原子核のアルファ崩壊系列

2015年2月

福井大学 工学部 物理工学科
松岡　弘和

目 次

第1章　序論

物理工学科で学んだことの集大成としての卒業研究のテーマとして私は原子核の α（アルファ）崩壊系列の時間発展の計算という課題を選んだ。この課題は, 数学の観点からは, 時間をかけて学んできた微分積分や線形代数の知識を活用できる線形 1 階連立常微分方程式の解法という問題とみることができるし, 情報技術という観点からは, 私が比較的得意分野だと感じている教科である C 言語でのプログラミングの応用課題となっている。工学的視点からは 2011 年の東日本大震災で発生した東京電力福島第一原子力発電所の炉心溶融事故以降社会的関心の高まった放射性物質の話題である。そして, 物理学としては私が興味をもつ原子核や素粒子といった物質のミクロな構造に奥では関連している課題として興味の持てるものである。こういった理由からこの課題を私の卒業研究の課題として選んだのである。

本研究は大沼・森近両氏による 2011 年度の卒業研究 [1] で開発された原子核の放射性崩壊系列の時間発展を連立常微分方程式を求める C 言語プログラムを発展させる。具体的には, 半減期や崩壊分岐比等の大量のデータをプログラム中に記述する方法を改良する。大沼・森近両氏は, このプログラムを質量数 90〜140 の核分裂生成物の β（ベータ）崩壊系列に対して使用したが, 本研究ではアクチナイド核の α 崩壊系列に対して使用する。β 崩壊系列については, 崩壊系列の下流に進むにつれて核種の半減期が長くなることが原因で, 系列の全放射線量はベクレル単位で時間に反比例することが大沼・森近両氏により見出された。本研究では, α 崩壊系列で, 系列の開始点として, 核図表上で周囲に配置される核より顕著に半減期の大きな核種 (^{238}U 等) を使うと, これは放射平衡と呼ばれる, β 崩壊系列とは異なった崩壊の様態を示すことがよく知られているが, まず, そのことを確かめる。次に, 崩壊の開始点を, もっと重く不安定な（半減期の短い）核にとると系列の核種分布や総放射線量はどのような挙動を示すかを調べる。例として日本の理化学研究所が発見した原子番号 113 番元素の放射性崩壊系列の経時変化等を示す。

本論文では, まず, 素粒子, 原子核, 原子核の放射性崩壊についての基礎知識を整理してまとめを示す。次に定数係数連立 1 階常微分方程式の解析的厳密解について詳しく説明し, 典型的な状況設定をいくつか行って, その解の挙動をグラフ化し, 性質について論じる。最後に, 研究の主目的である α 崩壊系列の計算結果を示し, その挙動について論じる。付録として本研究のために新規に開発した C 言語プログラムや, 改良した C 言語プログラムのプログラム・リストを採録する。

第2章　自然の階層構造

2.1　素粒子

素粒子 (elementary particle) とは, 現在発見されている中では最小の物質構造であり, 基本粒子とされているものをさす言葉である。

18 世紀にドルトンにより初めて (哲学的でなく) 科学的な原子論が成立し, その後 19 世紀末までは基本的な粒子は原子と考えられていた。

しかし, 1897 年のトムソンによる電子の発見および 1911 年のラザフォードによる原子核の発見により, 原子が物質構造の最小単位ではないということが発見された。その後の陽子と中性子の発見, さらに湯川秀樹の中間子論等によって素粒子の存在が確定され今日の素粒子物理学に至っている。

さて, 素粒子は大きくフェルミ粒子 (Fermion), ボース粒子 (Boson) にわけられる。フェルミ粒子及びボース粒子とは, その多粒子状態の波動関数が粒子座標の交換に対して完全反対称及び完全対象である粒子として定義される。その重要な帰結として, フェルミ粒子は一つの状態には一つの粒子しか入れない。これをパウリの禁止則とよぶ。一方, ボース粒子は何個でも同一の状態に入ることができる。特に, 非常に多くの個数の粒子が一状態に入った場合をボース・アインシュタイン凝縮とよぶ。

実験的には, スピンの大きさが $\hbar$ の半整数 ($\frac{1}{2}, \frac{3}{2}, \frac{5}{2}, \cdots$) 倍であればフェルミ粒子, 整数 (0,1,2, $\cdots$) 倍であればボース粒子と判定できる。この関係は「スピンと統計の関係」と呼ばれ, 適当な仮定の下に相対論的場の理論に基づき証明することができる。

また, クォークの複合粒子であるハドロン (hadron) については別途 2.1.3 で述べる。

2.1.1　フェルミ粒子

フェルミ粒子 (Fermion) は物質の主要な構成要素であり, クォーク (quark) とレプトン (lepton) に大別される。以下でその 2 つについて順次述べる。

クォークは電子の $\pm\frac{1}{3}$ または $\pm\frac{2}{3}$ 倍の電荷を持つ粒子である。なお,1 個の電子の持つ電荷の −1 倍を素電荷と呼び, 記号 e で表す。なお, 同じ記号を用いる数学定数の自然対数の底 (= 2.718 $\cdots$) とは無関係である。その最新の測定値は $e = 1.602176565(35) \times 10^{-19}$ クーロンである (丸括弧内の「35」は末尾の桁の「65」の含む誤差の標準偏差を表す [2])。この記号 e を用いると, クォークの電荷は $\pm\frac{1}{3}e$ または $\pm\frac{2}{3}e$ である。

クォークは up, down, strange, charm, bottom, top の 6 種類が存在することが実験的に確認されており, それぞれ, u, d, s, c, b, t の記号で表される。クォークは似た性質を持つもの同士の 2 つずつの組み合わせに分けることができる。この組み合わせは「世代 (generation)」とよばれ, 表 2.1 に示されている。今現在においてクォークを単独で取り

表 2.1: フェルミオンの一覧表。なお, これらのほかに, それぞれの反粒子が存在する。

	名称	記号	世代	電荷	質量
クォーク	up	u	第 1 世代	$\frac{+2}{3}e$	$2.3^{+0.7}_{-0.5}$MeV[1]
	down	d	第 1 世代	$\frac{-1}{3}e$	$4.8^{+0.7}_{-0.3}$MeV
	charm	c	第 2 世代	$\frac{+2}{3}e$	1.275 ± 0.025GeV
	strange	s	第 2 世代	$\frac{-1}{3}e$	95 ± 5MeV
	top	t	第 3 世代	$\frac{+2}{3}e$	
	bottom	b	第 3 世代	$\frac{-1}{3}e$	
レプトン	電子	e	第 1 世代	-1e	$^{0.510998928}_{\pm 0.000000011}$MeV
	電子ニュートリノ	ν_e	第 1 世代	0	
	ミュー粒子	μ	第 2 世代	-1e	$^{105.6583715}_{\pm 0.0000035}$MeV
	ミュー・ニュートリノ	ν_μ	第 2 世代	0	
	タウ粒子	τ	第 3 世代	-1e	1776.82 ± 0.16MeV
	タウ・ニュートリノ	ν_τ	第 3 世代	0	

出すことはできていないが, 実験においてはジェット (そろった方向に発生する多数のハドロンの束) によって存在を確認されている。

レプトンはクォークとは異なり単独で粒子として存在する。レプトンと呼ばれるものの中には電子 (electron), 電子ニュートリノ (electron neutrino), ミュー粒子 (muon), ミュー・ニュートリノ (muon neutrino), タウ粒子 (tauon, tau), タウ・ニュートリノ (tau neutrino) がある。それぞれ $e, \nu_e, \mu, \nu_\mu, \tau, \nu_\tau$ という記号で表される。また, クォークと同じように性質によって 2 つずつの組み合わせに分けることができる。この組み合わせを世代と呼ぶ。

レプトンの間には弱い相互作用が働き, 原子炉等における中性子や不安定原子核の β 崩壊に関与する。また, ミュー粒子は宇宙から降りそそぐ放射線, すなわち宇宙線の 2 次粒子として, ニュートリノは太陽の中心部での核反応や超新星爆発に伴って放出されるものとしても観測することができる。ニュートリノについては岐阜県の神岡鉱山をはじめとして世界数箇所に観測施設が存在している。これらクォークやレプトンは基本粒子と考えられている。なお, フェルミ粒子の諸性質のまとめを表 2.1 の下半分に示す。

2.1.2 ボース粒子

粒子間に力が働くのは, 力を媒介する粒子が存在するからだと考えられている。この粒子がボース粒子 (Boson) であり, 前節のフェルミ粒子と同様に, これらのボース粒子も基本粒子であるとされている。ボース粒子には, 電磁気力を媒介する粒子である光子

[1] eV とは, 電子ボルトとも呼ばれるエネルギーの単位であり, 1V の電位差において, 1 電気素量の粒子を加速させた時に得られる運動エネルギーのことである。なお, 1eV=1.602176565(35)×10^{-19}J となる。

表 2.2: 主なハドロンの一覧表

名称	英名	記号	電荷	質量
陽子	proton	p	$+e$	938.272046±0.000021MeV
中性子	neutron	n	0	939.565379±0.000021MeV
パイ中間子	pion	π^+	$+e$	139.57018±0.000MeV
		π^0	0	134.9766±0.0006MeV
		π^-	-e	139.57018±0.000MeV

(photon), 弱い力を媒介する粒子であるウィーク・ボソン (weak boson), 強い力を媒介する粒子であるグルーオン (gluon) が知られている。更に, 重力を媒介する粒子である重力子 (graviton) も存在すると考えられているが, 現在のところはまだ実験的に観測されたことがない。

また, ヒッグス粒子 (Higgs particle, Higgs Boson) とよばれるボース粒子も存在が仮定されてきたが, 2012 年に欧州原子核研究機構 (CERN) によって発見された粒子がこのヒッグス粒子であると考えられている。Higgs 粒子は物質のもつ質量の起源であると考えられている。

2.1.3 ハドロン

ハドロン (hadron) とは, 前述した基本粒子のうちのクォークが強い相互作用によって結びつくことで構成された複合粒子をさすことばであり, バリオン (baryon) とメソン (またはメゾン, meson) の 2 つに大別することができる。

バリオンは重粒子ともよばれる。バリオンは 3 つのクォークにより構成され, 代表的なものとして陽子と中性子とが挙げられる。陽子は u クォーク 2 個と d クォーク 1 個から構成され, 中性子は u クォーク 1 個と d クォーク 2 個から構成される複合粒子だと考えられている。1 個のバリオンを構成するクォークの組み合わせは, そのバリオンの総電荷量が常に電子の電荷の整数倍になるものに制限されている。そのようなクォークの可能な組み合わせで, クォーク概念が登場する以前の時代に素粒子であるとされていた粒子をすべて説明することができる。このことを陽子を例にとって確認してみると, 陽子は電 荷が $+\frac{2}{3}e$ の u クォークが 2 個と $-\frac{1}{3}e$ の d クォークが 1 個から成るのだから, その総電荷量を計算してみると,

$$\frac{+2}{3}e \times 2 + \frac{-1}{3}e = 1 \cdot e \tag{2.1}$$

という式の通り, ちゃんと素電荷 e の整数倍 (この場合は 1 倍) になっている。

メソンは中間子ともよばれる。メソンは 2 個のクォークにより構成され, 代表的なものとしてはパイ中間子 (pion) が挙げられる。パイ中間子も基本粒子であるクォークによって構成されており, 電荷が $+e$, 0, $-e$ で 3 種類が存在し, それぞれ π^+, π^0, π^- という記号で表される。主なハドロンの諸性質を表 2.2 にまとめておく。

2.2　原子核

原子核 (atomic nucleus) は, 2.1 節において既に述べたように, ラザフォードによって発見された。原子核は陽子と中性子から構成される。原子核は原子の中心に存在し, 原子の質量のほとんどを占める。例えば水素原子の場合, 99.95 %を占める。水素の原子核は陽子 1 個で構成されており, 直径は 10^{-15}m 程である。これは次節で述べる原子と比較すると 4 桁も小さい。原子核については 3 章で更に詳しく述べる。

2.3　原子

原子 (atom) は古代ギリシャ時代の「これ以上分解することができない基本粒子」という意味から生まれた言葉だが, 今日では原子核と（素粒子である）電子とに更に分解できることがわかっている。原子核の発見前は, 原子の構造のモデルとして, 1903 年にトムソンによるプラム・プディングモデル, 同じ年に長岡半太郎による土星型モデルが考案されていた。1911 年のラザフォードによる原子核の発見後は, ラザフォード本人による惑星モデルも考案されたが, 1913 年にボーアによって考案された前期量子論的な原子モデルが最も画期的であり, 今日でも半古典的量子化法の成功例として教育的な価値のある描像として活用されている。この原子モデルでは, 原子は原子核の周りを電子が回っていることにより形作られている。原子の大きさは 10^{-10} m程度とされている。この長さは, 非 SI 単位のオングストロームを使うと, ちょうど 1Å である。

第3章　原子核

3.1　構成要素

原子核 (atomic nucleus) は前述したように陽子 (proton) と中性子 (neutron) から構成されている。この事実は, 1932 年にチャドウィックによる中性子の発見により判明した。

陽子の個数を原子番号 (atomic number), 陽子と中性子の個数の和を質量数 (mass number) と呼ぶ。一般に, 質量数を A, 中性子の個数（中性子数, neutron number）を N, 原子番号を Z という記号を用いて表す。なお, 陽子と中性子を総称して核子 (nucleon) という。これらの 3 数の間の関係を数式で表すと,

$$A = Z + N \tag{3.1}$$

となる。したがって, 原子核の種類は A, Z, N のうちの 2 つの数だけで指定することができ, 通常は質量数 A と原子番号 Z で指定する。そのように指定された原子核は, Z 個の陽子と $A - Z$ 個の中性子よりなり, Ze の正電荷を持つ。

質量数 A と原子番号 Z によって指定されるそれぞれの原子核のことを核種とよぶ。質量数 A, 原子番号 Z, 中性子数 N, 元素記号 X の核種は ${}^{A}_{Z}\mathrm{X}_{N}$ のように表記する (脚注 [1] 参照。)。

具体例として, 原子炉の燃料として利用されている質量数が 235 のウラン原子核を例にとると, ウラン原子は原子番号 92 なので, 中性子数は $235 - 92 = 143$ であり, ${}^{235}_{92}\mathrm{U}_{143}$ という記号で表記される。

原子の化学的性質は, 陽子の個数（即ち原子番号）だけでほぼ決まってしまう。したがって, 陽子の個数が, 原子番号と呼ばれ, 元素名 (element name), 元素記号 (element symbol) と一対一に対応するのである。元素を化学的性質を考慮して表に配置したものが周期表 (periodic table) である。今日までに 115 個の元素の存在が実験的に確認されている。

同じ元素の中でも原子核内の中性子数が異なる原子を同位体（isotope）とよぶ。なお, 中性子数が同じものを同調核（isotone）, 質量数が同じものを同重核（isobar）と呼ぶ。同位体の例を挙げると, 水素には ${}^{1}_{1}\mathrm{H}_{0}$, ${}^{2}_{1}\mathrm{H}_{1}$, ${}^{3}_{1}\mathrm{H}_{2}$ の 3 つの同位体がある。同位体の中には安定して存在する安定同位体と放射性物質である放射性同位体がある。放射性同位体は, 時間とともに放射性崩壊を起こし, 放射線を出すとともに別種の原子核に変わる。放射性同位体は, 広く自然界に存在する天然放射性同位体と加速器や原子炉によって生成される人工放射性同位体に大別することができる。

[1] 但し, 元素記号の右下の添字は化学式では原子の個数を表すので読む人が意味を混同しないように注意して用いなければならない。

3.2 原子核の質量公式

原子核の質量公式 (mass formula) について述べる前に, まず, 質量とエネルギーの間に成り立つ関係, 即ち, アインシュタインによって見出された質量とエネルギーの等価性を表す下記の式が成り立つことを述べておく必要がある。

$$E = mc^2 \tag{3.2}$$

ただし, c は光速であり, $c = 299792458\mathrm{m{\cdot}s^{-1}}$ である。[2]

(3.2) 式により, 質量 m の物質はエネルギー E をもつと考えることができる。この式は, 単に, それまでは別種の量と考えていたエネルギーと質量とを, 換算率を c^2 だと定義して, 今後は比較してよいことにしよう, と提唱しているだけに過ぎないように思えるかもしれないが, 実はそれは正しい理解ではない。物質が反応をし, 別の物質になることを考えると,

$$\Delta E = \Delta mc^2 \tag{3.3}$$

となる。(3.3) 式は, 物質の反応で吸収・放出されるエネルギー ΔE と質量の変化 Δm を関係付ける有用な法則である。

(3.2) 式を用い, 原子核の結合エネルギー（binding energy）を質量から求めるための式を導こう。結合エネルギーとは, 原子核をばらばらな核子の集まりに分解するのに必要な最小エネルギーのことである。ばらばらに離れて静止している Z 個の陽子と N 個の中性子のエネルギーを E_1 とすると, (3.2) 式を用いて

$$E_1 = ZM_pc^2 + NM_nc^2 \tag{3.4}$$

となる。これら A 個の核子が集まって原子核を作ったときのエネルギーを E_2 とすると, 核子の質量を M_A として,

$$E_2 = M_Ac^2 \tag{3.5}$$

となる。このとき, 原子核となることで余るエネルギーが結合エネルギーであり, 記号 B であらわすと,

$$E_1 = E_2 + B \tag{3.6}$$

$$\begin{aligned} \therefore B &= E_1 - E_2 \\ &= (ZM_p + NM_n - M_{A,Z})c^2 = \Delta Mc^2 \end{aligned} \tag{3.7}$$

が成り立つ。(3.7) 式における B が核種の全結合エネルギーである。B は正の値を取る。また, ΔM を質量欠損と呼ぶ。結合エネルギー B を質量数 A で割ったもの, 即ち $\frac{B}{A}$ を核子 1 個あたりの結合エネルギーという。

(3.7) 式を用いると質量が既知の核種の B が計算できる。また, 核反応で放出・吸収されるエネルギーからも B が計算できる。ヴァイツゼッカーとベーテはこうして得られた多数の（現在では三千個にもなる）核種の B を N, Z, A の簡単な数式で表すことに成功した。それが原子核の質量公式である。以下が, Z 個の陽子と N 個の中性子を持つ原子の質量を求める式である。

$$M_{A,Z}c^2 = NM_nc^2 + ZM_pc^2 + Zm_ec^2 - a_vA + a_sA^{\frac{2}{3}} + a_c\frac{Z^2}{A^{\frac{1}{3}}} + a_a\frac{(N-Z)^2}{4A} + \frac{\delta}{A^{\frac{1}{2}}} \tag{3.8}$$

(3.8) 式において m_e は電子の質量を表す記号であり, 既に表 2.1 に記した通り,

$$m_e = 0.510998928 \pm 0.000000011\text{MeV} \tag{3.9}$$

である。パラメータ a_s,a_v,a_c,a_a,δ の値は, 多数の原子核の束縛エネルギーへの最小 2 乗法によるフィッティングの結果, 下記の値が得られている。

$$a_v = 15.67\text{MeV} \tag{3.10}$$

$$a_s = 17.23\text{MeV} \tag{3.11}$$

$$a_c = 0.714\text{MeV} \tag{3.12}$$

$$a_a = 93.15\text{MeV} \tag{3.13}$$

$$\delta = \begin{cases} -11.2\,\text{MeV}(\text{ZとNが偶数の場合 (偶偶核)}) \\ 0\,\text{MeV}(\text{Aが奇数の場合 (奇偶核)}) \\ +11.2\,\text{MeV}(\text{ZとNが奇数の場合 (奇奇核)}) \end{cases} \tag{3.14}$$

(3.8) 式の第 1 項は体積項, 第 2 項は表面項, 第 3 項はクーロン項, 第 4 項は非対称項, 第 5 項は対結合エネルギー項とそれぞれよばれる。それぞれの項についての詳しい説明は文献 [4] の 18 ページから始まる 2.3 節を参照されたい。

第4章　原子核の放射性崩壊

4.1　放射性同位体の性質

3.1 で触れた放射性同位体は不安定な状態にあり, 粒子線や電磁波を放出して安定な同位体になろうとする性質を持つ。こういった性質を持つ原子核を含んだ物質を東日本大震災以来よく聞くようになった放射性物質 (radioative material) と呼ぶ。この現象を原子核の放射性崩壊 (radioactive decay) という。

一般的傾向として, 原子核は重くなればなるほど陽子同士のクーロン斥力が強くなるため不安定となり, α 崩壊と呼ばれる放射性崩壊を起こしやすくなる。また陽子と中性子の個数の差が大きくなるほど不安定となり, β 崩壊と呼ばれる放射性崩壊を起こしやすくなる。

放射性物質の質量が半分になるまでにかかる時間を半減期 (halflife) という。物質によって半減期は長短さまざまである。あまりにも半減期の長いものは放射性崩壊が起こらないと見なしてよい。

4.2　α・β・γ 崩壊

原子核の放射性崩壊は大きく 3 つに分けることができ, α 崩壊・β 崩壊・γ（ガンマ）崩壊という。α 崩壊は質量数（核子の個数）を変える崩壊様式, β 崩壊は核種を変えるが質量数（核子の個数）を変えない崩壊様式, γ 崩壊は核種（陽子の個数と中性子の個数の組み合わせ）を変えない崩壊様式である。この 3 つについて以下で述べる。

4.2.1　質量数を変える崩壊様式

重い原子核は α 線を出して崩壊するものが多い。この崩壊を α 崩壊という。α 線の正体は ${}^{4}_{2}\mathrm{He}_2$ である。原子核 X の α 崩壊を反応式で表すと (4.1) 式のようになる。

$$ {}^{A}_{Z}\mathrm{X}_N \rightarrow {}^{A-4}_{Z-2}\mathrm{X}_{N-2} + {}^{4}_{2}\mathrm{He}_2 \tag{4.1} $$

この崩壊により質量数（核子の個数）が 4 だけ減少する。

陽子 2 個, 中性子 2 個をばらばらに放出するのではない理由は結合エネルギーにある。4 個の核子がばらばらな状態を基準とした場合の ${}^{4}_{2}\mathrm{He}_2$ の結合エネルギーは約 28 MeV[3] である。このエネルギーの分だけ陽子 2 個, 中性子 2 個をばらばらに放出するよりも残りの原子核が安定になるためである。

また, 中性子が過多な場合には, 中性子を 1 個放出する原子核も存在する。例として, ${}^{7}_{2}\mathrm{He}_{5}$ はこの現象を起こす。さらに数は少なくなるが, 陽子を 1 個放出するものも存在する。こちらは ${}^{4}_{3}\mathrm{Li}_{1}$ が例としてあげられる。なお, この 2 例は半減期がきわめて短い。

この他に α 粒子よりも重い ${}^{28}\mathrm{Mg}$ などの核を放出する崩壊様式や, 同程度の重さの 2 個の核にちぎれる自発的核分裂という現象も存在し, それらも「質量数を変える崩壊様式」に含まれる。

4.2.2 核種を変えるが質量数を変えない崩壊様式

安定な原子核と比較して, 陽子や中性子のどちらかが多い場合, その原子核は陽子が中性子に, あるいは中性子が陽子に変化するとともに,β 線を放出するなどしてより安定な原子核へと崩壊する。この崩壊を β 崩壊という。

β 線の正体は電子または陽電子である。電子を放出する場合を β^- 崩壊, 陽電子を放出する場合を β^+ 崩壊という。

また, 核が核外にある軌道電子を捕獲することがある。この現象を軌道電子捕獲 (electron capture, EC と略す) という。原子核 X の β^- 崩壊,β^+ 崩壊, EC を反応式で表現するとそれぞれ（4.2）式, （4.3）式, （4.4）式のようになる。

$$ {}^{A}_{Z}\mathrm{X}_{N} \rightarrow {}^{A}_{Z+1}\mathrm{X}_{N-1} + e^- + \bar{\nu}_e \tag{4.2} $$

$$ {}^{A}_{Z}\mathrm{X}_{N} \rightarrow {}^{A}_{Z-1}\mathrm{X}_{N+1} + e^+ + \nu_e \tag{4.3} $$

$$ {}^{A}_{Z}\mathrm{X}_{N} + e^- \rightarrow {}^{A}_{Z-1}\mathrm{X}_{N+1} + \nu_e \tag{4.4} $$

上記のとおり,β 崩壊では質量数に変化は起こらないが原子番号が増減し, 同時に電子ニュートリノ ν_e または反電子ニュートリノ $\bar{\nu_e}$ を放出する。

4.2.3 核種を変えない崩壊様式

4.2.1 や 4.2.2 で生じた原子核は多くが励起状態とよばれるエネルギーが高い状態にある。このためより低い状態に移行するために光子を放出する。これを γ 崩壊と言い, 放出される光子は γ 線とよばれている。光子を放出する理由は原子核は基底状態に近いほうがより安定するからである。この光子のエネルギーはエネルギー準位差に相当するエネルギーである。

γ 崩壊は,α 崩壊や β 崩壊と違い, 核種（陽子の個数と中性子の個数の組み合わせ）を変えない崩壊様式である。また, γ 崩壊は電磁相互作用に起因する崩壊様式であるが, α 崩壊は強い相互作用, β 崩壊は弱い相互作用に起因する崩壊様式である。

γ 線のエネルギーが低いとき, そして特に, γ 線のもつ角運動量が大きいときは, 原子核の周囲に存在する電子に γ 線が吸収されて, γ 線の代わりにその電子が原子から跳ね飛ばされる現象が起きやすい。この跳ね飛ばされた電子を内部転換 (internal conversion)

または内部電子転換と呼ぶ。ただし, この場合のγ線は仮想光子であるにもかかわらず, 本来のγ線が再吸収されるという誤解をもたれ易いため, 核子が電子を直接跳ね飛ばすという描像で説明されることも多い。

α崩壊, β崩壊の終状態が娘核の基底状態でない場合は, ほとんどの場合, 即座に基底状態に至るまでγ崩壊を繰り返す。これを即発γ線と呼ぶ。しかし, 励起状態の中には, 特異的に長寿命のものがあり, 核異性体 (nuclear isomer) と呼ばれる。核異性体から基底状態に至るまでのγ崩壊は, 寿命の分だけ遅れて放射される。本研究の主題である放射性崩壊系列の計算では, 核異性体は基底状態は別の核のように扱う必要がある。

4.3 放射性崩壊系列

地球上には, 非常に多くの核種が存在している。まず, 安定核 (stable nucleus) と不安定核 (unstable nucleus) に分けることができる。安定核とは自発的に崩壊を起こさない核種であり, 約 300 種類存在している。

不安定核は崩壊を起こす核種であり, 放射性核種 (radionuclide, radioactive nuclide) と呼ばれている。核種の存在・不存在の定義には任意性があるが, 核構造理論の立場では約 1 万種が存在する（強い相互作用による自己束縛系としての構造計算の対象になりえる）と考えることが多い。放射性核種は, さらに人工放射性核種 (artificial radionuclide) と天然放射性核種 (natural radionuclide) に分けられる。

人工放射性核種は原子力発電や核爆発実験などで生成される, もともとは存在していなかった放射性核種のことである。天然放射性核種はさらに, 原始放射性核種 (primordial radionuclide) と宇宙線生成放射性核種 (cosmogenic nuclide) とに分けられる。

原始放射性物質は地球が作られたときから存在している放射性物質である。これらの原始放射性物質は非常に長い半減期を持っており, 18 種類 [5] 存在している。宇宙線生成放射性核種は宇宙線と大気低層の気体が核反応を起こし, 崩壊したものであり, 約 20 種類 [5] 存在している。

放射性崩壊系列 (radioactive decay sequence) とは, この中の原始放射性物質が崩壊していく過程のことである。原始放射性物質には放射性崩壊系列に属するものと, 属さないものが存在しており, 例として前者にはウランやトリウム, 後者にはカリウム 40 がある。まずはα崩壊系列について述べる。

4.3.1 α崩壊系列

α崩壊系列としては 4 系列が存在している。そのうちの 3 系列は天然に存在するが 1 系列は存在していない。

天然に存在する 3 系列は, ウラン 238 を先祖とするウラン系列, トリウム 232 を先祖とするトリウム系列, ウラン 235 を先祖とするアクチニウム系列であり, それぞれ, 質量数が整数 n を用いて $4n+2$, $4n$, $4n+3$ と表される核種からなる崩壊系列である。次々に崩壊し最終的にそれぞれ質量数が 206, 208, 207 の鉛の安定な同位体になる。

天然に存在しない 1 系列は, ネプツニウム 237 のネプツニウム系列である。これは質量数が $4n + 1$ の核種からなる系列であるが, ネプツニウム 237 の半減期が短いので, 地球ができてから現在までの間に全て崩壊して消えてしまい, 天然には存在しない。

これら 4 つの崩壊系列を表 4.1〜表 4.4 に記す。

表 4.1: ウラン系列 ($4n+2$ 系列)。　崩壊様式の欄で, SF は　spontaneus fission(自発的核分裂), IT は　isometric transision(異性体遷移, 即ち寿命の長い励起状態から基底状態への γ 遷移) を表す。なお複数の崩壊チャンネルが競合して崩壊が分岐している場合は, 娘核を矢印 $\rightarrow$ の右側に示した。表は次ページにも継続している。

元素	質量数	陽子数	中性子数	半減期	崩壊様式
U	238	92	146	4.468×10^{9}y	α
					SF, 5.5×10^{-5}%
Th	234	90	144	24.10d	β^-
Pa	234	91	143	6.70h	β^-
	234m	91	143	1.159m	β^-,99.84%
					IT, 0.16%
U	234	92	142	2.455×10^{5}y	α
					SF, 1.6×10^{-9}%
					Mg, 1×10^{-11}%
					Ne, 9×10^{-12}%
Th	230	90	140	7.54×10^{4}y	α
					^{24}Ne, 6×10^{-11}%
					SF, 4×10^{-12}%
Ra	226	88	138	1.6×10^{3}y	α
					^{14}C, 3.2×10^{-9}%
Rn	222	86	136	3.8235d	α
Po	218	84	134	3.098m	α,99.98% $\rightarrow$Pb
					β^-,0.02% $\rightarrow$At
At	218	85	133	1.5s	α,99.90% $\rightarrow$Bi
					β^-,0.10% $\rightarrow$Rn
Rn	218	86	132	35ms	α
Pb	214	82	132	26.8m	β^-
Bi	214	83	131	19.9m	β^-,99.98% $\rightarrow$Po
					α,0.02% $\rightarrow$Tl
Po	214	84	130	164.3μs	α
Pb	212	82	130	10.64h	β^-
Bi	212	83	129	60.55m	β^-,64.06%
					α,35.94%
Po	212	84	128	0.299μs	α
Hg	210	80	130	300ns	β^-
Tl	210	81	129	1.30m	$\beta^-\rightarrow{}^{210}$Pb
					$\beta^- n$,3.2×10^{-9}% $\rightarrow{}^{209}$Pb
Pb	210	82	128	22.20y	$\beta^-\rightarrow$Bi
					α,7.0×10^{-3}%$\rightarrow$Hg
Bi	210	83	127	5.012d	$\beta^-\rightarrow$Po
					α,1.3×10^{-4}%$\rightarrow$Tl

元素	質量数	陽子数	中性子数	半減期	崩壊様式
Po	210	84	126	138.376d	α
Pb	209	82	127	3.253h	β^-
Bi	209	83	126		安定
Tl	208	81	127	3.053m	β^-
Pb	208	82	126		安定
Hg	206	80	126	8.32m	β^-
Tl	206	81	125	4.202m	β^-
Pb	206	82	124		安定

4.3.2 β 崩壊系列

核分裂反応で生まれた生成物 (fission product) は同じ質量数の安定核と比較して中性子が過多であるため, β^- 崩壊を起こして中性子を陽子に変換していき, 安定核の陽子：中性子比率に近づこうとする。これが β 崩壊系列であり, 質量数毎に系列が存在しているため, 核分裂生成物に関するものに限っても数十もの系列が存在する。例として, 質量数が 137 の β 崩壊系列を表 4.5 に示す。

文献 [1] では, β 崩壊系列の顕著な特徴として, 総崩壊率が時間に反比例して減少することが挙げられおり, また, その原因が 1 回崩壊する毎に半減期が大幅に増大することにあることが明確に説明されている。

表 4.2: トリウム系列 ($4n$ 系列)。

元素	質量数	陽子数	中性子数	半減期	崩壊様式
Th	232	90	142	1.40×10^{10}y	α
					SF,1.1×10^{-9}%
Ra	228	88	140	5.75y	β^-
Ac	228	89	139	6.15h	β^-
Th	228	90	138	1.9116y	α
					^{20}O,1.0×10^{-11}%
Ra	224	88	136	3.6319d	α
					^{14}C,4.0×10^{-9}%
Rn	220	86	134	55.6s	α
Po	216	84	132	0.145s	α
Pb	212	82	130	10.64h	β^-
Bi	212	83	129	60.55m	β^-,64.06 % →Po
					α,35.94 % →Tl
Po	212	84	128	0.229μs	α
Pb	210	82	128	22.20y	β^- →Bi
					α,1.9×10^{-6}%→Hg
Bi	210	83	127	5.012d	β^- →Po
					α,1.3×10^{-4}%→Tl
Po	210	84	126	138.376d	α
Tl	208	81	127	3.053m	β^-
Pb	208	82	126		安定
Hg	206	80	126	8.32m	β^-
Tl	206	81	125	4.202m	β^-
Pb	206	82	124		安定

表 4.3: アクチニウム系列 ($4n+3$ 系列)。

元素	質量数	陽子数	中性子数	半減期	崩壊様式
U	235	92	143	7.038×10^{8}y	α
					SF,7.0×10^{-9}%
					^{28}Mg,8.0×10^{-10}%
					Ne,8.0×10^{-10}%
Th	231	90	141	25.52h	β^-
					α,4.0×10^{-11}
Pa	231	91	140	3.276×10^{4}y	α
					SF,2.0×10^{-11}%
Ra	227	88	139	42.2m	β^-
Ac	227	89	138	21.772y	β^-, 98.62 %→Th
					α,1.38 % →Fr
Th	227	90	137	18.68d	α
Fr	223	87	136	22.00m	β^-→Ra
					α,(6.0×10^{-3} %)→At
Ra	223	88	135	11.43d	α
					^{14}C,8.9×10^{-8}%
At	219	85	134	56s	α,97.00 % →Bi
					β^- %→Rn
Rn	219	86	133	3.96s	α
Pb	215	82	133	147s	β^-
Bi	215	83	132	7.6m	β^-
Po	215	84	131	1.781ms	α→Pb
					β^-,(2.3×10^{-4} %)→At
At	215	85	130	0.10ms	α
Pb	211	82	129	36.1m	β^-
Bi	211	84	127	2.14m	α,99.72%→Tl
					β −,0.28%→Po
Po	211	84	127	0.516s	α
Pb	209	82	127	3.253h	β^-
Bi	209	83	126		安定
Hg	207	80	127	2.9m	β^-
Tl	207	81	126	4.77m	β^-
Pb	207	82	125		安定

表 4.4: ネプツニウム系列 ($4n+1$ 系列)。

元素	質量数	陽子数	中性子数	半減期	崩壊様式
Np	237	93	144	2.144×10^6y	α
					SF,2.0×10^{-10}%
Pa	233	91	142	26.975d	β^-
U	233	92	141	1.592×10^5y	α
					^{24}Ne,9×10^{-10}%
					SF,6×10^{-11}%
					^{28}Mg,1×10^{-13}%
Th	229	90	139	7392y	α
Ra	225	88	137	14.9d	β^-
Ac	225	89	136	10.0d	α
					^{14}C,4×10^{-12}%
Fr	221	87	134	4.9m	$\alpha \rightarrow$At
					β^-,0.10%→Ra
Ra	221	88	133	28s	β^-
					^{14}C,1×10^{-12}%
At	217	85	132	32.3ms	α,99.99%→Bi
					β^-,7.0×10^{-3}%→Bi
Rn	217	86	131	0.54ms	α
Bi	213	83	130	45.59m	β^-,97.80%→Po
					α,2.20%→Tl
Po	213	84	129	3.72μs	α
Bi	211	83	128	2.14m	α,99.72%→Tl
					β^-,0.28%→Po
Po	211	84	127	0.516s	α
Tl	209	81	128	2.161m	β^-
Pb	209	82	127	3.253h	β^-
Bi	209	83	126		安定
Tl	207	81	126	4.77m	β^-
Pb	207	82	125		安定
Hg	205	80	125	5.14m	β^-
Tl	205	81	124		安定

表 4.5: 質量数が 137 の β 崩壊系列。

元素	質量数	陽子数	中性子数	半減期	崩壊様式
Sn	137	50	87	190ms	β^-
Sb	137	51	86	492ms	β^-
Te	137	52	85	2.49s	β^-
I	137	53	84	24.5s	β^-
Xe	137	54	83	3.818m	β^-
Cs	137	55	82	30.08y	β^-
Ba	137	56	81		

第5章　放射性崩壊の時間発展を記述する連立常微分方程式とその解

本章では原子核の崩壊を表す方程式の一般解を求める。最も単純な場合から複雑な場合まで, 5.1 から 5.5 の 5 通りのケースについて, それぞれ一節を当てて説明する。この 5 通りの場合についての考察を一般化することで, 次章で述べる数十の核種からなるような崩壊系列にも適用できる汎用解法が自然と導かれる。この汎用解法については, 大沼・森近の卒業論文 [1] の 18〜19 頁の 3.2.2 節の数式 (3.10)〜(3.16) により完全に記述されているので, 本論文では説明を繰り返さない。本章執筆の目的は, 汎用解法の典型的な具体例を論じることで, 一般論だけを述べた大沼・森近の卒業論文 [1] を補うような説明を与えることである。

5.1　1種類の原子核の崩壊を表す微分方程式とその一般解

放射性崩壊による原子核の個数の時間的変化について考える。N 個の原子核のうち単位時間に λN 個が崩壊する場合を考えると, N は時間 t に関する 1 階の常微分方程式

$$\frac{dN}{dt} = -\lambda N \tag{5.1}$$

に従う。(5.1) 式における λ は崩壊定数とよばれ, 1 個の原子核が単位時間当たりに崩壊する確率を表している。初期条件は,

$$N(t_0) = N_0 \tag{5.2}$$

ととる。(5.1) 式の形は変数分離型とよばれ, 容易に解を求めることができる。まず, (5.1) 式の両辺を N で割ると下式を得る。

$$\frac{1}{N}\frac{dN}{dt} = -\lambda \tag{5.3}$$

次に (5.3) 式を時間 t に関して $t = t_0$ から $t = t_1$ まで定積分を行う。

$$\int_{t_0}^{t_1} \frac{1}{N}\frac{dN}{dt} = -\lambda \int_{t_0}^{t_1} dt \tag{5.4}$$

(5.4) 式における左辺は置換積分法を用い, 積分定数を t から N に変形する。したがって, 積分区間は $t_0 \leqq t \leqq t_1$ から $N_0 \geqq N \geqq N(t_1)$ に写像される。したがって, (5.4) 式は以下の

ようになる。

$$\begin{aligned}
&\int_{N_0}^{N(t_1)} \frac{1}{N} dN = -\lambda [t]_{t=t_0}^{t=t_1} \\
\Leftrightarrow\ &[\log N]_{N=N_0}^{N=N(t_1)} = -\lambda(t_1 - t_0) \\
\Leftrightarrow\ &\log N(t_1) - \log N_0 = -\lambda(t_1 - t_0) \\
\Leftrightarrow\ &\log N(t_1) = \log N_0 - \lambda(t_1 - t_0)
\end{aligned} \tag{5.5}$$

となる。ここで, (5.5) 式の両辺の exp をとると,

$$\begin{aligned}
&e^{\log N(t_1)} = e^{\log N_0 - \lambda(t_1 - t_0)} \\
\Leftrightarrow\ &N(t_1) = e^{\log N_0} e^{-\lambda(t_1 - t_0)} \\
\Leftrightarrow\ &N(t_1) = N_0 e^{-\lambda(t_1 - t_0)}
\end{aligned} \tag{5.6}$$

となる。次に, (5.6) 式において, $t_1 = t$ と置き換え, さらに式を簡略化するため $t_0 = 0$ とすると解として,

$$N(t) = N_0 e^{-\lambda t} \tag{5.7}$$

が得られる。(5.7) が 1 つの原子核の崩壊の時間変化を表す数式である。

5.2　1 種類の親核から 1 種類の娘核への崩壊

前節の例は, 崩壊の親核しか考慮していないので崩壊系列とは言えない。本節では, 崩壊系列の第一の例として, 最も単純な崩壊系列である 1 種類の親核から 1 種類の娘核への崩壊を取り上げる。具体的には, 親核 ${}^{137}_{55}\mathrm{Cs}_{82}$ が娘核 ${}^{137}_{56}\mathrm{Ba}_{83}$ へと半減期 30.1 年で β^- 崩壊する場合を考える。

${}^{137}_{55}\mathrm{Cs}_{82}$ の個数を $N_1(t)$ 個, ${}^{137}_{56}\mathrm{Ba}_{83}$ の個数を $N_2(t)$ 個とすると, それらの時間発展は下記の 2 本の微分方程式で記述される。

$$\frac{dN_1}{dt} = -\lambda N_1 \tag{5.8}$$

$$\frac{dN_2}{dt} = \lambda N_1 \tag{5.9}$$

初期条件として, $t_0 = 0$, $N_1(0) = N_0$, $N_2(0) = 0$ とする。

(5.8) 式は 5.1 節と同じように変数分離型であるので, 同じように解くことができ, その解は,

$$N_1(t) = N_0 e^{-\lambda t} \tag{5.10}$$

となる。

次に, $\frac{d}{dt}(N_1 + N_2) = 0$ より $N_1 + N_2 =$(一定) となり, 親核と娘核の個数の和は時間変化せず一定値を保つことがわかる。その一定値は $t = 0$ での値 $N_0 + 0 = N_0$ と等しいので,

$$N_2(t) + N_1(t) = N_0 \tag{5.11}$$

となり, (5.10) を代入すると,

$$N_2(t) = N_0(1 - e^{\lambda t}) \tag{5.12}$$

を得る。半減期の $T_{\frac{1}{2}} = 30.1$ 年から崩壊定数を計算すると,

$$\lambda = \frac{\ln 2}{T_{1/2}} \approx \frac{0.693}{30.1y} = 7.30 \times 10^{-10} s^{-1} \tag{5.13}$$

である。

以上の手順を踏んで求まった解 (5.10) 式および (5.12) 式の時間経過をグラフで表したものが図 5.1 である。

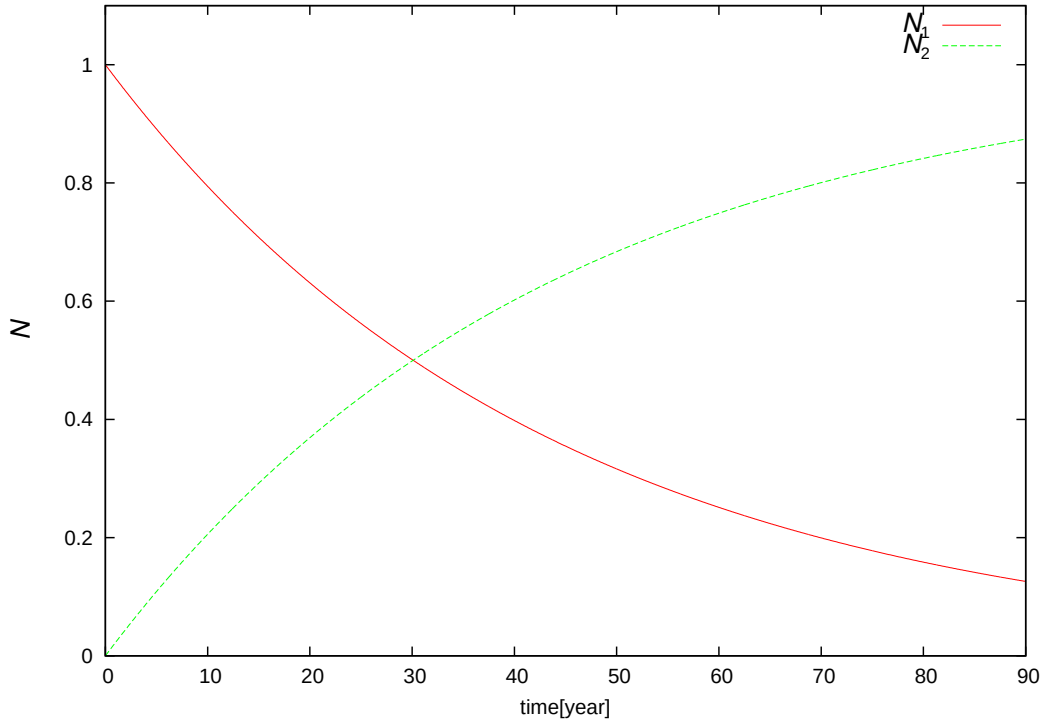

図 5.1: ${}^{137}_{55}Cs_{82}$ の崩壊の時間経過。赤色の実線は親核の個数（初期値を 1 に規格化）, 緑色の破線は娘核の個数を表す。

5.3 親核の崩壊先が 2 つに分岐する場合

第 2 の例として, ${}^{40}_{19}K_{21}$ の場合を考える。この崩壊は枝分かれを起こし, 分岐比が 10.72 %の確率で軌道電子捕獲を起こし ${}^{40}_{18}Ar_{22}$ に崩壊し安定状態になり, 89.28 %の確率で β^- 崩壊し ${}^{40}_{20}Ca_{20}$ に崩壊し安定状態になる。半減期は $T_{\frac{1}{2}} = 1.277 \times 10^9$ 年である。

${}^{40}_{19}K_{21}$ の個数を $N_1(t)$ 個, ${}^{40}_{18}Ar_{22}$ の個数を $N_2(t)$ 個, ${}^{40}_{20}Ca_{20}$ の個数を $N_3(t)$ 個とする。

このときの微分方程式は, それぞれ $N_1(t)$, $N_2(t)$, $N_3(t)$ の時間発展を記述する。

$$\frac{dN_1}{dt} = -\lambda N_1 \tag{5.14}$$

$$\frac{dN_2}{dt} = \lambda_{1\to 2} N_1 \tag{5.15}$$

$$\frac{dN_3}{dt} = \lambda_{1\to 3} N_1 \tag{5.16}$$

となる。初期条件として, $t_0 = 0$, $N_1(0) = N_0$, $N_2(0) = N_3(0) = 0$ とする。

(5.14) 式は (5.1) 式と同様に変数分離型であるため, 同じように解くことができ, その解は,

$$N_1(t) = N_0 e^{-\lambda t} \tag{5.17}$$

となる。また, $N_2(t)$ は,

$$\begin{aligned} N_2(t) &= \int_0^t \frac{dN_2(t)}{dt} dt + N_2(0) \\ &= \lambda_{1\to 2} \int_0^t \frac{dN_1(t)}{dt} dt \end{aligned} \tag{5.18}$$

となる。おなじく, $N_3(t)$ は,

$$\begin{aligned} N_3(t) &= \int_0^t \frac{dN_3(t)}{dt} dt + N_3(0) \\ &= \lambda_{1\to 3} \int_0^t \frac{dN_1(t)}{dt} dt \end{aligned} \tag{5.19}$$

となる。

ここで, $\lambda_{1\to 2}$, $\lambda_{1\to 3}$ は, 部分崩壊定数といい,

$$\lambda_{1\to 2} = \lambda_1 \times 0.1072 \tag{5.20}$$

$$\lambda_{1\to 3} = \lambda_1 \times 0.8928 \tag{5.21}$$

と定義される。 (5.18) 式, (5.19) 式により, $N_2(t)$ および $N_3(t)$ の割合は,

$$N_2(t) : N_3(t) = \lambda_{1\to 2} : \lambda_{1\to 3} \tag{5.22}$$

となる。

ここで, $\frac{d}{dt}(N_1 + N_2 + N_3) = 0$ より, $N_1 + N_2 + N_3 =$ 一定であるから, 3 核種の個数の和は時間変化せず一定値 N_0 を保つ。よって, それぞれの核種の個数の時間変化を表す解は,

$$N_1(t) = N_0 e^{\lambda t} \tag{5.23}$$

$$N_2(t) = \frac{\lambda_{1\to 2}}{\lambda_1}\{N_0 - N_1(t)\} \tag{5.24}$$

$$= N_0 \frac{\lambda_{1\to 2}}{\lambda_1} e^{\lambda t} \tag{5.25}$$

$$N_3(t) = \frac{\lambda_{1\to 3}}{\lambda_1}\{N_0 - N_1(t)\} \tag{5.26}$$

$$= N_0 \frac{\lambda_{1\to 3}}{\lambda_1} e^{\lambda t} \tag{5.27}$$

となる。

以上のようにして求まった解の (5.23) 式, (5.25) 式, (5.27) 式の時間変化をグラフに表したものが図 5.2 である。

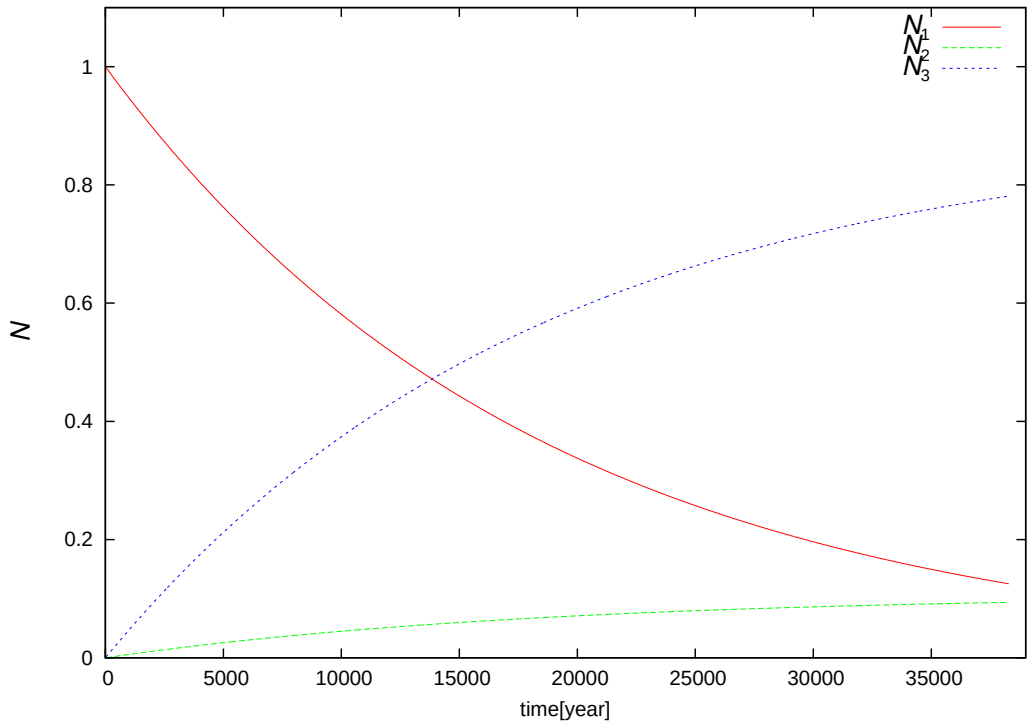

図 5.2: ${}^{40}_{19}\mathrm{K}_{21}$ の崩壊の時間変化。赤色の実線は ${}^{40}_{19}\mathrm{K}_{21}$ の個数（時刻 0 で 1 に規格化), 緑色の破線は ${}^{40}_{18}\mathrm{Ar}_{22}$ の個数, 青色の点線は ${}^{40}_{20}\mathrm{Ca}_{20}$ の個数を表す。

5.4 親核が娘核, 娘核が孫核へと 2 段階にわたり崩壊する場合

第 3 の例として, ${}^{90}_{38}\mathrm{Sr}_{52}$ を出発点とする崩壊系列を取り上げる。この崩壊は 2 段階にわたり崩壊を起こし, まず ${}^{90}_{38}\mathrm{Sr}_{52}$ が半減期 $T_{\frac{1}{2}} = 28.79$ 年で ${}^{90}_{39}\mathrm{Y}_{51}$ に β^- 崩壊し, 次に ${}^{90}_{39}\mathrm{Y}_{51}$ が半減期 $T_{\frac{1}{2}} = 64.10$ 時間で ${}^{90}_{40}\mathrm{Zr}_{50}$ に β^- 崩壊し安定核となる。

時刻 t における ${}^{90}_{38}\mathrm{Sr}_{52}$ の個数を $N_1(t)$ 個, ${}^{90}_{39}\mathrm{Y}_{51}$ の個数を $N_2(t)$ 個, ${}^{90}_{40}\mathrm{Zr}_{50}$ の個数を $N_3(t)$ 個とする。初期値をそれぞれ $N_1(0) = N_0$, $N_2(0) = 0$, $N_3(0) = 0$ とする。また, 崩壊定数は 1 段階目を λ_1, 2 段階目を λ_2 とすると,

$$\lambda_1 = \frac{\ln 2}{T_{1/2}} \approx \frac{0.693}{28.79y} = 7.269 \times 10^{-10} s^{-1} \tag{5.28}$$

および

$$\lambda_2 = \frac{\ln 2}{T_{1/2}} \approx \frac{0.693}{64.10h} = 3.004 \times 10^{-6} s^{-1} \tag{5.29}$$

となる。

N_1, N_2, N_3 の従う連立 1 階常微分方程式は, λ_1, λ_2 を使って以下のように表される。

$$\frac{d}{dt}\begin{pmatrix} N_1(t) \\ N_2(t) \\ N_3(t) \end{pmatrix} = \begin{pmatrix} -\lambda_1 & 0 & 0 \\ \lambda_1 & -\lambda_2 & 0 \\ 0 & \lambda_2 & 0 \end{pmatrix}\begin{pmatrix} N_1(t) \\ N_2(t) \\ N_3(t) \end{pmatrix} \tag{5.30}$$

あるいは, $\begin{pmatrix} N_1(t) \\ N_2(t) \\ N_3(t) \end{pmatrix} = \vec{n}(t), \begin{pmatrix} -\lambda_1 & 0 & 0 \\ \lambda_1 & -\lambda_2 & 0 \\ 0 & \lambda_2 & 0 \end{pmatrix} = \Lambda$ とおくと, (5.30) 式は

$$\frac{d}{dt}\vec{n}(t) = \Lambda\vec{n}(t) \tag{5.31}$$

とも表すことができる。

これの解は

$$\vec{n}(t) = e^{\Lambda t}\vec{n}(0) \tag{5.32}$$

とかける。$e^{\Lambda t}$ を求めるには, まず Λ の固有値を求める。行列 Λ の固有値を ϵ, 固有ベクトルを $\vec{v}$ とすると, 固有値方程式 $\Lambda\vec{v} = \epsilon\vec{v}$ が成り立つ。

行列 Λ の特性方程式 (または固有方程式ともいう) は固有値を ϵ, I を単位行列とすると

$$\det(\epsilon I - \Lambda) = 0 \tag{5.33}$$

である。したがって,

$$\det\begin{pmatrix} \epsilon + \lambda_1 & 0 & 0 \\ -\lambda_1 & \epsilon + \lambda_2 & 0 \\ 0 & -\lambda_2 & 0 \end{pmatrix} = (\epsilon + \lambda_1)(\epsilon + \lambda_2)\epsilon = 0 \tag{5.34}$$

となるので, $\epsilon = -\lambda_1, -\lambda_2, 0$ が固有値である。$\epsilon_1 = -\lambda_1$, $\epsilon_2 = -\lambda_2$, $\epsilon_3 = 0$ と書くことにする。

固有値 $\epsilon_i (1 \leq i \leq 3)$ に対応する固有ベクトル $\vec{v}_i = \begin{pmatrix} v_{i1} \\ v_{i2} \\ v_{i3} \end{pmatrix}$ は $\Lambda\vec{v}_i = \epsilon_i\vec{v}_i$ を満たすゼロベクトルではないベクトルである。それらを求めた結果は

$$\vec{v}_1 = \begin{pmatrix} 1 \\ \frac{-\lambda_1}{\lambda_1 - \lambda_2} \\ \frac{\lambda_2}{\lambda_1 - \lambda_2} \end{pmatrix}, \quad \vec{v}_2 = \begin{pmatrix} 0 \\ 1 \\ -1 \end{pmatrix}, \quad \vec{v}_2 = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \tag{5.35}$$

である。なお, これらにゼロでない任意の定数を乗じたものもやはり固有ベクトルである。

行列 P を $P = (\vec{v_1} \quad \vec{v_2} \quad \vec{v_3})$, 即ち $P_{ij} = v_{ji}$ と定義する。$\det P = 1 \neq 0$ なので, 逆行列 P^{-1} が存在する。

$$P^{-1}P = I = (\vec{e_1}, \vec{e_2}, \vec{e_3}) \tag{5.36}$$

であるから (ただし $\vec{e_1} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \vec{e_2} = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}, \vec{e_3} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$ とした。)

$$P^{-1}P = (P^{-1}\vec{v_1}, P^{-1}\vec{v_2}, P^{-1}\vec{v_3}) = (\vec{e_1}, \vec{e_2}, \vec{e_3}) \tag{5.37}$$

となる。したがって, $P^{-1}\vec{v_i} = \vec{e_i}(1 \leq i \leq 3)$ が成立する。これを用い相似変換を行うと,

$$\begin{aligned} P^{-1}\Lambda P &= P^{-1}(\Lambda\vec{v_1}, \Lambda\vec{v_2}, \Lambda\vec{v_3}) \\ &= P^{-1}(\epsilon_1\vec{v_1}, \epsilon_2\vec{v_2}, \epsilon_3\vec{v_3}) \\ &= (\epsilon_1 P^{-1}\vec{v_1}, \epsilon_2 P^{-1}\vec{v_2}, \epsilon_3 P^{-1}\vec{v_3}) \\ &= (\epsilon_1\vec{e_1}, \epsilon_2\vec{e_2}, \epsilon_3\vec{e_3}) \\ &= \begin{pmatrix} \epsilon_1 & 0 & 0 \\ 0 & \epsilon_2 & 0 \\ 0 & 0 & \epsilon_3 \end{pmatrix} \end{aligned} \tag{5.38}$$

となり, 対角行列であることが示せる。$D = P^{-1}\Lambda P$ と書けば両辺に左から P, 右から P^{-1} を乗じて $\Lambda = PDP^{-1}$ であるから, $n \leq 0$ に対し,

$$\begin{aligned} \Lambda^n &= (PDP^{-1})^n \\ &= PDP^{-1}PDP^{-1}\cdots PDP^{-1} \\ &= PD(P^{-1}P)D(P^{-1}P)\cdots(P^{-1}P)DP^{-1} \\ &= PDIDI\cdots IDP^{-1} \\ &= PD^nP^{-1} \end{aligned} \tag{5.39}$$

$$\tag{5.40}$$

および

$$D^n = \begin{pmatrix} {\epsilon_1}^n & 0 & 0 \\ 0 & {\epsilon_2}^n & 0 \\ 0 & 0 & {\epsilon_3}^n \end{pmatrix} \tag{5.41}$$

が成立するので,

$$\begin{aligned} e^{\Lambda t} &= \sum_{n=0}^{\infty} \frac{1}{n!}\Lambda^n t^n \\ &= P\begin{pmatrix} \sum_{n=0}^{\infty}\frac{1}{n!}(\epsilon_1 t)^n & 0 & 0 \\ 0 & \sum_{n=0}^{\infty}\frac{1}{n!}(\epsilon_2 t)^n & 0 \\ 0 & 0 & \sum_{n=0}^{\infty}\frac{1}{n!}(\epsilon_3 t)^n \end{pmatrix} P^{-1} \\ &= \begin{pmatrix} e^{{\epsilon_1}^t} & 0 & 0 \\ 0 & e^{{\epsilon_2}^t} & 0 \\ 0 & 0 & e^{{\epsilon_3}^t} \end{pmatrix} \end{aligned} \tag{5.42}$$

$$\tag{5.43}$$

となる。

したがって, 解は $\vec{n}(t) = e^{\Lambda t}\vec{n}(0)$ であるから,

$$\begin{pmatrix} N_1(t) \\ N_2(t) \\ N_3(t) \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ \frac{-\lambda_1}{\lambda_1-\lambda_2} & 1 & 0 \\ \frac{\lambda_2}{\lambda_1-\lambda_2} & -1 & 1 \end{pmatrix} \begin{pmatrix} e^{-\lambda_1 t} & 0 & 0 \\ 0 & e^{-\lambda_2 t} & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ \frac{\lambda_1}{\lambda_1-\lambda_2} & 1 & 0 \\ 1 & -1 & 1 \end{pmatrix} \begin{pmatrix} N_0 \\ 0 \\ 0 \end{pmatrix}$$

$$= \begin{pmatrix} N_0 e^{-\lambda_1 t} \\ N_0 \frac{\lambda_1}{\lambda_1-\lambda_2}(e^{-\lambda_2 t} - e^{-\lambda_1 t}) \\ N_0 \left(\frac{\lambda_2}{\lambda_1-\lambda_2} e^{-\lambda_1 t} - \frac{\lambda_1}{\lambda_1-\lambda_2} e^{-\lambda_2 t} + 1 \right) \end{pmatrix} \tag{5.44}$$

(5.45)

を得る。

以上の計算を経て得られた解の (5.44) 式の表す時間変化をグラフで表したものが図 5.3 である。しかし, 図 5.3 では $N_2(t)$ の値が小さすぎて読み取れない。その原因は娘核の半減期が, 親核の半減期や孫核の半減期（無限大）と比較して桁違いに短いからである。そこで, 両軸を対数目盛で表したグラフも図 5.4 として示しておく。

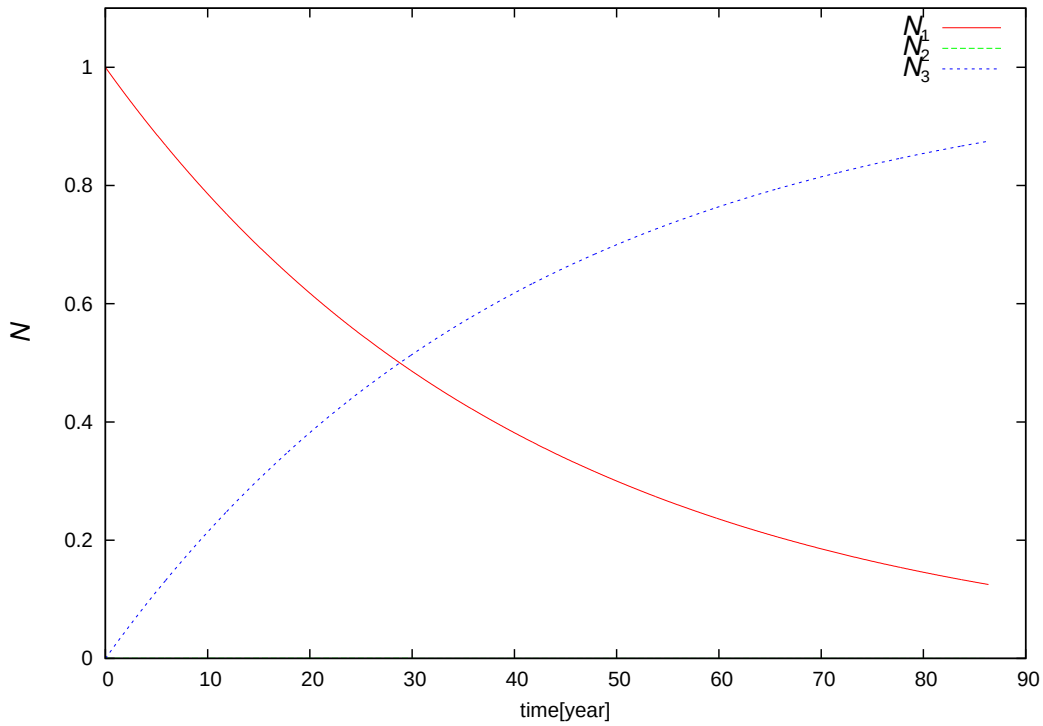

図 5.3: ${}^{90}_{38}\mathrm{Sr}_{52}$ を起点とする崩壊系列の時間変化。赤色の実線は ${}^{90}_{38}\mathrm{Sr}_{52}$ の個数（時刻 0 で 1 に規格化), 緑色の破線は ${}^{90}_{39}\mathrm{Y}_{51}$ の個数, 青色の点線は ${}^{90}_{40}\mathrm{Zr}_{50}$ の個数を表す。

5.5 親核が複数の経路を経て同一の娘核に崩壊する場合

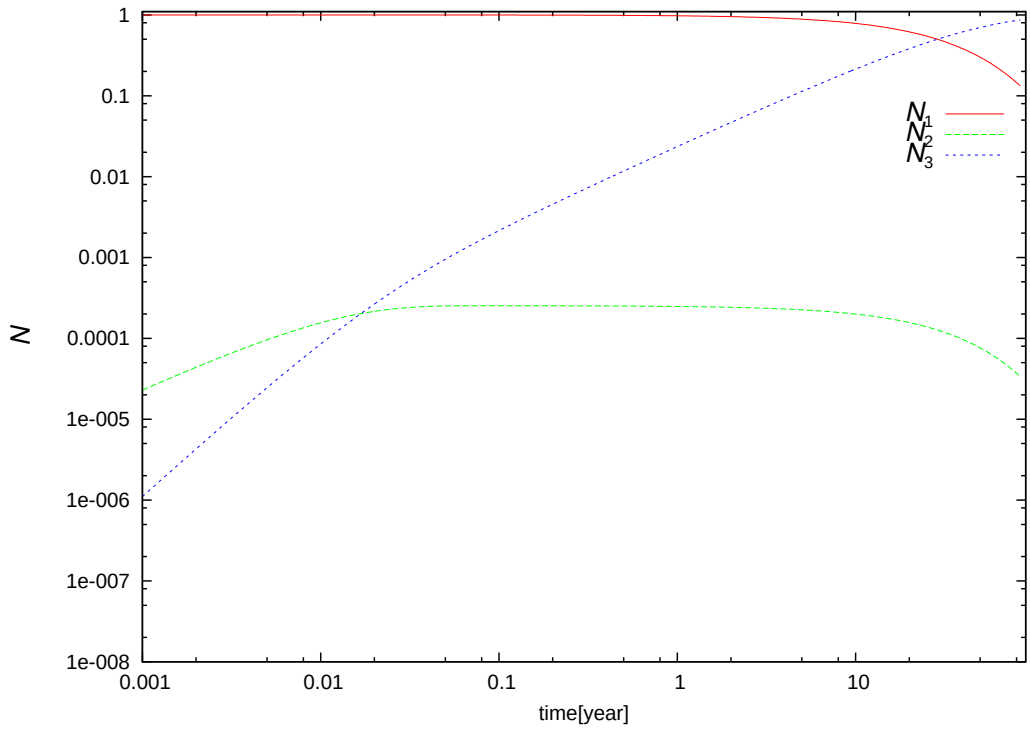

図 5.4: ${}^{90}_{38}\mathrm{Sr}_{52}$ を起点とする崩壊系列の時間経過。図 5.3 を両対数スケールで表示しなおしたものである。

第 4 の例として, 第 1 の例としてとりあげた ${}^{137}_{55}\mathrm{Cs}_{82}$ の場合を詳細に考えなおす。実は, この崩壊は半減期 $T_{\frac{1}{2}} = 30.08$ 年で 5.4%の確率で β^- 崩壊を起こし ${}^{137}_{56}\mathrm{Ba}_{81}$ に崩壊し安定核になるか, 94.6%の確率で β^- 崩壊を起こし ${}^{137}_{56}\mathrm{Ba}_{81}$ の励起状態である ${}^{137\mathrm{m}}_{56}\mathrm{Ba}_{81}$ に崩壊し, その後半減期 $T_{\frac{1}{2}} = 2.55$ 分で ${}^{137}_{56}\mathrm{Ba}_{81}$ に崩壊し安定核になる 2 通りの経路を経る。

実際には多くの β 崩壊においてこのようなことが起きている。即ち,β 崩壊のほとんどで, 親核の基底状態からの遷移が娘核の多くの励起状態に分岐して起こり, その直後に励起状態から基底状態への γ 崩壊が起きているのである。しかし,β 崩壊の半減期が γ 崩壊の半減期よりはるかに長く, かつ, γ 崩壊の半減期程度の時間で起こる変動を無視してよい場合は, 親核が 100%直接に娘核の基底状態へ β 崩壊すると近似しても支障ないのである。本例はそのような近似的扱いの妥当性の数値的な確認としても役立つ。

時刻 t における ${}^{137}_{55}\mathrm{Cs}_{82}$ の個数を $N_1(t)$ 個, ${}^{137\mathrm{m}}_{56}\mathrm{Ba}_{81}$ の個数を $N_2(t)$ 個, ${}^{137}_{56}\mathrm{Ba}_{81}$ の個数を $N_3(t)$ 個とする。初期値をそれぞれ $N_1(0) = N_0$, $N_2(0) = 0$, $N_3(0) = 0$ とする。また, 崩壊定数を λ_1, λ_2, 同じく部分崩壊定数を $\lambda_{1\to2}$, $\lambda_{1\to3}$ とすると, それぞれ

$$\lambda_1 = \frac{\ln 2}{T_{1/2}} \approx \frac{0.693}{30.09y} = 7.298 \times 10^{-10} s^{-1} \tag{5.46}$$

$$\lambda_2 = \frac{\ln 2}{T_{1/2}} \approx \frac{0.693}{2.55m} = 4.53 \times 10^{-3} s^{-1} \tag{5.47}$$

$$\lambda_{1\to2} = \lambda_1 \times 0.946 = 6.904 \times 10^{-10} s^{-1} \tag{5.48}$$

$$\lambda_{1\to3} = \lambda_1 \times 0.054 = 3.94 \times 10^{-11} s^{-1} \tag{5.49}$$

となる。

$N_1(0) = N_0, N_2(0) = 0, N_3(0) = 0$ の従う連立 1 階常微分方程式を考えると,

$$\frac{d}{dt}\begin{pmatrix} N_1(t) \\ N_2(t) \\ N_3(t) \end{pmatrix} = \begin{pmatrix} -\lambda_1 & 0 & 0 \\ \lambda_{1\to 2} & -\lambda_2 & 0 \\ \lambda_{1\to 3} & \lambda_2 & 0 \end{pmatrix}\begin{pmatrix} N_1(t) \\ N_2(t) \\ N_3(t) \end{pmatrix} \tag{5.50}$$

となる。$\begin{pmatrix} N_1(t) \\ N_2(t) \\ N_3(t) \end{pmatrix} = \vec{n}(t), \begin{pmatrix} -\lambda_1 & 0 & 0 \\ \lambda_{1\to 2} & -\lambda_2 & 0 \\ \lambda_{1\to 3} & \lambda_2 & 0 \end{pmatrix} = \Lambda$ とおくと, (5.50) 式は

$$\frac{d}{dt}\vec{n}(t) = \Lambda\vec{n}(t) \tag{5.51}$$

とも表すことができる。

行列 Λ は三角行列なので, その固有値は対角要素 $\lambda_1, \lambda_2, 0$ である。以下では固有値が縮退していない (即ちすべて値が異なる) と仮定して解を求める。

形式的な解は

$$\vec{n}(t) = e^{\Lambda t}\vec{n}(0) \tag{5.52}$$

である。固有値に縮退がないので Λ は相似変換で対角行列にできる。

即ち, $P^{-1}\Lambda P = D = \begin{pmatrix} -\lambda_1 & 0 & 0 \\ 0 & -\lambda_2 & 0 \\ 0 & 0 & 0 \end{pmatrix}$ を満たす行列 P が存在する。このとき,

$$e^{\Lambda t} = Pe^{Dt}P^{-1} = P\begin{pmatrix} e^{-\lambda_1 t} & 0 & 0 \\ 0 & e^{-\lambda_2 t} & 0 \\ 0 & 0 & 0 \end{pmatrix}P^{-1} \tag{5.53}$$

となる。したがって, $\vec{n_i}(t)(1 \le i \le 3)$ は, $e^{-\lambda_1 t}, e^{-\lambda_2 t}, 1$ の線形結合となるので,

$$n_i(t) = C_{i1}e^{-\lambda_1 t} + C_{i2}e^{-\lambda_2 t} + C_{i3}(1 \le i \le 3) \tag{5.54}$$

とおける。9 係数 $C_{ij}(1 \le i \le 3, 1 \le j \le 3)$ は,

$$\frac{d}{dt}\vec{n} = \Lambda\vec{n} \tag{5.55}$$

から得られる 9 条件のうち, 詳しく考察すると独立なものが 6 条件あり, これと初期条件の 3 条件とから定めることができる。

以上のようにして得られた解は

$$N_1(t) = N_0e^{-\lambda_1 t} \tag{5.56}$$

$$N_2(t) = N_0\frac{\lambda_{1\to 2}}{\lambda_1 - \lambda_2}(e^{-\lambda_2 t} - e^{-\lambda_1 t}) \tag{5.57}$$

$$N_3(t) = N_0\left\{1 - \left(1 - \frac{\lambda_{1\to 2}}{\lambda_1 - \lambda_2}\right)e^{-\lambda_1 t} - \frac{\lambda_{1\to 2}}{\lambda_1 - \lambda_2}e^{-\lambda_2 t}\right\} \tag{5.58}$$

である。

したがって, 以上の事柄を踏まえ (5.56) 式, (5.57) 式, (5.58) 式の時間経過をグラフで表したものが図 5.5 となる。しかし, 5.4 節と同じように図 5.5 では $N_3(t)$ の値が小さすぎるため確認できない。したがって, 両軸を対数目盛で表したものを図 5.6 に示す。

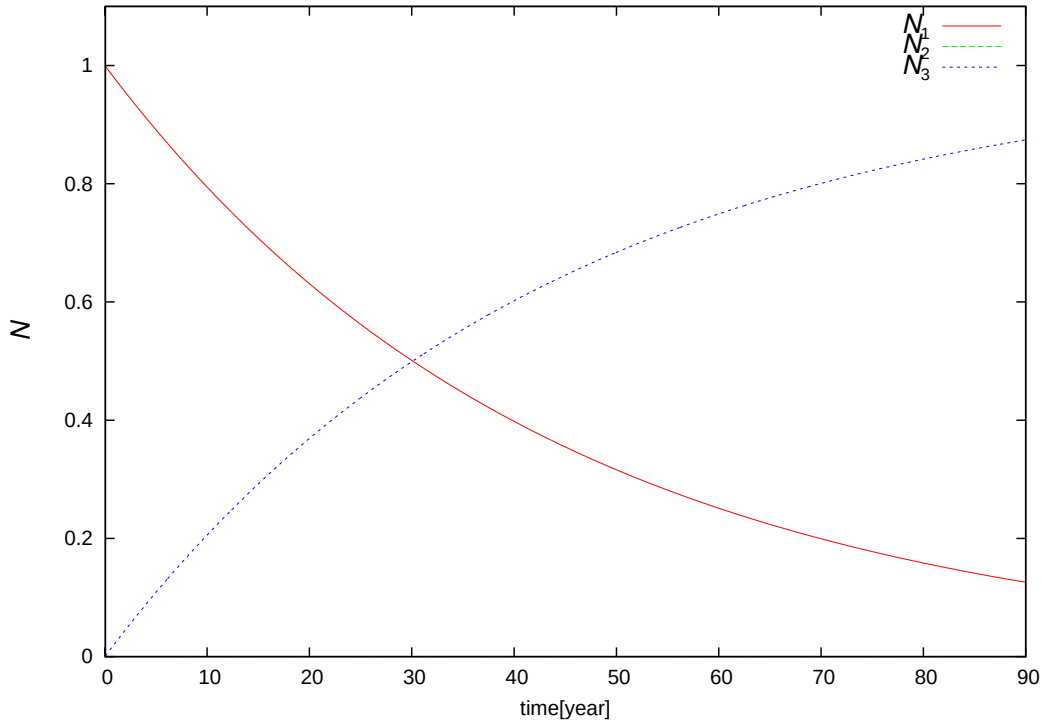

図 5.5: $^{137}_{55}\mathrm{Cs}_{82}$ からの崩壊の時間変化。赤色の実線は $^{137}_{55}\mathrm{Cs}_{82}$ の個数（時刻 0 で 1 に規格化), 緑色の破線は $^{137\mathrm{m}}_{56}\mathrm{Ba}_{81}$ の個数（本グラフではほとんど水平軸に重なるため描かれていない）, 青色の点線は $^{137}_{56}\mathrm{Ba}_{81}$ の個数を表す。

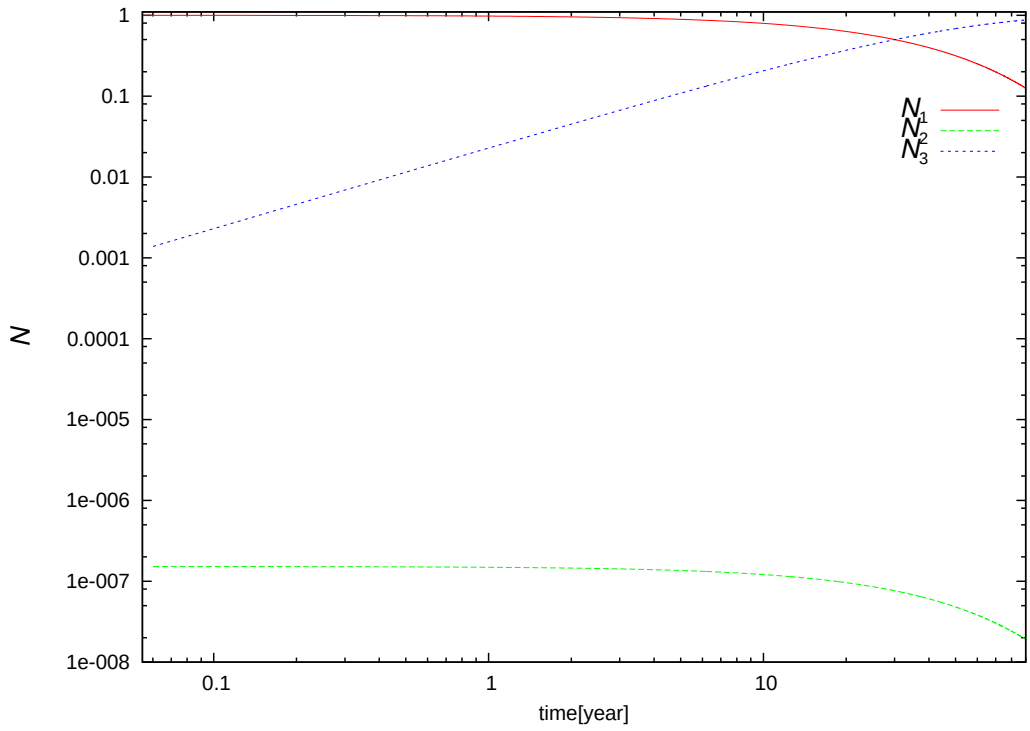

図 5.6: $^{137}_{55}Cs_{82}$ からの崩壊の時間変化。図 5.5 を両対数スケールで描き直したものである。

第6章　計算結果

6.1　数値計算プログラム作成の概略

前章で論じた4通りの具体例での解の求め方を一般化することで, 任意の崩壊系列の時間変化を表す連立常微分方程式の解法を作り上げることが可能であり, 実際に, その結果は, 大沼・森近の卒業論文 [1] の 18〜19 頁の 3.2.2 節の数式 (3.10)〜(3.16) に示されている。なお, 大沼・森近の卒業論文では, 半減期に偶然縮退がある場合には縮退をわずかに解くという近似法が使用されたが, 本研究の対象とする崩壊系列では偶然縮退はなかったので, この近似法は使用されていない。

大沼・森近はその一般解法を C 言語プログラムとして実装したが, 本論文では, それをプログラムの中核としてそのまま利用し, そこに新たに取り組もうとしている崩壊系列上に存在する核種の半減期と崩壊の分岐比のデータを設定するサブルーチン (C 言語の関数) を付け加えて, 重い原子核のアルファ崩壊系列に適用できるようにした。付録には, そのようにして拡張した C 言語のプログラムの全ソースコードを公開した。

大沼・森近の卒業論文 [1] に掲載されたプログラムでは, 半減期・分岐比等のデータ設定は, 質量数の値毎に存在する β 崩壊系列それぞれについて, (5.31) 式で定義された行列 Λ の行列要素に値を代入するサブルーチンを作り, それを呼び出すことで行われていた。本研究でのプログラム追加に当たっては, より本格的なプログラムに仕立て直すために, プログラムが入力データファイルを読み込むという形式に変更することも検討した。しかし, プログラミング作業のほとんどがデータファイルの入力作業になってしまい, C 言語の使用経験を積みたいという卒業研究の副目的のひとつにあまり適さないという短所がともなった。これに対して, 大沼・森近の卒業論文のように, データを C 言語プログラムのステートメントとして記述すれば, データの入力も, C 言語のプログラミングとして行われることになるので, C 言語に触れている時間は遥かに長くなる。そこで, 本卒業研究でも, データを C 言語のステートメントとして記述する方法を踏襲することに決定した。

本研究でのデータ入力ステートメントはステートメント間のデータ整合性を, 部分的にだが, 自動的に保証するために C 言語のマクロ機能を使うことにした。このため, 大森・森近の卒業論文と較べてやや抽象化している。しかし, それが C 言語のプログラムであることを意識することなくしては, 正しいデータ設定サブルーチンを作り上げることは決してできない。したがって, C 言語によるプログラミング経験を積むという副目的は本卒業研究を通じてかなり果たすことができたという実感を抱いている。

6.2 α 崩壊系列の簡単な計算例

実際の α 崩壊系列について論じる前に, 簡単な例を用いて説明する。原子核 A,B,C,D,E が,A→B→C→D→E の経路で崩壊する崩壊系列を考える。半減期は,

$T_{\frac{1}{2}}$(A) = 10^7 秒, $T_{\frac{1}{2}}$(B) = 10^2 秒, $T_{\frac{1}{2}}$(C) = 10^0 秒, $T_{\frac{1}{2}}$(D) = 10^4 秒, $T_{\frac{1}{2}}$(E) = ∞（安定核）

と設定する。

これらの原子核の個数を $N_{\mathrm{A}} \sim N_{\mathrm{E}}$ とすると, 崩壊過程を表す微分方程式は下記のようになる。

$$\frac{dN_{\mathrm{A}}}{dt} = -\lambda_{\mathrm{A}} N_{\mathrm{A}} \tag{6.1}$$

$$\frac{dN_{\mathrm{B}}}{dt} = \lambda_{\mathrm{A}} N_{\mathrm{A}} - \lambda_{\mathrm{B}} N_{\mathrm{B}} \tag{6.2}$$

$$\frac{dN_{\mathrm{C}}}{dt} = \lambda_{\mathrm{B}} N_{\mathrm{B}} - \lambda_{\mathrm{C}} N_{\mathrm{C}} \tag{6.3}$$

$$\frac{dN_{\mathrm{D}}}{dt} = \lambda_{\mathrm{C}} N_{\mathrm{C}} - \lambda_{\mathrm{D}} N_{\mathrm{D}} \tag{6.4}$$

$$\frac{dN_{\mathrm{E}}}{dt} = \lambda_{\mathrm{D}} N_{\mathrm{D}} \tag{6.5}$$

上式は定数係数連立 1 階線形常微分方程式なので一般解を求めることができる。

図 6.1 は $N_{\mathrm{A}} \sim N_{\mathrm{E}}$ の崩壊の時間変化を両対数スケールでプロットしたものである。但し, 時刻ゼロにおける起点の核の個数を 1 個に規格化した。図 6.2 は, 放射平衡とそれに至る前の過渡過程についての説明を図 6.1 に書き込んだものである。まず, 放射平衡が起きる前については, 黒色の矢印の部分が傾きが 1, オレンジ色の矢印の部分の傾きが 2, 水色の部分の矢印の傾きが 3 となっている。理由は以下の通りである。先程の方程式 (6.2) において,$\lambda_{\mathrm{B}} N_{\mathrm{B}}$ が小さいうちは,

$$\frac{N_{\mathrm{B}}}{dt} = \lambda_{\mathrm{A}} N_{\mathrm{A}} \tag{6.6}$$

と考えることができ,$\lambda_A N_A$ を定数とみなすことができる。したがって,

$$N_{\mathrm{B}} = \lambda_{\mathrm{A}} N_{\mathrm{A}} t \tag{6.7}$$

となり,t の 1 乗に比例することがわかる。同じように考えると,N_{C} は $\lambda_{\mathrm{C}} N_{\mathrm{C}}$ が小さいうちは t の 2 乗に比例,N_{D} は $\lambda_{\mathrm{D}} N_{\mathrm{D}}$ は t の 3 乗に比例していることがわかる。

次に, 紫色の矢印の区間は安定核である E の値を除いて値の変わらないプラトー領域となっており, このプラトー領域を含め以後で放射平衡 (secular equilibrium in radioactive decay) が起きている。

放射平衡とは, 親核と娘核の量的な関係が時間的にほぼ一定の割合で推移することであり, 親核のほうが娘核より半減期が長いときに起こる状態である。

続いて, 放射能の量で表したものが図 6.3 になる。こちらでも, 放射能の量が同じ値で

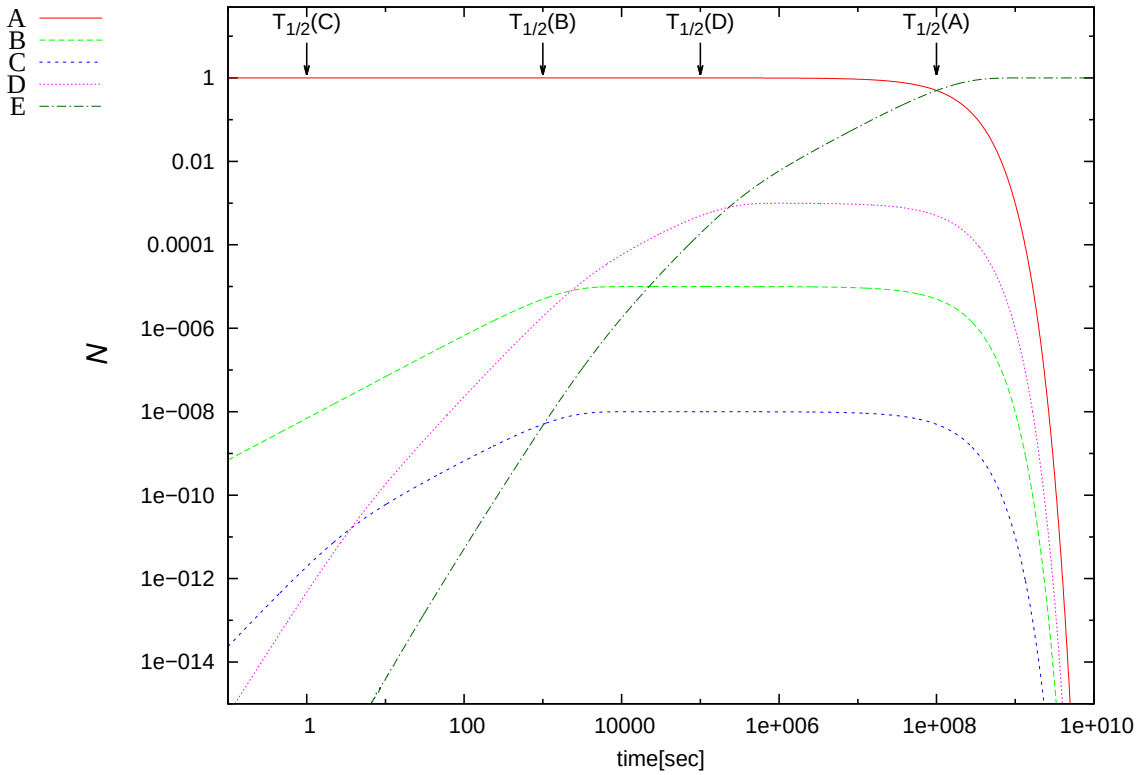

図 6.1: 原子核 A~E の時間変化。横軸は経過時間（秒）, 縦軸は核種の個数（時刻ゼロでの A の個数を 1 に規格化した）を表す。

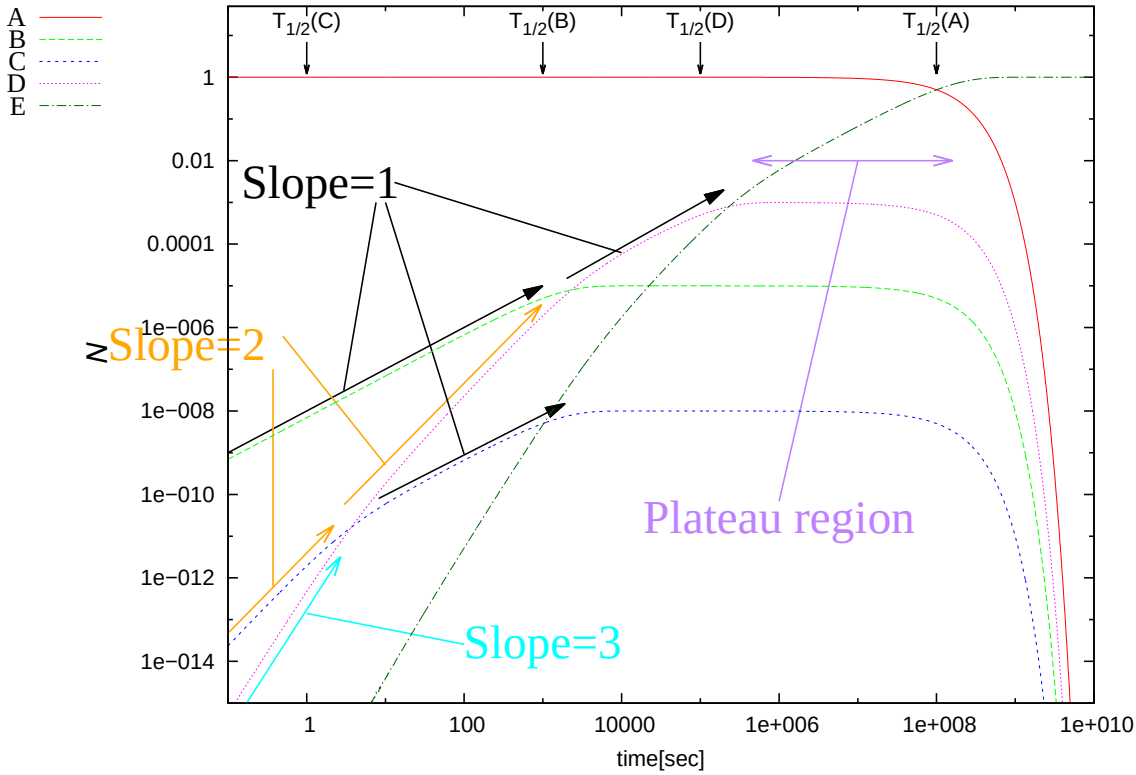

図 6.2: 原子核 A~E の時間変化。横軸は経過時間（秒）, 縦軸は核種の個数（時刻ゼロでの A の個数を 1 に規格化した）を表す。

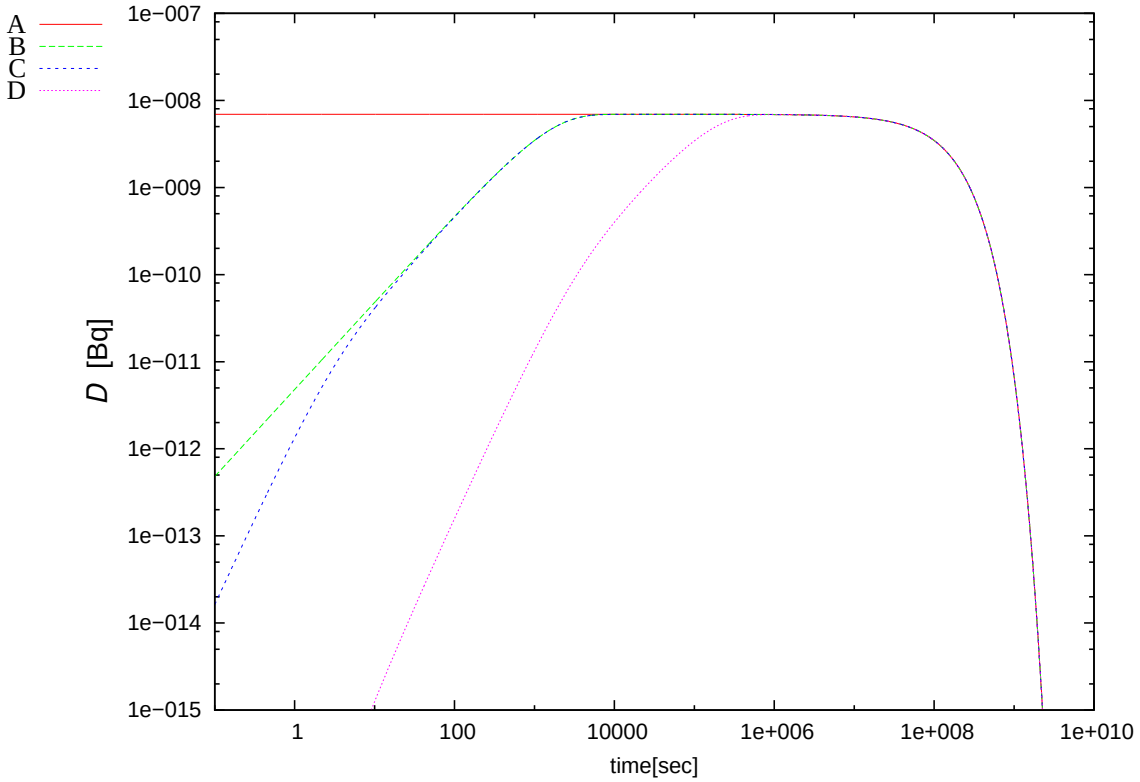

図 6.3: 原子核 A~E の時間変化。横軸は経過時間（秒）, 縦軸は放射能の量（ベクレル）を表す。

減少しており, 放射平衡が起こっていることがわかる。

さて, このような長寿命核種を起点とする α 崩壊系列は, ダムが途中に存在する川の流れに例えることができる。図 6.4 のようにダム A~D が存在する川があると考える。ただし, 一本の川とし, 途中の合流はないものとし, ダム D の流出先は海（安定核）と考える。

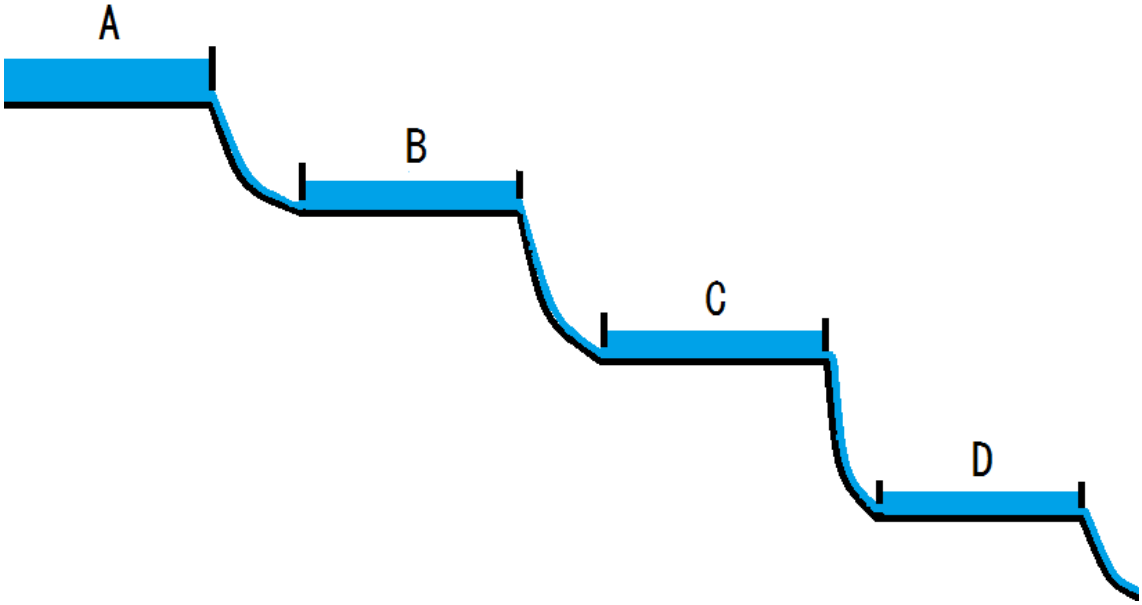

図 6.4: 川およびダムを図示したもの。上流からダムを A~D としている。

まず, 放射平衡が起こる前は, ダムに流入する水の量が流出する量よりも多いため, ダムに水がたまっていく。プラトー領域のところでは, ダムに流入する水の量と流出する水の量が同じとなるため, 水位は変わらない。それ以後では流出する量が流入する量よりも多いため, ダム内の水は減っていく。但し, 減っていく割合はそれぞれ同じである。

6.3 超長寿命核種を起点とする放射平衡

ここから実際の α 崩壊系列について論じていく。自然に存在する α 崩解系列の源には, 系列上の他の核種より桁違いに長い半減期を持ったウランないしトリウム同位体が存在している。まず最初に, それらの長寿命核種を源とする崩壊系列の下流の時間変化を計算する。α 崩壊系列は質量数が 4 だけ異なる核種が連なって作られるのであるから, 4 章で説明した通り, トリウム系列 ($A = 4n$), ネプツニウム系列 ($A = 4n + 1$), ウラン系列 ($A = 4n + 2$), アクチニウム系列 ($A = 4n + 3$) の 4 系列が存在する。

図 6.5 は半減期 140 億年の ^{232}Th を起点とするトリウム系列の崩壊の時間変化を両対数スケールでプロットしたものである。横軸は経過時間（秒）, 縦軸は崩壊系列上の核種の個数（時刻ゼロにおける起点の核の個数を 1 個に規格化した）を表している。10^9~10^{17} 秒の間に, 安定核である終点の ^{208}Pb および核分裂生成物 (fission product, FP)

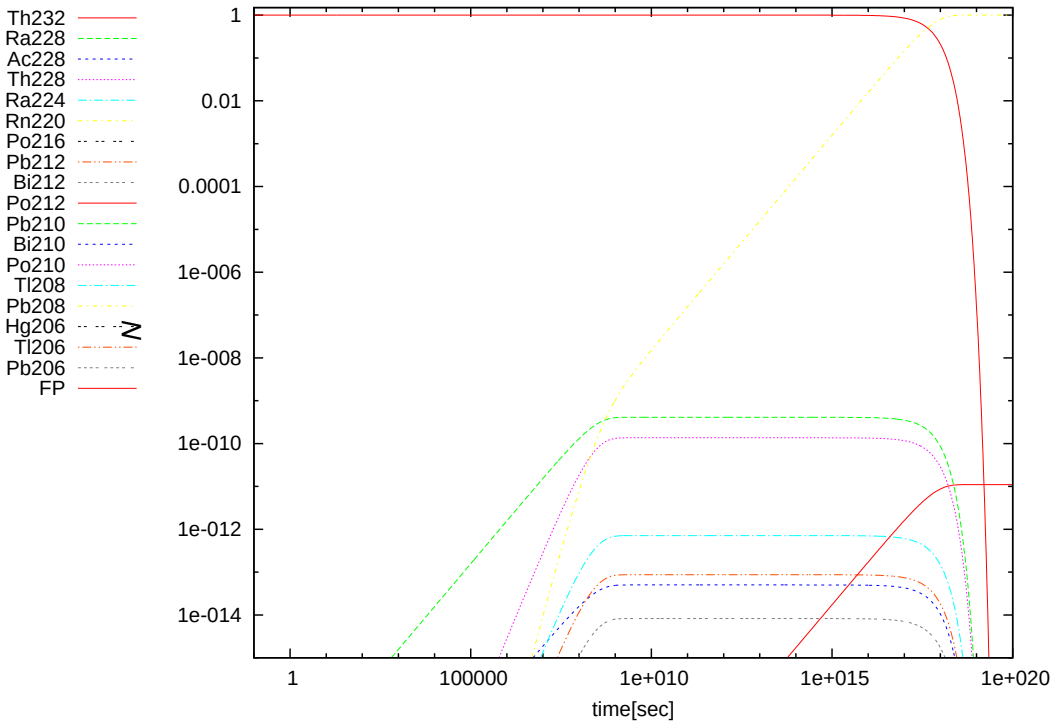

図 6.5: α 崩壊のトリウム系列の時間変化。横軸は経過時間（秒）, 縦軸は核種の個数（時刻ゼロでの ^{232}Th の個数を 1 に規格化した）を表す。

以外の核種の個数はほぼ一定値を保っている。これは 6.4 節で説明したように放射平衡のプラトー領域を示している。

続いて図 6.6 は図 6.5 を放射能の量で両対数スケールでプロットしたものである。横軸は経過時間（秒）, 縦軸は崩壊系列における放射能の量を表している。こちらも 10^9 秒以降では同じ割合で減少していることがわかる。したがって、こちらでも放射平衡が起きていることが読み取れる。

図 6.7 は半減期 7 億年の ^{235}U を起点とするアクチニウム系列の崩壊の時間変化を両対数スケールでプロットしたものである。横軸は経過時間（秒）, 縦軸は崩壊系列上の核種の個数（時刻ゼロにおける起点の核の個数を 1 個に規格化した）を表している。10^{12}~10^{15} 秒の間に, 安定核である終点の ^{207}Pb,^{209}Bi, および核分裂生成物の総量は増加しているが, 他の核種の個数はほぼ一定値を保っている。これもやはり放射平衡のプラトー領域を示している。

続いて図 6.8 は図 6.7 を放射能の量で両対数スケールでプロットしたものである。横軸は経過時間（秒）, 縦軸は崩壊系列における放射能の量を表している。こちらも 10^{12} 秒以降では同じ割合で減少していることがわかる。したがって、こちらでも放射平衡が起きていることが読み取れる。

図 6.9 は半減期 45 億年の ^{238}U を起点とするウラン系列の崩壊の時間変化を両対数スケールでプロットしたものである。横軸は経過時間（秒）, 縦軸は崩壊系列上の核種の個数（時刻ゼロにおける起点の核の個数を 1 個に規格化した）を表している。10^{14}~10^{17} 秒の間に, 安定核である終点の ^{209}Bi,^{208}Pb,^{206}Pb, および核分裂生成物の総量は増加しているが, 他の核種の個数はほぼ一定値を保っている。これもやはり放射平衡のプラトー

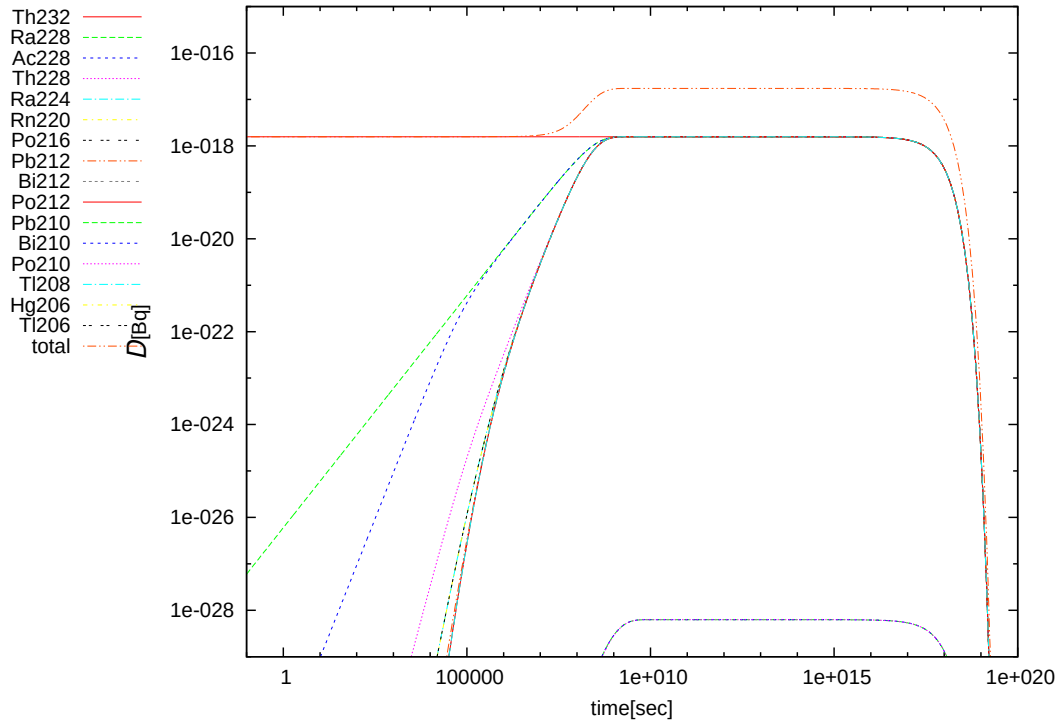

図 6.6: α 崩壊のトリウム系列の時間変化。横軸は経過時間（秒）, 縦軸は崩壊系列における放射能の量を表している。全放射能量は total と表示されている。

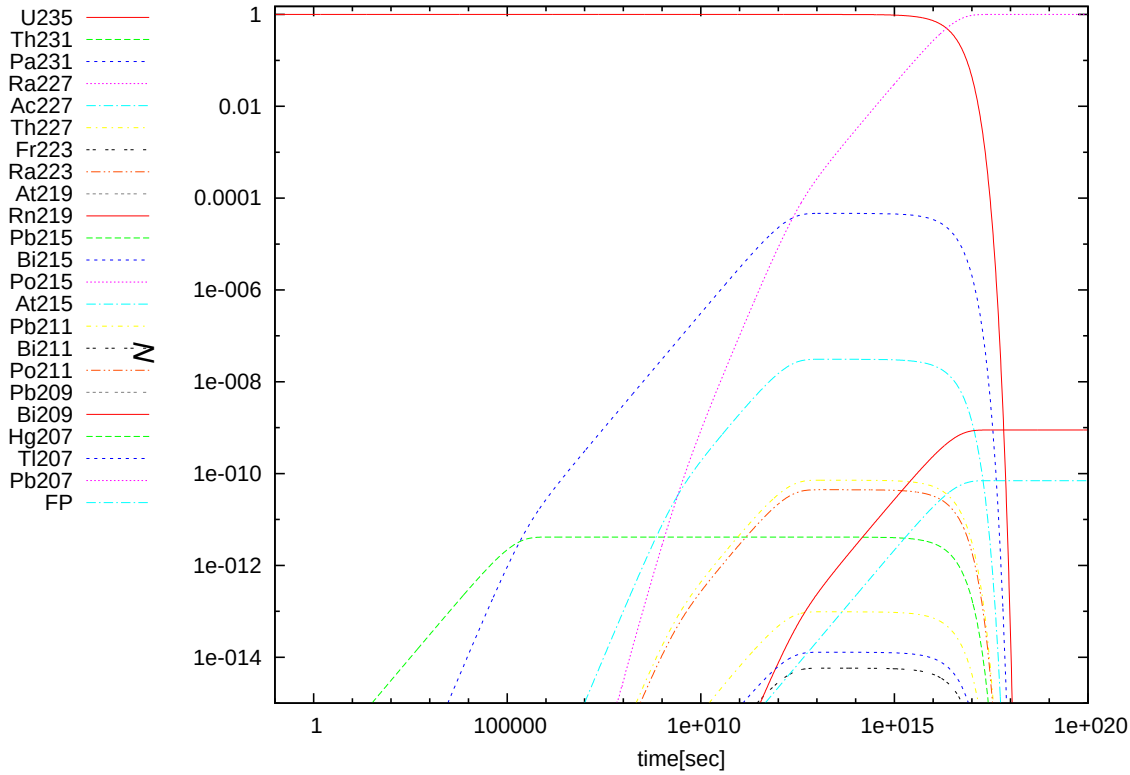

図 6.7: アクチニウム崩壊系列の時間変化。横軸は経過時間（秒）, 縦軸は核種の個数（時刻ゼロでの ^{235}U の個数を 1 に規格化した）を表す。

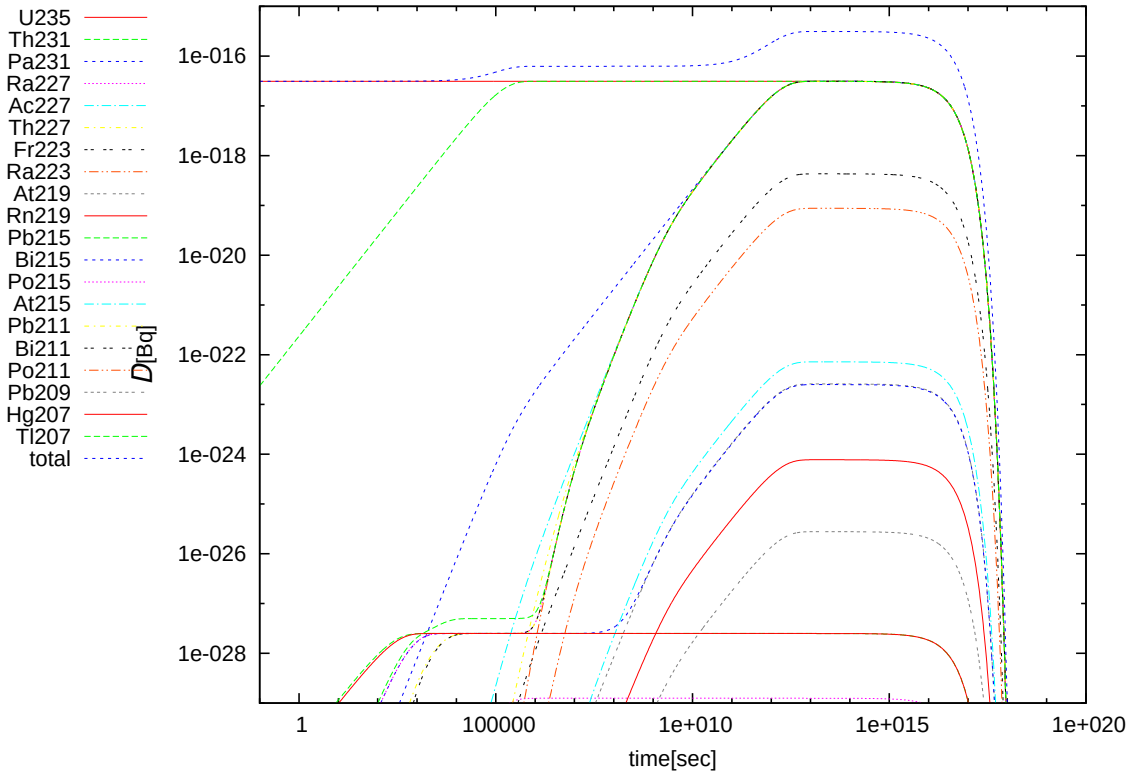

図 6.8: アクチニウム崩壊系列の時間変化。横軸は経過時間（秒）, 縦軸は崩壊系列における放射能の量を表している。全放射能量は total と表示されている。

領域を示している。

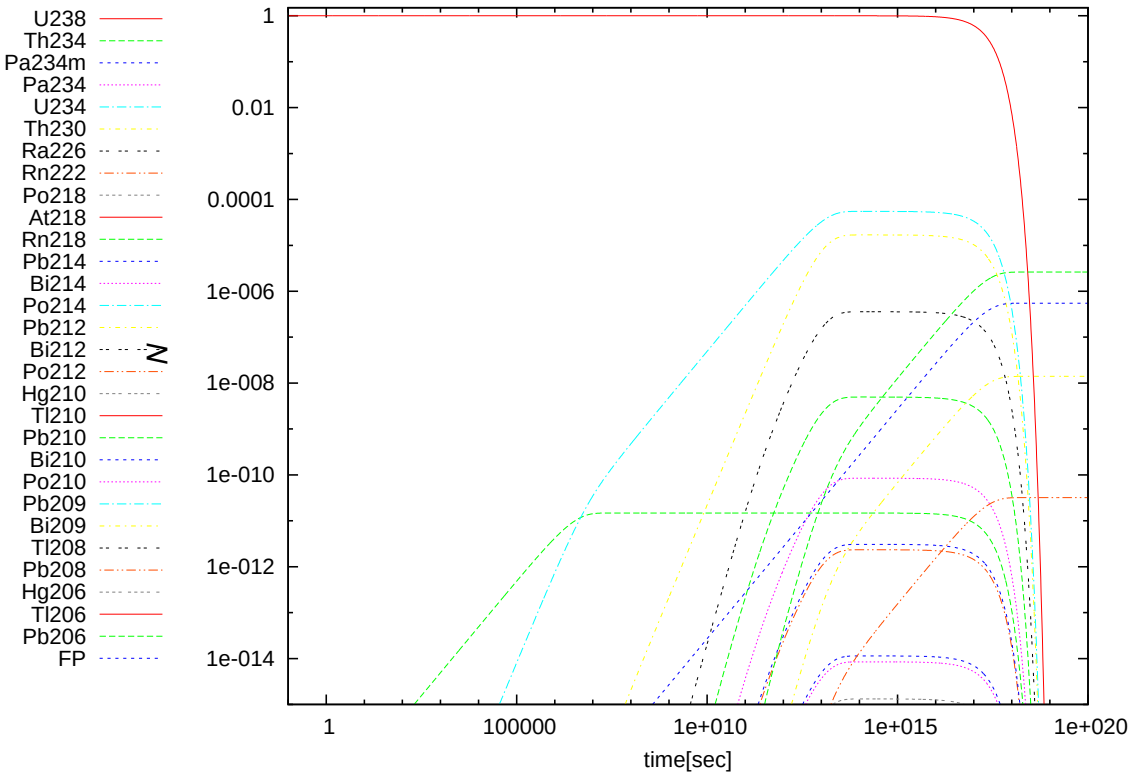

図 6.9: ウラン崩壊系列の時間変化。横軸は経過時間（秒）, 縦軸は核種の個数（時刻ゼロでの ^{238}U の個数を 1 に規格化した）を表す。

続いて図 6.10 は図 6.9 を放射能の量で両対数スケールでプロットしたものである。横軸は経過時間（秒）, 縦軸は崩壊系列における放射能の量を表している。こちらも 10^{14} 秒以降では同じ割合で減少していることがわかる。したがって、こちらでも放射平衡が起きていることが読み取れる。

図 6.11 は半減期 214 万年の ^{237}Np を起点とするネプツニウム系列の崩壊の時間変化を両対数スケールでプロットしたものである。横軸は経過時間（秒）, 縦軸は崩壊系列上の核種の個数（時刻ゼロにおける起点の核の個数を 1 個に規格化した）を表している。この場合にプラトー領域を示すのは, 10^6 〜 10^{13} 秒の間の ^{233}Pa だけである。これはその直後の核種の ^{233}U が 16 万年という起点の核とさほど違わない半減期を持つためである。^{233}U を起点とすれば, その下流はプラトー領域を形成しうるが, 起点が ^{237}Np であるため, 10^{13} 秒付近までは ^{233}U の個数が増え続けるので, 下流の核種の個数もそれに比例して増え続けるのである。しかし、10^{13} 秒付近からは同じ割合で個数が減少しているので、放射平衡は起きている。

続いて図 6.12 は図 6.11 を放射能の量で両対数スケールでプロットしたものである。横軸は経過時間（秒）, 縦軸は崩壊系列における放射能の量を表している。こちらも 10^{13} 秒以降では同じ割合で減少していることがわかる。したがって、こちらでも放射平衡が起きていることが読み取れる。

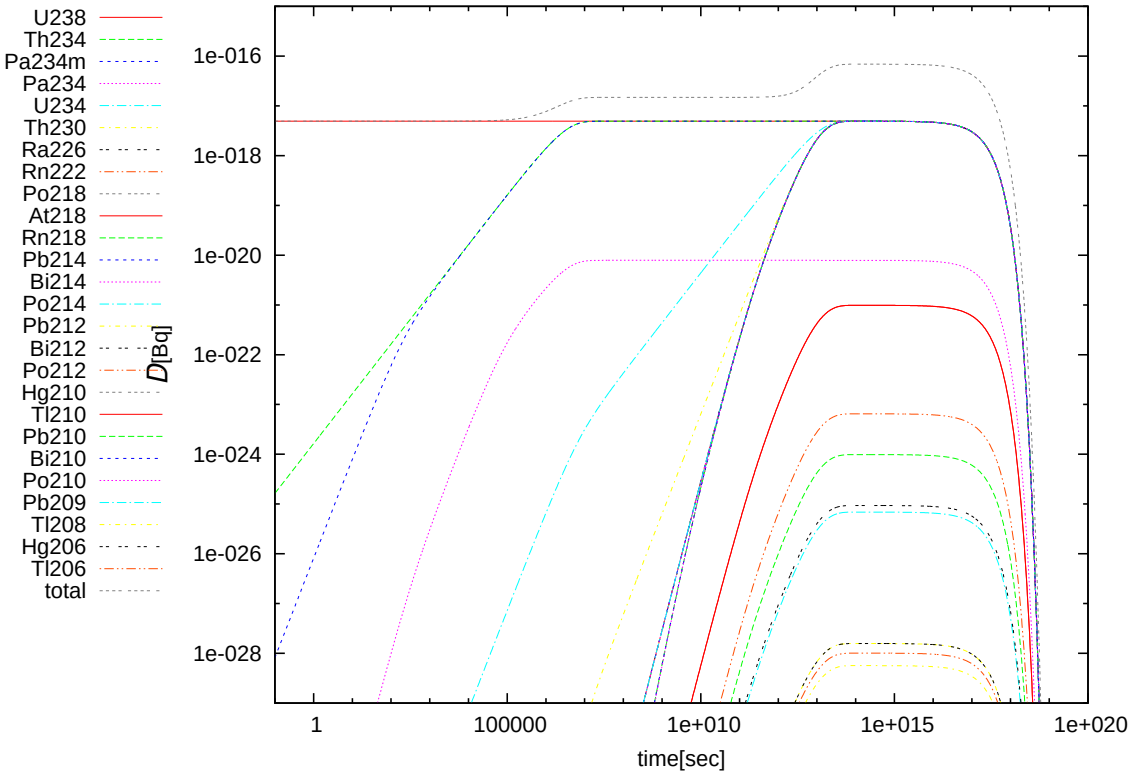

図 6.10: ウラン崩壊系列の時間変化。横軸は経過時間（秒）, 縦軸は崩壊系列における放射能の量を表している。全放射能量は total と表示されている。

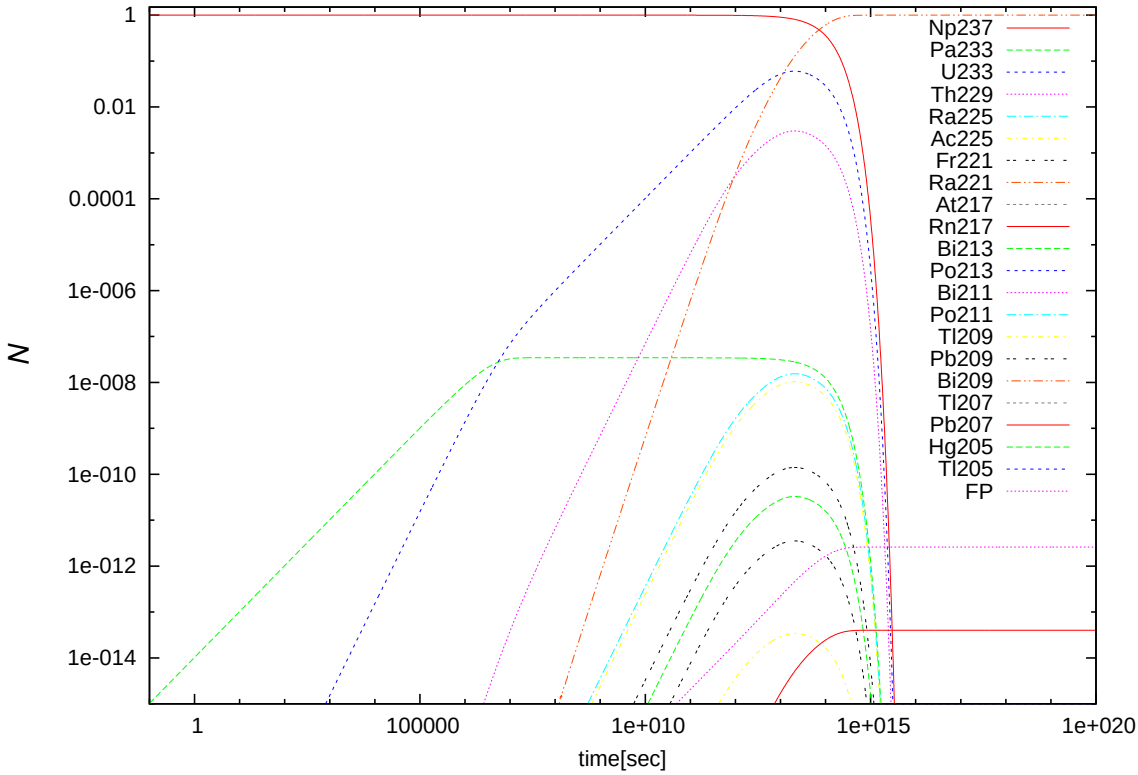

図 6.11: ネプツニウム崩壊系列の時間変化。横軸は経過時間（秒）, 縦軸は核種の個数（時刻ゼロでの ^{237}Np の個数を 1 に規格化した）を表す。

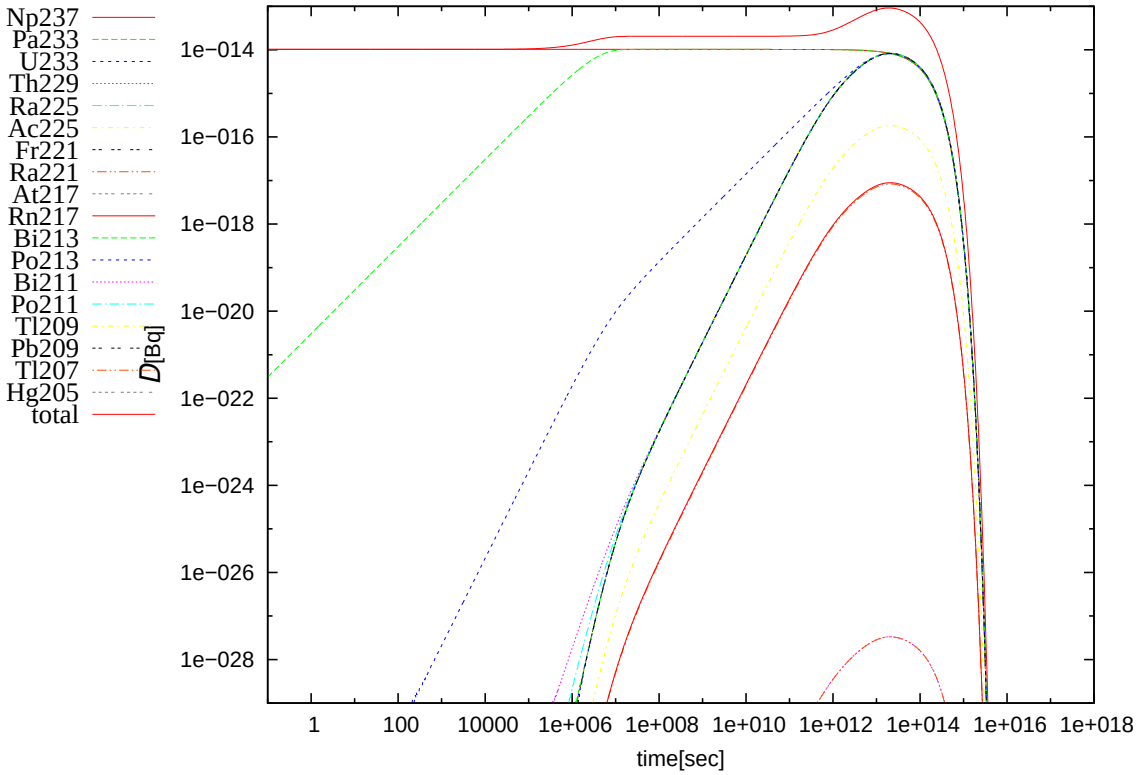

図 6.12: ネプツニウム崩壊系列の時間変化。横軸は経過時間（秒）, 縦軸は崩壊系列における放射能の量を表している。全放射能量は total と表示されている。

表 6.1: ウンウントリウムの崩壊系列。^{238}U 以降はウラン系列と同様であるため省略する。

元素	質量数	陽子数	中性子数	半減期	崩壊様式
Uut	278	113	165	0.24ms	α
Rg	274	111	163	6.4ms	α
Mt	270	109	161	2.0s	α
Bh	266	107	159	1.7s	α
Db	262	105	157	35s	α,67%
					SF,33%
Lr	258	103	155	4.1s	α,95%
					SF,5%
Md	254	101	153	28m	ϵ
Fm	254	100	154	3.240h	α,99.94%
					SF,0.06%
Fm	250	98	152	13.08y	α,99.92%
					SF,0.08%
Fm	246	96	150	4706y	α,99.97%
					SF,0.03%
Pu	242	94	148	3.240h	α
					SF,5.5×10^{-4}%
U	238	92	146	4.468×10^9y	α
					SF, 5.5×10^{-5}%

6.4 短寿命な超重核を起点とする α 崩壊系列

本研究では, 教科書にはのっていないような非常に重い短寿命な核種からの α 崩壊系列についての計算を行ってみた。そのような崩壊系列は, 半減期や分岐のデータが完全には揃っておらず, 正確に崩壊の時間変化を計算できるものは少ない。我々は, データの揃った数少ない系列の中で, 2004 年に日本の理化学研究所で発見されたウンウントリウム (Uut), 2002 年にロシアのドゥブナ合同原子核研究所で発見されたウンウンオクチウム (Uuo) の崩壊について計算する。前節と同じように計算に必要なプログラムは付録に収録しておく。また,4.3.1 節と同じように,2 つの崩壊系列を表 6.1 及び表 6.2 に記す。

図 6.13 には, $^{278}_{113}\mathrm{Uut}_{165}$ を起点とする崩壊系列の時間変化を両対数スケールでプロットしたものである。図からわかるとおり, この場合はほぼ放射平衡にはならないことがわかる。Uut はウラン系列の起点である ^{238}U へとつながっていくが, ^{238}U が桁違いに超寿命であるため, 計算をこの核で打ち切ってある。

続いて図 6.14 は図 6.13 を放射能の量で両対数スケールでプロットしたものである。横軸は経過時間（秒）, 縦軸は崩壊系列における放射能の量を表している。こちらでもほぼ放射平衡は起きていないことが読み取れる。ここで, 減少していく部分で直線の傾きを取ると −1 になっている。これは, 半減期の短い核から長い核へと崩壊が進んでい

表 6.2: ウンウンオクチウムの崩壊系列。

元素	質量数	陽子数	中性子数	半減期	崩壊様式
Uuo	294	118	176	0.89ms	α
Lv	290	116	174	15ms	α
Fl	286	114	172	0.16s	α,40%
					SF,60%
Cn	282	112	170	0.50ms	SF

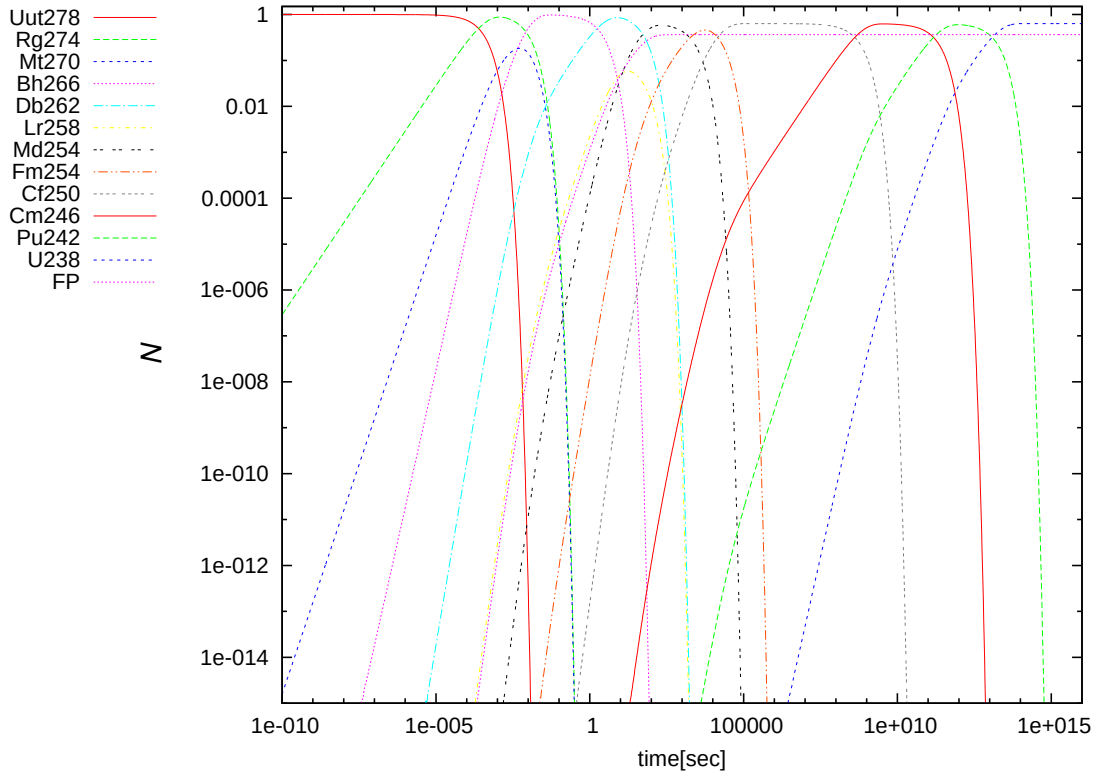

図 6.13: ^{278}Uut を起点とする α 崩壊系列の時間変化。横軸は経過時間（秒）, 縦軸は核種の個数（時刻ゼロでの起点の核の個数を 1 個に規格化した）を表す。

くためで, これは大沼・森近の卒業論文 [1] で詳しく書いてある β 崩壊系列と同じことが起きている。なぜならば,^{278}Uut を起点とするそれぞれの核種の放射能の量を

$$D = N_{\mathrm{i}}\lambda_{\mathrm{i}} \tag{6.8}$$

とおくと, 式 (6.8) の値が一定になり,D は t の -1 乗に比例することになるためである。また, 節で例えたようにダムがある川で例えると, 流入元のダムが空になり, 流入先のダムの水が溜りきった後で次のダムへと水が流出していく。このことの繰り返しであり, 節の例えとは異なる川の流れになる。

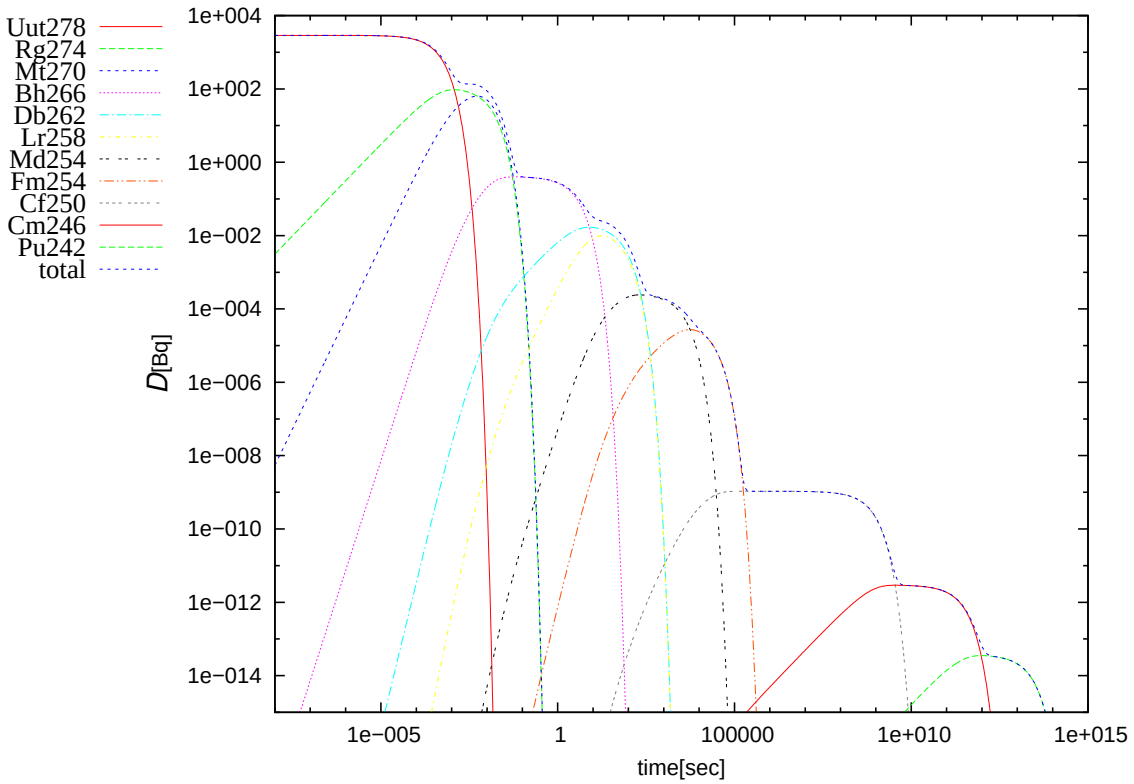

図 6.14: ^{278}Uut を起点とする α 崩壊系列の時間変化。横軸は経過時間（秒）, 縦軸は崩壊系列における放射能の量を表している。全放射能量は total と表示されている。

図 6.15 には, $^{294}_{118}\mathrm{Uuo}_{176}$ を起点とする崩壊系列の時間変化を両対数スケールでプロットしたものである。図からわかるとおり, この場合も放射平衡にはならないことがわかる。この崩壊系列は全て核分裂へとつながって行く。核分裂生成物は, 大沼・森近の卒業論文で計算されたような β 崩壊系列で崩壊を続けることになる。

続いて図 6.16 は図 6.15 を放射能の量で両対数スケールでプロットしたものである。横軸は経過時間（秒）, 縦軸は崩壊系列における放射能の量を表している。こちらでも放射平衡は起きていないことが読み取れる。こちらもデータは少ないが同じように、減少していく部分で直線の傾きを取ると -1 になっており、先程と同じようにベータ崩壊系列と同じことが起きている。理由も同様である。

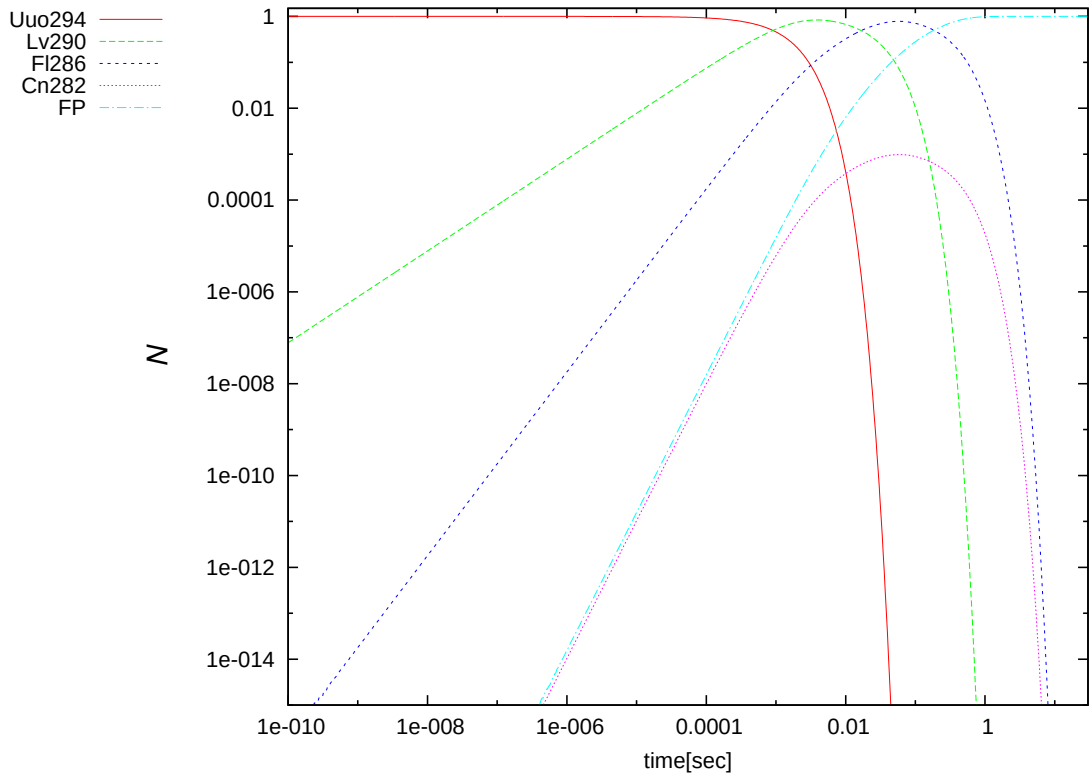

図 6.15: ^{294}Uuo を起点とする α 崩壊系列の時間変化。横軸は経過時間（秒）, 縦軸は核種の個数（時刻ゼロでの起点の核の個数を 1 個に規格化した）を表す。

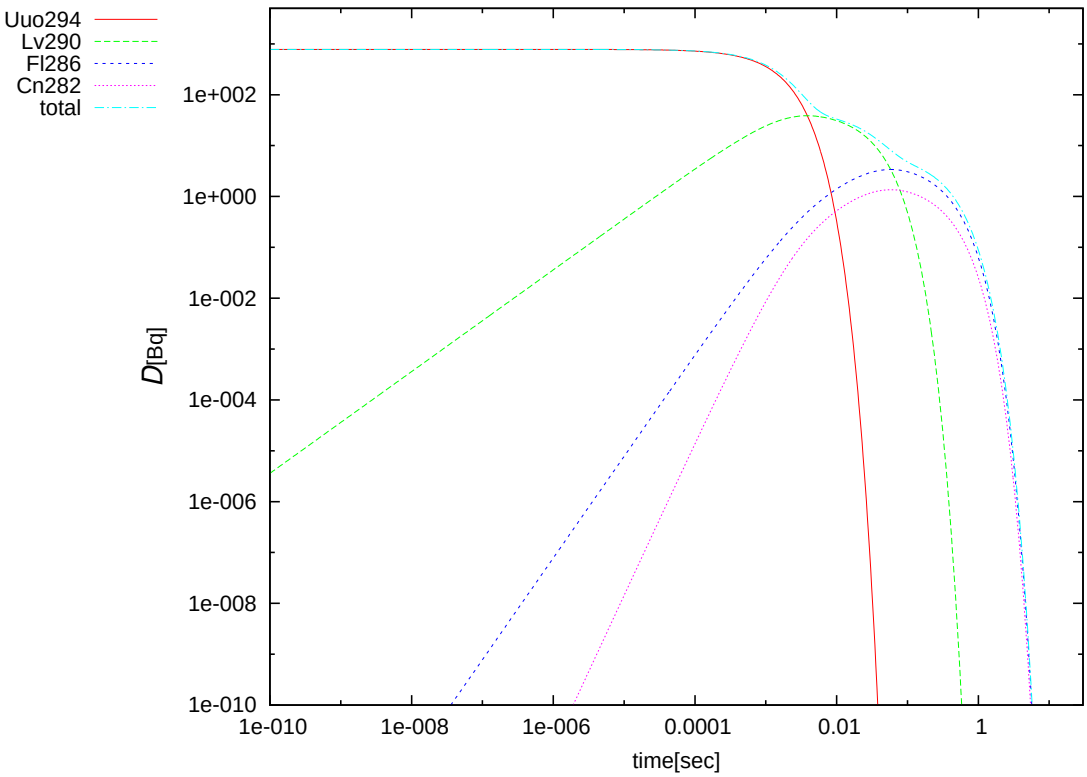

図 6.16: ^{294}Uuo を起点とする α 崩壊系列の時間変化。横軸は経過時間（秒）, 縦軸は崩壊系列における放射能の量を表している。全放射能量は total と表示されている。

第7章　まとめ

本研究では原子核の α 崩壊系列について, その崩壊を決定する時間に関する一階の定数係数線形連立常微分方程式を解き, その時間変化を調べた。崩壊系列中の他の核種と比較して, 半減期が圧倒的に長い核を起点とする崩壊については, 放射平衡と呼ばれる現象が見られることを確認できた。プラトー領域を伴う放射平衡の例として, まずトリウム系列を含む 3 系列を示した。また, プラトー領域を伴わない放射平衡の例として, 同程度の半減期を持つ核種を 2 個含むネプツニウム系列を示した。次に, 半減期のごく短い超重核を起点とする崩壊を 2 例挙げ, 放射平衡を示さないことを見た。

本研究では遺り残した課題として, 崩壊系列の起点を他の核種に取った場合の計算も行ってみることで, 部分的な放射平衡が現れることがあるのかどうかをより詳細に論じるという課題が考えられる。以上, 今後本研究を引き継いで発展させてくれる卒業研究生への課題としてここに記しておくものである。

参考文献

[1] 大沼正季, 森近厚平, 著,「^{235}U の全核分裂生成物の放射能の時間変化」, 福井大学工学部物理工学科卒業論文（2012）.

[2] Particle Physics Booklet, Particle Data Group (2012).

[3] J.K. Tuli, NUCLAR WALLET CARDS, *http://www.nndc.bnl.gov/wallet/* (2011).

[4] B・ポッフ,K. リーツ,C. ショルツ,F. サッチャ, 著 柴田利明, 訳,「素粒子・原子核物理入門　改訂新版」, 丸善出版,（2012）.

[5] 藤家洋一,「原子力 - 自然に学び, 自然を学ぶ - 」,ERC 出版,（2005）.

[6] 米国立核データセンター (National Nuclear Data Center), ウェブサイト　url *http://www.nndc.bnl.gov*

謝辞

本論文を作成するにあたり, 田嶋直樹先生には終始丁寧なご指導をしていただいたことに感謝し, お礼申し上げます。また博士前期課程 1 年生の伊藤研人氏, 杉浦友章氏には研究上の様々な面でお世話をしていただいたことに深く感謝いたします。同じ研究室で卒業研究に取り組んだ杉本智彦氏, 團野良樹氏にも同様に感謝いたします。最後に, 本研究に対してご意見をいただいた, 多くの物理工学科の先生方にもお礼申し上げ, 謝辞の言葉とさせていただきます。

付録

1) 原子質量超過から原子核の束縛エネルギーを計算するプログラム

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

main()
{
        int Z; //計算したい原子の原子番号
        int N; //中性子数
        double Mx; //原子質量超過
        double H = 938.7829; //(MeV):水素原子の質量
        double n = 939.5652; //(MeV):中性子の質量
        double u = 931.4939; //(MeV):原子質量単位
        double a = 1.433e-5; //(MeV):原子中の電子の束縛エネルギーの計算式の係数
        double E; //結合エネルギー

        printf("Z を入力");
        scanf("%d",&Z);

        printf("N を入力");
        scanf("%d",&N);

        printf("M を入力");
        scanf("%f",&M);

        printf("E=%f\n",M - Z *( H - u) - N *( n - u ) + a *( pow(Z,2.39) - Z );

        return 0;

}
```

2) 5.2 節で取り上げた放射性崩壊系列の解析解の値を計算するプログラム

```
/*
原子核の放射性崩壊を表す
時間に関する 1 階の常微分方程式の解の値を表示するプログラム
2014 年 7 月 23 日　created from scratch by 松岡　弘和
```

```
decay-ex1f に改良をくわえたもの
if 分岐を使い、年、日、時間、分、秒にそれぞれ対応。
*/

#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include <ctype.h>

int main()
{
   double t; //時間
   double halflife; //半減期
   double lambda; //崩壊定数λ
   double n1; //n1 原子核種１の個数（初期値＝１と規格化）
   double n2; //n2 原子核種２の個数（初期値＝０と規格化）
   double a; //作業変数 1
   double t2; //作業変数 2
   double halflife2; //作業変数 3(halflife で入力した年、日、時間、分を秒に変換)
   char timeunit[16]; // 読み込んだ半減期の数値の時間の単位

/*
   printf("# λを入力 : ");
   scanf("%lf",&lambda);
   printf("# lambda=%f\n",lambda);
*/

   printf("# T_(1/2)");
   scanf("%lf",&halflife);
   printf("# timeunit");
   scanf("%S",&timeunit);
   printf("# halflife=%f  unit=%s\n",halflife,timeunit);
//   exit(0);

   if(toupper(timeunit[0])=='Y'){
      halflife2=halflife * 365.25 * 86400;
}

   else if(toupper(timeunit[0])=='D'){
      halflife2=halflife * 86400;
}
   else if(toupper(timeunit[0])=='H'){
      halflife2=halflife * 3600;
}
   else if(toupper(timeunit[0])=='M'){
      halflife2=halflife * 60;
}
   else if(toupper(timeunit[0])=='S'){
      halflife2=halflife;
}
```

```
   else{
 printf("eroor\n");
};

  printf("%f\n",halflife2);

     t2=halflife2 / 500;
   for(t=0;t<halflife2 * 3;t=t+t2){
        a=-log(2) / halflife2 * t;
        n1=exp(a);
        n2=1 - n1;

        printf("%e %f %f\n",t,n1,n2);
      }
    return 0;

}
```

2') プログラム 2 への入力データ

```
30.1
Y
```

3) 5.3 節で取り上げた放射性崩壊系列の解析解の値を計算するプログラム

```
/*
原子核の放射性崩壊を表す
時間に関する 1 階の常微分方程式の解の値を表示するプログラム
卒業論文の 5 章 3 節の例のような場合において使用できる。
2014 年 7 月 27 日　created by 松岡　弘和
*/

#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include <ctype.h>



int main()

{
   double t; //時間 (秒)
   double halflife; //半減期 (timeunit で指定される単位)
   double n1; //n1 原子核種 1 の個数（初期値=1 と規格化）
   double n2; //n2 原子核種 2 の個数（初期値=0 と規格化）
```

```
double n3; //n3 原子核種 3 の個数（初期値=0 と規格化）
double a; //作業変数 1
double b; //作業変数 2
double c; //作業変数 3
double p1; //分岐比 1
double p2; //分岐比 2
double halflife2; //半減期 (秒単位)
double t2; //halflife2 を 500 で割った値
char timeunit[16]; // halflife の時間の単位


printf("# T_(1/2)?:");
scanf("%lf",&halflife);
printf("# timeunit?:");
scanf("%s",timeunit);

printf("分岐比 1：");
scanf("%lf",&p1);
printf("分岐比 2：");
scanf("%lf",&p2);

printf("# halflife=%f  unit=%s\n",halflife,timeunit);

if(toupper((int)timeunit[0])==(int)'Y'){
   halflife2=halflife * 365.25 * 86400;
}
else if(toupper((int)timeunit[0])==(int)'D'){
   halflife2=halflife * 86400;
}
else if(toupper((int)timeunit[0])==(int)'H'){
   halflife2=halflife * 3600;
}
else if(toupper((int)timeunit[0])==(int)'M'){
   halflife2=halflife * 60;
}
else if(toupper((int)timeunit[0])==(int)'S'){
   halflife2=halflife;
}
else{
   printf("eroor\n");
};

printf("# halflife=%f (sec)\n",halflife2);
printf("# time(sec), no. of nuclide 1, no. of nuclide 2,no. of nuclide 3\n");
t2=halflife2 / 500;
for(t=0;t<halflife2 * 3;t=t+t2){
     a=-log(2) / halflife2 * t;
     b=a * p1;
     c=a * p2;
     n1=exp(a);
```

```
        n2=b / a *(1-n1);
        n3=c / a *(1-n1);
        // 出力は、時刻（秒）、核 1 の個数（初期値=1）、核 2 の個数（初期値=0）、核 3 の
個数 (初期値=0)
        printf("%e %f %f %f\n",t,n1,n2,n3);
   }
       return 0;
}
```

3') プログラム 3 への入力データ

```
1.277e04
year
0.1072
0.8928
```

4) 5.4 節で取り上げた放射性崩壊系列の解析解の値を計算するプログラム

```
/*
原子核の放射性崩壊を表す
例 3 のような場合で使用できる 2 段階で崩壊した場合のプログラム
2014 年 8 月 27 日　created from scratch by 松岡　弘和
*/

#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include <ctype.h>

double time_unit_converter(double halflife1,char *timeunit);
double time_unit_converter2(double halflife2,char *timeunit2);
int main()
{
   double t; //時間
   double halflife1; //半減期
   double halflife2; //半減期
   double n1; //n1 原子核種１の個数（初期値＝１と規格化）
   double n2; //n2 原子核種２の個数（初期値＝０と規格化）
   double n3; //n2 原子核種 3 の個数（初期値＝０と規格化）
   double a; //作業変数 1
   double b; //作業変数 2
   double t2; //作業変数 3

   double tuc; //半減期を表す関数 1
   double tuc2; //半減期を表す関数 2
   double hl1; //関数内における作業変数 1
```

```
   double hl2; //関数内における作業変数 2
   char tu1[16]; //関数内における作業変数 3
   char tu2[16]; //関数内における作業変数 4

   printf("# T_(1/2)1");
   scanf("%lf",&hl1);
   printf("# timeunit");
   scanf("%s",tu1);
   printf("# T_(1/2)2");
   scanf("%lf",&hl2);
   printf("# timeunit2");
   scanf("%s",tu2);
   printf("# halflife=%f  unit=%s\n",hl1,tu1);
   printf("# halflife=%f  unit=%s\n",hl2,tu2);

   tuc=time_unit_converter(hl1,tu1);
   tuc2=time_unit_converter2(hl2,tu2);


   printf("%f\n",tuc);


   for(t=3600;t<tuc * 3;t=t*1.03){
        a=log(2) / tuc ;
        b=log(2) / tuc2 ;
        n1=exp(-a * t);
        n2=a / (a - b) * (exp(-b * t) - exp(-a * t));
        n3=b / (a - b) * exp(-a * t) - (a / (a - b)) * exp(-b * t) + 1;
        printf("%e %e %e %e\n",t,n1,n2,n3);
      }
   return 0;

}

double time_unit_converter(double halflife1,char *timeunit)
{
        double halflife1a;

   if(toupper((int)timeunit[0])==(int)'Y'){
      halflife1a=halflife1 * 365.25 * 86400;
   }

   else if(toupper((int)timeunit[0])==(int)'D'){
      halflife1a=halflife1 * 86400;
   }
   else if(toupper((int)timeunit[0])==(int)'H'){
      halflife1a=halflife1 * 3600;
   }
   else if(toupper((int)timeunit[0])==(int)'M'){
      halflife1a=halflife1 * 60;
```

```
   }
   else if(toupper((int)timeunit[0])==(int)'S'){
      halflife1a=halflife1;
   }
   else{
   printf("eroor\n");
   };
   return halflife1a;

}

double time_unit_converter2(double halflife2,char *timeunit2)
{
        double halflife2a;

   if(toupper((int)timeunit2[0])==(int)'Y'){
      halflife2a=halflife2 * 365.25 * 86400;
   }

   else if(toupper((int)timeunit2[0])==(int)'D'){
      halflife2a=halflife2 * 86400;
   }
   else if(toupper((int)timeunit2[0])==(int)'H'){
      halflife2a=halflife2 * 3600;
   }
   else if(toupper((int)timeunit2[0])==(int)'M'){
      halflife2a=halflife2 * 60;
   }
   else if(toupper((int)timeunit2[0])==(int)'S'){
      halflife2a=halflife2;
   }
   else{
   printf("eroor\n");
   };
   return halflife2a;

}
```

4') プログラム 4 への入力データ

```
28.79
Y
64.10
H
```

5) 5.5 節で取り上げた放射性崩壊系列の解析解の値を計算するプログラム

```
/*
原子核の放射性崩壊を表す
2 段階で崩壊した場合のプログラム
2014 年 9 月 5 日　created from scratch by 松岡　弘和

if 分岐を使い、年、日、時間、分、秒にそれぞれ対応。
*/

#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include <ctype.h>

double time_unit_converter(double halflife1,char *timeunit);
double time_unit_converter2(double halflife2,char *timeunit2);
int main()
{
   double t; //時間
   double halflife1; //半減期
   double halflife2; //半減期
   double n1; //n1 原子核種 1 の個数（初期値＝ 1 と規格化）
   double n2; //n2 原子核種 2 の個数（初期値＝ 0 と規格化）
   double n3; //n2 原子核種 3 の個数（初期値＝ 0 と規格化）
   double p1; //分岐比 1
   double p2; //分岐比 2
   double a; //作業変数 1
   double b; //作業変数 2
   double c; //作業変数 3
   double d; //作業変数 4
   double t2; //作業変数 5

   double sum; //半減期を表す関数 1
   double sum2; //半減期を表す関数 2
   double hl1; //
   double hl2; //
   char tu1[16]; //
   char tu2[16]; //

   printf("# T_(1/2)1");
   scanf("%lf",&hl1);
   printf("# timeunit");
   scanf("%S",&tu1);
   printf("p1");
   scanf("%lf",&p1);
   printf("# T_(1/2)2");
   scanf("%lf",&hl2);
   printf("# timeunit2");
   scanf("%S",&tu2);
   printf("p2");
   scanf("%lf",&p2);
```

```
   printf("# halflife=%f  unit=%s  p1=%f\n",hl1,tu1,p1);
   printf("# halflife=%f  unit=%s  p2=%f\n",hl2,tu2,p2);

   sum=time_unit_converter(hl1,tu1);
   sum2=time_unit_converter2(hl2,tu2);


  printf("%f\n",sum);

     t2=sum / 500;
   for(t=0;t<sum * 3;t=t+t2){
        a=log(2) / sum ;
        b=log(2) / sum2 ;
        c=a * p1;
        d=b * p2;
        n1=exp(-a * t);
        n2=c / (a - b) * (exp(-b * t) - exp(-a * t));
        n3=1 - (1 - c / (a - b) ) * exp(-a * t) - c / (a - b) * exp(-b * t);
        printf("%e %f %f %f\n",t,n1,n2,n3);
      }
    return 0;

}

double time_unit_converter(double halflife1,char *timeunit)
{
        double halflife1a;

   if(toupper(timeunit[0])=='Y'){
      halflife1a=halflife1 * 365.25 * 86400;
}

   else if(toupper(timeunit[0])=='D'){
      halflife1a=halflife1 * 86400;
}
   else if(toupper(timeunit[0])=='H'){
      halflife1a=halflife1 * 3600;
}
   else if(toupper(timeunit[0])=='M'){
      halflife1a=halflife1 * 60;
}
   else if(toupper(timeunit[0])=='S'){
      halflife1a=halflife1;
}
   else{
 printf("eroor\n");
};
  return halflife1a;

}
```

```
double time_unit_converter2(double halflife2,char *timeunit2)
{
        double halflife2a;

   if(toupper(timeunit2[0])=='Y'){
      halflife2a=halflife2 * 365.25 * 86400;
}

   else if(toupper(timeunit2[0])=='D'){
      halflife2a=halflife2 * 86400;
}
   else if(toupper(timeunit2[0])=='H'){
      halflife2a=halflife2 * 3600;
}
   else if(toupper(timeunit2[0])=='M'){
      halflife2a=halflife2 * 60;
}
   else if(toupper(timeunit2[0])=='S'){
      halflife2a=halflife2;
}
   else{
 printf("eroor\n");
};
  return halflife2a;

}
```

5') プログラム 5 への入力データ

```
/*
原子核の放射性崩壊を表す
2 段階で崩壊した場合のプログラム
2014 年 9 月 5 日　created from scratch by 松岡　弘和

if 分岐を使い、年、日、時間、分、秒にそれぞれ対応。
*/

#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include <ctype.h>

double time_unit_converter(double halflife1,char *timeunit);
double time_unit_converter2(double halflife2,char *timeunit2);
int main()
{
   double t; //時間
```

```
 double halflife1; //半減期
 double halflife2; //半減期
 double n1; //n1 原子核種１の個数（初期値＝１と規格化）
 double n2; //n2 原子核種２の個数（初期値＝０と規格化）
 double n3; //n2 原子核種３の個数（初期値＝０と規格化）
 double p1; //分岐比１
 double p2; //分岐比２
 double a; //作業変数１
 double b; //作業変数２
 double c; //作業変数３
 double d; //作業変数４
 double t2; //作業変数５

 double sum; //半減期を表す関数１
 double sum2; //半減期を表す関数２
 double hl1; //
 double hl2; //
 char tu1[16]; //
 char tu2[16]; //

 printf("# T_(1/2)1");
 scanf("%lf",&hl1);
 printf("# timeunit");
 scanf("%S",&tu1);
 printf("p1");
 scanf("%lf",&p1);
 printf("# T_(1/2)2");
 scanf("%lf",&hl2);
 printf("# timeunit2");
 scanf("%S",&tu2);
 printf("p2");
 scanf("%lf",&p2);
 printf("# halflife=%f  unit=%s  p1=%f\n",hl1,tu1,p1);
 printf("# halflife=%f  unit=%s  p2=%f\n",hl2,tu2,p2);

 sum=time_unit_converter(hl1,tu1);
 sum2=time_unit_converter2(hl2,tu2);


printf("%f\n",sum);

   t2=sum / 500;
 for(t=0;t<sum * 3;t=t+t2){
      a=log(2) / sum ;
      b=log(2) / sum2 ;
      c=a * p1;
      d=b * p2;
      n1=exp(-a * t);
      n2=c / (a - b) * (exp(-b * t) - exp(-a * t));
      n3=1 - (1 - c / (a - b) ) * exp(-a * t) - c / (a - b) * exp(-b * t);
```

```
        printf("%e %f %f %f\n",t,n1,n2,n3);
      }
    return 0;

}

double time_unit_converter(double halflife1,char *timeunit)
{
        double halflife1a;

   if(toupper(timeunit[0])=='Y'){
      halflife1a=halflife1 * 365.25 * 86400;
}

   else if(toupper(timeunit[0])=='D'){
      halflife1a=halflife1 * 86400;
}
   else if(toupper(timeunit[0])=='H'){
      halflife1a=halflife1 * 3600;
}
   else if(toupper(timeunit[0])=='M'){
      halflife1a=halflife1 * 60;
}
   else if(toupper(timeunit[0])=='S'){
      halflife1a=halflife1;
}
   else{
 printf("eroor\n");
};
  return halflife1a;

}

double time_unit_converter2(double halflife2,char *timeunit2)
{
        double halflife2a;

   if(toupper(timeunit2[0])=='Y'){
      halflife2a=halflife2 * 365.25 * 86400;
}

   else if(toupper(timeunit2[0])=='D'){
      halflife2a=halflife2 * 86400;
}
   else if(toupper(timeunit2[0])=='H'){
      halflife2a=halflife2 * 3600;
}
   else if(toupper(timeunit2[0])=='M'){
      halflife2a=halflife2 * 60;
}
```

```
   else if(toupper(timeunit2[0])=='S'){
      halflife2a=halflife2;
}
   else{
 printf("eroor\n");
};
  return halflife2a;

}
```

6) 一般の放射性崩壊系列を計算するためのファイル群

6-1) Makefile

```
.PHONY: all clean reset
all: decay_chain_ac.exe
decay_chain_ac.exe: decay_chain_ac.o set_lambda.o
gcc -o decay_chain_ac.exe decay_chain_ac.o set_lambda.o -lm
decay_chain_ac.o: decay_chain_ac.c
gcc -g -Wall -c decay_chain_ac.c
set_lambda.o: set_lambda.c
gcc -g -Wall -c set_lambda.c
clean:
rm -f decay_chain_ac.o set_lambda.o
reset:clean
rm -f decay_chain_ac.exe
```

6-2) Header file

```
/*
decay_chain_ac.h : to be included by decay_chain_ac.c and set_lambda.c
2014/9/5-10 created
*/

#define M 300 // maximum number of nuclides considered in the calculation
#define BIGNUM 1.0e+300 // a number used as infinity
#define BIGNUM0 1.0e+299 // a number used as maximum finite number

enum{sec,min,hour,day,year};

#ifdef MAIN_SOURCE_FILE
double lambda[M][M]; // decay-rate matrix
double halflife[M]; // array of halflives
char *nuclide[M]; // array of pointers to character strings for nuclides
int nn; // the number of nuclides considered in the decay calculation
```

```
#else
extern double lambda[M][M];
extern char *nuclide[M];
extern double halflife[M];
extern int nn;
#endif

double time_in_sec(double time,int timeunit);
double halflife_to_decayrate(double halflife);
```

6-3) The main source file of the C program

```
/*
decay_chain_ac.c : decay chain of Actinides
2014/9/2-10 Version 1 created by utilizing decay_chain_fp.c (fp=fission products)
*/

#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include <string.h> // for strncmp()

#define MAIN_SOURCE_FILE
#include "decay_chain_ac.h"

double n[M]; // population of each nuclei
double n0[M]; // initial population of nuclei
double c[M][M]; // coefficients used to express the solutions

int alpha_chain(char *init_nucl, double tmin, double tmax, int ntime, FILE *fp_evo);
int set_initial_state(char *init_nucl);
int solve_c();
double n_of_t(int i,double time);
int time_evolution(double tmin, double tmax, int ntime, FILE *fp_evo);
int make_script(FILE *fp);
int make_gnuplot_script(FILE *fp);
int check_sum_n(double tmin,double tmax,int ntime);
int check_derivative(double tmin,double tmax,int ntime);

int set_lambda();

int main(){
  double tmin=1.0e-3; // 1ms ; -1.0 for automatic determination;
  double tmax=1.0e+16; // 100years ; -1.0 for automatic determination;
  int ntime=2000; // -1 for default value = 100;
  FILE *fp_evo=NULL; // file to write time evolution
  FILE *fp_scr=NULL; // file to write a shell script to divide time evolution data
```

```
  fp_evo=fopen("time_evolution.dat","wt");
  if(fp_evo == NULL){
    fprintf(stderr,"failed to open time evolution output file\n");
    exit(1);
  }
  //alpha_chain("Uuo294",tmin=1e-005,tmax=1e+40,ntime,fp_evo);
  //alpha_chain("Uut278",tmin=1e-06,tmax=1e+16,ntime,fp_evo);
  //alpha_chain("U238",tmin=1e-005,tmax=1e+40,ntime,fp_evo);
  //alpha_chain("Np237",tmin=1e-005,tmax=1e+40,ntime,fp_evo);
  //alpha_chain("U235",tmin=1e-005,tmax=1e+30,ntime,fp_evo);
  //alpha_chain("Th232",tmin=1e-005,tmax=1e+40,ntime,fp_evo);
  if(fclose(fp_evo)!=0){
    fprintf(stderr,"failed to close evolution output file\n");
  }

  fp_scr=fopen("time_evolution.sh","wt");
  if(fp_scr == NULL){
    fprintf(stderr,"failed to open shell-script to grep time_evolution.dat\n");
    exit(1);
  }
  make_script(fp_scr);
  if(fclose(fp_scr)!=0){
    fprintf(stderr,"failed to close shell-script to grep time_evolution.dat\n");
  }

  fp_scr=fopen("time_evolution.plt","wt");
  if(fp_scr == NULL){
    fprintf(stderr,"failed to open gnuplot script to plot time_evolution data\n");
    exit(1);
  }
  make_gnuplot_script(fp_scr);
  if(fclose(fp_scr)!=0){
    fprintf(stderr,"failed to close gnuplot script to plot time_evolution data\n");
  }
  return 0;
}

int alpha_chain(char *init_nucl, double tmin, double tmax, int ntime, FILE *fp_evo){
  int i,j,k,l,ng,degeneracy;

  set_lambda();
  set_initial_state(init_nucl);

  printf("initial population = 1 for %s\n",init_nucl);

  for(j=0;j<nn;j++){
    printf(" For nuclide %s, n(t=0)=%6.3f\n",nuclide[j],n0[j]);
    for(i=0;i<j;i++){ // check whether there is the same nuclei
      if(strncmp(nuclide[i],nuclide[j],16)==0){
        printf("Error: Entries no. %d and no. %d are the same\n",i,j);
```

```
    }
  }
}

printf("\n lambda[0..%d][0..%d]=\n",nn-1,nn-1);
ng=0; // increased from zero if matrix lambda is not lower triangular
degeneracy=0; // changed to >0 if diagonal elements are equal
for(k=0;k<nn;k++){
  for(l=0;l<nn;l++){
    //printf(" %15.7e",lambda[k][l]);
    printf(" %s",lambda[k][l] == 0.0 ? "." : "*" );
    if(l>k && lambda[k][l]!=0.0) {
    ng++;
    fprintf(stderr,"\n error non zero lambda for %d %s %d %s\n",k,nuclide[k],l,nuclide[
    if(l!=k &&
      fabs(lambda[k][k]-lambda[l][l])/
      (fabs(lambda[k][k])+fabs(lambda[l][l])+1.0e-100)
      <1.0e-13) degeneracy++;
  }
  printf("\n");
}
if(ng>0){
  printf("lambda is not lower triangular %d\n",ng);
  exit(1);
}
if(degeneracy>0){
  printf("eigen values of lambda are degenerated %d/2\n",degeneracy);
  exit(1);
}

printf("\n n0[0..%d]=\n",nn-1);
for(k=0;k<nn;k++){
  printf(" %12.5e",n0[k]);
}
printf("\n");

solve_c();

printf("\n c[0..%d][0..%d]=\n",nn-1,nn-1);
for(k=0;k<nn;k++){
  for(l=0;l<nn;l++){
    printf(" %15.7e",c[k][l]);
  }
  printf("\n");
}

if(tmin < 0.0){
  tmin=BIGNUM;
  for(i=0;i<nn;i++){
    if(halflife[i]>BIGNUM0) continue;
```

```
      if(tmin>halflife[i]) tmin=halflife[i];
    }
    tmin*=0.01;
  }
  if(tmax < 0.0){
    tmax=0.0;
    for(i=0;i<nn;i++){
      if(halflife[i]<BIGNUM0) tmax+=halflife[i];
    }
    tmax*=30.0;
  }
  if(tmin>tmax || tmin <= 0.0 || tmax > BIGNUM0){
    fprintf(stderr,"error: tmin=%e tmax=%e\n",tmin,tmax);
    exit(0);
  }
  if(ntime<0) ntime=100;

  time_evolution(tmin,tmax,ntime,fp_evo);
  check_sum_n(tmin,tmax,ntime);
  check_derivative(tmin,tmax,ntime);
  return 0;
}

int set_initial_state(char *init_nucl){
 int i,c=0;

 for(i=0;i<nn;i++){
   if(strncmp(init_nucl,nuclide[i],16)==0){
     n0[i]=1.0;
     c++;
   }
   else{
     n0[i]=0.0;
   }
 }
 if(c==1){
   return 0;
 }
 else{
   fprintf(stderr,"set_initial_state: error: %d %s\n",c,init_nucl);
   for(i=0;i<nn;i++) fprintf(stderr,"%d %s\n",i,nuclide[i]);
   exit(1);
 }
}

int solve_c(){
  int i,j,k;
  double cjk; //c_[j][k]
  double laikckj; //lambda_[i][k]*c_[k][j]
```

```
  for(j=0;j<nn;j++){
    if(j==0){
      c[j][j]=n0[0];
    }
    else{
      cjk=0;
      for(k=0;k<=j-1;k++){
        cjk+=c[j][k];
      }
      c[j][j]=n0[j]-cjk;
    }
    if(j<nn-1){
      for(i=j+1;i<nn;i++){
        laikckj=0;
        for(k=j;k<=i-1;k++){
          laikckj+=(lambda[i][k]*c[k][j]);
        }
        c[i][j]=(laikckj)/(-lambda[i][i]+lambda[j][j]);
      }
    }
  }
  return 0;
}

double n_of_t(int i,double time){
  int j;
  double s=0;

  for(j=0;j<=i;j++){
    s+=c[i][j]*exp(lambda[j][j]*time);
  }
  return s;
}

int time_evolution(double tmin, double tmax, int ntime, FILE *fp_evo){
  int i,it;
  double r,rr,t,bq;

  r=pow(tmax/tmin,1.0/(ntime-1));
  printf("\n# time n[0..%d]\n",nn-1);
  rr=1;
  for(it=0; it<=ntime; it++){
    if(it==0){
      t=0;
    }
    else{
      t=tmin*rr;
      rr*=r;
    }
    for(i=0;i<nn;i++){
```

```
      n[i]=n_of_t(i,t);
      bq=n[i]*fabs(lambda[i][i]); // becquerel= decays per second
      if(bq<1.0e-99) bq=0.0;
      if(fabs(n[i])<1.0e-99) n[i]=0.0;
      if(fp_evo != NULL){
        fprintf(fp_evo,"%11.5e %12.5e %12.5e %d %s \n",t,n[i],bq,it,nuclide[i]);
      }
    }
  }
  return 0;
}

int make_script(FILE *fp){
  int i;

  if(fp==NULL) return 0;
  for(i=0;i<nn;i++){
    fprintf(fp,"grep %s time_evolution.dat > %s.dat\n",nuclide[i],nuclide[i]);
  }
  return 0;
}

int make_gnuplot_script(FILE *fp){
  int i;

  if(fp==NULL) return 0;
  fprintf(fp,"set logscale xy\n i=1\n j=2\n plot [1e-2:1e+32][1e-16:1.5]\\\n");
  for(i=0;i<nn-1;i++){
    fprintf(fp,"\"%s.dat\" u i:j t \"%s\" w l,\\\n",nuclide[i],nuclide[i]);
  }
  fprintf(fp,"\"%s.dat\" u i:j t \"%s\" w l\n",nuclide[nn-1],nuclide[nn-1]);
  return 0;
}

int check_sum_n(double tmin,double tmax,int ntime){
  double s=0.0,s0=0.0,ds=0.0,maxerror=0.0,time=0.0;
  double r,rr;
  int i,itime=0;

  for(i=0;i<=nn-1;i++){
    s0+=n0[i];
  }
  r=pow(tmax/tmin,1.0/(ntime-1));
  rr=1;
  for(itime=1;itime<=ntime;itime++){
    time=tmin*rr;
    rr*=r;
    s=0.0;
    for(i=0;i<nn;i++){
      n[i]=n_of_t(i,time);
```

```
      if(n[i]>=0){
        s+=n[i];
      }
      else if(n[i]<-1.0e-16){ // -1.0e-20 makes false err message
        printf("check_sum_n:error:negative n[%d]=%15.7e at t=%d\n"
        ,i,n[i],itime);
      }
    }
    ds=fabs(s-s0);
    if(ds>1.0e-9){
      printf("check_sum_n:warning:large error %.5e at time %.5e\n",ds,time);
    }
    if(maxerror<ds) maxerror=ds;
  }
  printf("\nSigma n's max error =%.7e, max relative error=%.7e\n"
  ,maxerror,maxerror/(s0+1.0e-99));
  return 0;
}

int check_derivative(double tmin,double tmax,int ntime){
  double sumn,dn,ds,dt,time,dtopt,r,rr;
  double relative_error, max_relative_error=0.0, maxerror=0.0;
  double ds1,ds2,ds3,ds4,ds5,dsmin;
  int i,j,k,itime;
  const int printlevel = 0;

  r=pow(tmax/tmin,1.0/(ntime-1));
  rr=1;
  for(itime=0;itime<=ntime;itime++){ //ntime_partition
    if(itime==0){
      time=0;
    }
    else{
      time=tmin*rr;
      rr*=r;
    }
    if(printlevel>0) printf("%11.5e",time);
    for(i=0;i<nn;i++){
      n[i]=n_of_t(i,time);
      sumn=0.0;
      for(j=0;j<nn;j++){
        sumn+=lambda[i][j]*n[j];
      }
      dsmin=ds1=ds2=ds3=ds4=ds5=BIGNUM;
      dt=1.0e-12;
      for(k=1;k<=200;k++){ // search dt which minimizes the error
        dn=(- n_of_t(i,time+dt*2)+8*n_of_t(i,time+dt)
        -8*n_of_t(i,time-dt) + n_of_t(i,time-dt*2))/(12*dt);
        ds5=ds4; ds4=ds3; ds3=ds2; ds2=ds1; ds1=fabs(dn-sumn);
        ds=(ds1+ds2+ds3+ds4+ds5)/5.0;
```

```
        if(dsmin > ds) {dsmin=ds; dtopt=dt;}
        dt=dt*1.77827941; // 10^(1/4)=1.778
        if(dt>tmax) break;
      }
      ds=dsmin;
      if(maxerror<ds) maxerror=ds;
      relative_error=ds/(fabs(sumn)+1.0e-50);
      if(printlevel==1) printf(" %12.5e",ds);
      if(printlevel==2) printf(" %12.5e",relative_error);
      if(printlevel==3) printf(" %12.5e %12.5e",ds,dtopt);
      if(printlevel>3) printf(" %12.5e %12.5e",relative_error,dtopt);
      if(max_relative_error<relative_error) max_relative_error=relative_error;
    }
    if(printlevel>0)printf("\n");
  }
  printf("\ndn/dt's max error=%.7e, max relative error=%.7e\n"
  ,maxerror,max_relative_error);
  return 0;
}

//-----------------------------------------------------------------------
double time_in_sec(double time,int timeunit){
  switch(timeunit){
    case sec :return time;break;
    case min :return time*60.0;break;
    case hour :return time*3600.0;break;
    case day :return time*86400.0;break;
    case year :return time*86400.0*365.0;break;
    default :
      fprintf(stderr,"time_in_sec:unknown time unit\n");
      exit(1);
  }
}

double halflife_to_decayrate(double halflife){ // T_{1/2} -> lambda converter
  const double ln2=log(2.0);
  static int zero_counter=0;

  if(halflife>BIGNUM0){
    return (++zero_counter +10)*1.0e-31; // to avoid degeneracy of zero
  }
  return (ln2/(halflife));
}

//-----------------------------------------------------------------------
```

6-4) Data routine for the example chain

```
/*
```

```
set_lambda.c : set decay rate values for decay chains of ex
2015/1/30 created
*/

#include <stdio.h>
#include <stdlib.h>  // for exit()

#include "decay_chain_ac.h"

#define DIAGELEM(NC,HL,TU) {nuclide[NC]=#NC;\
                            halflife[NC]=time_in_sec(HL,TU);\
                            lambda[NC][NC]=-halflife_to_decayrate(halflife[NC]);}
// NC:nuclide, HL:halflife, TU:time unit

#define OFFDIAG(P,D,BR); lambda[D][P]=-lambda[P][P]*(BR)
// P:parent nuclide, D:daughter nuclide, BR:branching ratio

int set_lambda(){
  enum{A,
       B,
       C,
       D,
       E,
       END};
  int i,j;
  nn=END; // the number of nuclides, nn is a global variable
  if(M<nn) exit(1);
  for(i=0;i<nn;i++){for(j=0;j<nn;j++)lambda[i][j]=0;}

// diagonal elements of decay-rate matrix
  DIAGELEM(A, 10e7   ,sec );
  DIAGELEM(B, 10e2   ,sec );
  DIAGELEM(C, 1.0   ,sec );
  DIAGELEM(D, 10e4   ,sec );
  DIAGELEM(E, BIGNUM   ,sec );
// off-diagonal elements of decay-rate matrix
  OFFDIAG(A, B, 1.0  );
  OFFDIAG(B, C, 1.0  );
  OFFDIAG(C, D, 1.0  );
  OFFDIAG(D, E, 1.0  );
  return 0;
}
```

6-5) Data routine for the Thorium chain

```
/*
set_lambda.c : set decay rate values for decay chains of Actinides
2014/9/8-10 created
```

```
*/

#include <stdio.h>
#include <stdlib.h>  // for exit()

#include "decay_chain_ac.h"

#define DIAGELEM(NC,HL,TU) {nuclide[NC]=#NC;\
                            halflife[NC]=time_in_sec(HL,TU);\
                            lambda[NC][NC]=-halflife_to_decayrate(halflife[NC]);}
// NC:nuclide, HL:halflife, TU:time unit

#define OFFDIAG(P,D,BR) lambda[D][P]=-lambda[P][P]*(BR)
// P:parent nuclide, D:daughter nuclide, BR:branching ratio

int set_lambda(){
  enum{Th232,
       Ra228,
       Ac228,
       Th228,
       Ra224,
       Rn220,
       Po216,
       Pb212,
       Bi212,
       Po212,
       Pb210,
       Bi210,
       Po210,
       Tl208,
       Pb208,
       Hg206,
       Tl206,
       Pb206,
       FP,
       END};
  int i,j;
  double r; // branching ratio
  nn=END; // the number of nuclides, nn is a global variable
  if(M<nn) exit(1);
  for(i=0;i<nn;i++){for(j=0;j<nn;j++)lambda[i][j]=0;}

// diagonal elements of decay-rate matrix
//   nuclides in A=4n sequence
  DIAGELEM(Th232, 1.40e10     ,year );
  DIAGELEM(Ra228, 5.75      ,year );
  DIAGELEM(Ac228, 6.15      ,hour );
  DIAGELEM(Th228, 1.9116      ,year );
  DIAGELEM(Ra224, 3.6319      ,day );
  DIAGELEM(Rn220, 55.6      ,sec );
```

```
  DIAGELEM(Po216, 0.145     ,sec );
  DIAGELEM(Pb212, 10.64     ,hour);
  DIAGELEM(Bi212, 60.55     ,min );
  DIAGELEM(Po212, 0.299e-6  ,sec );
  DIAGELEM(Pb210, 22.20  ,year );
  DIAGELEM(Bi210, 5.012  ,day );
  DIAGELEM(Po210, 138.376  ,day );
  DIAGELEM(Tl208, 3.053     ,min );
  DIAGELEM(Pb208, BIGNUM    ,sec );
  DIAGELEM(Hg206, 8.32     ,min );
  DIAGELEM(Tl206, 4.202     ,min );
  DIAGELEM(Pb206, BIGNUM     ,min );
  DIAGELEM(FP,    BIGNUM, min  );
//   nuclides in A=4n+1 sequence

//   nuclides in A=4n+2 sequence

//   nuclides in A=4n+3 sequence

// off-diagonal elements of decay-rate matrix
//   decays from nuclides in A=4n sequence
  r=1.1e-11;
  OFFDIAG(Th232, Ra228, 1-r  );
  OFFDIAG(Th232, FP,    r  );
  OFFDIAG(Ra228, Ac228, 1.0  );
  OFFDIAG(Ac228, Th228, 1.0  );
  r=1e-13;
  OFFDIAG(Th228, Ra224, 1-r  );
  OFFDIAG(Th228, Pb208, r  );
  r=4.0e-11;
  OFFDIAG(Ra224, Rn220, 1-r  );
  OFFDIAG(Ra224, Pb210, r  );
  OFFDIAG(Rn220, Po216, 1.0  );
  OFFDIAG(Po216, Pb212, 1.0  );
  OFFDIAG(Pb212, Bi212, 1.0  );
  r=0.3594e-2;
  OFFDIAG(Bi212, Po212, 1-r  );
  OFFDIAG(Bi212, Tl208, r    );
  OFFDIAG(Po212, Tl208, 1.0  );
  r=1.9e-8;
  OFFDIAG(Pb210, Bi210, 1-r  );
  OFFDIAG(Pb210, Hg206, r  );
  r=1.3e-6;
  OFFDIAG(Bi210, Po210, 1-r  );
  OFFDIAG(Bi210, Tl206, r  );
  OFFDIAG(Tl208, Pb208, 1.0  );
  OFFDIAG(Hg206, Tl206, 1.0  );
  OFFDIAG(Tl206, Pb206, 1.0  );

//   decays from nuclides in A=4n+1 sequence
```

```
//   decays from nuclides in A=4n+2 sequence

//   decays from nuclides in A=4n+3 sequence

  return 0;
}
```

6-6) Data routine for the Neptunium chain

```
/*
set_lambda.c : set decay rate values for decay chains of Actinides
2014/9/8-10 created
*/

#include <stdio.h>
#include <stdlib.h>  // for exit()

#include "decay_chain_ac.h"

#define DIAGELEM(NC,HL,TU) {nuclide[NC]=#NC;\
                            halflife[NC]=time_in_sec(HL,TU);\
                            lambda[NC][NC]=-halflife_to_decayrate(halflife[NC]);}
// NC:nuclide, HL:halflife, TU:time unit

#define OFFDIAG(P,D,BR); lambda[D][P]=-lambda[P][P]*(BR)
// P:parent nuclide, D:daughter nuclide, BR:branching ratio

int set_lambda(){
  enum{Np237,
       Pa233,
       U233,
       Th229,
       Ra225,
       Ac225,
       Fr221,
       Ra221,
       At217,
       Rn217,
       Bi213,
       Po213,
       Bi211,
       Po211,
       Tl209,
       Pb209,
       Bi209,
       Tl207,
       Pb207,
```

```
        Hg205,
        Tl205,
        FP,
        END};
  int i,j;
  double r,r2,r3; // branching ratio
  nn=END; // the number of nuclides, nn is a global variable
  if(M<nn) exit(1);
  for(i=0;i<nn;i++){for(j=0;j<nn;j++)lambda[i][j]=0;}

// diagonal elements of decay-rate matrix
//   nuclides in A=4n sequence

//   nuclides in A=4n+1 sequence
  DIAGELEM(Np237, 2.144e6   ,year );
  DIAGELEM(Pa233, 26.975    ,day );
  DIAGELEM(U233,  1.592e5   ,year );
  DIAGELEM(Th229, 7932      ,year );
  DIAGELEM(Ra225, 14.9      ,day );
  DIAGELEM(Ac225, 10.0      ,day );
  DIAGELEM(Fr221, 4.9     ,min );
  DIAGELEM(Ra221, 28     ,sec );
  DIAGELEM(At217, 32.3e-3   ,sec );
  DIAGELEM(Rn217, 0.54e-3   ,sec );
  DIAGELEM(Bi213, 45.59     ,min );
  DIAGELEM(Po213, 3.72e-6   ,sec );
  DIAGELEM(Bi211, 2.14     ,min );
  DIAGELEM(Po211, 0.516     ,sec );
  DIAGELEM(Tl209, 2.161     ,min );
  DIAGELEM(Pb209, 3.253     ,hour );
  DIAGELEM(Bi209, BIGNUM    ,sec );
  DIAGELEM(Tl207, 4.77     ,min );
  DIAGELEM(Pb207, BIGNUM     ,min );
  DIAGELEM(Hg205, 5.14     ,min );
  DIAGELEM(Tl205, BIGNUM     ,min );
  DIAGELEM(FP,    BIGNUM     ,min );
//   nuclides in A=4n+2 sequence

//   nuclides in A=4n+3 sequence

// off-diagonal elements of decay-rate matrix
//   decays from nuclides in A=4n sequence

//   decays from nuclides in A=4n+1 sequence
  r=2.0e-12;
  OFFDIAG(Np237, Pa233, 1-r  );
  OFFDIAG(Np237, FP,    r  );
  OFFDIAG(Pa233,  U233,  1.0  );
  r=9e-12;
  r2=6e-13;
```

```
  r3=1e-15;
  OFFDIAG( U233, Th229, 1-r-r2-r3  );
  OFFDIAG( U233, Pb209, r  );
  OFFDIAG( U233, FP,    r2  );
  OFFDIAG( U233, Hg205, r3  );
  OFFDIAG(Th229, Ra225, 1.0  );
  OFFDIAG(Ra225, Ac225, 1.0  );
  r=4e-14;
  OFFDIAG(Ac225, Fr221, 1-r  );
  OFFDIAG(Ac225, Bi211, r  );
  r=0.1e-2;
  OFFDIAG(Fr221, At217, 1-r  );
  OFFDIAG(Fr221, Ra221, r  );
  r=1e-14;
  OFFDIAG(Ra221, Rn217, 1-r  );
  OFFDIAG(Ra221, Pb207, r  );
  r=7.0e-5;
  OFFDIAG(At217, Bi213, 1-r  );
  OFFDIAG(At217, Rn217, r  );
  OFFDIAG(Rn217, Po213, 1.0  );
  r=2.20e-2;
  OFFDIAG(Bi213, Po213, 1-r  );
  OFFDIAG(Bi213, Tl209, r    );
  OFFDIAG(Po213, Pb209, 1.0  );
  r=0.28e-2;
  OFFDIAG(Bi211, Tl207, 1-r  );
  OFFDIAG(Bi211, Po211, r  );
  OFFDIAG(Po211, Pb207, 1.0  );
  OFFDIAG(Tl209, Pb209, 1.0  );
  OFFDIAG(Pb209, Bi209, 1.0  );
  OFFDIAG(Tl207, Pb207, 1.0  );
  OFFDIAG(Hg205, Tl205, 1.0  );
//   decays from nuclides in A=4n+2 sequence

//   decays from nuclides in A=4n+3 sequence

  return 0;
}
```

6-7) Data routine for the Uranium chain

```
/*
set_lambda.c : set decay rate values for decay chains of Actinides
2014/9/8-10 created
*/

#include <stdio.h>
#include <stdlib.h>  // for exit()
```

```
#include "decay_chain_ac.h"

#define DIAGELEM(NC,HL,TU) {nuclide[NC]=#NC;\
                            halflife[NC]=time_in_sec(HL,TU);\
                            lambda[NC][NC]=-halflife_to_decayrate(halflife[NC]);}
// NC:nuclide, HL:halflife, TU:time unit

#define OFFDIAG(P,D,BR); lambda[D][P]=-lambda[P][P]*(BR)
// P:parent nuclide, D:daughter nuclide, BR:branching ratio

int set_lambda(){
  enum{U238,
       Th234,
       Pa234m,
       Pa234,
       U234,
       Th230,
       Ra226,
       Rn222,
       Po218,
       At218,
       Rn218,
       Pb214,
       Bi214,
       Po214,
       Pb212,
       Bi212,
       Po212,
       Hg210,
       Tl210,
       Pb210,
       Bi210,
       Po210,
       Pb209,
       Bi209,
       Tl208,
       Pb208,
       Hg206,
       Tl206,
       Pb206,
       FP,
       END};
  int i,j;
  double r,r2,r3; // branching ratio
  nn=END; // the number of nuclides, nn is a global variable
  if(M<nn) exit(1);
  for(i=0;i<nn;i++){for(j=0;j<nn;j++)lambda[i][j]=0;}

// diagonal elements of decay-rate matrix
```

```
//   nuclides in A=4n sequence

//   nuclides in A=4n+1 sequence

//   nuclides in A=4n+2 sequence
  DIAGELEM(U238, 4.468e9   ,year );
  DIAGELEM(Th234, 24.10    ,day );
  DIAGELEM(Pa234m, 1.159   ,min );
  DIAGELEM(Pa234, 6.70   ,hour );
  DIAGELEM(U234, 2.455e5      ,year );
  DIAGELEM(Th230, 7.54e4      ,year );
  DIAGELEM(Ra226, 1600      ,year );
  DIAGELEM(Rn222, 3.8235     ,day );
  DIAGELEM(Po218, 3.098   ,min );
  DIAGELEM(At218, 1.5     ,sec );
  DIAGELEM(Rn218, 35e-3   ,sec );
  DIAGELEM(Pb214, 26.8     ,min );
  DIAGELEM(Bi214, 19.9     ,min );
  DIAGELEM(Po214, 164.3e-6     ,min );
  DIAGELEM(Pb212, 10.64    ,hour);
  DIAGELEM(Bi212, 60.55    ,min);
  DIAGELEM(Po212, 0.299e-6    ,sec);
  DIAGELEM(Hg210, 300e-9     ,sec );
  DIAGELEM(Tl210, 1.30     ,min );
  DIAGELEM(Pb210, 22.20     ,year );
  DIAGELEM(Bi210, 5.012     ,day );
  DIAGELEM(Po210, 138.376     ,day );
  DIAGELEM(Pb209, 3.253     ,hour );
  DIAGELEM(Bi209, BIGNUM    ,sec );
  DIAGELEM(Tl208, 3.053   ,min);
  DIAGELEM(Pb208, BIGNUM    ,sec );
  DIAGELEM(Hg206, 8.32    ,min );
  DIAGELEM(Tl206, 4.202    ,min );
  DIAGELEM(Pb206, BIGNUM    ,sec );
  DIAGELEM(FP,    BIGNUM   ,sec)
//   nuclides in A=4n+3 sequence

// off-diagonal elements of decay-rate matrix
//   decays from nuclides in A=4n sequence

//   decays from nuclides in A=4n+1 sequence

//   decays from nuclides in A=4n+2 sequence
  r=5.5e-7
  OFFDIAG(U238, Th234, 1-r  );
  OFFDIAG(U238, FP,  r  );
  OFFDIAG(Th234, Pa234m, 1.0  );
  r=0.16e-2
  OFFDIAG(Pa234m, U234, 1-r  );
  OFFDIAG(Pa234m, Pa234, r  );
```

```
  OFFDIAG(Pa234, U234, 1.0  );
  r=1.6e-11;
  r2=1e-13;
  r3=9e-14;
  OFFDIAG(U234, Th230, 1-r-r2-r3  );
  OFFDIAG(U234, FP, r  );    //SF
  OFFDIAG(U234, Hg210, r2  );   //Mg
  OFFDIAG(U234, Pb214, r3  );   //Ne
  r=6e-13;
  r2=4e-14;
  OFFDIAG(Th230, Ra226, 1-r-r2 );
  OFFDIAG(Th230, Hg206, r    );   //24Ne
  OFFDIAG(Th230, FP , r2  );
  r=3.2e-11;
  OFFDIAG(Ra226, Rn222, 1-r  );
  OFFDIAG(Ra226, Pb212, r  );  //14C
  OFFDIAG(Rn222, Po218, 1.0  );
  r=0.02e-2;
  OFFDIAG(Po218, Pb214, 1-r  );
  OFFDIAG(Po218, At218, r  );
  r=0.10e-2;
  OFFDIAG(At218, Bi214, 1-r );
  OFFDIAG(At218, Rn218, r );
  OFFDIAG(Rn218, Po214, 1.0  );
  OFFDIAG(Pb214, Bi214, 1.0 );
  r=0.02e-2;
  OFFDIAG(Bi214, Po214, 1-r  );
  OFFDIAG(Bi214, Tl210, r  );
  OFFDIAG(Po214, Pb210, 1.0  );
  OFFDIAG(Pb212, Bi212, 1.0 );
  r=35.94e-2
  OFFDIAG(Bi212, Po212, 1-r );
  OFFDIAG(Bi212, Tl208, r );
  OFFDIAG(Po212, Pb208, 1.0 );
  OFFDIAG(Hg210, Tl210, 1.0 );
  r=7.0e-5;
  OFFDIAG(Tl210, Pb210, 1-r  );
  OFFDIAG(Tl210, Pb209, r  ); //β n
  r=1.9e-8;
  OFFDIAG(Pb210, Bi210, 1-r  );
  OFFDIAG(Pb210, Hg206, r  );
  r=1.3e-6;
  OFFDIAG(Bi210, Po210, 1-r );
  OFFDIAG(Bi210, Tl206, r  );
  OFFDIAG(Po210, Pb206, r  );
  OFFDIAG(Pb209, Bi209, 1.0 );
  OFFDIAG(Tl208, Pb208, 1.0 );
  OFFDIAG(Hg206, Tl206, 1.0 );
  OFFDIAG(Tl206, Pb206, 1.0 );
//   decays from nuclides in A=4n+3 sequence
```

```
  return 0;
}
```

6-8) Data routine for the Actinide chain

```
/*
set_lambda.c : set decay rate values for decay chains of Actinides
2014/9/8-10 created
*/

#include <stdio.h>
#include <stdlib.h>  // for exit()

#include "decay_chain_ac.h"

#define DIAGELEM(NC,HL,TU) {nuclide[NC]=#NC;\
                            halflife[NC]=time_in_sec(HL,TU);\
                            lambda[NC][NC]=-halflife_to_decayrate(halflife[NC]);}
// NC:nuclide, HL:halflife, TU:time unit

#define OFFDIAG(P,D,BR); lambda[D][P]=-lambda[P][P]*(BR)
// P:parent nuclide, D:daughter nuclide, BR:branching ratio

int set_lambda(){
  enum{U235,
       Th231,
       Pa231,
       Ra227,
       Ac227,
       Th227,
       Fr223,
       Ra223,
       At219,
       Rn219,
       Pb215,
       Bi215, //
       Po215, //
       At215,
       Pb211,
       Bi211,
       Po211,
       Pb209,
       Bi209,
       Hg207,
       Tl207,
       Pb207,
       FP,
```

```
      END};
  int i,j;
  double r,r2,r3; // branching ratio
  nn=END; // the number of nuclides, nn is a global variable
  if(M<nn) exit(1);
  for(i=0;i<nn;i++){for(j=0;j<nn;j++)lambda[i][j]=0;}

// diagonal elements of decay-rate matrix
//   nuclides in A=4n sequence

//   nuclides in A=4n+1 sequence

//   nuclides in A=4n+2 sequence

//   nuclides in A=4n+3 sequence
  DIAGELEM(U235, 7.038e8   ,year );
  DIAGELEM(Th231, 25.52    ,hour );
  DIAGELEM(Pa231, 3.276e4   ,year );
  DIAGELEM(Ra227, 42.2      ,min );
  DIAGELEM(Ac227, 21.772      ,year );
  DIAGELEM(Th227, 18.68      ,day );
  DIAGELEM(Fr223, 22.00     ,min );
  DIAGELEM(Ra223, 11.43   ,day );
  DIAGELEM(At219, 56     ,sec );
  DIAGELEM(Rn219, 3.96   ,sec );
  DIAGELEM(Pb215, 147     ,sec );
  DIAGELEM(Bi215, 7.6     ,min );
  DIAGELEM(Po215, 1.781e-3     ,sec );
  DIAGELEM(At215, 0.10e-3     ,sec );
  DIAGELEM(Pb211, 36.1     ,min );
  DIAGELEM(Bi211, 2.14     ,min );
  DIAGELEM(Po211, 0.516     ,sec );
  DIAGELEM(Pb209, 3.253     ,hour );
  DIAGELEM(Bi209, BIGNUM    ,sec );
  DIAGELEM(Hg207, 2.9    ,min );
  DIAGELEM(Tl207, 4.77    ,min );
  DIAGELEM(Pb207, BIGNUM    ,sec );
  DIAGELEM(FP, BIGNUM    ,sec );
// off-diagonal elements of decay-rate matrix
//   decays from nuclides in A=4n sequence

//   decays from nuclides in A=4n+1 sequence

//   decays from nuclides in A=4n+2 sequence

//   decays from nuclides in A=4n+3 sequence
  r=7.0e-11;
  r2=8.0e-12;
  r3=8.0e-12;
  OFFDIAG(U235, Th231, 1-r-r2-r3  );
```

```
  OFFDIAG(U235, FP,  r  );
  OFFDIAG(U235, Hg207, r2  );
  OFFDIAG(U235, Pb215, r3  );
  r=4.0e-13;
  OFFDIAG(Th231, Pa231, 1-r );
  OFFDIAG(Th231, Ra227, r  );
  r=2e-13;
  OFFDIAG(Pa231, Ac227, 1-r );
  OFFDIAG(Pa231, FP, r  );
  OFFDIAG(Ra227, Ac227, 1.0  );
  r=1.38e-2;
  OFFDIAG(Ac227, Th227, 1-r );
  OFFDIAG(Ac227, Fr223, r  );
  OFFDIAG(Th227, Ra223, 1.0  );
  r=6.0e-5;
  OFFDIAG(Fr223, Ra223, 1-r );
  OFFDIAG(Fr223, At219, r  );
  r=8.9e-10;
  OFFDIAG(Ra223, Po215, 1-r );
  OFFDIAG(Ra223, Pb209, r  );
  r=3.00e-2;
  OFFDIAG(At219, Bi215, 1-r );
  OFFDIAG(At219, Rn219, r  );
  OFFDIAG(Rn219, Po215, 1.0  );
  OFFDIAG(Pb215, Bi215, 1.0  );
  OFFDIAG(Bi215, Po215, 1.0  );
  r=2.3e-6;
  OFFDIAG(Po215, Pb211, 1-r );
  OFFDIAG(Po215, At215, r  );
  OFFDIAG(At215, Bi211, 1.0  );
  OFFDIAG(Pb211, Bi211, 1.0  );
  r=0.28e-2;
  OFFDIAG(Bi211, Tl207, 1-r );
  OFFDIAG(Bi211, Po211, r  );
  OFFDIAG(Po211, Pb207, 1.0  );
  OFFDIAG(Pb209, Bi209, 1.0  );
  OFFDIAG(Hg207, Tl207, 1.0  );
  OFFDIAG(Tl207, Pb207, 1.0  );
  return 0;
}
```

6-9) Data routine for the chain from Uut

```
/*
set_lambda.c : set decay rate values for decay chains of Actinides
2014/9/8-10 created
*/
```

```
#include <stdio.h>
#include <stdlib.h>  // for exit()

#include "decay_chain_ac.h"

#define DIAGELEM(NC,HL,TU) {nuclide[NC]=#NC;\
                            halflife[NC]=time_in_sec(HL,TU);\
                            lambda[NC][NC]=-halflife_to_decayrate(halflife[NC]);}
// NC:nuclide, HL:halflife, TU:time unit

#define OFFDIAG(P,D,BR) lambda[D][P]=-lambda[P][P]*(BR)
// P:parent nuclide, D:daughter nuclide, BR:branching ratio

int set_lambda(){
  enum{Uut278,
       Rg274,
       Mt270,
       Bh266,
       Db262,
       Lr258,
       Md254,
       Fm254,
       Cf250,
       Cm246,
       Pu242,
       U238,
       FP,
       END};
  int i,j;
  double r; // branching ratio
  nn=END; // the number of nuclides, nn is a global variable
  if(M<nn) exit(1);
  for(i=0;i<nn;i++){for(j=0;j<nn;j++)lambda[i][j]=0;}

// diagonal elements of decay-rate matrix
  DIAGELEM(Uut278, 0.24e-3     ,sec );
  DIAGELEM(Rg274, 6.4e-3      ,sec );
  DIAGELEM(Mt270, 2.0e-3      ,sec );
  DIAGELEM(Bh266, 1.7      ,sec );
  DIAGELEM(Db262, 35      ,sec );
  DIAGELEM(Lr258, 4.1      ,sec );
  DIAGELEM(Md254, 28   ,min);
  DIAGELEM(Fm254, 3.240   ,hour);
  DIAGELEM(Cf250, 13.08   ,year);
  DIAGELEM(Cm246, 4706   ,year);
  DIAGELEM(Pu242, 3.75e5   ,year);
  DIAGELEM(U238, BIGNUM   ,sec);
  DIAGELEM(FP, BIGNUM      ,sec );
```

```
// off-diagonal elements of decay-rate matrix
  OFFDIAG(Uut278, Rg274, 1.0  );
  OFFDIAG(Rg274, Mt270, 1.0  );
  OFFDIAG(Mt270, Bh266, 1.0  );
  OFFDIAG(Bh266, Db262, 1.0  );
  r=0.33;
  OFFDIAG(Db262, Lr258, 1-r  );
  OFFDIAG(Db262, FP,    r    );
  r=0.05;
  OFFDIAG(Lr258, Md254, 1-r );
  OFFDIAG(Lr258, FP, r  );
  OFFDIAG(Md254, Fm254, 1.0  );
  r=0.06e-2;
  OFFDIAG(Fm254, Cf250, 1-r  );
  OFFDIAG(Fm254, FP, r  );
  r=0.08e-2;
  OFFDIAG(Cf250, Cm246, 1-r  );
  OFFDIAG(Cf250, FP, r  );
  r=0.03e-2;
  OFFDIAG(Cm246, Pu242, 1-r  );
  OFFDIAG(Cm246, FP, r  );
  r=5.5e-6;
  OFFDIAG(Pu242, U238, 1-r  );
  OFFDIAG(Pu242, FP, r  );
  return 0;
}
```

6-10) Data routine for the chain from Uuo

```
/*
set_lambda.c : set decay rate values for decay chains of Actinides
2014/9/8-10 created
*/

#include <stdio.h>
#include <stdlib.h>  // for exit()

#include "decay_chain_ac.h"

#define DIAGELEM(NC,HL,TU) {nuclide[NC]=#NC;\
                            halflife[NC]=time_in_sec(HL,TU);\
                            lambda[NC][NC]=-halflife_to_decayrate(halflife[NC]);}
// NC:nuclide, HL:halflife, TU:time unit

#define OFFDIAG(P,D,BR); lambda[D][P]=-lambda[P][P]*(BR)
// P:parent nuclide, D:daughter nuclide, BR:branching ratio

int set_lambda(){
```

```
  enum{Uuo294,
       Lv290,
       Fl286,
       Cn282,
       FP,
       END};
  int i,j;
  double r; // branching ratio
  nn=END; // the number of nuclides, nn is a global variable
  if(M<nn) exit(1);
  for(i=0;i<nn;i++){for(j=0;j<nn;j++)lambda[i][j]=0;}

// diagonal elements of decay-rate matrix
  DIAGELEM(Uuo294, 0.89e-3   ,sec );
  DIAGELEM(Lv290, 15e-3   ,sec );
  DIAGELEM(Fl286, 0.16   ,sec );
  DIAGELEM(Cn282, 0.50e-3   ,sec );
  DIAGELEM(FP, BIGNUM   ,sec );

// off-diagonal elements of decay-rate matrix
  OFFDIAG(Uuo294, Lv290,  1.0  );
  OFFDIAG(Lv290, Fl286,   1.0  );
  r=0.4
  OFFDIAG(Fl286, FP,  1-r );
  OFFDIAG(Fl286, Cn282, r   );
  OFFDIAG(Cn282, FP,     1.0   );

  return 0;
}
```

7) 放射性崩壊系列の総放射能量をベクレル単位で計算するプログラム

```
/*
総放射線量を計算するプログラム。
2014/10/29 created Matsuoka Hirokazu
*/


#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include <string.h> // for strncmp()

int main(){

  FILE *fp; // file to write time evolution
  FILE *fp2;
  char buffer[256];
```

```
  double t,n,d;
  int i;
  int i0=-1;
  double dsum=0.0;
  double t0=-1.0;

  fp=fopen("time_evolution.dat","rt");
  fp2=fopen("totBq.dat","wt");
  while(fgets(buffer,256,fp)!=NULL){
    //printf("%s",buffer);
    sscanf(buffer,"%le %le %le %d",&t,&n,&d,&i);
    if(i!=i0){
      if(i0>=0)printf("%e %e :totbq\n",t0,dsum);
      if(i0>=0)fprintf(fp2,"%e %e\n",t0,dsum);
      i0=i;
      t0=t;
      dsum=0.0;
    }
    dsum+=d;
    //printf("%e %e %e %d\n",t,n,d,i);
  }
  printf("%e %e :totbq\n",t0,dsum);
  fprintf(fp2,"%e %e\n",t0,dsum);
  close(fp);
  close(fp2);
}
```