図5　TK! Solver の V シートと R シート

```
(1r) R(式): L + V = F                                                    199/!

==================== V(変数) シート ============================================
St I(入力値) N(変数) O(出力値) U(単位)    C(注釈)
-- --------- ------- --------- -------   -------
             L
             V
             F




==================== R(式) シート ==============================================
S R(式)
- -----
* L + V = F
```

図6　全式のリスト (TK! Solver)

```
S R(式)
- -----
  "フラッシュ計算
* L + V = F
* L*x1 + V*y1 = F*z1
* L*x2 + V*y2 = F*z2
* L*x3 + V*y3 = F*z3
* y1 = K1*x1
* y2 = K2*x2
* y3 = K3*x3
* (x1+x2+x3) - (y1+y2+y3) = 0
* LOG(p1) = A1 - B1/(t+C1)
* LOG(p2) = A2 - B2/(t+C2)
* LOG(p3) = A3 - B3/(t+C3)
* K1 = p1/P
* K2 = p2/P
* K3 = p3/P
```

$$x^* = x_1 - f(x_1) \cdot \frac{x_2 - x_1}{f(x_2) - f(x_1)} \quad \cdots\cdots\cdots\cdots (9)$$

この x^* によって，x_1 あるいは x_2 のいずれかを根をはさむように置き替えて試行を繰り返す。この手法の特長は根をはさんだ後の収束が確実なことと，f(x) が直線に近くなると極めて高速に収束することである。しかし，**図2** に示すような根の近くで曲率の大きな関数の場合に，極端に収束が遅くなる欠点があって，このままでは汎用的な手法としては不十分である。このためこの欠点を補うような手法と組み合わせて使うことが多い。たとえば修正ハミング法[4]は収束が高速なことで知られている。

図3 に示すサブルーチン REGL は，Regula-falsi 法に2分法と外挿法とを組み合わせたもので[4]，これでも十分実用にたえる。使用方法はリスト内の注釈と，**図4** に示したメインプログラムおよび計算例(前出のフラッシュ計算のもの) を参考にして欲しい。このルーチンは Q と IS とをそれぞれのループごとに用意すればネスティングして用いることができる。なお，根をはさむまでは，x の初期値から最初に x を変化させたときの関数の傾きによって根を探す方向が決められている。

TK! Solver 方程式解法ソフト(1)

TK! Solver は代数方程式などからなる連立方程式を解くためのパーソナルコンピュータ用簡易プログラム (TK は問題解法の Tool Kit の略) で，技術計算などでよく扱われる数学モデルを容易に解くことを目指している。スプレッドシート (表言語) の元祖として有名な VisiCalc の開発者の手によるものだけあって，機能と使いやすさについては BYTE 誌のレビューでも評価が高かった[5]。米国では IBM-PC でのユーザが多いが，Macintosh でも応用ソフトとして登録されている[6]。日本語版は UP 10 E 用が販売されているが，近く PC-9801 用も発売されるとのことである。

TK! Solver による解法

さて，本例題を TK! Solver で解いてみよう。TK! Solver を起動した状態の画面には，変数の一覧を表示する V (変数) シートと，解くべき方程式を入力する R (式) シートが上下に分かれて配置されている (**図5**)。式 (1) から式 (8) までを順次キーインしていくが，一つの式を入力し終ると，その式に含まれる新しい変数が自動的に V シートに登録されていく。図5は式 (1) の入力を終ったときの様子を示しており，**図6** は入力した全式のリストである。

次に，変数の中で数値を与えるべきもの (z1, A1, t, P など) には V シート上で I (入力値) の欄に値をキーインする。また，繰り返し計算を要する変数には，左端の St 欄を G (Guess の意) とし，その初期値を I (入力値) 欄に与える。TK! Solver では線形の連立方程式も繰り返し計算で解くので，L の他に x1, x2, x3 に G の指定を行う。単位や注釈を加えたものの一部が**図7** である。なお，シート間は [:] (Switch コマンド) で往来でき，シート内はカーソルを自由に移動できる。これだけ準備をして [!] (Solve コマンド) を実行すれば，解析と計算が実行される。計算結果は V シートの O (出力値) 欄に表示される (**図8**)。

BASIC や FORTRAN では“=”が左辺の変数に値を代入する記号として用いられているのに対し，ここでの“=”は数学的な等号を意味している。したがって，左辺に式が書かれていてもよいわけで，これが方程式解法ソフトの特徴と言えよう。

TK! Solver では [:] や [!] の他に方程式のコピー・修正や変数の削除・移動，あるいはファイルの管理などのコマ

図7　計算の準備を終了したVシート（一部分）

St	I(入力値)	N(変数)	O(出力値)	U(単位)	C(注釈)
G	50	L		kgmol/h	液流量
		V		kgmol/h	蒸気流量
	100	F		kgmol/h	フィード量
		SL			
G	.3	x1			
		y1			
	.5	z1			
G	.3	x2			
		y2			
	.3	z2			
G	.4	x3			
		y3			
	.2	z3			
		K1			

図8　計算完了時のVシート

St	I(入力値)	N(変数)	O(出力値)	U(単位)	C(注釈)
		L	33.209309	kgmol/h	液流量
		V	66.790691	kgmol/h	蒸気流量
	100	F		kgmol/h	フィード量
		SL			
		x1	.31490193		
		y1	.59203347		
	.5	z1			
		x2	.34142771		
		y2	.27940154		
	.3	z2			
		x3	.34367036		
		y3	.12856499		
	.2	z3			
		K1	1.8800566		
		K2	.81833294		
		K3	.37409393		
		p1	2143.2645	mmHg	ベンゼン飽和蒸気圧
	6.90565	A1			
	1211.03	B1			
	118	t		°C	温度
	220.79	C1			
		p2	932.89955	mmHg	トルエン飽和蒸気圧
	6.95464	A2			
	1344.8	B2			
	219.482	C2			
		p3	426.46708	mmHg	キシレン飽和蒸気圧
	6.99052	A3			
	1453.43	B3			
	215.307	C3			
	1140	P		mmHg	圧力

図9　ベンゼン－トルエン－キシレン系のフラッシュ計算（EQUATRAN-Mのソースリストと計算結果）

```
 1:  /* フラッシュ計算 (ベンゼン - トルエン - キシレン) */
 2:
 3:  VAR  F          "フィード量      [kgmol/h]" ..
 4:       ,t         "温度            [℃]      " ..
 5:       ,P         "圧力            [mmHg]   " ..
 6:       ,L         "液量            [kgmol/h]" ..
 7:       ,V         "蒸気量          [kgmol/h]" ..
 8:       ,z(3)      "フィード組成    [ - ]    " ..
 9:       ,x(3)      "液組成          [ - ]    " ..
10:       ,y(3)      "蒸気組成        [ - ]    " ..
11:       ,K(3)      "平衡比          [ - ]    " ..
12:       ,p(3)      "飽和蒸気圧      [mmHg]   " ..
13:       ,A(3)      "Antoine式の係数  A       " ..
14:       ,B(3)      "Antoine式の係数  B       " ..
15:       ,C(3)      "Antoine式の係数  C       "
16:
17:         L + V = F
18:         L*x + V*y = F*z
19:         y = K*x
20:         K = p/P
21:         LOG10(p) = A-B/(t+C)
22:            A = (6.90565, 6.95464, 6.99052)
23:            B = (1211.03, 1344.80, 1453.43)
24:            C = (220.790, 219.482, 215.307)
25:  eq8:   SUM(x-y) = 0
26:
27:  INPUT   F,z,t,P
28:  OUTPUT  L,V,x,y,K
29:
30:  RESET   L#50[0,100] BY eq8
```

```
[   入力データ   ]
F                =  100              : フィード量      [kgmol/h]
z                =                   : フィード組成    [ - ]
  1) 0.5              2) 0.3              3) 0.2
t                =  118              : 温度            [℃]
P                =  1140             : 圧力            [mmHg]
<R  0> L          = 50                -0.06342      =? 0      : -0.06342
<R  1> L          = 50.05             -0.0636       =? 0      : -0.0636
<R  2> L          = 32.609324         0.00239       =? 0      : 0.00239
<R  3> L          = 33.241555         -0.00013      =? 0      : -0.00013
<R  4> L          = 33.209391         -3.24520E-007 =? 0      : -3.24520E-007
<R  5> L          = 33.209309         4.37379E-011  =? 0      : 4.37379E-011

[   計算結果   ]
L                =  33.209309        : 液量            [kgmol/h]
V                =  66.790691        : 蒸気量          [kgmol/h]
x                =                   : 液組成          [ - ]
  1) 0.3149019        2) 0.3414277        3) 0.3436704
y                =                   : 蒸気組成        [ - ]
  1) 0.5920335        2) 0.2794015        3) 0.128565
K                =                   : 平衡比          [ - ]
  1) 1.880057         2) 0.8183329        3) 0.3740939
```

ンド体系がよくできており，初心者でもすぐに慣れることができる。変数名に大文字と小文字が使い分けられるのはひじょうに便利である。一つの方程式の長さは256文字まで許されており，方程式の数も100本以上扱かえる（式の長さとメモリの大きさによる）ので，かなり複雑な数学モデルでも解くことができる。

収束計算手法

ところで技術計算では入力変数と出力変数を逆にしたいことがよくある。本例で言えば，Lを既知としてtを求めたい場合がそうであるが，Lに入力値を与えtに推定値を与える（St欄をGとする）だけで簡単に対応することができる（Backsolvingといっている）。また，ケーススタディでパラメータ変数の値をリストとして登録しておき，全ケースを計算後に数表ないし，キャラクタグラフとして出力することができる（リスト解法）。計算時に用いる単位と出力として表示する単位とを使い分けることができる（単位変換）のは便利である。

逐次解法（収束計算を含む解法）に用いられている収束計算手法についてマニュアルに記載はないが，ニュートンラフソン法に基づくものと思われる。この手法はなかなか強力で，もちろん初期値にもよるが，非線形性の強い問題でも収束性はよい。

有効な使用法

TK! Solverはその名のとおり道具であるから，これを

うまく使いこなすにはそれなりの配慮が必要である。まず，逐次解法においてどの変数を繰り返し変数に指定するかの選択が重要で，必要なだけの指定がないと計算はストップする。特に非線形なモデルでは適切な変数を指定することによって，安定かつ迅速に収束させることができる。この選択は，問題の物理的な意味を考えて行うと良いことが多い点は，プログラミングをする場合と同様である。

次に，方程式と未知変数の過不足について留意しなければならない。正確に記述された数学モデルであれば両者の数は一致するはずであるが，与条件を落したり，変数名を書き誤ったりして，時には過不足を生ずることがある。TK! Solver では部分解法といって，与えられた条件から解ける部分解を求めてくる。あるいは冗長な方程式があっても，解いた結果を代入して等号の成立を確認する。これらは一見便利な機能であるが，十分注意して計算結果を検査しないと数学モデルの誤りを見過すことになりかねない。一方，次項で説明する EQUATRAN-M の方は，1つでも矛盾が見つかると計算を始めてくれないと気難しいが安心でもある。

EQUATRAN-M 方程式解法ソフト(2)

EQUATRAN-M（イコートラン　エム）はもともと化学プロセスの計算のために大型計算機用に開発されたものであるが，機能的には全く汎用となっている。

EQUATRAN-M では方程式の他に，変数の定義や入出力する変数の指定などをすべて一つのテキスト（ソーステキストと呼ぶ）として作成するようになっている。EQUATRAN-M は方程式を記述するための非手続型の言語であって，この点で前述の TK! Solver とはかなり使い勝手が異なっている。**図9** にフラッシュ計算の例題のためのソーステキストのリストと計算結果を示した。

リスト中の／＊　＊／で囲まれた部分は注釈で，テキストの任意の位置に書くことができる。3～15行目は変数を定義している部分で，VARIABLE 文（または VAR 文）と呼ばれる文である。行末の‥は行の継続を示す。EQUATRAN-M では2次元までの配列変数（ベクトルとマトリクスに相当）を使うことができるが，配列変数はその次元と要素数を VARIABLE 文で指定する。VARIABLE 文中の“　”で囲まれた部分は変数の「説明項」といい，変数の意味や単位などを書くことができる。説明項は入出力の表示などに利用される。本例では物質の成分数の要素を持つ変数（x, y, z, K など）を1次元の配列（ベクトル）として扱った。

17～25行目が方程式を記述した部分である。式を書く順序は任意でよく，式の変形も不要である。式中に添字をつけずに書かれた配列変数名は配列全体を表わしている。このため配列のすべての要素についての式が一行で書けるので，記述がたいへんコンパクトになる。19行目の K＊x のようなベクトル同士の演算は演算の種類によらず，2つのベクトルの対応する要素間の演算として定義されている。マトリクス間の演算も同様である。25行目の SUM( ) は $\sum_{i=1}^{N}$ に相当する関数である。同じ行の eq8 は式のラベルといい，個々の式を識別するためのシンボル名がつけられる。

27行目の INPUT 文はケーススタディのために計算の実行時に値を読み込む変数を指定している。

30行目の RESET 文は収束計算の方法を指定する文で，

```
RESET  L # 50 [0, 100] BY eq8
```

では，L を収束計算の独立変数とし，式 eq8 の両辺の値を一致させるように収束計算を行うことを指定している。＃50 [0, 100] の部分は，L の初期値と変域を与える。RESET 文による指定は省略も可能で，省略した場合自動的に収束計算の方法が決定される。

図9のソーステキストを EQUATRAN-M 内蔵のスクリーンエディタを使って作成した後，RUN コマンドを実行すると，INPUT 文のための入力プロンプトが表示される。これに値を与えると図9の下の結果が得られる。EQUATRAN-M では収束計算の手法として，ニュートンラフソン法の改良法が使われている。図9の出力中に〈R 0〉などとある行は，収束の様子を示すオプショナルな出力で，独立変数 L の値の他に式 eq8 の両辺とその差が示されている。

フラッシュ計算 part 2

前述のフラッシュ計算の例では平衡比 K は，温度と圧

力のみの関数であったが，物質系によっては液組成の影響が無視できない。この場合平衡比は

$$K_i = \gamma_i p_i / P \qquad (i = 1, 2 \cdots N) \qquad \cdots\cdots\cdots (10)$$

で表わされる。γ_i は活量係数と呼ばれ，一般に液組成と温度の関数となる。活量係数を与える式は多く提案されているが，最もポピュラーなものの一つにWilson式がある。多成分系の活量係数はWilson式で次のように表わされる。

$$\ln \gamma_i = -\ln\left(\sum_{j=1}^{N} x_j \cdot \Lambda_{ij}\right) + 1 - \sum_{k=1}^{N} \frac{x_k \Lambda_{ki}}{\sum_{j=1}^{N} x_j \Lambda_{kj}}$$

$$(i = 1, 2 \cdots N) \qquad \cdots\cdots\cdots\cdots (11)$$

$$\Lambda_{ij} = \frac{v_j}{v_i} \exp\left\{-\frac{\lambda_{ij} - \lambda_{ii}}{R(t + 273.15)}\right\}$$

$$(i = 1, 2 \cdots N;\ j = 1, 2 \cdots N) \qquad \cdots\cdots\cdots (12)$$

ここでの λ_{ij} はWilson式のパラメータと呼ばれ実測値より決定される。v_i は各成分のモル容積，Rは気体定数である。

具体例として**表2**にアセトン－メタノール－水の3成分系について，Wilson式のパラメータとAntoine式の係数を示した。

活量係数を考慮した場合のフラッシュ計算は，平衡比が液組成の複雑な関数となるので，液量Lのほかに液組成 x_i または活量係数 γ_i を独立変数とする多変数の収束計算が必要となる。このためプログラミングはかなりやっかいであるが，方程式解法ソフトを使えば，式を追加するだけで簡単に対応できる。

図10にEQUATRAN-Mを使った場合のソーステキストと計算結果を示した。ソーステキスト31行目（(11)式）のx＊Lamはベクトルとマトリクスの演算になっているが，これはベクトルxがマトリクスLamの各行ベクトルに対して演算されることを示している。したがってx＊Lamは i 行 j 列が $x_j \Lambda_{ij}$ からなるマトリクスを表わしていることになる。マトリクスに対して∑の関数SUMを適用すると各行の要素を加えてできるベクトルが得られる。また，マトリクスの名前に‾（オーバーライン）を付けると（たとえばLam‾），転置マトリクスを表わすことができる。34行目のマトリクスvtは v_j/v_i を計算するための補助的な変数で，vを列ベクトルにもつマトリクスとなっ

表2　アセトン－メタノール－水系のフラッシュ計算

Antoine式の係数[7]

	A	B	C
アセトン	7.11714	1210.595	229.664
メタノール	8.08097	1582.271	239.726
水	8.07131	1730.630	233.426

Wilson式のパラメータ（$\lambda_{ij} - \lambda_{ii}$）[7]

i \ j	1（アセトン）	2（メタノール）	3（水）
1（アセトン）	0	− 333.977	6696.186
2（メタノール）	715.549	0	916.036
3（水）	1100.784	326.213	0

モル容積

	v
アセトン	74.05
メタノール	40.73
水	18.07

計算条件

フィード量	100 kgmol/h
フィード組成	(0.4, 0.4, 0.2)
圧力	760 mmHg
温度	62 ℃

図10　アセトン－メタノール－水系のフラッシュ計算
(EQUATRAN-M のソースリストと計算結果)

```
 1:  /* フラッシュ計算 (アセトン - メタノール - 水) */
 2:
 3:  VAR  F          "フィード量      [kgmol/h]" ..
 4:       ,t         "温度            [℃]      " ..
 5:       ,P         "圧力            [mmHg]   " ..
 6:       ,L         "液量            [kgmol/h]" ..
 7:       ,V         "蒸気量          [kgmol/h]" ..
 8:       ,z(3)      "フィード組成    [ - ]    " ..
 9:       ,x(3)      "液組成          [ - ]    " ..
10:       ,y(3)      "蒸気組成        [ - ]    " ..
11:       ,K(3)      "平衡比          [ - ]    " ..
12:       ,p(3)      "飽和蒸気圧      [mmHg]   " ..
13:       ,g(3)      "活量係数(γ)    [ - ]    " ..
14:       ,Lam(3,3)  "Wilson式ﾊﾟﾗﾒｰﾀ(Λ)      " ..
15:       ,lam(3,3)  "Wilson式ﾊﾟﾗﾒｰﾀ(λ-λ)    " ..
16:       ,v(3)      "モル容積        [cc/mol] " ..
17:       ,R = 1.986 "気体定数      [cal/molK]" ..
18:       ,A(3)      "Antoine式の係数  A       " ..
19:       ,B(3)      "Antoine式の係数  B       " ..
20:       ,C(3)      "Antoine式の係数  C       " ..
21:       ,vt(3,3)
22:
23:         L + V = F
24:         L*x + V*y = F*z
25:         y = K*x
26:         K = g*p/P
27:         LOG10(p) = A-B/(t+C)
28:            A = ( 7.11714, 8.08097, 8.07131)
29:            B = (1210.595,1582.271,1730.630)
30:            C = ( 229.664, 239.726, 233.426)
31:  eq11: LOGE(g) = -LOGE(SUM(x*Lam))+1 ..
32:                          -SUM((x*Lam)/SUM(x*Lam))
33:         Lam = v/vt*EXP(-lam/R/(t+273.15))
34:            vt=v
35:            lam = (       0, -333.977, 6696.186) ..
36:                  ( 715.549,       0,  916.036) ..
37:                  (1100.784,  326.213,       0)
38:            v   = (   74.05,    40.73,    18.07)
39:  eq8:  SUM(x-y) = 0
40:
41:  INPUT   F,z,t,P
42:  OUTPUT  L,V,x,y,K,g
43:  RESET   L#50[0,100] BY eq8   ..
44:          ,g#1  [0,2]  BY eq11
```

```
[   入力データ   ]
F                =  100              : フィード量      [kgmol/h]
z                =                   : フィード組成    [ - ]
  1) 0.4              2) 0.4                3) 0.2
t                =  62               : 温度            [℃]
P                =  760              : 圧力            [mmHg]

[   計算結果   ]
L                =  47.152342        : 液量            [kgmol/h]
V                =  52.847658        : 蒸気量          [kgmol/h]
x                =                   : 液組成          [ - ]
  1) 0.2884978        2) 0.4134476          3) 0.2980545
y                =                   : 蒸気組成        [ - ]
  1) 0.4994857        2) 0.3880016          3) 0.1125126
K                =                   : 平衡比          [ - ]
  1) 1.731333         2) 0.9384541          3) 0.3774901
g                =                   : 活量係数(γ)    [ - ]
  1) 1.421358         2) 1.038299           3) 1.755869
```

ている。収束計算の指定は44行目にg(γ)についてのものが追加されているが，これもベクトルgの全要素についての指定になっている。

この他にEQUATRAN-Mでは，配列変数の部分配列を添字で表現することもできるなど，配列に関する記述力が豊富なのでFORTRANのDO文のような手続き型の文を使わずに，ほとんどの方程式を簡潔に書くことができる。

おわりに

このように，解きたい方程式群をキーインした後，数値を設定する変数に値を入力するだけで解の得られる方程式解法ソフトは，ユーザが問題に直面して，そのつどアルゴリズムを組み立てプログラムを作成する手間を節約してくれる。したがって，技術計算のように非定型な業務のツールとしては誠に便利で，プログラミングに要した時間を問題の解析などのより創造的な業務に振り向けることができる。あるいは，専用のプログラムを作るに先立って，基礎方程式を確認するのに使っている人もいる。大学では研究用のほかに，数値計算とプログラミングの講義・演習を修了した学生に対して，設計や実験結果の解析の授業に取り入れられたら，また面白いのではなかろうか。

(おぐち　ごろう，さどとも　ひでお　三井東圧化学㈱)

参考文献

1) 宮原昱中; 化学工学の数値計算，コンピュートロール， No. 12 (1985)，p 49－54
2) 宮原他;「EQUATRAN-M」－技術計算用連立方程式解法言語 (3)，ケミカルエンジニヤリング，第30巻第10号，(1985)　p 18－26
3) 化学工学協会; 化学工学便覧，改訂四版，1978，p 29
4) 宮原昱中，佐渡友秀夫; 改良 Regula falsi，化学工学，第37巻第9号，(1973)　p 953－956
5) A. R. Miller; Softwave Review "TK! Solver", BYTE, Dec. (1984)　p 263－272
6) 宮本勲; マッキントッシュの全容を探る－3，bit, 17 (1985)　p 83
7) J. Gmehling, U. Onken; "VAPOR-LIQUID EQUILIBRIUM DATA COLLECTION", Chemistry Data Series Vol. 1, Part 1, DECHEMA, 1977, p 565