**Part1. Linear System Setup**

**1. Modify the input processing routine you wrote for HW#3 such that the 2-D geometry data specified in the above format can be properly read. Determine the node numbers and sizes, etc. which will be needed to set up the linear system of nodal balance equations. Apply the natural ordering scheme. You will need a mapping scheme to relate the mesh and the region numbers.**

이번 과제에서 다루는 문제는 이차원 문제로, 입력 파일을 처리하는 과정에 존재하는 HW 3에서 다루었던 일차원 문제와의 차이는 y축 방향으로의 변수가 추가된 것 외에는 없다. 따라서 주요 내용은 Hw 3에서 사용하였던 루틴과 동일하게 사용 가능하며, y축 방향에 해당하는 변수를 추가로 설정, 자료를 읽을 수 있게 수정하였다. 계수 행렬을 설정하는 경우, 동서남북 메쉬와 서로 연관되기 때문에 각 메쉬의 정보를 미리 저장해두어야 한다. 이를 해결하기 위해 여러 방법이 존재할 수 있지만, 가장 간단하게 풀고자 하는 원자로 내의 메쉬 수와 동일한 행렬을 설정, 매질 정보를 각 성분으로 갖도록 하였다. 매질정보는, 원하는 메쉬의 좌표 (i, j)를 알고 있다면 확인이 가능하다.

|  |
| --- |
| **read(io5, \*) ng, ncomp****allocate(comp(ncomp))** **do i=1, ncomp** **allocate(comp(i)%nusigf(ng), comp(i)%d\_coeff(ng), comp(i)%sigr(ng), comp(i)%chi(ng))** **allocate(comp(i)%sigs(ng, ng))****end do****do i=1, ncomp** **read(io5, \*)**  **do j=1, ng** **read(io5, \*) ig, comp(i)%d\_coeff(j), comp(i)%sigr(j), comp(i)%nusigf(j), comp(i)%chi(j)** **end do** **do j=1, ng** **read(io5,'(I3, 100(e13.6))') ig,(comp(i)%sigs(jg, j), jg=1, ng)** **end do** **end do** **do i=1, ncomp** **allocate(comp(i)%gb(ng), comp(i)%ge(ng))** **do j=1, ng** **igb=500; ige=0** **do k=1, ng** **if (comp(i)%sigs(k, j).ne.0) then** **if (k.lt.igb) igb=k** **if (k.gt.ige) ige=k** **end if** **end do** **comp(i)%gb(j)=igb** **comp(i)%ge(j)=ige** **if (igb.eq.500) comp(i)%gb(j)=1** **if (ige.eq.0) comp(i)%ge(j)=1** **end do** **end do** |

**2. Determine the coefficients of the nodal balance equation for each group which represents the one-group balance equation having all the sources terms on the RHS. Use only four entries for the off-diagonal entries for each node and each group which dictate coupling with the neighboring nodes. Employ the proper boundary condition at the boundary nodes. Print the linear system coefficients for the given test problem.**

이번 과제에서 다루는 문제는 이차원 평판 원자로로, 기존의 문제에서 y 축이 추가된 경우이다. 이 때 Diffusion Equation은 다음과 같이 나타난다.



이것을 수치 해석적으로 풀기 위해선 내부의 보존 법칙을 만족하는 Box Scheme을 적용하여야 한다. Box Scheme을 적용하여 X 축과 y 축에 대해 적분한 결과는 다음과 같다.



결과를 살펴보면, 대부분이 1차원 평판원자로의 문제와 동일하지만, y 축을 고려하기 때문에 남쪽과 북쪽 노드와 연결된 것을 확인할 수 있다. 연결된 노드가 늘어났기 때문에, 행렬의 비대각 성분도 늘어나게 된다. 실제로 북쪽 노드와 남쪽 노드에 해당하는 비대각 성분이 각각 한줄씩 증가하여 기존의 삼대각 행렬은 오대각 행렬이 된다. 실제 계수 행렬은 다음과 같이 표현된다.



행렬이 오대각(Penta-diagonal) 행렬로 나타나기 때문에, 실제 프로그램 내에서 구현할 경우 나머지 항은 고려하지 않아도 된다. 이는 행렬 구성 시 사용되는 메모리를 절약할 수 있게 해주며, 불필요한 계산을 줄여 계산 속도 향상을 기대할 수 있게 한다.

중성자속 벡터의 경우 위 아래로 평판의 한 줄에 해당하는 노드 수만큼 성분을 추가하여 값이 0이 되게 하였다. 이 경우 계산량이 증가한다는 단점이 있지만, 코드 작성 시 반복문 내에 조건문을 사용하지 않아도 되기 때문에, 간단한 코드로 효율적인 계산을 할 수 있다.

산란 선원은 우변에서 고려하게 된다. 이 문제에서는 upscattering이 없다 가정하였으므로, 이전 차에 구한 에너지군의 중성자속을 이용하여 우변에 더해주면 된다. 산란선원을 고려한 우변의 선원은 다음과 같이 나타난다.



각 행렬은 노드 수 \* 에너지군 수만큼 존재하지만, 각 에너지군 마다 독립적으로 선원 과 이항행렬이 존재하기 때문에, 한번에 푸는 대신 에너지군 마다 단계적으로 문제를 풀 수 있다. 이렇게 풀게 되면 푸는 과정에서 이전 에너지군의 중성자속을 가장 최신값으로 사용할 수 있기 때문에, 좀 더 빠른 수렴속도를 기대할 수 있게 된다. 문제에서 풀게되는 행렬식은 다음과 같다.



다음은 실제 계수 행렬을 작성하기 위한 코드의 일부분이다. 북쪽 외의 나머지 방향은 경계에 해당하는 노드 번호만 다를 뿐 실제 수행되는 계산은 동일하다. 우변의 경우 중성자속이 갱신될 때마다 변하기 때문에, 실제 문제를 푸는 부분에서 정의하였다.

|  |
| --- |
| **!Center** **do ig=1, ng** **do i=1, tn** **mig\_mat(3, i, ig)=comp(ntype(i))%sigr(ig)** **end do** **end do** **!Relation with North** **do ig=1, ng** **betal=albedo(3)/2** **do ix=1, tnmeshx !Boundary** **i=ix** **betar=comp(ntype(i))%d\_coeff(ig)/nhy(i)** **dt=2\*betal\*betar/(betal+betar)** **mig\_mat(3, i, ig)=mig\_mat(3, i, ig)+dt/nhy(i)** **end do** **do iy=2, tnmeshy !Innermesh** **do ix=1, tnmeshx** **i=(iy-1)\*tnmeshx+ix** **il=(iy-2)\*tnmeshx+ix** **betal=comp(ntype(il))%d\_coeff(ig)/nhy(il)** **betar=comp(ntype(i))%d\_coeff(ig)/nhy(i)** **dt=2\*betal\*betar/(betal+betar)** **mig\_mat(1, i, ig)=-dt/nhy(i)** **mig\_mat(3, i, ig)=mig\_mat(3, i, ig)+dt/nhy(i)** **end do** **end do** **end do** |

**Part2. Point and Line SOR, and Spectral Radius**

**1. Write a routine that performs pointwise SOR iteration to solve each one-group problem. Suppose that the source is known and the coefficient matrix has one diagonal entry and 4 off-diagonal entries. Test this routine for the first group with uniform source density of 1 n/sec/cm3 with  and , respectively. Determine the number of iteration to reach the residual reduction of 0.001 for the each cases to compare the two iteration numbers.**

이차원 문제의 경우 비대각 성분이 늘어났으며, 기존의 대각 성분에 인접해 있지 않아, 이를 LU Factorization 하는 경우 기존에 0이 었던 성분이 0이 아닌 값을 같게 된다(Fill in). 따라서 LU Factorization 을 이용하여 반복해법을 수행하는 것은 비효율적이다. 이 때 이용할 수 있는 방법이 Jacobi, Gauss-Seidel, Jacobi 등과 같은 간접해법이다. 이 중 Jacobi는 계산을 수행하면서 한 번 반복계산을 수행하는 동안 전차의 결과만을 이용하고 계산이 끝나고 결과를 갱신하는 방법이다. 반면 Gauss-Seidel은 해당 차에 계산한 결과가 있다면, 그 결과를 다음 해를 구하는 과정에 사용한다. 즉 Jacobi 와 다르게 계산을 수행해 나가면서 차근차근 결과를 갱신한다. SOR 은 Gauss-Seidel 로 얻은 결과를 바로 갱신하지 않고, 가중치를 이용하여 일종의 Interpolation 된 결과로 갱신하는 것이다. Point 계산은 매 노드마다 각각 계산을 수행하는 것을 의미한다. 관련 식은 다음과 같다.



다음은 실제 Point SOR 을 수행하는 코드이다.

|  |
| --- |
| **phi\_ex=phi** **do i=1, tn** **!Point GS** **phi\_gs(i, ig)=1/mig\_mat(3,i,ig)\*(b(i,ig)-mig\_mat(1,i,ig)\*phi(i-tnmeshx,ig)-mig\_mat(2,i,ig)\*phi(i-1,ig)** **+ -mig\_mat(4,i,ig)\*phi\_ex(i+1,ig)-mig\_mat(5,i,ig)\*phi\_ex(i+tnmeshx,ig))** **!SOR Iterative** **phi(i,ig)=w\*phi\_gs(i,ig)+(1-w)\*phi\_ex(i,ig)** **end do**  |

수렴 조건으로는 Reduction Error을 이용하였으며, 수렴 조건으로는 문제에 주어진 0.001을 이용하였다. 가중치에 따른 수렴 횟수는 다음과 같다.

|  |  |  |
| --- | --- | --- |
| Group | W | Iteration Number |
| Group 1 | 1 | 310 |
| 1.25 | 187 |
| Group 2 | 1 | 44 |
| 1.25 | 27 |

결과를 보면 Group에 무관하게 w가 1.25일 때 적은 계산횟수로 수렴하는 것을 확인할 수 있다. 이는 SOR 자체가 Gauss-Seidel의 결과를 가중 평균하여 보다 정해에 가깝도록 추정하는 것이기에 당연한 결과이다.

Group 별로 살펴보면 Group 1의 경우 Group 2보다 계산 횟수가 많다. 이는 Group 1에 해당하는 계수 행렬에 비해 Group 2의 계수 행렬의 대각 성분이 비 대각 성분에 비해 큰 값을 가지고 있기 때문이다(Diagonally dominant). 따라서 Dominance ratio가 작고, 수렴속도가 빠르다.

**2. Write a routine that performs Line SOR iteration to solve for each one-group problem. Take each row of nodes as a line. Use the LU factorization and substitution routines you wrote for HW#2 to solve each line problem. Suppose that the north and south coupling terms are moved to the RHS and thus the source term is known and the coefficient matrix of each block has one diagonal entry and 2 off-diagonal entries, namely, west and east coupling. Perform the same test and 2.1 and compare with the point SOR results.**

Point SOR에 비해 Line SOR은 한 줄(x축 기준)에 해당하는 노드를 한 블록으로 묶어 한꺼번에 계산하는 방법이다. 관련 식은 다음과 같다.



비대각 성분 중 북쪽 및 남쪽 노드와 관련된 성분은 우변으로 넘겨 선원 항으로 정리한다. 이 과정에서 북쪽 노드의 중성자속은 앞 단계의 계산 과정에서 이미 구한 값이 있기 때문에 갱신된 값을 사용하며, 남쪽 노드는 아직 계산을 하지 않았으므로 이전 차의 값을 이용한다. 정리하면 계수 행렬이 삼대각 행렬이 되므로, 일차원 Diffusion Equation과 동일한 형태가 되고, LU Factorization을 이용한 계산 수행이 가능해진다.

다음은 계수 행렬 중 북쪽 성분과 남쪽 성분을 제외한 후 LU 행렬을 만드는 부분이다.

|  |
| --- |
| **!LU\_Factorization** **U=mig\_mat(2:4,:,:)** **do ig=1, ng** **l(2,:,ig)=1** **do i=2, tn** **mtemp(1:2, i)=[-u(1,i,ig)/u(2,i-1,ig), 1]** **L(1, i, ig)=-mtemp(1, i)** **U(:, i, ig)=[mtemp(1, i)\*u(2,i-1,ig)+mtemp(2,i)\*u(1,i,ig),**  **+ mtemp(1, i)\*u(3,i-1,ig)+mtemp(2,i)\*u(2,i,ig), mtemp(2,i)\*u(3,i,ig)]** **if (U(1,i,ig).le.1E-15) U(1,i,ig)=0** **end do** **end do** |

이 부분은 한번 결정되면 계산 내내 변하는 일이 없기 때문에 반복문 밖에 위치한다.

다음은 생성된 L, U행렬을 이용하여 실제 Line SOR 계산을 수행하는 부분이다.

|  |
| --- |
| phi\_ex=phi  **do** iy=1, tnmeshy ! Line(or Block) GS ib=(iy-1)\*tnmeshx+1; ie=iy\*tnmeshx ! Set RHS value **do** i=ib, ie mphi(i,ig)=mig\_mat(1,i,ig)\*phi(i-tnmeshx,ig)+mig\_mat(5,i,ig)\*phi\_ex(i+tnmeshx,ig) s(i)=b(i,ig)-mphi(i,ig) **end do**  !!!!!!! Solve LU !!!!!!! !Forward substitution (Ly=s) y(ib)=s(ib) **do** i=ib+1, ie; y(i)=s(i)-l(1,i,ig)\*y(i-1); **end do**  !Backward substitution (U\*phi=y) phi\_gs(ie,ig)=y(ie)/U(2,ie,ig)  **if** (phi(ie,ig).le.1E-15) phi(ie,ig)=0 **do** i=ie-1,ib,-1 phi\_gs(i,ig)=(y(i)-U(3,i,ig)\*phi\_gs(i+1,ig))/U(2,i,ig) **if** (phi\_gs(i,ig).le.1E-15) phi\_gs(i,ig)=0 **end do** !!!!!!! End LU !!!!!!!  !SOR Iterative phi(ib:ie,ig)=w\*phi\_gs(ib:ie,ig)+(1-w)\*phi\_ex(ib:ie,ig) **end do** |

위에 제시한 코드를 이용하여 주어진 문제를 푼 결과는 다음과 같다.

|  |  |  |
| --- | --- | --- |
| Group | W | Iteration Number |
| Group 1 | 1 | 157 |
| 1.25 | 95 |
| Group 2 | 1 | 24 |
| 1.25 | 15 |

GROUP 및 가중치에 대한 경향은 Point SOR의 결과와 동일하다. 전체적인 반복 횟수가 Point SOR에 비해 줄어들었다. 이는 Point SOR이 전 부분 간접 해법을 이용하여 해를 구하는 것과 달리 큰 블록을 세분화하여 직접해법을 이용하여 해를 구하기 때문이다. 우선 Iteration 만을 가지고 따져보면 Point 방법에 비해 Line 방법이 더 효율적인 계산을 가능하게 해준다.

**3. Write a routine to perform the power iteration to determine the spectral radius of the point Gauss-Seidel iteration matrix for each group and consequently the optimum over relaxation factor (ω) for each group. Use the criteria to terminate to power iteration discussed in the class. Then determine the number of iteration needed for each to achieve the inner iteration error reduction given in the input.**

SOR 방법은 적절한 w 값을 결정하지 못하면 반복 계산 횟수가 증가하여 비효율적이다. 심한 경우 Gauss-Seidel 보다 비효율적인 경우도 존재한다. 효율적으로 SOR 방법을 이용하기 위해선 해당 문제에 맞는 적절한 가중치 (w)를 이용하여야 한다. 최적의 w를 구하기 위해서는 해당 문제의 Jacobi 혹은 Gauss-Seidel 반복 행렬의 고유치가 필요하다.



Jacobi 반복 행렬과 SOR 반복 행렬을 이용하여 위와 같은 식을 얻어 낼 수 있다. 고유치(혹은 Spectral Radius)가 작을수록 수렴 속도가 빨라지므로 최소값을 갖게 하기 위해 이라 하면 식은 다음과 같이 정리 된다.



이 중 은 2보다 큰 값을 가지므로 의 조건을 만족시키지 못한다.

따라서 가 우리가 원하는 해이며,



를 만족하기 때문에 최종적으로 다음과 같이 정리된다.



최적의 가중치를 구하기 위해선 Gauss-Seidel 반복 행렬의 고유치가 필요하다. 이를 위해서는 반복 계산을 수행해야 하지만 그 과정이 불편한 관계로 SOR에서 사용되는 Gauss-Seidel 부분을 이용하여 중성자속을 구한 후 그 오차를 이용하여 가중치를 구하였다.

 



충분한 반복 계산을 수행한 경우 오차 감소 비율은 SOR 반복 행렬의 고유치와 같다. 이를 이용하여 가중치가 정해진 경우 수렴 조건을 만족하기 위한 최소 반복 횟수를 계산을 통해 구할 수 있다. 자세한 계산 과정은 다음과 같다.





다음은 실제 최적의 가중치 및 최소 반복 횟수를 구하기 위해 사용한 코드이다. J 값에 따라 내부 계산 방법이 다르며, 방법에 해당하는 가중치를 계산할 수 있다.

|  |
| --- |
| **do** ig=1, ng n\_iter=0 pre\_g=ig **do** **while**(1) n\_iter=n\_iter+1 phi\_ex(:,ig)=phi(:,ig) rhogs\_ex=rhogs  **if** (j.eq.1) **call** psor **if** (j.eq.2) **call** lsor2  err\_phi(:,ig)=phi(1:tn,ig)-phi\_ex(1:tn,ig) norm2=norm1; norm1=0 **do** i=1, tn; norm1=norm1+err\_phi(i,ig)\*err\_phi(i,ig); **end do** norm1=dsqrt(norm1) rhogs=norm1/norm2  rel\_err=abs(rhogs-rhogs\_ex)/rhogs  **if**( rel\_err .le. 1E-6) **exit** **end do**  wopt(ig,j)=2/(1+dsqrt(1-rhogs)) min\_iter(ig,j)=log(epsin)/log(wopt(ig,j)-1) **write**(\*,\*) j, ig, wopt(ig, j), min\_iter(ig, j) **write**(io8,\*) j, ig, wopt(ig, j), min\_iter(ig, j)  **end do** |

앞에서 작성한 코드를 이용하여 가중치를 1에서 2까지 0.01씩 바꿔가며 최적의 가중치를 찾는 계산을 수행하였다. 다음은 Point SOR과 Line SOR의 결과를 그래프로 그린 것이다.

(좌 : Point SOR, 우 : Line SOR)



계산을 통해 얻은 결과와 실험을 통해 얻은 결과는 다음과 같다.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Method | Group | Optimum W(Calc.) | Iteration Number(Calc.) | Optimum W(Exam) | Iteration Number(Exam) |
| Point SOR | Group 1 | 1.7458 | 24 | 1.78 | 44 |
| Group 2 | 1.4419 | 9 | 1.44 | 15 |
| Line SOR | Group 1 | 1.6614 | 17 | 1.68 | 34 |
| Group 2 | 1.3244 | 7 | 1.32 | 13 |

최대 0.04의 오차로 실험값과 계산을 통한 추정값이 잘 일치하는 것을 볼 수 있다. 최소 반복 횟수는 다소 차이가 존재하는데 그 이유는 실제로 초기 반복 계산 과정에서는 중성자속의 오차 감속 비율이 반복 행렬의 고유치와 일치하지 않기 때문이다.

**4. Do the same estimation of number of iterations for Line SOR and compare the number of iterations of the point and Line SORs. Discuss the reasons for the reduced number of iterations for LSOR. Then determine FLOP (floating point operation) per iteration for both schemes and then compare the total amount of FLOPs required to achieve the same error reduction for the two schemes.**

최적의 가중치를 사용하였을 때, LSOR 의 반복 횟수가 PSOR 보다 적게 나타난다. 이는 LSOR이 더 효율적인 계산을 하고 있을을 의미한다. 이를 보다 정확하게 하기 위하여 실제 반복 횟수당 계산량(FLOPs)를 비교하여 보았다.

우선 작성한 코드를 통해 어림잡아 추측해 볼 수 있다. 위에 첨부한 코드를 살펴보자.

Point SOR

G.S 해 추정: (나눗셈 1회 + 곱셈 1회 + 곱셈 4회 + 뺄셈 4회) \* 총 노드 수 = 10 \* 총 노드 수

SOR 해 추정: (곱셈 2회 + 덧셈 1회 + 뺄셈 1회 (변수 설정 시 생략 가능)) \* 총 노드 수

= 3 \* 총 노드 수

총 계산량 : 13 \* 총 노드수

Line SOR

우변 설정: (곱셈 2회 + 덧셈 2회) \* 블록 내 노드 수 \* 총 블록 수 = 4 \* 총 노드 수

Forward Substitution (Ly=s): (곱셈 1회 + 덧셈 1회) \* 블록 내 노드 수 \* 총 블록 수

= 2 \* 총 노드 수

Backward Substitution (U\*phi=y): (나눗셈 1회 + 곱셈 1회 + 덧셈 1회) \* 블록 내 노드 수 \* 총 블록 수 = 3 \* 총 노드 수

SOR 해 추정: 3 \* 총 노드 수

총 계산량 : 12 \* 총 노드 수

어림 결과 한번 반복 계산 시 계산량 역시 Line SOR이 적은 것을 확인 할 수 있다. 반복 계산을 고려하면, 반복 계산 횟수가 적은 LSOR 이 더욱 효과적이다.

보다 정확히 하기 위하여 코드에 계산을 할 때마다 횟수를 체크하여 총 계산량을 측정해보았다.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Method | FLOPs perIteration | FLOPs per LUFactorization | MinimumIteration # | Total FLOPs | ComputingTime (s) |
| Point SOR | 15606+18207+2601=**36414** |  | 24+9=33 | 1201662 | 0.187 |
| Line SOR | 12903+12903+2601=**28407** | 10400+26000+5200=**41600** | 17+7=24 | 723368 | 0.125 |

FLOPs 에 적힌 수들은 차례대로 덧셈 혹은 뺄셈, 곱셈, 나눗셈 계산 횟수를 의미한다. LSOR과 PSOR 모두 나눗셈 계산량은 동일하고 LU Factorization을 수행하는 과정에서 나눗셈을 많이 하지만 이는 반복 계산에 포함되지 않으므로 큰 영향을 주지 않는다. 모든 연산의 속도가 같다고 가정하면, 총 계산량은 PSOR이 LSOR에 비해 약 1.5배 정도 많으며, 실제 계산 수행 결과 1.5 배 정도 많은 수행 시간이 소요되었다. 실험 결과 LSOR이 PSOR에 비해 1.5배 정도 효율적임을 확인하였다. (이는 문제마다 다를 수 있다.)



위의 그림은 LSOR 결과와 PSOR 결과의 상대 오차를 1차원으로 그린 것이다. 변화 경향이 존재하긴 하지만 최대값이 0.04%를 넘지 않는 것으로 볼 때 두 결과가 일치함을 확인할 수 있다. 이는 PSOR 대신 LSOR을 사용해도 동일한 해를 얻음을 의미한다.

**Part3. Inner and Outer Iteration**

**1. Implement the fission source iteration using the Chebyshev acceleration routine you wrote for the previous HW set. In order to solve each group problem by an iterative method, you now need another level of iteration which is called inner iteration while the fission source iteration which is to update the eigenvalue and the fission source is called outer iteration. Implement the Line SOR linear system solvers within the outer iteration loop. Use the estimated number of iterations determined in Part 2 as the fixed number of inner iterations for each group.**

실제로 주어진 2차원 문제를 Chebyshev 가속 방법을 적용하여 풀 수 있다. 하지만 기존에 Inner Iteration Solver로 이용하던 LU 방법은 많은 Fill in 현상으로 인해 비효율적이므로, 앞에서 수행한 SOR 방법을 Inner iteration Solver로 대체하였다. Point 와 Line 두 방법 중 Line 방법이 더 효율적임이 확인되었기 때문에 Line SOR을 Solver로 이용하였다. 참고로 Point SOR을 Solver로 이용한 경우에도 동일한 해를 얻을 수 있었음을 밝힌다.

기존에 과제 3에서 사용하였던 코드를 수정하였으며, 수정된 코드는 다음과 같다. (Inner Iteration 수행 부분)

|  |
| --- |
| ! \*\*\*\*\*\*\*\*\* Inner Iteration Start\*\*\*\*\*\*\*\*\*\* !Loop over groups **do** ig=1, ng  pre\_g=ig w=wopt2(ig,2) !Determine source at each node for the current group **do** i=1, tn gb => comp(ntype(i))%gb; ge => comp(ntype(i))%ge sigs => comp(ntype(i))%sigs chi => comp(ntype(i))%chi nume=0 **do** jg=gb(ig), ge(ig) nume=nume+sigs(jg,ig)\*phi(i,jg) **end do** s(i)=1/k\_ex\*chi(ig)\*psi(i)+nume **if** (s(i).le.1E-15) s(i)=0  **nullify**(gb, ge, sigs, chi)  **end do**  !Solve LU b(:,ig)=s(:) **call** lsor **end do** |

**2. Solve the given problem to determine the k-effective and the flux distribution. Start from uniform flux distribution. After convergence, plot the flux distributions for each group using a 3-D plot package. Plot the fission source distribution as well. If you are motivated, monitor the evolution of the shape of the fluxes during the initial stage of outer iteration by plotting the shapes at different outer iteration steps.**

|  |  |  |
| --- | --- | --- |
| Method | Multiplication Factor | Iteration Number |
| Fis Iteration(No Chebyshev) | 0.9378 | 35 |
| Chebyshev | 0.9378 | 15 |

계산 결과 Chebyshev 가속을 이용하였을 때, 이용하지 않았을 때보다 약 2.4배 정도 빠른 계산으로 동일한 해를 얻을 수 있었다.

다음은 차례대로(왼쪽->오른쪽, 위->아래) GROUP 1의 중성자속, GROUP 2의 중성자속, 핵분열 선원을 그린 것이다.



Group 1은 높은 에너지 영역, Group 2는 낮은 에너지 영역이다. 주어진 입력 자료를 고려할 때 Group 1에서는 핵분열 시 생성된 중성자(, upscattring 없음)가, Group 2에서는 산란에 의한 중성자()가 주된 선원이 된다.

먼저 Group 1의 중성자속 그림을 살펴보면 원자로의 중심에 해당하는 노드에서 중성자속이 가장 크고 외곽으로 갈수록 점점 줄어드는 것을 볼 수 있다. 이는 각 노드의 구성 물질때문이다. 중심에 있는 노드들은 핵분열 반응단면적이 크고 외곽에는 핵분열 반응단면적이 0이다. Group 1의 선원은 대부분 핵분열에 의한 것임을 고려할 때 타당한 결과이다. 또한 외곽의 경계조건은 중성자속이 0이 되는 조건이므로 위와 같은 변화를 보인다.

Group 2는 좀 더 복잡한 경향을 보인다. 우선 경계 조건에 의하여 중심에서 외곽으로 갈수록 중성자속이 감소하는 경향을 보인다. 하지만 중간에 굴곡이 존재하는데 이는 외곽 노드와 중간 노드 사이에 존재하는 물질의 Removal 반응 단면적이 크기 때문이다. 때문에 흡수 반응이나 다른 영역으로 이동하는 산란반응이 많이 발생하여, 해당 영역에서의 중성자속은 줄어들고, 인접한 영역의 중성자속은 증가한다. 실제로 산란 반응 단면적도 사이에 존재하는 물질인 물질 9번에서 가장 큰 값을 보인다.

핵분열 선원 그림을 살펴보면 전체적으로 중심에서 외곽으로 갈수록 줄어드는 것을 볼 수 있다. 이는 중성자속 그림과 연관해 보면 간단히 설명된다. GROUP 2에서 대부분의 핵분열이 발생한다. Group 2의 중성자속은 사이에 존재하는 매질에서 작게 나타나지만 그 부분의 핵분열 반응단면적이 다른 곳에 비해 크기 때문에 핵분열 선원은 큰 값을 갖는다. 중간에 비연속면이 존재하는 것은 중성자속은 연속하지만 매질이 바뀌면서 중성자속이 비연속적인 변화를 하기 때문이다.

|  |  |  |
| --- | --- | --- |
| C:\Documents and Settings\rpl\My Documents\MATLAB\phi1_00.jpg | C:\Documents and Settings\rpl\My Documents\MATLAB\phi1_01.jpg | C:\Documents and Settings\rpl\My Documents\MATLAB\phi1_02.jpg |
| C:\Documents and Settings\rpl\My Documents\MATLAB\phi1_03.jpg | C:\Documents and Settings\rpl\My Documents\MATLAB\phi1_04.jpg | C:\Documents and Settings\rpl\My Documents\MATLAB\phi1_05.jpg |
| C:\Documents and Settings\rpl\My Documents\MATLAB\phi1_06.jpg | C:\Documents and Settings\rpl\My Documents\MATLAB\phi1_10.jpg | C:\Documents and Settings\rpl\My Documents\MATLAB\phi1_15.jpg |

반복 계산에 따른 Group 1의 중성자속 변화를 관찰한 것이다. 초기에는 0에서 시작하여 처음엔 크게 변화하다가 수렴 횟수가 커질수록 작은 변화로 해에 근접해 감을 확인할 수 있다. 최종해에 근접하는 것을 볼 수 있다. Group 2의 중성자 속과 핵분열 선원의 중성자 속 역시 처음엔 크게 변화하지만 반복 횟수가 6회 이상이 될수록 해에 근접하여 큰 변화를 보이지 않는다. 큰 의미를 찾을 수 없어 자세한 그림은 생략하였다.

**3. Examine various inner iteration error reduction factors (e.g. double and halve and also 10 and 0.1 times) to see how the number of outer iterations varies. Discuss the optimum inner reduction factor in the aspect of the total amount of FLOPs considering both inner and outer iterations.**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 1E-1 | 1E-2 | 1E-3 | 1E-4 | 1E-5 |
| Inner Iteration | 6+3=9 | 12+5=17 | 18+6=24 | 24+8=32 | 30+10 |
| Outer Iteration | 28 | 19 | 15 | 15 | 14 |
| Total | 252 | 323 | 360 | 480 | 560 |

내부 반복 계산의 수렴 조건을 달리하며 동일한 문제를 풀었다. 그 결과 수렴 조건이 큰 경우, 즉 비교적 엉성하게 내부 계산을 수행한 경우 정확한 해를 구하기 위해서 외부 반복 계산을 많이 수행하였다. 그러나 일정 조건 이상 주었을 때 외부 반복 계산 횟수는 일정하였다. 이는 내부 계산에서 일정한 정확도 이상의 중성자속 값을 구할 필요가 없다는 것을 의미한다. 내부 계산시 사용되는 계산량은 앞에서 언급하였으며 외부 계산시 사용되는 계산량은 Innertaion 에 비해 그리 크지 않다(약 1.2~1.5배 정도). 따라서 Outer iteration 의 가중치는 Inner Iteration 에 비해 그리 크지 않으며, 1로 두고 계산한 경우 수렴조건이 0.1인 경우 계산량이 가장 적은 것을 확인하였다. 이는 정확한 내부 계산 수행보다 약간 엉성하게 계산하는 것이 빠른 속도를 보장함을 의미한다.

**Part4. Node Size Effect**

**1. Write an edit routine that obtains the assembly-wise flux and fission source distributions. You need to sum up the nodal volume weighted fluxes for each assembly (same composition region) and then divide by the node volume. Output the map of the assembly groupwise fluxes and the fission source.**

최종 계산 결과 및 입력 자료에서 얻은 결과를 이용하여 Assembly-wise 된 결과를 얻을 수 있다. 실제 계산 과정이 매우 간단하여 코드 삽입은 생략하였다. 다음 그림들은 Assembly-wise 된 결과이다. 차례대로 Group 1, Group 2의 Flux, Fission Source 를 나타낸다. Assembly-wise 된 결과는 각 매질에 따라 해당 값이 어떤 변화를 보이는지 간략히 보여준다.



**2. Examine the effect of the node size on the accuracy of eigenvalue and the assembly-wise fission source distribution, e. g. by halving and doubling starting from 16x16 nodes per assembly. Try to examine the true solution from the three solutions. Examine the effect on the number of inner and outer iterations as well. Try finer and coarser discretizations as much as you can.**



노드 크기를 반으로 줄이는 경우 오차의 감소율이 일정하다고 할 때 위와 같이 쓸 수 있다. 이를 이용하여 노드 크기를 달리하여 실험 한 경우 결과를 바탕으로 정해의 추정이 가능하다.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Node NumberIn Assembly | 1E-1 | 1E-2 | 1E-3 | 1E-4 | 1E-5 |
| Eigenvalue | 0.933475  | 0.935733  | 0.937738  | 0.938515  | 0.938734  |
|  |  |  | 0.8880  | 0.3865  | 0.2851  |
| True Solution |  |  | 0.9536  | 0.9390  | 0.9388  |

노드 크기가 큰 경우 오차 감소율이 크기 때문에 정확한 계산이 제대로 수행되지 않았을 것으로 추정된다. 따라서 추정한 정해 역시 실제 값과 많이 다를 것이다. 반면 노드 크기가 작아질수록 정밀한 계산이 가능하고, 오차 감소율은 일정한 값에 수렴한다. 이를 바탕으로 정해를 추정한 경우 앞서의 결과보다 더 정확할 것이라 예상할 수 있다.

**3. Discuss all the important points you have learned in this homework.**

Point SOR 에 비해 Line SOR이 계산량도 적고 반복 횟수도 적어 더 효율적인 계산 수행이 가능하게 해주는 것을 확인하였다. 또한 2차원 문제에서 많은 Fill-in 현상 때문에 직접 해법을 이용하기 곤란한 경우 SOR 방법 등의 간접 해법을 이용하여 효과적으로 해를 구할 수 있음을 확인하였다. SOR 방법의 경우 w에 따라 영향을 많이 받기 때문에 적절한 w 값을 사전에 결정하는 것이 효율적인 계산을 위해서 반드시 필요하다.

내부 수렴조건을 엄격히 설정한 경우 정확한 중성자속 추정이 가능하지만, 계산량이 증가한다. 내부 수렴조건을 조금 엉성하게 설정한 경우 중성자속 추정은 덜 정확하지만, 계산량이 줄어든다. 이를 바탕으로 적절한 내부 수렴조건을 설정하여 효율적인 계산을 가능하도록 하여야 한다.

노드 크기를 달리하여 계산한 결과가 존재할 때, 이들의 오차 감소율을 이용하여 정해를 추정할 수 있다. 그러나 노드 크기가 터무니없이 큰 경우 추정한 정해는 정확하지 못하다.

**HW#4:**

**2-D 2-G Eigenvalue Problem with**

**Iterative Solution**

과 목 명 : 원자로 수치해석과 설계

제출일자 : 2010.6.23

학 번 : 2006-12015

이 름 : 임 창 현