**Part 1. Discretization and Direct Solution**

**1. Write an input processing routine which reads all the parameters given above. The input parameters are given in pair or triplet in the order shown above. The cross sections for each composition are given in one line. As the first trial input, use the following numbers.**

주어진 입력 파일의 자료를 변수에 대입하기 위한 서브루틴을 작성하였다. 이 서브루틴은 크게 두 가지 일을 수행한다. 하나는 입력한 명령어를 분석하여 입력파일명을 결정하고, Wielandt shift방법을 이용한 가속 여부를 결정한다. 명령 입력은 다음과 같다.

|  |
| --- |
| Command 'Inputfile(Optional)' 'δk(Optional)'  ex) senp.exe nd1d\_0.inp 0.05 |

프로그램을 실행 시 원하는 입력파일명을 명령어 뒤에 입력해주면 서브루틴에서 입력파일을 열어 준다. 단 입력파일의 확장자는 반드시 ‘inp’ 이어야 한다. 입력 파일을 연 후 한 줄씩 자료를 읽게 된다. 우선 자료의 타이틀을 읽은 후 그에 맞게 해당하는 변수에 주어진 값을 대입해준다. 이를 위해서 case 문을 이용하였다. 문제에 주어진 자료는 조성이 하나이고, 도메인의 타입이 ‘11111'인 입력파일로 저장을 하였고, 프로그램 실행 시 입력파일명이 입력되지 않을 때 사용할 기본 입력파일로 설정하였다.

δk를 입력해주면 δk만큼 이동한 Wielandt shift를 이용하여 계산을 수행하게 된다. 파일명을 입력하지 않고 숫자를 입력하게 되면 기본 입력파일을 가지고 Wielandt shift 계산을 수행한다.

|  |
| --- |
| **...**  **if** (icount .ge. 1) **then**;  **call** getarg(1, temp)  inputfile=temp  temp=adjustr(temp); ext=temp(298:300)  **if** (ext .ne. 'inp') **then**  inputfile='nd1d\_0.inp' ! read Default Input  **read**(temp, \*) del\_k; ws=1;  **end if**  **if** (icount .eq. 2) **then**  **call** getarg(2, temp); ws=1;  **read**(temp, \*) del\_k  **end if**  **else if** (icount .eq. 0) **then**  inputfile='nd1d\_0.inp' ! read Default Input  **else**; **stop** 'input is invalid'  **end if**  **...**  **select case**(trim(blockname))  **case**('REGION'); **read**(io8, \*) m, msize  **case**('COMPOSITION')  **read**(io8, \*) itype  **allocate**(typedata(3,itype))  **do** i=1,itype  **read**(io8, \*) (typedata(j, i), j=1,3)  **end do**  **case**('TYPE')  **allocate**(mtype(m))  **read**(io8, \*) temp  **read**(temp, '(100i1)') (mtype(j), j=1, m)  **case**('BOUND\_COND'); **read**(io8, \*) (albedo(j), j=1,2)  **case**('NODE'); **read**(io8, \*) n  **case**('CONV\_CRT'); **read**(io8, \*) conv\_k, conv\_f  **case default**  **write**(\*, '(a128)') 'Blockname' //blockname// 'Not Allowed!!'  **end select**  **...** |

**2. Derive the nodal balance equation using the box scheme for the interior node and for the left and right boundary nodes.**

뒷장에 첨부

**3. Write a routine that sets up a tridiagonal linear system which incorporates the N nodal balance equations. Treat the fission source term on the RHS a known term which is to be determined from the previous outer iteration.**

1차원 평판원자로에 대한 box scheme을 적용한 nodal balance equation은 다음과 같다.



위의 nodal balance equation을 풀기 위하여  형태로 행렬 및 벡터를 설정하는 서브루틴을 작성하였다. 행렬 및 벡터를 구성하는데 필요한 자료들은 1번의 입력 서브루틴에서 입력파일로부터 얻을 수 있다. 평판 원자로가 총 N개의 노드로 이루어져 있을 때, 행렬 M은 N by N의 크기를 갖는다. 행렬 M은 중성자의 이동에 관련된 항들을 포함하는데, 행과 노드가 대응하여 행렬을 구성하게 된다. 행렬의 대각 성분에는 해당 노드의 flux에 대한 항이, 대각 성분의 양 옆에는 인접한 노드의 flux에 대한 항이 들어가게 된다. 양 끝의 노드를 제외한 모든 성분이 동일한 규칙을 갖기 때문에 반복문을 이용하여 쉽게 행렬을 작성할 수 있다. 양 끝의 노드에 해당하는 성분은 반복문 밖에서 작성하도록 하였다.

행렬 F는 핵분열에 의해 생성되는 중성자에 관한 항을 성분으로 가지고 있다. 이는 해당 노드의 flux에만 영향을 받으므로 대각 성분만을 갖는 행렬이 된다. 이 역시 반복문을 이용하여 간단하게 입력해주었다. 벡터 (N by 1)는 구하고자 하는 벡터로 초기치는 알 수 없다. 그러나 반복계산을 하는데 반드시 필요한 값이기 때문에 초기치를 1로 설정하였다.

**4. Write a routine that performs LU factorization of the tridiagonal linear system.**

선형방정식을 풀기 위해선 행렬 M의 역행렬을 구해야 한다. 하지만 역행렬을 구하는 일이 쉽지 않고, 구하더라도 0이 아닌 성분이 많아지기 때문에 효율적인 계산이 어렵게 된다. 따라서 LU factorization을 이용하여 보다 빠른 계산을 하도록 한다.

****

행렬 A가 주어졌을 때 LU factorization을 통해서 상대각 행렬(U)와 하대각 행렬(L)의 곱으로 표현 할 수 있다. 하대각 행렬은 행렬 A의 하대각 성분을 소거하기 위한 계수들을 성분으로 가지며, 상대각 행렬은 행렬 A의 하대각 성분들을 소거하고 최종적으로 남은 항들을 성분으로 갖는 행렬이 된다.

계수 행렬 은 행렬 A의 n열 하대각 성분을 소거하는 계수를 성분으로 가지며, 다음과 같은 특성이 있다.





계수 행렬의 역행렬은 계수 행렬에 음수를 취한 결과와 같다. 또한 계수 행렬의 곱은 하나의 계수 행렬에 나머지 계수 행렬의 0이 아닌 성분을 채워넣는 것과 같다. 이를 이용하여 역행렬과 행렬곱을 수행하지 않고 행렬 L을 구하도록 서브루틴을 작성하였다.

|  |
| --- |
| **…**  **do** j=1, n; l(j, j)=1; **end do**    **do** i=1, n-1    ! initialize matrix  mtemp=0;  **do** j=1, n; mtemp(j, j)=1; **end do**    **do** j=i+1, n  inv\_diag=1/u(i, i)  mtemp(j, i)=-u(j, i)\*inv\_diag  l(j,i)=-mtemp(j,i)  **end do**  utemp=u  **call** multi\_mat(mtemp, utemp, u)    **end do**  **…** |

**5. Write a routine that solves the tridiagonal linear system by forward and backward substitutions. Then solve the linear system with the unity initial guess of flux and eigenvalue that determines the RHS.**

문제 4의 서브루틴을 이용하여 구한 행렬 L과 U는 다음과 같다. 이번에 다루는 행렬이 삼대각 행렬이므로, 행렬 L과 U는 대각 성분 및 대각 성분에 바로 인접한 성분만 0이 아닌 값을 갖는다.



위의 두 행렬을 이용하여 역행렬을 구하지 않고, 두 차례의 선형방정식을 풀어 결과를 얻을 수 있다.





우변의 벡터 b는 이전 단계에서 구한 플럭스 및 핵분열 반응단면적을 알기 때문에 정해진 값이다. 만약 처음 계산을 수행한다면 플럭스는 초기 추정값을 대입하면 된다. 행렬식을 방정식으로 표현하게 되면 총 N개의 연립 일차 방정식이 나온다. 첫 번째 행에 해당하는 일차 방정식은 미지수가 하나이므로() 바로 해를 구할 수 있다. 그 외의 방정식은 위의 행에 해당하는 방정식의 해를 필요로 한다. 첫 방정식의 해를 알고 있으므로 두번째 방정식의 해()를 얻을 수 있고, 이를 반복하면 모든 방정식의 해를 구하여 벡터 y를 얻을 수 있다. 첫번째 행부터 계산해 내려가기 때문에 Forward substitution이라 한다.



위의 계산과 동일한 과정이다. 벡터 b대신 y를 알고 있고, 행렬 U는 상대각 행렬이기 때문에, 마지막 행에 해당하는 방정식부터 해를 구하게 된다. 진행방향을 제외한 구체적인 방법은 위와 동일하다. 마지막 행부터 올라가며 해를 구하기 때문에 Backward substitution이라 한다.

|  |
| --- |
| ...  ! Forward substitution (Ly=b)    y(1,1)=b(1,1)    **do** i=2,n; y(i,1)=b(i,1)-L(i, i-1)\*y(i-1,1); **end do**    ! Backward substitution (Uphi=y)    phi(n,1)=y(n,1)/U(n, n)    **do** i=n-1,1,-1; phi(i,1)=(y(i,1)-U(i, i+1)\*phi(i+1,1))/U(i,i); end do  ... |

**Part 2. Power Iteration and Wielandt Shift**

**1. Write an outer iteration routine that performs the power iteration. The outer iteration routine needs to setup the RHS and then to call the linear system solution routine you wrote above. The new solution should be used to update the eigenvalue. The outer iteration should be terminated when both convergence criteria are met.**



Power iteration을 통해 위의 문제를 풀게 되는데 이는 outer iteration과 inner iteration의 두 단계로 수행된다. Outer iteration에서는 전 단계에서 구한 플럭스를 이용하여 우변을 업데이트한다. 이 후 위의 문제 5에서 작성한 서브루틴을 호출하여 선형방정식을 풀어 벡터 를 얻는다(Inner iteration). 이 과정에서 행렬 M은 계산을 반복하더라도 변하지 않는다. 따라서 반복문 밖에서 LU factorization을 수행하여 행렬 L 및 U를 구하여도 된다. Inner iteration을 마치면 플럭스 자료가 갱신된다. 갱신된 플럭스 및 전단계의 플럭스 값을 비교하여 그 단계에 해당하는 고유치를 계산할 수 있다.



모든 계산을 마친 후 고유치 및 의 상대 오차를 추가로 계산하여 두 값이 주어진 수렴조건을 만족할 때 Outer iteration은 종료된다. 모든 계산이 종료되면 반복 횟수, Dominance ratio, 고유치의 최댓값 및 정규화된 플럭스를 출력파일에 저장한다.

|  |
| --- |
| **…**  **do** **while**(.true.) !Outer Iteration    n\_iter=n\_iter+1    !setup the RHS  **call** multi\_mat(FM, phi\_ex, psi)  b=psi/k\_ex    !Solve LU (Inner Iteration)  **call** solve\_LU(L, U, phi, b)  !Multiplication factor  nume=0; deno=0  **do** i=1, tn; nume=nume+psi(i, 1)\*psi(i, 1); **end do**  **do** i=1, tn; deno=deno+psi(i, 1)\*psi\_ex(i, 1); **end do**  k\_eff=k\_ex\*dsqrt(nume/deno)  **…**  ! Terminate condition  **if** ( (abs(err\_k).le.conv\_k).and.(conv\_psi.le.conv\_f) ) **exit**    **end do**  **…** |

**2. Obtain the power iteration for the uniform composition case and compare the eigenvalue error with the one that can be analytically estimated. Confirm the dependence of the eigenvalue error on the node size.**

위에서 작성한 서브루틴을 이용하여 문제에서 주어진 조성에 대하여 선형계를 풀었다. 그 결과는 다음과 같다.

D:\Study\RNAD\Hw2\results\nd1d_0.emf

|  |  |
| --- | --- |
| **Reference Eigenvalue** | **1.039103E+00** |
| **Eigenvalue** | **1.039104E+00** |
| **Error of Eigenvalue** | **-1.309438E-06** |
| **Dominance ratio** | **9.807868E-01** |
| **Iteration** | **336** |

고유치의 참조값은 주어진 조성의 자료를 이용하여 이론식에 대입한 결과이다. 살펴보면 고유치는 참조값에 거의 근접하게 나타났다. 왼쪽 경계는 반사 경계로 설정되어있기 때문에 왼쪽을 대칭으로 하는 모양이 나타났다. 전체적으로 조성이 같기 때문에 플럭스의 변화는 완만하게 나타난다.

D:\Study\RNAD\Hw2\change.emf

|  |  |
| --- | --- |
|  |  |
| **20** | **2.272854E-09** |
| **25** | **7.959274E-08** |
| **37.5** | **2.750873E-07** |
| **50** | **5.178532E-07** |

Hw #1에서 고유치의 오차는 메쉬 크기의 제곱에 비례하는 것을 확인하였다. 이번에는 Power iteration을 통해 얻은 고유치를 참조값과 비교하여 오차의 메쉬 크기에 대한 의존성이 동일하게 존재하는지 확인하였다. 위의 그래프에서 x축은 메쉬 크기의 제곱에 해당하고, y축은 고유치의 오차를 나타낸다. 그래프의 개형이 직선으로 나타나므로 여전히 오차의 메쉬 크기에 대한 의존성은 동일하게 존재하고 있다고 할 수 있다.

**3. Modify the outer iteration so that the Wielandt shift method can be employed. Add another input line to specify the first step to begin the Wielandt shift and the value of the shift (e.g. 5 and 0.05). Incorporate the Wielandt shift accordingly.**

명령어에 δk 값이 입력되었을 때 Power iteration을 수행하는 서브루틴 대신 Wielandt shift를 수행하는 서브루틴이 호출되도록 설정하였다. Wielandt shift를 이용할 경우 고유치의 이동량만큼 Dominance ratio가 작아지기 때문에 이전보다 빠른속도로 원하는 값에 수렴할 것을 기대할 수 있다. Wielandt shift를 이용하는 경우 풀어야 하는 선형방정식은 다음과 같다.



최종적으로 풀어야하는 선형계의 형태는 Power iteration의 경우와 같으며, 행렬 와 벡터 를 계산해주면, 나머지는 Power iteration의 계산식을 그대로 이용할 수 있다. 단 Wielandt shift를 이용하는 경우 선형계의 좌변에 존재하는 행렬이 계속 바뀌기 때문에 매번 LU factorization을 수행해주어야 한고, 고유치를 이동시켜 계산하였기 때문에 Power iteration과는 다른 방법으로 고유치를 계산하여야 한다.



|  |
| --- |
| **…**  k\_s=k\_ex+del\_k  M\_s=MM-FM/k\_s    !LU Factorization  **call** set\_LU(M\_s, L, U)    !Setup the RHS  **call** multi\_mat(FM, phi\_ex, psi)  b=psi/k\_ex-psi/k\_s    **call** solve\_LU(L, U, phi, b)  **…** |

**4. Compare the performance of the power iteration and Wielandt shift based on the consideration of dominance ratio.**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 0.05 | 0.1 | 0.5 | 1 | 5 | 10 |
|  | 336 | 26 | 45 | 131 | 185 | 287 | 336 |
|  | 1.039104E+00 | 1.039104E+00 | 1.039104E+00 | 1.039104E+00 | 1.039104E+00 | 1.039104E+00 | 1.039104E+00 |
|  | 9.807868E-01 | 7.009142E-01 | 8.175638E-01 | 9.431285E-01 | 9.615891E-01 | 9.768861E-01 | 9.807868E-01 |
|  | - | 7.009173E-01 | 8.175641E-01 | 9.431286E-01 | 9.615891E-01 | 9.768862E-01 | 9.788326E-01 |

Wielandt shift 이용의 장점은 무엇보다 수렴속도가 빨라지는데 있다. 이를 확인하기 위하여 고유치의 증가분()을 0.05부터 5까지 주어가며 계산을 해보았다. 고유치의 증가분이 0인 자료는 Wielandt shift를 이용하지 않은 결과이다. 먼저 반복 횟수를 통해 고유치의 변화가 작을 수록 수렴속도가 빨라지는 것을 확인할 수 있었다. 단 변화가 너무 작아 실제 고유치보다 변화된 고유치가 작아지면() 제대로 된 계산이 수행되지 않았다. 반대로 고유치의 증가분이 고유치에 비해 매우 클 때, 수렴은 하였지만 가속효과가 전혀 나타나지 않은 것을 확인할 수 있었다. 따라서 Wielandt shift를 효율적으로 사용하기 위해서는 가 적절한 값을 가지도록 설정해주어야 한다.

Dominance ratio는 반복계산 시 수렴 속도를 의미한다. 이 값이 낮을수록 적은 반복 횟수로도 정확한 결과를 얻을 수 있으며, 1에 가까운 값을 가질수록 수렴속도가 늦어진다. 이는 결과를 통해서도 나타나는데, 가 작을수록 Dominance ratio는 작은 값을 가지고 있고, 반복 횟수 역시 적다. 하지만 고유치는 동일하게 나타난다.



Power iteration을 수행한 결과를 통해 Wielandt shift시 dominance ratio를 추정하여 그 수렴속도를 예측해볼 수 있다. 실제 수치 계산 수행 결과, 예측 값과 거의 동일한 결과를 얻을 수 있었다.

Wielandt shift 의 사용 유무에 따른 플럭스를 비교해보았다. Wielandt shift를 사용한 자료는 를 0.05로 두었을 때의 결과를 이용하였다. 그래프 상으로도 거의 겹쳐져서 나타나고 있으며 각 노드 별 상대 오차를 살펴보아도 최대값이 0.2%를 안넘는 것을 볼 수 있다. 즉, Wielandt shift의 사용은 계산 결과의 정확성을 감소시키지 않는다는 결론을 내릴 수 있다.

D:\Study\RNAD\Hw2\MATLAB\plot\comp.emf

**5. Run the problem for the input sets given in the files in the class bulletin and plot the flux distribution in each case.**

|  |  |  |
| --- | --- | --- |
| Input file | Nd1d\_1.inp | 옆의 그림은 해당 입력파일 및 조성, 그리고 그에 해당하는 정규화된 플럭스를 그린 것이다. 경계 조건을 살펴보면 양쪽 경계에서 모두 플럭스가 0이 되는데 양쪽의 플럭스값이 0을 향해 가는 것을 통해 올바르게 나타났음을 알 수 있다. 플럭스 모양을 살펴보면 양쪽 가장자리에서 기울기의 변화가 나타나는 곳이 있는데, 이는 조성의 변화가 발생하였기 때문이다. 조성(2) 구간에서는 핵분열이 발생하여 중성자가 증가하기 때문에 플럭스가 크게 나타난다. |
| Type | 12221 |
| Bound condition | 1.e30 1.e30 |
| D:\Study\RNAD\Hw2\results\nd1d_1.emf | |
| Input file | Nd1d\_2.inp | 2번 입력 파일의 경우 양쪽 경계 조건이 다르게 주어졌다. 왼쪽 경계조건은 반사계수가 무한대, 즉 플럭스가 0이고, 오른쪽 경계에서는 플럭스가 매우 크거나, 변화가 없는 반사 경계가 되도록 조건이 주어졌다. 이 경우에는 반사 경계보다는 플럭스의 값이 경계에서 매우 크게 나타나고 있다. 조성을 살펴보면 왼쪽 가장자리 부근에서 핵분열이 발생하지 않는 구간이 존재한다.(조성 1) 하지만 플럭스 자체가 0에 수렴하고 있기 때문에 그 변화가 뚜렷하게 나타나지 않고 있다. |
| Type | 12222 |
| Bound condition | 1.E30 0 |
| D:\Study\RNAD\Hw2\results\nd1d_1.emf | |
| Input file | Nd1d\_3.inp | 양 쪽의 경계가 반사경계로 설정되어 있으며, 실제로 적절하게 나타나고 있다. 조성 2보다, 조성 3에서 흡수단면적이 크기에, 중간 영역에서 소멸되는 중성자가 많다. 따라서 중간 영역의 플럭스가 작게 나타난다. 전체 영역에 걸쳐 핵분열이 발생하기 때문에 플럭스에는 별 영향을 끼치지 못한다고 생각된다. |
| Type | 23332 |
| Bound condition | 0 0 |
| D:\Study\RNAD\Hw2\results\nd1d_1.emf | |
| Input file | Nd1d\_4.inp | 조성 1은 흡수 단면적이 작은 대신, 핵분열 반응도 발생하지 않는다. 때문에 전체적인 중성자 수가 적다. 경계조건은 양쪽 경계에서 플럭스가 0이다. |
| Type | 121 |
| Bound condition | 1.E30 1.E30 |
| D:\Study\RNAD\Hw2\results\nd1d_1.emf | |
| Input file | Nd1d\_5inp | 경계 조건은 양 쪽에서 플럭스가 0으로, 제대로 나타나고 있다. 양 끝의 조성은 핵분열 단면적이 0으로, 중성자가 생성되지 않고, 중간 영역에서만 핵분열이 발생한다. 정 가운데는 주변보다 흡수 단면적이 높아 중성자의 소멸이 많이 발생한다. |
| Type | 12321 |
| Bound condition | 1.E30 1.E30 |
| D:\Study\RNAD\Hw2\results\nd1d_1.emf | |
| Input file | Nd1d\_6.inp | 조성 4의 흡수단면적이 다른 조성에 비해 5배 정도 크다. 때문에 오른쪽 영역에는 중성자가 흡수 되어 거의 존재하지 않으며 따라서 중성자의 흐름도 거의 없다. 조성이 2에서 3으로 변하는 30번 노드에서 플럭스의 기울기가 변하는데 이는 흡수단면적의 변화 때문에 발생한 것이다. 양 쪽의 경계 조건은 0.5로 동일한데 이는 외부가 진공 상태로 외부에서의 중성자 유입이 없다는 것을 의미한다. |
| Type | 2223324444 |
| Bound condition | 0.5 0.5 |
| D:\Study\RNAD\Hw2\results\nd1d_1.emf | |
| Input file | Nd1d\_7.inp | 플럭스의 모양이 언뜻 ‘Nd1d\_1.inp’의 경우와 유사하다. 열번째 노드에서 불연속점이 나타나는데 이는 조성이 바뀌기 때문이다. 조성 1에서는 핵분열 반응도 발생하지 않고, 확산 계수도 0.1로 크게 감소하기 때문에 이 구간에서는 중성자의 흐름이 거의 존재하지 않을 것이다. 따라서 조성이 바뀌는 경계에서 중성자의 흐름이 극대화되고 불연속점이 나타나게 된다. |
| Type | 12222 |
| Bound condition | 1.E30 1.E30 |
| D:\Study\RNAD\Hw2\results\nd1d_1.emf | |

**HW#2:**

**1-D, 1-G Neutron Diffusion Eigvenvalue Problem**

과 목 명 : 원자로 수치해석과 설계

제출일자 : 2010.03.26

학 번 : 2006-12015

이 름 : 임 창 현