JAEA-Data/Code
2009-013

JAEA-Data/Code

一点炉動特性モデルを適用した臨界実験装置シミュレーター（CASIM）

Critical Assembly Simulator with the One-Point Reactor Kinetics (CASIM)

外池 幸太郎　内山 軍蔵

Kotaro TONOIKE and Gunzo UCHIYAMA

安全研究センター
原子力エネルギー関連施設安全評価研究ユニット

Nuclear Facility Safety Research Unit
Nuclear Safety Research Center

November 2009

Japan Atomic Energy Agency

日本原子力研究開発機構

本レポートは独立行政法人日本原子力研究開発機構が不定期に発行する成果報告書です。
本レポートの入手並びに著作権利用に関するお問い合わせは、下記あてにお問い合わせ下さい。
なお、本レポートの全文は日本原子力研究開発機構ホームページ（http://www.jaea.go.jp）
より発信されています。

独立行政法人日本原子力研究開発機構　研究技術情報部　研究技術情報課
〒319-1195　茨城県那珂郡東海村白方白根 2 番地 4
電話 029-282-6387, Fax 029-282-5920, E-mail:ird-support@jaea.go.jp

This report is issued irregularly by Japan Atomic Energy Agency
Inquiries about availability and/or copyright of this report should be addressed to
Intellectual Resources Section, Intellectual Resources Department,
Japan Atomic Energy Agency
2-4 Shirakata Shirane, Tokai-mura, Naka-gun, Ibaraki-ken 319-1195 Japan
Tel +81-29-282-6387, Fax +81-29-282-5920, E-mail:ird-support@jaea.go.jp

JAEA-Data/Code 2009-013

一点炉動特性モデルを適用した臨界実験装置シミュレーター（CASIM）

日本原子力研究開発機構　安全研究センター
原子力エネルギー関連施設安全評価研究ユニット

外池　幸太郎、内山　軍蔵

（2009 年 8 月 28 日　受理）

一般公衆向けの講演会等において核分裂連鎖反応の様子や基本的な原子炉の運転操作についての理解を助けるためのシミュレーターを開発した。ここで言う「基本的な」とは反応度フィードバックが存在しないことを意味し、単純な一点炉動特性モデルを用いている。したがって、核分裂連鎖反応のエネルギーにより炉心の温度上昇が見られる発電炉や研究炉ではなく、微小出力で常温で運転される臨界実験装置を模擬したものである。このことから、シミュレーターの名称を「CASIM : Critical Assembly SIMulator」とした。

CASIM では、中性子源の挿入・引抜、及び一種類の反応度制御のみが操作項目である。表示としては中性子源位置、中性子量（出力）の線型及び対数プロットを有する。炉周期の表示は備えない。

シミュレーションする動特性の設定として、遅発中性子割合 β を「0.74%」と「0.37%」から選べる。前者はウラン燃料を、後者はプルトニウム燃料を想定している。さらに、手動の反応度制御が困難な仮想的な状況として「$\beta = 0.19\%$」及び「遅発中性子なし」も選択することができる。

CASIM は標準的な Windows® パーソナルコンピューターで動作し、関西光科学研究所で開催された講演会等で使用されている。

原子力科学研究所（駐在）：〒319-1195　茨城県那珂郡東海村白方白根 2-4

Critical Assembly Simulator with the One-Point Reactor Kinetics (CASIM)

Kotaro TONOIKE and Gunzo UCHIYAMA

Nuclear Facility Safety Research Unit
Nuclear Safety Research Center
Japan Atomic Energy Agency
Tokai-mura, Naka-gun, Ibaraki-ken

(Received August 28, 2009)

A simulator has been developed to help, in a lecture meeting for instance, public understanding of fission chain reactions or basic nuclear reactor operations. "Basic" means, here, that there is no reactivity feedback and the simple one-point reactor kinetics are applied. The simulator, therefore, models a critical assembly operated with very low power in a room temperature, instead of a power reactor or a research reactor whose core temperature is raised by the energy of fission chain reaction. Thus, the simulator has been named as "CASIM : Critical Assembly SIMulator".

CASIM has only two operation items: the neutron source insertion/extraction, and the reactivity control of one kind. It has indications of the neutron source position and the neutron level (the power) in linear and logarithmic scales. No reactor period meter is equipped.

The delayed neutron fraction β, as a parameter of the simulated kinetics, is selectable from "0.74%" and "0.37%". The former is for a uranium fuel and the latter is for a plutonium fuel. Hypothetical situations such as "$\beta = 0.19\%$" or "Delayed Neutron : None" are also options to demonstrate the extreme conditions manually uncontrollable.

CASIM runs on a standard Windows® personal computer and has been used in lecture meetings such as a seminar held in the Kansai Photon Science Institute.

Keywords: Simulator, One-Point Reactor Kinetics, Critical Assembly, Windows®

目　次

Contents

1. はじめに

原子炉を用いてエネルギーを取り出す原子力利用において、核分裂連鎖反応は最も基本的な現象である。その核分裂連鎖反応の様子を探る実験装置のひとつに臨界実験装置がある[1)]。これは、小型だが、臨界量の核燃料物質と臨界状態を制御する機構を備え、核分裂連鎖反応を維持できる原子炉である。出力も非常に小さく、多くの場合、室温で運転される。

このような臨界実験装置は、核分裂連鎖反応に関する研究に用いられるだけでなく、原子力技術を新たに学ぼうとする学生や技術者の教育にも大変有用なものである[2)]。また、原子力利用を推進するにあたって、一般公衆の理解促進のためにも、臨界実験装置の運転を実演し、多くの方に見学・体験してもらうことは有益なことと考えられる[3)]。

しかしながら、小型・小出力と言えども、臨界実験装置は原子炉として厳格な安全規制の下におかれるため、その設置・運転・維持には多くのコストを要する。それゆえ日本国内でも臨界実験装置の数は減り、研究目的以外に教育や広報目的に用いられる余地は一層少なくなってきている。

一方で、計算技術の急速な発達は続き、学生や技術者はもちろん一般家庭へのパーソナルコンピューターの普及も拡大している。ここに、シミュレーターでよければ、原子炉に関する教育・広報に広く利用できる可能性がある。

そもそも、原子炉の運転員の教育・訓練にシミュレーターを用いることは、計算技術の応用の一つの典型として行われてきた[4)]。しかし、原子炉そのものよりは安価であっても、シミュレーターを構成する計算機は大型のものであり、学生や技術者個々人が保有すること、ましてや一般家庭に置くことは不可能であった。その理由には、計算機の規模以外にも、シミュレーターの目指す方向性として、原子炉で起こる現象のできるだけ多くを模擬し挙動が現実に近いものとすること、それに伴って、実際の原子炉と同じだけ複雑な操作を行わなければならない、あるいは、行えるようにすることがある[5)]。このため、中・小規模な部類のシミュレーターでも炉心冷却設備を持つ研究炉規模の原子炉を模擬したものであった。このような複雑なシミュレーターを小型のパーソナルコンピューターだけで構成することは、計算能力の点で現在では問題はなくても、操作の点で困難を伴う。また、初学者の学生、技術者や一般公衆向けには機能が過剰すぎる。

そこで、核分裂連鎖反応の様子をできるだけ忠実に模擬するものの、逆に、模擬する現象を極力少なくし、誰でもどこでも簡単な操作で動作するシミュレーターもまた有用なものとなり得ると考え、以下の技術的な目標を設定し、開発を行った。

- 外部中性子源の操作と反応度の操作の 2 種類のみを操作量とし、一点炉動特性モデルに従う模擬とする。(逆に、この他の理論モデルは導入しない。)
- 反応度フィードバックが無い小さな出力の原子炉を模擬する。すなわち、臨界実験装置のシミュレーターと位置づける。

- 遅発中性子が存在することで原子炉が手動で制御できること、逆に、遅発中性子がなければ原子炉の操作が困難であることを実演できる。
- 量販されているパーソナルコンピューターで動作し、特別な設定も不要なソフトウェアとする。

臨界実験装置のシミュレーターであることから、このソフトウェアの名称を「CASIM : Critical Assembly SIMulator」とすることにした。

目標の最後に挙げたように、量販されているパーソナルコンピューターで簡便に扱えるためには、臨界実験装置の動特性を少ない計算量で解かなければならない。しかし、核分裂連鎖反応にはシミュレーションを難しくするような特徴があり、計算には工夫が必要となる。この点について第 2 章で詳しく述べる。まず一点炉動特性モデルの説明をしているが、この部分は原子炉物理学の教科書の内容を要約したものに過ぎない。初学者の方には入門用に、指導者の方には復習用に利用して頂ければ幸いである。続いて CASIM に実装されている計算手法を述べているが、この部分が本報告書の技術的に中核の部分である。

他の目標は、できるだけ単純なシミュレーターとすること、初学者にとっても理解しやすいものとすることであり、計算手法よりもむしろ、ソフトウェアの外観や操作のデザインに依る。この点については、操作の説明も含めて第 3 章で詳しく述べる。ここは、いわばシミュレーション対象の臨界実験装置の操作方法の説明でもある。

第 2 章では、一点炉動特性モデルとその解法の説明はしているが、解そのもの、すなわち、原子炉の挙動の議論はしていない。また、第 3 章においても、臨界実験装置の操作の説明はしているが、その結果どのような応答が期待されるかは触れていない。したがって、本報告書の読者でかつ CASIM を使用する者は（独習者あるいは指導者)、一点炉動特性モデルに基づき、どのような操作に対してどのような結果が得られるかの基礎知識を有していることが前提となる。基礎知識を持たない者が CASIM を使用する際には、適切な指導が併せて行われることが強く望まれる。

第 4 章では、CASIM の開発の成果をまとめるとともに、開発の経緯についても触れる。

2. 動作原理

2.1 一点炉動特性モデル[6)]

2.1.1 核分裂連鎖反応の基礎と近似によるモデル導出の概要

ウランやプルトニウムなど（以後、単に「燃料」と呼ぶ。）の核分裂性の原子核に中性子が衝突し、もし核分裂反応が起きると、新たな中性子が発生する。この新たな中性子もまた、燃料の原子核と出会い、核分裂反応を引き起こす可能性がある。

燃料の原子核と中性子が衝突する機会の有無、衝突した際に核分裂反応が起きるか否か、核分裂反応が起きたとして新たな中性子が何個発生するか、これらはすべて具体的な数値を決めることはできず、個々の事象が発生する確率だけが知られている。しかし、これらの数値を積み重ねることにより、膨大な数の中性子の集合体としては、統計的・平均的な振る舞いを決めることができる。この関係は、個々の質点の運動を記述する古典力学と、膨大な数の粒子の集合体を記述する統計力学の関係とほぼ同様である。

統計的な振る舞いについてのイメージを補強するために、さらに仮定を追加する。中性子の量も膨大だが、燃料の原子核の数は、中性子よりも桁違いに多いとする。このことは、核分裂反応で燃料の原子核が消耗しても、全体量に比べれば無視できることを意味する。また、中性子の量に比例して核分裂反応の量も増えることとなり、つまり、中性子の量と出力が比例することも意味する。

したがって、原子炉の核分裂連鎖反応の様子を定量的に議論するためには、炉心における中性子の振る舞いを数学的に記述できなければならない。電気的に中性な中性子は、物質中でもあたかもガスのように振る舞い、例えばボルツマンの輸送方程式を用いて定式化できる。

この方程式は、空間座標、中性子のエネルギー、中性子の運動の方向、及び時間を変数として持ち、解析的な手法はもちろん、コンピューターを用いた数値的な手法でも完全に解くことは難しい。このため種々の近似を行うことになる。

まず、中性子密度の空間分布と中性子の運動の方向をフィックの法則を用いて結びつけ、輸送方程式を拡散方程式に近似できる。さらに、中性子のエネルギーが一種類のみであると仮定して、いわゆる1群拡散モデルを得る。

ここで、中性子密度の空間分布の形が時間的に変化しないと仮定すると、原子炉内のある一点における中性子密度の時間変化を常微分方程式だけで記述できる。これが「一点炉動特性モデル」と呼ばれるものである。

本節及び次節以降で概観する一点炉動特性モデルについては、多くの原子炉物理学の教科書で詳細に述べられているので、参考にされたい。本報告は主に文献6)の「原子炉の理論と解析」に依っている。

2.1.2 基礎的な動特性方程式の例

以上を踏まえて、原子炉出力の時間変化に関する動特性を記述するために、3 つの量を導入する。まず、中性子は平均的に l の寿命を持つものとする。これには、寿命を終えるにあたって核分裂反応を引き起こす場合も、引き起こさない場合も含まれる。次に、中性子増倍率 k である。核分裂連鎖反応の世代ごとに平均的に中性子が増減する率を表す。$k=1$ のとき中性子の増減はなく一定である。$k<1$ であれば減少し、$k>1$ であれば増加する。最後に中性子量を示す N である。これは、炉心全体の中性子の総量でも、炉心内のある代表の一点における中性子の密度でも、炉心全体の中性子密度の平均値でもよい。この任意性は前節で述べた中性子密度の空間分布の形が時間的に変化しないことによる。これらの 3 つの量を用いると、原子炉の最も基本的な動特性方程式として、中性子量 N の時間変化を記述する時間 t についての常微分方程式が得られる。

$$\frac{dN}{dt}=\frac{k-1}{l}N(t) \tag{2-1}$$

この方程式の解は、$t=0$ における N の初期値を N_0 とすれば、

$$N(t)=N_0\,e^{st} \tag{2-2}$$

という形で書ける。ただし、$s=(k-1)/l$ であり、中性子量が増減する時間的な速さを表す量となる。特に s の逆数は「原子炉ペリオド」と呼ばれ、中性子量が e 倍（約 2.7 倍）に増加するために要する時間である。

仮に、核分裂連鎖反応の世代ごとに中性子量が 0.1% ずつ増える状況を考える。つまり中性子増倍率が $k=1.001$ であり、「徐々に増える」という印象を持たれるかもしれない。ところが、通常の原子炉（例えば中性子の減速に水を用いる原子炉）では、中性子の寿命は高だか $l=10^{-4}$ s 程度であることが知られており、原子炉ペリオド $1/s$ は 0.1 s 程度となってしまう。この状況を 1 s 放置すれば、中性子量つまり出力は 2 万倍にも増加する。これでは手動による原子炉の出力制御はまったく困難であろう。

中性子量の世代ごとの変化を一桁小さく 0.01% と仮定する。つまり中性子増倍率が $k=1.0001$ だとしても、原子炉ペリオド $1/s$ は 1 s 程度である。原子炉の炉心への燃料の装荷は、燃料体や燃料集合体単位で行われる離散的な操作であることがほとんどである。臨界に近い状態で燃料を 1 体追加すると、中性子増倍率 k の変化は 0.01% よりもずっと大きくなる。制御棒を用いて k を自動的に調整するにしても、k の変化を 0.01% より小さく、かつ、中性子寿命 l に比べて十分速く行うことは難しい。

手動、自動のいずれにしても原子炉で核分裂連鎖反応を制御することは非常に困難なものと感じられるが、実際にははるかに容易である。その理由を次節で述べる。

2.1.3 遅発中性子の存在とその効果

前節で式(2-1)とその解である式(2-2)に含まれる中性子寿命 l は、厳密には、原子炉内で核分裂反応そのものから中性子が生まれ、その中性子が何らかの理由で失われるまでの平均的な時間である。失われる理由には原子核による吸収や原子炉外への漏出がある。ところが、核分裂反応で発生する中性子の一部には、反応で直ちに発生するのではなく、しばらく別の形で存在した後に遅れて中性子になるものがあり、遅発中性子と呼ばれる。一方、反応で直ちに発生する中性子は即発中性子と呼ばれる。すなわち、前節で式(2-1)と式(2-2)に基づいて核分裂連鎖反応の制御が困難と述べた議論は、即発中性子しか存在しないと仮定した場合のものとなる。

遅発中性子は、核分裂反応そのものから発生するのではなく、核分裂した原子核の破片、つまり核分裂片がさらに崩壊する際に発生する。したがって、遅発中性子も考慮して前節の式(2-1)のような方程式を立てるためには、単純な中性子寿命 l に加えて、核分裂片の崩壊までの寿命も考慮しなければならない。

遅発中性子が発生する核分裂片には様々な核種があるが、すべてを動特性方程式に組み込むことは現実的ではなく、核分裂片の崩壊までの寿命に着目して比較的少数のグループに分けることが便利である。このグループ分けはいくつかの方法が提唱されているが、広く用いられるものにKeepinによる分類がある。遅発中性子が発生する核分裂片を寿命に基づいて6種類に分類し、それぞれのグループを1種類の核分裂片で代表させるために、平均の寿命を提示している。また、すべての遅発中性子のうち、グループごとに発生する割合も示している。表2-1にKeepinが ^{235}Uの熱核分裂反応について提示した値を示す[7,8]。

グループは「群」と呼ばれ、添字で番号を付して表示される。1つの群を1つの寿命の仮想的な核種で代表させるとき、その核種は「遅発中性子先行核」と呼ばれる。その寿命は、崩壊定数 λ_i の形で示されている。また、遅発中性子のうち、群ごとの寄与率が相対収率 α_i として示されている。

表2-1 ^{235}Uの熱核分裂に関する遅発中性子先行核のデータ

群	崩壊定数 λ_i (s^{-1})	相対収率 α_i
1	0.0124 ± 0.0003	0.033 ± 0.003
2	0.0305 ± 0.0010	0.219 ± 0.009
3	0.111 ± 0.004	0.196 ± 0.022
4	0.301 ± 0.011	0.395 ± 0.011
5	1.14 ± 0.15	0.115 ± 0.009
6	3.01 ± 0.29	0.042 ± 0.008

これらの値は、他の核分裂性核種（例えば ^{239}Pu）では若干異なるが、原子炉の速い応答を除いて、秒単位の比較的ゆっくりとした応答の議論ではあまり問題にならない。また、Keepinは熱中性子による核分裂の場合としてこの値を示したが、核分裂反応を引き起こす中性子のエネルギーが4 MeV以下であればあまり変化がないことも知られている。

ここで重要なことは、遅発中性子が全中性子に占める割合が、^{235}U と ^{239}Pu では大きく異なることである。^{235}U の核分裂反応にだけ着目し（連鎖反応は考えない。）、即発中性子と遅発中性子を数え上げた場合、遅発中性子は全ての中性子の約 0.64% である。一方 ^{239}Pu の場合は 0.20%、^{241}Pu では 0.49% とされている [8]。さらに、連鎖反応を媒介する中性子の集団を考えると、即発中性子に比べて遅発中性子は発生する際のエネルギーが低いことに注意しなければならない。つまり、エネルギーの高い即発中性子は、減速するまでに原子炉の外に漏れる傾向がやや強く次の核分裂反応を引き起こさない確率が若干高い。逆に、遅発中性子は発生する際のエネルギーが低い分、原子炉の外に漏れることなく減速を終え、燃料の核種に出会える機会が若干多い。このため、核分裂連鎖反応に寄与している遅発中性子の割合は、^{235}U の 0.64% や ^{239}Pu の 0.20% に比べて実効的には大きくなる。熱体系の小型の臨界実験装置でウランを用いている場合では、例えば 0.74% などである [9]。^{239}Pu と ^{241}Pu が混在しているプルトニウム燃料を用いている場合は、例えば、0.37% などとなる。本報告では、これらの実効的な遅発中性子の割合を β で表記する。

遅発中性子先行核の崩壊定数 λ_i、相対収率 α_i、遅発中性子割合 β、及び中性子寿命 l を用いると、遅発中性子が存在する場合の実効的な中性子寿命 $<l>$ は、

$$<l>=(1-\beta)\,l+\sum_{i=1}^{6}\alpha_i\beta\left(\frac{1}{\lambda_i}+l\right) \tag{2-3}$$

と表すことができる。ここで、第 1 項が中性子そのものの寿命の寄与（即発中性子の寿命の寄与と言っても良い）、総和記号の項が遅発中性子先行核の寿命の寄与と考えられる。これに表 2-1 に掲げた数値を代入すると、$<l>$ は 0.1 s 程度となる。これは、中性子寿命 l が 10^{-4} s 程度であることに比べて相当に長く、中性子増倍率 k が 0.1% 変化するような場合でも原子炉ペリオドは数十 s から 100 s 程度で、十分に制御が可能となる。

以後、l は即発中性子の寿命を表す量として「即発中性子寿命」と呼ぶ。

2.1.4 遅発中性子を考慮した動特性方程式

遅発中性子が存在することを前提に、式 (2-1) を書き換えると、中性子量 N に遅発中性子先行核の量 C_i (i=1,...,6) を加えた連立の微分方程式となる。式 (2-1) で示したような即発中性子による連鎖は、β だけ減じることとなる一方、$\lambda_i C_i$ (i=1,...,6) の発生率で 6 種類の遅発中性子先行核の寄与が新たに加わる。したがって、中性子量 N については、以下の関係が成り立つ。

$$\frac{dN}{dt}=\frac{k(1-\beta)-1}{l}N(t)+\sum_{i=1}^{6}\lambda_i C_i(t) \tag{2-4}$$

一方、遅発中性子先行核の量 C_i については、

$$\frac{dC_i}{dt}=\beta_i\frac{k}{l}N(t)-\lambda_i C_i(t)\quad (i=1,\ldots,6) \tag{2-5}$$

となる。ただし、$\beta_i=\alpha_i\cdot\beta$ とする。

ここで、動特性について議論する際に有用な2つの量、反応度 ρ と世代時間 Λ を定義する。反応度 ρ は、臨界からの隔たりを示す量で、中性子増倍率 k の変化率として次のように定義される。

$$\rho = \frac{k-1}{k} \tag{2-6}$$

世代時間 Λ は、即発中性子寿命 l と中性子増倍率 k を用いて

$$\Lambda = \frac{l}{k} \tag{2-7}$$

と定義される。これらの反応度 ρ と世代時間 Λ を用いると式 (2-4) と式 (2-5) は次のように書き換えられる。

$$\frac{dN}{dt} = \frac{\rho-\beta}{\Lambda} N(t) + \sum_{i=1}^{6} \lambda_i C_i(t) \tag{2-8}$$

$$\frac{dC_i}{dt} = \frac{\beta_i}{\Lambda} N(t) - \lambda_i C_i(t) \quad (i = 1,\ldots,6) \tag{2-9}$$

これらを一点炉動特性方程式と呼び、シミュレーターの基本式とする。なお、外部中性子源が存在する場合には、式 (2-8) の右辺に発生率 S の源の項が加わる。

2.1.5 逆時間方程式

式 (2-8) と式 (2-9) の動特性方程式を数値的に積分して解く方法は次節以降で詳述するが、その結果を検証するために予め別の方法による解も得ておく。

初期条件として中性子量が臨界状態で一定に保たれているところに、ある瞬間、ステップ状に反応度が加わるような場合は、比較的容易に解くことができる。そのひとつの方法が、中性子量は、時間 $t = 0$ におけるステップ状の反応度 ρ の投入のあと

$$N(t) = \sum_{i=1}^{7} A_i e^{s_i t} \quad \text{ただし、} \sum_{i=1}^{7} A_i = 1 \tag{2-10}$$

という指数関数の和の形で変化すると仮定するものである。ただし中性子量の初期値は1とした。遅発中性子先行核の濃度についても同様に仮定し、ラプラス変換を式 (2-8) と式 (2-9) に適用すると、

$$sN = \frac{\rho-\beta}{\Lambda} N + \sum_{i=1}^{6} \lambda_i C_i \tag{2-11}$$

$$sC_i = \frac{\beta_i}{\Lambda} N - \lambda_i C_i \quad (i = 1,\ldots,6) \tag{2-12}$$

となる。この連立方程式が解を持つためには、次の行列式の値がゼロでなければならない。

$$\begin{vmatrix} \frac{\rho-\beta}{\Lambda}-s & \lambda_1 & \lambda_2 & \cdots & \lambda_6 \\ \frac{\beta_1}{\Lambda} & -\lambda_1-s & & & \\ \frac{\beta_2}{\Lambda} & & -\lambda_2-s & & \\ \vdots & & & \ddots & \\ \frac{\beta_6}{\Lambda} & & & & -\lambda_6-s \end{vmatrix} = 0 \qquad (2\text{-}13)$$

この関係に式 (2-6) と式 (2-7) の定義を逆に適用すると、ρ を与えたときの s の方程式

$$\rho = \frac{sl}{sl+1} + \frac{1}{sl+1}\sum_{i=1}^{6}\left(\frac{s\beta_i}{s+\lambda_i}\right) \qquad (2\text{-}14)$$

を得る。この方程式の 7 つの根が式 (2-10) の s_i となる。そのうち最大の根 s_1 が中性子量の時間変化の速さを示す量となり重要である。s_1 の逆数は、中性子量が e 倍に増加するために要する時間で、式 (2-2) に現れた s の逆数と同様に、原子炉ペリオドと呼ばれる。

この方程式は、反応度と原子炉応答の速さを関係付けるものであり、s が時間の逆数の次元を持つことに因んで「逆時間方程式」と呼ばれている。すべての根 s_i と式 (2-10) の A_i を正確に求めるためには数値的に解く必要があり、その詳細な方法と計算結果は付録Aに示す。

2.2 数値解法 [10,11)

2.2.1 陰的解法の必要性と補外による精度向上

常微分方程式を数値的に解く時、十分に細かい刻み幅で積分する基本的な操作（例えばRunge-Kutta法）と、いくつかの刻み幅による積分結果を刻み幅が無限小の条件へ外挿することにより精度を向上させる操作（例えばRichardson補外）を組み合わせれば、計算に要する時間は別にして、どのような場合にでも安定した解が得られるように想像される。しかし、一点炉動特性方程式を実際に解いてみると、不具合が多く見出される。

不具合の症状は、予想をはるかに超えた多くの計算時間を要するという形で現れるが、その原因を探ると、刻み幅を変えて積分すると結果が一様な変化を示さず振動することがわかる。このような振動が起こると、刻み幅が無限小の条件へ外挿することができない。仮に数値処理の偶然から外挿値を得たとしても、その値が正当である保証はない。そしてこの過程で、無意味に刻み幅を小さくして試行を繰り返すことにより、膨大な計算時間を消費することとなる。

このような振動あるいは解の発散が起こる理由は、一点炉動特性方程式の特殊な性質による。一点炉動特性方程式が大きく異なる時間成分を解として持つ、つまり、即発中性子寿命に依存して非常に速い時間変化を示す項と、遅発中性子先行核の崩壊定数に依存して遅い時間変化を示す項の和が解となっていることである。このような方程式は俗に「硬い連立方程式」と呼ばれており、前進型の（陽的な）解法では安定した解が得られないことが知られている。このような方程式を安定して解くためには、後退型の（陰的な）解法を用いなければならない。

さて、Runge-Kutta法は、陽的なEuler法を改良したものであり、基本的に前進解法である。したがって、先に述べたように、刻み幅制御（安定した解が得られるように十分に細かい刻み幅を自動的に探る機能）を導入しても、「硬い連立方程式」の解の不安定性を回避することはできない。

一方、Richardson補外は、刻み幅制御の方法の一例であり、微分方程式の解法そのものはいろいろな方法から選択することができる。単純なEuler法でもRunge-Kutta法でもよい。特に、修正中点法による微分方程式の解法と有理関数補外を組み合わせた方法は極めて高精度な計算を実現できることが知られており、Bulirsch-Stoer法と呼ばれる。しかし、残念ながら、修正中点法も、Euler法やRunge-Kutta法と同じく陽的な解法であり、「硬い連立方程式」の問題には適用することができない。

しかし、陰的な微分方程式の解法と刻み幅制御のRichardson補外を組み合わせて用いることは可能であり、「硬い連立方程式」である一点炉動特性方程式に適用できる。

2.2.2 陰的な解法の数学的な表現

ここでは、xの関数yの微分方程式を一定の刻み幅hで解く場合の陰的な解法を示す。比較のために、より基本的な陽的な解法も併せて述べる。

陽的な解法とは、ステップnの既知の関数値と微分値をもとに次のステップ$n+1$の関数値を求める方法であり、級数の安定性の観点から不安定、つまり発散する場合が多い。

この不安定性を克服するためには、刻み幅 h を十分に小さくする以外に方法はない。一方、陰的な解法とは、ステップ n の既知の関数値と次のステップ $n+1$ の未知の微分値をもとに次のステップ $n+1$ の関数値を求める方法であり、安定な級数となる。刻み幅 h を大きくとっても、解の精度は落ちるものの、発散することはない。ステップ $n+1$ の未知の微分値は、通常は、1 次線形近似によりステップ n の既知の関数値から求めることとなる。

ごく単純な例として、定係数方程式

$$y' = cy \tag{2-15}$$

を考える。この方程式を刻み幅 h で解く陽的な Euler 法は、

$$y_{n+1} = y_n + hy'_n = (1+ch)y_n \tag{2-16}$$

と表現することができる。この方法は、c と h の値によっては不安定となる。一方、陰的な Euler 法は、

$$y_{n+1} = y_n + hy'_{n+1} = \frac{y_n}{1-ch} \tag{2-17}$$

となり、たとえ $h \to \infty$ であっても、$y_{n+1} \to 0$ となり安定である。

これを連立方程式の場合に拡張することは容易である。ベクトルと行列を用いて、

$$\mathbf{y}' = \mathbf{C}\mathbf{y} \tag{2-18}$$

とする。ここで、$\mathbf{C}$ は行列、$\mathbf{y}'$ と $\mathbf{y}$ はベクトルである。このとき陰的な Euler 法は

$$\mathbf{y}_{n+1} = (\mathbf{1}-\mathbf{C}h)^{-1}\mathbf{y}_n \tag{2-19}$$

となる。

さらに、定係数方程式ではなく、$\mathbf{y}'$ が x と $\mathbf{y}$ の関数

$$\mathbf{y}' = \mathbf{f}(x,\mathbf{y}) \tag{2-20}$$

である場合を考える。陰的な Euler 法は

$$\mathbf{y}_{n+1} = \mathbf{y}_n + h\mathbf{f}(x_{n+1},\mathbf{y}_{n+1}) \tag{2-21}$$

となる。一般に、これは連立非線形方程式であり、ステップごとに反復法で解かなければならないが、$\mathbf{y}$ の変化が十分に小さければ次のように線形化近似することができる。

$$\mathbf{y}_{n+1} = \mathbf{y}_n + h\left\{\mathbf{f}(x_{n+1},\mathbf{y}_n) + \left.\frac{\partial \mathbf{f}}{\partial \mathbf{y}}\right|_{\mathbf{y}_n}(\mathbf{y}_{n+1}-\mathbf{y}_n)\right\} \tag{2-22}$$

したがって、次の式を得る。

$$\left[\mathbf{1} - h\left.\frac{\partial \mathbf{f}}{\partial \mathbf{y}}\right|_{\mathbf{y}_n}\right]\mathbf{y}_{n+1} = \mathbf{y}_n + h\left\{\mathbf{f}(x_{n+1},\mathbf{y}_n) - \left.\frac{\partial \mathbf{f}}{\partial \mathbf{y}}\right|_{\mathbf{y}_n}\mathbf{y}_n\right\} \tag{2-23}$$

さらに変形して、

$$\mathbf{y}_{n+1}=\left[\mathbf{1}-h\left.\frac{\partial \mathbf{f}}{\partial \mathbf{y}}\right|_{\mathbf{y}_n}\right]^{-1}\left[\mathbf{y}_n+h\left\{\mathbf{f}(x_{n+1},\mathbf{y}_n)-\left.\frac{\partial \mathbf{f}}{\partial \mathbf{y}}\right|_{\mathbf{y}_n}\mathbf{y}_n\right\}\right] \tag{2-24}$$

となる。なお、この式と定係数方程式の場合を比較すると、

$$\mathbf{C}=\left.\frac{\partial \mathbf{f}}{\partial \mathbf{y}}\right|_{\mathbf{y}_n} \tag{2-25}$$

$$\mathbf{f}(x_{n+1},\mathbf{y}_n)-\left.\frac{\partial \mathbf{f}}{\partial \mathbf{y}}\right|_{\mathbf{y}_n}\mathbf{y}_n=\mathbf{C}\mathbf{y}_n-\mathbf{C}\mathbf{y}_n=\mathbf{0} \tag{2-26}$$

であり、式 (2-19) と一致する。

ここまでは精度が 1 次の例を示した。1 次のものは非常に頑健でありよい安定性を示すが、ほとんどの場合、高次の方法も安定である。2 次の方法は単純で、つぎのように陽的な方法と陰的な 1 次の方法の平均をとればよい。

$$y_{n+1}=y_n+\frac{h}{2}(y'_n+y'_{n+1}) \tag{2-27}$$

式 (2-15) の方程式については、次のように解を得ることができる。

$$y_{n+1}=\frac{1+ch/2}{1-ch/2}y_n \tag{2-28}$$

連立方程式で非定係数の場合、

$$\mathbf{y}_{n+1}=\mathbf{y}_n+\frac{h}{2}\left\{\mathbf{f}(x_n,\mathbf{y}_n)+\mathbf{f}(x_{n+1},\mathbf{y}_n)+\left.\frac{\partial \mathbf{f}}{\partial \mathbf{y}}\right|_{\mathbf{y}_n}(\mathbf{y}_{n+1}-\mathbf{y}_n)\right\} \tag{2-29}$$

から、

$$\left[\mathbf{1}-\frac{h}{2}\left.\frac{\partial \mathbf{f}}{\partial \mathbf{y}}\right|_{\mathbf{y}_n}\right]\mathbf{y}_{n+1}=\left[\mathbf{y}_n+\frac{h}{2}\left\{\mathbf{f}(x_n,\mathbf{y}_n)+\mathbf{f}(x_{n+1},\mathbf{y}_n)-\left.\frac{\partial \mathbf{f}}{\partial \mathbf{y}}\right|_{\mathbf{y}_n}\mathbf{y}_n\right\}\right] \tag{2-30}$$

$$\mathbf{y}_{n+1}=\left[\mathbf{1}-\frac{h}{2}\left.\frac{\partial \mathbf{f}}{\partial \mathbf{y}}\right|_{\mathbf{y}_n}\right]^{-1}\left[\mathbf{y}_n+\frac{h}{2}\left\{\mathbf{f}(x_n,\mathbf{y}_n)+\mathbf{f}(x_{n+1},\mathbf{y}_n)-\left.\frac{\partial \mathbf{f}}{\partial \mathbf{y}}\right|_{\mathbf{y}_n}\mathbf{y}_n\right\}\right] \tag{2-31}$$

を得る。定係数の場合は、

$$\mathbf{C}=\left.\frac{\partial \mathbf{f}}{\partial \mathbf{y}}\right|_{\mathbf{y}_n} \tag{2-32}$$

$$\mathbf{f}(x_n,\mathbf{y}_n)=\mathbf{f}(x_{n+1},\mathbf{y}_n)=\left.\frac{\partial \mathbf{f}}{\partial \mathbf{y}}\right|_{\mathbf{y}_n}\mathbf{y}_n=\mathbf{C}\mathbf{y}_n \tag{2-33}$$

であり、式(2-31)は、

$$\mathbf{y}_{n+1} = [\mathbf{1} - h\mathbf{C}/2]^{-1}[\mathbf{1} + h\mathbf{C}/2]\mathbf{y}_n \tag{2-34}$$

というように、式(2-28)に対応する形となる。

2.2.3 一点炉動特性方程式への適用

遅発中性子を6群で近似する場合、一点炉動特性方程式は、

$$\frac{dY_1}{dx} = \frac{\rho - \beta}{\Lambda} Y_1 + \sum_{i=1}^{6} \lambda_i Y_{i+1} + S \tag{2-35}$$

$$\frac{dY_{i+1}}{dx} = \frac{\beta_i}{\Lambda} Y_1 - \lambda_i Y_{i+1} \quad (i = 1, \cdots, 6) \tag{2-36}$$

と表現できる。ここでは、「2.2.2 **陰的な解法の数学的な表現**」の表記方法とあわせるために、時間を x、中性子量を Y_1、遅発中性子先行核密度を $Y_2,...,Y_7$ で示している。その他の表記は慣例に従った。

反応度 ρ が時間 x によらず一定で、かつ中性子源強度が $S=0$ とすると、方程式は式(2-18)の形で表現することができ、この場合、行列 $\mathbf{C}$ は次のとおりとなる。

$$\mathbf{C} = \begin{bmatrix} \frac{\rho-\beta}{\Lambda} & \lambda_1 & \lambda_2 & \lambda_3 & \lambda_4 & \lambda_5 & \lambda_6 \\ \frac{\beta_1}{\Lambda} & -\lambda_1 & & & & & \\ \frac{\beta_2}{\Lambda} & & -\lambda_2 & & & & \\ \frac{\beta_3}{\Lambda} & & & -\lambda_3 & & & \\ \frac{\beta_4}{\Lambda} & & & & -\lambda_4 & & \\ \frac{\beta_5}{\Lambda} & & & & & -\lambda_5 & \\ \frac{\beta_6}{\Lambda} & & & & & & -\lambda_6 \end{bmatrix} \tag{2-37}$$

反応度 ρ が時間 x の関数である場合、中性子源 S が存在する場合も、式(2-37)の表現を式(2-31)にそのまま使うことができる。中性子源 S は $\mathbf{f}$ の算出において考慮すればよい。$\mathbf{f}$ は、式(2-35)、式(2-36)の右辺そのものである。

2.2.4 コーディングの具体例

➢ 時間及び中性子量から中性子量の微分値の算出

表2-2に、時間と中性子量（遅発中性子先行核濃度を含む。以下同じ。）から、中性子量の微分値を計算するプログラムを示す。プログラミング言語はVisual C#®（VC#）[12]を用いている。t は時間(s)、y[]は中性子量を入力する変数である。y[]は長さが 7 の配列で、要素 0 が中性子量 N に、要素 1～6 が遅発中性子先行核濃度 C_i に相当する。計算

結果として dydx[]に y[]の時間微分が格納される。base.Rho()は反応度を、base.S()は中性子源強度を返す時間 t の関数である。base.PL と Beta はそれぞれ即発中性子寿命 (s) と遅発中性子割合、Alpha[]と Lamda[]はそれぞれ遅発中性子先行核の相対収率と崩壊定数 (s^{-1}) であり、これらは定数である。

このプログラムは、式 (2-35) と式 (2-36) を実装したものである。

表 2-2　中性子量の微分値の計算（VC#）

```
private void GetDyDx(double t, double[] y, double[] dydx) {
    double rho = base.Rho(t);
    double gt = (1 - rho) * base.PL;

    //dN/dT
    dydx[0] = (rho - Beta) / gt * y[0];
    for(int i = 0; i <= 5; i++) dydx[0] += Lamda[i] * y[i + 1];
    dydx[0] += base.S(t);

    //dCi/dT
    for(int i = 0; i <= 5; i++)
        dydx[i + 1] = Beta * Alpha[i] / gt * y[0] - Lamda[i] * y[i + 1];
}
```

➢ 陰的差分により時間メッシュを一つ先に進む

ここでは、時間 t で始まる幅 dt の時間メッシュを陰的差分により一つ先に進むプログラムを示す。時間 t における既知の中性子量は yold[]で与えられ、結果は yold[]に上書きされるものとする。精度は 2 次、つまり、式 (2-29)、式 (2-30)、及び式 (2-31) に対応するものである。

まず、式 (2-35) と式 (2-36) に基づき、

$$\left.\frac{\partial \mathbf{f}}{\partial \mathbf{y}}\right|_{\mathbf{y}_n} \tag{2-38}$$

を表す二次元配列 C[,]を作成する。表 2-3 に遅発中性子先行核の崩壊定数 λ_i のみからなる要素を事前にセットするプログラムを示す。ここでセットされる要素はシミュレーション中に変化することはなく、一度だけ実行すれば良い。次に表 2-4 で時間経過とともに変化する要素をセットするプログラムを示す。変数名は概ね前出の通りである。

表 2-3　定係数行列のセットアップ―固定部分（VC#）

```
for(int i = 0; i <= 5; i++) {
    C[0, i + 1] = Lamda[i]; C[i + 1, i + 1] = -Lamda[i];
}
```

表 2-4　定係数行列のセットアップ―変動部分（VC#）

```
double rho = base.Rho(t);
double gt = (1.0 - rho) * base.PL;

//Arrangement of df/dy --> C[,]
C[0, 0] = (rho - Beta) / gt;
for(int i = 0; i <= 5; i++) C[i + 1, 0] = Beta * Alpha[i] / gt;
```

時間メッシュ始端 t と終端 t+dt における時間微分値を求め、式(2-30)の右辺を求めるプログラムを表 2-5 に示す。まず、t を用いて GetDyDx を呼び出した結果を仮に yout[]に格納し、引き続いて t+dt を用いて GetDyDx を呼び出した結果を ytmp[]に格納する。これらの時間微分値、及び C[,]と yold[]の積を用いて式(2-30)の右辺を計算し ytmp[]に格納する。

表 2-5　式(2-30)の右辺の計算（VC#）

```
//dy/dt Estimation at Mesh Begin t --> yout[]
GetDyDx(t, yold, yout);
//dy/dt Estimation at Mesh End t + dt --> ytmp[]
GetDyDx(t + dt, yold, ytmp);

//Equation(2-30) Right Hand Side --> ytmp[]
for(int i = 0; i < VAR_SIZE; i++) {
    double tmp = 0.0;
    for(int j = 0; j < VAR_SIZE; j++) tmp += C[i, j] * yold[j];
    ytmp[i] = yold[i] + dt_2 * (yout[i] + ytmp[i] - tmp);
}
```

表 2-6 に、式(2-30)の左辺の係数行列を求め 7 行 7 列の大きさを持つ二次元配列 WkM[,]に格納するプログラムを示す。

表 2-6　式(2-30)の左辺係数の計算（VC#）

```
//[1-hC/2] Equation(2-30) Left Hand Side Coeficient --> WkM[,]
for(int i = 0; i < VAR_SIZE; i++) {
    for(int j = 0; j < VAR_SIZE; j++) WkM[i, j] = -dt_2 * C[i, j];
    WkM[i, i] += 1.0;
}
```

最後に、式(2-31)のとおり WkM[,]の逆行列を左から ytmp[]に掛ければ、メッシュ終端における中性子量が得られる。これは、表 2-7 に示したプログラムのように、LU 分解法により連立方程式の解を求める操作で実現できる。この中で LU は LU 分解の機能を担うオブジェクトで、LU.SetEquation で与えられた WkM[,]を LU 分解した結果を内部に保持する。引き続く LU.Solve で与えた ytmp[]に対して連立方程式を解く操作を行い結果を ytmp[]に上書きする。

表 2-7　メッシュ終端の中性子量の計算（VC#）

```
//[1-hc/2]inv ytmp[] --> ytmp[]
LU.SetEquation(WkM);
LU.Solve(ytmp);

//---------------------------------
for(int i = 0; i < VAR_SIZE; i++) yold[i] = ytmp[i];
t += dt;
```

2.3 シミュレーションにおける時間経過の制御

2.3.1 表示及び操作上の時間の経過

CASIM には、外観を実装する ControlPanel オブジェクト、シミュレーション上の時計である MasterClock オブジェクト、反応度の状態を表す Reactivity オブジェクト、中性子源の状態を表す Source オブジェクトがある。さらに、臨界実験装置の動特性と中性子量の状態を表す Reactor オブジェクトが、シミュレーションの中核として存在する。これらのオブジェクトの関係を図 2-1 に示す。

ControlPanel は外観上 0.25 秒ごとに状態を更新する。操作者が反応度や中性子源の操作を複数回行うと、Reactivity や Source で随時受け付けられるが、計算への反映は 0.25 秒ごとに一括して行われる。

いわゆるリアルタイムのシミュレーションを行う場合には、同じ 0.25 秒だけ動特性方程式の積分を進めて結果を表示する。もし、例えば 1.25 秒の積分を繰り返して表示を更新すれば、時間的に 5 倍に加速したシミュレーションを行うことになる。

ControlPanel は、状態の更新に際して、MasterClock にシミュレーション上の経過時間 T を問い合わせる。MasterClock が実際の時刻と時間的な加速を考慮した T を回答すると、Reactivity と Source にシミュレーションが T まで進行したことを通知し、さらに、Reactor オブジェクトに対して T までシミュレーションを進めて新しい中性子量を回答するように要求する。その結果を用いて表示が更新される。

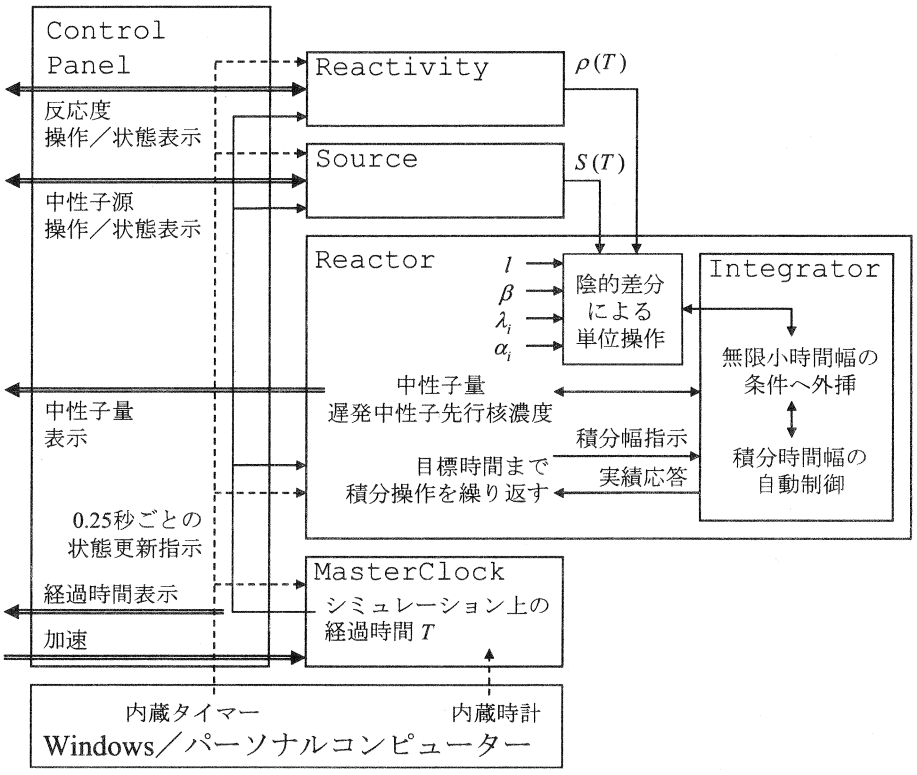

図 2-1　CASIM の構成とデータの流れの概要

2.3.2 シミュレーション上の時間の経過

図 2-2 に示すように、ControlPanel が状態更新する前回と今回のシミュレーション上の経過時間を T_{old} と T_{new} とする。始端 T_{old} と終端 T_{new} の時間幅の間に反応度 ρ や中性子源 S の操作が行われた効果は、まず、一括して終端の $\rho(T_{new})$ や $S(T_{new})$ の値に反映される。その上で動特性方程式の積分を行う際には、$\rho(T)$ は $\rho(T_{old})$ と $\rho(T_{new})$ の間で、$S(T)$ は $S(T_{old})$ と $S(T_{new})$ の間で直線的に変化する。したがって、この時間幅の間に反応度をプラスする操作とマイナスする操作が行われた場合でも、最終的な反応度の値は積分に反映されるものの、途中の経緯は無視される。これらの $\rho(T)$ や $S(T)$ の値は、Reactor が積分を行うときに、Reactivity や Source から適宜与えられる。

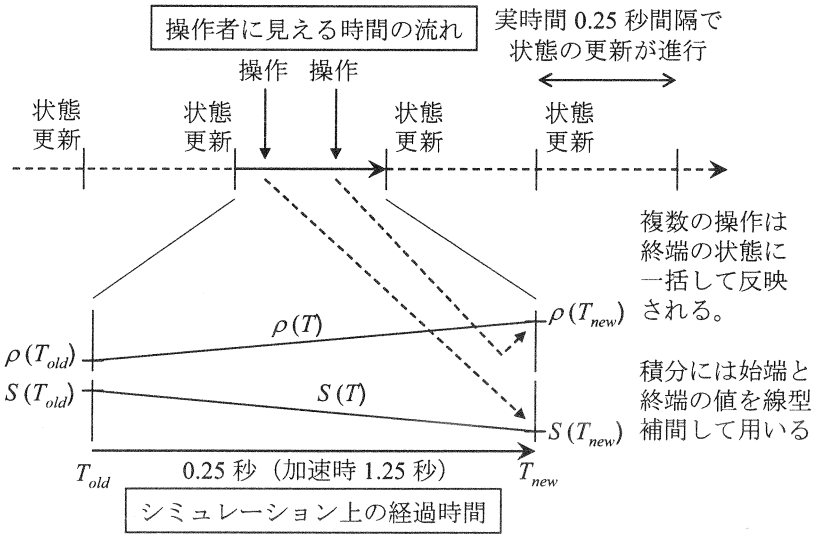

図 2-2　状態更新の間隔とシミュレーション上の経過時間の関係

Reactor は積分を行う Integrator オブジェクトを内蔵する。Integrator は独自に積分の時間幅を制御する機能を持つ。Reactor が Integrator に時間幅 h_{try} の積分を試みるように指示すると、Integrator は最適な時間幅を見つけて積分を行い結果を返すとともに、積分を行った時間幅の実績 h_{did} と次回の積分で推奨される時間幅 h_{next} も報告する。

h_{try} で指示した時間幅の間に急激な変化が起きた場合には、h_{did} は h_{try} よりもずっと小さくなることがある。おそらく h_{next} も小さい。Reactor は h_{next} を次の h_{try} としながら Integrator への指示を繰り返し、始端 T_{old} から終端 T_{new} までの積分を完遂する。

逆に緩やかな変化しか見られないときには、h_{try} で指示したとおりの時間幅で積分が行われ、h_{did} は h_{try} と等しくなり、h_{next} は h_{did} よりも大きな値となる。h_{next} を h_{try} として積分を繰り返していれば、次第に積分の時間幅は大きくなる。そのうち、h_{next} が終端 T_{new} よりもはみ出すことになるが、その場合は終端 T_{new} に適合するように切り詰めた積分を Integrator に指示する。Reactor は ControlPanel から受ける積分の指示と指示の

間でも h_{next} の値を受け継ぐように保持しているので、変化が緩やかな積分が続くと h_{next} はさらに大きくなり、ついには、始端 T_{old} と終端 T_{new} の時間幅を超える。このような場合には、Reactor が ControlPanel から受ける指示と、Reactor が Integrator に与える指示では、積分の時間幅は等しくなる。

2.3.3 積分の時間幅の制御

Reactor が内蔵する Integrator には Bulirsch-Stoer 法が実装されている。この手法では、「**2.2.1 陰的解法の必要性と補外による精度向上**」で述べたように、積分結果を刻み幅が無限小の条件へ外挿する。このために、要求された積分の時間幅 h_{try} をさらに細かく分割し、分割数を変えて何通りかの積分を行い、これらの結果を分割数が無限大の条件へ外挿して最終的な結果とする。分割数は、2、6、10、14、22、34、50、70 の 8 通りあり、少ない分割数から積分の試行を始め、外挿結果の収束を監視しつつ順次多い分割数へ進む。積分の時間幅を調整する仕組みは、概ね以下のとおりである。

予め指定した精度まで収束したならば試行を終了し、分割数の進行の具合と収束の度合いから変化の緩やかさを判定する。h_{try} に比べて変化が十分に緩やかであれば、実際に積分を行った時間幅 h_{did} として h_{try} と同じ値を、次回推奨される積分の時間幅 h_{next} としてより大きな値を Reactor に報告する。h_{next} を大きくする度合いは計算結果に応じて変化するが、極めて緩やかな場合には h_{next} は最大で h_{did} の 10 倍となる

収束しなかった場合は、h_{try} に比べて変化が激しい、つまり、より短い時間幅で積分を行うべきものと判断し、Integrator は自ら h_{try} の値を小さくし、分割数を 2 に戻して上述の試行をやり直す。h_{try} の値を小さくする度合いも計算結果に応じて変化するが、最大で 10^{-5} まで小さくなることがある。収束を見たならば、結果的に積分に成功した時間幅を h_{did} として Reactor に報告する。同時に報告される h_{next} は、h_{did} よりは大きくなるものの、h_{try} に比べると小さいことが多い。

今回、Bulirsch-Stoer 法をシミュレーターに適用しているので、どれだけ大きな h_{next} が推奨値として報告されても、次回の積分について h_{try} を一定値以上に大きくすることができない。しかし、シミュレーターとしての制約がなく、単純に長い時間にわたって緩やかに変化する一点炉動特性方程式の積分を行うのであれば、「**2.2 数値解法**」で述べた陰的解法とあいまって、即発中性子寿命 l に比べてずっと大きな時間幅の積分、つまり大変少ない計算量で積分を安定して行うことができる。実際に積分を行った結果を図 2-3 に示す。遅発中性子割合 β が 0.738%、0.369% 及び 0.1845% の 3 種類の場合について、臨界状態で 10 秒経過したところで +0.0738 %Δk のステップ状の反応度を加えたときの中性子量の時間変化を求めた。初期状態の中性子量は 1 に規格化されている。

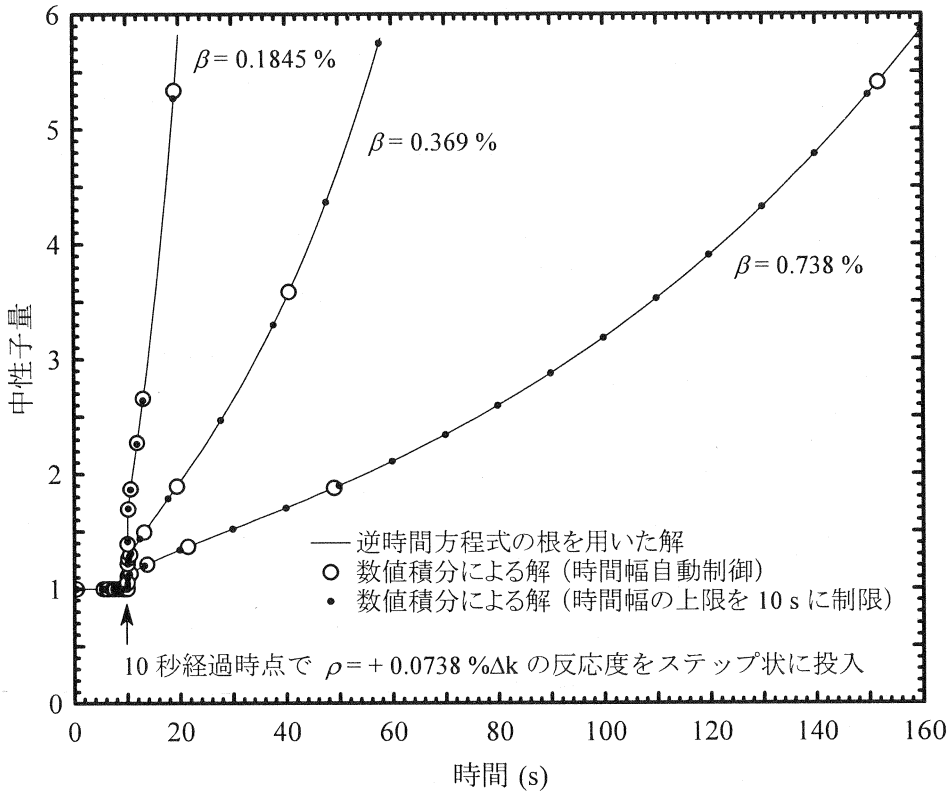

図 2-3　ステップ状の反応度投入に対する数値積分の結果

白丸で示す結果は、積分する時間幅を完全に自動で調整したものであり、反応度添加直後の即発中性子による速い変化が落ち着き、遅発中性子による緩やかな変化が支配的になると、積分の時間幅が大きくなることがわかる。小さな点は時間幅の最大値を 10 秒に制限した場合の結果を示す。いずれの場合も、臨界状態から積分の時間幅 0.5 秒で計算を開始たところ、次のステップでは時間幅が自動的に 5 秒に拡大し、時間 $t = 0.5$ s から $t = 5.5$ s にわたって積分が行われた。時間幅を完全に自動で調整した場合には、次のステップでさらに 10 倍の 50 秒の時間幅で $t = 5.5$ s から $t = 55.5$ s までの積分を試しているが、そこで反応度投入に遭遇して時間幅を短くしている。時間幅の上限を 10 秒に制限した場合についても、$t = 5.5$ s から $t = 15.5$ s までの 10 秒間の積分を試みたところやはり反応度投入に遭遇して時間幅を再調整している。このように、時間幅を縮小する際の元の試行の幅が 50 秒と 10 秒で異なるので、縮小結果は必ずしも同じにはならない。図 2-3 の $\beta = 0.1845\%$ の場合、積分の時間幅は最高でも 10 秒に達していないのに、白丸と黒い小さな点が一致していないのは、このためである。

反応度操作の直後は、即発中性子の影響により短時間ではあるが比較的激しい変化が見られる。Bulirsch-Stoer 法により、そのような場合でも積分の時間幅を短くしてその局面を乗り切り、変化が緩やかになれば時間幅を元に戻す操作が自動的に行える。図 2-4 に計算例を示す。これは、図 2-3 の 9.9 秒から 10.3 秒の部分を拡大したもので、ただし、

遅発中性子割合 $\beta = 0.1845\%$ の計算結果は省いている。臨界状態から積分の時間幅 0.5 秒で計算を開始したところ、次の時間幅は自動的に 5 秒に拡大した。さらに時間幅を拡大して積分を試みたところ 10 秒経過時点の反応度投入に遭遇したので、時間幅を自動的に縮小して積分をやりなおしている。ステップ状の反応度投入であることと、即発中性子の影響で速い変化があることから、10 秒経過前後の積分の時間幅は特に小さくなっている。その後、即発中性子の影響が小さくなると、急速に積分の時間幅は回復する。

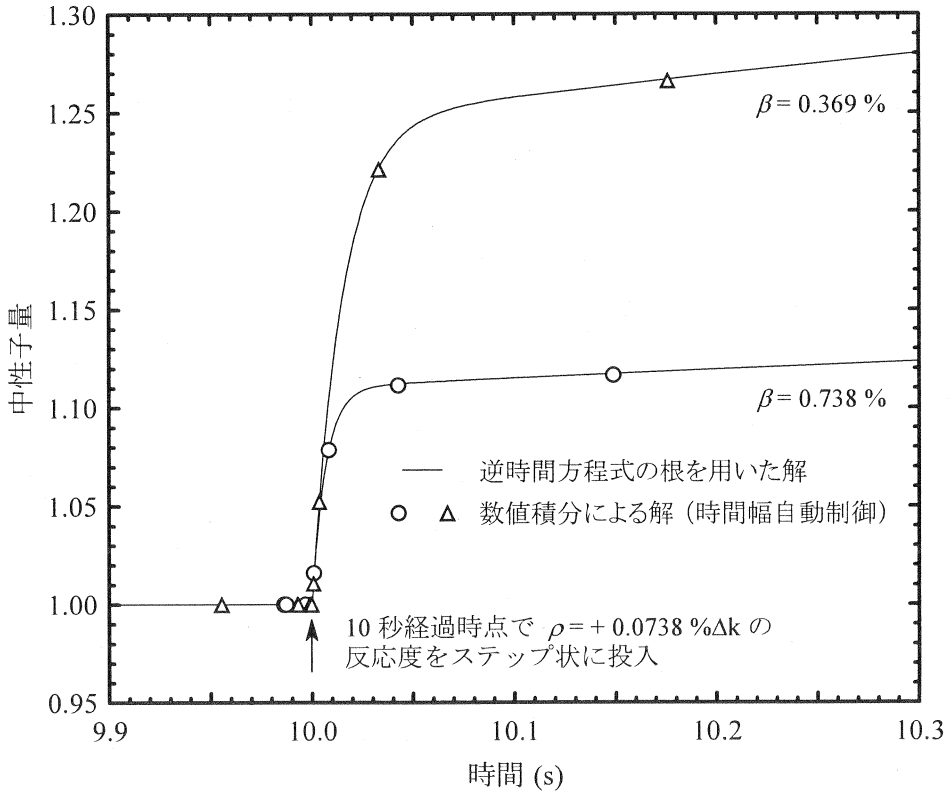

図 2-4　ステップ状の反応度投入に対する数値積分の結果（拡大図）

このように、遅発中性子の振舞いが支配的な緩やかな変化を大きな時間幅で積分する場合も、即発中性子が支配的な速い時間変化を小さな時間幅で積分する場合も、数値積分の結果は逆時間方程式の根を用いた半解析的な解とよく一致しており、ここまで述べた積分の手法はシミュレーターに十分適用できるものである。

3. 操　作

3.1 プログラムのインストールと起動

CASIM は、一つの実行ファイル（exe ファイル）と複数のライブラリー（dll ファイル）からなる。必要なファイルの一覧と実行ファイルを基準にしたディレクトリー構造を付録 B に示すが、これらのファイルを Windows® が実行されているパーソナルコンピューターの任意のディレクトリーにコピーするだけでインストールできる。CASIM の実行には Windows® に.Net Framework（2.0 以上）がインストールされている必要がある。要件の詳細は付録 C に示す。

Windows® の画面上で実行ファイルをダブルクリックすることによって CASIM は起動するが、デフォルトでは図 3-1 に示すように日本語の表示が用いられる。実行ファイルのショートカットを別途作成して、ショートカットをダブルクリックすることによっても CASIM を起動できる。

英語表示を用いることもできるが、そのオプションの指定方法は付録 D に示す。

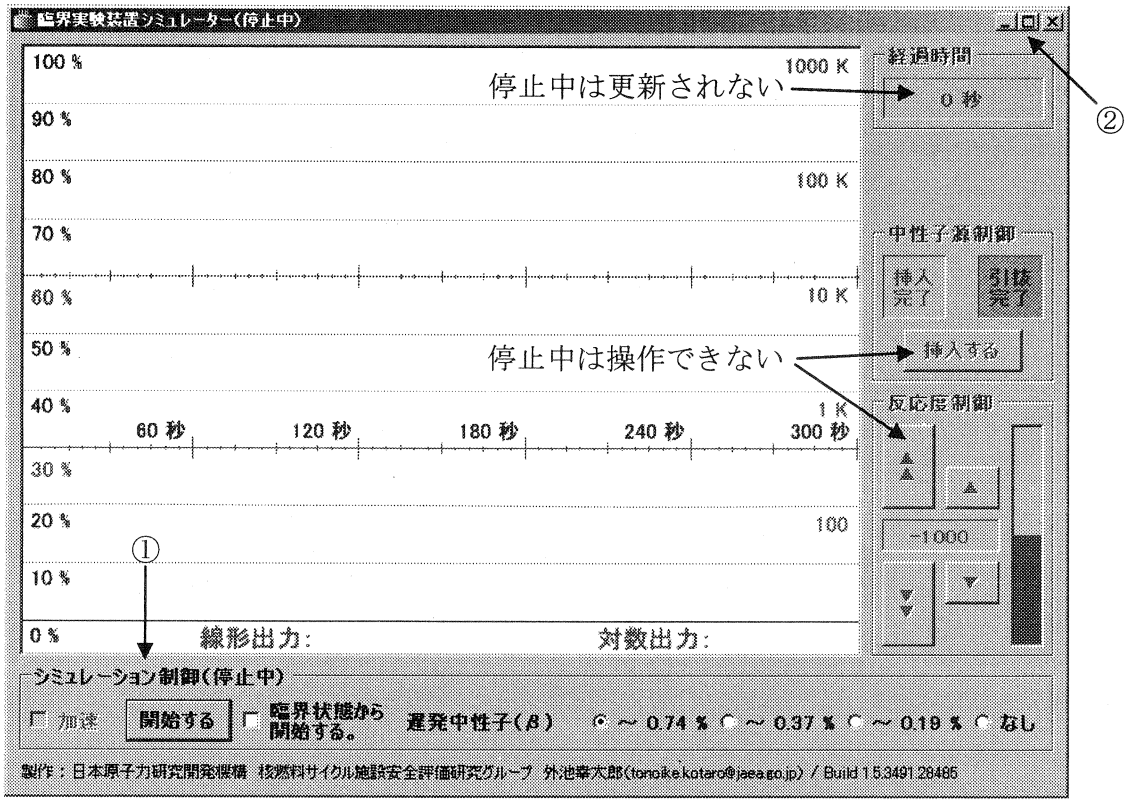

図 3-1　CASIM の起動直後の画面

3.2 シミュレーションの設定と操作

シミュレーションの設定と操作は、「シミュレーション制御」フレーム（図 3-1 の①）の中で行う。シミュレーションが開始され実行中の間は「シミュレーション制御」の文字の後に「実行中」と表示される。プログラム起動直後はシミュレーションは行われておらず「停止中」と表示される。この「実行中」「停止中」の表示は、シミュレーションの動作状態を示すものであって、シミュレーション対象の臨界実験装置（以下、単に「臨界実験装置」と表記する。）の運転状態（起動中や運転終了）を示すものではないので、注意すること。

このフレームより上の画面は、臨界実験装置の操作ボタンや状態表示等であり、シミュレーションが「実行中」のときにのみ操作が可能で中性子量や経過時間の表示も更新される。この操作方法は「**3.3 臨界実験装置の操作と表示**」で説明する。「停止中」は一切の操作ができない。

画面のサイズは、「停止中」は、最小化の状態、通常の状態（800×600 ピクセルに固定）、最大化の状態の 3 段階でのみ変更が可能である（図 3-1 の②）。「実行中」は、通常と最大化の状態間の変更ができない。最大化あるいは通常の状態でシミュレーションを開始したら、「実行中」は最小化することと元に戻す操作のみが行える。

3.2.1 遅発中性子割合 β の選択

操作に先立って遅発中性子割合 β を選択する。画面上では図 3-2 に示すように「～ 0.74%」、「～ 0.37%」、「～ 0.19%」及び「なし」から選択できるように表示されている。0.74% と 0.37% は、それぞれ、ウラン燃料とプルトニウム燃料を想定した値である。

0.19% は、現実には存在しない値だが、遅発中性子割合が小さくなると反応度操作に対して中性子量の変化が鋭敏になり調整が難しくなる様子を体験できる仮想的なものである。

「遅発中性子なし」は、さらに極端な設定で、もし遅発中性子がなく即発中性子のみが存在する場合は臨界近傍では核分裂連鎖反応を手動で制御することができないことを体験させるためのものである。「**2.1.2 基礎的な動特性方程式の例**」の式 (2-1) に基づく。

図 3-2　遅発中性子割合を選択する画面

3.2.2 シミュレーションの開始・終了及び加速

➢ 開始・終了

図 3-3 に示す「開始する」のボタンを押すとシミュレーションが始まる。シミュレーション実行中は、図 3-4 に示すように、ボタンの表示は「停止する」に変化する。

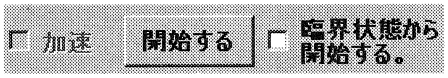

図 3-3　シミュレーションの開始操作を行う画面

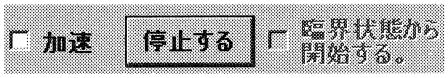

図 3-4　シミュレーションの停止操作を行う画面

「臨界状態から開始する。」のチェックは、通常は外れており、シミュレーションが図 3-5 に示す未臨界状態から始まる。具体的には、反応度が - 0.74 %Δk（中性子実効増倍率が約 0.993）で、かつ、中性子源が挿入された状態である。中性子源の強度は、中性子量が 12.4（最低レンジ「100」の 12.4%）となるように調整されている。

「臨界状態から開始する。」にチェックをした上でシミュレーションを開始すると、図 3-6 に示すように、反応度ゼロで中性子源が引き抜かれた臨界状態からシミュレーションが始まる。中性子量は 10.0（最低レンジ「100」の 10.0 %）に予め調整される。

図 3-4 に示す「停止する」のボタンを押すとシミュレーションは停止する。一度停止すると、次は新規に開始することしかできない。（「再開」の機能はない。）

➢ 加速

シミュレーション実行中に図 3-4 に示す「加速」にチェックを入れると、シミュレーションの時間の流れが 5 倍速くなる。これにより、遅発中性子の変化が落ち着くまでの待ち時間を短縮できる。（教育的な観点から使用が是か非かは十分に検討されたい。）

チェックを外せば通常に戻るほか、中性子源や反応度の操作をした場合も自動的に元に戻る。

➢ 画面サイズの変更

シミュレーション停止中であれば、図 3-1 に示す②のボタンで、画面を「最大化」したり「標準」に戻したりサイズを変更できる。開始した後（実行中）は図 3-5 や図 3-6 に示すように「最小化」のみ可能である。

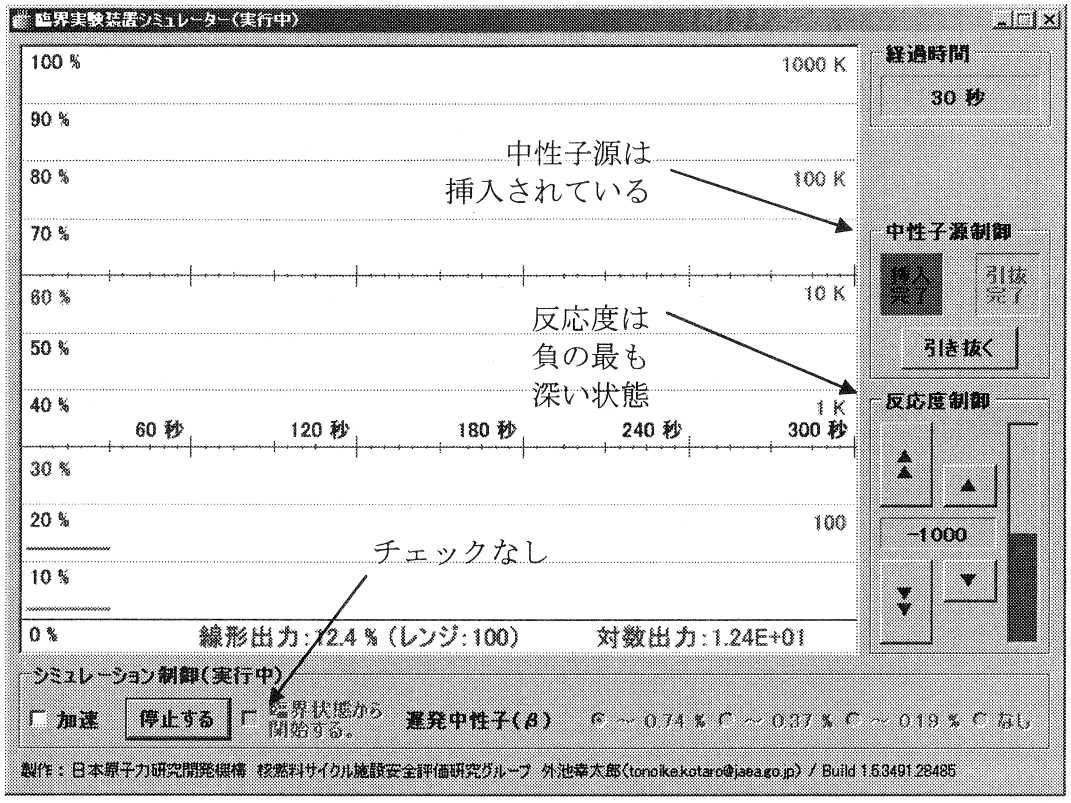

図 3-5　シミュレーション開始直後（未臨界状態）の画面

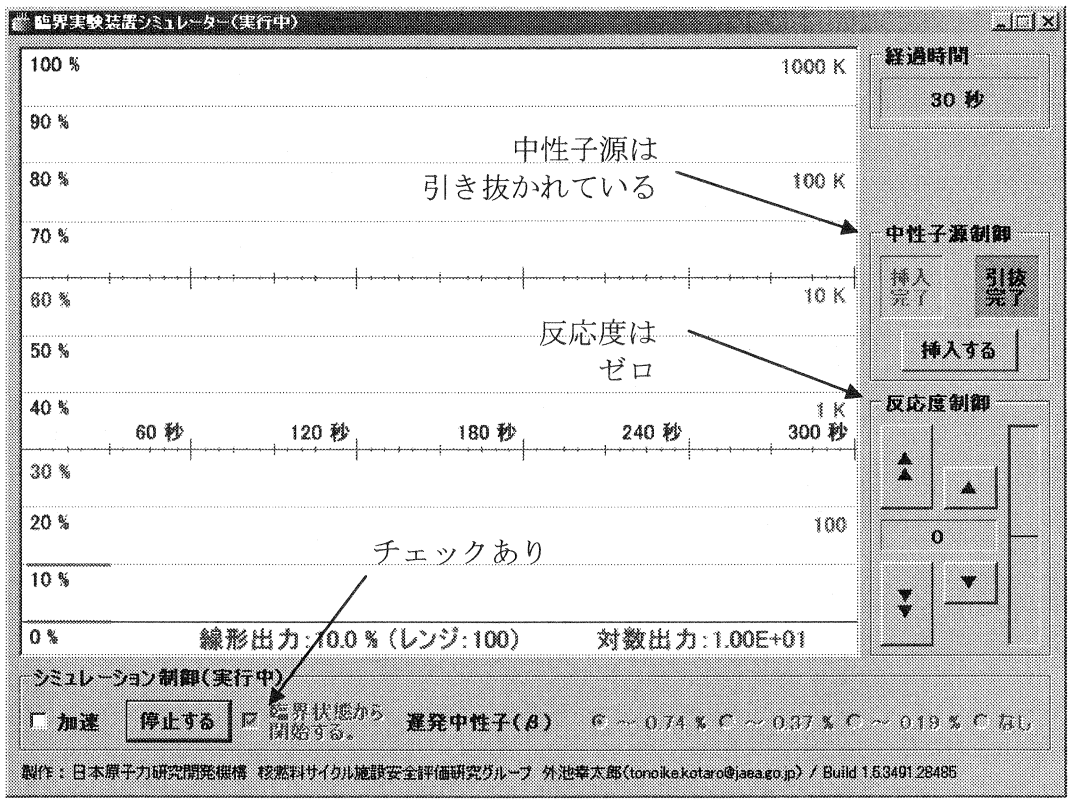

図 3-6　シミュレーション開始直後（臨界状態）の画面

3.3 臨界実験装置の操作と表示

3.3.1 中性子量（出力）の表示

画面中央（図 3-7）には中性子量の時間変化が線型目盛りと対数目盛りでプロットされる。CASIM の画面上では中性子量を「出力」と表現しており、本報告でも以降、これに倣う。これは、シミュレーターという性質上、実際の臨界実験装置の運転で多く用いられる表現の方が良いと判断したためである。また、「**1. はじめに**」と「**2.2.3 一点炉動特性方程式への適用**」で示したように、CASIM では反応度フィードバックは存在しないと仮定しているので、温度条件は無関係であり、絶対出力も考慮されていない。さらに、「**2.1.1 核分裂連鎖反応の基礎と近似によるモデル導出の概要**」で述べたように中性子量と出力は単純に比例していると考えて良い。

したがって、ここでは中性子量と出力は同義と考えてよく、CASIM では縦軸の出力の単位は任意であり特に単位記号は付していない。

➢ 対数出力

対数目盛りは画面の右端に付されており、下端が「10」、上端が「1000 K」で、画面上に 5 桁幅にわたって出力の変化を表示できる。ここで、「K」は 1000 倍を表す。デジタル値は「対数出力：」のタイトルの後に指数形式で表示される。プロットとデジタル表示は青色である。

➢ 線型出力

線型目盛りは画面の左端に付されているとおり、下端が「0 %」、上端が「100 %」で画面上で高々 1 桁幅の出力の変化しか表示できない。このため、レンジ切り替え機能を有している。

レンジは約半桁幅ごとに 9 つ、「100」、「300」、「1 K」、「3 K」、「10 K」、「30 K」、「100 K」、「300 K」、及び「1000 K」が用意されている。これらのレンジ名称は、そのレンジ表示における出力の最大値（100%値）を用いている。したがって、レンジ名称の値に出力（%）を乗じれば、対数出力と等しくなる。

プロットは赤線で描かれ、出力が上昇中に 72% を超えると、又は下降中に 19% を下回ると自動的にレンジが切り替わる。デジタル値は「線型出力：」のタイトルのあとに、% 形式とレンジ名称を組み合わせて表示される。

図 3-7 では、「100」レンジから始まって、3 回のレンジ切り替えを経て最終的に「3 K」レンジに達している。最後の時点で線型出力は 27.0% なので、これに 3000 を乗じれば対数出力と等しい 8.10×10^2 を得る。

➢ 時間軸

表示の横幅は 300 秒分に相当する。シミュレーション開始時には、図 3-5 や図 3-6 に示すように、画面の左端からプロットが始まり右へ延びていく。プロットが右端に達すると、古いプロットは順次左へスクロールし、最新の出力は右端に追加して描かれる。

線型出力 30% と 60% のレベルには、5 秒刻みの目盛りを有する横軸を設けてあるので簡易の倍増時間測定も可能である。図 3-7 は一定反応度 + 10.0 ¢ による出力の上昇の様子を示す。約 198 秒時点で 30% に達し約 266 秒時点で 60% に達しているため、倍増時間は約 68 秒と読み取ることができる。これは炉周期に換算すると 98 秒であり、反応度とよく一致している。（付録 A の図 A-2 参照。）

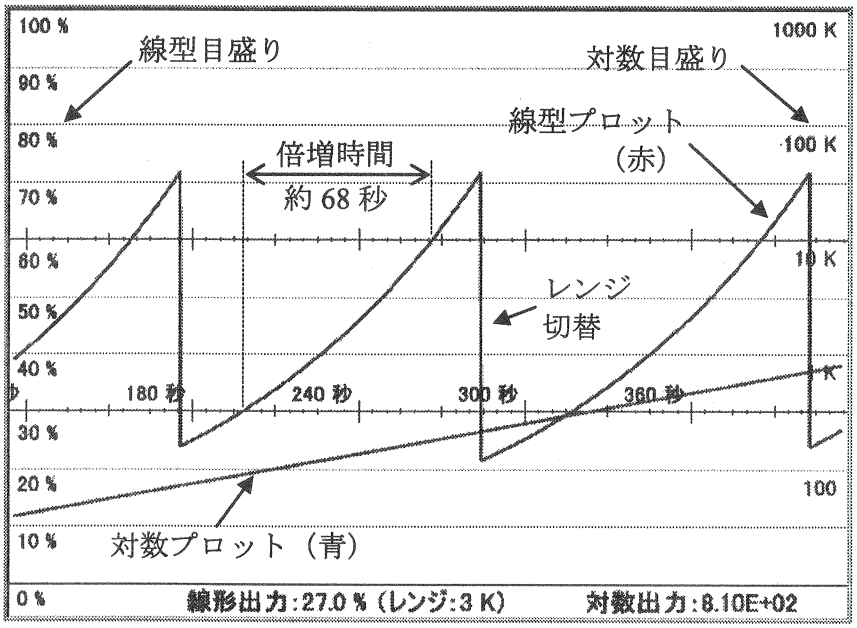

図 3-7　一定反応度（+ 10.0 ¢）による出力の上昇

➢ 出力の表示範囲からの逸脱（シミュレーションの自動中止）

出力が 1×10^6 を超えると、対数出力の青色プロットは画面上端の 1000 K を超え、線型出力の赤色プロットも 1000 K レンジの 100% 表示を超える。したがって双方のプロットが画面上端から逸脱するものの、この直後にはシミュレーションは続行される。しかし、出力が 2×10^6（線型出力で 1000 K レンジの 200% 相当）に達すると、図 3-8 に示す警告を発して自動的にシミュレーションの内部の計算を停止する。この表示が出ても CASIM のプログラム自体は正常に動作しており、「OK」ボタンを押せばシミュレーションが停止している状態（図 3-1、図 3-3）に復旧するので、シミュレーションを新規に開始することができる。

出力の低下については、表示範囲から逸脱してもシミュレーションは停止しない。

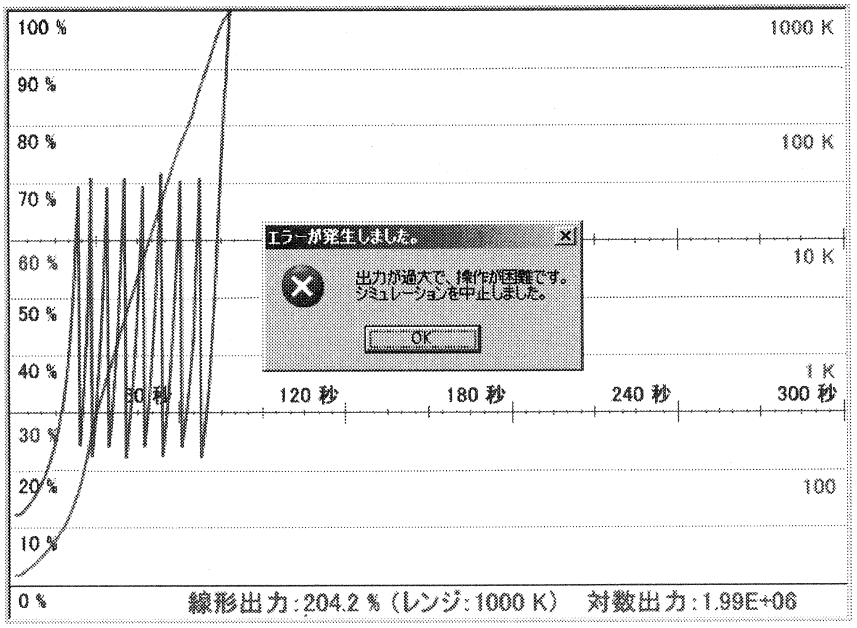

図 3-8　出力過大の警告

3.3.2 中性子源の操作

中性子源制御パネルを操作することにより、中性子源の炉心への完全な挿入と完全な引き抜きの操作が行える。途中で停止することはできない。図 3-9 に示すように「挿入完了」の赤表示が点灯しているときは、中性子源は炉心に完全に挿入されている。「引抜完了」の緑表示が点灯しているときは、中性子源は炉心から完全に取り除かれており、つまり、炉心は中性子源の影響をまったく受けていない。

挿入と引き抜きの操作をするボタンは一つだけであり、「挿入完了」の状態では「引き抜く」と表示される。この時ボタンを押すと「挿入完了」が消灯し、中性子源の引き抜きが始まる。引き抜き動作中は、ボタンには黄色表示の点滅と引き抜き方向の矢印が現れる。引き抜き動作が終了すると「引抜完了」が点灯し、ボタンには「挿入する」と表示される。再度ボタンを押すと逆の動作で中性子源が挿入される。

これらの挿入と引き抜きの動作はいずれも約 5 秒を要し、この間の炉心に近い約半分の行程で、位置に応じた強さで炉心に中性子源が影響する。

キーボードで「Ctrl」＋「N」キー操作を行うと、中性子源の挿入・引き抜き操作のボタンを画面上でクリックするのと同じ動作となる。

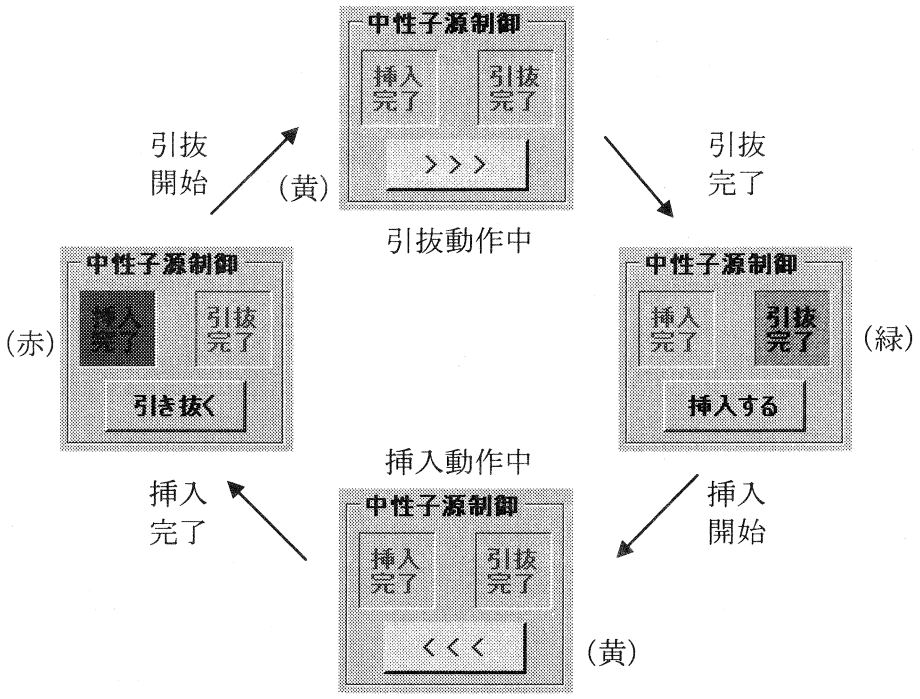

図 3-9　中性子源制御パネルの操作と状態表示

3.3.3 反応度の操作

反応度制御パネルを用いて、ステップワイズな反応度操作を行える。1 ステップの反応度価値は 0.00074 %Δk に固定されている。反応度の負の側に 1000 ステップ、正の側に 500 ステップの操作の幅がある。遅発中性子割合 β を 0.74 % とした場合、1 ステップが 0.1 ¢ になるが、β を 0.37% 又は 0.19% にすると、1 ステップの反応度価値は 0.2 ¢ 又は 0.4 ¢ となる。

操作には図 3-10 に示す粗調整ボタンと微調整ボタンを用いる。粗調整ボタンでは約 10 ステップずつ、微調整ボタンでは正確に 1 ステップずつ操作できる。「約」としてあるのは、粗調整ボタンは ±3 ステップの範囲内でランダムに狂いが出るためである。このボタンを押したまま保持しても 1 回の反応度操作しか発生せず、連続的な反応度操作はできない。反応度操作量を増やすためには繰り返しボタンを押す必要がある。

反応度操作のステップ数は中央部にデジタル表示される。また、右端には反応度のグラフ表示があり、反応度が正の場合は赤色（図 3-10）で、負の場合は濃緑色（図 3-11 左）で棒が表示される。ただし、このグラフの表示範囲は ±100 ステップ（± 0.074 %Δk）分しかない。

図 3-11 の右側に示すように、デジタル表示がゼロで、グラフ表示の棒が消えて線で表示されているときが反応度ゼロの状態で臨界となる。

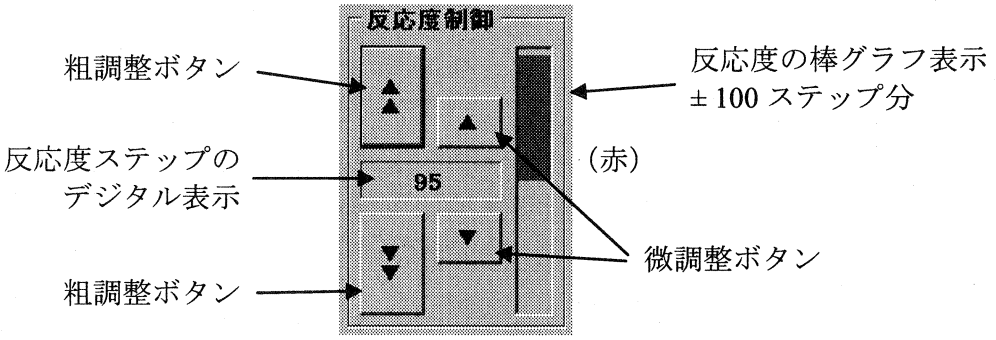

図 3-10　反応度制御パネル（反応度+ 95 ステップ×0.00074 %Δk）

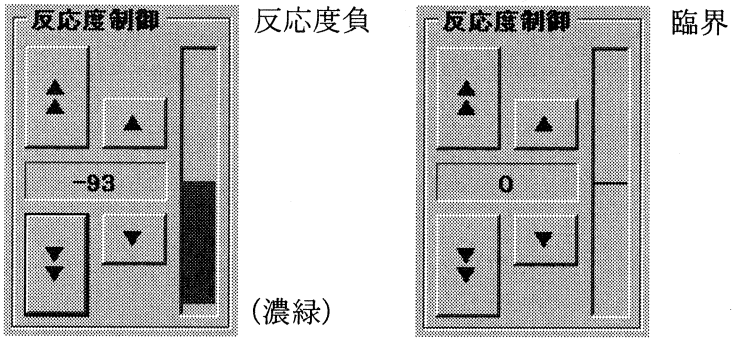

図 3-11　反応度制御パネルの状態表示
（左：反応度- 93 ステップ×0.00074 %Δk ／ 右：臨界の場合）

キーボードでも反応度の操作は可能である。微調整はカーソル移動の「↑」キーと「↓」キーによる。粗調整には「Ctrl」＋「↑」キーと「Ctrl」＋「↓」キーを用いる。

加えて、負の反応度を 1000 ステップ分一気に投入することも、キーボードの「Ctrl」＋「Shift」＋「S」キーにより可能である。これは、ロッドドロップ実験、あるいはスクラム操作に相当し、キーボード操作により反応度制御パネルは図 3-12 に示す状態となる。このとき、中性子実効増倍率は約 0.993 である。

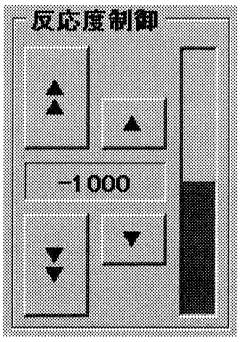

図 3-12　反応度制御パネルの状態表示
（負の反応度が最も大きい - 1000 ステップ×0.00074 %Δk）

4. まとめ

ここまでに述べた CASIM の原理及び機能に基づき、臨界実験装置を未臨界状態から起動し、臨界を達成して維持し、負の反応度を投入して停止するまでの一連のシミュレーションを行った結果を図 4-1 に示す。

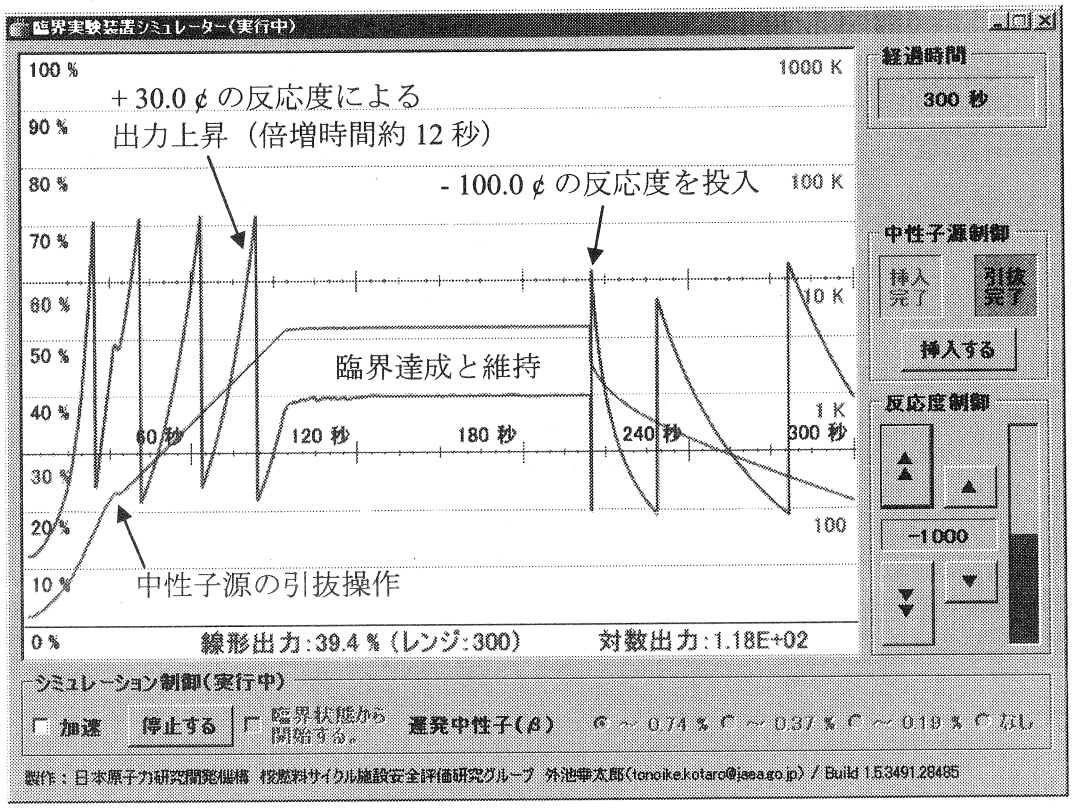

図 4-1　未臨界状態からの起動、臨界維持、停止の一連の操作結果

遅発中性子割合 $\beta = 0.74\%$ を選択している。中性子源が挿入された未臨界状態からシミュレーションを開始し、反応度を速やかに+ 300 ステップ（+ 0.222 %Δk = + 30.0 ¢）まで投入し、出力を上昇させている。中性子源を引き抜いた後は、対数表示で直線的に出力が上昇しており、安定した原子炉ペリオド（倍増時間で約 12 秒）が観察できる。約 100 秒が経過した時点で出力を一定とするために臨界状態へ調整を開始した。このとき出力は「10 K」レンジの約 40% である。約 205 秒経過時点まで臨界を維持した後、- 1000 ステップ（- 0.74 %Δk = - 100.0 ¢）の反応度を一気に投入して臨界実験装置を停止させている。なお、300 秒間の表示枠の中で一連の操作を行うために出力上昇時の反応度は相当に大きくしている。実際の臨界実験装置ではこのような急速な出力上昇は通常行われないので注意されたい。

ここに示した以外にも、学習のための基本的な操作として、中性子源を挿入したまま反応度を増やす操作と出力の漸近値の読み取りを繰り返し、逆増倍曲線を作成し臨界点の予測を行うことができる。また、臨界状態から正の反応度を投入して出力の倍増時間を測定することができ、反応度と原子炉ペリオド（倍増時間）の関係を学ぶこともできる。これらの定量性のある操作だけでなく、臨界に近い状態で反応度操作に応じて臨界実験装置の出力がどのように時間変化するか、操作と応答を十分なリアリティーを持って体感することができる。

このような臨界実験装置のシミュレーターを標準的なパーソナルコンピューターで実行できるプログラムとして開発し、配布できるように整備した。原子力技術の初学者である学生や技術者による原子炉物理学の学習、臨界実験装置の運転員の操作の基礎訓練、一般公衆による臨界実験装置の運転の体験などに広く利用されることを希望する。

参考までに、以下に開発の端緒と経緯を紹介しておく。

このシミュレーターの核となる部分は別の目的で作成されたものである。過渡臨界実験装置 TRACY [13] で臨界事故を模擬した実験を行う際に、反応度を加える操作と出力ピークが現れるタイミングについて簡易に検討できる動特性解析コードを整備する提案がありプログラムを試作したことが端緒となっている。核分裂連鎖反応で発生するエネルギーによる燃料の温度上昇と反応度へのフィードバックを考慮しつつ、第 2 章で述べた数値積分の問題に取り組んだ。その結果、積分の時間幅の自動調整を行いつつ、TRACY の実験開始直後に現れる最初の出力ピークの形状を再現できることが確認できた。このとき、プログラミング言語は Visual Basic® を用い、ActiveX® DLL のライブラリーとして整備した。これを Excel® から呼び出し、計算パラメーターの入力と計算結果の出力をスプレッドシート上で行った。

このシステムは、その後、JCO 臨界事故 [14] においても、事故発生直後の最初の出力ピークの大きさについて目安を得るために用いられた。大型計算機やエンジニアリングワークステーションを用いることなく、Windows® が稼動するパーソナルコンピューターだけで対話形式で解析が行えた。

その後、「1. **はじめに**」に述べたような、一般公衆向けの講演会用に簡単なシミュレーターを整備する提案があり、既に作成済みの計算プログラムにリアルタイムで動作するグラフィカルユーザーインターフェイスを新たに付け加えることにより、CASIM を整備した。この際、.Net Framework を利用するために、まず、Visual Basic®.NET を用いた。さらに、本報告書をまとめるにあたって、C++ に類似した.Net Framework 向けの言語である Visual C#® で書き直した。

謝　辞

CASIM は、関西光科学研究所で開催されているスーパーサイエンスセミナー（S-Cube）において、（財）原子力安全研究協会理事長の松浦祥次郎氏が「原子炉を運転できるわけ ― 有難い自然の仕組み ―」（第 140 回、2007 年 10 月 26 日）と題してご講演になった中で用いられた。CASIM の開発は、特に遅発中性子割合 β の値を仮想的な小さな値あるいはゼロにする設定は、同氏のご発案がきっかけとなっている。

同セミナーの事務局を担当された関西光科学研究所の西村昭彦氏と、松浦氏のご発案を著者に紹介して頂いた原子力研修センターの杉本純氏には、CASIM の開発着手から講演会における使用までの間、様々なご助言・ご示唆を頂いた。

原子炉動特性の数値解法に著者が取り組むことになった端緒は、「**4. まとめ**」にも記したが、安全試験施設管理部の柳澤宏司氏が過渡臨界実験装置 TRACY の実験開始当初に提起された議論による。著者がパーソナルコンピューター用のグラフィカルなプログラムの開発を手がけるようになったきっかけも、同氏に開発用ソフトウェアを紹介して頂いたことによる。

CASIM の動作確認は安全試験施設管理部の井澤一彦氏に、本報告の査読は安全試験施設管理部の曽野浩樹氏に担当して頂き、多くの有用なコメントを頂いた。

これらの方々に、この場を借りて、深く御礼申し上げる。

参考文献

1) ”京都大学臨界集合体実験装置（KUCA）大学院実験テキスト”、京都大学原子炉実験所
2) 仁科浩二郎、”KUCA による大学院生合同実験授業の経験”、日本原子力学会誌, **24**(11), p.865-872 (1982).
3) 伊藤哲夫、”実践的な実習教育で体系的理解に成果―近大炉による体験型実習研修会―”、エネルギーレビュー, 9 月号, p.16-19 (2009).
4) 原子力安全基盤機構、”原子力施設運転管理年報 平成 20 年版（平成 19 年度実績）”、p.259-279 (2008).
5) M. Kitamura, T. Ohi, T. Yamamoto and K. Akagi, "Development of High Precision Plant Simulator for Pressurized Water Reactor Plants using Distributed Architecture," *J. Nucl. Sci. Technol.*, **36**(4), p.344-357 (1999).
6) ジームス J. ドゥデルスタット、ルイス J. ハミルトン著、成田正邦、藤田文行訳、”原子炉の理論と解析”、現代工学社、ISBN4-87472-080-3.
7) G. R. Keepin, T. F. Wimett and R. K. Zeigler, "Delayed Neutrons from Fissionable Isotopes of Uranium, Plutonium and Thorium," LA-2118, USAEC (1957).
8) G. R. Keepin, "Physics of Nuclear Kinetics," Addison-Wesley (1965).
9) K. Tonoike, Y. Miyoshi, T. Kikuchi and T. Yamamoto, "Kinetic Parameter $\beta_{\mathrm{eff}}/\ell$ Measurement on Low Enriched Uranyl Nitrate Solution with Single Unit Cores (600ϕ, 280T, 800ϕ) of STACY," *J. Nucl. Sci. Technol.*, **39**(11), p.1227-1236 (2002).
10) William H. Press、Brian P. Flannery、Saul A. Teukolsky、William T. Vetterling 著、丹慶勝市、奥村晴彦、佐藤俊郎、小林誠訳、”NUMERICAL RECIPES in C”、技術評論社、ISBN4-87408-560-1.
11) William H. Press, Brian P. Flannery, Saul A. Teukolsky, William T. Vetterling, "NUMERICAL RECIPES — The Art of Scientific Computing (Third Edition)," CAMBRIDGE UNIVERSITY PRESS, ISBN-10: 0521880688.
12) ”プログラミング言語 C#”、JISX3015:2008（ISO/IEC 23270:2006）.
13) H. Yanagisawa *et al.*, "Experiments on transient behavior of a low-enriched uranyl nitrate solution system with TRACY to study hypothetical criticality accidents in reprocessing plants," *Proc. 6th Int. Conf. Nuclear Criticality Safety (ICNC-99)*, Versailles, France, **2**, p.900 (1999).
14) K. Tonoike, T. Nakamura, Y. Yamane and Y. Miyoshi, "Power Profile Evaluation of the JCO Precipitation Vessel Based on Record of the Gamma-ray Monitor", *Nucl. Technol.*, **143**(3), p.364-372 (2003).

付録

A. 逆時間方程式の数値解法

A.1 基本的な考え方

逆時間方程式は、時定数の逆数 s (s^{-1}) と反応度 ρ を動特性パラメーターを用いて関係づけるものであり、次のように表現される。

$$\rho = \frac{sl}{sl+1} + \frac{1}{sl+1}\sum_{i=1}^{6}\left(\frac{s\beta_i}{s+\lambda_i}\right) \qquad \text{(A-1)}$$

ここで、l は即発中性子寿命 (s)、β_i はグループ i の遅発中性子割合、λ_i はグループ i の遅発中性子先行核の崩壊定数 (s^{-1}) である。ある反応度 ρ を与えたとき、この方程式には複数の根 s が存在するが、その中でも最も値が大きく正の値にもなり得る根が原子炉の安定ペリオドとなり重要である。

図 A-1 によれば、根 s が存在する範囲は、l や λ_i を用いて示すことができる。例えば、最も小さい根 s_7 は、$-1/l$ と $-\lambda_6$ の間にある。また、s_2 は $-\lambda_2$ と $-\lambda_1$ の間である。したがって、これらの解の存在する範囲を利用して解を探索できる見込みがある。

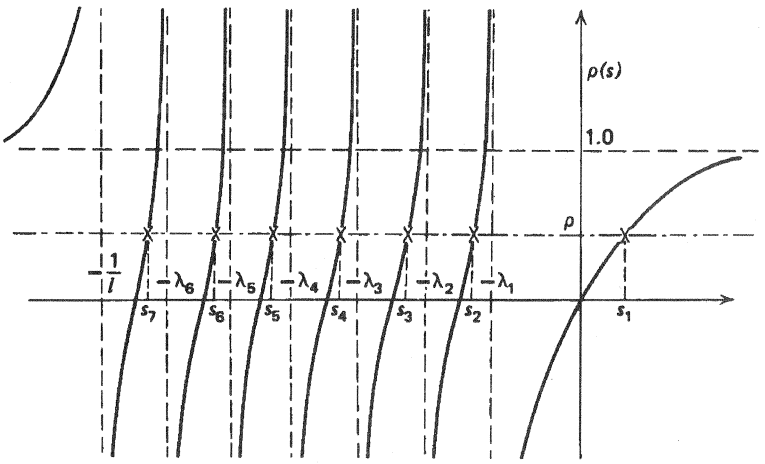

図 A-1　逆時間方程式の根

しかし、図 A-1 を見てわかるように、式 (A-1) のままでは $-1/l$ や $-\lambda_i$ は極であり取り扱いにくい。そこで、式 (A-1) を通分して多項式に変形し極を持たない形とする。この変形により 7 次方程式になるが、実数の根が 7 個あることは既知なので比較的見通しの良い問題となる。

s_2 から s_7 までの 6 個の根は、存在範囲を $-1/l$ や $-\lambda_i$ を用いて決めれば、それぞれの範囲の中で根は一つしかないので、二分探索法で粗く求めることが可能である。その上で、隣り合う根を 2 つずつまとめて Bairstow 法により根の精度を高めるとともに、多項式を 2 次ずつ減次する。この結果、最後の根で最も重要な s_1 は 1 次方程式を解くことで得られる。

A.2 プログラム構成の概要

数値的に逆時間方程式を解くプログラムの例を表 A-1 と表 A-2 に示す。これらはすべてプログラミング言語 Visual C#® (VC#) で記述されており、それぞれの表が VC# のひとつのソースファイルに相当する。2 つのファイルをひとつのプロジェクトでコンパイルし、後述の数値計算ライブラリと共に用いれば Windows® パソコンのコマンドプロンプトで実行できる。

表 A-1 はメインプログラムである。主な機能は、起動されたときに渡されるコマンドライン引数を解釈し、プログラムの流れを制御する。

表 A-2 には計算のための Parameters と NumericServices の 2 つのクラスが含まれる。Parameters クラスは主に l、β_i、λ_i のような動特性パラメーターを保管し取り扱う。根を求める計算を呼び出す機能もあるが、実際の計算の機能は持たない。NumericServices クラスは、前述の逆時間方程式を多項式へ変形する機能（DevelopEquation メソッド）と、二分探索と Bairstow 法により 7 次方程式の根を求める機能（SolveEquation メソッド）を持つ。逆時間方程式を解くための独特な工夫はすべてこれらのメソッドに含まれている。さらに、GetAmplitude メソッドでは、中性子量の時間変化 $N(t)$ を 7 つの根を用いて

$$N(t)=\sum_{j=1}^{7}A_j e^{s_j t} \tag{A-2}$$

と表した際の各振幅 A_j を Gauss-Jordan 法により計算する。

二分探索法と Bairstow 法は別途整備した汎用の数値計算ライブラリから提供される MathRFandNLSE クラスを用いている。Gauss-Jordan 法は同じライブラリから提供される GaussJ クラスを用いている。

A.3 逆時間方程式の多項式への変形

式 (A-1) の両辺に $sl+1$ を乗ずると、

$$(sl+1)\rho = sl + \sum_{i=1}^{6}\left(\frac{s\beta_i}{s+\lambda_i}\right) \tag{A-3}$$

となる。さらに、総和記号の中の分数を通分するために両辺に

$$\prod_{i=1}^{6}(s+\lambda_i) \tag{A-4}$$

を乗じると、

$$(sl+1)\prod_{i=1}^{6}(s+\lambda_i)\,\rho = sl\prod_{i=1}^{6}(s+\lambda_i) + \sum_{i=1}^{6}s\beta_i \prod_{j=1,j\neq i}^{6}(s+\lambda_j) \tag{A-5}$$

となる。根 s に着目すると左辺は 7 次である。右辺第 1 項も 7 次である。右辺第 2 項の中の総乗記号は、1 から 6 の j のうち i と等しくないもの 5 つを乗じることを示している。したがって総乗記号の部分だけで 5 次であり、右辺第 2 項は全体では 6 次である。

NumericServices クラスの DevelopEquation メソッド（表 A-2）では、式 (A-4) を

展開した 6 次多項式の係数を sV[]に格納している。この配列の長さは 7 であり、sV[0]が定数項、sV[1]が 1 次項の係数と続き、sV[6]が 6 次項の係数でこの場合の値は 1.0 である。この多項式の格納方法は他の多項式にも共通である。式 (A-5) 右辺第 2 項の総乗記号の部分は、6 種類の 5 次多項式となるが、それぞれの係数を sW[0~5][]に格納する。

以上から、式 (A-5) の左辺は、

sV[]に (sl+1) を乗じた 7 次式

であり、右辺は、

sV[]に sl を乗じた 7 次式、

sW[0]に $s\beta_1$ を乗じた 6 次式、sW[1]に $s\beta_2$ を乗じた 6 次式、

sW[2]に $s\beta_3$ を乗じた 6 次式、sW[3]に $s\beta_4$ を乗じた 6 次式、

sW[4]に $s\beta_5$ を乗じた 6 次式、及び sW[5]に $s\beta_6$ を乗じた 6 次式の和

となっている。この右辺を左辺に移項したものが多項式の方程式であり、その係数は z[]に格納される。

A.4 多項式方程式の根の計算

前節で得られた多項式を表す配列 z[]は NumericServices クラスの SolveEquation メソッドに渡され、7 つの根が計算される。同時に即発中性子寿命 pl、遅発中性子先行核の崩壊定数 lamda[]も渡され、根の推定に用いられる。このとき、図 A-1 の λ_6 が lamda[5]に対応することに始まり、λ_1 が lamda[0]に対応するように値が格納されていることが前提となる。このことは、後で述べる動特性パラメーターの入力ファイルを作成する際に重要となる。

7 つの根は s[]に格納される。最初に $-1/l$ と $-\lambda_6$ の間にある解を MathRFandNLSE クラスの RtBis メソッドが提供する二分探索法により粗く推定し s[6]に置く。同様に $-\lambda_6$、$-\lambda_5$、$-\lambda_4$、$-\lambda_3$、$-\lambda_2$、$-\lambda_1$ のそれぞれの間にある解の推定値を s[5]から s[1]に置く。この際、推定の精度は λ_i の値に応じて変化させる。

その後、s[1]と s[2]、s[3]と s[4]、s[5]と s[6]をそれぞれペアにして根の改善を行う。2 つの根は GetTwoRoots メソッドで 2 次方程式の係数に変換された上で、MathRFandNLSE クラスの QRoot メソッドに渡される。この QRoot メソッドでは Bairstow 法により 2 次方程式の係数の形で根の改善が行われる。その後 GetTwoRoots メソッドで再び 2 つの根に戻されるとともに、元の 7 次方程式を順に 2 次ずつ減次する。このため、s[5]と s[6]のペアの改善を行う際には 3 次方程式になっており、最後に 1 次方程式が残る。

7 つ目の根が、最初に述べたように、原子炉の安定ペリオドを決定する重要なものであるが、1 次方程式を解いて得たものを s[0]に格納して戻る。これは図 A-1 の s_1 に相当する。

A.5 成分ごとの振幅の計算

前節で得られた 7 つの根 s_i を用いると、中性子量の時間変化は式 (A-2) で表すことができるが、さらに各項の振幅 A_i を求めなければならない。ここでは、初期条件として $t=0$ の直前まで反応度 ρ はゼロで、中性子量と遅発中性子先行核濃度の時間変化もゼロであると仮定する。また、$t=0$ においてステップ状の反応度変化があり、以後反応度は一定とする。

中性子量の $t=0$ における初期値を 1 に規格化した場合、

$$\sum_{j=1}^{7} A_j = 0 \tag{A-6}$$

となる。

一点炉動特性方程式のうち遅発中性子先行核の関係

$$\frac{dC_i}{dt} = \frac{\beta_i}{\Lambda} N(t) - \lambda_i C_i(t) \quad i=1,\ldots,6 \tag{A-7}$$

に関しても式 (A-2) の形で解を記述できるので

$$C_i(t) = \sum_{j=1}^{7} C_{ij} e^{s_j t} \quad i=1,\ldots,6 \tag{A-8}$$

を仮定する。式 (A-8) を式 (A-7) に代入すると

$$\sum_{j=1}^{7} C_{ij} s_j e^{s_j t} = \frac{\beta_i}{\Lambda} \sum_{j=1}^{7} A_j e^{s_j t} - \lambda_i \sum_{j=1}^{7} C_{ij} e^{s_j t} \quad i=1,\ldots,6 \tag{A-9}$$

を得る。この式は任意の $t\,(>0)$ について成立するので各 $j\,(=1,\ldots,7)$ に対して

$$C_{ij} = \frac{\beta_i}{\Lambda(s_j + \lambda_i)} A_j \quad i=1,\ldots,6 \tag{A-10}$$

でなければならない。

遅発中性子先行核について $t=0$ において $dC_i/dt = 0$ なので式 (A-7) と式 (A-8) から

$$C_i = \sum_{j=1}^{7} C_{ij} = \frac{\beta_i}{\lambda_i \Lambda} \quad i=1,\ldots,6 \tag{A-11}$$

である。これに式 (A-10) を代入すると

$$\frac{1}{\lambda_i} = \sum_{j=1}^{7} \frac{1}{(s_j + \lambda_i)} A_j \quad i=1,\ldots,6 \tag{A-12}$$

でなければならない。なお、式 (A-7) で世代時間 Λ を特に定義せずに導入したが、この代入においてキャンセルされている。

式 (A-6) と式 (A-12) を整理して行列で表現すると以下のとおりとなる。

$$\begin{pmatrix} 1 & 1 & 1 & 1 \\ \frac{\lambda_1}{s_1+\lambda_1} & \frac{\lambda_1}{s_2+\lambda_1} & \cdots & \frac{\lambda_1}{s_7+\lambda_1} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\lambda_6}{s_1+\lambda_6} & \cdots & \cdots & \frac{\lambda_6}{s_7+\lambda_6} \end{pmatrix} \begin{pmatrix} A_1 \\ A_2 \\ \vdots \\ A_7 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix} \tag{A-13}$$

表 A-2 の GetAmplitude メソッドでは、数値計算ライブラリから提供される GaussJ クラスの Solve メソッドを呼び出してこの連立方程式を解き、A_i を求めている。

表 A-1　逆時間方程式を数値的に解くプログラム―メインプログラム（VC#）

```
using System;
using System.IO;
using System.Text;
using System.Reflection;
using con = System.Console;
using par = InhourEquation.Parameters;
using num = InhourEquation.NumericServices;
using res = InhourEquation.Properties.Resources;

namespace InhourEquation {
    class Program {
        const string PARAM_FILE = "param.txt";
        const string RESTXT_PARA = "InhourEquation.Resource.paramtmp.txt";
        const string RESTXT_HELP = "InhourEquation.Resource.help.txt";

        static Assembly _Me = Assembly.GetExecutingAssembly();

        static void Main(string[] args) {
            double dol = 0.0, dolend = 0.0, dolstep = -1.0;
            try {
                switch(args.Length) {
                case 0:
                    dol = dolstep = 0.02;
                    dolend = 3.0 + dolstep * 0.5;
                    break;
                case 1:
                    if(string.Compare(args[0], "/defpara", true) == 0) {
                        TypeDefaultParameters(); return;
                    }
                    if(string.Compare(args[0], "/?", true) == 0) {
                        TypeHelp(); return;
                    }
                    if(string.Compare(args[0], "/help", true) == 0) {
                        TypeHelp(); return;
                    }
                    if(!double.TryParse(args[0], out dol)) {
                        con.WriteLine(res.ErrArg);
                        con.WriteLine(res.ArgCantRho);
                        return;
                    };
                    break;
                case 2:
                    if(!double.TryParse(args[1], out dolstep)) {
                        con.WriteLine(res.ErrArg);
                        con.WriteLine(res.ArgCnatRhoStep);
                        return;
                    }
                    dol = dolstep;
```

```
            if(!double.TryParse(args[0], out dolend)) {
                con.WriteLine(res.ErrArg);
                con.WriteLine(res.ArgCantRhoEnd);
                return;
            }
            dolend += dolstep * 0.5;
            break;
        default:
            con.WriteLine(res.ErrArg);
            con.WriteLine(res.ArgTooMany);
            return;
        }
    } catch(Exception ex) {
        con.WriteLine(res.ErrArg);
        con.WriteLine(ex.Message);
        return;
    }
    if(File.Exists(PARAM_FILE))
        if(!ReadParameter(PARAM_FILE)) return;

    if(dolstep < 0.0) {
        ListOmegas(dol);
    } else {
        ListReactorPeriods(dol, dolend, dolstep);
    }
}
private static bool ReadParameter(string path) {
    bool retcode = false;
    StreamReader sr;
    try {
        sr = new StreamReader(path);
    } catch(Exception ex) {
        con.WriteLine(res.ErrOpen);
        con.WriteLine(ex.Message);
        return retcode;
    }
    try {
        if(!double.TryParse(GetOneLine(sr), out par.Pl)) {
            con.WriteLine(res.ErrFile);
            con.WriteLine(res.CantPL);
            sr.Close(); return false;
        }
        if(!double.TryParse(GetOneLine(sr), out par.BetaEff)) {
            con.WriteLine(res.ErrFile);
            con.WriteLine(res.CantBeta);
            sr.Close(); return false;
        }
        for(int i = 0; i < 6; i++) {
            if(!double.TryParse(GetOneLine(sr), out par.Alpha[i])) {
                con.WriteLine(res.ErrFile);
                con.WriteLine(res.CantAlpha);
                sr.Close(); return false;
            }
        }
        for(int i = 0; i < 6; i++) {
            if(!double.TryParse(GetOneLine(sr), out par.Lamda[i])) {
                con.WriteLine(res.ErrFile);
                con.WriteLine(res.CantLamda);
                sr.Close(); return false;
            }
        }
        sr.Close(); return true;
    } catch(Exception ex) {
        con.WriteLine(res.ErrFileIO);
        con.WriteLine(ex.Message);
        sr.Close(); return false;
    }
}
private static string GetOneLine(StreamReader sr) {
    for(; ; ) {
        string oneline = sr.ReadLine();
        if(oneline == null) return null;
        if(oneline.StartsWith("//")) continue;
        if(oneline.Trim().Length == 0) continue;
        return oneline;
    }
```

```
        }
        private static void ListOmegas(double dol) {
            con.WriteLine("i,Amp(i),S(i)");
            par.GetSandAmp(dol * par.BetaEff);
            for(int i = 0; i < 7; i++)
                con.WriteLine(string.Format("{0},{1},{2}", i + 1, par.Amp[i], par.S[i]));
        }
        private static void ListReactorPeriods(double dolstart, double dolend,
                                               double dolstep) {
            con.WriteLine("Rho($),Period(s)");
            for(double rho = dolstart; rho < dolend; rho += dolstep) {
                par.GetS(rho * par.BetaEff);
                con.WriteLine(string.Format("{0},{1}", rho, 1.0 / par.S[0]));
            }
        }
        private static void TypeDefaultParameters() {
            StreamReader sr
                = new StreamReader(_Me.GetManifestResourceStream(RESTXT_PARA));
            while(!sr.EndOfStream) {
                string oneline = sr.ReadLine();
                switch(oneline) {
                case "$$PL$$":
                    con.WriteLine(par.NEUTRON_LIFETIME); break;
                case "$$BETA$$":
                    con.WriteLine(par.BETA_EFF); break;
                case "$$ALPHA$$":
                    con.WriteLine(par.ALPHA_1);
                    con.WriteLine(par.ALPHA_2);
                    con.WriteLine(par.ALPHA_3);
                    con.WriteLine(par.ALPHA_4);
                    con.WriteLine(par.ALPHA_5);
                    con.WriteLine(par.ALPHA_6); break;
                case "$$LAMDA$$":
                    con.WriteLine(par.LAMDA_1);
                    con.WriteLine(par.LAMDA_2);
                    con.WriteLine(par.LAMDA_3);
                    con.WriteLine(par.LAMDA_4);
                    con.WriteLine(par.LAMDA_5);
                    con.WriteLine(par.LAMDA_6); break;
                default:
                    con.WriteLine(oneline); break;
                }
            }
            sr.Close();
        }
        private static void TypeHelp() {
            StreamReader sr
                = new StreamReader(_Me.GetManifestResourceStream(RESTXT_HELP));
            while(!sr.EndOfStream) {
                string oneline = sr.ReadLine();
                if(!oneline.StartsWith("---------+")) con.WriteLine(oneline);
            }
            sr.Close();
        }
    }
}
```

表 A-2　逆時間方程式を数値的に解くプログラム―計算クラス（VC#）

```
using System;
using sm = System.Math;
using par = InhourEquation.Parameters;
using num = InhourEquation.NumericServices;
using rf = JAEA.Tonoike.Math.MathRFandNLSE;
using ev = JAEA.Tonoike.Math.EvalFunc;
using gj = JAEA.Tonoike.Math.SolutionLAE.GaussJ;
using fx = JAEA.Tonoike.Math.FunctionOfX;

namespace InhourEquation {
    static class Parameters {
        ////////////////////
        //Default values//
        ////////////////////

        //Prompt neutron life time (s)
        public const double NEUTRON_LIFETIME = 0.0000454;

        //Delayed neutron fractions
        public const double BETA_EFF = 0.00738;

        //Relative yields of delayed neutron precursors
        public const double ALPHA_1 = 0.033;
        public const double ALPHA_2 = 0.219;
        public const double ALPHA_3 = 0.196;
        public const double ALPHA_4 = 0.395;
        public const double ALPHA_5 = 0.115;
        public const double ALPHA_6 = 0.042;

        //Decay constants of delayed neutron precursors (/s)
        public const double LAMDA_1 = 0.0124;
        public const double LAMDA_2 = 0.0305;
        public const double LAMDA_3 = 0.111;
        public const double LAMDA_4 = 0.301;
        public const double LAMDA_5 = 1.11;
        public const double LAMDA_6 = 3.01;

        ////////////////////////
        //Parameter storages//
        ////////////////////////

        public static double BetaEff = BETA_EFF;
        public static double[] Alpha = { ALPHA_1, ALPHA_2, ALPHA_3,
                                  ALPHA_4, ALPHA_5, ALPHA_6 };
        public static double[] Lamda = { LAMDA_1, LAMDA_2, LAMDA_3,
                                  LAMDA_4, LAMDA_5, LAMDA_6 };
        public static double Pl = NEUTRON_LIFETIME;

        ////////////////////////
        //Solution storages//
        ////////////////////////

        //Inverse of time constant (/s)
        public static double[] S = new double[7];
        //Amplitude of each exponential component
        public static double[] Amp = new double[7];

        public static void GetS(double rho) {
            double[] Beta = new double[6];
            for(int i = 0; i < 6; i++) Beta[i] = BetaEff * Alpha[i];

            // Develop a polynomial equation representing the inhour equation.
            // The polynomial of 7th degree will be in Z[]

            double[] Z = new double[8];
            num.DevelopEquation(Lamda, Beta, Pl, rho, Z);

            // Solve the equation Z[]=0.  Pl and Lamda[] are provided to give
            // hints for the initial guess of roots.  Calculated will be in S[].

            num.SolveEquation(Pl, Lamda, Z, S);
        }
```

```
    public static void GetSandAmp(double rho) {
        GetS(rho);
        num.GetAmplitude(Lamda, S, Amp);
    }
}
static class NumericServices {
    public static void DevelopEquation(double[] lamda, double[] beta, double pl,
                                       double rho, double[] Z) {
        // A polynomial of 6th degree : V
        // (s + Lamda0)(s + Lamda1)...(s + Lamda5)

        double[] V = new double[7];
        for(int i = 0; i < lamda.Length; i++) V[i] = lamda[i];
        ev.PolDevlp(V);

        // 6 polynomials of 5th degree : sW[i][6] i = 0 to 5
        // (s + Lamda0)(s + Lamda1)...(s + Lamda5) / (s + Lamda i)

        double[][] W = new double[6][];
        for(int i = 0; i <= 5; i++) {
            W[i] = new double[6];
            for(int j = 0; j < i; j++) W[i][j] = lamda[j];
            for(int j = i + 1; j < 6; j++) W[i][j - 1] = lamda[j];
            ev.PolDevlp(W[i]);
        }

        // A polynomial of 7th degree : Z
        // z[7]s^7 + z[6]s^6 + z[5]s^5 + z[4]s^4
        //         + z[3]s^3 + z[2]s^2 + z[1]s + z[0] = 0

        double z7 = pl * rho - pl;
        Z[7] = z7;
        for(int k = 6; k >= 1; k--) {
            Z[k] = z7 * V[k - 1] + rho * V[k];
            for(int i = 0; i <= 5; i++) Z[k] -= beta[i] * W[i][k - 1];
        }
        Z[0] = rho * V[0];
    }

    public static void SolveEquation(double pl, double[] lamda,
                                     double[] Z, double[] s) {
        // Search roots roughly by the bisection method.  Accuracy requirement
        // is not so high and varies depending on the lamda values.

        _Z = Z;
        fx vZ = new fx(valZ);
        s[6] = rf.RtBis(vZ, -1 / pl, -lamda[5], lamda[5] / 10.0);
        s[5] = rf.RtBis(vZ, -lamda[5], -lamda[4], lamda[4] / 10.0);
        s[4] = rf.RtBis(vZ, -lamda[4], -lamda[3], lamda[3] / 10.0);
        s[3] = rf.RtBis(vZ, -lamda[3], -lamda[2], lamda[2] / 10.0);
        s[2] = rf.RtBis(vZ, -lamda[2], -lamda[1], lamda[1] / 10.0);
        s[1] = rf.RtBis(vZ, -lamda[1], -lamda[0], lamda[0] / 10.0);

        // Improve roots by the Bairstow's method.  In each step, A couple of
        // roots will be improved with the original polynomial being divided
        // by the quadratic factor determined by the root pair.

        GetTwoRoots(Z, ref s[1], ref s[2]);
        GetTwoRoots(Z, ref s[3], ref s[4]);
        GetTwoRoots(Z, ref s[5], ref s[6]);

        // The last root, finally, determined by two coefficients.

        s[0] = -Z[0] / Z[1];
    }
    private static void GetTwoRoots(double[] Z, ref double s1, ref double s2) {
        const double EX = 0.00000000000000022204460492503131 * 512;

        double b = -s1 - s2;
        double c = s1 * s2;
        rf.QRoot(Z, ref b, ref c, EX);

        double d = sm.Sqrt(b * b - 4.0 * c);
        if(b < 0) d = -d;
        d += b;
        d *= -0.5;
```

```
            s1 = d;
            s2 = c / d;
            if(s1 < s2) { double temp = s1; s1 = s2; s2 = temp; }

            double[] Z2 = new double[Z.Length];
            ev.PolDiv(Z, new double[] { c, b, 1.0 }, Z2, new double[Z.Length]);
            for(int i = 0; i < Z.Length; i++) Z[i] = Z2[i];
        }
        private static double[] _Z;
        private static double valZ(double x) {
            double sum = _Z[0], xm = 1.0;
            for(int i = 1; i < _Z.Length; i++) sum += (xm *= x) * _Z[i];
            return sum;
        }

        public static void GetAmplitude(double[] lamda, double[] s, double[] amp) {
            double[,] P = new double[7, 7], Q = new double[7, 1];

            for(int j = 0; j < 7; j++) P[0, j] = 1.0;
            Q[0, 0] = 1.0;
            for(int i = 1; i < 7; i++) {
                for(int j = 0; j < 7; j++)
                    P[i, j] = lamda[i - 1] / (s[j] + lamda[i - 1]);
                Q[i, 0] = 1.0;
            }
            gj.Solve(P, Q);
            for(int i = 0; i < 7; i++) amp[i] = Q[i, 0];
        }
    }
}
```

A.6 計算結果（炉周期）

図 A-2 に炉周期の計算結果のうち遅発臨界の範囲を図示する。反応度をパラメーターとして変化させ最大の時定数 s_1 を求め、その逆数をプロットしたものである。この計算で用いられた動特性パラメーターは表 A-2 の冒頭に定数として示されている値であり、遅発中性子先行核の相対収率 α_i、同崩壊定数 λ_i は ^{235}U のもので Keepin による。遅発中性子割合 β と即発中性子寿命 l は代表的な臨界実験装置の特性値を参考にした。倍増時間についても併せて示す。

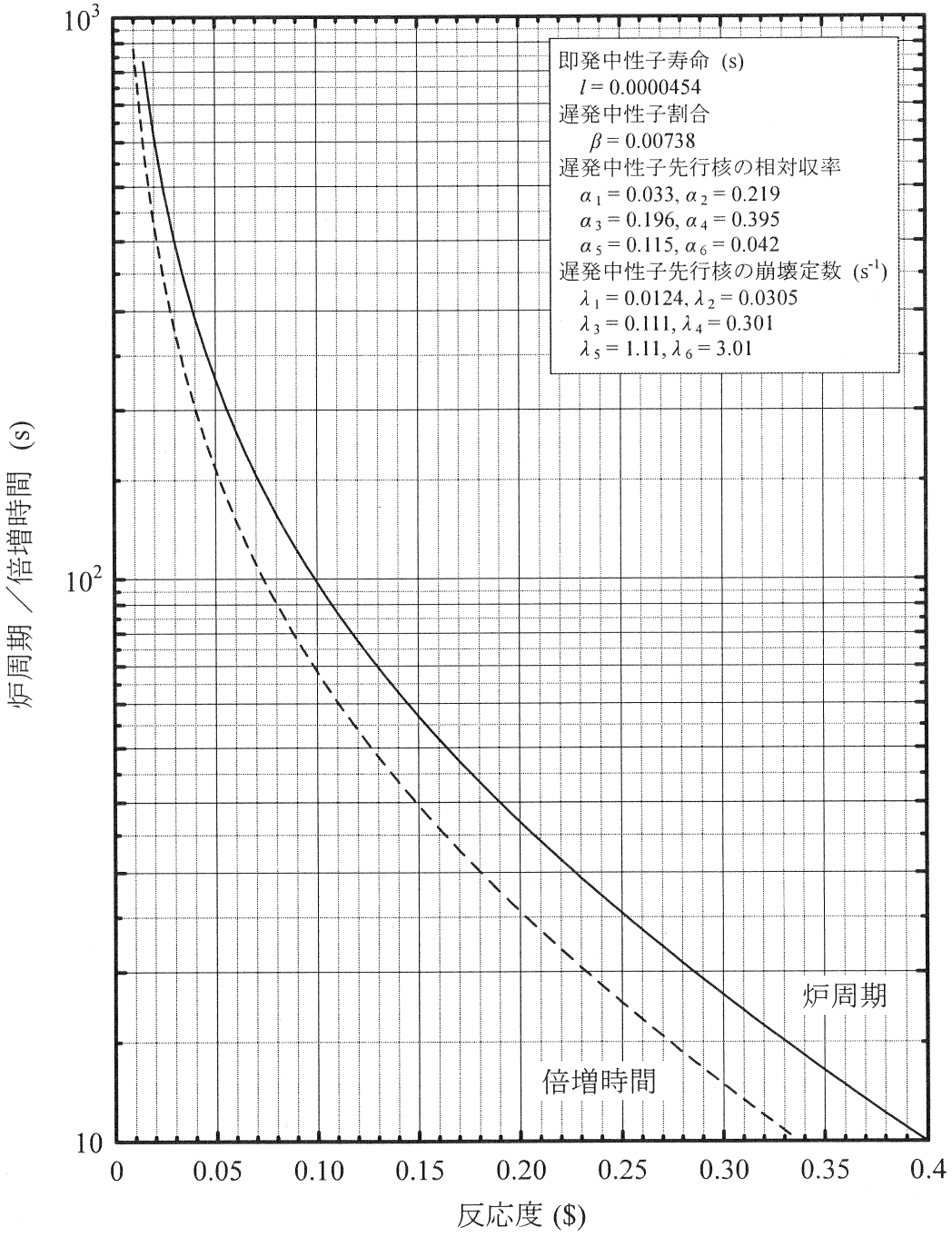

図 A-2　反応度と炉周期／倍増時間の関係（遅発臨界）

即発臨界の範囲の計算結果を図 A-3 に示す。この計算では、即発中性子寿命 l についてもパラメーターとし、10^{-3} (s) から 10^{-8} (s) の範囲で 6 種類の計算を行った。

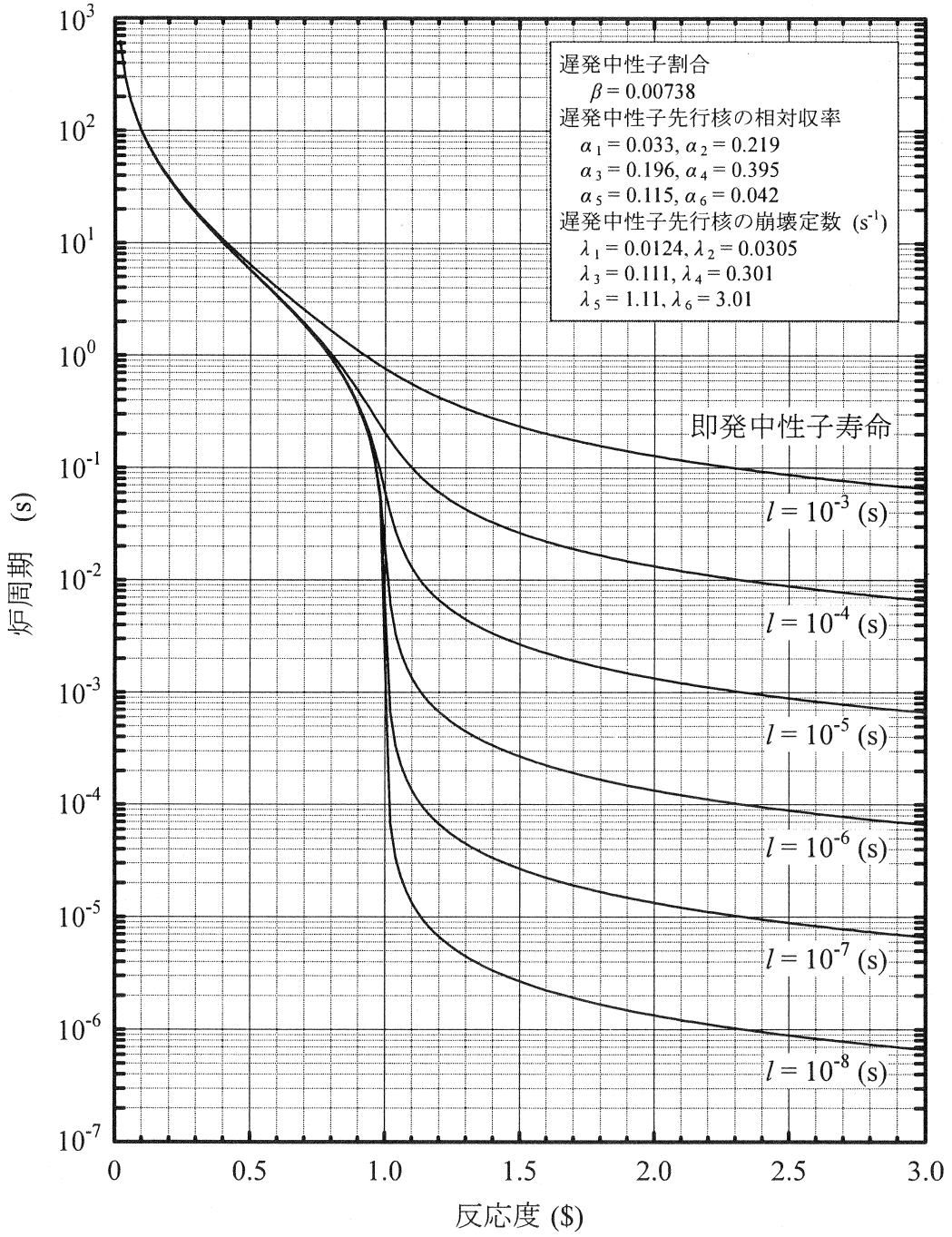

図 A-3　反応度と炉周期の関係（即発臨界）

A.7 計算結果（7 つの根と振幅）

表 A-1 に示したプログラムは、ある一つの数値をコマンドライン変数として与えるとその数値を $ 単位の反応度と解釈し、逆時間方程式の 7 つの根及び成分ごとの振幅も計算して出力する。表 A-3 に計算結果の例を示す。なお用いた定数は図 A-2 に掲げたものと同じである。

表 A-3　様々な反応度における 7 つの根と振幅

反応度	i	根 s_i	振幅 A_i
-10 $	1	-0.01236	0.00342
	2	-0.02984	0.02169
	3	-0.10889	0.01882
	4	-0.29014	0.03418
	5	-1.09850	0.00942
	6	-2.99861	0.00342
	7	-1665.24956	0.90905
- 0.1 $	1	-0.00581	0.72162
	2	-0.01529	0.07995
	3	-0.07316	0.06387
	4	-0.20150	0.03291
	5	-1.00629	0.00816
	6	-2.90544	0.00296
	7	-179.04612	0.09054
+ 0.1 $	1	0.01014	1.28387
	2	-0.01378	-0.02999
	3	-0.06276	-0.08774
	4	-0.18731	-0.03987
	5	-0.98691	-0.01158
	6	-2.88490	-0.00426
	7	-146.85699	-0.11043
+ 1 $	1	7.50854	11.66578
	2	-0.01275	-0.02332
	3	-0.03796	-0.20800
	4	-0.13705	-0.19583
	5	-0.68465	-0.89859
	6	-2.26720	-1.20052
	7	-8.94384	-8.13952
+ 3 $	1	332.67099	1.49820
	2	-0.01253	-0.00978
	3	-0.03283	-0.07408
	4	-0.11892	-0.07178
	5	-0.35736	-0.22033
	6	-1.17670	-0.08818
	7	-3.07650	-0.03404

A.8 プログラムのその他の機能

表 A-1 に示したプログラムは、コマンドライン変数の与え方によって複数種類の動作をする。

一つの数値を与えたときは、その数値は $ 単位の反応度と解釈され、A.7 節に示したように、その反応度に対応した逆時間方程式の 7 つの根 s_i と振幅 A_i を計算し、結果を標準出力へ出力する。

二つの数値を与えたときは、反応度を系統的に変化させながら炉周期 (s) を計算し、結果を標準出力へ出力する。与える最初の数値が反応度変化の上限値 ($)、二つ目の数値が反応度変化のステップ ($) である。計算は反応度が正の範囲（ゼロは含まない。最初の計算は「ゼロ＋ステップ」の反応度による。）で行われる。A.6 節に示したグラフはこの機能で得られたものである。

「/defpara」という文字列を与えたときは、計算は行わず、計算に用いるデフォルトの定数を標準出力へ出力する。その結果を表 A-4 に示す。この出力のフォーマットについては後で再度説明する。

表 A-4 「/defpara」オプションの出力結果

```
//-----------------------------------
//Kinetic Parameters for 'GETPERIOD'
//-----------------------------------

//Prompt neutron life time (sec)

4.54E-05

//Dlayed neutron fraction
//    Beta-eff

0.00738

//Relative yields of delayed neutron precursors in  6 groups
//    Alpha(i) i=1,2,3,4,5,6

0.033
0.219
0.196
0.395
0.115
0.042

//Decay constants of delayed neutron precursors in 6 groups
//    Lamda(i) i=1,2,3,4,5,6 (/sec)

0.0124
0.0305
0.111
0.301
1.11
3.01
```

「/help」または「/?」という文字列を与えたときは、このプログラムの使用方法を標準出力に出力する。

このプログラムは、表 A-4 に示したデフォルトの定数以外に、ユーザーが独自の動特性パラメーターを指定して計算を行うことができる。そのためには、表 A-4 に示すフォーマットに従って動特性パラメーターの値を書き込んだファイル"param.txt"を準備し、実行ファイルと同じディレクトリに置かなければならない。"param.txt"では、「//」で始まる行と空白行は無視される。それ以外に 18 個の数値を 1 行ずつ記述しなければならない。その内訳は、即発中性子寿命 l(s) の値、遅発中性子割合 β の値、遅発中性子先行核の相対収率 α_i の値 6 個、及び遅発中性子先行核の崩壊定数 λ_i の値 6 個である。記述する順序はここに述べた（表 A-4 に掲げた）とおりとする。特に、λ_i については値の小さなものから順に並べるものとし、α_i と λ_i はグループが同じ順序でなければならない。

B. CASIM を構成するファイルと配置

CASIM を実行するために必須のファイルは、実行ファイルの「casim.exe」、及びライブラリーの「casimphy.dll」と「casimmat.dll」である。これらのファイルは同一のディレクトリーに配置しなければならない。

これに加えて、日本語表示や英語表示のためのリソースファイルとして、casim.exe と同じディレクトリーに「ja」と「en」というサブディレクトリーを置き、それぞれのサブディレクトリーの中に「casim.resources.dll」を置く。「ja」と「en」の中に同じ名前のファイルが置かれることになるが、内容は異なるので注意すること。

表 B-1 にこれらのファイルの一覧をディレクトリー構造とともに示す。これらのファイルとディレクトリーの名称、配置のディレクトリー構造は変更してはならない。しかし、全体を置くディレクトリーは、パーソナルコンピューターのローカルディスク上であれば任意である。(ネットワークディスク上に置いた場合はデフォルトの設定では起動に失敗する。セキュリティー設定を変更すれば起動可能な場合もあるが推奨しない。)

ショートカットを利用する場合には、casim.exe に対して作成すること。

表 B-1　CASIM を構成するファイルと配置

```
パーソナルコンピューターのローカルディスク上の任意のディレクトリ
|
|--casim.exe (必須)
|--casimmat.dll (必須)
|--casimphy.dll (必須)
|
+--ja (なくてもよい)
|   |--casim.resources.dll
|
+--en (なくてもよい)
    |--casim.resources.dll
```

C. CASIM の実行に必要なパーソナルコンピュータの要件

CASIM の実行には、Microsoft 社の OS「Windows®」が稼動するパーソナルコンピューターで、.Net（「ドットネット」と発音する）Framework 2.0 以降が装備されているものが必要である。.Net Framework 2.0 以降が装備されていれば、どの Windows® のバージョンでも動作することが期待できる。

.Net Framework は、様々な言語や手法で記述されたプログラムを共通の中間言語にコンパイルし、実行時に必要に応じてさらに機械語にコンパイルしつつ動作させるミドルウェアである。2009 年 6 月 17 日現在、Microsoft 社の以下の Web ページから入手できる。あるいは、Windows® の更新機能である Windows Update や Microsoft Update を用いても.Net Framework の導入が可能である。

http://msdn.microsoft.com/ja-jp/netframework/cc807036.aspx

画面表示のサイズの関係上、ディスプレイの解像度は 800×600 ピクセル以上が必要である。CPU、メモリー、ハードディスクの要件は、ほとんどない。上述の.Net Framework 2.0 が稼動しさえすれば、現状で実用に供されているほとんどのパーソナルコンピューターで動作する。

ただし、.Net Framework 2.0 を稼動状態にするための作業負荷等を考慮すれば、OS は Windows® 2000 又は XP あるいはそれ以降、CPU は Pentium® III 800 MHz 以上、メモリーは 256 MByte 以上、ハードディスクの空き容量は 200 MByte 以上であることが望ましい。

D. 英語表示オプション

CASIM の画面は図 D-1 に示すような英語による表示も可能である。ただし、付録 B の表 B-1 に示したサブディレクトリー「en」とその中のファイル「casim.resources.dll」が必要である。

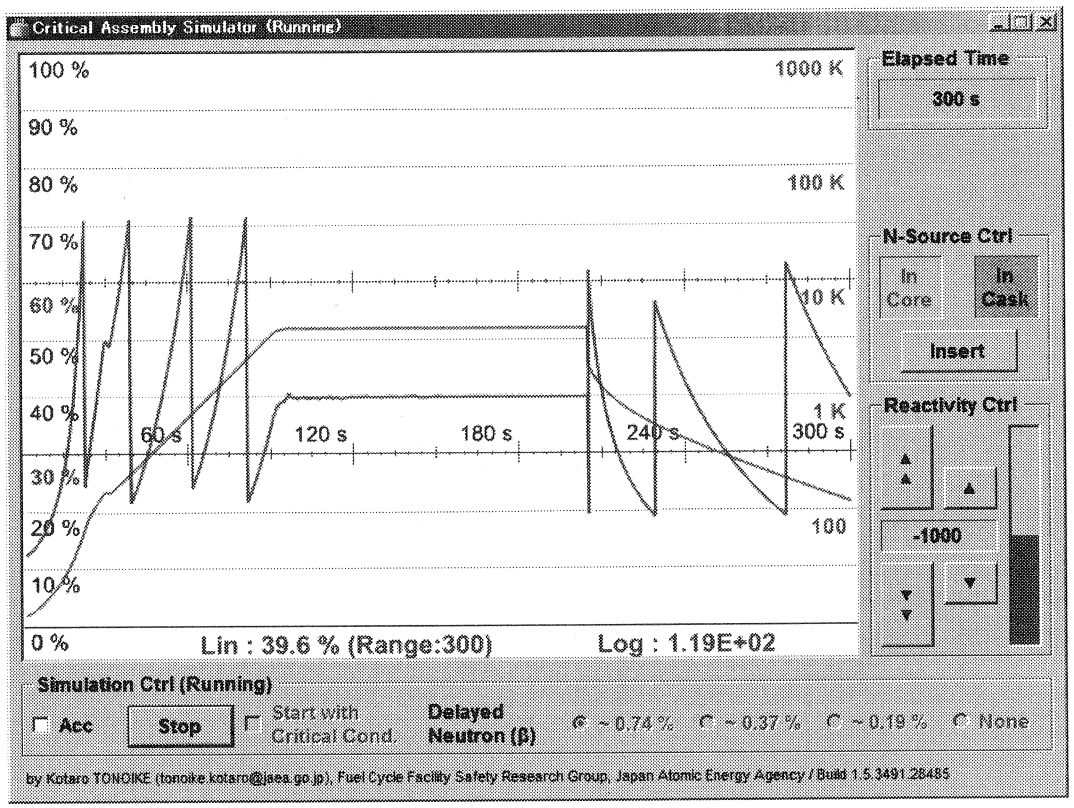

図 D-1　英語表示を用いた CASIM の実行
（シミュレーションの内容は図 4-1 とほぼ同じである。）

この英語表示を行うには、CASIM の実行ファイル（表 B-1 に示す casim.exe）を「/e」のコマンドライン変数を付して起動すればよい。また、CASIM を英語版の Windows® が稼動するパーソナルコンピューターにインストールした場合には、コマンドライン変数を付さなくても、casim.exe を起動すれば英語表示となる。

「/e」のコマンドライン変数の設定の仕方にはいくつかの方法があるが、以下に代表的なものを例示しておく。

- コマンドプロンプトから起動する方法

  コマンドプロンプトで、casim.exe がインストールされているディレクトリに移動し、「casim.exe /e」と入力して起動する。「casim.exe」と「/e」の間にはスペースをひとつ置く。

➢ ショートカットを用いる方法

任意のディレクトリ（多くの場合はデスクトップ）に casim.exe のショートカットを作成することができる。ショートカットを作成した後、そのショートカットを右クリックしてプロパティーを表示させ、「リンク先」が元の casim.exe を示していることを確認する。この末尾に、ひとつスペースを置いて「/e」を追加し、「OK」を押してプロパティー画面を閉じる。以上の操作を行ったショートカットをダブルクリックしてCASIM を起動すると英語表示となる。

This is a blank page.

国際単位系（SI）

表１．SI 基本単位

基本量	SI 基本単位	
	名称	記号
長さ	メートル	m
質量	キログラム	kg
時間	秒	s
電流	アンペア	A
熱力学温度	ケルビン	K
物質量	モル	mol
光度	カンデラ	cd

表２．基本単位を用いて表されるSI組立単位の例

組立量	SI 基本単位	
	名称	記号
面積	平方メートル	m^2
体積	立法メートル	m^3
速さ，速度	メートル毎秒	m/s
加速度	メートル毎秒毎秒	m/s^2
波数	毎メートル	m^{-1}
密度，質量密度	キログラム毎立方メートル	kg/m^3
面積密度	キログラム毎平方メートル	kg/m^2
比体積	立方メートル毎キログラム	m^3/kg
電流密度	アンペア毎平方メートル	A/m^2
磁界の強さ	アンペア毎メートル	A/m
量濃度(a)，濃度	モル毎立方メートル	mol/m^3
質量濃度	キログラム毎立法メートル	kg/m^3
輝度	カンデラ毎平方メートル	cd/m^2
屈折率(b)	（数字の） 1	1
比透磁率(b)	（数字の） 1	1

(a) 量濃度（amount concentration）は臨床化学の分野では物質濃度（substance concentration）ともよばれる。
(b) これらは無次元量あるいは次元１をもつ量であるが、そのことを表す単位記号である数字の１は通常は表記しない。

表３．固有の名称と記号で表されるSI組立単位

組立量	SI 組立単位			
	名称	記号	他のSI単位による表し方	SI基本単位による表し方
平面角	ラジアン(b)	rad	1 (b)	m/m
立体角	ステラジアン(b)	sr(c)	1 (b)	m^2/m^2
周波数	ヘルツ (d)	Hz		s^{-1}
力	ニュートン	N		m kg s^{-2}
圧力，応力	パスカル	Pa	N/m^2	m^{-1} kg s^{-2}
エネルギー，仕事，熱量	ジュール	J	N m	m^2 kg s^{-2}
仕事率，工率，放射束	ワット	W	J/s	m^2 kg s^{-3}
電荷，電気量	クーロン	C		s A
電位差（電圧），起電力	ボルト	V	W/A	m^2 kg s^{-3} A^{-1}
静電容量	ファラド	F	C/V	m^{-2} kg^{-1} s^4 A^2
電気抵抗	オーム	Ω	V/A	m^2 kg s^{-3} A^{-2}
コンダクタンス	ジーメンス	S	A/V	m^{-2} kg^{-1} s^3 A^2
磁束	ウエーバ	Wb	Vs	m^2 kg s^{-2} A^{-1}
磁束密度	テスラ	T	Wb/m^2	kg s^{-2} A^{-1}
インダクタンス	ヘンリー	H	Wb/A	m^2 kg s^{-2} A^{-2}
セルシウス温度	セルシウス度(e)	℃		K
光束	ルーメン	lm	cd sr(c)	cd
照度	ルクス	lx	lm/m^2	m^{-2} cd
放射性核種の放射能（f）	ベクレル (d)	Bq		s^{-1}
吸収線量，比エネルギー分与，カーマ	グレイ	Gy	J/kg	m^2 s^{-2}
線量当量，周辺線量当量，方向性線量当量，個人線量当量	シーベルト (g)	Sv	J/kg	m^2 s^{-2}
酸素活性	カタール	kat		s^{-1} mol

(a)SI接頭語は固有の名称と記号を持つ組立単位と組み合わせても使用できる。しかし接頭語を付した単位はもはやコヒーレントではない。
(b)ラジアンとステラジアンは数字の１に対する単位の特別な名称で、量についての情報をつたえるために使われる。実際には、使用する時には記号radおよびsrが用いられるが、習慣として組立単位としての記号である数字の１は明示されない。
(c)測光学ではステラジアンという名称と記号srを単位の表し方の中に、そのまま維持している。
(d)ヘルツは周期現象についてのみ、ベクレルは放射性核種の統計的過程についてのみ使用される。
(e)セルシウス度はケルビンの特別な名称で、セルシウス温度を表すために使用される。セルシウス度とケルビンの単位の大きさは同一である。したがって、温度差や温度間隔を表す数値はどちらの単位で表しても同じである。
(f)放射性核種の放射能（activity referred to a radionuclide）は、しばしば誤った用語で"radioactivity"と記される。
(g)単位シーベルト（PV,2002,70,205）についてはCIPM勧告2（CI-2002）を参照。

表４．単位の中に固有の名称と記号を含むSI組立単位の例

組立量	SI 組立単位		
	名称	記号	SI 基本単位による表し方
粘度	パスカル秒	Pa s	m^{-1} kg s^{-1}
力のモーメント	ニュートンメートル	N m	m^2 kg s^{-2}
表面張力	ニュートン毎メートル	N/m	kg s^{-2}
角速度	ラジアン毎秒	rad/s	m m^{-1} s^{-1}=s^{-1}
角加速度	ラジアン毎秒毎秒	rad/s^2	m m^{-1} s^{-2}=s^{-2}
熱流密度，放射照度	ワット毎平方メートル	W/m^2	kg s^{-3}
熱容量，エントロピー	ジュール毎ケルビン	J/K	m^2 kg s^{-2} K^{-1}
比熱容量，比エントロピー	ジュール毎キログラム毎ケルビン	J/(kg K)	m^2 s^{-2} K^{-1}
比エネルギー	ジュール毎キログラム	J/kg	m^2 s^{-2}
熱伝導率	ワット毎メートル毎ケルビン	W/(m K)	m kg s^{-3} K^{-1}
体積エネルギー	ジュール毎立方メートル	J/m^3	m^{-1} kg s^{-2}
電界の強さ	ボルト毎メートル	V/m	m kg s^{-3} A^{-1}
電荷密度	クーロン毎立方メートル	C/m^3	m^{-3} sA
表面電荷	クーロン毎平方メートル	C/m^2	m^{-2} sA
電束密度，電気変位	クーロン毎平方メートル	C/m^2	m^{-2} sA
誘電率	ファラド毎メートル	F/m	m^{-3} kg^{-1} s^4 A^2
透磁率	ヘンリー毎メートル	H/m	m kg s^{-2} A^{-2}
モルエネルギー	ジュール毎モル	J/mol	m^2 kg s^{-2} mol^{-1}
モルエントロピー，モル熱容量	ジュール毎モル毎ケルビン	J/(mol K)	m^2 kg s^{-2} K^{-1} mol^{-1}
照射線量（X線及びγ線）	クーロン毎キログラム	C/kg	kg^{-1} sA
吸収線量率	グレイ毎秒	Gy/s	m^2 s^{-3}
放射強度	ワット毎ステラジアン	W/sr	m^4 m^{-2} kg s^{-3}=m^2 kg s^{-3}
放射輝度	ワット毎平方メートル毎ステラジアン	W/(m^2 sr)	m^2 m^{-2} kg s^{-3}=kg s^{-3}
酵素活性濃度	カタール毎立方メートル	kat/m^3	m^{-3} s^{-1} mol

表５．SI 接頭語

乗数	接頭語	記号	乗数	接頭語	記号
10^{24}	ヨタ	Y	10^{-1}	デシ	d
10^{21}	ゼタ	Z	10^{-2}	センチ	c
10^{18}	エクサ	E	10^{-3}	ミリ	m
10^{15}	ペタ	P	10^{-6}	マイクロ	μ
10^{12}	テラ	T	10^{-9}	ナノ	n
10^{9}	ギガ	G	10^{-12}	ピコ	p
10^{6}	メガ	M	10^{-15}	フェムト	f
10^{3}	キロ	k	10^{-18}	アト	a
10^{2}	ヘクト	h	10^{-21}	ゼプト	z
10^{1}	デカ	da	10^{-24}	ヨクト	y

表６．SIに属さないが、SIと併用される単位

名称	記号	SI 単位による値
分	min	1 min=60s
時	h	1h =60 min=3600 s
日	d	1 d=24 h=86 400 s
度	°	1°=(π/180) rad
分	’	1’=(1/60)°=(π/10800) rad
秒	”	1”=(1/60)’=(π/648000) rad
ヘクタール	ha	1ha=1hm^2=10^4m^2
リットル	L，l	1L=11=1dm^3=10^3cm^3=$10^{-3}m^3$
トン	t	1t=10^3 kg

表７．SIに属さないが、SIと併用される単位で、SI単位で表される数値が実験的に得られるもの

名称	記号	SI 単位で表される数値
電子ボルト	eV	1eV=1.602 176 53(14)×10^{-19}J
ダルトン	Da	1Da=1.660 538 86(28)×10^{-27}kg
統一原子質量単位	u	1u=1 Da
天文単位	ua	1ua=1.495 978 706 91(6)×10^{11}m

表８．SIに属さないが、SIと併用されるその他の単位

名称	記号	SI 単位で表される数値
バール	bar	1 bar=0.1MPa=100kPa=10^5Pa
水銀柱ミリメートル	mmHg	1mmHg=133.322Pa
オングストローム	Å	1 Å=0.1nm=100pm=10^{-10}m
海里	M	1 M=1852m
バーン	b	1 b=100fm^2=(10^{-12}cm)2=$10^{-28}m^2$
ノット	kn	1 kn=(1852/3600)m/s
ネーパ	Np	SI単位との数値的な関係は、対数量の定義に依存。
ベル	B	
デジベル	dB	

表９．固有の名称をもつCGS組立単位

名称	記号	SI 単位で表される数値
エルグ	erg	1 erg=10^{-7} J
ダイン	dyn	1 dyn=10^{-5}N
ポアズ	P	1 P=1 dyn s cm^{-2}=0.1Pa s
ストークス	St	1 St =1cm^2 s^{-1}=$10^{-4}m^2$ s^{-1}
スチルブ	sb	1 sb =1cd cm^{-2}=10^4cd m^{-2}
フォト	ph	1 ph=1cd sr cm^{-2} 10^4lx
ガル	Gal	1 Gal =1cm s^{-2}=$10^{-2}ms^{-2}$
マクスウェル	Mx	1 Mx = 1G cm^2=10^{-8}Wb
ガウス	G	1 G =1Mx cm^{-2} =10^{-4}T
エルステッド（c）	Oe	1 Oe ≙ (10^3/4π)A m^{-1}

(c) ３元系のCGS単位系とSIでは直接比較できないため、等号「 ≙ 」は対応関係を示すものである。

表10．SIに属さないその他の単位の例

名称	記号	SI 単位で表される数値
キュリー	Ci	1 Ci=3.7×10^{10}Bq
レントゲン	R	1 R = 2.58×10^{-4}C/kg
ラド	rad	1 rad=1cGy=10^{-2}Gy
レム	rem	1 rem=1 cSv=10^{-2}Sv
ガンマ	γ	1 γ =1 nT=10-9T
フェルミ		1フェルミ=1 fm=10-15m
メートル系カラット		1メートル系カラット = 200 mg = 2×10-4kg
トル	Torr	1 Torr = (101 325/760) Pa
標準大気圧	atm	1 atm = 101 325 Pa
カロリー	cal	1cal=4.1858J（「15℃」カロリー），4.1868J（「IT」カロリー）4.184J（「熱化学」カロリー）
ミクロン	μ	1 μ =1μm=10^{-6}m

（第8版，2006年改訂）

この印刷物は再生紙を使用しています。