รายการอ้างอิง

[1] ภุชวดี เขียวรอด. การพัฒนาโปรแกรมคำนวณการจัดการแกนเชื้อเพลิงนิวเคลียร์แบบ 2 มิติและหนึ่งกลุ่มพลังงาน. วิทยานิพนธ์ ปริญญามหาบัณฑิต ภาควิชานิวเคลียร์เทคโนโลยี วิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัยม, 2544.

[2] ไพศาล เติมสินวาณิช. การคำนวณเชิงตัวเลขเพื่อออกแบบเครื่องปฏิกรณ์นิวเคลียร์ที่สภาวะใต้วิกฤต. วิทยานิพนธ์ ปริญญามหาบัณฑิต ภาควิชานิวเคลียร์เทคโนโลยี วิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย, 2544.

[3] ยุทธพงศ์ บุษมงคล. แผนการจัดการเชื้อเพลิงในแกนเครื่องปฏิกรณ์ ปปว-1/1. วิทยานิพนธ์ ปริญญามหาบัณฑิต ภาควิชานิวเคลียร์เทคโนโลยี วิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย, 2529.

[4] รังษี พรเจริญ. การประเมินค่าคงที่ของกลุ่มนิวตรอนสำหรับสมการการแพร่หลายกลุ่มพลังงานโดยวิธียุบกลุ่ม. วิทยานิพนธ์ ปริญญามหาบัณฑิต ภาควิชานิวเคลียร์เทคโนโลยี วิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย, 2547.

[5] Duderstadt, J. J. and Hamilton, L. J. Nuclear Reactor Analysis. 2nd edition. New York: John Wiley & Sons, 1974.

[6] Lamarsh, J. R. Introduction to Nuclear Engineering. 2nd. New York : Addison-Wesley Publishing Company, 1975.

[7] สัญชัย นิลสุวรรณโมษิต. วิศวกรรมแกนปฏิกรณ์เบื้องต้น. เอกสารประกอบการสอนรายวิชา 2111642 ภาควิชานิวเคลียร์เทคโนโลยี วิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย, 2544 (เอกสารไม่ตีพิมพ์)

[8] Almenas, K. and Lee, R. Nuclear Engineering An Introduction. College Park Campus Department of Chemical and Nuclear Engineering The University of Maryland United State of America. Springer-Verlag Berlin and Heidelberg GmbH & Co. K, 1992.

[9] Glasstone, S. and Sesonske, A. Nuclear Reactor Engineering (Reactor Systems Engineering. 4th edition, volume one, United States of America: Chapman & Hall, Inc., 1994.

[10] ปราโมทย์ เดชะอำไพ. ระเบียบวิธีเชิงตัวเลขในงานวิศวกรรม. พิมพ์ครั้งที่ 5. พระนคร: สำนักพิมพ์แห่งจุฬาลงกรณ์มหาวิทยาลัย, 2541.

[11] Chapra, S. C. and Raymond, P. C. Numerical Methods for Engineers. 3rd edition, Singapore: McGraw-Hill Companies, 1998.

[12] OKUMURA, K. et al. SRAC (Ver. 2002) ;The comprehensive neutronics calculation code system. Vol 1. Japan : JAEA publication, 2002.

ภาคผนวก

ภาคผนวก ก

ข้อมูลของแกนปฏิกรณ์นิวเคลียร์แบบ PWR[5]

ตารางที่ ก.1 ข้อมูลของแกนปฏิกรณ์นิวเคลียร์แบบ PWR

General Data	PWR (W)	PWR (B&W)	PWR (CE)
Thermal output (MWt)	3411	3600	3800
Electrical output (MWe)	1150	1200	1300
Efficiency	33.7	33.3	34.2
Fuel type	UO_2	UO_2	UO_2
Coolant	H_2O	H_2O	H_2O
Structural material	Zircaloy	Zircaloy	Zircaloy
Moderator	H_2O	H_2O	H_2O
Core Data			
Active height (cm)	366	363	381
Equivalent active diameter (cm)	337	352	363
Fuel weight (kg)	90,200	94,900	103,000
Fuel Assemblies Type	square bundles	square bundles	square bundles
Number of Assemblies	193	205	241
fuel-element array	17x17	17x17	16x16
Assembly dimension (cm)	21.4x21.4	21.7x21.7	20.3x20.3
Assembly pitch (cm)	21.5	21.8	20.7
Number of fuel elements / assembly	264	264	236
Fuel Element Data Type	clad rod	clad rod	clad rod
fuel-element pitch (cm)	1.25	1.27	1.28
Fuel element O.D. (cm)	0.94	0.96	0.97
Clad thickness (cm)	0.0572	0.0597	0.0635
Fuel-pellet diameter (cm)	0.819	0.823	0.825
Pellet-clad gap (cm)	0.0082	0.01	0.0089
Fuel enrichment	2.1 / 2.6 / 3.1	2.91	1.9 / 2.4 / 2.9

ภาคผนวก ข

SRAC COMPUTER CODE[12]

The SRAC is a code system applicable to neutronics analysis of a variety of reactor types. Since the publication of the second version of the users manual (JAERI-1305) in 1986 for the SRAC system, a number of additions and modifications to the functions and the library data have been made to establish a comprehensive neutronics code system. The current system includes major neutron data libraries (JENDL-3.3, JENDL-3.2, ENDF/B-VI, JEF-2.2, etc.), and integrates five elementary codes for neutron transport and diffusion calculation, they are, PIJ based on the collision probability method applicable to 16 kind of lattice models, SN transport codes ANISN(1D) and TWOTRAN(2D), diffusion codes TUD(1D) and CITATION(multi-D).

The system also includes two auxiliary codes: ASMBURN for fuel assembly burn-up calculation and COREBN for multi-dimensional core burn-up calculation.

This report is the third revision of the users manual which consists of the following three volumes:

Vol.1 General Description and Input Instruction

Vol.2 Mathematics of SRAC

Vol.3 ASMBURN and COREBN ; auxiliary burn-up codes of SRAC

ภาคผนวก ค

ตัวอย่างอินพุตไฟล์

ตัวอย่างการกำหนดค่าให้ไฟล์อินพุตโดยใช้โปรแกรมสร้างเท็กซ์ไฟล์ธรรมดา

```
#Example of X,Y,Z loading core
# <CARD A>    DIMENSION             nx  ny  nz  MATCOUNT
  366.0  366.0  370.0     15  15  1     193
# <CARD B>    (INNER : Flux) ERROR CRITERIA      (OUTTER : KERR) ERROR CRITERIA
              1.0E-3                      1.0E-4
# <CARD C>    THERMAL POWER (MW)
  3411.00
# <CARD D>    ENERGY GROUP
     2
# <CARD E>    PATTERN  PLANE 1
  0  0  0  0  3  3  3  3  3  3  3  0  0  0  0
  0  0  3  3  3  3  3  3  3  3  3  3  3  0  0
  0  3  3  3  3  3  3  3  3  3  3  3  3  3  0
  0  3  3  3  3  3  3  3  3  3  3  3  3  3  0
  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3
  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3
  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3
  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3
  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3
  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3
  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3
  0  3  3  3  3  3  3  3  3  3  3  3  3  3  0
  0  3  3  3  3  3  3  3  3  3  3  3  3  3  0
  0  0  3  3  3  3  3  3  3  3  3  3  3  0  0
  0  0  0  0  3  3  3  3  3  3  3  0  0  0  0
# <CARD F>    MACROSCOPIC CROSS SECTION LIBRARY
   D:\Thesis\DENUMG-FINAL-2009\SAMPLES\macref-2g.xs
# <CARD G>    BURNUP OPTION (0 = NO CALCULATE , 1 = CALCULATE)
   1
# <CARD H>    SHIM OPTION (0 = NO SHIM , 1 = SHIM)
   1
# <CARD I>    DURATION TIME (DAYS)
   40   20
# <CARD J>    METRIC TONS OF URANIUM LOADED IN CORE
   90.20
# <CARD K>      MWD/MTU
#  1     PLANE 1
   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000
   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000
   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000
   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000
   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000
   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000
   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000
   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000
   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000
   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000
   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000
   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000
   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000
   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000
   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000
   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000
   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000
   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000
   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000
   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000
   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000
   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000
   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000
   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000
   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000
   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000
   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000
   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000
   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000
   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000
   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000
   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000
   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000
   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000
   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000
   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000
   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000
   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000
   0.0000E+000   0.0000E+000   0.0000E+000
#   <CARD L>     LOADING OPTION (0 = LAST BATCH , N = RELOAD BATCH)
    1     20
#   <CARD M>     FLUX PRINT OPTION (0 = NOT PRINT, 1 = PRINT)
    0
# <CARD N>
# PREVLOAD  NEXTLOAD  MATNLOAD
```

0	1	3
0	2	3
0	3	3
0	4	3
0	5	3
0	6	3
0	7	3
0	8	3
0	9	3
0	10	3
0	11	3
0	15	3
0	16	3
0	17	3
0	18	3
0	19	3
0	20	3
0	30	3
0	31	3
0	32	3
0	44	3
0	45	3
0	46	3
0	58	3
0	59	3
0	60	3
0	61	3
0	73	3
0	74	3
0	75	3
0	89	3
0	90	3
0	104	3
0	105	3
0	119	3
0	120	3
0	121	3
0	133	3
0	134	3
0	135	3
0	136	3
0	148	3
0	149	3
0	150	3
0	162	3
0	163	3
0	164	3
0	174	3
0	175	3
0	176	3
0	177	3
0	178	3
0	179	3
0	183	3
0	184	3
0	185	3
0	186	3
0	187	3
0	188	3
0	189	3
0	190	3
0	191	3
0	192	3
0	193	3
1	21	3
2	91	3
3	47	3
4	122	3
5	57	3
6	181	3
7	29	3
8	12	3
9	23	3
10	92	3
11	36	3
12	37	3
13	38	3
14	39	3
15	40	3
16	169	3
17	27	3
18	14	3
19	76	3
20	24	3
21	49	3
22	50	3
23	51	3
24	80	3
25	68	3
26	84	3
27	53	3
28	54	3
29	55	3
30	26	3
31	88	3
32	77	3
33	63	3
34	79	3
35	95	3

```
    36          96          3
    40          112         3
    41          127         3
    42          85          3
    43          71          3
    44          87          3
    45          33          3
    46          48          3
    47          64          3
    48          81          3
    56          83          3
    57          70          3
    58          56          3
    59          43          3
    60          22          3
    61          35          3
    62          94          3
    63          97          3
    72          142         3
    73          41          3
    74          28          3
    75          34          3
    76          78          3
    77          65          3
    87          69          3
    88          86          3
    89          42          3
    90          62          3
    91          93          3
    92          66          3
    102         128         3
    103         101         3
    104         132         3
    105         152         3
    106         108         3
    107         52          3
    117         129         3
    118         116         3
    119         160         3
    120         166         3
    121         153         3
    122         125         3
    132         100         3
    133         159         3
    134         172         3
    135         151         3
    136         138         3
    137         124         3
    138         111         3
    146         113         3
    147         130         3
    148         146         3
    149         161         3
    150         107         3
    151         123         3
    152         109         3
    153         67          3
    154         141         3
    158         98          3
    159         99          3
    160         115         3
    161         131         3
    162         117         3
    163         106         3
    164         168         3
    165         139         3
    166         140         3
    167         82          3
    168         110         3
    169         126         3
    170         114         3
    171         143         3
    172         144         3
    173         145         3
    174         170         3
    175         118         3
    176         180         3
    177         167         3
    178         25          3
    179         154         3
    180         155         3
    181         156         3
    182         157         3
    183         158         3
    184         102         3
    185         171         3
    186         182         3
    187         165         3
    188         13          3
    189         137         3
    190         72          3
    191         147         3
    192         103         3
    193         173         3

END
```

ภาคผนวก ง

ตัวอย่างไฟล์ค่าคงที่กลุ่ม

ตัวอย่างไฟล์อินพุตสำหรับค่าคงที่ภาคตัดขวางมหภาค ในตัวอย่างนี้จะพิจารณาจำนวนวัสดุที่ต่างกันทั้งหมด 3 ชนิด คือเชื้อเพลิงเสริมสมรรถนะ 2.1% เป็นวัสดุหมายเลข 1 เชื้อเพลิงเสริมสมรรถนะ 2.6% เป็นวัดสุหมายเลข 2 และ เชื้อเพลิงเสริมสมรรถนะ 3.2% เป็นวัสดุหมายเลข 3 เป็นค่าคงที่ภาคตัดขวางมหภาค 2 กลุ่มพลังงาน และมีจำนวนขั้นการเผาผลาญเชื้อเพลิง 35 ขั้น

```
#     <CARD XS1> MATNUM       BUSTEP
              3            35
#    <CARD XS2>    STEP
       500.00       600.00       700.00       800.00       900.00
      1000.00      1200.00      1400.00      1600.00      1800.00
      2000.00      2200.00      2400.00      2500.00      5000.00
      7500.00     10000.00     12500.00     15000.00     17500.00
     20000.00     25000.00     30000.00     35000.00     40000.00
     45000.00     50000.00     55000.00     60000.00     70000.00
     80000.00     90000.00    100000.00    120000.00
#    <CARD XS3>   MATID      MTUSTEP
              1            0
# <CARD XS4>
# LIBMACvEf   LIBMACEf    LIBMACEc    LIBMACEa    LIBCHI      LIBDIFFCO1  LIBDIFFCO2  LIBMACEt     LIBMACAc
  5.2125e-003 2.0094e-003 6.9531e-003 8.9624e-003 1.0000e+000 1.4526e+000 1.4544e+000 2.3065e-001 4.1115e-002
  8.9752e-002 3.6839e-002 2.0526e-002 5.7365e-002 0.0000e+000 4.5132e-001 4.5635e-001 7.5623e-001 3.0141e+000
  2.0522e-001 1.6511e-002
  1.2757e-004 6.9874e-001
        :
#    <CARD XS3>      MATID      MTUSTEP
              1            34
# <CARD XS4>
# LIBMACvEf   LIBMACEf    LIBMACEc    LIBMACEa    LIBCHI      LIBDIFFCO1  LIBDIFFCO2  LIBMACEt    LIBMACAc
  5.1965e-003 1.9999e-003 7.0328e-003 9.0327e-003 1.0000e+000 1.4500e+000 1.4519e+000 2.3106e-001 4.1470e-002
  8.9411e-002 3.6479e-002 2.3795e-002 6.0274e-002 0.0000e+000 4.5259e-001 4.5727e-001 7.5345e-001 2.9828e+000
  2.0538e-001 1.6674e-002
  1.3324e-004 6.9305e-001
        :
#   <CARD XS3>       MATID      MTUSTEP
              2            1
# <CARD XS4>
# LIBMACvEf   LIBMACEf    LIBMACEc    LIBMACEa    LIBCHI      LIBDIFFCO1  LIBDIFFCO2  LIBMACEt    LIBMACAc
  5.1965e-003 1.9999e-003 7.0328e-003 9.0327e-003 1.0000e+000 1.4500e+000 1.4519e+000 2.3106e-001 4.1470e-002
  8.9411e-002 3.6479e-002 2.3795e-002 6.0274e-002 0.0000e+000 4.5259e-001 4.5727e-001 7.5345e-001 2.9828e+000
  2.0538e-001 1.6674e-002
  1.3324e-004 6.9305e-001
        :
#   <CARD XS3>       MATID      MTUSTEP
              2            34
# <CARD XS4>
# LIBMACvEf   LIBMACEf    LIBMACEc    LIBMACEa    LIBCHI      LIBDIFFCO1  LIBDIFFCO2  LIBMACEt    LIBMACAc
  5.1950e-003 1.9984e-003 7.0388e-003 9.0372e-003 1.0000e+000 1.4500e+000 1.4519e+000 2.3106e-001 4.1475e-002
  8.9728e-002 3.6553e-002 2.4032e-002 6.0585e-002 0.0000e+000 4.5261e-001 4.5727e-001 7.5337e-001 2.9805e+000
  2.0538e-001 1.6676e-002
  1.3389e-004 6.9265e-001
        :
#  <CARD XS3>            MATID      MTUSTEP
              3            1
# <CARD XS4>
# LIBMACvEf   LIBMACEf    LIBMACEc    LIBMACEa    LIBCHI      LIBDIFFCO1  LIBDIFFCO2  LIBMACEt    LIBMACAc
  5.1965e-003 1.9999e-003 7.0328e-003 9.0327e-003 1.0000e+000 1.4500e+000 1.4519e+000 2.3106e-001 4.1470e-002
  8.9411e-002 3.6479e-002 2.3795e-002 6.0274e-002 0.0000e+000 4.5259e-001 4.5727e-001 7.5345e-001 2.9828e+000
  2.0538e-001 1.6674e-002
  1.3324e-004 6.9305e-001
        :
# <CARD XS3>            MATID      MTUSTEP
              3            34
# <CARD XS4>
# LIBMACvEf   LIBMACEf    LIBMACEc    LIBMACEa    LIBCHI      LIBDIFFCO1  LIBDIFFCO2  LIBMACEt    LIBMACAc
  5.1950e-003 1.9984e-003 7.0388e-003 9.0372e-003 1.0000e+000 1.4500e+000 1.4519e+000 2.3106e-001 4.1475e-002
  8.9728e-002 3.6553e-002 2.4032e-002 6.0585e-002 0.0000e+000 4.5261e-001 4.5727e-001 7.5337e-001 2.9805e+000
  2.0538e-001 1.6676e-002
  1.3389e-004 6.9265e-001

END
```

ภาคผนวก จ

ตัวอย่างเอาต์พุตไฟล์

```
<0> -------------   Input file   -------------

#Example of X,Y,Z loading core
#      DIMENSION           nx  ny  nz  MATCOUNT
  340.0  340.0  380.0      7   7   7      37
# (INNER : Flux) ERROR CRITERIA      (OUTTER : K) ERROR CRITERIA
                1.0E-3                      1.0E-4
# THERMAL POWER (MW)
  3411.00
# ENERGY GROUP
     2
# PATTERN  PLANE 1
  0  0  1  1  1  0  0
  0  1  1  1  1  1  0
  1  1  1  1  1  1  1
  1  1  1  1  1  1  1
  1  1  1  1  1  1  1
  0  1  1  1  1  1  0
  0  0  1  1  1  0  0
# PATTERN  PLANE 2
  0  0  1  1  1  0  0
  0  1  1  1  1  1  0
  1  1  1  1  1  1  1
  1  1  1  1  1  1  1
  1  1  1  1  1  1  1
  0  1  1  1  1  1  0
  0  0  1  1  1  0  0
# PATTERN  PLANE 3
  0  0  1  1  1  0  0
  0  1  1  1  1  1  0
  1  1  1  1  1  1  1
  1  1  1  1  1  1  1
  1  1  1  1  1  1  1
  0  1  1  1  1  1  0
  0  0  1  1  1  0  0
# PATTERN  PLANE 4
  0  0  1  1  1  0  0
  0  1  1  1  1  1  0
  1  1  1  1  1  1  1
  1  1  1  1  1  1  1
  1  1  1  1  1  1  1
  0  1  1  1  1  1  0
  0  0  1  1  1  0  0
# PATTERN  PLANE 5
  0  0  1  1  1  0  0
  0  1  1  1  1  1  0
  1  1  1  1  1  1  1
  1  1  1  1  1  1  1
  1  1  1  1  1  1  1
  0  1  1  1  1  1  0
  0  0  1  1  1  0  0
# PATTERN  PLANE 6
  0  0  1  1  1  0  0
  0  1  1  1  1  1  0
  1  1  1  1  1  1  1
  1  1  1  1  1  1  1
  1  1  1  1  1  1  1
  0  1  1  1  1  1  0
  0  0  1  1  1  0  0
# PATTERN  PLANE 7
  0  0  1  1  1  0  0
  0  1  1  1  1  1  0
  1  1  1  1  1  1  1
  1  1  1  1  1  1  1
  1  1  1  1  1  1  1
  0  1  1  1  1  1  0
  0  0  1  1  1  0  0
# MACROSCOPIC CROSS SECTION LIBRARY
   D:\Thesis\DENUMG-FINAL-2009\macref-2g.xs
# BURNUP OPTION (0 = NO CALCULATE , 1 = CALCULATE)
   0
# SHIM OPTION (0 = NO SHIM , 1 = SHIM)
   0
# DURATION TIME (DAYS)
   40   20
# METRIC TONS OF URANIUM LOADED IN CORE
   90.20
#    MWD/MTU
#  1      PLANE 1
   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000
   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000
   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000
   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000
```

```
   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000
   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000
   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000
   0.0000E+000   0.0000E+000
#  1     PLANE 2
   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000
   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000
   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000
   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000
   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000
   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000
   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000
   0.0000E+000   0.0000E+000
#  1     PLANE 3
   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000
   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000
   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000
   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000
   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000
   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000
   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000
   0.0000E+000   0.0000E+000
#  1     PLANE 4
   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000
   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000
   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000
   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000
   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000
   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000
   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000
   0.0000E+000   0.0000E+000
#  1     PLANE 5
   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000
   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000
   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000
   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000
   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000
   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000
   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000
   0.0000E+000   0.0000E+000
#  1     PLANE 6
   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000
   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000
   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000
   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000
   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000
   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000
   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000
   0.0000E+000   0.0000E+000
#  1     PLANE 7
   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000
   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000
   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000
   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000
   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000
   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000
   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000   0.0000E+000
   0.0000E+000   0.0000E+000
#  LOADING OPTION (0 = LAST BATCH , N = RELOAD BATCH)
   0     1

   END

<0> ------------------------------------------

Calculation Started on 13:57:26


           0           0 5.03253e-001 6.51519e-001 5.03256e-001           0           0
           0 6.16816e-001 1.08491e+000 1.24542e+000 1.08492e+000 6.16823e-001           0
 5.03253e-001 1.08491e+000 1.49142e+000 1.62938e+000 1.49143e+000 1.08492e+000 5.03259e-001
 6.51519e-001 1.24542e+000 1.62938e+000 1.75630e+000 1.62939e+000 1.24543e+000 6.51525e-001
 5.03256e-001 1.08492e+000 1.49143e+000 1.62939e+000 1.49143e+000 1.08493e+000 5.03259e-001
           0 6.16823e-001 1.08492e+000 1.24543e+000 1.08493e+000 6.16824e-001           0
           0           0 5.03259e-001 6.51525e-001 5.03259e-001           0           0

LOAD NEW BATCH ,     0
keff [  0] =  1.1964 , SHIM[  0] =   0.0000e+000
BURNSUM =   1.7686e+004
Calculation Finished on 13:57:26
```

ภาคผนวก ฉ

SOURCE CODE

chemical shim.c

```
#include "variables.h"
float chemical_shim (float *MACEr,float *MACEa,float *MACEt,float keff1,float chemical_0,int
sign,int matcount, int nz){
   int i,j,k,group;
   int locateij,eslocate;
   int mat;
   float maximumEr;
   float weight[MXGROUP];
   float EaTemp,EtTemp,EsTemp,ExcessTemp;
   maximumEr = 0;
   group = energy_group-1;
   for (i = 0 ; i < matcount*nz ; i++){
       locateij = ij2t (i,group,matcount*nz,energy_group);
       eslocate = ijk2t (group,group,i,energy_group,energy_group,matcount*nz);
       MACEt[locateij] = MACEt[locateij] + (0.01*MACEa[locateij]);
       MACEr[locateij] = MACEt[locateij] - MACEs[eslocate];
       L[locateij] = MACEr[locateij]/DIFFCO1[locateij];

   }
   return (0.01*MACEa[locateij]);
}
```

converts.c

```
#include "variables.h"
int ijk2t (int i,int j,int k,int nx,int ny,int nz){
    int ii,jj,kk;
    int counter;
    counter = 0;
    for (kk = 0; kk < nz; kk++)
    for (ii = 0; ii < nx; ii++)
                for (jj = 0; jj < ny; jj++){
         if ((ii == i)&&(jj == j)&&(kk == k)){
          return (counter);
         }
        counter++;
        }
    return (9999);
}

int ij2t (int i,int j,int nx, int ny){

  if (i >= nx) return (9999999);
  if (j >= ny) return (9999999);
  return (i*ny+j);
}

float integral (float *XOLD , int matcount , int energy_group , int nz , float dx , float dy , float
dz){

  int i,j,k;
  float int_x ;
  int_x = 0;
  for (i = 0 ; i < matcount*nz*energy_group ; i++){
       int_x = int_x + XOLD[i]*dx*dy*dz;
  }
  return (int_x);
}

int printflux (int *LoadXYZ,float *XNEW,int energy_group,int nx,int ny,int nz,FILE *fout1){
  int group,i,j,k,l,m;
  int locateij,locateijk;
  for (group = 0 ; group < energy_group ; group++){
      fprintf (fout1,"\nGroup *** %3d ***\n\n",group);
      for (k = 0 ; k < nz ; k++){
        for (i = 0 ; i < nx ; i++){
          for (j = 0 ; j < ny ; j++){
            locateijk = ijk2t(i,j,k,nx,ny,nz);
            locateij = MATRIXTC[locateijk] + k*matcount + group*matcount*nz;
            if (LoadXYZ [locateijk] > 0){
                fprintf (fout1,"%15.5e",XNEW[locateij]);
            }
            else fprintf (fout1,"%15d",0);
          }
                fprintf (fout1,"\n");
```

```
        }
                fprintf (fout1,"\n");
      }
  }

}

int peaking (int *LoadXYZ,float *RPEAK,int nx,int ny,FILE *fout1){
  int group,i,j,k,l,m;
  int locateij,locateijk;
  for (i = 0 ; i < nx ; i++){
    for (j = 0 ; j < ny ; j++){
      locateij = ij2t(i,j,nx,ny);
      if (LoadXYZ [locateij] > 0){
          fprintf (fout1,"%13.5e",RPEAK[MATRIXTC[locateij]]);
      }
      else fprintf (fout1,"%13d",0);
    }
    fprintf (fout1,"\n");
  }

}
float fluxavg (float *XNEW, float *XAVG, int matcount, int energy_group, int nz, int group, float
dx ,float dy ,float dz,FILE *favg1){

   int i,j,k;
   int locateij;
   float sum_flux;
   float fmax ;

   fmax = 0.0;
   sum_flux = 0.0;

   for (i = 0 ; i < matcount*nz ; i++){
     locateij = ij2t (group,i,energy_group,matcount*nz);
     if (fmax < XNEW[locateij]) fmax = XNEW[locateij];
     sum_flux = sum_flux + XNEW[locateij];
     }
   sum_flux = sum_flux/(matcount*nz);
   printf ("GROUP = %5d , \nAVG = %15.4e , \nMAX = %15.4e , \nPEAKING
= %15.4e\n",group,sum_flux,fmax,fmax/sum_flux);

   return (sum_flux);
}
int updateMWD (float BurnValue,float *PZONE,float *MWDMTU,int matcount,int nz,int BUSTEP){

    int i,j,k,group;
    int locateij;
    int locateijk;
    float temp;
    temp = BurnValue/(u_weight/(matcount*nz));
    for (i = 0 ; i < matcount*nz ; i++){
      if ((PZONE[i]*temp) > MWDSTEP[BUSTEP-1]) {
         printf ("*****  DATA OUT OF BOUND  ***** %15.4e , %15.4e\n",PZONE[i]*temp , MWDSTEP[BUSTEP-
1]);
         return (10);
         } else
      MWDMTU[i] = MWDMTU[i]+(PZONE[i]* temp);
    }
  return (0);
}

float calculateMWD (float *BURNMWD,float *MWDMTU,int matcount,int nz,FILE *CALMWD){

    int i,j,k,group;
    int locateij;
    float MWDTEMP;

    for (i = 0 ; i < matcount ; i++){
      MWDTEMP = 0.0;
      for (k = 0 ; k < nz ; k++){
        locateij = ij2t (k,i,nz,matcount);
        MWDTEMP = MWDTEMP + MWDMTU[locateij];
      }
      BURNMWD[i] = MWDTEMP/nz;
  }
    MWDTEMP = 0.0;
    return (MWDTEMP/matcount);

}
```

Error.c

```
void ERRORINDICATE (int ERRORMESSAGE)
{
     switch (ERRORMESSAGE){
           case  1: printf ("\nCOMMAND:\\> DENUMG <input file> <output file>\n");
```

```
                    printf ("\nERROR (1) ****** no input ******\n\n");
                    break;
            case  2: printf ("\nCOMMAND:\\> DENUMG <input file> <output file>\n");
                    printf ("\nERROR (2) ****** no output file ******\n\n");
                    break;
            case  3: printf ("\nCOMMAND:\\> DENUMG <input file> <output file>\n");
                    printf ("\nERROR (3) ****** too many parameter ******\n\n");
                    break;
            case  4: printf ("\nERROR (4) ****** Can not open input file ******\n\n");
                    break;
            case  5: printf ("\nERROR (5) ****** Can not create output file ******\n\n");
                    break;
            case  6: printf ("\nERROR (6) ****** Can not create input temporary file ******\n\n");
                    break;
            case  7: printf ("\nERROR (7) ****** Can not create FP002 file ******\n\n");
                    break;
            case  8: printf ("\nERROR (8) ****** Matrix Parameter too large ******\n\n");
                    break;
            case  9: printf ("\nERROR (9) ****** Singular Matrix was found ******\n\n");
                    break;
            case 10: printf ("\nERROR (10) ****** K-Calculation was not converged ******\n\n");
                    break;
            case 11: printf ("\nERROR (11) ****** Can not create cross section temporary file
******\n\n");
                    break;
            case 12: printf ("\nERROR (12) ****** Can not create mwd.o file ******\n\n");
                    break;
            case 13: printf ("\nERROR (13) ****** Can not find cross section file ******\n\n");
                    break;
            case 14: printf ("\nERROR (14) ****** Can not create flux.o file ******\n\n");
                    break;
            case 15: printf ("\nERROR (15) ****** Data out of bound in cross section (MWD/MTU
exceeded) ******\n\n");
                    break;
                   }
}
```

Filemanage.c

```
#include "variables.h"
extern FILE *filein,*fileout;

int filemanage (int argc, char *argv[]){
    if (argc == 1){
            return (1);
       }
    if (argc == 2){
            return (2);
       }
    if (argc > 3){
            return (3);
       }
    if ((filein=fopen (argv[1],"r"))== NULL){   // open input file
            return (4);                         // return error <code 4> if input file is not exist
       } else printf ("\nINPUT  : %s\n",argv[1]);

    if ((fileout=fopen (argv[2],"w"))== NULL){  // open input file
            return (5);                         // return error <code 5> if can not open output file
       } else printf ("OUTPUT : %s\n\n",argv[2]);

    return 0;
}
```

Fluxcalculation.c

```
#include "variables.h"

int fluxcalc(int *LoadXYZ,float *XNEW,float *XOLD,float *MACEf,float POWER,
            float *PZONE, float *RPEAK,int matcount,int nz,int energy_group,float dx,float dy,float
dz){
  int i,j,k,l,maclocate;
  int group;
  int locateijk,locateij;
  float w = 212.0e6*1.6021892e-19;
  float int_a = 0;
  float phi_max[MXGROUP];
  float source_strength;
  float sumfluxall;
  float P;

  int_a = 0;
  for (i = 0 ; i < matcount*nz ; i++){
    for (group = 0 ; group < energy_group ; group++){
      locateij = ij2t(group,i,energy_group,matcount*nz);
      maclocate = ij2t (i,group,matcount*nz,energy_group);
      int_a = int_a + (XOLD[locateij]*MACEf[maclocate]*dx*dy*dz);
    }
  }
```

```
  sumfluxall = int_a;
  source_strength = POWER/(w*sumfluxall);
  if (nz == 1) {source_strength = source_strength * Pi /2/zdim;}
  for (group = 0 ; group < energy_group ; group++){
    for (i = 0 ; i < matcount*nz ; i++){
      locateij = ij2t (group,i,energy_group,matcount*nz);
      XNEW[locateij] = source_strength*XOLD[locateij];
    }
  }
  for (i = 0 ; i < matcount*nz ; i++){
    int_a = 0;
    for (group = 0 ; group < energy_group ; group++){
      locateij = ij2t(group,i,energy_group,matcount*nz);
      maclocate = ij2t (i,group,matcount*nz,energy_group);
      int_a = int_a + (XOLD[locateij]*MACEf[maclocate]*dx*dy*dz);
    }
      PZONE[i] = POWER*int_a/sumfluxall;
  }
  for (i = 0 ; i < matcount ; i++){
    int_a = 0.0;
    for (k = 0 ; k < nz ; k++){
      locateij = ij2t(k,i,nz,matcount);
      int_a = int_a + PZONE[locateij];
    }
    RPEAK[i] = int_a/(POWER/matcount);
  }
  return (0);
}
```

Fuelload.c

```
#include "variables.h"
extern int ij2t (int i,int j,int nx,int ny);
extern int ijk2t (int i,int j,int k,int nx,int ny,int nz);

int fuelload (int *PREVLOAD,int *NEXTLOAD,int *MATNLOAD,int *CHCKLOAD,float *MWDMTU,int matcount,int
nz){
    int locateij1,locateij2,locateij3,locateij4;
    int i,j,k,l;
    int mload;
    int mattemp1;
    int mattemp2;

    l = 0;
    for (i = 0 ; i < matcount ; i++)
      CHCKLOAD[i] = 0;

    for (i = 0 ; i < matcount ; i++){
      if ((PREVLOAD[i] != 0 )&&(CHCKLOAD[i] == 0 )){
      CHCKLOAD[i] = 1;
      for (j = 0 ; j < matcount ; j++){
        if ((PREVLOAD[j] == NEXTLOAD[i])&&(CHCKLOAD[j] == 0)) {
          for (l = 0 ; l < nz ; l++){
            locateij1 = ij2t (l,PREVLOAD[j]-1,nz,matcount);
            locateij2 = ij2t (l,NEXTLOAD[j]-1,nz,matcount);
            MWDMTU[locateij2] = MWDMTU[locateij1];
            MATNLOAD[j] = MA[locateij1];
            MA[locateij2] = MATNLOAD[j];
          }
        CHCKLOAD[j] = 1;
        break;
        }
      }
          for (l = 0 ; l < nz ; l++){
            locateij1 = ij2t (l,PREVLOAD[i]-1,nz,matcount);
            locateij2 = ij2t (l,NEXTLOAD[i]-1,nz,matcount);
            MWDMTU[locateij2] = MWDMTU[locateij1];
            MATNLOAD[i] = MA[locateij1];
            MA[locateij2] = MATNLOAD[i];
          }
        }
    }
//    LOAD FRESH FUEL
    for (i = 0 ; i < matcount ; i++){
      if (PREVLOAD[i] == 0 ){
          for (l = 0 ; l < nz ; l++){
               locateij2 = ij2t (l,NEXTLOAD[i]-1,nz,matcount);
               MWDMTU[locateij2] = 0.0;
               MA[locateij2] = MATNLOAD[i];
          }
      }
    }

    return 0;
}
```

Inputcopy.c

```
#include "variables.h"

int inputcopy (FILE *filein,FILE *fileout)
{
    char ch;                                          // Character read from file
    int counter = 0;                                  // Counter for lines
    int loop,linecount,copy1;                         // Loops

    fprintf (fileout,"\n<0> -------------   Input file   -------------\n\n");
    while(!feof(filein)){
            ch=fgetc (filein);
            if (ch != -1) fputc (ch,fileout);
    }
    fprintf (fileout,"\n<0> ----------------------------------------\n\n");
    return 0;
}
```

Main.c

```
#include "variables.h"
extern filemanage (int argc, char *argv[]);
extern void ERRORINDICATE (int ERRORMESSAGE);
extern int inputcopy(FILE *filein,FILE *fileout);
extern int preprocessor (FILE *filein,FILE *fileout);
extern int fluxcalc(int *LoadXYZ,float *XNEW,float *XOLD,float *MACEf,float POWER,float *PZONE,
                    float *RPEAK,int matcount,int nz,int energy_group,float dx,float dy,float dz);
extern float solvematrix(int *LoadXYZ,float *XOLD,float *XNEW,float *YOLD,float *YNEW,float *SOURCE,
                     float  MATRIX[NMX][7][MXGROUP],float *MACEs,float *MACEf,float *DIFFCO,
                     int matcount,int nx,int ny,int nz,float dx,float dy,float dz,
                     int energy_group);
extern float chemical_shim (float *MACEr,float *MACEa,float *MACEt,float keff1,float chemical_0,int
sign,int matcount, int nz);
extern int printflux (int *LoadXYZ,float *XNEW,int energy_group,int nx,int ny,int nz,FILE *fout1);
extern float fluxavg (float *XNEW, float *XAVG, int matcount, int energy_group, int nz, int group,
float dx ,float dy ,float dz,FILE *favg1);
extern int makemacxs (float *LIBMACvEf,float *LIBMACEf,float *LIBMACEc,float *LIBMACEa,float
*LIBCHI,float *LIBDIFFCO1,
                      float *LIBDIFFCO2,float *LIBMACEt,float *MACvEf,float *MACEf,float
*MACEc,float *MACEa,float *CHI,
                      float *DIFFCO1,float *DIFFCO2,float *MACEt,float *MACAc,float *MACEr,float
*MACEs,float *L,float *MWDMTU,
                      int *ZO,int *MA,int *LY,int matcount,int nx,int ny,int nz,int energy_group,int
BUSTEP,
                      float *MWDSTEP);
extern int makematrix(int energy_group,int nx,int ny,int nz,int matcount);
extern int updateMWD(float BurnInterval,float *PZONE,float *MWDMTU,int matcount,int nz,int BUSTEP);
extern int peaking (int *LoadXYZ,float *RPEAK,int nx,int ny,FILE *fout1);
extern float calculateMWD (float *BURNMWD,float *MWDMTU,int matcount,int nz,FILE *CALMWD);
extern int fuelload (int *PREVLOAD,int *NEXTLOAD,int *MATNLOAD,int *CHCKLOAD,float *MWDMTU,int
matcount,int nz);
FILE *filein,*fileout;

int main(int argc, char *argv[])
{
    int   ERRORMESSAGE = 0;
    int   subreturn = 0;
    int   kloopadj;
    int   cycle_count;
    int   i,j,k,l1,l2,l3,group,l4,l5;
    int   locateij,locateijk;
    float BurnValue;
    float  K[1000];
    float SHIM[1000];
    float BURNSUM [1000];
    float BURNTEMPBOC;
    float BURNTEMPEOC;
    char szInput[80];
    int   LOOP1,LOOP2 ;
    int   CALLOOP;
    float summ;
    float ktemp;
    int   OUT_LOOP;
    FILE  *mwdout;
    FILE  *fluxout;

    if ((subreturn = filemanage(argc,argv)) != 0)
       ERRORINDICATE (subreturn); else
       if ((subreturn = inputcopy(filein,fileout)) != 0){
          ERRORINDICATE (subreturn);
          printf ("(--- file process error  ---) check %d\n",subreturn);
          fclose (filein);
          fclose (fileout);
          return 0;
              }
              else
              printf ("(--- file process passed ---)\n",subreturn);
    if ((subreturn = preprocessor(filein,fileout)) != 0){
```

```
      ERRORINDICATE (subreturn);
      printf ("(--- preprocessor error  ---) check %d\n",subreturn);
      return 0;
      } else {
      if ((mwdout = fopen ("mwdout.o","w+t"))== NULL){
           return (12);
         }
      if (FLUXOPT == 1) if ((fluxout = fopen ("fluxout.o","w+t"))== NULL){
           return (14);
         }

        fprintf (fileout,"\n\n");
        printf ("(--- preprocessor passed  ---)\n" );
        for (LOOP2 = 0 ; LOOP2 < NLOAD ; LOOP2++)
        {
        BurnValue = 0.0;
        if (BurnOPT == 0) LOOP1 = 1; else LOOP1 = BurnTIME;
        15 = 0;
        BURNTEMPBOC = 0;
        fprintf (mwdout,"------ BOC ----------------------\n");
        for (14 = 0 ; 14 < matcount ; 14++){
          BURNTEMPBOC = BURNTEMPBOC + MWDMTU[14];
          fprintf (mwdout,"%13.4e",MWDMTU[14]);
          15++;
          if (15 == 5) {fprintf (mwdout,"\n"); 15 = 0;}
        }
        for (12 = 0 ; 12 < LOOP1 ; 12++)
        {
        subreturn = makemacxs (LIBMACvEf,LIBMACEf,LIBMACEc,LIBMACEa,LIBCHI,LIBDIFFCO1,LIBDIFFCO2,
                               LIBMACEt,MACvEf,MACEf,MACEc,MACEa,CHI,DIFFCO1,DIFFCO2,MACEt,MACAc,
                               MACEr,MACEs,L,MWDMTU,ZO,MA,LY,matcount,nx,ny,nz,energy_group,
                               BUSTEP,MWDSTEP);
        if (subreturn != 0 ) { printf ("TERMINATE WITH ERROR\n") ; break;}
        printf ("%4d  ",12);
        chemical_t = 0.0;
        OUT_LOOP = 0;
        CALLOOP = 0;
        do
        {
           makematrix(energy_group,nx,ny,nz,matcount);
           keff1 =
solvematrix(LoadXYZ,XOLD,XNEW,YOLD,YNEW,SOURCE,MATRIX,MACEs,MACEf,DIFFCO1,matcount,nx,ny,nz,dx,dy,dz
,energy_group);
           K[12] = keff1;
           if (CALLOOP == 0) ktemp = keff1;
           if (SHIMOPT == 1){
              if (keff1 > 1.001) chemical_t = chemical_t+chemical_shim
(MACEr,MACEa,MACEt,keff1,chemical_0,1,matcount,nz);
           } else OUT_LOOP = 1;
           SHIM[12] = chemical_t;
           CALLOOP++;
        } while ((keff1 > 1.001)&&(OUT_LOOP == 0));
        fluxcalc(LoadXYZ,XNEW,XOLD,MACEf,POWER,PZONE,RPEAK,matcount,nz,energy_group,dx,dy,dz);
        peaking (LoadXYZ,RPEAK,nx,ny,fileout);
        subreturn = updateMWD(BurnInterval,PZONE,MWDMTU,matcount,nz,BUSTEP);
        if (subreturn != 0 ) { printf ("TERMINATE WITH ERROR , %5d\n",subreturn) ; break;}
        BURNSUM[12] = calculateMWD (BURNMWD,MWDMTU,matcount,nz,mwdout);
        if ((ktemp < 0.9999)&&(CALLOOP ==1)) break;
        }
        15 = 0;
        BURNTEMPEOC = 0.0;
        fprintf (mwdout,"\n------ EOC ----------------------\n");
        for (14 = 0 ; 14 < matcount ; 14++){
          BURNTEMPEOC = BURNTEMPEOC + MWDMTU[14];
          fprintf (mwdout,"%13.4e",MWDMTU[14]);
          15++;
          if (15 == 5) {fprintf (mwdout,"\n"); 15 = 0;}
        }
        fprintf (mwdout,"\n-- xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx--\n");
        if (LOADOPT != 0) {
            printf ("LOAD NEW BATCH\n");
            fprintf (fileout,"\nLOAD NEW BATCH , %5d\n",LOOP2);
            fuelload (PREVLOAD,NEXTLOAD,MATNLOAD,CHCKLOAD,MWDMTU,matcount,nz);
            }
        if (FLUXOPT==1) printflux (LoadXYZ,XNEW,energy_group,nx,ny,nz,fluxout);
        for (i = 0 ; i < 12 ; i++){
          fprintf (fileout,"keff [%3d] = %9.6f , SHIM[%3d] = %13.4e\n",i,K[i],i,SHIM[i]);
        }
        fprintf (fileout,"BURNSUM = %13.4e\n",BURNTEMPEOC-BURNTEMPBOC);

        }
       }
    time (&start);
    date_time1 = localtime (&start);
    strftime(szInput,80,"%X ",date_time1);
    fprintf (fileout,"Calculation Finished on %s\n",szInput);
```

```
        fclose (filein);
        fclose (fileout);
        fclose (mwdout);
        if (FLUXOPT==1) fclose (fluxout);
    free (LoadXYZ);
    free (LIBDIFFCO1);
    free (LIBDIFFCO2);
    free (LIBCHI);
    free (LIBMACEt);
    free (LIBMACEs);
    free (LIBMACEa);
    free (LIBMACvEf);
    free (LIBMACEf);
    free (MACvEf);
    free (MACEf);
    free (MACEc) ;
    free (MACEa);
    free (CHI);
    free (DIFFCO1);
    free (DIFFCO2);
    free (MACEt);
    free (MACAc);
    free (MACEr);
    free (L);
    free (SOURCE);
    free (MACEs);
    free (XvEfn);
    free (XNEW);
    free (XOLD);
    free (YNEW);
    free (YOLD);
    free (FLUX);
    free (LY);
    free (MA);
    free (ZO);
    free (MWDMTU);
    free (PZONE);
        printf ("succeeded\n");
        return 0;
}
```

Makemacxs.c

```
#include "variables.h"
extern FILE *ppfile, *pptemp;
extern FILE *fileout;
extern int makematrix (int energy_group,int nx,int ny,int nz,int matcount);
extern int solvematrix(void);
extern int ij2t (int i,int j,int nx,int ny);
float interpolation (float MACa,float MACb,float MWDa,float MWDb,float MWDx){
    float result;
    result = MACa + ((MWDx-MWDa)*(MACb-MACa)/(MWDb-MWDa));
    return result;
}

int makemacxs (float *LIBMACvEf,
               float *LIBMACEf,
               float *LIBMACEc,
               float *LIBMACEa,
               float *LIBCHI,
               float *LIBDIFFCO1,
               float *LIBDIFFCO2,
               float *LIBMACEt,
               float *MACvEf,
               float *MACEf,
               float *MACEc,
               float *MACEa,
               float *CHI,
               float *DIFFCO1,
               float *DIFFCO2,
               float *MACEt,
               float *MACAc,
               float *MACEr,
               float *MACEs,
               float *L,
               float *MWDMTU,
               int   *ZO,
               int   *MA,
               int   *LY,
               int   matcount,
               int   nx,
               int   ny,
               int   nz,
               int   energy_group,
               int   BUSTEP,
               float *MWDSTEP){
```

```
    int    i,j,k,l,m;
    int    locateij;
    int    low_interpo;
    int    high_interpo;
    int    locatelow;
    int    locatehi;
    int    maclocate;
    int    locateijk;
    int    chilocate;
    int    eslocate;
    int    micestocate;
    int    temp;

    for (i = 0 ; i < matcount*nz ; i++){
      for (j = 0 ; j < BUSTEP ; j++){
        if (MWDMTU[i] > MWDSTEP[j]) low_interpo = j;
        else if (MWDMTU[i] <= MWDSTEP[j]) {
                high_interpo = j;
                break;
             }
        else if (MWDMTU[i] >  MWDSTEP[BUSTEP]) {
                printf ("**DATA OUT OF BOUND\n" );
                return (15);
             }
      }
    if (high_interpo == 0) {high_interpo = 1;low_interpo = 0;}
          for (k = 0 ; k < energy_group ; k++){
            locatelow = ijk2t(low_interpo,k,MA[i]-1,BUSTEP,energy_group,MATNUM);
            locatehi  = ijk2t(high_interpo,k,MA[i]-1,BUSTEP,energy_group,MATNUM);
            maclocate = ij2t (i,k,matcount*nz,energy_group);
            MACvEf [maclocate] = interpolation
(LIBMACvEf[locatelow],LIBMACvEf[locatehi],MWDSTEP[low_interpo],MWDSTEP[high_interpo],MWDMTU[i]);
            MACEf  [maclocate] = interpolation
(LIBMACEf[locatelow],LIBMACEf[locatehi],MWDSTEP[low_interpo],MWDSTEP[high_interpo],MWDMTU[i]);
            MACEc  [maclocate] = interpolation
(LIBMACEc[locatelow],LIBMACEc[locatehi],MWDSTEP[low_interpo],MWDSTEP[high_interpo],MWDMTU[i]);
            MACEa  [maclocate] = interpolation
(LIBMACEa[locatelow],LIBMACEa[locatehi],MWDSTEP[low_interpo],MWDSTEP[high_interpo],MWDMTU[i]);
            CHI    [maclocate] = interpolation
(LIBCHI[locatelow],LIBCHI[locatehi],MWDSTEP[low_interpo],MWDSTEP[high_interpo],MWDMTU[i]);
            DIFFCO1[maclocate] = interpolation
(LIBDIFFCO1[locatelow],LIBDIFFCO1[locatehi],MWDSTEP[low_interpo],MWDSTEP[high_interpo],MWDMTU[i]);
            DIFFCO2[maclocate] = interpolation
(LIBDIFFCO2[locatelow],LIBDIFFCO2[locatehi],MWDSTEP[low_interpo],MWDSTEP[high_interpo],MWDMTU[i]);
            MACEt  [maclocate] = interpolation
(LIBMACEt[locatelow],LIBMACEt[locatehi],MWDSTEP[low_interpo],MWDSTEP[high_interpo],MWDMTU[i]);
            MACAc  [maclocate] = interpolation
(LIBMACAc[locatelow],LIBMACAc[locatehi],MWDSTEP[low_interpo],MWDSTEP[high_interpo],MWDMTU[i]);
          }
          for (k = 0 ; k < energy_group ; k++){
            chilocate = ij2t (i,k,matcount*nz,energy_group);
            for (l = 0 ; l < energy_group ; l++){
              locateij = ij2t (k,l,energy_group,energy_group);
              locatelow = ijk2t (low_interpo,locateij,MA[i]-
1,BUSTEP,energy_group*energy_group,MATNUM);
              locatehi  = ijk2t (high_interpo,locateij,MA[i]-
1,BUSTEP,energy_group*energy_group,MATNUM);
              locateijk = ij2t (i,locateij,matcount*nz,energy_group*energy_group);
              temp = ij2t (i,l,matcount*nz,energy_group);
              MACEs[locateijk] =
interpolation(LIBMACEs[locatelow],LIBMACEs[locatehi],MWDSTEP[low_interpo],MWDSTEP[high_interpo],MWDM
TU[i]);
              XvEfn[locateijk] = CHI[chilocate]*MACvEf[temp];
              if (k == l) {
                  maclocate = ij2t (i,k,matcount*nz,energy_group);
                  MACEr[maclocate] = MACEt[maclocate]-MACEs[locateijk];
                  L    [maclocate] = MACEr[maclocate]/DIFFCO1[maclocate];
                  B2   [maclocate] = Bg2/DIFFCO1[maclocate];
              }
            }
          }
    }
    return 0;
}
```

Makematrix.c

```
#include "variables.h"

extern int ij2t (int i,int j,int nx,int ny);
extern int ijk2t (int i,int j,int k,int nx,int ny,int nz);

int makematrix (int energy_group,int nx,int ny,int nz,int matcount){

    int   i,j,k,m;
    int   temp;
```

```
    int   tc,tb,tf,tu,td,tl,tr;
    int   group;
    int   matid;
    int   maclocate;
    int   mattemp;
    int   locateij;
    int   l1,l2,l3,l4;

// CLEAR VALUE IN ARRAY
    for (i = 0; i < energy_group; i++)
        for (j = 0; j < nx*ny*nz; j++)
            for (k = 0; k < 7; k++)
                MATRIX[j][k][i]= 0.0;
// ------------------------------------------------------
  for (group = 0; group < energy_group; group++){
      temp = 0;
      for (k = 0; k < nz; k++){
          for (i = 0; i < nx; i++){
              for (j = 0; j < ny; j++){
                  tc = ijk2t (i,j,k,nx,ny,nz);
                  if ((i-1) > - 1) tb = ijk2t(i-1,j,k,nx,ny,nz); else tb = -1;
                  if ((j-1) > - 1) tl = ijk2t(i,j-1,k,nx,ny,nz); else tl = -1;
                  if ((k-1) > - 1) td = ijk2t(i,j,k-1,nx,ny,nz); else td = -1;
                  if ((i+1) < nx)  tf = ijk2t(i+1,j,k,nx,ny,nz); else tf = -1;
                  if ((j+1) < ny)  tr = ijk2t(i,j+1,k,nx,ny,nz); else tr = -1;
                  if ((k+1) < nz)  tu = ijk2t(i,j,k+1,nx,ny,nz); else tu = -1;
                  if (LoadXYZ[tc] != 0) {
                     maclocate = ijk2t (MATRIXTC[tc],group,k,matcount,energy_group,nz);
                     if (nz == 1){
                         MATRIX[temp][0][group] = -(2.0/dx/dx+2.0/dy/dy-B2[maclocate]+L[maclocate]);
                         }
                     else
                         MATRIX[temp][0][group] = -(2.0/dx/dx+2.0/dy/dy+2.0/dz/dz+L[maclocate]);
                  if ((tb !=-1)&&(LoadXYZ[tb] != 0)) {
                     MATRIX[temp][1][group] = 1.0/dx/dx;}      //ab = 1
                  if ((tf !=-1)&&(LoadXYZ[tf] != 0)){
                     MATRIX[temp][2][group] = 1.0/dx/dx;}      //af = 2
                  if ((tl !=-1)&&(LoadXYZ[tl] != 0)){
                     MATRIX[temp][3][group] = 1.0/dy/dy;}      //al = 3
                  if ((tr !=-1)&&(LoadXYZ[tr] != 0)){
                     MATRIX[temp][4][group] = 1.0/dy/dy;}      //ar = 4
                  if ((td !=-1)&&(LoadXYZ[td] != 0)){
                     if (nz == 1)
                     MATRIX[temp][5][group] = 0.0;             //ad = 5
                     else
                     MATRIX[temp][5][group] = 1.0/dz/dz;}      //ad = 5
                  if ((tu !=-1)&&(LoadXYZ[tu] != 0)){
                     if (nz == 1)
                     MATRIX[temp][6][group] = 0.0;             //au = 6
                     else
                     MATRIX[temp][6][group] = 1.0/dz/dz;}      //au = 6
                     temp = temp + 1;
                     }
              }
          }
      }
  }

    for (group = 0; group < energy_group ; group++){
        temp = 0;
        for (k = 0 ; k < nz ; k++){
             for (m = 0 ; m < matcount ; m++){
             temp = temp + 1;
            }
        }
    }
    return 0;


}
```

Preprocessor.c

```
#include "variables.h"

extern int ijk2t (int i,int j,int k,int nx,int ny,int nz);
extern int ij2t (int i,int j,int nx, int ny);

int preprocessor (FILE *filein,FILE *fileout)
{

    char  linein [200];
    char  strmaterial [20];
    char  szInput[80];
    FILE  *fileintemp;
        FILE  *pptemp1;
        FILE  *macxstemp;
```

```
    FILE  *MACREF;
    FILE  *HIST;
    int   i,j,k,l,hnz;            // index for loop
    int   locateij;
    int   locateijk;
        int   mattemp;
        int   steptemp;
        int   ccount;
        int   dcount;
    time (&start);
    date_time1 = localtime (&start);
    strftime(szInput,80,"%X ",date_time1);
    fprintf (fileout,"Calculation Started on %s\n",szInput);
    rewind (filein);
    if ((fileintemp = fopen ("FILEINTEMP","w+t"))== NULL){
            return (6);
       }    if ((pptemp1 = fopen ("FP012","w+t"))== NULL){
            return (7);
       }    if ((macxstemp = fopen ("MACXSTEMP","w+t"))== NULL){
            return (11);
       }
    while (!feof(filein)) {
        fgets(linein,200,filein);
        if ((linein[0] != '#')&&(!feof(filein))) { // Rearrange input file without Comment #
           fprintf (fileintemp,"%s",linein);
        }
    }
    rewind(fileintemp);
    fscanf (fileintemp,"%f%f%f%d%d%d%d",&xdim                    // Read X dimension
                                       ,&ydim                    // Read Y dimension
                                       ,&zdim                    // Read Z dimension
                                       ,&nx                      // Read number of node X
                                       ,&ny                      // Read number of node Y
                                       ,&nz                      // Read number of node Z
                                       ,&matcount);
    if ((nx*ny*nz) > NMX) {
       fprintf (fileout,"\n*****  Matrix size is too large handle (> %d). calculation
terminated!!!\n",NMX);
       if (remove ("FP002")==-1) printf ("\nERROR REMOVE FP002\n\n");
       return (8);
    }

    nnn   = nx*ny*nz;

    dx = xdim/(nx+1);
    dy = ydim/(ny+1);
    dz = zdim/(nz+1);
    if (nz==1) dz = 1;
    Bg2 = (Pi/zdim)*(Pi/zdim);

    keff0 = 1.0;                                          // Initialize K 0 (Previous K)
    keff1 = 1.0;                                          // Initialize K 1 (Present  K)
    kerr  = 1.0;                                          // Initialize relative error

    fscanf (fileintemp,"%f%f",&Ferr_criteria              // Read Flux error Criteria
                             ,&Kerr_criteria);            // Read K error Criteria
    fscanf (fileintemp,"%f"  ,&POWER);                    // Read Power
    fscanf (fileintemp,"%d"  ,&energy_group);             // Read number of energy group

    LoadXYZ  = (int*) malloc((nx*ny*nz)*sizeof(int*));   // allocate memory for matrix nx*ny*nz
    MATRIXTC = (int*) malloc((nx*ny*nz)*sizeof(int*));   // allocate memory for matrix nx*ny*nz
    MA     = (int*)   malloc(matcount*nz*sizeof (int *)); // material in use (different fuel)
    ccount = 0;
        for (k = 0; k < nz ; k++){
      dcount = 0 ;
      for (i = 0; i < nx ; i++){
        for (j = 0; j < ny ; j++){
                  locateijk = ijk2t(i,j,k,nx,ny,nz);
          fscanf (fileintemp,"%d",&LoadXYZ[locateijk]);  // read each element from core
          if (LoadXYZ[locateijk] != 0 ) {
              MA[ccount] = LoadXYZ[locateijk];
              MATRIXTC[locateijk] = dcount;
              fprintf (pptemp1,"MA[%5d] = %5d , locateijk = %5d , MATRIXTC = %5d
\n",ccount,MA[ccount],locateijk,MATRIXTC[locateijk]);
              ccount++;
              dcount++;
          }
       }
      }
        }
    fscanf (fileintemp,"%s",macrefpath);
    fscanf (fileintemp,"%d",&BurnOPT);
    fscanf (fileintemp,"%d",&SHIMOPT);
    fscanf (fileintemp,"%f",&BurnTIME);
    fscanf (fileintemp,"%f",&BurnInterval);
    fscanf (fileintemp,"%f",&u_weight);
```

```
    printf ("\nMAC-XS-FILE : %s\n",macrefpath);
    printf ("\nBURNUP = %d (0 = No , 1 = Yes)\n",BurnOPT);
    printf ("\nBURN TIME = %.3f days\n",BurnTIME*BurnInterval);
    printf ("* BURN INTERVAL = %f days\n",BurnInterval);

// Open Macroscopic Cross Section File
    if ((MACREF = fopen (macrefpath,"r"))== NULL){
         return (13);
       } else printf ("\nOPEN MACREF CROSS SECTION SUCCESSFUL\n");

    while (!feof(MACREF)) {
        fgets(linein,200,MACREF);
        if ((linein[0] != '#')&&(!feof(MACREF))) { // Rearrange macrefpath file without Comment #
           fprintf (macxstemp,"%s",linein);
        }
    }
    rewind(macxstemp);
//  ---------------------------------------------------------------
    MWDMTU = (float*) malloc(matcount*nz*sizeof (float *));
    BURNMWD= (float*) malloc(matcount*sizeof (float *));
    for (i = 0 ; i < matcount*nz ; i++){
    fscanf (fileintemp,"%f",&MWDMTU[i]);
    }
    fscanf (fileintemp,"%d",&LOADOPT);
    fscanf (fileintemp,"%d",&NLOAD);
    fscanf (fileintemp,"%d",&FLUXOPT);
    if (LOADOPT != 0) {
        printf ("\nfound batch loading = %4d\n",LOADOPT);
        PREVLOAD = (int*) malloc (LOADOPT*matcount*sizeof (int* ));
        NEXTLOAD = (int*) malloc (LOADOPT*matcount*sizeof (int* ));
        MATNLOAD = (int*) malloc (LOADOPT*matcount*sizeof (int* ));
        CHCKLOAD = (int*) malloc (LOADOPT*matcount*sizeof (int* ));
        for (i = 0 ; i < LOADOPT ; i++)
          for (j = 0 ; j < matcount ; j++){
          locateij = ij2t (i,j,LOADOPT,matcount);
          fscanf (fileintemp,"%d%d%d",&PREVLOAD[locateij],&NEXTLOAD[locateij],&MATNLOAD[locateij]);
          CHCKLOAD[locateij] = 0;
          fprintf (pptemp1,"%4d , %4d -
> %4d , %4d\n",locateij,PREVLOAD[locateij],NEXTLOAD[locateij],MATNLOAD[locateij]);

          }

    }

//  Read Macro Scopic Cross-Section

    fscanf (macxstemp,"%d%d",&MATNUM,&BUSTEP);

    MWDSTEP   = (float*) malloc((BUSTEP)*sizeof(float*));
    MWDSTEP[0] = 0.0;
    for (i = 1 ; i < BUSTEP ; i++){
      fscanf (macxstemp,"%f",&MWDSTEP[i]);
    }

    LIBMACvEf = (float*) malloc(energy_group*MATNUM*BUSTEP*sizeof(float *));
    LIBMACEf  = (float*) malloc(energy_group*MATNUM*BUSTEP*sizeof(float *));
    LIBMACEc  = (float*) malloc(energy_group*MATNUM*BUSTEP*sizeof(float *));
    LIBMACEa  = (float*) malloc(energy_group*MATNUM*BUSTEP*sizeof(float *));
    LIBCHI    = (float*) malloc(energy_group*MATNUM*BUSTEP*sizeof(float *)); // CHI
    LIBDIFFCO1= (float*) malloc(energy_group*MATNUM*BUSTEP*sizeof(float *)); // Diffusion Coeff
    LIBDIFFCO2= (float*) malloc(energy_group*MATNUM*BUSTEP*sizeof(float *)); // Diffusion Coeff
    LIBMACEt  = (float*) malloc(energy_group*MATNUM*BUSTEP*sizeof(float *));
    LIBMACAc  = (float*) malloc(energy_group*MATNUM*BUSTEP*sizeof(float *));
    LIBMACEs  = (float*) malloc(energy_group*energy_group*MATNUM*BUSTEP*sizeof(float*));

    for (i = 0 ; i < MATNUM ; i++){
      for (j = 0 ; j < BUSTEP ; j++){
        fscanf (macxstemp,"%d%d",&mattemp,&steptemp);
        for (k = 0 ; k < energy_group ; k++){
          locateijk = ijk2t(j,k,i,BUSTEP,energy_group,MATNUM);
          fscanf  (macxstemp,"%f%f%f%f%f%f%f%f%f",&LIBMACvEf  [locateijk]
                                                 ,&LIBMACEf   [locateijk]
                                                 ,&LIBMACEc   [locateijk]
                                                 ,&LIBMACEa   [locateijk]
                                                 ,&LIBCHI     [locateijk]
                                                 ,&LIBDIFFCO1 [locateijk]
                                                 ,&LIBDIFFCO2 [locateijk]
                                                 ,&LIBMACEt   [locateijk]
                                                 ,&LIBMACAc   [locateijk]);
        }
        for (k = 0 ; k < energy_group ; k++){
          for (l = 0 ; l < energy_group ; l++){
              locateij  = ij2t (k,l,energy_group,energy_group);
              locateijk = ijk2t (j,locateij,i,BUSTEP,energy_group*energy_group,MATNUM);
              fscanf (macxstemp,"%f",&LIBMACEs[locateijk]);
```

```
        }
      }
    }
  }

  MACvEf = (float*) malloc(energy_group*matcount*nz*sizeof(float *));
  MACEf  = (float*) malloc(energy_group*matcount*nz*sizeof(float *));
  MACEc  = (float*) malloc(energy_group*matcount*nz*sizeof(float *));
  MACEa  = (float*) malloc(energy_group*matcount*nz*sizeof(float *));
  CHI    = (float*) malloc(energy_group*matcount*nz*sizeof(float *));

  DIFFCO1= (float*) malloc(energy_group*matcount*nz*sizeof(float *));

  DIFFCO2= (float*) malloc(energy_group*matcount*nz*sizeof(float *));

  MACEt  = (float*) malloc(energy_group*matcount*nz*sizeof(float *));
  MACAc  = (float*) malloc(energy_group*matcount*nz*sizeof(float *));
  MACEr  = (float*) malloc(energy_group*matcount*nz*sizeof(float *));
  MACEs  = (float*) malloc(energy_group*energy_group*matcount*nz*sizeof(float*));
  L      = (float*) malloc(energy_group*matcount*nz*sizeof(float *));
  B2     = (float*) malloc(energy_group*matcount*nz*sizeof(float *));
  SOURCE = (float*) malloc(energy_group*matcount*nz*sizeof(float *));
  XvEfn  = (float*) malloc(energy_group*energy_group*matcount*nz*sizeof(float*));
  XNEW   = (float*) malloc(energy_group*matcount*nz*sizeof(float *));
  XOLD   = (float*) malloc(energy_group*matcount*nz*sizeof(float *));
  YNEW   = (float*) malloc(energy_group*matcount*nz*sizeof(float *));
  YOLD   = (float*) malloc(energy_group*matcount*nz*sizeof(float *));
  FLUX   = (float*) malloc(energy_group*matcount*nz*sizeof(float *));
  PZONE  = (float*) malloc(energy_group*matcount*nz*sizeof(float *));
  RPEAK  = (float*) malloc(matcount*sizeof(float *));

 for (i = 0; i < energy_group*matcount*nz ; i++){
   XOLD [i] = 1.0;
   XNEW [i] = 1.0;
   YOLD [i] = 0.0;
   YNEW [i] = 0.0;
 }

       fclose (fileintemp);
       fclose (pptemp1);
       fclose (macxstemp);
       fclose (MACREF);
       return 0;
}
```

Variables.h

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <math.h>
#include <process.h>
#define Pi 22/7
#define MXISOTOPE  11                              // Maximum Isotope
#define MAX_RESERVE 2
#define NMX 125000                                 // Maximum nx * ny * nz
#define GMX 50                                     // Geometry maximum
#define MXGROUP 8                                  // Maximum Energy Group
#define MXMAT 20                                   // Maximum Material
#define MXNAME 10                                  // Maximum Material Name String
#define MXMATDIF  10

/* Global Variable*/
float  Ferr_criteria;            // Flux Error Criteria
float  Kerr_criteria;            // Keffective Error Criteria
int    nx,ny,nz,n,nn,nnn;        // nodes x,y,z, nnn = x*y*z
int    matcount;                 // number of material from input file
int    *LoadXYZ;                 // stored 1 dimension array size of nx*ny*nz
int    *MATRIXTC;
int    *MATRIXTF;
int    *MATRIXTB;
int    *MATRIXTR;
int    *MATRIXTL;
int    *MATRIXTU;
int    *MATRIXTD;
int    id[NMX];
int    *matnumber;
int    matcomp[MXNAME];          // material composition count
int    correctfac;
int    maxmatcomp;
int    MATNUM,BUSTEP;

char   filenameout [256];
char   filenamein  [256];
char   macrefpath[100];          // read path from input data MACREF

float  xdim;
```

```
float  ydim;
float  zdim;
float  *SOURCE;
float  *MICEf;
float  *MICvEf;
float  *MICEa;
float  *MICEr;
float  *MICEc;
float  *MICEt ;
float  *MICSs ;
float  *LIBMACvEf;
float  *LIBMACEf;
float  *LIBMACEc;
float  *LIBMACEa;
float  *LIBCHI;
float  *LIBDIFFCO1;
float  *LIBDIFFCO2;
float  *LIBMACEt;
float  *LIBMACAc;
float  *LIBMACEs;
float  *MWDSTEP;
float  *MWDMTU;
float  *MACvEf;
float  *MACEf;
float  *MACEc;
float  *MACEa;
float  *CHI;
float  *DIFFCO1;
float  *DIFFCO2;
float  *MACEt;
float  *MACAc;
float  *MACEs;
float  *MACSs;
float  *MACEr;
float  u_weight;
float  dx,dy,dz;
float  MATRIX[NMX][7][MXGROUP];
float  *XNEW, *XOLD;              // [Fast][Thermal] group of previous generation (n-1)
float  *YNEW, *YOLD;              // [Fast][Thermal] group of current generation (n)
float  *XvEfn , *L ;
float  *B2;
float  *Ndensity;
float  *FLUX;
float  keff0;
float  keff1;
float  kerr;
double dif;
time_t start,end;
float  POWER;
int    energy_group;
float  *MICEs;
int    *LY;
int    *MA;
int    *ZO;
int    *PREVLOAD;
int    *NEXTLOAD;
int    *MATNLOAD;
int    *CHCKLOAD;
int    BurnOPT;
int    SHIMOPT;
int    LOADOPT;
int    NLOAD;
int    FLUXOPT;
float  BurnInterval;
float  BurnTIME;
float  chemical_t;
float  chemical_0;
float  *PZONE;
float  *RPEAK;
float  *BURNMWD;
float  Bg2;

struct tm *date_time1, *date_time2;
```

solvematrix.c

```
#include "variables.h"

extern FILE  *filein,*fileout;
extern int makematrix (void);
extern float chemical_shim (float *MACEr,float *MACEt,float KNEW,int sign);
extern int ijk2t (int i,int j,int k,int nx,int ny,int nz);
extern float integral (float *XOLD , int matcount , int energy_group , int nz , float dx , float
dy , float dz);
extern int fluxcalc(int *LoadXYZ,float *XNEW,float *XOLD,float *MACEf,float POWER,float *PZONE,
                    float *RPEAK,int matcount,int nz,int energy_group,float dx,float dy,float dz);
```

```
extern float fluxavg (float *XNEW, float *XAVG, int matcount, int energy_group, int nz, int group,
float dx ,float dy ,float dz,FILE *favg1);

int gauss (float *abuff,float *XOLD,float *YOLD,float *YNEW,int matcount,int nx,int ny,int nz,int
group,float ErrCriteria){
    int   i,j,k,ci,g;
    int    nnn;
    int    matXnz;
    int    tc,tb,tf,tr,tl,td,tu;
    int    locatec;
    int    locateb;
    int    locatef;
    int    locater;
    int    locatel;
    int    located;
    int    locateu;
    int    index;
    int    ctindex;
    int    iterationcount;
    int    errorhelp = 0;
    float  Xerror;
    float  XBUFF[7];
    float *XDUMMY;
    nnn = nx*ny*nz;
    matXnz = matcount*nz;
    XDUMMY = (float*) malloc (matcount*nz*energy_group*sizeof(float *));
    for (i = 0 ; i < matXnz*energy_group ; i++)
         XDUMMY[i] = XOLD[i];
    for (i = 0; i < matXnz ; i++){
         YNEW[group*matXnz+i] = YOLD[group*matXnz+i];
    }
    index = 0;
    iterationcount = 0;
    do {
      Xerror = 0;


      for (k = 0; k < nz; k++){
          for (i = 0; i < nx; i++){
              for (j = 0; j < ny; j++){
                  for (ci = 0 ; ci < 7 ; ci++) XBUFF[ci] = 0.0;
                  tc = ijk2t (i,j,k,nx,ny,nz);
                  if (LoadXYZ[tc] > 0) {              // ac = 0
                     locatec = group*matXnz+MATRIXTC[tc]+k*matcount;
                     XBUFF[0] = XOLD[locatec];
                  if ((i-1) > - 1) {
                     tb = ijk2t(i-1,j,k,nx,ny,nz);  // ab = 1
                     if (LoadXYZ[tb] > 0){
                     locateb = group*matXnz+MATRIXTC[tb]+k*matcount;
                     XBUFF[1] = XOLD[locateb];
                     }
                  }
                  if ((i+1) <  nx) {
                     tf = ijk2t(i+1,j,k,nx,ny,nz);  // af = 2
                     if (LoadXYZ[tf] > 0){
                     locatef = group*matXnz+MATRIXTC[tf]+k*matcount;
                     XBUFF[2] = XOLD[locatef];
                     }
                  }
                  if ((j-1) > - 1) {
                     tl = ijk2t(i,j-1,k,nx,ny,nz);  // al = 3
                     if (LoadXYZ[tl] > 0){
                     locatel = group*matXnz+MATRIXTC[tl]+k*matcount;
                     XBUFF[3] = XOLD[locatel];
                     }
                  }
                  if ((j+1) <  ny) {
                     tr = ijk2t(i,j+1,k,nx,ny,nz);  // ar = 4
                     if (LoadXYZ[tr] > 0){
                     locater = group*matXnz+MATRIXTC[tr]+k*matcount;
                     XBUFF[4] = XOLD[locater];
                     }
                  }
                  if ((k-1) > - 1) {
                     td = ijk2t(i,j,k-1,nx,ny,nz);  // ad = 5
                     if (LoadXYZ[td] > 0){
                     located = group*matXnz+MATRIXTC[td]+(k-1)*matcount;
                     XBUFF[5] = XOLD[located];
                     }
                  }
                  if ((k+1) <  nz) {
                     tu = ijk2t(i,j,k+1,nx,ny,nz);  // au = 6
                     if (LoadXYZ[tu] > 0){
                     locateu = group*matXnz+MATRIXTC[tu]+(k+1)*matcount;
                     XBUFF[6] = XOLD[locateu];
                     }
```

```
                }

                for (ci = 1; ci < 7; ci++){
                    index = (MATRIXTC[tc])*7+k*matcount*7+ci;
                    YOLD[locatec] = YOLD[locatec] - abuff[index]*XBUFF[ci];
                }
                ctindex = (MATRIXTC[tc])*7+k*matcount*7;
                XNEW[locatec] = (YOLD[locatec]/abuff[ctindex]) ;
                Xerror = Xerror+(fabs(XNEW[locatec]-XOLD[locatec])/XNEW[locatec]);
                XOLD[locatec] = XNEW[locatec];
             }
            }
          }
      }


   for (i = 0; i < matXnz ; i++){
       YOLD[group*matXnz+i] = YNEW[group*matXnz+i];
   }
   iterationcount++;
  }  while ((iterationcount < 100)&&(Xerror > ErrCriteria));
  for (i = 0 ; i < matXnz*energy_group ; i++)
    XOLD[i] = XDUMMY[i];

  free (XDUMMY);
}

float solvematrix(int   *LoadXYZ,
                  float *XOLD,
                  float *XNEW,
                  float *YOLD,
                  float *YNEW,
                  float *SOURCE,
                  float  MATRIX[NMX][7][MXGROUP],
                  float *MACEs,
                  float *MACEf,
                  float *DIFFCO,
                  int   matcount,
                  int   nx,
                  int   ny,
                  int   nz,
                  float dx,
                  float dy,
                  float dz,
                  int   energy_group){

  int    loop_count = 0;
  int    i,j,k,l,tt,kei;
  int    m;
  int    index;
  int    temp;
  int    group;
  int    chi;
  int    CHIindex;
  float  FI, FF;
  float  *abuff;
  float  EsTemp;
  int    dbuff;
  float  *EsSum ;
  float  SIGN,w;
  float  ErrorPrevious = 1.0;
  float  SOURCETemp;
  int    kloopadj;
  int    locateij, locateijk, locateijkl;
  int    maceslocate;
  int    maclocate;
  int    fluxlocate;
  float  avgvEf;
  float  avgEr;
  float  avgEa;

  abuff  = (float*) malloc(matcount*nz*energy_group*7*sizeof (float *));
  EsSum = (float*) malloc(matcount*nz*energy_group*sizeof (float *));

  keff0 = 1.0;
  kerr  = 1.0;
  loop_count = 0;
  do
  {
    FI = integral (XOLD , matcount , energy_group , nz , dx , dy , dz);
    for (i = 0; i < matcount*nz*energy_group ; i++){
         SOURCE[i] = 0.0;
         EsSum[i] = 0.0;
    }
    for (group = 0 ; group < energy_group ; group++){
```

```
    for (i = 0 ; i < matcount*nz ; i++){
      SOURCETemp = 0.0;
      EsTemp = 0.0;
      for (k = 0 ; k < energy_group ; k++){
        locateij   = ij2t (group,k,energy_group,energy_group);
        locateijk  = ij2t (i,locateij,matcount*nz,energy_group*energy_group);
        fluxlocate = ij2t (k,i,energy_group,matcount*nz);
        SOURCETemp = SOURCETemp + XvEfn[locateijk]*XOLD[fluxlocate];

        if (group != k) {
            maceslocate = ijk2t (k,group,i,energy_group,energy_group,matcount*nz);
            EsTemp = EsTemp + MACEs[maceslocate]*XOLD[fluxlocate];

        } else maceslocate = 19191999;

      }
      locateij = ij2t(group,i,energy_group,matcount*nz);
      maclocate = ij2t (i,group,matcount*nz,energy_group);
      SOURCE[locateij] = SOURCETemp;
      EsSum [locateij] = EsTemp;
      YOLD[locateij] = -1/DIFFCO1[maclocate]*((SOURCE[locateij]*(1/keff0)+EsSum[locateij]));
    }
    for (i = 0 ; i < matcount*nz ; i++){
        locateij = ij2t (group,i,energy_group,matcount*nz);
    }
    index = 0;
    for (i = 0; i < matcount*nz; i++){
      for (j = 0 ; j < 7 ; j++){
        abuff[index] = MATRIX[i][j][group];
        index++;
      }
    }

    gauss(abuff,XOLD,YOLD,YNEW,matcount,nx,ny,nz,group,Ferr_criteria );

   for (i = 0 ; i < matcount*nz ; i++)
     {
        locateij = ij2t (group,i,energy_group,matcount*nz);
        XOLD[locateij] = XNEW[locateij];
     }
   }
   FF = integral (XNEW , matcount , energy_group , nz , dx , dy , dz);
  keff1 = FF/FI*keff0;
  kerr = fabs (keff1-keff0);
  keff0 = keff1;
  for (i = 0; i < matcount*nz*energy_group ; i++){
    XOLD[i] = XNEW[i];
  }
  loop_count++;

  } while ((kerr > Kerr_criteria)&&(loop_count < 300));
  printf  (      "%3d  KEFF = %12.6f   kerr = %12.6f   \n",loop_count+1, keff1,kerr);

  free (abuff);
  free (EsSum);

  return (keff1);
}
```

ประวัติผู้เขียนวิทยานิพนธ์

นายธนรรจน์ แสงจันทร์ เกิดวันที่ 26 สิงหาคม พ.ศ. 2517 สำเร็จการศึกษาจากมหาวิทยาลัยเทคโนโลยีพระจอมเกล้าธนบุรี คณะวิศวกรรมศาสตร์ สาขาวิชาวิศวกรรมอิเล็กทรอนิกส์และโทรคมนาคม เมื่อปี พ.ศ. 2540 และเข้าศึกษาต่อภาควิชานิวเคลียร์เทคโนโลยี จุฬาลงกรณ์มหาวิทยาลัยในปีการศึกษา 2549

ประวัติการทำงาน เข้ารับราชการที่สำนักงานปรมาณูเพื่อสันติในตำแหน่งวิศวกรนิวเคลียร์ ปี 2540 หน้าที่รับผิดชอบปฏิบัติการเดินเครื่องปฏิกรณ์นิวเคลียร์วิจัย ปปว-1/1 ในปี 2550 ได้โอนไปปฏิบัติงานที่สถาบันเทคโนโลยีนิวเคลียร์แห่งชาติ (องค์การมหาชน)