JAERI-M
94-049

DANKE:球,棒,平板に対するダンコフ補正因子モンテカルロ計算プログラム

1994年3月

奥野　浩・小室　雄一

日本原子力研究所
Japan Atomic Energy Research Institute

JAERI-Mレポートは，日本原子力研究所が不定期に公刊している研究報告書です。
入手の問合わせは，日本原子力研究所技術情報部情報資料課（〒319-11茨城県那珂郡東海村）あて，お申しこしください。なお，このほかに財団法人原子力弘済会資料センター（〒319-11 茨城県那珂郡東海村日本原子力研究所内）で複写による実費頒布をおこなっております。

JAERI-M reports are issued irregularly.
Inquiries about availability of the reports should be addressed to Information Division, Department of Technical Information, Japan Atomic Energy Research Institute, Tokai-mura, Naka-gun, Ibaraki-ken 319-11, Japan.

編集兼発行　日本原子力研究所
印　　刷　㈱原子力資料サービス

JAERI-M 94-049

DANKE：球，棒，平板に対するダンコフ補正因子モンテカルロ計算プログラム

日本原子力研究所東海研究所燃料サイクル安全工学部
奥野　浩・小室　雄一

（1994年2月7日受理）

燃料棒（あるいは燃料塊）の配列において，隣接する燃料棒（あるいは燃料塊）による吸収のため，共鳴エネルギー領域の中性子束は相当減少する。ダンコフ補正因子は，実効多群断面積を求める際にこの非均質効果を表すのに用いられる。モンテカルロ法を用いてダンコフ補正因子を計算するプログラムMCDANを検討したところ，(1)燃料棒の配列体系において軸方向への中性子の飛行が考慮されていない，(2) MCDANでは中性子発生点を決めるのに，円（または球）内の半径方向に一様乱数を用いているが，これによって燃料表面から余弦分布する保証は得られない，という2点が明らかとなった。このため，これらの点を修正するとともに，板状燃料の配列及び燃料棒の六角格子配列へ適用できるようにMCDANプログラムを拡張した。改良版をDANKEプログラムと名付けた。DANKEは臨界安全性評価コードシステムJACS中の輸送計算用実効断面積作成コードMAIL 3.0に組込まれた。本報告には，DANKEプログラムで採用している基礎式，プログラムの使用手引き及び他手法との比較計算を記す。

東海研究所：〒319－11　茨城県那珂郡東海村白方字白根2－4

JAERI-M 94-049

DANKE: A Monte Carlo Computer Program for Calculating the Dancoff Correction Factor for Spheres, Rods and Slabs

Hiroshi OKUNO and Yuichi KOMURO

Department of Fuel Cycle Safety Research
Tokai Research Establishment
Japan Atomic Energy Research Institute
Tokai-mura, Naka-gun, Ibaraki-ken

(Received February 7, 1994)

In an array of fuel rods or fuel lumps, a flux of neutrons in the resonance energy region considerably decreases through absorption by the neighboring fuel rods or fuel lumps. The Dancoff correction factor is used to express this heterogeneity effect in obtaining the effective multigroup cross sections. Examination of MCDAN, a computer program for calculating the Dancoff correction factor using the Monte Carlo method, revealed two facts: (1) axial movement of neutrons were not considered arrays of fuel rods, (2) random distribution in radial argument of circle or sphere could not reproduce cosine distribution of neutrons at a surface point. Therefore, the program was revised for these points and further extended to be applicable to arrays of planes and hexagonal arrays of rods. The improved version was called DANKE program, which was incorporated into MAIL3.0, a computer code generating effective cross section sets for transport codes, in the nuclear criticality safety evaluation code system JACS. This report includes basic expressions that DANKE program adopts, a program manual and comparisons with calculations based on other methods.

Keywords: DANKE, MCDAN, Dancoff Correction Factor, Monte Carlo, Resonance Integral, MAIL3.0, JACS, Nuclear Criticality, Safety

目　　次

Contents

1. はじめに

燃料棒（あるいは燃料塊）の配列において、隣接する燃料棒（あるいは燃料塊）による吸収のため、共鳴エネルギー領域の中性子束は相当減少する。ダンコフ補正因子は、実効多群断面積を求める際にこの非均質効果を表すのに用いられる[1]。

ダンコフ補正因子を用いないと、典型的な軽水炉の炉心体系では中性子増倍率の計算で２～３％程度の誤差を生じてしまう。また、直径0.1mmのUO_2粒子の立方配列がガドリニウム入り硝酸に浸っている体系（ＯＥＣＤ問題19）では、ダンコフ補正因子の計算誤差が、10％以上も中性子増倍率を低く見積ってしまう結果を導いた[2]。

ダンコフ補正因子の値を求めるには、古くは表[3]を参照して内挿していた。やがて、計算機が発達するのに伴い、必要に応じて計算するようになってきた。決定論的な方法に従って計算する深井の式[4]、SUPERDANコード[5]、モンテカルロ法に従って計算するプログラムMCDAN[6]などが知られている。このうちMCDANは、棒状燃料の正方格子配列及び球状燃料の立方格子配列に適用できると報告書に記されている。ソースプログラムも報告書に含まれており、理解・応用が容易である。しかし中身を検討したところ、次のことが明らかとなった。

(1) MCADANではセルの周期性を用いて燃料の無限配列を模擬して中性子を追跡している。これはモンテカルロ法の特性をうまく利用した方法と言える。

(2) 燃料棒の配列体系において軸方向への中性子の飛行が考慮されていない。

(3) MCDANプログラムでは、燃料領域を表す円（または球）内の半径方向に一様乱数を用いて中性子発生点を決めているが、これによって燃料表面から余弦分布する保証はない。

このため、(2)及び(3)を修正するとともに、MCDANプログラムでは取扱えなかった板状燃料の配列及び燃料棒の六角格子配列へ適用できるようにプログラムを拡張した。改良版の計算プログラムをDANKEと名付けた。DANKEは臨界安全性評価コードシステムJACS[7]中の実効断面積作成コードMAIL3.0[8]に組込まれた。本報告には、次章以降にDANKEプログラムで採用している基礎式、プログラム構成、入力データ形式、計算例のほか、他の手法により求めた計算との比較結果をまとめる。また、付録にはソースプログラムのリストを記載する。

2. 基　礎　式

減速材中に置かれた燃料塊（あるいは燃料棒）の配列を想定し、中性子が燃料中で一様・等方に発生するものとする。このとき、燃料中で発生して燃料中では一度も衝突しなかった中性子が、減速材中でも衝突することなく他の燃料表面に到達する確率はダンコフ補正因子と呼ばれている。このときダンコフ補正因子は、燃料表面から余弦分布に従って発生した中性子が減速材中において一度も衝突することなく他の燃料表面に到達する確率として、次式のCで表される[9,10]。

$$C = \frac{\int\int_{n\cdot\Omega>0} (n\cdot\Omega)\exp[-\Sigma_m R(S,\Omega)]dS\,d\Omega}{\int\int_{n\cdot\Omega>0} (n\cdot\Omega)dS\,d\Omega} \qquad (1)$$

ここで

$R(S,\Omega)$：　燃料表面間の距離、

Σ_m　：　減速材の全断面積、

n　：　燃料表面外向き法線方向の単位ベクトル、

Ω　：　中性子の単位速度ベクトル（$= v/|v|$）、

dS　：　燃料表面積の微小要素、

である。

（1）　板状燃料の配列

平板燃料の配列では燃料表面はどこも一様なので、(1)式において燃料表面積についての積分は落ちて、

$$C = \frac{\int_0^{\pi/2} \cos\theta\cdot\exp[-\Sigma_m L/\cos\theta]\cdot 2\pi\sin\theta\,d\theta}{\int_0^{\pi/2} \cos\theta\cdot 2\pi\sin\theta\,d\theta} \qquad (2)$$

となる。ここで、

θ ： 中性子の進行方向と燃料表面外側法線方向とのなす角、

L ： 燃料面間距離、

である。

さらに、$\cos^2\theta = u$ と置くことにより、

$$C = \int_0^1 du \cdot \exp[-\Sigma_m L/\sqrt{u}] \qquad (3)$$

という関係が得られる。DANKEプログラムでは、uに(0,1)の一様乱数を用いることによりダンコフ補正因子を求めている。(3)式がDANKEプログラムで用いている板状燃料配列でのダンコフ補正因子の表式である。

なお、$u = 1/t^2$ と置くことにより、

$$C = 2E_3(\Sigma_m L) \qquad (4)$$

という関係が得られる。ここでE_3は指数積分関数[11]で、一般に

$$E_n(x) = \int_1^\infty dt \cdot \exp[-xt]/t^n, \quad (n = 0,1,2,\ldots) \qquad (5)$$

で定義される。

（2） 棒状燃料の正方及び六方格子配列

円柱の形をした棒状燃料の配列に対しては、軸方向一様なので、(1)式において表面積分として円の表面を考慮すればよい。このとき、

$$(n \cdot \Omega) = \cos\phi_\rho \cdot \cos\theta_z$$

$$\int_{\text{外側}} d\Omega\ [\cdots\cdots] = \int_{-\pi/2}^{\pi/2} \cos\theta_z d\phi_\rho \cdot \int_{-\pi/2}^{\pi/2} d\theta_z\ [\cdots\cdots]$$

$$\int ds\ [\cdots\cdots] = \int_0^{2\pi} \rho d\phi\ [\cdots\cdots]$$

ここで

θ_z ： 水平面（z軸に垂直な面）と中性子の進行方向のなす角、

ϕ_ρ ： 中性子の進行方向を水平面に垂直に投影した方向と燃料表面外側法線方向とのなす角、

ϕ ： 燃料棒表面（円）上の中性子発生点を表す角、

ρ ： 燃料棒の半径

であることに注意すると、(1)式は、

$$C = \frac{\rho \int_0^{2\pi} d\phi \int_{-\pi/2}^{\pi/2} \cos\phi_\rho d\phi_\rho \int_{-\pi/2}^{\pi/2} \cos^2\theta_z \exp[-\Sigma_m |\rho-\rho'|/\cos\theta_z] \cdot d\theta_z}{\rho \int_0^{2\pi} d\phi \int_{-\pi/2}^{\pi/2} \cos\phi_\rho d\phi_\rho \int_{-\pi/2}^{\pi/2} \cos^2\theta_z d\theta_z} \quad (6)$$

となる。なお、このとき中性子の進行方向のx、y成分はそれぞれ、$\Omega_x = \cos(\phi+\phi_\rho)$、$\Omega_y = \sin(\phi+\phi_\rho)$となる。

(6)式において、分母$=2\pi^2\rho$ であり、[0,1]の一様乱数 u_1、u_2、u_3 を用いて、

$\phi = 2\pi u_1$、

$\sin\phi_\rho = 2u_2-1$、

$\sin\theta_z = u_3$

と置くと、

$$C = \frac{1}{\pi} \int_0^1 du_1 \int_0^1 du_2 \int_0^1 du_3 (1-u_3^2)^{1/2} \exp[-\Sigma_n |\rho-\rho'|/(1-u_3^2)^{1/2}] \quad (7)$$

が得られる。$\rho'-\rho$の方向がΩの方向に一致するので、中性子が他の燃料棒と衝突するまで中性子を追跡するとよいことが分る。(7)式がDANKEプログラムで用いている棒状燃料配列でのダンコフ補正因子の表式である。正方格子と六角格子の違いは各セルの周期境界条件設定の仕方の違いのみに現れる。

なお、次式で定義されるBickley関数[12]

$$Ki_n(x) = \int_0^{\pi/2} d\theta \sin^{n-1}\theta \cdot \exp[-x/\sin\theta] \quad (8)$$

を用いると、(6)式から

$$C = \frac{1}{\pi^2}\int_0^{2\pi} d\phi \int_{-\pi/2}^{\pi/2} \cos\phi_\rho d\phi_\rho Ki_3(\Sigma_m|\rho-\rho'|) \qquad (9)$$

となる。

（3） 球状燃料の立方格子配列

球状燃料の配列に対しては、(1)式において、

$$(n\cdot\Omega) = \cos\theta'$$

$$\int_{外側} d\Omega\,[\cdots\cdots] = \int_0^{\pi/2} d\theta' \int_0^{2\pi} \sin\theta' d\phi'\,[\cdots\cdots]$$

$$\int dS\,[\cdots\cdots] = \int_0^{\pi} \rho d\theta \int_0^{2\pi} \rho\sin\theta d\phi\,[\cdots\cdots]$$

ここで

θ'： 中性子発生点での燃料表面外側法線方向と中性子の進行方向のなす角、

ϕ'： nの方向をz'軸、n×Ωの方向をy'軸にとる座標系でのΩの方位角、

θ ： 燃料球面上で中性子発生点を表す天頂角、

ϕ ： 燃料球面上で中性子発生点を表す方位角、

ρ ： 燃料球の半径

であるから、(1)式の分母が$4\pi^2\rho^2$であり、[0,1]の一様乱数u_1、u_2、u_3、u_4を用いて、

$\phi = 2\pi u_1$、

$\cos\theta = 2u_2-1$、

$\phi' = 2\pi u_3$、

$\cos 2\theta' = 2u_4-1$

と置くと、

$$C = 2\pi^2\rho^2 \int_0^1 du_1 \int_0^1 du_2 \int_0^1 du_3 \int_0^1 du_4 \exp[-\Sigma_n |r-r'|] \qquad (10)$$

が得られる。r'-rの方向がΩの方向に一致することより、中性子が他の燃料球と衝突するまで中性子を追跡するとよいことが分る。(10)式がDANKEプログラムで用いている球状燃料配列でのダンコフ補正因子の表式である。

なお、Ωの方向余弦は

$$\Omega \cdot e_i = \sum_j (\Omega \cdot e'_j)(e' \cdot e_i),$$

の式に沿って求めた。ここで、i，j = x，y，z である。

3. プログラムの構成

DANKEコードにおけるプログラムの樹状構造を図1に示す。主プログラム及び各副プログラムの機能をアルファベット順に以下に示す。

(1) DANC

1個の中性子が他の燃料の表面に到達するまでの距離TOTXを計算し、そのときの到達率CIJを計算する。

(2) DRC

球状燃料の配列体系において方向余弦（direction cosine）を計算する。

(3) FLTRN

0と1の間の一様乱数を選ぶ。多群エネルギーのモンテカルロ臨界計算コードKENO－Ⅳ[13]より借用した。

(4) HIT

立方体のセルにおいて、セルの中心に位置する球に中性子が衝突するか否かを判定する。衝突する場合には、球に衝突するまでの飛行距離も計算する。

(5) HIT2

正方形のセルにおいて、セルの中心に位置する円に中性子が衝突するか否かを判定する。衝突する場合には、円に衝突するまでの飛行距離も計算する。

(6) INSECT

中性子と立方体セル境界との交差点の座標及び交差点までの中性子の飛行距離、さらに、周期性を用いて、セル内の再入射可能な位置に中性子の座標を戻す。

(7) INSECT2

中性子と正方形セル境界との交差点の座標及び交差点までの中性子の飛行距離、さらに、周期性を用いて、セル内の再入射可能な位置に中性子の座標を戻す。

(8) INSECT3

辺の比が1:√3の長方形セルにおいて、中性子が対辺の２隅の四半円あるいはセル中心の円に衝突するか否かを判定する。衝突する場合には、円に衝突するまでの飛行距離も計算する。衝突しない場合は、中性子と長方形セル境界との交差点の座標及び交差点までの中性子の飛行距離、さらに、周期性を用いて、セル内の再入射可能な位置に中性子の座標を戻す。

(9) MAIN

中性子の到達率CIJの計算をNBバッチに渡りNPB個の粒子について平均を求めることにより、ダンコフ補正因子を計算する主プログラム。

(10) READIN

入力データを読み込み、計算に必要な定数を設定する。

(11) START

球状燃料表面上の中性子発生点の座標と中性子が飛行する方向の方向余弦を決める。さらに、セルを規定する６面のうち、中性子の進行方向にある３つの面を求める。

(12) START2

棒状燃料水平断面表面（円）上の中性子発生点の座標と中性子が飛行する方向の方向余弦を決める。さらに、正方形セルを規定する４辺のうち、中性子の進行方向にある２つの辺を求める。

(13) START3

棒状燃料水平断面表面（円）上の中性子発生点の座標と中性子が飛行する方向の方向余弦を決める。さらに、辺の比が1:√3の長方形セルを規定する４辺のうち、中性子の進行方向にある２つの辺を求める。

4. 入力データ形式

カード1　　FORMAT(I1)

カラム 1　　ITYPE ： 燃料の配列型

=1　板状燃料の配列

=2　棒状燃料の正方格子配列

=3　棒状燃料の六角格子配列

=4　球状燃料の立方格子配列

カード2　　FORMAT(3F10)

カラム 1-10　R ： 燃料棒または燃料球の半径（ITYPE=2-4）［cm］；板状燃料の厚さ（ITYPE=1）［cm］

11-20　SIG ： 減速材の巨視的断面積［cm^{-1}］

21-30　VF ： 球状燃料配列中での燃料占拠率（ITYPE=4）［－］；棒状燃料配列または平板燃料配列のピッチ（ITYPE=2-4）［cm］

註：一世代当りの中性子数及び計算世代数は入力で指定するのではなく、プログラム内で定数としている。

5. 計　算　例

水減速材中に直径１㎜（半径0.05㎜）の球状燃料が立方格子状に無限配列された体系のダンコフ補正因子を計算する。水対燃料体積比V_M/V_Fが3.83のときにはセル内の燃料占拠率$V_F/(V_F+V_M)=0.207$である。また、共鳴エネルギー領域での水の全断面積として1.49cm^{-1}を用いる。計算の結果、ダンコフ補正因子は１世代当り5，000個の中性子を発生させ、5世代当りの平均値として0.722（標準偏差0.003）という値を得た。

この計算におけるジョブ制御文及び入力データ、それに出力リストを以下に示す。

(1) ジョブ制御文及び入力データ

```
   T(00) W(00) C(01) E(00) I(00)
//*********************<<J3755.CODE.CNTL(DANKE2)>>************************
// EXEC FORTEX,SO='J3755.DANKE2',Q='.FORT77',A='ELM(*),SOURCE',
//       B=NOMAP,LCT=62
// EXEC     LKEDEX,SSLB='NO'
// EXEC     GOEX
//SYSIN     DD  *
4
      0.05      1.490      0.207
```

(2) 出力リスト

```
R =  5.00000E-02 SIG =  1.49000E+00 XMAX =  6.71141E+00
VF =  2.07000E-01 PITCH =  1.36252E-01
NUMBER OF BATCHES =    5 NEUTRONS / BATCH =      5000
BATCH        C          AVGC                AVGX              NHITS
   1 7.23546E-01 7.23546E-01 0.00000E+00 2.48304E-01       5000
   2 7.22171E-01 7.22858E-01 9.76563E-04 2.49099E-01       4999
   3 7.24154E-01 7.23290E-01 9.76563E-04 2.50607E-01       4999
   4 7.18140E-01 7.22003E-01 2.76214E-03 2.56356E-01       4999
   5 7.21775E-01 7.21957E-01 2.39208E-03 2.51136E-01       5000
```

6. 比 較 計 算

棒状燃料配列に対するDANKEプログラムの妥当性を検討するため、別のダンコフ補正因子計算手法（または計算コード）から得られる値と比較した。表１は日本原子力研究所の軽水臨界実験装置ＴＣＡの標準燃料棒（ペレット直径1.25cm）が正方格子配列をしている体系で燃料棒ピッチを1.5cmから4.0cmの間で変えたときのダンコフ補正因子を求めたものである。計算にはDANKEプログラムのほか、MCDANプログラム[6]、MAILコード[14]に内蔵されている深井の式[4]（決定論的方法）の３つを比較した。表１から次のことが明らかになった。①MCADNプログラムにより求めたダンコフ補正因子は、他の２つの方式で求めたものと数値的に全く異なる。②DANKEプログラムにより求めたダンコフ補正因子は、ピッチが4.0cmの場合を除き、深井の式に従い求めた値との相対的な差は10％以内で一致している。

次に、参考文献4)には深井の式とモンテカルロ法の比較計算結果が記載されてる。DANKEプログラムを用いて計算したダンコフ補正因子とMAILコードに内蔵されている深井の式を用いて得られた値を併せて示したのが表２及び表３である。表２はペレット直径0.635cmからなる燃料棒の正方格子配列、表３はペレット直径0.635cm及び1.524cmからなる燃料棒の六角格子配列である。これらの表から以下のことが明らかになった。①DANKEプログラムにより求めたダンコフ補正因子は、参考文献4)に記載されているモンテカルロ法により得られたものと有効数字２桁の範囲でよく一致した。②MAILコードに内蔵の深井の式の値と文献値がよく一致しており、MAILコードの深井の式は正しくコード化されていると判断される。③モンテカルロ法と深井の式は近い値になっている。

以上の結果、DANKEプログラムが有効に働くと判断できる。

謝　　辞

第２章に記したBickley関数については原子炉システム研究室の土橋敬一郎室長に御教示頂きました。DANKEプログラムのJACSコードシステム中のMAIL3.0コードへの組込みは㈱ＣＲＣ総合研究所の塩田雅之氏の労を煩わしました。本報告書の作成を燃料サイクル安全工学部内藤俶孝次長に勧めて頂きました。また、同部燃料サイクル安全評価研究室野村靖室長には、本報告書を閲読の上、有益な御指摘を戴きました。これらの方々に謝意を表します。

参 考 文 献

1)松村哲夫・三村昌弘・内藤俶孝、「軽水炉用多群定数ライブラリーの開発」、電力中央研究所研究報告279065(1980).

2) H. J. Smith and A. Santamarina, "Analysis of the International Criticality Benchmark N°19 of a Realistic Fuel Dissolver," NEACRP-L-325 (1990).

3) Argonne National Laboratory, "Reactor Physics Constants," ANL-5800 (2nd Edition), (1963).

4) Y. Fukai,"New Analytical Formula for Dancoff Correction for Cylindrical Fuel Lattices," Nucl. Sci. Eng. 9, 372 (1961).

5) J. R. Knight, "SUPERDAN: Computer Programs for Calculating the Dancoff Factor of Spheres, Cylinders, and Slabs," ORNL/NUREG/CSD/TM-2 (1978).

6) J. P. McNeece, T. J. Trapp and J. K. Thompson, "MCDAN - A Monte Carlo Computer Code for Calculating the Dancoff Correction Factor for Spheres and Rods," PNL-3086 (1979).

7) Y. Komuro, Y. Naito, Y. Nomura and M. Shioda, "Improvement of JACS Code System," Proc. of the ICNC'91 International Conference on Nuclear Criticality Safety," IV-36 (1991).

8) 小室雄一・奥野浩・内藤俶孝・酒井友宏・塩田雅之・奥田泰久、「断面積セット作成プログラムMAIL3.0の使用手引書」、JAERI-M90-126(1990).

9) G. I. Bell and S. Glasstone, "Nuclear Reactor Theory," Van Nostrand Reinhold Company, New York (1970).

10)W. Rothenstein, "Collision Probabilities and Resonance Integrals for Lattices," Nucl. Sci. Eng. 7, 162 (1960).

11)M. Abramowitz and I. A. Sregun, "Handbook of Mathematical Functions," Dover Pub., New York (1965).

12)W. G. Bickley and J. Naylor, Phil. Mag. 20 Series F, 343 (1935).

13)L. M. Petrie and N. F. Cross, "KENO IV, An Improved Monte Carlo Criticality Program," ORNL-4938 (1975).

14)Y. Naito, S. Tsuruta, T. Matsumura and T. Ohuchi, "MGCL-PROCESSOR: A Computer Code System for Processing Multigroup Constants Library MGCL," JAERI-M 9396 (1981).

15)H. Reif, BNL-4869 (1960); Trans. Am. Nuc. Soc. 3, 170 (1960).

表１　ダンコフ補正係数の比較（その１）

ペレット半径0.625cm、水のΣ_{eff}= 1.487cm^{-1}、
クラッドなし、正方格子

ピッチ[cm]	DANKE	MCDAN	深井*1
1.5	0.333	0.411	0.356
1.956*2	0.105	0.147	0.115
2.5	0.0308	0.0471	0.0324
3.0	0.0107	0.0174	0.0107
4.0	0.00151	0.00287	0.00131

註*1：MAILコード[14]内蔵
　*2：最適燃料棒ピッチ

表２　ダンコフ補正因子の比較（その２）

ペレット半径0.125in、水のΣ_{eff}= 1.492cm^{-1}、
クラッドなし、正方格子

V_M/V_F	ピッチ[in]	DANKE	ﾓﾝﾃｶﾙﾛ法*1	深井*2	深井*3
0.273	0.25	0.797	0.7995	0.826	---
0.6	0.28	0.624	0.628	0.639	0.642
1.0	0.313	0.480	0.479	0.489	0.469
3.0	0.443	0.178	0.174	0.185	0.185

註*1：文献4)及び15)
　*2：MAILコード[14]内蔵
　*3：文献4)

表３　ダンコフ補正因子の比較（その３）

水のΣ_{eff}= 1.492cm^{-1}、クラッドなし、六角格子

ﾍﾟﾚｯﾄ半径 [in]	V_M/V_F	ピッチ [cm]	DANKE	ﾓﾝﾃｶﾙﾛ法*1	深井*2	深井*3
0.125	1.0	0.8552	0.472	0.474	0.466	0.466
0.125	3.0	1.209	0.174	0.179	0.172	0.172
0.3	1.0	2.052	0.212	0.217	0.233	0.235
0.3	3.0	2.903	0.0327	0.04	0.036	0.038

註*1：文献4)及び15)
　*2：MAILコード[14]内蔵
　*3：文献4)

```
=ANALYSIS/77= ** TREE STRUCTURE **  ENTRY POINT = MAIN

MAIN  -----READIN
        +--DANC  ----*FLTRN
                   +--START2----*FLTRN
                   +--INSEC2
                   +--HIT2
                   +--START3----*FLTRN
                   +--INSEC3-----HIT2
                   +--START ----*FLTRN
                   I          +--DRC    ----*FLTRN
                   +--INSECT
                   +--HIT
```

図1　DANKEプログラムにおけるプログラム間の樹状構造

付　　録　DANKEのソースリスト

DANKEプログラムのソースリストを次頁以降に示す。MAINプログラムの後にサブルーチンをアルファベット順に並べてある。

なお、ソースプログラムは日本原子力研究所東海研究所の大型計算機の磁気ディスク上のユーザーファイルJ3755.DANKE2.FORT77及びカートリッヂGA0587の208番目に同名のファイルとして収納されている。

```
      C     DANKE : A MONTE CARLO COMPUTER PROGRAM FOR CALCULATING THE DANCOFF
      C     CORRECTION FACTOR FOR SPHERES, RODS AND SLABS
      C     WRITTEN BY H. OKUNO AT JAERI
      C
      C     REVISED FROM MCDAN WRITTEN BY J.P.MCNEECE
      C     BATTELLE, PACIFIC NORTHWEST LABORATORY
      C
      C      NB = NUMBER OF NEUTRON BATCHES      (  ANY NUMBER, AT LEAST 3 )
      C     NPB = NUMBER OF NEUTRONS PER BATCH   (  SHOULD BE 3000 TO 5000 )
      C
)000001       DATA NB/5/, NPB/5000/
      C
      C     READ IN DATA TO DESCRIBE THE ARRAY SYSTEM
      C
)000002       CALL READIN(ITYPE)
      C
)000003       WRITE( 6, 104 )  NB, NPB
)000004   104 FORMAT(' NUMBER OF BATCHES = ', I3, ' NEUTRONS / BATCH =', I10 )
)000005       WRITE( 6, 105 )
)000006   105 FORMAT(' BATCH       C          AVGC                AVGX
     &NHITS' )
      C
)000007       SUM1 = 0.0
)000008       SUM2 = 0.0
      C
)000009       DO 1 I=1,NB
)000010         NHITS = 0
)000011         W     = 0.0
)000012         SUM3  = 0.0
)000013         STD   = 0.0
      C
)000014         DO 2 J=1,NPB
)000015           TOTX = 0.0
      C
)000016           CALL DANC ( ITYPE, CIJ, IT, TOTX )
      C
 000017           IF( IT .EQ. 1 ) NHITS = NHITS + 1
 000018           W = W + CIJ
 000019           SUM3 = SUM3 + TOTX
 000020     2   CONTINUE
      C
 000021         W = W / FLOAT( NPB )
 000022         AVGX = SUM3 / FLOAT( NPB )
 000023         SUM1 = SUM1 + W
 000024         SUM2 = SUM2 + W*W
 000025         AVGW = SUM1 / FLOAT( I )
 000026         IF( I .EQ. 1 ) GO TO 11
 000027          ARG = SUM2 / FLOAT( I-1 )
     &                              - AVGW * AVGW * FLOAT( I ) / FLOAT( I - 1 )
 000028         IF( ARG .LT. 0.0 ) ARG = 0.0
 000029         STD = SQRT( ARG )
      C
 000030    11   WRITE( 6 , 103 ) I , W , AVGW , STD , AVGX , NHITS
 000031   103   FORMAT(1H , I4 , 1P4E12.5 , I10 )
 000032     1 CONTINUE
      C
 000033       STOP
 000034       END
```

```
000001          SUBROUTINE DANC ( ITYPE, CIJ, IT, TOTX )
000002          COMMON /AA/ XMAX
000003          COMMON /PP/ PI,PI2,DPI,PI4I
000004          COMMON /RR/ R,P2,P2R3,SIG
000005          COMMON /DC/ DCA,DCB,DCG
000006          COMMON /DP/ XP,YP,ZP
000007          GO TO (100, 200, 300, 400, 700, 700) ITYPE
          C***  R = PLATE THICKNESS
000008      100 IT = 1
000009          AL = 2.0 * P2 - R
000010          COP1 = FLTRN ( DUM )
000011          IF ( COP1 .LT. 1.E-30 )  THEN
000012            CIJ = 0.
000013          ELSE
000014            TOTX = AL / SQRT( COP1 )
000015            CIJ =  EXP ( - SIG * TOTX )
000016          END IF
000017          RETURN
          C***  R    = ROD RADIUS  (CM)
          C     SIG  = MODERATOR MACRO X-SECTION (CM-1)
          C     XMAX = MAX NUMBER MFP N WILL TRACKED (MFP)
000018      200 CONTINUE
000019          CALL START2( X , Y , P2 , DPI , R )
000020          CALL INSEC2( TOTX, X, Y )
000021       20    XDIS = 0.0
000022             CALL HIT2( XDIS, IT, DCA, DCB, X, Y, 0., 0., R, PI2 )
          CCC    IF NO HIT THEN WHERE DOES N
000023             IF( IT .EQ. 0 ) CALL INSEC2( XDIS , X ,Y )
          CCC    CALCULATE TOTAL DISTANCE NEUTRON HAS TRAVELED
000024             TOTX = TOTX + XDIS
000025             IF( TOTX .LT. XMAX .AND. IT .EQ. 0 ) GO TO 20
000026          SIP2 = FLTRN ( DUM )
000027          COP2 = SQRT ( 1. - SIP2*SIP2 )
000028          IF ( COP2 .LT. 1.E-20 )  THEN
000029            CIJ = 0.
000030          ELSE
000031            TOTX = TOTX /  COP2
000032             CIJ = PI4I * COP2 * EXP( -SIG * TOTX )
000033          END IF
000034          RETURN
          C***
000035      300 CONTINUE
000036          TOTX = 0.
000037          CALL START3( X , Y , P2 , P2R3, DPI , R )
          C     WRITE(6,6000) DCA,DCB,XP,YP
          C     WRITE(6,6010) X,Y,XDIS,TOTX,IT
000038       30 CONTINUE
000039          CALL INSEC3( XDIS, IT, X, Y, PI2 )
000040             TOTX = TOTX + XDIS
          C     WRITE(6,6010) X,Y,XDIS,TOTX,IT
000041             IF( TOTX .GT. XMAX .OR. IT .EQ. 1 ) GO TO 31
000042             CALL HIT2( XDIS, IT, DCA, DCB, X, Y, 0., 0., R, PI2 )
          C     WRITE(6,6010) X,Y,XDIS,TOTX,IT
000043             IF( IT .EQ. 0 ) GO TO 30
000044          TOTX = TOTX + XDIS
000045       31 CONTINUE
          C     WRITE(6,6010) X,Y,XDIS,TOTX,IT
000046          SIP2 = FLTRN ( DUM )
000047          COP2 = SQRT ( 1. - SIP2*SIP2 )
000048          IF ( COP2 .LT. 1.E-20 )  THEN
```

```
000049          CIJ = 0.
000050        ELSE
000051          TOTX = TOTX /  COP2
000052           CIJ = PI4I * COP2 * EXP( -SIG * TOTX )
000053        END IF
000054        RETURN
      C6000 FORMAT(//1HO,'DCA, DCB, XP, YP = ', 4F10.4 )
      C6010 FORMAT(1HO,'X, Y, XDIS, TOTX, IT = ', 4F10.4, I5 )
      C***
000055    400 CONTINUE
      CCC   CALL ROUTINE FOR STARTING NEUTRON
000056        CALL START( X, Y, Z, P2, DPI, R )
000057        CALL INSECT(TOTX, X, Y, Z )
      C
000058     10    XDIS = 0.0
      CCC       SEE IF NEUTRON HITS SPHERE IN BOX
000059          CALL HIT( XDIS, IT, DCA,DCB,DCG, X,Y,Z, 0.,0.,0., R, PI2 )
      CCC       IF NO HIT THEN WHERE DOES N
000060          IF( IT .EQ. 0 ) CALL INSECT( XDIS , X ,Y , Z )
      CCC       CALCULATE TOTAL DISTANCE NEUTRON HAS TRAVELED
000061          TOTX = TOTX + XDIS
000062          IF( TOTX .LT. XMAX .AND. IT .EQ. 0 ) GO TO 10
000063        CIJ = EXP( -SIG * TOTX )
000064        RETURN
      C
000065    700 CIJ = -1.
000066        RETURN
000067        END
```

```
000001          SUBROUTINE DRC( DPI, COTO, SITO, COPO, SIPO, DCA, DCB, DCG )
000002          DIMENSION RMAT( 3, 3 ), DCS2( 3 ), DCS3( 3 )
000003          U1 = 2. * FLTRN( DUM ) -1.0
000004          T2 = 0.5 * ACOS( U1 )
000005          COT2 = COS( T2 )
000006          SIT2 = SIN( T2 )
000007          U2 = DPI * FLTRN( DUM )
        C
000008          DCS2(1) = SIT2 * COS( U2 )
000009          DCS2(2) = SIT2 * SIN( U2 )
000010          DCS2(3) = COT2
        C
000011          RMAT(1,1) = COTO * COPO
000012          RMAT(2,1) = -SIPO
000013          RMAT(3,1) = SITO * COPO
000014          RMAT(1,2) = COTO * SIPO
000015          RMAT(2,2) = COPO
000016          RMAT(3,2) = SITO * SIPO
000017          RMAT(1,3) = -SITO
000018          RMAT(2,3) = 0.
000019          RMAT(3,3) = COTO
        C
000020          DO 10 I = 1, 3
000021          SUM = 0.
000022          DO 20 J = 1, 3
000023          SUM = SUM + DCS2(J) * RMAT(J,I)
000024       20 CONTINUE
000025          DCS3(I) = SUM
000026       10 CONTINUE
        C
000027          DCA = DCS3(1)
000028          DCB = DCS3(2)
000029          DCG = DCS3(3)
        C
000030          RETURN
000031          END
```

```
      C SUBROUTINE FOR DETERMINING IF NEUTRON HITS SPHERE IN BOX
      C
000001      SUBROUTINE HIT( XDIS,IT, DCA,DCB,DCG, X1,Y1,Z1, X2,Y2,Z2, R, PI2)
      C
000002      IT = 0
      C
      C CALCULATE DIRECTION COSINES OF LINE FROM ( X1,Y1,Z1 ) TO CENTER OF
      C  SPHERE IN BOX.  ( THE CENTER IS AT ( X2,Y2,Z2) )
      C
000003        X = X1 - X2
000004        Y = Y1 - Y2
000005        Z = Z1 - Z2
000006        D = SQRT( X*X + Y*Y + Z*Z )
000007      DCA1 = -X /D
000008      DCB1 = -Y /D
000009      DCG1 = -Z /D
      C
      C CALCULATE COSINE OF ANGLE BETWEEN N PATH AND LINE TO CENTER OF SPHERE
      C
000010      A1 = DCA*DCA1 + DCB*DCB1 + DCG*DCG1
      C
      C IF A1 LE 0.0  THEN N SURELY CAN'T HIT SPHERE.
      C
000011      IF( A1 .LE. 0.0 ) RETURN
000012      ABTL = ACOS( A1 )
000013        A2 =  SIN( ABTL )
      C
      C CALCULATE LENGTH OF NORMAL TO N PATH AND CENTER OF SPHERE
      C
000014      A = D * A2
000015      IF( A .GT. R ) RETURN
000016      IT = 1
      C
      C CALCULATE DISTANCE TO WHERE N HITS SURFACE OF SPHERE
      C
000017      XDIS = ABS( D - R )
000018      IF( A2 .GT. 16.0E-65 )
     &                    XDIS = R * SIN( PI2 - ABTL - ACOS(A/R) ) / A2
000019      RETURN
000020      END
```

```
      C SUBROUTINE FOR DETERMINING IF NEUTRON HITS CIRCLE IN BOX
      C
000001      SUBROUTINE HIT2( XDIS, IT, DCA, DCB, X1, Y1, X2, Y2, R, PI2 )
      C
000002      IT = 0
      C
      C CALCULATE DIRECTION COSINES OF LINE FROM ( X1,Y1 ) TO CENTER OF SPHERE
      C  IN BOX.  ( THE CENTER IS AT ( X2,Y2 ) )
      C
000003         X = X1 - X2
000004         Y = Y1 - Y2
000005         D = SQRT( X*X + Y*Y )
000006      DCA1 = -X /D
000007      DCB1 = -Y /D
      C
      C CALCULATE COSINE OF ANGLE BETWEEN N PATH AND LINE TO CENTER OF SPHERE
      C
000008      A1 = DCA*DCA1 + DCB*DCB1
      C
      C IF A1 LE 0.0  THEN N SURELY CAN'T HIT SPHERE.
      C
000009      IF( A1 .LE. 0.0 ) RETURN
000010      ABTL = ACOS( A1 )
000011        A2 =  SIN( ABTL )
      C
      C CALCULATE LENGTH OF NORMAL TO N PATH AND CENTER OF SPHERE
      C
000012      A = D * A2
000013      IF( A .GT. R ) RETURN
000014      IT = 1
      C
      C CALCULATE DISTANCE TO WHERE N HITS SURFACE OF CIRCLE
      C
000015      XDIS = ABS( D - R )
000016      IF( A2 .GT. 16.0E-65 )
           &                    XDIS = R * SIN( PI2 - ABTL - ACOS(A/R) ) / A2
000017      RETURN
000018      END
```

```
      C SUBROUTINE FOR DETERMINING INTERSECTION OF N AND BOX BOUNDARY
      C
000001        SUBROUTINE INSECT( XDIS , X ,Y , Z )
      C
000002        COMMON /DC/ DCA,DCB,DCG
000003        COMMON /DP/ XP,YP,ZP
000004        DIMENSION D(3)
000005        D( 1 ) = 999.0
000006        D( 2 ) = 999.0
000007        D( 3 ) = 999.0
      C
      C CALCULATE DISTANCES N MUST TRAVEL TO INTERSECT THE 3 POSSIBLE PLANES.
      C
000008        IF( DCA .NE. 0.0 )  D(1) = ( XP - X ) / DCA
000009        IF( DCB .NE. 0.0 )  D(2) = ( YP - Y ) / DCB
000010        IF( DCG .NE. 0.0 )  D(3) = ( ZP - Z ) / DCG
      C
      C DETERMINE WHITCH IS SHORTEST
      C
000011        IS = 1
000012        IF( D(2) .LT. D(1)   ) IS=2
000013        IF( D(3) .LT. D(IS ) ) IS=3
000014        XDIS = D(IS)
      C
      C CALCULATE X,Y,Z CO-ORDINATES OF INTERSECTION POINT
      C
000015        X = X + D(IS) * DCA
000016        Y = Y + D(IS) * DCB
000017        Z = Z + D(IS) * DCG
      C
      C MOVE N POSITION SO IT REENTERS BOX CORRECTLY
      C
000018        GO TO ( 1 , 2 , 3 ) , IS
      C
000019      1 X = X - 2.0*XP
      C
      C CHECK FOR INTER SECTION AT CORNERS AND EDGES
      C
000020        IF( D(2) .EQ. D(1) )  Y = Y - 2.0*YP
000021        IF( D(3) .EQ. D(1) )  Z = Z - 2.0*ZP
000022        RETURN
      C
000023      2 Y = Y - 2.0*YP
000024        IF( D(3) .EQ. D(2) )  Z = Z - 2.0*ZP
000025        RETURN
      C
000026      3 Z = Z - 2.0*ZP
000027        RETURN
000028        END
```

```
      C SUBROUTINE FOR DETERMINING INTERSECTION OF N AND BOX BOUNDARY
      C
000001        SUBROUTINE INSEC2( XDIS , X ,Y )
      C
000002        COMMON /DC/ DCA,DCB,DCG
000003        COMMON /DP/ XP,YP,ZP
000004        DIMENSION D(2)
000005        D( 1 ) = 999.0
000006        D( 2 ) = 999.0
      C
      C CALCULATE DISTANCES N MUST TRAVEL TO INTERSECT THE 2 POSSIBLE LINES.
      C
000007        IF( DCA .NE. 0.0 )  D(1) = ( XP - X ) / DCA
000008        IF( DCB .NE. 0.0 )  D(2) = ( YP - Y ) / DCB
      C
      C DETERMINE WHITCH IS SHORTEST
      C
000009        IS = 1
000010        IF( D(2) .LT. D(1)   ) IS=2
000011        XDIS = D(IS)
      C
      C CALCULATE X,Y CO-ORDINATES OF INTERSECTION POINT
      C
000012        X = X + D(IS) * DCA
000013        Y = Y + D(IS) * DCB
      C
      C MOVE N POSITION SO IT REENTERS BOX CORRECTLY
      C
000014        GO TO ( 1 , 2 ) , IS
      C
000015      1 X = X - 2.0*XP
      C
      C CHECK FOR INTERSECTION AT CORNERS
      C
000016        IF( D(2) .EQ. D(1) )  Y = Y - 2.0*YP
000017        RETURN
      C
000018      2 Y = Y - 2.0*YP
000019        RETURN
000020        END
```

```
      C SUBROUTINE FOR DETERMINING INTERSECTION OF N AND BOX BOUNDARY
      C
000001        SUBROUTINE INSEC3( XDIS , IT , X ,Y, PI2 )
      C
000002        COMMON /DC/ DCA,DCB,DCG
000003        COMMON /DP/ XP,YP,ZP
000004        COMMON /RR/ R,P2,P2R3,SIG
000005        DIMENSION D(2)
000006        D( 1 ) = 999.0
000007        D( 2 ) = 999.0
      C
      C CALCULATE DISTANCES N MUST TRAVEL TO INTERSECT THE 2 POSSIBLE LINES.
      C
000008        IF( DCA .NE. 0.0 )  D(1) = ( XP - X ) / DCA
000009        IF( DCB .NE. 0.0 )  D(2) = ( YP - Y ) / DCB
      C
      C DETERMINE WHITCH IS SHORTEST
      C
000010        IS = 1
000011        IF( D(2) .LT. D(1)   ) IS=2
      C
      C CALCULATE WHETHER N SHOULD HIT A CIRCLE AT TWO CORNERS OR NOT
      C
000012        IF (IS .EQ. 1 ) THEN
000013           CALL HIT2( XDIS, IT, DCA, DCB, X, Y, XP,-YP, R, PI2 )
000014        ELSE
000015           CALL HIT2( XDIS, IT, DCA, DCB, X, Y,-XP, YP, R, PI2 )
000016        END IF
000017        IF ( IT .EQ. 1 ) RETURN
000018        CALL HIT2( XDIS, IT, DCA, DCB, X, Y, XP, YP, R, PI2 )
000019        IF ( IT .EQ. 1 ) RETURN
      C
000020        XDIS = D(IS)
      C
      C CALCULATE X,Y CO-ORDINATES OF INTERSECTION POINT
      C
000021        X = X + D(IS) * DCA
000022        Y = Y + D(IS) * DCB
      C
      C MOVE N POSITION SO IT REENTERS BOX CORRECTLY
      C
000023        GO TO ( 1 , 2 ) , IS
      C
000024      1 X = X - 2.0*XP
000025        RETURN
      C
000026      2 Y = Y - 2.0*YP
000027        RETURN
000028        END
```

```
000001          SUBROUTINE READIN ( ITYPE )
000002          COMMON /AA/ XMAX
000003          COMMON /PP/ PI, PI2, DPI, PI4I
000004          COMMON /RR/ R, P2, P2R3, SIG
          C
          C               R = SPHERE OR ROD RADIUS, OR PLATE THICKNESS (CM)
          C             SIG = MODERATOR MACRO X-SECTION (CM-1)
          C            XMAX = MAX NUMBER MFP N WILL TRACKED (MFP)
          C              VF = VOLUME FRACTION OF SPHERES        FOR ITYPE = 4
          C                   LATTICE PITCH FOR RODS      (CM)              2 AND 3
          C                   LATTICE PITCH FOR PLATES    (CM)              1
          C           ITYPE = 1 FOR ARRAY OF PLATES                ;
          C                 = 2 FOR SQUARE ARRAY OF RODS           ;
          C                 = 3 FOR TRIANGULAR ARRAY OF RODS       ;
          C                 = 4 FOR CUBIC ARRAYS OF SPHERES
          C
          C XMAX CAN BE ANY VALUE. SHOULD BE AT LEAST 10
          C
000005            XMAXO = 10.0
          C
000006            PI = 3.14159
000007            PI2 = PI / 2.0
000008            PI4I= 4.0/ PI
000009            DPI = 2.0 * PI
          C
          C READ LUMP GEOMETRY INDICATOR
          C
000010            READ( 5, 98 )  ITYPE
000011       98   FORMAT( I1 )
          C
000012            READ( 5 , 101 )  R , SIG , VF
000013      101   FORMAT( 3F10.0 )
          C
          C CALCULATE LATTICE PITCH
          C
000014            IF ( ITYPE . EQ. 4 ) THEN
000015            PITCH = R * ( 4.0 * PI / ( 3.0 * VF ) ) ** ( 1.0 / 3.0 )
000016            ELSE
000017            PITCH = VF
000018            END IF
000019            P2 = PITCH / 2.0
000020            P2R3 = SQRT(3.) * P2
          C
          C CONVERT XMAX TO CM
          C
000021             XMAX = XMAXO / SIG
000022             WRITE( 6 , 104 ) R, SIG , XMAX , VF , PITCH
000023      104   FORMAT( ' R = ' , 1PE12.5 , ' SIG = ', 1PE12.5 , ' XMAX = ' ,
                 & 1PE12.5 ,/
                 &          ' VF = ' , 1PE12.5 , ' PITCH = ' , 1PE12.5 )
          C
000024             RETURN
000025             END
```

```
      C SUBROUTINE FOR STARTING NEUTRON
      C
000001      SUBROUTINE START( X1, Y1, Z1, P2, DPI, R )
000002      COMMON /DC/ DCA,DCB,DCG
000003      COMMON /DP/ XP,YP,ZP
      C
      CCC   PICK A POINT ON SURFACE OF SPHERE
      C
000004      COTO = 2. * FLTRN(DUM) - 1.0
000005      SITO = SQRT((1.-COTO)*(1.+COTO))
000006      PO   = FLTRN(DUM) * DPI
000007      COPO = COS(PO)
000008      SIPO = SIN(PO)
000009      T1 =  R * SITO
000010      X1 = T1 * COPO
000011      Y1 = T1 * SIPO
000012      Z1 =  R * COTO
      C
      C**** CALCULATE DIRECTION COSINES ( DCA, DCB, DCG )
      C
000013      CALL DRC( DPI, COTO, SITO, COPO, SIPO, DCA, DCB, DCG )
      C
      C**** 3 PLANE CAN N HIT
      C
000014      XP = DCA * P2 / ABS( DCA )
000015      YP = DCB * P2 / ABS( DCB )
000016      ZP = DCG * P2 / ABS( DCG )
      C
000017      RETURN
000018      END
```

```
      C SUBROUTINE FOR STARTING NEUTRON
      C
000001      SUBROUTINE START2( X1 , Y1 , P2 , DPI , R )
000002      COMMON /DC/ DCA,DCB,DCG
000003      COMMON /DP/ XP,YP,ZP
      C
      C PICK A POINT ON SURFACE OF ROD
      C
000004      TO = FLTRN(DUM) * DPI
000005      X1 =  R * COS( TO )
000006      Y1 =  R * SIN( TO )
      C
000007      U2 = 2. * FLTRN ( DUM ) - 1.
000008      T2 = ASIN ( U2 )
000009      DCA = COS( TO + T2 )
000010      DCB = SIN( TO + T2 )
      C
      C       2 LINE CAN N HIT
      C
000011      XP = DCA * P2 / ABS( DCA )
000012      YP = DCB * P2 / ABS( DCB )
      C
000013      RETURN
000014      END
```

```
      C SUBROUTINE FOR STARTING NEUTRON
      C
000001      SUBROUTINE START3( X1 , Y1 , P2 , P2R3 , DPI , R )
000002      COMMON /DC/ DCA,DCB,DCG
000003      COMMON /DP/ XP,YP,ZP
      C
      C PICK A POINT ON SURFACE OF ROD
      C
000004      TO = FLTRN(DUM) * DPI
000005      X1 =  R * COS( TO )
000006      Y1 =  R * SIN( TO )
      C
000007      U2 = 2. * FLTRN ( DUM ) - 1.
000008      T2 = ASIN ( U2 )
000009      DCA = COS( TO + T2 )
000010      DCB = SIN( TO + T2 )
      C
      C       2 LINE CAN N HIT
      C
000011      XP = DCA * P2 / ABS( DCA )
000012      YP = DCB * P2R3 / ABS( DCB )
      C
000013      RETURN
000014      END
```