PNC TN8410 93−083

限定資料

本資料は　年　月　日付けで登録区分、
変更する。 2001. 6. 6
[技術情報室]

CMPO混合溶媒による硝酸抽出のモデル化および計算コード

−TRU元素の湿式分離研究におけるシミュレーションコードの開発−

1993年4月

動力炉・核燃料開発事業団
東海事業所

本資料の全部または一部を複写・複製・転載する場合は、下記にお問い合わせください。

〒319-1184　茨城県那珂郡東海村大字村松4番地49
核燃料サイクル開発機構
技術展開部　技術協力課

Inquiries about copyright and reproduction should be addressed to:
Technical Cooperation Section,
Technology Management Division,
Japan Nuclear Cycle Development Institute
4-49 Muramatsu, Tokai-mura, Naka-gun, Ibaraki, 319-1184
Japan

© 核燃料サイクル開発機構 (Japan Nuclear Cycle Development Institute)
2001

Ⓣ

限定資料
PNC TN8410 93-083
1993年4月

登録区分
2001. 6. 6
変更表示

CMPO混合溶媒による硝酸抽出のモデル化および計算コード

―TRU元素の湿式分離研究におけるシミュレーションコードの開発―

駒　義和*　　星野忠也**
根本慎一*　　小沢正基*
富樫昭夫*

要　　旨

使用済み燃料の再処理で発生する高レベル廃液から超ウラン元素(TRU)を回収する研究が国内外で進められている。動燃においても1990年（平成2年度）から米国Argonne National Laboratory（ANL）で開発されたTRUEX法の応用研究を開始した。既に、高レベル放射性物質研究施設(CPF)においてホット試験を実施し、FBR照射燃料再処理試験で発生した高放射性廃液からAmやCmを効率良く回収できることを確認している。このプロセスは抽出溶媒の特性、対象とする溶液及びそれに含まれる対象元素が異なるため、既存のシミュレーションコードでは模擬することができない。そのため、TRUEX法で使用されるCMPO－TBP－n-dodecane混合溶媒による主要な化学種の分配をモデル化し、計算コード化する作業が必要である。

本報告書では、硝酸の抽出についてモデル化・コード化し、向流多段接触器を用いたホット試験の濃度プロファイルと比較した。

硝酸はCMPOとTBPのそれぞれにより抽出され、抽出剤分子間の相互作用はないものと仮定して、個々の抽出平衡の寄与を求めることで計算した。

我々のバッチ試験の結果との比較では、CMPO混合溶媒が通常使用される領域である、0.7から1.4MのTBPの濃度範囲で良く一致した。また、ミキサセトラを使用した向流多段ホット試験の濃度プロファイルとの比較では、0.1M以上の硝酸濃度の領域では良く一致した。

---

*　東海事業所 再処理技術開発部 プロセス・分析開発室
**　（株）ペスコ

目　　次

1. はじめに …… 1
2. 目的 …… 2
3. 分配比のモデル化 …… 3
3.1 TBP による硝酸抽出の寄与 …… 3
3.2 CMPO による硝酸抽出の寄与 …… 4
3.3 活動度係数 …… 5
3.4 硝酸の分配比 …… 5
4. 計算コードへの組み込み …… 8
4.1 向流多段抽出計算コードの選択 …… 8
4.2 分配比のコード化 …… 8
5. 計算の妥当性 …… 11
5.1 硝酸濃度およびTBP濃度依存性 …… 11
5.2 連続試験への適用 …… 11
6. おわりに …… 19
7. 参考文献 …… 20
8. 計算コードリスト …… 22
8.1 ヘッダファイルリスト …… 22
8.1.1 SEPHIS.H …… 22
8.2 関数コードリスト …… 25
8.2.1 C.C …… 25
8.2.2 C_CALC.C …… 34
8.2.3 C_CHEM.C …… 39
8.2.4 C_STAGE.C …… 46
8.2.5 C_INIT.C …… 56
8.2.6 C_DISP.C …… 64
8.2.7 C_PRN.C …… 70
8.2.8 ETC.C …… 79

1. はじめに

使用済み燃料の再処理で発生する高レベル廃液から超ウラン元素(TRU)を回収する研究が国内外で進められている。回収の対象となるのは、長寿命でありなおかつα線を放出する同位体が存在するAmやCmであり、これらは硝酸溶液中でIII価となる元素である。これらを溶媒抽出法により回収するために、二座配位型の抽出剤の合成、その応用方法の開発が米国、仏国で積極的に行われている[1-3]。

動燃においても1990年（平成2年度）から米国Argonne National Laboratory (ANL)で開発されたTRUEX法の応用研究を開始した。既に、高レベル放射性物質研究施設(CPF)を利用したホット試験も実施し、FBR照射燃料再処理試験で発生した高放射性廃液（ラフィネート）からAmやCmを効率良く回収できることを確認している[4-6]。この試験は、向流多段抽出器として小型のミキサセトラを使用して行ったが、抽出溶媒の特性、対象とする溶液及びそれに含まれる対象元素が異なるため、既存のシミュレーションコードでは模擬することができない。

TRUEX法は濃縮高レベル廃液等からのTRUの回収に適用するだけでなく、PUREX法と組み合わせ、従来の再処理プロセスにTRU回収機能を付加した複合プロセスを構築できる可能性がある[6]。基本的な性能が確認された後には、プロセス設計に対する要求が増大するものと考えられる。そのため、TRUEX法で使用されるCMPO－TBP－n-dodecane混合溶媒による主要な化学種の分配をモデル化し、計算コード化する作業が必要である。

ANLからは、既に硝酸抽出及びAmの抽出をモデル化する作業の報告がなされている[7]。また、パーソナルコンピュータ上で表計算アプリケーションソフトウエアを利用したシミュレーションも行われている[8]。

本報告書では、硝酸の抽出についてモデル化・コード化し、向流多段接触器を用いたホット試験の濃度プロファイルと比較した結果について述べる。

2. 目的

高レベル廃液からTRU元素を回収するプロセスを構築するためには、主な元素の挙動を予測するシミュレーションコードが必要である。CMPO－TBP混合溶媒を使用する溶媒抽出プロセスの模擬には、硝酸、Ⅲ価のアクチニドおよびランタニド、U、Puの抽出挙動のモデル化が要求される。

今回は、主要な化学種の中でも計算する上で最も基礎的な硝酸の抽出についてモデル化、さらにそれを基に計算コード化し、既存のコード(SEPHIS)へ組み込み、計算の信頼性を確認することを目的とする。

3. 分配比のモデル化

硝酸の抽出のモデル化には、Chaiko らが報告している方法[7]を用いる。

硝酸は、CMPO と TBP のそれぞれにより抽出される。ここで、抽出剤分子間の相互作用はなく、個々の抽出剤による抽出の和で表現できるものと仮定する。

3.1 TBPによる硝酸抽出の寄与

TBP の抽出反応として、以下の2つを考える。

$$H^+ + NO_3^- + \overline{TBP} \longleftrightarrow \overline{TBP \cdot HNO_3} \tag{1}$$

$$H^+ + NO_3^- + 2\,\overline{TBP} \longleftrightarrow \overline{(TBP)_2 \cdot HNO_3} \tag{2}$$

ここで、KT1 と KT2 は平衡定数であり、次のように表される。

$$KT1 = \frac{[\overline{TBP \cdot HNO_3}]}{[\overline{TBP}][H^+][NO_3^-]y_{12}^2} \tag{3}$$

$$KT2 = \frac{[\overline{(TBP)_2 \cdot HNO_3}]}{[\overline{TBP}]^2[H^+][NO_3^-]y_{12}^2} \tag{4}$$

TBPにより抽出される有機相中の全硝酸濃度は、

$$[\overline{HNO_3}]_{TBP} = KT1\ [\overline{TBP}][H^+][NO_3^-]y_{12}^2 + KT2\ [\overline{TBP}]^2[H^+][NO_3^-]y_{12}^2 \tag{5}$$

である。ここで、$[\overline{TBP}]$ はフリーのTBPの濃度である。分配比を求めるためには、(5)式を解かなければならない。この式の中で $[H^+]$ と $[NO_3^-]$ は与えられている変数であり、KT1 および KT2 は文献[7]から引用して用いるので既知であるものとする。また、活動度係数 y_{12} は水相の硝酸濃度の関数として別に計算できるものとする。以上から、フリーのTBP濃度を求める必要があることが分かる。フリーの TBP の濃度は TBP の全濃度から HNO_3 の抽出に寄与している成分を除いたものであり、

$$[\overline{TBP}] = [\overline{TBP}]_t - [\overline{TBP \cdot HNO_3}] - 2\,[\overline{(TBP)_2 \cdot HNO_3}] \tag{6}$$

と表される。$[\overline{TBP}]_t$ は全TBP濃度である。この式の $[\overline{TBP \cdot HNO_3}]$ と $[\overline{(TBP)_2 \cdot HNO_3}]$ の項は未知である

ので、(3) および (4) 式を代入して消去する。すると、

$$[\overline{TBP}] = [\overline{TBP}]_i - KT1\,[\overline{TBP}][H^+][NO_3^-]y_{12}^2 - 2\,KT2\,[\overline{TBP}]^2[H^+][NO_3^-]y_{12}^2 \tag{7}$$

となるが、これは、$[\overline{TBP}]$ についての二次式であり、他の変数も前述のとおり既知であるものであり解くことができる。$[\overline{TBP}]$ について整理すると、

$$2\,KT2\,[\overline{TBP}]^2[H^+][NO_3^-]y_{12}^2 + [\overline{TBP}]\,(1 + KT1\,[H^+][NO_3^-]y_{12}^2) - [\overline{TBP}] = 0 \tag{8}$$

となる。解として求めた $[\overline{TBP}]_f$ を (5) 式に代入すると、TBP により抽出された有機相中の硝酸濃度が求められる。

3.2　CMPO による硝酸抽出の寄与

CMPO による抽出反応は、

$$H^+ + NO_3^- + \overline{CMPO} \overset{KC1}{\longleftrightarrow} \overline{CMPO \cdot HNO_3} \tag{9}$$

$$2\,H^+ + 2NO_3^- + \overline{CMPO} \overset{KC2}{\longleftrightarrow} \overline{CMPO \cdot 2HNO_3} \tag{10}$$

を考える。抽出平衡定数は、

$$KC1 = \frac{[\overline{CMPO \cdot HNO_3}]}{[\overline{CMPO}][H^+][NO_3^-]y_{12}^2} \tag{11}$$

$$KC2 = \frac{[\overline{CMPO \cdot 2\,HNO_3}]}{[\overline{CMPO}][H^+]^2[NO_3^-]^2y_{12}^4} \tag{12}$$

である。CMPO により抽出される硝酸の有機相中の濃度は、

$$[\overline{HNO_3}]_{CMPO} = KC1[\overline{CMPO}][H^+][NO_3^-]y_{12}^2 + 2 \cdot KC2[\overline{CMPO}][H^+]^2[NO_3^-]^2y_{12}^4 \tag{13}$$

となる。フリーの CMPO 濃度は、

$$[\overline{CMPO}] = [\overline{CMPO}]_t - [\overline{CMPO \cdot HNO_3}] - [\overline{CMPO \cdot 2\,HNO_3}] \tag{14}$$

であるから、(11) および (12) 式を代入して消去する。

$$[\overline{CMPO}] = [\overline{CMPO}]_t - KC1\,[\overline{CMPO}][H^+][NO_3^-]y_{12}^2 - KC2\,[\overline{CMPO}][H^+]^2[NO_3^-]^2y_{12}^4 \tag{15}$$

$[\overline{CMPO}]$ について一次であるこの式を整理すると、

$$[\overline{CMPO}]\,(\,KC2\,[H^+]^2[NO_3^-]^2y_{12}^4 + KC1\,[H^+][NO_3^-]y_{12}^2 + 1\,) = [\overline{CMPO}]_t \tag{16}$$

となる。解として求めた $[\overline{CMPO}]$ を (13) 式に代入すると、有機相中の CMPO により抽出された硝酸濃度が求められる。

3.3 活動度係数

Chaiko らは硝酸の活動度係数に Bromley[9]の報告している計算方法を用いているが、今回の作業では、簡単のため、硝酸の活動度係数には Davis らの報告[10]のデータを使用した。Chaiko らも一部の濃度範囲では、Davis らの求めた数値をそのまま使用している。水溶液中の硝酸濃度に対する硝酸溶液の平均イオン活動度を図 3-1 に示す。図中には、硝酸濃度が 0.001 から 10.0 M までの平均イオン活動度係数の値をプロットし、それらを 4 次式でフィッティングした結果を実線で表した。

活動度の近似式としては次式を用いた。

$$y_{12} = 0.8872 - 0.2384 \times [HNO_3] + 0.1401 \times [HNO_3]^2 - 0.0138 \times [HNO_3]^3 + 0.000881 \times [HNO_3]^4 \tag{17}$$

3.4 硝酸の分配比

TRUEX溶媒により抽出される硝酸は、TBP と CMPO にそれぞれ抽出されたものの和であるから、

$$[\overline{HNO_3}]_t = [\overline{HNO_3}]_{TBP} + [\overline{HNO_3}]_{CMPO} \tag{18}$$

と書ける。$[\overline{HNO_3}]_t$ は有機相中の全硝酸濃度である。これを水相の硝酸濃度で除すると分配比が得られる。

$$D_{HNO3} = \overline{[HNO_3]_l} \,/\, \overline{[HNO_3]_{aq}} \tag{19}$$

この分配比をシミュレーションコードに組み込む。

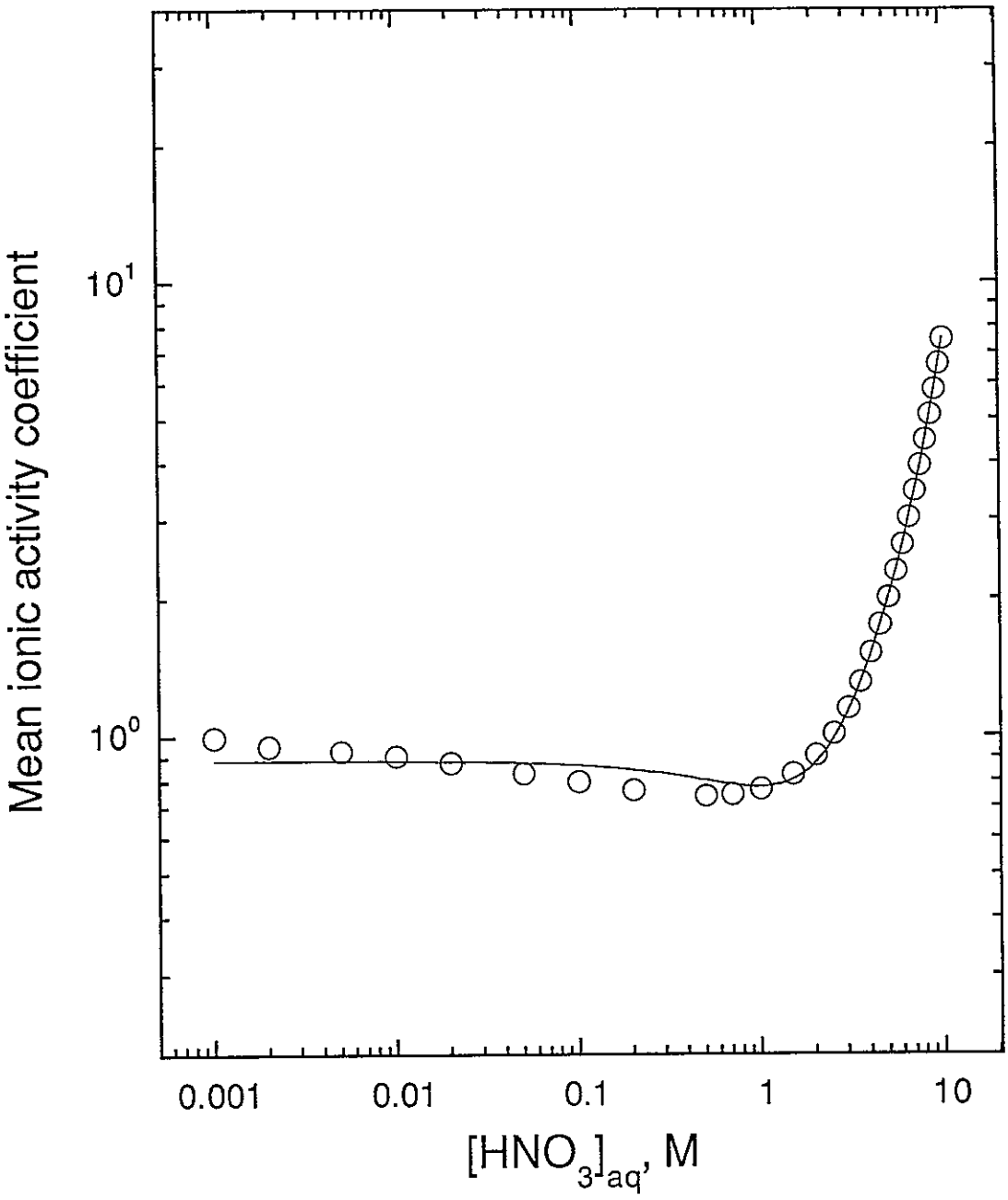

図 3-1　硝酸の平均イオン活動度の文献値と計算に用いた曲線

4. 計算コードへの組み込み

4.1 向流多段抽出計算コードの選択

向流接触の多段抽出器を使用した溶媒抽出のシミュレーションコードとして事業団内では、"MIXSET"が広く利用されている。"MIXSET"は、1977年に作成されて[11]以来、改良・拡張が加えられており[12,13]、最近では、パーソナルコンピュータ上で動作しFP成分の分配比も組み込んだものが作成された[14]。

一方、その他のシミュレーションコードとしては、Oak Ridge National Laboratory が開発し、"MIXSET"の分配比等の計算方法にも影響を与えている"SEPHIS"がある。"SEPHIS"は、液体金属冷却高速増殖炉を対象として希釈Purexプロセスをモデル化して、1972年に発表された[15]。"SEPHIS"も数多くの改善がなされ[16-18]、現在では、"MOD4"と呼ばれる版が用いられている。"SEPHIS"もパーソナルコンピュータ上で動作するように移植されている。"MIXSET"も大形計算機上と同様に FORTRAN で記述されているが、"SEPHIS"は移植に当たって BASIC で書き直されており[19,20]、"MOD4 Ver. 2.11"となっている。

今回の作業は、作業量を軽減するためにソースコードの量が少ない"SEPHIS"を選定した。作業には、パーソナルコンピュータ（日本電気（株） PC-9801 CV21）を使用した。また、ソースコードの記述は、パーソナルコンピュータやワークステーションのプログラムの記述で主流となっている C 言語を選択し、ボーランド社 TURBO C++ を利用した。

"SEPHIS"は Purexプロセスだけでなく、トリウム燃料再処理の Thorexプロセスも計算できる仕様になっていたが、C 言語に移植する際にこの Thorexプロセス計算に係る部分は削除している。

4.2 分配比のコード化

前述の方法に基づいてコードとしたものを図 4-1 に示す。計算に係わる部分は、活動度を計算する関数（calc_activity）、およびその値を利用して分配比を求める関数（calc_distribution）からなる。calc_distribution は、水相中の H^+、HNO_3^- の濃度、有機相のCMPOとTBPの濃度を引数とする。

これの関数を元の"SEPHIS"の該当する部分と入れ換え、TRUEXプロセスの硝酸の抽出のみを計算できるようにした。ただし、その他の計算条件の入力、密度計算等、分配計算以外の部分についてはまったく手を加えていない。そのため、今回作成したコードについては、CMPO と TBP の濃度はそれぞれ 0.2 M、1.0 M と固定し、これらを変化させるような入力条件を与えるような変更は施していない。

また、"SEPHIS"はTBP－炭化水素混合溶媒を対象としており、10 から 30 % のTBP濃度に対応でき、

溶媒密度の計算を内部で行う。TRUEX溶媒は、通常、1.0 M から1.4 M TBPの濃度で使用するが、溶媒の密度を正確に計算させるためには、TBPおよびCMPOの濃度に応じてTBP濃度を入力する必要がある。

```
double calc_activity (double h)
{
        double y;

        y = 8.8810e-4 * h - 1.383e-2;
        y *= h;
        y += 1.401e-1;
        y *= h;
        y -= 2.384e-1;
        y *= h;
        y += 0.8872;

        return y;
}

double calc_distribution (double h, double no3, double cmpo, double tbp)
{
        double d, y, hno3, tbp_f, cmpo_f, c2, c1;

        y = calc_activity(h);

        c2 = 2.0 * KT2 * h * no3 * y * y;
        c1 = KT1 * h * no3 * y * y + 1.0;
        if (c2 < 1.0e-10)
                tbp_f = 0.0;
        else
                tbp_f = (- c1 + sqrt(c1 * c1 + 4.0 * tbp * c2)) / 2.0 / c2;

        c1 = h * no3 * y * y;
        c2 = KC2 * c1 * c1 + KC1 * c1 + 1.0;
        if (c2 < 1.0e-10)
                cmpo_f = 0.0;
        else
                cmpo_f = cmpo / c2;

        hno3 = KT1 * tbp_f * h * no3 * y * y;
        hno3 += KT2 * tbp_f * tbp_f * h * no3 * y * y;
        hno3 += KC1 * cmpo_f * h * no3 * y * y;
        hno3 += 2.0 * KC2 * cmpo_f * h * h * no3 * no3 * y * y * y * y;

        if (hno3 < 1.0e-10)
                d = 0.0;
        else
                d = hno3 / h;

        return d;
}
```

図 4-1　活動度および分配比の計算に係る関数

5. 計算の妥当性

5.1 硝酸濃度およびTBP濃度依存性

コード化した分配比は、Chaicoらの計算とほぼ一致していた。また、計算した値を我々がバッチ試験で求めた値[21]と比較した様子を図5-1に示す。

TBP濃度が0 Mでの値が異なっているものの、通常使用されるTBPの濃度範囲である0.7 Mから1.4 Mでは、実験値と良く一致していることが分かる。

5.2 連続試験への適用

向流多段抽出を行ったホット試験（以下、「ホット試験」という）の結果と計算値を比較する。試験の例としては、第1回および第2回ホット試験[4,5]を取り上げる。

第1回ホット試験のフローシートを図5-2に、計算の入力条件を図5-3にそれぞれ示す。入力データとして注意するべき項目は前述のように、TBP濃度であるが、0.2 M CMPO－1.1 M TBPに相当する値である36 %の値で計算させた。

計算の結果を試験での分析値とともに図5-4に示す。この図において、抽出溶媒を供給する抽出部が1段であり、逆抽出バンクで溶媒が供給されるのが17段、廃溶媒が排出されるのが、32段である。図から分かるように、抽出部での試験の値は計算値と良く一致しているが、硝酸濃度が0.1 M以下となる領域では、分析値が計算値を大きく上回る傾向が見られる。特に、有機相中の硝酸濃度の差異が大きく、これには分析誤差が含まれているものと考えられる。

第2回ホット試験のフローシートを図5-5に、計算の入力条件を図5-6にそれぞれ示す。この試験では洗浄にダブルスクラブを採用しており、高レベル廃液供給段と第1スクラブ液供給段の間の硝酸濃度が、抽出部よりも高くなるようなフローシートである。TBP濃度としては33 %を入力した。

また、計算の結果を図5-7にそれぞれ示す。水相中の硝酸濃度は、すべての領域で良く一致している。有機相についても、逆抽出部での値を除きほぼ一致している。

以上の比較から、硝酸抽出に関しては計算により濃度プロファイルを十分予測できることが確認された。

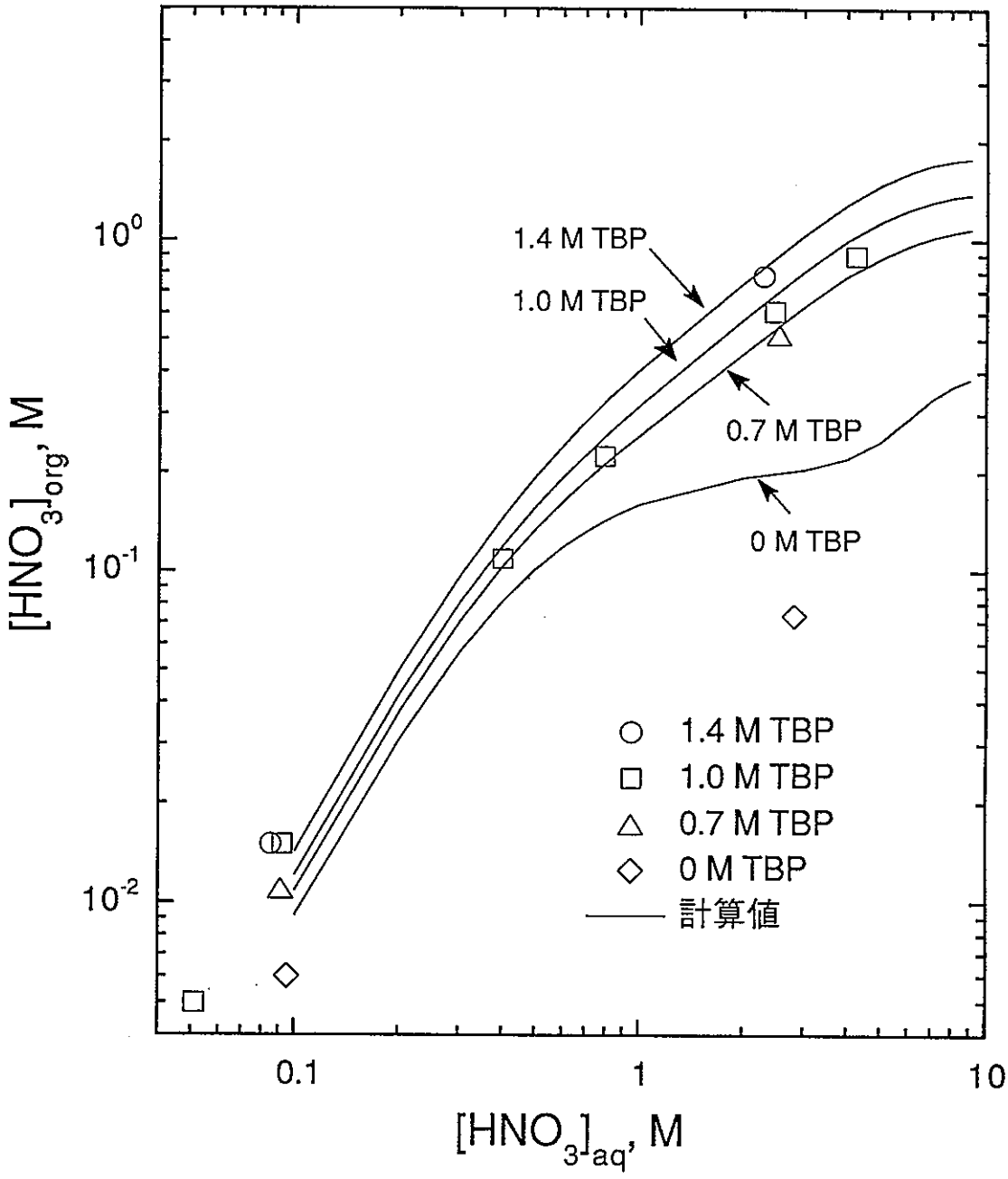

図 5-1　硝酸の水相と有機相間の分配の計算値との比較

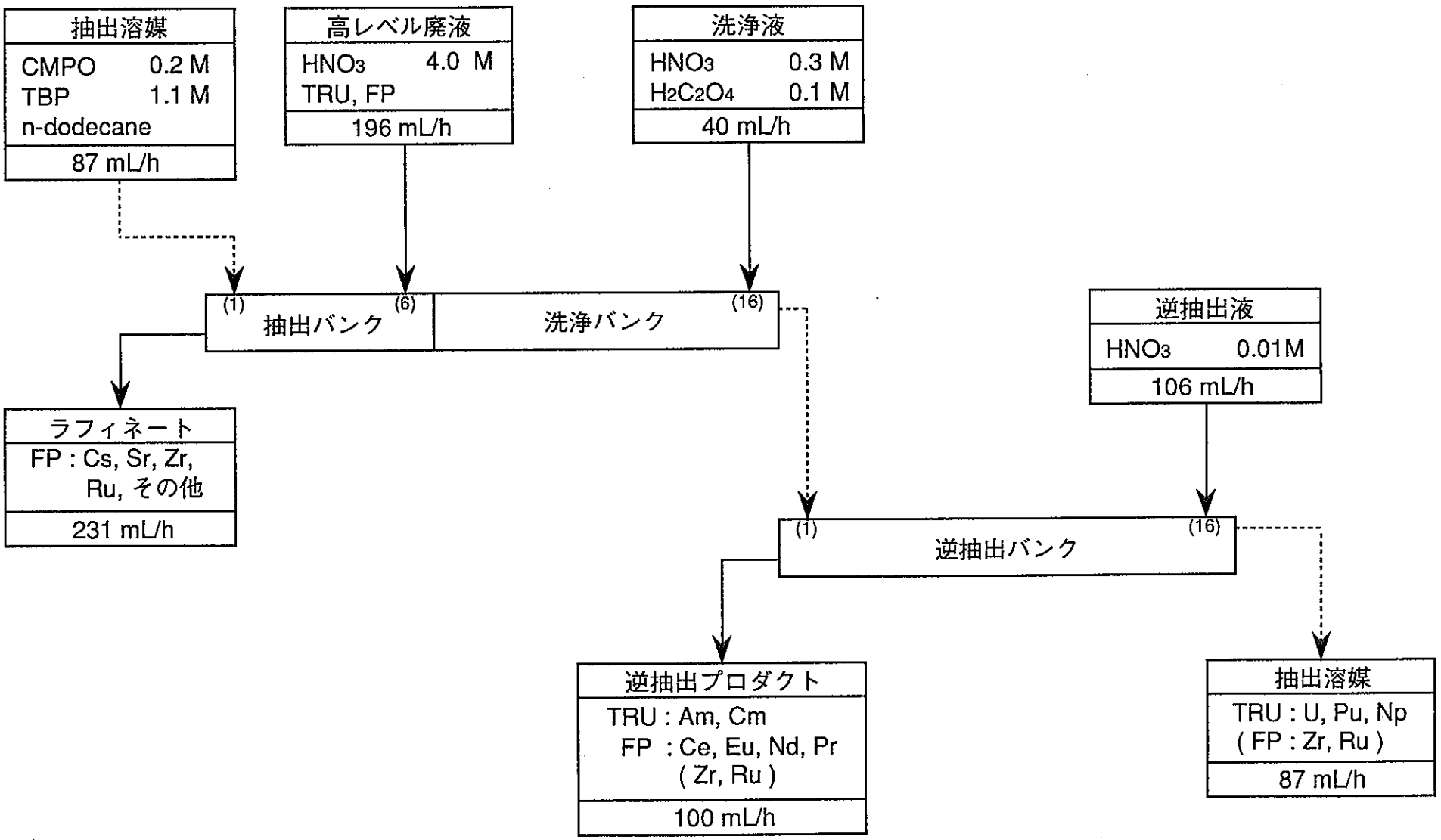

図 5-2 第 1 回の向流多段ホット試験のフローシート

PROBLEM DEFINITION DATA FOR SEPHIS MOD4, Ver. 2.11:

Probrem: TRUEX - 1st run -

PUREX PROCESS
Total stages: 32
Volume % TBP in solvent: 36.0
Initial or default temperature (flC): 30.0
No unusual stage connections
No Plutonium reaction.
Minutes per time increment: 1.00
Minutes between printing of the concentration profile: 50
Calculations will stop after 200 minutes or when a tolerance of 0.1000 % per minute is reached.
Total mixer volumes are defined (phase ratios = flow ratios).
Total settler volumes are defined (phase ratios = flow ratios).
Initial concentration profile is zero.
Runge-Ketta integration will be used.
Extra product stream(s).

The FILESPEC for this Problem Definition data file is: 1st.dat
The FILESPEC for this Problem Results data file is: 1st.p2

FEED AND PRODUCT STREAM DATA:

	STAGE NO.	NITRIC ACID (M)	URANIUM (G/L)	PU (IV) (G/L)	PU (III) (G/L)	REDUCTANT (M)	NITRATE ION (M)	FLOW RATE (L/min)	TEMP (C)
AQUEOUS	1	1.00e-02	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	1.80e+00	30.0
AQUEOUS	17	3.00e-01	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	6.70e-01	30.0
AQUEOUS	27	4.50e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	3.30e+00	30.0
36.0 % TBP	32	0.00e+00	0.00e+00	0.00e+00				1.45e+00	30.0
AQUEOUS	16	Product stream removed (actual flow rate to be computed)						1.30e+00	

図 5-3 第 1 回の向流多段ホット試験を模擬した計算の入力条件

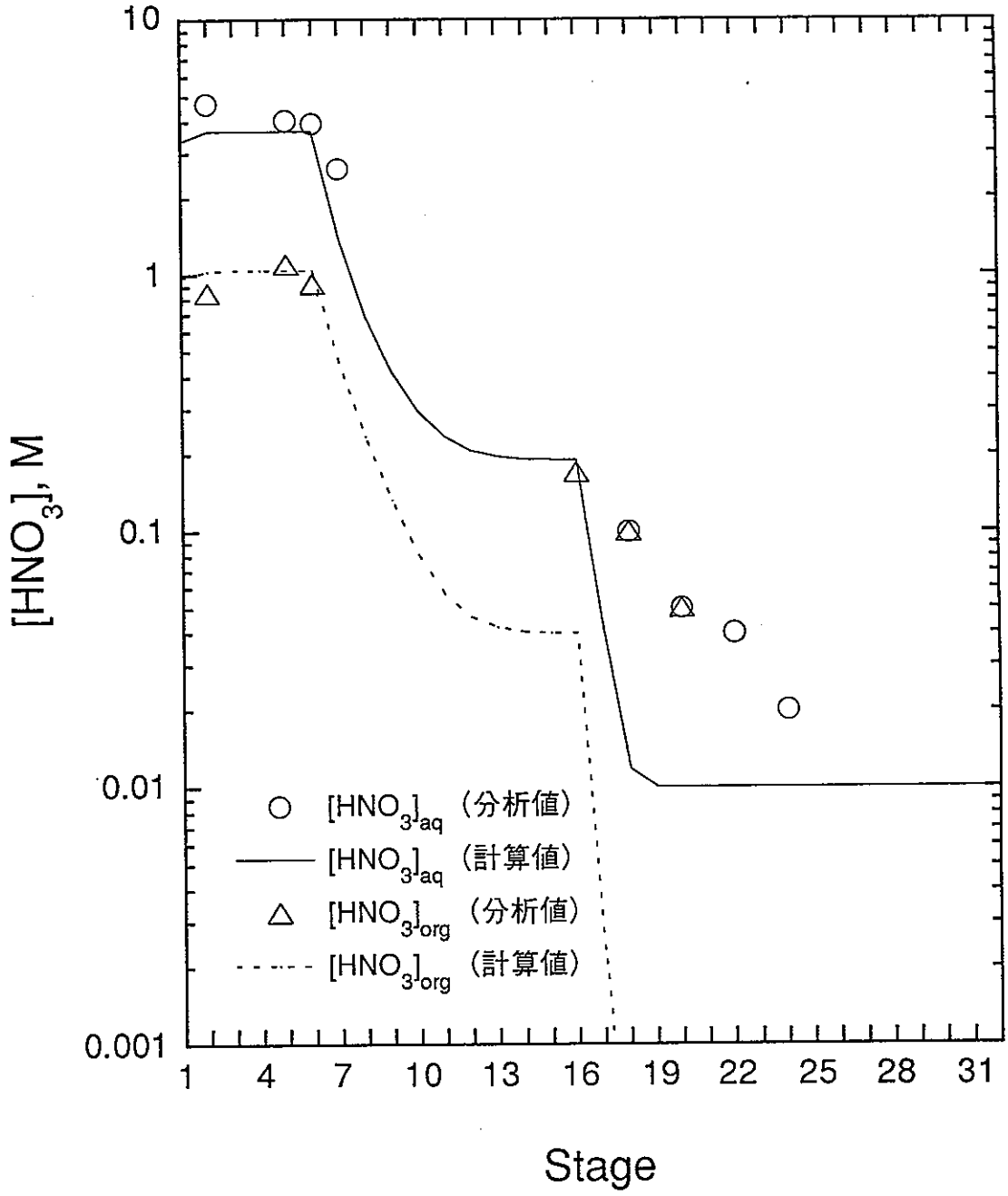

図 5-4 第 1 回向流多段ホット試験の濃度プロファイル

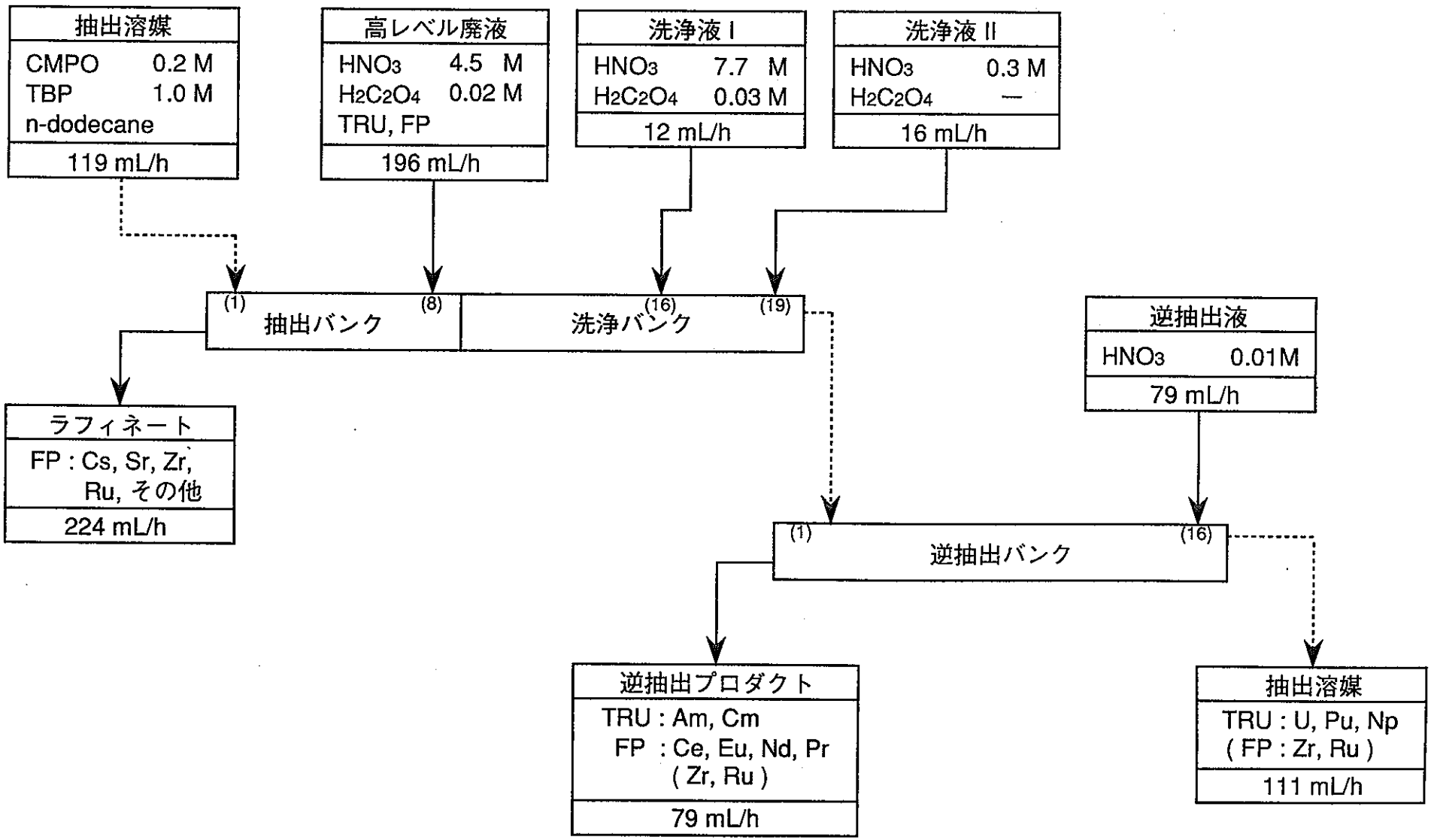

図 5-5 第 2 回の向流多段ホット試験のフローシート

```
PROBLEM DEFINITION DATA FOR SEPHIS MOD4, Ver. 2.11:

     Probrem:  TRUEX  - 2nd run -


          PUREX PROCESS
          Total stages:  35
          Volume % TBP in solvent:   33.0
          Initial or default temperature (fIC):   30.0
          No unusual stage connections
          No Plutonium reaction.
          Minutes per time increment:     1.00
          Minutes between printing of the concentration profile:    50
          Calculations will stop after  200 minutes or when a tolerance of   0.1000 % per minute is reached.
          Total mixer volumes are defined (phase ratios = flow ratios).
          Total settler volumes are defined (phase ratios = flow ratios).
          Initial concentration profile is zero.
          Runge-Ketta integration will be used.
          Extra product stream(s).

          The FILESPEC for this Problem Definition data file is:  2nd.dat
          The FILESPEC for this Problem Results data file is:  2nd.p2
```

FEED AND PRODUCT STREAM DATA:

	STAGE NO.	NITRIC ACID (M)	URANIUM (G/L)	PU (IV) (G/L)	PU (III) (G/L)	REDUCTANT (M)	NITRATE ION (M)	FLOW RATE (L/min)	TEMP (C)
AQUEOUS	1	1.00e-02	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	1.30e+00	30.0
AQUEOUS	17	3.00e-01	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	2.70e-01	30.0
AQUEOUS	20	7.70e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	2.00e-01	30.0
AQUEOUS	28	4.50e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	3.30e+00	30.0
33.0 % TBP	35	0.00e+00	0.00e+00	0.00e+00				2.00e+00	30.0
AQUEOUS	16	Product stream removed (actual flow rate to be computed)						1.30e+00	

図 5-6 第 2 回の向流多段ホット試験を模擬した計算の入力条件

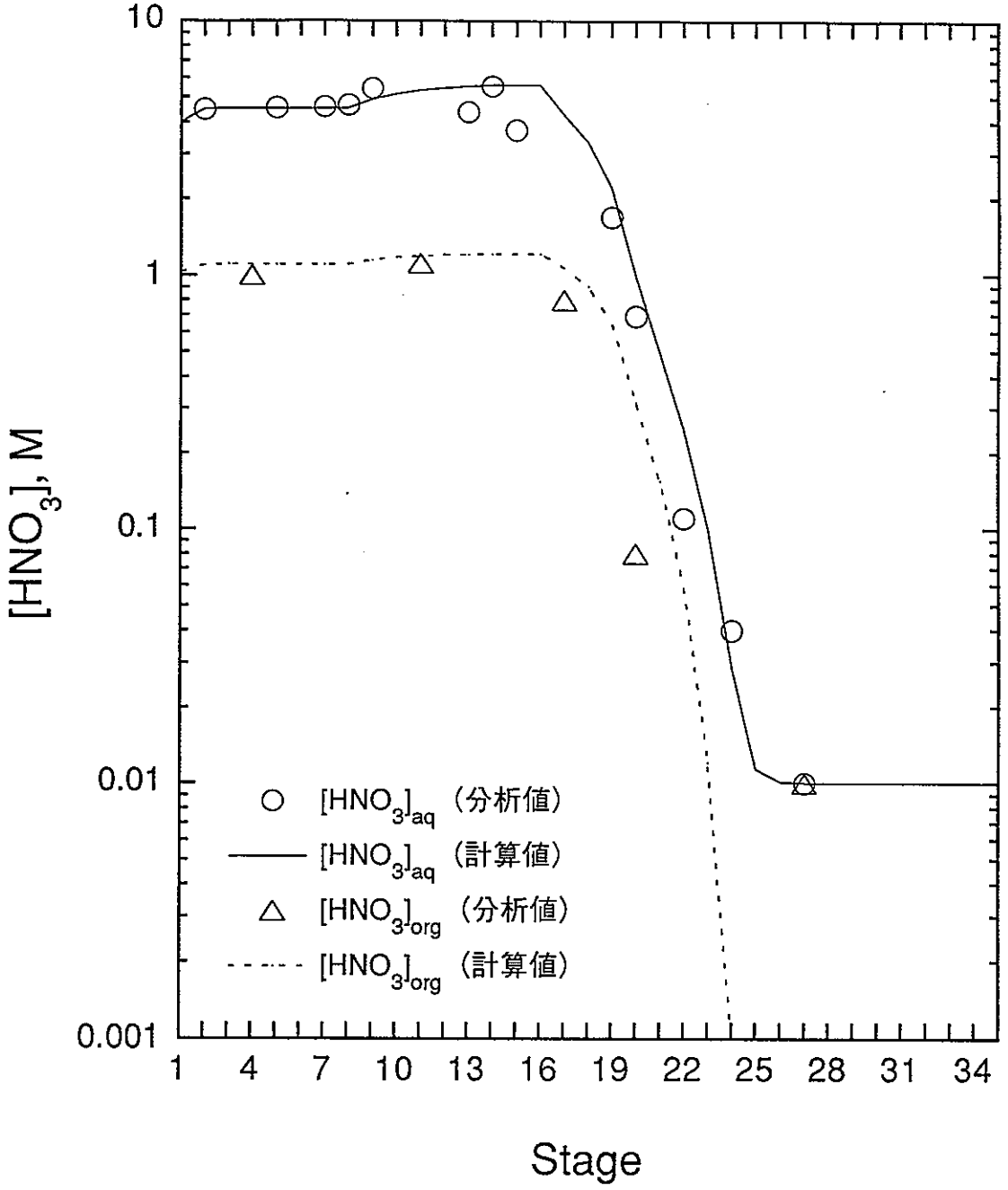

図 5-7 第 2 回向流多段ホット試験の濃度プロファイル

6. おわりに

高レベル廃液からIII価の超ウラン元素が抽出できるCMPO－TBP混合溶媒を使用するプロセスのシミュレーションコードを開発するため、最も基礎的な硝酸抽出についてモデル化し、それを基に計算コード化した。

硝酸はCMPOとTBPのそれぞれにより抽出され、抽出剤分子間の相互作用はないものと仮定して、個々の抽出平衡の寄与を求めることで計算した。

我々のバッチ試験の結果と比較したところ、CMPO混合溶媒が通常使用される領域である、0.7から1.4 M の TBP の濃度範囲で良く一致した。また、ミキサセトラを使用した向流多段ホット試験の濃度プロファイルとの比較では、0.1 M 以上の硝酸濃度の領域では良く一致した。

以上のように、硝酸抽出のモデル化・コード化については、満足できる結果が得られた。しかし、TRUEX法のシミュレーションコードを作成するためには、硝酸溶液中でIII価であるAm、Cmおよび希土類元素、さらに、UやPuの計算が不可欠である。III価金属の抽出のモデル化については、現在、進行中であり、引き続き分配比のコード化を行いたい。また、CMPO混合溶媒には第三相を生成しやすいという特徴があるが、この事象を予め予測することも望まれる。今後は、ベースとなるコードである"SEPHIS"の構造に関する見直しも含め、これらの点について作業を進める予定である。

7. 参考文献

1) W. W. Schulz, E. P. Horwitz, "The TRUEX Process and the Management of Liquid TRU Waste", Sep. Sci. Tech., 23(12&13), 1355-1372, 1988

2) E. P. Horwitz, W. W. Schulz, "The TRUEX Process: A Vital Tool for Disposal of U. S. Defense Nuclear Waste", New Separations Chemistry for Radioactive Waste and Other Specific Applications", Italy, May 16-18, 1990

3) C. Cuillerder, C. Musikas, P. Hoel, L. Nigond, X. Vitart, "Malonamides as New Extractants for Nuclear Waste Solutions", Sep. Sci. Tech., 26(9), 1299-1244, 1991

4) "湿式核種分離技術研究 ー平成二年度実施概要及び今後の計画ー", PNC ZN8410 91-171, 1991

5) "湿式核種分離技術研究 ー平成三年度実施概要及び今後の計画ー", PNC ZN8410 92-172, 1992

6) M. Ozawa, S. Nemoto, A. Togashi, T. Kawata, K. Onishi, "Partitioning of Actinides and Fission Products in Highly-Active Raffinate from Purex Process by Mixer-Settlers", Solv. Extr. Ion Exch., 10(5), 829-846, 1992

7) D. J. Chaico, D. R. Freidrickson, L. Reighley-Yinger, G. F. Vandegrift, "Thermodynamic Modeling of Chemical Equilibria in Metal Extraction", Sep. Sci. Tech., 23, 1435, 1988

8) R. A. Leonard, "Use of Electonic Worksheets for Calculation of Stagewise Solvent Extraction Processes", Sep. Sci. and Tech., 22, 535, 1987

9) L. A. Bromley, "Thermodynamic Properties of Strong Electrolytes in Aqueous Solutions", AIChE J., 19, 313, 1973

10) W. Davis, Jr., H. J. DE Bruin, J. Inorg. Nucl. Chem., "New Activity Coefficients of 0 - 100 per Cent Aqueous Nitric Acid", 26, 1069, 1964

11) "Purexプロセス計算コード MIXSET", PNCT 841-77-60, 1977

12) "Purexプロセス計算コード Revised MIXSET", PNCT 841-79-26, 1979

13) "Purexプロセスにおける超ウラン元素の分配計算コード Advanced MIXSET", PNCT 841-81-35, 1981

14) "抽出計算コード(MIXSET)のパーソナルコンピュータへの移植と拡張", 住友金属鉱山株式会社 作業報告書, 1991

15) W. S. Groenier, "Calculation of the Transient Behavior of a Dilute-Purex Solvent Extraction Process Having Application to the Reprocessing of LMFBR Fuels", ORNL-4746, 1972

16) G. L. Richardson, "Effect of High Solvent Irradiation Exposures on TBP Processing of Spent LMFBR Fuel", HEDL-TME 73-51, 1973

17) S. B. Watson, R. H. Rainy, "Modifications of the SEPHIS Computer Code for Calculating the Purex

Solvent Extraction System", ORNL/TM-5123, 1975

18) A. D Mitchell, "A Comparison Between SEPHIS-MOD4 and Previous Model of the Purex Solvent Extraction System", ORNL/TM-6565, 1979

19) W. S. Groenier, "User's manual for SEPHIS MOD 4, Version 2.11", ORNL/TM-11588, 1991

20) W. S. Groenier, "Technical manual for SEPHIS MOD 4, Version 2.11", ORNL/TM-11589, 1991

21) "湿式核種分離研究 ー平成２年度コールド試験報告書ー", PNC ZN8410 91-189, 1991

8. 計算コードリスト

ソースコードは以下の９つのファイルから構成される。

SEPHIS.H
C.C
C_CALC.C
C_CHEM.C
C_STAGE.C
C_INIT.C
C_DISP.C
C_PRN.C
ETC.C

本章では、ヘッダファイルである "SEPHIS.H" とその他のソースコードのリストを示す。

8.1 ヘッダファイルリスト

変数、構造体や関数の宣言を含むヘッダファイルのリストを次に示す。

8.1.1 SEPHIS.H

```
/*    header file for sephis     */

#define STAGE 50
#define STRING 255

typedef struct {
        char s_date[9], s_time[9], title[STRING+1];
        int iproce, ntost;
        double ctbp, tempi;
        int nstr, istr, jstr, irxn;
        double dtheta, dprint, tstop, tol;
        int ivolm, ivols, ipro, ifast, newout;
} General;

typedef struct {
        double xfd[6], yfd[3], afdrt, ofdrt, afdtem, ofdtem;
} Feed;

typedef struct {
        double alvrt, olvrt;
} Product;

typedef struct {
        double x[2][6], y[2][6], xs[2][3][6], ys[2][3][6];
        double tprof[2], rxntrm[6], ats[2][3], ots[2][3];
} Profile;

typedef struct {
        double at, ot, anet, onet;
} Cascade;
```

```
typedef struct {
        double avol, ovol;
} Volume;

typedef struct {
        double codum, codumm[6];
} Change;

typedef struct {
        double aden, oden;
        double aout, oout, aaout, ooout;
        double tsat, ext1, ext2, ext3;
} Etc;


/*      c_init.c                */

void set_general ( FILE *fp, General *general );
int set_feed ( FILE *fp, Feed *feed, General *general );
int set_product ( FILE *fp, Product *product );
int set_profile ( FILE *fp, Profile *profile, General *general );
void conv_feed ( Feed *feed, General *general );
void set_cascade ( Cascade *cascade, General *general, Product *product, Feed *feed );
void conv_profile_aq ( Profile *profile, General *general );
void conv_profile_org ( Profile *profile, General *general );
void set_volume ( FILE *fp, int ivol, int ntost, Volume *volume, Cascade *cascade );
void initial_equilibrium ( General *general, Profile *profile, Volume *v_mixer, Volume *v_settler );


/*      c_calc.c                */

void total_solutes ( double soin[], Feed *feed, General *general );
void transfer_aq ( General *general, int stage, double ain[], double deltax[], Feed *feed, Profile *profile, Cascade
*cascade );
void transfer_org ( General *general, int stage, double oin[], double deltay[], Feed *feed, Profile *profile, Cascade
*cascade );
double heat_transfer_aq ( int i, Feed *feed, Profile *profile, Cascade *cascade );
double heat_transfer_org ( int i, General *general, Feed *feed, Profile *profile, Cascade *cascade );
void ucorpu ( General *general, double dtry[], double ary[], double tempc );
double molal_aq ( double aq[] );
double molal_org ( double org[], double ctbp, double t, double temp );
double molar_aq ( double aq[] );
double molar_org ( double org[], double ctbp, double t, double temp );


/* c_stage.c        */

void mixer_runge_kutta ( General *general, Profile *profile, Cascade *cascade, Volume *mixer, double ain[], double
oin[], double dtry[] );
void mixer_trapezoidal ( General *general, Cascade *cascade, Profile *profile, Volume *mixer, double ain[], double
oin[], double dtry[] );
void mixer_fast ( General *general, Profile *profile, Cascade *cascade, Volume *mixer, double *deltax, double *deltay,
double *ain, double *oin, double dtry[] );
void settler_trapezoidal ( General *general, Profile *profile, Cascade *cascade, Volume *settler );
void calc_inventory_change ( General *general, Profile *profile, Volume *v_mixer, Change *change );
int stages ( int ndirec, int isol5, General *general, Feed *feed, Profile *profile, Cascade *cascade, Change *change,
Volume *v_mixer, Volume *v_settler );
```

```
/*    c_chem.c    */

void mchemp ( General *general, Profile *profile, Volume *mixer, double *ain, double *oin, double dtry[] );
void schemp ( General *general, Profile *profile );


/*    c_disp.c            */

void display_general_2 ( General *general, double prtime );
void status_screen ( void );
void main_title ( void );
void display_stage_number ( int stage );
void display_inventory_change ( General *general, Change *change );
void display_inventory_bar ( double *sot, double *soin );


/*    c_prn.c             */


void printout_problem ( General *general, FILE *fp1, char *file1, char *file2 );void printout_streams_header ( FILE *fp
);
void printout_feed_streams ( FILE *fp, General *general, Feed *feed );
void printout_product_streams ( FILE *fp, int ntost, Product *product );
void printout_aqueous_profile_header ( FILE *fp );
void printout_aqueous_profile ( FILE *fp, int ntost, Profile *profile );
void printout_organic_profile_header ( FILE *fp );
void printout_organic_profile ( FILE *fp, int ntost, Profile *profile );
void printout_volumes_header ( FILE *fp );
void printout_volumes ( FILE *fp, int ntost, Volume *mixer, Volume *settler, Cascade *cascade );
void printout_result_header ( FILE *fp, General *general, char *s_date, char *s_time );
void printout_result_aqueous_header ( FILE *fp );
void printout_result_organic_header ( FILE *fp );
void printout_result_aqueous ( FILE *fp, int i, Profile *profile, Product *product, Etc *etc, double conva );
void printout_result_organic ( FILE *fp, int i, Profile *profile, Cascade *cascade, Product *product, Etc *etc, Change
*change );
void printout_result ( FILE *fp, General *general, Profile *profile, Product *product, Cascade *cascade, Etc *etc,
Change *change );
void printout_footer ( FILE *fp );


/*    etc.c  */

char *string_time ( char *s_time );
char *string_date ( char *s_date );
void enable_bottom_line ( void );
void disable_bottom_line ( void );
void elase_cursur ( void );
void display_cursur ( void );
void draw_box ( int r1, int c1, int r2, int c2 );
```

8.2 関数コードリスト

8.2.1 C.C

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <conio.h>
#include <math.h>

#include "sephis.h"


int main ( void )
{
        General general;
        Feed *feed;
        Product *product;
        Profile *profile;
        Cascade *cascade;
        Volume *v_mixer, *v_settler;
        Etc *etc;
        Change *change;
        double soin[4], sot[4], mb[3], mmb, totime, prtime;
        FILE *fp1, *fp2, *fp3;
        int ndirec, nstop, nbort, nprnt, nmaxt, tome, nstg, isol5, reply;
        char buffer[STRING+3];
        char datfil1[STRING+1], datfil2[STRING+1], datfil3[STRING+1];
        char s_date[9], s_time[9];

        int i, j, k, l, v, rl;

        v = 6;
        ndirec = 1;

        feed = (Feed *) calloc(STAGE, sizeof (Feed));
        if (feed == NULL) {
                puts("Not enough memory.");
                return 0;
        }

        product = (Product *)calloc(STAGE, sizeof (Product));
        if (product == NULL) {
                puts("Not enough memory.");
                return 0;
        }

        profile = (Profile *) calloc(STAGE, sizeof (Profile));
        if (profile == NULL) {
                puts("Not enough memory.");
                return 0;
        }

        cascade = (Cascade *) calloc(STAGE, sizeof (Cascade));
        if (cascade == NULL) {
```

```
        puts("Not enough memory.");
        return 0;
    }

    v_mixer = (Volume *) calloc(STAGE, sizeof (Volume));
    if (v_mixer == NULL) {
        puts("Not enough memory.");
        return 0;
    }

    v_settler = (Volume *) calloc(STAGE, sizeof (Volume));
    if (v_settler == NULL) {
        puts("Not enough memory.");
        return 0;
    }

    etc = (Etc *) calloc(STAGE, sizeof (Etc));
    if (etc == NULL) {
        puts("Not enough memory.");
        return 0;
    }

    change = (Change *) calloc(STAGE, sizeof (Change));
    if (change == NULL) {
        puts("Not enough memory.");
        return 0;
    }

    clrscr( );
    enable_bottom_line( );

    main_title( );

    gotoxy(1, 24);
    cputs("Do you want to continue (Y/N)? ");
    reply = getche( );

    if (reply != 'Y' && reply != 'y') {
        disable_bottom_line( );
        return 0;
    }

    gotoxy(1, 24);
    clreol( );

    cputs("Will the fill on a diskette (Y/N)? ");
    reply = getche( );

    if (reply == 'Y' || reply == 'y') {
        gotoxy(1, 24);
        clreol( );

        cputs("Insert diskette -- Press any key to continue.");
        getch( );
    }

    gotoxy(1, 25);
```

```
        clreol();
        gotoxy(1,24);
        clreol();

        cputs("Enter drive (Path) and FILENAME of Problem Definition data file.");

        gotoxy(1,25);
        cputs("Make your entry here. ");

        buffer[0] = (char)STRING;
        cgets(buffer);
        strcpy(datfil1, buffer+2);
        strcat(datfil1, ".dat");

        strcpy(datfil2, buffer+2);
        strcat(datfil2, ".p2");

        strcpy(datfil3, buffer+2);
        strcat(datfil3, ".rsl");


        fp1 = fopen(datfil1, "rt");
        if (fp1 == NULL) {
                cputs("¥nFile not found¥n");
                fclose(fp1);
                disable_bottom_line();
                return 0;
    }

        elase_cursur();
        set_general(fp1, &general);

        if (general.iproce != 0) {
                display_cursur();
                cputs("¥nOnly Purex process is available¥n");
                fclose(fp1);
                disable_bottom_line();
                return 0;
        }

        fp2 = fopen(datfil2, "wt");
        if (fp2 == NULL) {
                display_cursur();
                cputs("¥nFile not opened¥n");
                fclose(fp1);
                disable_bottom_line();
                return 0;
    }

        fp3 = fopen(datfil3, "wt");
        if (fp3 == NULL) {
                display_cursur();
                cputs("¥nFile not opened¥n");
                fclose(fp1);
                disable_bottom_line();
                return 0;
    }
```

```
status_screen( );

printout_problem(&general, fp2, datfil1, datfil2);
gotoxy(1, 6);
cputs("Problem is defined.");

isol5 = set_feed(fp1, feed, &general);
if (general.newout == 1) set_product(fp1, product);
gotoxy(1, 7);
cputs("Feed and product streams are identified.");

printout_streams_header(fp2);
printout_feed_streams(fp2, &general, feed);
if (general.newout == 1)
        printout_product_streams(fp2, general.ntost, product);

if (set_profile(fp1, profile, &general) == 1) isol5 = 1;
gotoxy(1, 8);
cputs("Initial concentration profile is established.");

conv_feed(feed, &general);

set_cascade(cascade, &general, product, feed);
gotoxy(1, 9);
cputs("Cascade flows are established.");

if (general.ipro != 0) {
        printout_aqueous_profile_header(fp2);
        printout_aqueous_profile(fp2, general.ntost, profile);
        conv_profile_aq(profile, &general);

        printout_organic_profile_header(fp2);
        printout_organic_profile(fp2, general.ntost, profile);
        conv_profile_org(profile, &general);
}

set_volume(fp1, general.ivolm, general.ntost, v_mixer, cascade);
set_volume(fp1, general.ivols, general.ntost, v_settler, cascade);
gotoxy(1, 10);
cputs("Stage volumes and flows are defined.");

printout_volumes_header(fp2);
printout_volumes(fp2, general.ntost, v_mixer, v_settler, cascade );

fclose(fp1);

if (general.ipro != 0)
        initial_equilibrium(&general, profile, v_mixer, v_settler);

total_solutes(soin, feed, &general);

string_time(s_time);
string_date(s_date);

printout_result_header(fp2, &general, s_date, s_time);
```

```
        totime = prtime = 0.0;
        tome = 0;

        display_general_2(&general, prtime);

        nstop = nbort = nprnt = nmaxt = 1;
        nstg = 0;

        goto program_control;

    calculate_concentration:

        gotoxy(57, 6);
        cprintf("%7.2f", totime + general.dtheta);

        ndirec = stages(ndirec, isol5, &general, feed, profile, cascade, change, v_mixer, v_settler);

        ndirec *= -1;
        nstop = 0;
        nstg++;

        totime = general.dtheta * (double)nstg;

        nbort = nprnt = nmaxt = 1;
        tome = (int)totime;

        display_inventory_change(&general, change);

        for (i = 0; i < general.ntost; i++) {
            if (i == 0) {
                etc->ooout = cascade->ot;
                etc->aaout = product->alvrt;
            } else {
                if (i == general.ntost-1) {
                    (etc+general.ntost-1)->aaout = (cascade+general.ntost-1)->at;
                    (etc+general.ntost-1)->ooout = (product+general.ntost-1)->olvrt;
                } else {
                    (etc+i)->aaout = (product+i)->alvrt;
                    (etc+i)->ooout = (product+i)->olvrt;
                }
            }
        }

        for (i = 0; i < 4; i++) {
            sot[i] = 0.0;

            for (j = 0; j < general.ntost; j++) {
                sot[i] += (etc+j)->aaout * (profile+j)->xs[1][2][i];
            }
        }

        for (i = 0; i < 3; i++) {
            for (j = 0; j < general.ntost; j++) {
                sot[i] += (etc+j)->ooout * (profile+j)->ys[1][2][i];
            }
        }
```

```
        sot[2] += sot[3];

        display_inventory_bar(sot, soin);

        for (i = 0; i < general.ntost; i++) {
                if ((change+i)->codum > general.tol) nstop = 1;

                (profile+i)->tprof[0] = (profile+i)->tprof[1];

                for (j = 0; j < 3; j++) {
                        (profile+i)->ats[0][j] = (profile+i)->ats[1][j];
                        (profile+i)->ots[0][j] = (profile+i)->ots[1][j];
                }

                for (j = 0; j < 3; j++) {
                        (profile+i)->x[0][j] = (profile+i)->x[1][j];
                        (profile+i)->y[0][j] = (profile+i)->y[1][j];

                        for (k = 0; k < 3; k++) {
                                (profile+i)->xs[0][k][j] = (profile+i)->xs[1][k][j];
                                (profile+i)->ys[0][k][j] = (profile+i)->ys[1][k][j];
                        }
                }

                for (j = 3; j < v; j++) {
                        (profile+i)->x[0][j] = (profile+i)->x[1][j];

                        for (k = 0; k < 3; k++) {
                                (profile+i)->xs[0][k][j] = (profile+i)->xs[1][k][j];
                        }
                }
        }

        if (nstop == 0) goto stop_or_not;

        if (kbhit() != 0) {
                getch();
                nbort = 0;
                goto stop_or_not;
        }

program_control:

        if (totime >= general.tstop) {
                nmaxt = 0;
                goto stop_or_not;
        }

        if (fabs(totime - prtime) > 0.001) goto calculate_concentration;

        nprnt = 0;

        prtime += general.dprint;
        gotoxy(16, 13);
        cprintf("%4.0f", prtime);

stop_or_not:
```

```
        fprintf(fp2, "¥nTIME = %7.2f MINUTES¥n", totime);

        printout_result(fp2, &general, profile, product, cascade, etc, change);

        if (nstop == 0) goto steady_state;
        if (nbort == 0) goto any_key_pressed;
        if (nprnt == 0) goto continue_calculation;
        if (nmaxt == 0) goto reach_max_time;

    continue_calculation:

        for (i = 0; i < general.ntost; i++) {
            for (j = 0; j < v; j++)
                (profile+i)->x[1][j] = (profile+i)->x[0][j];

            for (j = 0; j < 3; j++)
                (profile+i)->y[1][j] = (profile+i)->y[0][j];
        }

    goto calculate_concentration;

    steady_state:


        fprintf(fp3, "%s%s¥n%s¥n", general.title, s_time, s_date);
        fprintf(fp3, "%f¥n", totime);

        for (i = 0; i < general.ntost; i++) {
            fprintf(fp3, "%d,%f,", i+1, (profile+i)->x[1][0]);
            fprintf(fp3, "%f,%f¥n", (profile+i)->x[1][1], (profile+i)->x[1][2]);
            fprintf(fp3, "%f,%f,%f¥n", (profile+i)->y[1][0], (profile+i)->y[1][1], (profile+i)->y[1][2]);
            fprintf(fp3, "%f,%f,%f¥n", (profile+i)->x[1][3], (profile+i)->x[1][4], (profile+i)->x[1][5]);
            fprintf(fp3, "%f,%f,%f¥n", (etc+i)->aden, (etc+i)->aout, (profile+i)->tprof[1]);
            fprintf(fp3, "%f,%f,%f¥n", (etc+i)->ext2, (etc+i)->ext3, (etc+i)->ext1);
            fprintf(fp3, "%f,%f,", (etc+i)->oden, (etc+i)->oout);
            fprintf(fp3, "%f,%f¥n", (etc+i)->tsat, (change+i)->codum);
        }


        printout_footer(fp2);

        if (nbort == 0 || nmaxt == 0) goto terminate;

        gotoxy(1, 3);
        for (i=0; i < 35; i++) putch(' ');
        gotoxy(1, 4);
        for (i=0; i < 35; i++) putch(' ');

        textcolor(CYAN);
        gotoxy(1, 3);
        cprintf("Steady-state condition is met ---");
        gotoxy(1, 4);
        cprintf("   Press any key to continue   ");
        textcolor(WHITE);

        getch();
```

```
        goto terminate;

any_key_pressed:

        gotoxy(1, 3);
        for (i=0; i < 35; i++) putch(' ');
        gotoxy(1, 4);
        for (i=0; i < 35; i++) putch(' ');

        textcolor(CYAN);
        gotoxy(1, 3);
        cprintf("Calculations stopped ----");
        gotoxy(1, 4);
        cprintf("Press any key to continue");
        textcolor(WHITE);

        getch( );

        goto continue_or_not;

reach_max_time:

        gotoxy(1, 3);
        for (i=0; i < 35; i++) putch(' ');
        gotoxy(1, 4);
        for (i=0; i < 35; i++) putch(' ');

        textcolor(CYAN);
        gotoxy(1, 3);
        cprintf("Maximum time reached ----");
        gotoxy(1, 4);
        cprintf("   Press any key to continue");
        textcolor(WHITE);

        getch( );

continue_or_not:

        gotoxy(1, 3);
        for (i=0; i < 35; i++) putch(' ');
        gotoxy(1, 4);
        for (i=0; i < 35; i++) putch(' ');

        textcolor(CYAN);
        gotoxy(1, 3);
        cprintf("Do you want to continue for more time (Y/N)? ");
        textcolor(WHITE);

        reply = getch( );

        if (reply != 'Y' && reply != 'y')
                goto steady_state;
        else {
                if (nmaxt == 0) general.tstop *= 2.0;

                gotoxy(16, 14);
```

```
            cprintf("%4.0f", general.tstop);
            gotoxy(1,3);
            for (i=0; i < 50; i++) putch(' ');

            textcolor(CYAN);
            gotoxy(1,3);
            cprintf("    CALCULATING --- PLEASE WAIT  ");
            gotoxy(1,4);
            cprintf("    Press any key to quit    ");
            textcolor(WHITE);

            goto continue_calculation;
        }

    terminate:


        fclose(fp2);
        fclose(fp3);

        display_cursur( );
        disable_bottom_line( );

        return 1;
    }
```

8.2.2 C_CALC.C

```
#include <stdio.h>
#include <math.h>

#include "sephis.h"

#define KT1 0.16
#define KT2 0.24
#define KC1 6.4
#define KC2 0.02

void total_solutes ( double soin[], Feed *feed, General *general )
{
	int i, j;

	for (i = 0; i < 4; i++)
		soin[i] = 0.0;

	for (i = 0; i < general->ntost; i++) {
		for (j = 0; j < 4; j++) {
			soin[j] += (feed+i)->afdrt * (feed+i)->xfd[j];
		}
	}

	for (i = 0; i < general->ntost; i++) {
		for (j = 0; j < 3; j++) {
			soin[j] += (feed+i)->ofdrt * (feed+i)->yfd[j];
		}
	}

	soin[2] += soin[3];

	return;
}


void transfer_aq ( General *general, int stage, double ain[], double deltax[], Feed *feed, Profile *profile, Cascade
*cascade )
{
	int i, v;

	v = 6;

	for (i = 0; i < v; i++) {
		deltax[i] = 0.0;

		ain[i] = feed->afdrt * feed->xfd[i];

		if (stage != 0) {
			ain[i] += (cascade-1)->anet * (profile-1)->xs[0][2][i];
			deltax[i] = (profile-1)->xs[1][2][i];
			deltax[i] -= (profile-1)->xs[0][2][i];
			deltax[i] *= (cascade-1)->anet;

			if (general->ifast != 2)
```

```
                    ain[i] += deltax[i] / 2.0;
            }

            if (fabs(ain[i]) < 1.0e-15 ) ain[i] = 0.0;
        }

        return;
}


void transfer_org ( General *general, int stage, double oin[], double deltay[], Feed *feed, Profile *profile, Cascade
*cascade )
{
        int i, in;

        in = general->istr - 1;

        for (i = 0; i < 3; i++) {
                deltay[i] = 0.0;

                oin[i] = (feed+stage)->ofdrt * (feed+stage)->yfd[i];

                if (stage != general->ntost-1) {
                        oin[i] += (cascade+stage+1)->onet * (profile+stage+1)->ys[0][2][i];
                        deltay[i] = (profile+stage+1)->ys[1][2][i];
                        deltay[i] -= (profile+stage+1)->ys[0][2][i];
                        deltay[i] *= (cascade+stage+1)->onet;

                        if (general->ifast != 2) {
                                oin[i] += deltay[i] / 2.0;
                        }
                }

                if (general->nstr == 1) {
                        if (stage == general->istr - 2) {
                                oin[i] = (feed+stage)->ofdrt * (feed+stage)->yfd[i];
                                deltay[i] = 0.0;
                        }

                        if (stage == general->jstr-1) {
                                oin[i] += (cascade+in)->onet * (profile+in)->ys[0][2][i];
                                deltay[i] += ((profile+in)->ys[1][2][i] - (profile+in)->ys[0][2][i]) * (cascade+in)->onet;
                        }

                        if (stage+1 == general->jstr && general->ifast != 2) {
                                oin[i] += ((profile+in)->ys[1][2][i] - (profile+in)->ys[0][2][i]) * (cascade+in)->onet / 2.0;
                        }
                }

                if (fabs(oin[i]) < 1.0e-15) oin[i] = 0.0;
        }

        return;
}


double heat_transfer_aq ( int i, Feed *feed, Profile *profile, Cascade *cascade )
```

```
{
        double ahtin;

        ahtin = (feed+i)->afdrt * (feed+i)->afdtem;

        if (i != 0) {
                ahtin += (cascade+i-1)->anet * (profile+i-1)->ats[0][2];
                ahtin += ((profile+i-1)->ats[1][2] - (profile+i-1)->ats[0][2]) * (cascade+i-1)->anet / 2.0;
        }

        return ahtin;
}


double heat_transfer_org ( int i, General *general, Feed *feed, Profile *profile, Cascade *cascade )
{
        intin;
        double ohtin;

        in = general->istr - 1;

        ohtin = (feed+i)->ofdrt * (feed+i)->ofdtem;

        if (i+1 != general->ntost) {
                ohtin += (cascade+i+1)->onet * (profile+i+1)->ots[0][2];
                ohtin += ((profile+i+1)->ots[1][2] - (profile+i+1)->ots[0][2]) * (cascade+i+1)->onet / 2.0;
        }

        if (general->nstr == 0) return ohtin;

        if (i+1 == in)
                ohtin = (feed+i)->ofdrt * (feed+i)->ofdtem;

        if (i+1 == general->jstr) {
                ohtin += (cascade+in)->onet * (profile+in)->ots[0][2];
                ohtin += ((profile+in)->ots[1][2] - (profile+in)->ots[0][2]) * (cascade+in)->onet / 2.0;
        }

        return ohtin;
}

double calc_activity (double h)
{
        double y;

        y = 8.8810e-4 * h - 1.383e-2;
        y *= h;
        y += 1.401e-1;
        y *= h;
        y -= 2.384e-1;
        y *= h;
        y += 0.8872;

        return y;
}

double calc_distribution (double h, double no3, double cmpo, double tbp)
```

```
{
        double d, y, hno3, tbp_f, cmpo_f, c2, c1;

        y = calc_activity(h);

        c2 = 2.0 * KT2 * h * no3 * y * y;
        c1 = KT1 * h * no3 * y * y + 1.0;
        if (c2 < 1.0e-10)
                tbp_f = 0.0;
        else
                tbp_f = (- c1 + sqrt(c1 * c1 + 4.0 * tbp * c2)) / 2.0 / c2;

        c1 = h * no3 * y * y;
        c2 = KC2 * c1 * c1 + KC1 * c1 + 1.0;
        if (c2 < 1.0e-10)
                cmpo_f = 0.0;
        else
                cmpo_f = cmpo / c2;

        hno3 = KT1 * tbp_f * h * no3 * y * y;
        hno3 += KT2 * tbp_f * tbp_f * h * no3 * y * y;
        hno3 += KC1 * cmpo_f * h * no3 * y * y;
        hno3 += 2.0 * KC2 * cmpo_f * h * h * no3 * no3 * y * y * y * y;

        if (hno3 < 1.0e-10)
                d = 0.0;
        else
                d = hno3 / h;

        return d;
}

void ucorpu ( General *general, double dtry[], double ary[], double tempc )
{
        double          tbp = 1.0;
        double          cmpo = 0.2;

        dtry[0] = calc_distribution(ary[0], ary[0] + ary[3], cmpo, tbp);
        dtry[1] = 0.0;
        dtry[2] = 0.0;

        return;
}


double molal_aq ( double aq[] )
{
        double conva;

        conva = 1.0 - (0.07240001 * aq[1] + 0.13 * aq[2] + 0.0309 * aq[0] + 0.031 * aq[5]);
        conva = pow(conva, -1.0);

        return conva;
}

double molal_org ( double org[], double ctbp, double t, double temp )
{
```

```
        double convo, wo, uos, puos, hos;

        uos = t / (2.0 + 0.092 * t);
        puos = t / (2.0 + 0.18 * t);
        hos = 1.0 - 0.00609 * (3.95 - 0.0144 * temp) * pow(ctbp, 1.65);
        hos *= t / (1.0 + 0.043 * t);
        wo = (3.95 - 0.0144 * temp) * pow(ctbp, 1.65);
        wo *= 1.0 - org[1] / uos - org[2] / puos - 0.65 * org[0] / hos;

        convo = 1.0 - (0.097 * org[1] + 0.139 * org[2] + 0.043 * org[0] + 0.0174 * wo);
        convo = pow(convo, -1.0);

        return convo;
}

double molar_aq ( double aq[] )
{
        double conva;

        conva = 1.0 + 0.07240001 * aq[1] + 0.13 * aq[2] + 0.0309 * aq[0] + 0.031 * aq[5];

        return conva;
}

double molar_org ( double org[], double ctbp, double t, double temp )
{
        double convo, wo;

        wo = (4.2 - 0.015 * temp) * pow(ctbp, 1.69);
        wo *= (t - 2.0 * org[1] - 2.0 * org[2] - 0.6 * org[0]) / t;

        convo = 1.0 + 0.097 * org[1] + 0.139 * org[2] + 0.043 * org[0] + 0.0174 * wo;

        return convo;
}
```

8.2.3 C_CHEM.C

```
#include <stdio.h>
#include <math.h>

#include "sephis.h"


void mchemp ( General *general, Profile *profile, Volume *mixer, double *ain, double *oin, double dtry[] )
{
        double stoic[6][3] = {      { 0.0, 2.0, 2.0 },
                                    { 0.0, 0.5, 0.0 },
                                    { -1.0, -1.0, -1.0 },
                                    { 1.0, 1.0, 1.0 },
                                    { -1.0, -0.5, -1.0 },
                                    { 0.0, 0.0, 0.0 } };
        double rx[6], solamt[6], solvol[6], ary[6];
        double w, ww, rxnamt, extmax, rk, rcu, rho, rexp, extent, for1, totnit, extinc, amtint, extent1, exdif;
        double a, aa, b, bb, c, cc;
        int i, kz, itry, ick, v;

        v = 6;

        for (i = 0; i < 6; i++)
                rx[i] = profile->x[1][i];

        for (i = 0; i < 3; i++) {
                solamt[i] = mixer->avol * profile->x[1][i] + mixer->ovol * profile->y[1][i];
                solvol[i] = mixer->avol + dtry[i] * mixer->ovol;
        }

        for (i = 3; i < 6; i++) {
                solamt[i] = mixer->avol * profile->x[1][i];
                solvol[i] = mixer->avol;
        }

        switch (general->irxn) {
                case 1:
                        goto instantaneous_reduction;
                        break;
                case 2:
                        goto uranium_iv;
                        break;
                case 3:
                        goto hydroxylamine_nitrate;
                        break;
        }

/*      即時還元      */

instantaneous_reduction:

        w = -solamt[2] / stoic[2][general->irxn-1];
        ww = -solamt[4] / stoic[4][general->irxn-1];

        if (w < ww) rxnamt = w; else rxnamt = ww;
```

```
        if (general->ifast == 2) {
                w = -(ain[2] + oin[2]) / stoic[2][general->irxn-1];
                ww = -ain[4] / stoic[4][general->irxn-1];

                if (w < ww)
                        rxnamt = w * general->dtheta;
                else
                        rxnamt = ww * general->dtheta;
        }

        goto end_reduction;

/*  U(IV)による還元          */

uranium_iv:

        if (rx[2] < 1e-10 || rx[4] < 1e-10)
                goto instantaneous_reduction;

        w = 1.0 - pow(0.5, 6.0 * general->dtheta);
        ww = 2.0 * solamt[4] / solamt[2];

        if (w <= ww)
                if (w <= 1.0) extmax = w; else extmax = 1.0;
        else
                if (ww <= 1.0) extmax = ww; else extmax = 1.0;

        rk = 170.0 * general->dtheta * rx[4];
        rcu = solamt[2] / (2.0 * solamt[4]);
        rho = rx[0] * rx[0];
        rexp = 1e+30;

        if (rk * (1.0 - rcu) < (70.0 * rho)) rexp = exp(rk * (1.0 - rcu) / rho);
        extent = rk / (rk + rho);

        if (fabs(1.0 - rcu) > 0.05) extent = (1.0 - rexp) / (rcu - rexp);
        if (extent > extmax) extent = extmax;

        rxnamt = solamt[2] * extent;

        goto end_reduction;

/*  HANによる還元   */

hydroxylamine_nitrate:

        if (rx[0] < 1e-10 || rx[2] < 1e-10 || rx[4] < 1e-10)
                goto instantaneous_reduction;

        for1 = solamt[2] / solamt[4];
        totnit = rx[0] + 2.0 * rx[1] + 4.0 * rx[2] + 3.0 * rx[3] + rx[5] + .33;

        rk = rx[4] / (rx[0] * rx[0] * totnit);
        rk *= rk;
        rk *= exp(31000.0 / 1.987 * (1.0 / 303.16 - 1.0 / (273.16 + profile->tprof[1])));
        rk *= 1.74 * rx[2] * general->dtheta;
```

```
        if ((1.0 / for1) < 1.0) extmax = 1.0 / for1; else extmax = 1.0;

        extinc = .25 * extmax;
        extent = .5 * extmax;
        b = solamt[2] / solvol[3];

        a = rx[3] + b;
        aa = a * a;
        bb = a * b;
        cc = b * b;

        for (i = 0; i < 25; i++) {
                a = 1.0 / (1.0 - extent);

                if (for1 <= 0.01)
                        amtint = aa * a * extent - 2.0 * bb * log(a) + cc * extent;
                else {
                        if (fabs(1 - for1) >= 0.01) {
                                b = 1 - for1 * extent;
                                c = 1 / (1 - for1);

                                amtint = -(for1 - a * extent - for1 / b + 2.0 * c * for1 * log(a * b)) * aa * c * c + 2.0 * bb * c *
(for1 * extent / b - c * log(a * b)) + cc * extent / b;
                        } else {
                                amtint = aa * (pow(a, 3.0) - 1.0) / 3.0 - bb * (a * a - 1.0) + cc * (a - 1.0);
                        }
                }

                if (amtint > rk) extent1 = extent - extinc;
                if (amtint < rk) extent1 = extent + extinc;

                extinc = extinc / 2;
                exdif = (extent1 - extent) / extent;
                extent = extent1;

                if (fabs(exdif) < 1.0e-5) break;
        }

        rxnamt = extent * solamt[2];

end_reduction:

        for (i = 0; i < 6; i++) {
                solamt[i] += rxnamt * stoic[i][general->irxn-1];
                if (solamt[i] < 0.0) solamt[i] = 0.0;

                ary[i] = solamt[i] / solvol[i];
                profile->rxntrm[i] = rxnamt * stoic[i][general->irxn-1] / general->dtheta;
        }

        if (general->ifast == 2) return;

        for (itry = 0; itry < 5; itry++) {
                ick = 0;
                ucorpu(general, dtry, ary, profile->tprof[1]);
```

```
            for (i = 0; i < 3; i++) {
                profile->x[1][i] = solamt[i] / (mixer->avol + dtry[i] * mixer->ovol);
                if (fabs(profile->x[1][i] - ary[i]) > (0.0001 * profile->x[1][i]))
                    ick = 1;

                ary[i] = profile->x[1][i];
                if (profile->x[1][i] <= 1e-20)
                    profile->x[1][i] = 0.0;

                profile->y[1][i] = dtry[i] * profile->x[1][i];
            }

            for (i = 3; i < v; i++) {
                profile->x[1][i] = solamt[i] / mixer->avol;
                if (fabs(profile->x[1][i] - ary[i]) > (0.0001 * profile->x[1][i]))
                    ick = 1;

                ary[i] = profile->x[1][i];
                if (profile->x[1][i] <= 1e-20)
                    profile->x[1][i] = 0.0;
            }

            if (ick == 0) break;
        }

        return;
    }


    void schemp ( General *general, Profile *profile )
    {
        double stoicx[6][3] = {  { 0.0, 2.0, 2.0 },
                                 { 0.0, 0.5, 0.0 },
                                 { -1.0, -1.0, -1.0 },
                                 { 1.0, 1.0, 1.0 },
                                 { -1.0, -0.5, -1.0 },
                                 { 0.0, 0.0, 0.0 } };
        double rx[6], ry[3];
        double w, ww, delx, dely, extmax, rk, rcu, rho, rexp, extent, for1, totnit, extinc, amtint, extent1, exdif;
        double a, aa, b, bb, c, cc;
        int i, kz;

        for (kz = 0; kz < 3; kz++) {
            for (i = 0; i < 6; i++)
                rx[i] = profile->xs[1][kz][i];

            for (i = 0; i < 3; i++)
                ry[i] = profile->ys[1][kz][i];

            switch (general->irxn) {
                case 1:
                    goto instantaneous_reduction_settler;
                    break;
                case 2:
                    goto uranium_iv_settler;
                    break;
                case 3:
```

```
                    goto hydroxylamine_nitrate_settler;
                    break;
            }

/*   即時還元   */

instantaneous_reduction_settler:

            w = -rx[2] / stoicx[2][general->irxn-1];
            ww = -rx[4] / stoicx[4][general->irxn-1];

            if (w < ww) delx = w; else delx = ww;

            dely = 0.0;

            goto end_reduction_settler;

/*   U(IV)による還元          */

uranium_iv_settler:

            if (rx[2] < 1e-10 || rx[4] < 1e-10)
                  goto instantaneous_reduction_settler;

            w = 1.0 - (0.5, 6.0 * general->dtheta);
            ww = 2.0 * rx[4] / rx[2];

            if (w <= ww)
                  if (w <= 1.0) extmax = w; else extmax = 1.0;
            else
                  if (ww <= 1.0) extmax = ww; else extmax = 1.0;

            rk = 170.0 * general->dtheta * rx[4];
            rcu = rx[2] / (2.0 * rx[4]);
            rho = rx[0] * rx[0];
            rexp = 1e+30;

            if (rk * (1.0 - rcu) < (70.0 * rho))
                  rexp = exp(rk * (1.0 - rcu) / rho);

            extent = rk / (rk + rho);

            if (fabs(1.0 - rcu) > .05) extent = (1.0 - rexp) / (rcu - rexp);
            if (extent > extmax) extent = extmax;

            delx = rx[2] * extent;
            dely = 0.0;

            goto end_reduction_settler;

/*   HANによる還元   */

hydroxylamine_nitrate_settler:

            if (rx[0] < 1e-10 || rx[2] < 1e-10 || rx[4] < 1e-10)
                  goto instantaneous_reduction_settler;
```

```
        for1 = rx[2] / rx[4];
        totnit = rx[0] + 2.0 * rx[1] + 4.0 * rx[2] + 3.0 * rx[3] + rx[5] + .33;

        rk = rx[4] / (rx[0] * rx[0] * totnit);
        rk *= rk;
        rk *= exp(31000.0 / 1.987 * (1.0 / 303.16 - 1.0 / (273.16 + profile->tprof[1])));
        rk *= 1.74 * rx[2] * general->dtheta;

        if ((1.0 / for1) < 1.0) extmax = 1.0 / for1; else extmax = 1.0;

        extinc = .25 * extmax;
        extent = .5 * extmax;
        b = rx[2];
        a = rx[3] + b;
        aa = a * a;
        bb = a * b;
        cc = b * b;

        for (i = 0; i < 25; i++) {
            a = 1.0 / (1.0 - extent);

            if (for1 <= 0.01)
                amtint = aa * a * extent - 2.0 * bb * log(a) + cc * extent;
            else {
                if (fabs(1.0 - for1) >= 0.01) {
                    b = 1.0 - for1 * extent;
                    c = 1.0 / (1.0 - for1);

                    amtint = -(for1 - a * extent - for1 / b + 2.0 * c * for1 * log(a * b)) * aa * c * c + 2.0 * bb
* c * (for1 * extent / b - c * log(a * b)) + cc * extent / b;
                } else {
                    amtint = aa * (pow(a, 3.0) - 1.0) / 3.0 - bb * (a * a - 1.0) + cc * (a - 1.0);
                }
            }

            if (amtint > rk) extent1 = extent - extinc;
            if (amtint < rk) extent1 = extent + extinc;

            extinc = extinc / 2.0;
            exdif = (extent1 - extent) / extent;
            extent = extent1;

            if (fabs(exdif) < 1.0e-5) break;
        }

        delx = extent * rx[2];
        dely = 0.0;

end_reduction_settler:

        for (i = 0; i < 6; i++) {
            profile->xs[1][kz][i] += stoicx[i][general->irxn-1] * delx;
            if (profile->xs[1][kz][i] <= 1e-20) profile->xs[1][kz][i] = 0.0;
        }

        for (i = 0; i < 3; i++)
            if (profile->ys[1][kz][i] <= 1e-20) profile->ys[1][kz][i] = 0.0;
```

```
        }

        return;
    }
```

8.2.4 C_STAGE.C

```
#include <stdio.h>
#include <conio.h>
#include <math.h>

#include "sephis.h"

#define CCD 0.001


/*          ミキサのRunge-Kutta法による計算      */

void mixer_runge_kutta ( General *general, Profile *profile, Cascade *cascade, Volume *mixer, double ain[], double
oin[], double dtry[] )
{
        int mdex, mck, v, i;
        double ary[6], tx[6], f[6], phi[6], dif;

        mck = 0;
        v = 6;

        for (i = 0; i < v; i++) {
                tx[i] = ary[i] = profile->x[1][i];
        }

        ucorpu(general, dtry, ary, profile->tprof[1]);

runge_kutta_1:

        for (i = 0; i < 3; i++) {
                f[i] = ain[i] + oin[i];
                f[i] -= (cascade->at + cascade->ot * dtry[i]) * profile->x[0][i];
                f[i] /= mixer->avol + mixer->ovol * dtry[i];
                ary[i] = profile->x[0][i] + general->dtheta * f[i] / 2.0;
        }

        for (i = 3; i < v; i++) {
                f[i] = (ain[i] - cascade->at * profile->x[0][i]) / mixer->avol;
                ary[i] = profile->x[0][i] + general->dtheta * f[i] / 2.0;
        }

        mck++;

        ucorpu(general, dtry, ary, profile->tprof[1]);

        mdex = 0;

        for (i = 0; i < v; i++) {
                if (fabs(ary[i]) >= 1.0e-6 ) {
                        dif = fabs((tx[i] - ary[i]) / ary[i]);
                        if (dif > CCD) mdex = 1;
                }

                tx[i] = ary[i];
        }
```

```
    if (mdex == 1 && mck < 5) goto runge_kutta_1;

    for (i = 0; i < v; i++) {
        profile->x[1][i] = ary[i];
        phi[i] = f[i];
    }

runge_kutta_2:

    for (i = 0; i < 3; i++) {
        f[i] = ain[i] + oin[i];
        f[i] -= (cascade->at + cascade->ot * dtry[i]) * profile->x[1][i];
        f[i] /= mixer->avol + mixer->ovol * dtry[i];
        ary[i] = profile->x[0][i] + general->dtheta * f[i] / 2.0;
    }

    for (i = 3; i < v; i++) {
        f[i] = (ain[i] - cascade->at * profile->x[1][i]) / mixer->avol;
        ary[i] = profile->x[0][i] + general->dtheta * f[i] / 2.0;
    }

    mck++;

    ucorpu(general, dtry, ary, profile->tprof[1]);

    mdex = 0;

    for (i = 0; i < v; i++) {
        if (fabs(ary[i]) >= 1.0e-6 ) {
            dif = fabs((tx[i] - ary[i]) / ary[i]);
            if (dif > CCD) mdex = 1;
        }

        tx[i] = ary[i];
    }

    if (mdex == 1 && mck < 10) goto runge_kutta_2;

    for (i = 0; i < v; i++) {
        profile->x[1][i] = ary[i];
        phi[i] += 2.0 * f[i];
    }

runge_kutta_3:

    for (i = 0; i < 3; i++) {
        f[i] = ain[i] + oin[i];
        f[i] -= (cascade->at + cascade->ot * dtry[i]) * profile->x[1][i];
        f[i] /= mixer->avol + mixer->ovol * dtry[i];
        ary[i] = profile->x[0][i] + general->dtheta * f[i];
    }

    for (i = 3; i < v; i++) {
        f[i] = (ain[i] - cascade->at * profile->x[1][i]) / mixer->avol;
        ary[i] = profile->x[0][i] + general->dtheta * f[i];
    }
```

```
        mck++;

        ucorpu(general, dtry, ary, profile->tprof[1]);

        mdex = 0;

        for (i = 0; i < v; i++) {
                if (fabs(ary[i]) >= 1.0e-6 ) {
                        dif = fabs((tx[i] - ary[i]) / ary[i]);
                        if (dif > CCD) mdex = 1;
                }

                tx[i] = ary[i];
        }

        if (mdex == 1 && mck < 15) goto runge_kutta_3;

        for (i = 0; i < v; i++) {
                profile->x[1][i] = ary[i];
                phi[i] += 2.0 * f[i];
        }

runge_kutta_4:

        for (i = 0; i < 3; i++) {
                f[i] = ain[i] + oin[i];
                f[i] -= (cascade->at + cascade->ot * dtry[i]) * profile->x[1][i];
                f[i] /= mixer->avol + mixer->ovol * dtry[i];
                ary[i] = profile->x[0][i] + (phi[i] + f[i]) * general->dtheta / 6.0;
        }

        for (i = 3; i < v; i++) {
                f[i] = (ain[i] - cascade->at * profile->x[1][i]) / mixer->avol;
                ary[i] = profile->x[0][i] + (phi[i] + f[i]) * general->dtheta / 6.0;
        }

        mck++;

        ucorpu(general, dtry, ary, profile->tprof[1]);

        mdex = 0;

        for (i = 0; i < v; i++) {
                if (fabs(ary[i]) >= 1.0e-6 ) {
                        dif = fabs((tx[i] - ary[i]) / ary[i]);
                        if (dif > CCD) mdex = 1;
                }

                tx[i] = ary[i];
        }

        if (mdex == 1 && mck < 20) goto runge_kutta_4;

        for (i = 0; i < 3; i++) {
                if (ary[i] <= 1.0e-20 ) ary[i] = 0.0;
                profile->x[1][i] = ary[i];
```

```
            profile->y[1][i] = dtry[i] * ary[i];
        }

        for (i = 3; i < v; i++) {
            if (ary[i] <= 1.0e-20 ) ary[i] = 0.0;
            profile->x[1][i] = ary[i];
        }

        return;
}


/*    ミキサのTrapezoidal法による計算  */

void mixer_trapezoidal ( General *general, Cascade *cascade, Profile *profile, Volume *mixer, double ain[], double
oin[], double dtry[] )
{
        int mck, mdex, i, v;
        double dvols, dflos, ary[6], dif;

        v = 6;

        for (i = 0; i < v; i++) {
            ary[i] = profile->x[1][i];
        }

        mck = 0;

trapezoidal_1:

        mdex = 0;

        ucorpu(general, dtry, ary, profile->tprof[1]);

        for (i = 0; i < 3; i++) {
            dvols = mixer->avol + dtry[i] * mixer->ovol;
            dflos = cascade->at + dtry[i] * cascade->ot;

            profile->x[1][i] = (mixer->avol * profile->x[0][i] + mixer->ovol * profile->y[0][i] + general->dtheta *
(ain[i] + oin[i] - (cascade->at * profile->x[0][i] + cascade->ot * profile->y[0][i]) / 2.0)) / (dvols + general->dtheta * dflos
/ 2.0);

            if (profile->x[1][i] <= 1.0e-20) profile->x[1][i] = 0.0;

            profile->y[1][i] = profile->x[1][i] * dtry[i];

            if (fabs(profile->x[1][i]) >= 1.0e-09) {
                dif = fabs((profile->x[1][i] - ary[i]) / profile->x[1][i]);
                if (dif >= CCD) mdex = 1;
            }

            ary[i] = profile->x[1][i];
        }

        for (i = 3; i < v; i++) {
            dvols = mixer->avol;
            dflos = cascade->at;
```

```
            profile->x[1][i] = (mixer->avol * profile->x[0][i] + general->dtheta * (ain[i] - (cascade->at *
profile->x[0][i]) / 2.0)) / (dvols + general->dtheta * dflos / 2.0);

            if (profile->x[1][i] <= 1.0e-20) profile->x[1][i] = 0.0;

            if (fabs(profile->x[1][i]) >= 1.0e-09) {
                dif = fabs((profile->x[1][i] - ary[i]) / profile->x[1][i]);
                if (dif >= CCD) mdex = 1;
            }

            ary[i] = profile->x[1][i];
        }

        mck++;

        if (mdex == 1 && mck <= 20) goto trapezoidal_1;

        return;
}


void mixer_fast ( General *general, Profile *profile, Cascade *cascade, Volume *mixer, double *deltax, double *deltay,
double *ain, double *oin, double dtry[] )
{
        int mck, mdex, i, v;
        double dif, wght, ary[6];

        v = 6;

        for (i = 0; i < v; i++) {
            ary[i] = profile->x[1][i];
        }

        mck = 0;

fast_1:

        mdex = 0;

        ucorpu(general, dtry, ary, profile->tprof[1]);

        for (i = 0; i < 3; i++) {
            wght = general->dtheta * (cascade->at + dtry[i] * cascade->ot) / (mixer->avol + dtry[i] * mixer->ovol);
            ary[i] = profile->x[0][i] * exp(-wght) + (ain[i] + oin[i] + profile->rxntrm[i]) * (1.0 - exp(-wght)) /
(cascade->at + dtry[i] * cascade->ot) + (deltax[i] + deltay[i]) * (exp(-wght) + wght - 1.0) / (wght * (cascade->at + dtry[i] *
cascade->ot));

            dif = 0.0;

            if (fabs(ary[i]) >= 1.0e-6)
                dif = fabs((profile->x[1][i] - ary[i]) / ary[i]);

            if (dif > CCD) mdex = 1;

            if (ary[i] < 1.0e-10) ary[i] = 0.0;
```

```
            profile->x[1][i] = ary[i];
            profile->y[1][i] = ary[i] * dtry[i];
        }

        for (i = 3; i < v; i++) {
            wght = general->dtheta * cascade->at / mixer->avol;
            ary[i] = profile->x[0][i] * exp(-wght) + (ain[i] + profile->rxntrm[i]) * (1.0 - exp(-wght)) / cascade->at +
deltax[i] * (exp(-wght) + wght - 1.0) / (wght * cascade->at);

            dif = 0.0;

            if (fabs(ary[i]) >= 1.0e-6)
                dif = fabs((profile->x[1][i] - ary[i]) / ary[i]);

            if (dif > CCD) mdex = 1;

            if (ary[i] < 1.0e-10) ary[i] = 0.0;

            profile->x[1][i] = ary[i];
        }

        mck++;

        if (mdex == 1 && mck < 15) goto fast_1;

        return;
}


/*          セトラの計算（Trapezoidal法による）        */

void settler_trapezoidal ( General *general, Profile *profile, Cascade *cascade, Volume *settler )
{
        int i, kz, kzb, v;
        double facta, facto;

        v = 6;

        facta = facto = 150.0;

        if (settler->avol > 0.0)
            facta = 3.0 * cascade->at * general->dtheta / settler->avol;
        if (settler->ovol > 0.0)
            facto = 3.0 * cascade->ot * general->dtheta / settler->ovol;

        profile->ats[1][0] = profile->ats[0][0] * exp(-facta) + profile->tprof[0] * (1.0 - exp(-facta)) + (profile->tprof[1] -
profile->tprof[0]) * (exp(-facta) + facta - 1.0) / facta;
        profile->ots[1][0] = profile->ots[0][0] * exp(-facto) + profile->tprof[0] * (1.0 - exp(-facto)) + (profile->tprof[1] -
profile->tprof[0]) * (exp(-facto) + facto - 1.0) / facto;

        for (i = 0; i < 3; i++) {
            profile->xs[1][0][i] = profile->xs[0][0][i] * exp(-facta);
            profile->xs[1][0][i] += profile->x[0][i] * (1.0 - exp(-facta));
            profile->xs[1][0][i] += (profile->x[1][i] - profile->x[0][i]) * (exp(-facta) + facta - 1.0) / facta;
            if (profile->xs[1][0][i] <= 1e-20) profile->xs[1][0][i] = 0.0;

            profile->ys[1][0][i] = profile->ys[0][0][i] * exp(-facto);
```

```
            profile->ys[1][0][i] += profile->y[0][i] * (1.0 - exp(-facto));
            profile->ys[1][0][i] += (profile->y[1][i] - profile->y[0][i]) * (exp(-facto) + facto - 1.0) / facto;
            if (profile->ys[1][0][i] <= 1e-20) profile->ys[1][0][i] = 0.0;
        }

        for (i = 3; i < v; i++) {
            profile->xs[1][0][i] = profile->xs[0][0][i] * exp(-facta);
            profile->xs[1][0][i] += profile->x[0][i] * (1.0 - exp(-facta));
            profile->xs[1][0][i] += (profile->x[1][i] - profile->x[0][i]) * (exp(-facta) + facta - 1.0) / facta;
            if (profile->xs[1][0][i] <= 1e-20) profile->xs[1][0][i] = 0.0;
        }

        for (kz = 1; kz < 3; kz++) {
            kzb = kz - 1;

            profile->ats[1][kz] = profile->ats[0][kz] * exp(-facta);
            profile->ats[1][kz] += profile->ats[0][kzb] * (1.0 - exp(-facta));
            profile->ats[1][kz] += (profile->ats[1][kzb] - profile->ats[0][kzb]) * (exp(-facta) + facta - 1.0) / facta;
            profile->ots[1][kz] = profile->ots[0][kz] * exp(-facto);
            profile->ots[1][kz] += profile->ots[0][kzb] * (1.0 - exp(-facto));
            profile->ots[1][kz] += (profile->ots[1][kzb] - profile->ots[0][kzb]) * (exp(-facto) + facto - 1.0) / facto;

            for (i = 0; i < 3; i++) {
                profile->xs[1][kz][i] = profile->xs[0][kz][i] * exp(-facta);
                profile->xs[1][kz][i] += profile->xs[0][kzb][i] * (1.0 - exp(-facta));
                profile->xs[1][kz][i] += (profile->xs[1][kzb][i] - profile->xs[0][kzb][i]) * (exp(-facta) + facta - 1.0) /
facta;
                if (profile->xs[1][kz][i] <= 1e-20) profile->xs[1][kz][i] = 0.0;

                profile->ys[1][kz][i] = profile->ys[0][kz][i] * exp(-facto);
                profile->ys[1][kz][i] += profile->ys[0][kzb][i] * (1.0 - exp(-facto));
                profile->ys[1][kz][i] += (profile->ys[1][kzb][i] - profile->ys[0][kzb][i]) * (exp(-facto) + facto - 1.0) /
facto;
                if (profile->ys[1][kz][i] <= 1e-20) profile->ys[1][kz][i] = 0.0;
            }

            for (i = 3; i < v; i++) {
                profile->xs[1][kz][i] = profile->xs[0][kz][i] * exp(-facta);
                profile->xs[1][kz][i] += profile->xs[0][kzb][i] * (1.0 - exp(-facta));
                profile->xs[1][kz][i] += (profile->xs[1][kzb][i] - profile->xs[0][kzb][i]) * (exp(-facta) + facta - 1.0) /
facta;
                if (profile->xs[1][kz][i] <= 1e-20) profile->xs[1][kz][i] = 0.0;
            }
        }

return;
}


void calc_inventory_change ( General *general, Profile *profile, Volume *v_mixer, Change *change )
{
        int i, v;
        double cnginv;

        v = 6;

        for (i = 0; i < 3; i++) {
```

```
            cnginv = 0.0;

            if ((profile->x[1][i] + profile->y[1][i]) > 1.0e-6) {
                cnginv = v_mixer->avol * (profile->x[1][i] - profile->x[0][i]);
                cnginv += v_mixer->ovol * (profile->y[1][i] - profile->y[0][i]);
                cnginv /= general->dtheta * (v_mixer->avol * profile->x[1][i] + v_mixer->ovol * profile->y[1][i]);
            }

            cnginv = 100.0 * fabs(cnginv);

            if (change->codum < cnginv) change->codum = cnginv;
            if (change->codumm[i] < cnginv) change->codumm[i] = cnginv;
        }

        for (i = 3; i < v; i++) {
            cnginv = 0.0;

            if (profile->x[1][i] > 1.0e-6) {
                cnginv = v_mixer->avol * (profile->x[1][i] - profile->x[0][i]);
                cnginv /= general->dtheta * v_mixer->avol * profile->x[1][i];
            }

            cnginv = 100.0 * fabs(cnginv);

            if (change->codum < cnginv) change->codum = cnginv;
            if (change->codumm[i] < cnginv) change->codumm[i] = cnginv;
        }

return;
}


int stages ( int ndirec, int isol5, General *general, Feed *feed, Profile *profile, Cascade *cascade, Change *change,
Volume *v_mixer, Volume *v_settler )
{
        int i, j, v, nscan, mscan, imchem;
        double dtry[6], ain[6], oin[3], deltax[6], deltay[6], sph, temp, t;
        double ahtin, ohtin, conva, convo;
        double aq[6], org[3];

        Volume mixer, settler;

        sph = 0.321 + 0.078 * general->ctbp;
        t = 3.65145 * general->ctbp;

        v = 6;
        mscan = 0;

stages_loop:

        for (nscan = 0; nscan < general->ntost; nscan++) {
            j = nscan;
            if (ndirec < 0) j = general->ntost - nscan - 1;

            display_stage_number(j+1);

            (change+j)->codum = 0.0;
```

```
transfer_aq(general, j, ain, deltax, feed+j, profile+j, cascade+j);
transfer_org(general, j, oin, deltay, feed, profile, cascade);

for (i = 0; i < v; i++) {
        (change+j)->codumm[i] = 0.0;
        aq[i] = (profile+j)->x[1][i];
}

for (i = 0; i < 3; i++) {
        org[i] = (profile+j)->y[1][i];
}

temp = (profile+j)->tprof[1];

conva = molal_aq(aq);
convo = molal_org(org, general->ctbp, t, temp);

mixer.avol = (v_mixer+j)->avol / conva;
mixer.ovol = (v_mixer+j)->ovol / convo;
settler.avol = (v_settler+j)->avol / conva;
settler.ovol = (v_settler+j)->ovol / convo;

ahtin = heat_transfer_aq(j, feed, profile, cascade);
ohtin = heat_transfer_org(j, general, feed, profile, cascade);

(profile+j)->tprof[1] = (cascade+j)->at + sph * (cascade+j)->ot;
(profile+j)->tprof[1] *= -(profile+j)->tprof[0];
(profile+j)->tprof[1] += ahtin + sph * ohtin;
(profile+j)->tprof[1] *= general->dtheta / (mixer.avol + sph * mixer.ovol);
(profile+j)->tprof[1] += (profile+j)->tprof[0];

imchem = 0;

switch (general->ifast) {
        case 0:
                mixer_runge_kutta(general, profile+j, cascade+j, &mixer, ain, oin, dtry);
                if (general->irxn > 0 && isol5 == 1)
                        mchemp(general, profile+j, &mixer, ain, oin, dtry);
                break;
        case 1:
                mixer_trapezoidal(general, cascade+j, profile+j, &mixer, ain, oin, dtry);
                if (general->irxn > 0 && isol5 == 1)
                        mchemp(general, profile+j, &mixer, ain, oin, dtry);
                break;
        case 2:
                while (1) {
                        mixer_fast(general, profile+j, cascade+j, &mixer, deltax, deltay, ain, oin, dtry);
                        if (imchem == 1 || general->irxn == 0) break;
                        imchem++;
                        if (isol5 == 1)
                          mchemp(general, profile+j, &mixer, ain, oin, dtry);
                }
                break;
}

settler_trapezoidal(general, profile+j, cascade+j, &settler);
```

```
        calc_inventory_change(general, profile+j, v_mixer+j, change+j);

        if (general->irxn > 0 && isol5 == 1)
            schemp(general, profile+j);
    }

    ndirec *= -1;
    mscan++;

    if (general->ifast == 2 && mscan < 2) goto stages_loop;
    if (general->ifast < 2) ndirec *= -1;

    return ndirec;
}
```

8.2.5 C_INIT.C

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdio.h>
#include <math.h>

#include "sephis.h"

void set_general ( FILE *fp, General *general )
{
	fscanf(fp, "%s¥n%s¥n", general->s_time, general->s_date);
	fgets(general->title, STRING, fp);
	fscanf(fp, "%d,%lf,%lf,", &general->ntost, &general->ctbp, &general->tempi);
	fscanf(fp, "%d,%d,%d¥n", &general->nstr, &general->istr, &general->jstr);
	fscanf(fp, "%d,%d,%lf,", &general->irxn, &general->iproce, &general->dtheta);
	fscanf(fp, "%lf,%lf,%lf¥n", &general->dprint, &general->tstop, &general->tol);
	fscanf(fp, "%d,%d,%d,", &general->newout, &general->ivolm, &general->ivols);
	fscanf(fp, "%d,%d¥n", &general->ipro, &general->ifast);

	return;
}


int set_feed ( FILE *fp, Feed *feed, General *general )
{
	int nst, jhas, index, isol5;
	double fdrt, con[6], temp;
	int v, i;

	isol5 = 0;

	for (i = 0; i < general->ntost; i++)
		(feed+i)->afdtem = (feed+i)->ofdtem = general->tempi;

	while (1) {
		fscanf(fp, "%d,%d,%lf¥n", &nst, &jhas, &fdrt);
		nst--;

		if (jhas == 0)
			v = 3;
		else
			v = 6;

		for (i = 0; i < v; i++)
			fscanf(fp, "%lf,", con+i);

		fscanf(fp, "%lf,%d¥n", &temp, &index);

		if (jhas == 0) {
			(feed+nst)->ofdrt = fdrt;
			if (fabs(temp) > 0.001) (feed+nst)->ofdtem = temp;

			for (i = 0; i < 3; i++)
				(feed+nst)->yfd[i] = con[i];
```

```
            } else {
                (feed+nst)->afdrt = fdrt;
                if (fabs(temp) > 0.001) (feed+nst)->afdtem = temp;

                for (i = 0; i < v; i++)
                    (feed+nst)->xfd[i] = con[i];
            }

            if (con[4] > 1.0e-6) isol5 = 1;

            if (index == 0) break;
        }

        return isol5;
}

int set_product ( FILE *fp, Product *product )
{
        int nst, jhas, index;
        double otrt;

        while (1) {
            fscanf(fp, "%d,%d,%lf,%d¥n", &nst, &jhas, &otrt, &index);
            nst--;

            if (jhas == 0)
                (product+nst)->olvrt = otrt;
            else
                (product+nst)->alvrt = otrt;

            if (index == 0) break;
        }

        return 0;
}

int set_profile ( FILE *fp, Profile *profile, General *general )
{
        int i, j, k, v, isol5;

        isol5 = 0;
        v = 6;

        for (i = 0; i < general->ntost; i++) {
            (profile+i)->tprof[0] = (profile+i)->tprof[1] = general->tempi;

            for (j = 0; j < 3; j++) {
                (profile+i)->ats[0][j] = (profile+i)->ats[1][j] = general->tempi;
                (profile+i)->ots[0][j] = (profile+i)->ots[1][j] = general->tempi;
            }
        }

        if (general->ipro == 0) return isol5;

        for (i = 0; i < general->ntost; i++) {
            for (j = 0; j < v; j++)
                fscanf(fp, "%lf,", &((profile+i)->x[1][j]));
```

```
            fscanf(fp, "%lf¥n", &((profile+i)->tprof[1]));

            if (fabs((profile+i)->tprof[1]) < 0.001)
                  (profile+i)->tprof[0] = (profile+i)->tprof[1] = general->tempi;

            for (j = 0; j < 6; j++)
                  (profile+i)->rxntrm[j] = 0.0;

            if (fabs((profile+i)->x[1][4]) > 1.0e-6) isol5 = 1;

            for (j = 0; j < 3; j++)    {
                  for (k = 0; k < 2; k++) {
                        (profile+i)->ats[k][j] = (profile+i)->tprof[0];
                        (profile+i)->ots[k][j] = (profile+i)->tprof[0];
                  }
            }
      }

      for (i = 0; i < general->ntost; i++) {
            for (j = 0; j < 3; j++)
                  fscanf(fp, "%lf,", &((profile+i)->y[1][j]));
      }

      return isol5;
}

void conv_feed ( Feed *feed, General *general )
{
      int v, i, j;
      double aq[6], org[3], temp, t, conva, convo;

      t = 3.65145 * general->ctbp;
      v = 6;

      for (i = 0; i < general->ntost; i++) {
            if (fabs((feed+i)->afdrt+(feed+i)->ofdrt) >= 1.0e-6) {
                  if (fabs((feed+i)->afdrt) >= 1.0e-6)
                        temp = (feed+i)->afdtem;

                  if (fabs((feed+i)->ofdrt) >= 1.0e-6)
                        temp = (feed+i)->ofdtem;

                  (feed+i)->xfd[1] /= 238.0;
                  (feed+i)->xfd[2] /= 239.0;
                  (feed+i)->xfd[3] /= 239.0;
                  (feed+i)->yfd[1] /= 238.0;
                  (feed+i)->yfd[2] /= 239.0;

                  for (j = 0; j < 6; j++)
                        aq[j] = (feed+i)->xfd[j];

                  for (j = 0; j < 3; j++)
                        org[j] = (feed+i)->yfd[j];

                  conva = molal_aq(aq);
                  convo = molal_org(org, general->ctbp, t, temp);
```

```
                (feed+i)->afdrt /= conva;
                (feed+i)->ofdrt /= convo;

                for (j = 0; j < v; j++)
                        (feed+i)->xfd[j] *= conva;

                for (j = 0; j < 3; j++)
                        (feed+i)->yfd[j] *= convo;
            }
        }

        return;
}


void set_cascade ( Cascade *cascade, General *general, Product *product, Feed *feed )
{
        int i;

        if (product->alvrt > feed->afdrt) product->alvrt = feed->afdrt;
        cascade->anet = feed->afdrt - product->alvrt;
        cascade->at = feed->afdrt;

        for (i = 1; i < general->ntost; i++) {
                if ((product+i)->alvrt >= ((cascade+i-1)->anet + (feed+i)->afdrt))
                        (product+i)->alvrt = (cascade+i-1)->anet + (feed+i)->afdrt;

                (cascade+i)->anet = (cascade+i-1)->anet + (feed+i)->afdrt - (product+i)->alvrt;
                (cascade+i)->at = (cascade+i)->anet + (product+i)->alvrt;
        }

        if ((product+general->ntost-1)->olvrt > (feed+general->ntost-1)->ofdrt)
                (product+general->ntost-1)->olvrt = (feed+general->ntost-1)->ofdrt;

        (cascade+general->ntost-1)->onet = (feed+general->ntost-1)->ofdrt - (product+general->ntost-1)->olvrt;
        (cascade+general->ntost-1)->ot = (feed+general->ntost-1)->ofdrt;

        for (i = general->ntost-2; i >= 0; i--) {
                if ((product+i)->olvrt >= ((cascade+i+1)->onet + (feed+i)->ofdrt))
                        (product+i)->olvrt = (cascade+i+1)->onet + (feed+i)->ofdrt;

                (cascade+i)->onet = (cascade+i+1)->onet + (feed+i)->ofdrt - (product+i)->olvrt;

                if (general->nstr == 1 && i == general->istr-2)
                        (cascade+i)->onet = (feed+i)->ofdrt - (product+i)->olvrt;

                if (general->nstr == 1 && i == general->jstr-1){
                        (cascade+i)->onet = (cascade+general->istr-1)->onet + (cascade+i+1)->onet;
                        (cascade+i)->onet += (feed+i)->ofdrt - (product+i)->olvrt;
                }

                (cascade+i)->ot = (cascade+i)->onet + (product+i)->olvrt;
        }

        return;
}
```

```
void conv_profile_aq ( Profile *profile, General *general )
{
	int v, i, j, k, l;
	double aq[6], conva;

	v = 6;

	for (i = 0; i < general->ntost; i++) {
		(profile+i)->x[1][1] /= 238.0;
		(profile+i)->x[1][2] /= 239.0;
		(profile+i)->x[1][3] /= 239.0;

		for (j = 0; j < v; j++)
			aq[j] = (profile+i)->x[1][j];

		conva = molal_aq(aq);

		for (j = 0; j < v; j++) {
			aq[j] *= conva;

			for (k = 0; k < 2; k++) {
				(profile+i)->x[k][j] = aq[j];

				for (l = 0; l < 3; l++)
					(profile+i)->xs[k][l][j] = aq[j];
			}
		}
	}

	return;
}


void conv_profile_org ( Profile *profile, General *general )
{
	int i, j, k, l;
	double org[3], convo, t, temp;

	t = 3.65145 * general->ctbp;

	for (i = 0; i < general->ntost; i++) {
		(profile+i)->y[1][1] /= 238.0;
		(profile+i)->y[1][2] /= 239.0;

		for (j = 0; j < 3; j++)
			org[j] = (profile+i)->y[1][j];

		temp = (profile+i)->tprof[1];

		convo = molal_org(org, general->ctbp, t, temp);

		for (j = 0; j < 3; j++) {
			org[j] *= convo;

			for (k = 0; k < 2; k++) {
```

```
                        (profile+i)->y[k][j] = org[j];

                        for (l = 0; l < 3; l++)
                                (profile+i)->ys[k][l][j] = org[j];
                }
            }
        }

        return;
}


void set_volume ( FILE *fp, int ivol, int ntost, Volume *volume, Cascade *cascade )
{
        int nst, index, ns, idex, px, i;
        double vola, volo, vla, vlo;

        if (ivol != 3) {
                px = 0;
                fscanf(fp, "%d,%lf,%lf,%d¥n", &nst, &vola, &volo, &index);

label_v1:
                ns = nst;
                vla = vola;
                vlo = volo;
                idex = index;

label_v2:
                (volume+ns-1)->avol = vla;
                (volume+ns-1)->ovol = vlo;

                if (px == 1) goto label_v3;

                if (idex == 1)
                        fscanf(fp, "%d,%lf,%lf,%d¥n", &nst, &vola, &volo, &index);
                else
                        goto label_v4;

label_v3:
                if (nst == ns + 1) {
                        px = 0;
                        goto label_v1;
                } else {
                        ns++;
                        px = 1;
                        goto label_v2;
                }

label_v4:
                if (ns != ntost) {
                        ns++;
                        goto label_v2;
                }
        }

        for (i = 0; i < ntost; i++) {
                switch (ivol) {
```

```
                case 2:
                        (volume+i)->ovol = (volume+i)->avol * (cascade+i)->ot;
                        (volume+i)->ovol /= (cascade+i)->at + (cascade+i)->ot;
                        (volume+i)->avol -= (volume+i)->ovol;
                        break;
                case 3:
                        (volume+i)->avol = (cascade+i)->at;
                        (volume+i)->ovol = (cascade+i)->ot;
                        break;
        }

        if ((volume+i)->avol < 0.0001) (volume+i)->avol = 0.0;
        if ((volume+i)->ovol < 0.0001) (volume+i)->ovol = 0.0;
    }

    return;
}

void initial_equilibrium ( General *general, Profile *profile, Volume *v_mixer, Volume *v_settler )
{
    int i, j, k, l, ick, v;
    double dtry[6], av, ov;

    v = 6;

    for (i = 0; i < general->ntost; i++) {
        av = (v_mixer+i)->avol + (v_settler+i)->avol;
        ov = (v_mixer+i)->ovol + (v_settler+i)->ovol;

        for (j = 0; j < 3; j++) {
            ick = 0;

            ucorpu(general, dtry, (profile+i)->x[1], (profile+i)->tprof[1]);

            for (k = 0; k < 3; k++) {
                (profile+i)->x[0][k] = ov * (profile+i)->y[1][k];
                (profile+i)->x[0][k] += av * (profile+i)->x[1][k];
                (profile+i)->x[0][k] /= av + ov * dtry[k];

                (profile+i)->y[0][k] = (profile+i)->x[1][k] - (profile+i)->x[0][k];
                (profile+i)->y[0][k] *= av / ov;
                (profile+i)->y[0][k] += (profile+i)->y[1][k];

                if (fabs((profile+i)->x[0][k] - (profile+i)->x[1][k]) > 0.01 * (profile+i)->x[0][k])
                    ick = 1;

                (profile+i)->x[1][k] = (profile+i)->x[0][k];
                (profile+i)->y[1][k] = (profile+i)->y[0][k];
            }

            for (k = 3; k < v; k++)
                (profile+i)->x[0][k] = (profile+i)->x[1][k];

            if (ick == 0) break;
        }

        for (j = 0; j < 3; j++) {
```

```
                for (k = 0; k < 2; k++) {
                        (profile+i)->ats[k][j] = (profile+i)->ots[k][j] = (profile+i)->tprof[0];

                        for (l = 0; l < v; l++)
                                (profile+i)->xs[k][j][l] = (profile+i)->x[0][l];

                        for (l = 0; l < 3; l++)
                                (profile+i)->ys[k][j][l] = (profile+i)->y[0][l];
                }
        }
    }

    return;
}
```

8.2.6 C_DISP.C

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>

#include "sephis.h"


void display_general_2 ( General *general, double prtime )
{
	gotoxy(1, 11);

	switch (general->ifast) {
		case 0:
			cputs("Using Runge-Kutta integration.");
			break;
		case 1:
			cputs("Using trapezoidal integration.");
			break;
		case 2:
			cputs("Using 'fast' integration.");
			break;
	}

	gotoxy(1, 12);

	switch (general->irxn) {
		case 0:
			cputs("No plutonium reaction.");
			break;
		case 1:
			cputs("Pu(IV) reduced instantaneously.");
			break;
		case 2:
			cputs("Pu(IV) reduced by U(IV).");
			break;
		case 3:
			cputs("Pu(IV) reduced by hydroxylamine nitrate.");
			break;
	}

	gotoxy(1, 13);
	cputs( "Print time =      min.");
	gotoxy(1, 14);
	cputs( "Maximum time =     min.");
	gotoxy(16, 13);
	cprintf("%4.0f", prtime);
	gotoxy(16, 14);
	cprintf("%4.0f", general->tstop);

	gotoxy(58, 10);
	cprintf("%8.4f", general->tol);

	return;
}
```

```
void status_screen ( void )
{
        int i;

        clrscr( );

        textcolor(CYAN);
        gotoxy(30, 1);
        cputs("*** STATUS SCREEN ***");
        gotoxy(1, 3);
        cputs("  CALCULATING --- PLEASE WAIT  ");
        gotoxy(1, 4);
        cputs("Press any key to quit prematurely");

        textcolor(WHITE);

        gotoxy(1, 1);
        cputs("SEPHIS MOD4 Ver. 2.11");
        gotoxy(59, 3);
        cputs("TIME  STAGE");
        gotoxy(58, 4);
        cputs("(MIN.)  NO.");
        gotoxy(58, 5);
        cputs("------ -----");
        gotoxy(50, 8);
        cputs("INVENTORY CHANGE (%/min.):");
        gotoxy(52, 9);
        cputs("Max.:");
        gotoxy(68, 9);
        cputs(">Stage:");
        gotoxy(52, 10);
        cputs("Tol.:");
        gotoxy(68, 10);
        cputs(">Comp.:");
        gotoxy(50, 12);
        cputs("OVERALL MATERIAL BALANCE (%):");
        gotoxy(54, 13);
        cputs("Nitric Acid:");
        gotoxy(54, 14);
        cputs("Uranium:");

        gotoxy(54, 15);
        cputs("Plutonium:");

        for (i = 56; i < 73; i++) {
                gotoxy(i, 2);
                putch('-');
        }

        for (i = 48; i < 79; i++) {
                gotoxy(i, 7);
                putch('-');
                gotoxy(i, 11);
                putch('-');
                gotoxy(i, 16);
```

```
        putch('-');
    }

    for (i = 3; i < 7; i++) {
        gotoxy(56, i);
        putch('|');
        gotoxy(72, i);
        putch('|');
    }

    for (i = 8; i < 16; i++) {
        gotoxy(47, i);
        putch('|');
        gotoxy(79, i);
        putch('|');
    }

    gotoxy(56, 2);
    putch('·');
    gotoxy(72, 2);
    putch('·');
    gotoxy(47, 7);
    putch('·');
    gotoxy(56, 7);
    putch('·');
    gotoxy(72, 7);
    putch('·');
    gotoxy(79, 7);
    putch('·');
    gotoxy(47, 11);
    putch('·');
    gotoxy(79, 11);
    putch('·');
    gotoxy(47, 16);
    putch('·');
    gotoxy(79, 16);
    putch('·');

    for (i = 2; i < 79; i++) {
        gotoxy(i, 22);
        putch('-');
    }

    for (i = 2; i < 30; i+=3) {
        gotoxy(i, 22);
        putch('+');
    }

    for (i = 34; i < 80; i+=5) {
        gotoxy(i, 22);
        putch('+');
    }

    gotoxy(1, 23);
    cputs("90 91 92 93 94 95 96 97 98 99");
    gotoxy(32, 23);
    cputs("99.1 99.2 99.3 99.4 99.5 99.6 99.7 99.8 99.9 100");
```

```
        gotoxy(27,24);
        cputs("Overall Material Balance (%)");
        gotoxy(1,18);
        cputs("URANIUM");
        gotoxy(1,20);
        cputs("PLUTONIUM");
        gotoxy(1,16);
        cputs("NITRIC ACID");

        return;
}


void main_title ( void )
{
        int i;

        draw_box(1,1,23,80);
        draw_box(10,23,14,56);

        for (i=3; i<24; i++) {
                gotoxy(i,10);
                putch('-');
        }

        for (i=55; i<79; i++) {
                gotoxy(i,10);
                putch('-');
        }

        gotoxy(1,10);
        putch('·');
        gotoxy(23,10);
        putch('·');
        gotoxy(55,10);
        putch('·');
        gotoxy(80,10);
        putch('·');

        textcolor(CYAN);
        gotoxy(29,3);
        cputs("PC - S E P H I S - MOD4");

        textcolor(WHITE);
        gotoxy(31,4);
        cputs("Version 2.11 (1990)");
        gotoxy(13,6);
        cputs("A Solvent Extraction Process Having Interacting Solutes");
        gotoxy(17,8);
        cputs("For Uranium, Plutonium, Thorium, and Nitric Acid");
        gotoxy(16,9);
        cputs("Mass Transfer in Purex- or Thorex-type Flowsheets");
        gotoxy(34,11);
        cputs("W.S. Groenier");
        gotoxy(30,12);
        cputs("Fuel Recycle Division");
        gotoxy(26,13);
```

```
        cputs("Oak Ridge National Laboratory");

        textcolor(CYAN);
        gotoxy(24, 17);
        cputs("*** PROBLEM CALCULATION ONLY ***");

        textcolor(WHITE);
        gotoxy(20, 19);
        cputs("Problem Definition file must be available");
        gotoxy(21, 20);
        cputs("Choose the location for this file below");
        gotoxy(14, 21);
        cputs("If not prepared, return to MENU and run option No. 1");

        return;
}


void display_stage_number ( int stage )
{
        gotoxy(66, 6);
        cprintf("%3d", stage);

        return;
}


void display_inventory_change ( General *general, Change *change )
{
        double comax;
        int ico, ist, i, j, v;

        v = 6;

        comax = 0.0;
        ico = ist = 0;

        for (i = 0; i < general->ntost; i++) {
                for (j = 0; j < v; j++) {
                        if ((change+i)->codumm[j] > comax) {
                                comax = (change+i)->codumm[j];
                                ico = j;
                                ist = i;
                        }
                }
        }

        if (comax > 100.0) comax = 100.0;

        gotoxy(76, 9);
        cprintf("%3d", ist+1);
        gotoxy(77, 10);
        cprintf("%2d", ico+1);
        gotoxy(58, 9);
        cprintf("%8.4f", comax);

        return;
```

```
}

void display_inventory_bar ( double *sot, double *soin )
{
        int i, j, r1, mmb;
        double mb[3];

        for (i = 0; i < 3; i++) {
                r1 = 17 + 2 * i;

                if (soin[i] < 1.0e-6)
                        mb[i] = 0.0;
                else {
                        gotoxy(2, r1);
                        for (j=0; j < 78; j++) putch(' ');
                        mb[i] = 100.0 * sot[i] / soin[i];
                        gotoxy(69, 13 + i);
                        cprintf("%6.2f", mb[i]);

                        if (mb[i] > 100.0) mmb = 79;
                        if (100.0 >= mb[i] && mb[i] > 99.0) {
                                mmb = (int)(5.0 * (10.0 * mb[i] - 990.0) + 29.0);
                        }

                        if (99.0 >= mb[i] && mb[i] >= 90.0) {
                                mmb = (int)(3.0 * (int)(mb[i] - 90.0) + 2.0);

                                for (j = 2; j <= mmb; j++) {
                                        gotoxy(j, r1);
                                        putch('=');
                                }
                        }

                        if (mb[i] > 99.0) {
                                for (j = 2; j < 30; j++) {
                                        gotoxy(j, r1);
                                        putch('=');
                                }
                                for (j = 30; j <= mmb; j++) {
                                        gotoxy(j, r1);
                                        putch('#');
                                }
                        }
                }
        }

        return;
}
```

8.2.7 C_PRN.C

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <math.h>

#include "sephis.h"


void printout_problem ( General *general, FILE *fp, char *file1, char *file2 )
{
	int i;

	fprintf(fp, "PROBLEM DEFINITION DATA FOR SEPHIS MOD4, Ver. 2.11:");
	for (i=0; i<64; i++) fputc(' ', fp);
	fprintf(fp, "%s %s¥n¥n", general->s_date, general->s_time);
	fprintf(fp, "	Probrem: %s¥n¥n", general->title);
	fprintf(fp, "		");
	fputs("PUREX PROCESS¥n", fp);
	fprintf(fp, "		Total stages: %d¥n", general->ntost);
	fprintf(fp, "		Volume %% TBP in solvent: %5.1f¥n", (general->ctbp * 100.0));
	fprintf(fp, "		Initial or default temperature (°C): %5.1f¥n", general->tempi);
	fprintf(fp, "		");

	if (general->nstr == 0)
		fprintf(fp, "No unusual stage connections¥n");
	else
		fprintf(fp, "Organic taken from stage %d feed stage %d.¥n", general->istr, general->jstr);

	fprintf(fp, "		");

	switch (general->irxn) {
		case 0:
			fputs("No Plutonium reaction.¥n", fp);
			break;
		case 1:
			fputs("Instantaneous reduction of Pu(IV).¥n", fp);
			break;
		case 2:
			fputs("Reduction of Pu(IV) by U(IV).¥n", fp);
			break;
		case 3:
			fputs("Reduction of Pu(IV) by hydroxylamine nitrate.¥n", fp);
			break;
	}

	fprintf(fp, "		Minutes per time increment: %7.2f¥n", general->dtheta);
	fprintf(fp, "		Minutes between printing of the concentration profile: %4.0f¥n", general->dprint);

	fprintf(fp, "		Calculations will stop after %4.0f minutes or when a tolerance of %8.4f %% per minute is
reached.¥n", general->tstop, general->tol);

	fprintf(fp, "		");

	switch (general->ivolm) {
```

```
        case 1:
                fputs("Mixer volumes are defined for each phase.¥n", fp);
                break;
        case 2:
                fputs("Total mixer volumes are defined (phase ratios = flow ratios).¥n", fp);
                break;
        case 3:
                fputs("Mixer volumes = (phase flow)(unit time).¥n", fp);
                break;
}

fprintf(fp, "        ");

switch (general->ivols) {
        case 1:
                fputs("Settler volumes are defined for each phase.¥n", fp);
                break;
        case 2:
                fputs("Total settler volumes are defined (phase ratios = flow ratios).¥n", fp);
                break;
        case 3:
                fputs("Settler volumes = (phase flow)(unit time).¥n", fp);
                break;
}

fprintf(fp, "        ");

switch (general->ipro) {
        case 0:
                fputs("Initial concentration profile is zero.¥n", fp);
                break;
        case 1:
                fputs("Initial concentration profile is defined.¥n", fp);
                break;
        case 2:
                fputs("Initial concentration profile is read from a previous problem.¥n", fp);
                break;
}

fprintf(fp, "        ");

switch (general->ifast) {
        case 0:
                fputs("Runge-Ketta integration will be used.¥n", fp);
                break;
        case 1:
                fputs("Trapezoidal integration will be used.¥n", fp);
                break;
        case 2:
                fputs("Fast integration will be used.¥n", fp);
                break;
}

fprintf(fp, "        ");

if (general->newout == 0)
        fputs("No extra product streams.¥n", fp);
```

```
        else
                fputs("Extra product stream(s).¥n", fp);

        fprintf(fp, "¥n        ");
        fprintf(fp, "The FILESPEC for this Problem Definition data file is:  %s¥n", file1);
        fprintf(fp, "        ");
        fprintf(fp, "The FILESPEC for this Problem Results data file is:  %s¥n¥n", file2);

        return;
}

void printout_streams_header ( FILE *fp )
{
        fputs("FEED AND PRODUCT STREAM DATA:¥n", fp);

        fputs("            ", fp);
        fputs("STAGE  NITRIC ACID    URANIUM     PU (IV)    PU (III)    REDUCTANT    NITRATE ION    FLOW RATE
TEMP¥n", fp);
        fputs("            ", fp);
        fputs(" NO.    (M)      (G/L)      (G/L)      (G/L)      (M)        (M)       (L/min)      (C)¥n", fp);
        fputs("            ", fp);
        fputs("----- ----------- ----------- ----------- ----------- ---------- ---------- ---------- ------¥n", fp);

        return;
}

void printout_feed_streams ( FILE *fp, General *general, Feed *feed )
{
        int i, j;

        for (i = 0; i < general->ntost; i++) {
                if (fabs((feed+i)->afdrt+(feed+i)->ofdrt) >= 1.0e-6 ) {
                        if (fabs((feed+i)->afdrt) >= 1.0e-6 ) {
                                fprintf(fp, "      AQUEOUS %3d", i+1);

                                for (j = 0; j < 6; j++)
                                        fprintf(fp,"    %.2e", (feed+i)->xfd[j]);

                                fprintf(fp,"    %.2e    %5.1f¥n",
                                        (feed+i)->afdrt, (feed+i)->afdtem);
                        }

                        if (fabs((feed+i)->ofdrt) >= 1.0e-6 ) {
                                fprintf(fp, "   %5.1f %% TBP %3d", general->ctbp * 100.0, i+1);

                                for (j = 0; j < 3; j++)
                                        fprintf(fp,"    %.2e", (feed+i)->yfd[j]);

                                for (j = 0; j < 42; j++) fputc(' ', fp);

                                fprintf(fp,"    %.2e    %5.1f¥n",
                                        (feed+i)->ofdrt, (feed+i)->ofdtem);
                        }
                }
        }

        return;
```

```
}

void printout_product_streams ( FILE *fp, int ntost, Product *product )
{
    int i, j;

    for (i = 0; i < ntost; i++) {
        if (fabs((product+i)->alvrt) >= 1.0e-6) {
            fprintf(fp, "     AQUEOUS %3d    ", i+1);
            fputs("Product stream removed (actual flow rate to be computed)", fp);
            for (j = 0; j < 22; j++) fputc(' ', fp);

            fprintf(fp,"    %.2e¥n", (product+i)->alvrt);
        }

        if (fabs((product+i)->olvrt) >= 1.0e-6) {
            fprintf(fp, "     ORGANIC %3d    ", i+1);
            fputs("Product stream removed (actual flow rate to be computed)", fp);
            for (j = 0; j < 22; j++) fputc(' ', fp);

            fprintf(fp,"    %.2e¥n", (product+i)->olvrt);
        }
    }

    return;
}

void printout_aqueous_profile_header ( FILE *fp )
{
    fputs("¥nINITIAL AQUEOUS PROFILE DATA:¥n", fp);
    fputs("          ", fp);
    fputs("STAGE  NITRIC ACID   URANIUM    PU (IV)   PU (III)   REDUCTANT   NITRATE ION
TEMP¥n", fp);
    fputs("          ", fp);
    fputs(" NO.    (M)      (G/L)     (G/L)     (G/L)     (M)       (M)              (C)¥n", fp);
    fputs("          ", fp);
    fputs("----- ----------  ----------  ----------  ----------  ----------  ----------          ------¥n", fp);

    return;
}

void printout_aqueous_profile ( FILE *fp, int ntost, Profile *profile )
{
    int i, j;

    for (i = 0; i < ntost; i++) {
        fprintf(fp, "          %3d", i+1);

        for (j = 0; j < 6; j++)
            fprintf(fp, "    %.2e", (profile+i)->x[1][j]);

        fprintf(fp, "          %5.1f¥n", (profile+i)->tprof[1]);
    }
```

```
        return;
}


void printout_organic_profile_header ( FILE *fp )
{
        fputs("¥nINITIAL ORGANIC PROFILE DATA:¥n", fp);
        fputs("            ", fp);
        fputs("STAGE  NITRIC ACID    URANIUM      PU (IV)¥n", fp);
        fputs("            ", fp);
        fputs(" NO.     (M)       (G/L)       (G/L)¥n", fp);
        fputs("            ", fp);
        fputs("----- ----------- ----------- -----------¥n", fp);

        return;
}


void printout_organic_profile ( FILE *fp, int ntost, Profile *profile )
{
        int i, j;

        for (i = 0; i < ntost; i++) {
                fprintf(fp, "            %3d", i+1);

                for (j = 0; j < 3; j++)
                        fprintf(fp, "    %.2e", (profile+i)->y[1][j]);

                fputc('¥n', fp);
        }

        return;
}


void printout_volumes_header ( FILE *fp )
{
        fputs("¥nSTAGE VOLUME AND FLOW RATE DATA (Molal basis):¥n", fp);
        fputs("            MIXER VOLUME BY PHASE", fp);
        fputs("    SETTLER VOLUME BY PHASE", fp);
        fputs("      MIXER FLOW RATE        INTERSTAGE FLOW RATE¥n", fp);

        fputs("   STAGE ------------------------- ------------------------- ------------------------- -------------------------¥n", fp);
        fputs("   NO.    AQUEOUS     ORGANIC     AQUEOUS     ORGANIC     AQUEOUS     ORGANIC     AQUEOUS
  ORGANIC¥n", fp);
        fputs("   ----- ----------- ----------- ----------- ----------- ---------- ----------- ---------- -----------¥n", fp);

        return;
}


void printout_volumes ( FILE *fp, int ntost, Volume *mixer, Volume *settler, Cascade *cascade )
{
        int i;

        for (i = 0; i < ntost; i++) {
                fprintf(fp, "    %3d", i+1);
```

```
            fprintf(fp, "    %.2e", (mixer+i)->avol);
            fprintf(fp, "    %.2e", (mixer+i)->ovol);
            fprintf(fp, "    %.2e", (settler+i)->avol);
            fprintf(fp, "    %.2e", (settler+i)->ovol);
            fprintf(fp, "    %.2e", (cascade+i)->at);
            fprintf(fp, "    %.2e", (cascade+i)->ot);
            fprintf(fp, "    %.2e", (cascade+i)->anet);
            fprintf(fp, "    %.2e¥n", (cascade+i)->onet);
        }

        return;
}


void printout_result_header ( FILE *fp, General *general, char *s_date, char *s_time )
{
        int i;

        fputs( "¥nPROBLEM RESULTS FOR SEPHIS MOD4, Ver. 2.11:", fp);

        for (i=0; i < 71; i++) fputc(' ', fp);
        fprintf(fp, "%s %.5s¥n¥n", s_date, s_time);
        fprintf(fp, "    Probrem: %s¥n", general->title);
        fputs( "    (mixer concentrations shown)¥n", fp);

        return;
}


void printout_result_aqueous_header ( FILE *fp )
{
        fputs("¥n AQUEOUS PHASE:¥n", fp);
        fputs(" STAGE  NITRIC ACID   URANIUM    PU (IV)   PU (III)   REDUCTANT  NITRATE ION   DENSITY
FLOW RATE   TEMP¥n", fp);
        fputs(" NO.    (M)     (G/L)     (G/L)     (G/L)     (M)     (M)     (g/mL)    (L/min)   (C)¥n", fp);
        fputs("----- ---------- ---------- ---------- ---------- ---------- ---------- ---------- ---------- ------¥n", fp);

        return;
}


void printout_result_organic_header ( FILE *fp )
{
        fputs("¥n ORGANIC PHASE:¥n", fp);
        fputs(" STAGE  NITRIC ACID   URANIUM    PU (IV)   U EXTRACT  PU EXTARACT  H+ EXTRACT   DENSITY
 FLOW RATE   TBPSAT   INVENTORY¥n", fp);
        fputs(" NO.    (M)     (G/L)     (G/L)     FACTOR    FACTOR    FACTOR    (g/mL)    (L/min)   (%)
CHANGE (%)¥n", fp);
        fputs("----- ---------- ---------- ---------- ---------- ---------- ---------- ---------- ---------- ------ ----------¥n",
fp);

        return;
}


void printout_result_aqueous ( FILE *fp, int i, Profile *profile, Product *product, Etc *etc, double conva )
{
```

```
    int j, k;

    fprintf(fp, " %3d ", i+1);

    for (j = 0; j < 6; j++)
        fprintf(fp, "   %5.2e", profile->x[1][j]);

    fprintf(fp, "   %5.2e   %5.2e   %5.1f¥n",
        etc->aden, etc->aout, profile->tprof[1]);

    if (fabs(product->alvrt) >= 1.0e-6) {
        fprintf(fp, " %3d ", i+1);
        fputs("  PRODUCT STREAM", fp);
        for (k=0; k < 74; k++) fputc(' ', fp);
        fprintf(fp, "   %5.2e¥n", conva * product->alvrt);
    }

    return;
}


void printout_result_organic ( FILE *fp, int i, Profile *profile, Cascade *cascade, Product *product, Etc *etc, Change
*change )
{
    int j, k;

    if (fabs(product->olvrt) >= 1.0e-6) {
        fprintf(fp, " %3d ", i+1);
        fputs("  PRODUCT STREAM", fp);
        for (k=0; k < 73; k++) fputc(' ', fp);
        fprintf(fp, "   %5.2e¥n", product->olvrt * etc->oout / cascade->ot);
    }

    fprintf(fp, " %3d ", i+1);

    for (j = 0; j < 3; j++)
        fprintf(fp, "   %5.2e", profile->y[1][j]);

    fprintf(fp, "   %5.2e   %5.2e   %5.2e",
        etc->ext2, etc->ext3, etc->ext1);
    fprintf(fp, "   %5.2e   %5.2e", etc->oden, etc->oout);
    fprintf(fp, "   %5.1f   %5.2e¥n", etc->tsat, change->codum);

    return;
}


void printout_result ( FILE *fp, General *general, Profile *profile, Product *product, Cascade *cascade, Etc *etc,
Change *change )
{
    double aq[6], org[3], solden, wo, t, conva, convo, temp;
    int v, i, j;

    v = 6;
    t = 3.65145 * general->ctbp;

    printout_result_aqueous_header(fp);
```

```
solden = 1000.0 * (general->ctbp * 266.32 / 273.6 + (1.0 - general->ctbp) * 170.34 / 227.5);

for (i = 0; i < general->ntost; i++) {
        for (j = 0; j < v; j++)
                aq[j] = (profile+i)->x[1][j];

        for (j = 0; j < 3; j++)
                org[j] = (profile+i)->y[1][j];

        temp = (profile+i)->tprof[1];

        conva = molar_aq(aq);
        convo = molar_org(org, general->ctbp, t, temp);

        (etc+i)->aden = (997.07 + 394.04 * aq[1] + 492.02 * (aq[2] + aq[3]) + 63.01 * aq[0] + 213.1 * aq[5]) /
(1000.0 + 72.4 * aq[1] + 130.0 * (aq[2] + aq[3]) + 30.9 * aq[0] + 31.0 * aq[5]);

        (etc+i)->aout = (cascade+i)->at * conva;
        (etc+i)->oout = (cascade+i)->ot * convo;

        for (j = 0; j < v; j++)
                (profile+i)->x[1][j] = aq[j] / conva;

        for (j = 0; j < 3; j++)
                (profile+i)->y[1][j] = org[j] / convo;

        (profile+i)->x[1][1] *= 238.0;
        (profile+i)->x[1][2] *= 239.0;
        (profile+i)->x[1][3] *= 239.0;

        printout_result_aqueous(fp, i, profile+i, product+i, etc+i, conva);
}

printout_result_organic_header(fp);

for (i = 0; i <general->ntost; i++) {
        (etc+i)->ext1 = (etc+i)->ext2 = (etc+i)->ext3 = 0.0;

        (etc+i)->tsat = (2.0 * (profile+i)->y[1][1] + 2.0 * (profile+i)->y[1][2] + (profile+i)->y[1][0]) * 100.0 / t;
        (profile+i)->y[1][1] *= 238.0;
        (profile+i)->y[1][2] *= 239.0;

        if ((etc+i)->aout > 1.0e-6) {
                (etc+i)->ext1 = (etc+i)->oout / (etc+i)->aout;

                if ((profile+i)->x[1][1] > 1e-10) {
                        (etc+i)->ext2 = (etc+i)->ext1 * (profile+i)->y[1][1];
                        (etc+i)->ext2 /= (profile+i)->x[1][1];
                }

                if (((profile+i)->x[1][2] + (profile+i)->x[1][3]) > 1e-10) {
                        (etc+i)->ext3 = (etc+i)->ext1 * (profile+i)->y[1][2];
                        (etc+i)->ext3 /= (profile+i)->x[1][2] + (profile+i)->x[1][3];
                }

                if ((profile+i)->x[1][0] > 1e-10) {
```

```
                    (etc+i)->ext1 *= (profile+i)->y[1][0] / (profile+i)->x[1][0];
                } else {
                    (etc+i)->ext1 = 0.0;
                }
            }

            wo = (4.2 - 0.015 * temp) * pow(general->ctbp, 1.69);
            wo *= (t - 2.0 * (profile+i)->y[0][1] - 2.0 * (profile+i)->y[0][2] - 0.6 * (profile+i)->y[0][0]) / t;

            (etc+i)->oden = solden + 18.02 * wo + 394.04 * (profile+i)->y[0][1] + 492.02 * (profile+i)->y[0][2] +
63.01 * (profile+i)->y[0][0];
            (etc+i)->oden /= 1000.0 + 17.4 * wo + 97.0 * (profile+i)->y[0][1] + 139.0 * (profile+i)->y[0][2] + 43.0 *
(profile+i)->y[0][0];

            printout_result_organic(fp, i, profile+i, cascade+i, product+i, etc+i, change+i);
        }

        return;
}


void printout_footer ( FILE *fp )
{
        fprintf(fp, "¥nValid ranges:¥n");
        fprintf(fp, "    0.0 to 5.0 M H+¥n");
        fprintf(fp, "    0.0 to 200 gU/L¥n");
        fprintf(fp, "    0.0 to 100 gPu/L¥n");
        fprintf(fp, "    10 to 30% TBP¥n");
        fprintf(fp, "    10 to 40° C¥n");

        return;
}
```

8.2.8 ETC.C

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <time.h>


char *string_time ( char *s_time )
        {
        time_t t;
        struct tm *tp;

        time(&t);
        tp = localtime(&t);

        sprintf(s_time, "%02d:%02d:%02d", tp->tm_hour, tp->tm_min, tp->tm_sec);

        return s_time;
        }


char *string_date ( char *s_date )
        {
        time_t t;
        struct tm *tp;

        time(&t);
        tp = localtime(&t);

        sprintf(s_date, "%02d/%02d/%02d", tp->tm_year, tp->tm_mon+1, tp->tm_mday);

        return s_date;
        }


void enable_bottom_line ( void )
{
        puts("¥x1b[>1h");
        return;
}


void disable_bottom_line ( void )
{
        puts("¥x1b[>1l");
        return;
}


void elase_cursur ( void )
{
        puts("¥x1b[>5h");
        return;
}
```

```
void display_cursur ( void )
{
        puts("¥x1b[>5l");
        return;
}


void draw_box ( int r1, int c1, int r2, int c2 )
{
        int i, j;

        gotoxy(c1, r1);
        putch('・');
        gotoxy(c2, r1);
        putch('・');

        for (i = r1 + 1; i < r2; i++) {
                gotoxy(c1, i);
                putch('|');
                gotoxy(c2, i);
                putch('|');
        }

        gotoxy(c1, r2);
        putch('・');
        gotoxy(c2, r2);
        putch('・');

        for (j = c1 + 1; j < c2; j++) {
                gotoxy(j, r1);
                putch('-');
                gotoxy(j, r2);
                putch('-');
        }

        return;
}
```