KAERI/TR-3637/2008

기술보고서

1 차원 쌍곡선 방정식을 이용한 이상유동해석 코드 개발

Development of One Dimensional Hyperbolic Coupled Solver for Two-Phase Flows

2008 년 8 월

한국원자력연구원

Korea Atomic Energy Research Institute

제 출 문

한국원자력연구원장 귀하

본 보고서를 2008 년도 “고정밀 열수력 전산수치해석기술 개발” 과제의 기술보고서로 제출합니다.

제목 : 1 차원 쌍곡선 방정식을 이용한 이상유동해석 코드 개발 (Development of One Dimensional Hyperbolic Coupled Solver for Two-Phase Flows)

2008 년 8 월

과제명: 고정밀 열수력 전산수치해석기술 개발

주저자: 김의진

공저자: 김종태

정재준

1 차원 쌍곡선 방정식을 이용한 이상유동해석 코드 개발

요 약 문

본 연구는 액체/기체 혼합체의 이상유동(two-phase flows)에 관한 기초적인 연구를 수행하여 이상유동 해석에 필요한 해석코드를 개발하는데 목적이 있다. 이를 위하여 상류차분법(upwind difference scheme) 중의 하나인 Roe Scheme을 채택하였다. 압축성 기체 유동해석에서 개발된 상류차분법은 수치적 안정성, 특히 충격파와 같은 불연속면을 안정적으로 포획(capturing)하는 특성으로 인하여 전산유체역학 분야에서 많이 사용되고 있다. 그리고 Toumi[1], Stadtke[2]등은 이러한 상류차분법을 확장하여 다상유동 (multi-phase flow)에 적용하였다. 본 연구에서는 대류항의 계산을 위해 상류차분법 중 확장된 Roe Scheme을 이용하여 1차원 이상유동을 모사하였으며, 기초 연구로서 단면적이 일정한 경우의 유동만 계산할 수 있도록 하였다. 이에 따라 Roe-averaged matrix를 얻음으로써 충격파관과 수도밸브 문제의 이상유동에 관한 두 가지 예시 계산을 수행하였다. 이를 통해 일반적인 상류차분법과 마찬가지로 본 연구에서 사용한 확장된 Roe의 기법 역시 충격파와 같은 불연속면을 포획하는 특성이 있다는 것을 알 수 있다. 또한 셀 수가 적은 경우에도 기공률의 급격한 변화에 따른 진동이 일어나지 않는다는 점에서 큰 진동이 없이 불연속면의 계산이 가능한 이점이 있다는 것을 알 수 있다.

Development of One Dimensional Hyperbolic Coupled Solver for Two-Phase Flows

Summary

The purpose of this study is a code development for one dimensional two-phase two-fluid flows. In this study, the computations of two-phase flow were performed by using the Roe scheme which is one of the upwind schemes. The upwind scheme is widely used in the computational fluid dynamics because it can capture discontinuities clearly such as a shock. And this scheme is applicable to multi-phase flows by the extension methods which were developed by Toumi, Stadtke, etc. In this study, the extended Roe upwind scheme by Toumi for two-phase flow was implemented in the one-dimensional code. The scheme was applied to a shock tube problem and a water faucet problem. This numerical method seems efficient for non oscillating solutions of two phase flow problems, and also capable for capturing discontinuities.

목 차

표 목차

그림 목차

부호설명

영어 문자

U	보존량 벡터
F	플럭스 벡터
u	유동장 속도
e	내부에너지
h	엔탈피
p	압력
H	총엔탈피
E	총에너지

그리스 문자

α	기공률
ρ	밀도
Γ	질량 변화
F^w	wall drag
F^i	interphase drag force
Q^w	wall heat flux
Q^i	interphase heat exchange
B	체적력(usually gravity)
CFL	Courant-Friedrichs-Lewy Number

상첨자

+	양의 방향
−	음의 방향
n	시간단계
−1	역행렬

하첨자

v	기체
l	액체

1. 서 론

공학 시스템의 많은 분야에서 이상 혹은 다상 유동 문제가 점점 더 중요하게 여겨지고 있다. 디자인의 최적화와 기기의 안전한 사용을 위한 이상유동 문제의 해석은 산업기술뿐만 아니라 좀 더 나은 이해를 필요로 하는 자연 현상으로도 확장 될 수 있다. 이상유동 현상은 지구의 기상학적 현상, 즉 비바람, 황사 등과 같이 주변에서 널리 볼 수 있으며, 공학적으로는 발전 시스템, 공정 시스템, 수송 시스템 등 응용 범위가 넓다.

본 연구는 액체/기체 혼합체의 이상유동(two-phase flows)에 관한 기초적인 연구를 수행하여 이상유동 해석에 필요한 해석코드를 개발하는데 목적이 있다. 초기의 CFD 알고리듬은 주로 중심차분법을 이용하여 대류항을 이산화하였다. 이렇게 이산화된 시스템은 본질적으로 내재변(LHS)의 행렬이 대각우위행렬 (diagonally dominant matrix)이 아니기 때문에 Jacobi나 Gauss-Seidel등의 일반적인 완화법(relaxation method)을 사용할 경우 불안정하다. 1980년대 중반부터는 유동장의 심한 불연속 현상도 잘 예측할 수 있는 특성 때문에 상류차분법이 널리 쓰이기 시작하였다. 상류차분법은 내재하고 있는 대각우위 (diagonally dominant)특성 때문에 내재변을 역전시키기 위해서 여러 가지 근사적 방법을 사용할 수 있는 장점을 가지고 있다. 또한 압축성 기체 유동해석에서 개발된 상류차분법은 수치적 안정성 특히 충격파와 같은 불연속면을 안정적으로 포획(capturing)하는 특성으로 인하여 전산유체역학 분야에서 많이 사용되고 있다. 그리고 Toumi[1], Stadtke[2]등은 이러한 상류차분법을 확장하여 다상유동 (multi-phase flow)에 적용하였다.

본 연구에서는 대류항의 계산을 위해 상류차분법 중 확장된 Roe Scheme을 이용하여 1차원 이상유동을 모사하였으며, 기초 연구로서 단면적이 일정한 경우의 유동만 계산할 수 있도록 하였다.

본 논문의 2절에서는 two-phase two-fluid model의 지배방정식을

설명하였고, 3절에서는 수치해법을 소개하였다. 그리고 4절에서는 개발된 코드의 예시계산결과에 관해 기술하였고, 5절에 요약 및 결론을 수록하였다. 부록에는 코드의 구조도와 전체 subroutine의 list를 수록하였다.

2. 1 차원 Two-Phase Two-Fluid Model 의 지배방정식

이상유동을 모사하기 위해 1차원 two-phase two-fluid model을 고려한다. 액체의 밀도는 상수로 가정하며, 기체의 밀도는 압력과 엔탈피의 함수로 한다. 기체는 이상기체로 가정한다. 각 상은 아래 첨자로 구분하며, v, l 은 각각 기체와 액체를 나타낸다.

2.1 질량 보존방정식

Continuity equation for vapor phase

$$\frac{\partial \alpha_v \rho_v}{\partial t} + \frac{\partial \alpha_v \rho_v u_v}{\partial x} = \Gamma_v \tag{2.1}$$

Continuity equation for liquid phase

$$\frac{\partial \alpha_l \rho_l}{\partial t} + \frac{\partial \alpha_l \rho_l u_l}{\partial x} = \Gamma_l \tag{2.2}$$

2.2 운동량 보존방정식

Momentum equation for vapor phase

$$\frac{\partial \alpha_v \rho_v u_v}{\partial t} + \frac{\partial \alpha_v \rho_v u_v^2}{\partial x} + \alpha_v \frac{\partial p}{\partial x} + I = \alpha_v \rho_v B + F_v^w + F^i \tag{2.3}$$

Momentum equation for liquid phase

$$\frac{\partial \alpha_l \rho_l u_l}{\partial t} + \frac{\partial \alpha_l \rho_l u_l^2}{\partial x} + \alpha_l \frac{\partial p}{\partial x} - I = \alpha_l \rho_l B + F_l^w - F^i \tag{2.4}$$

2.3 에너지 보존방정식

Energy equation for vapor phase

$$\frac{\partial}{\partial t}\left[\alpha_v\rho_v\left(e_v+\frac{u_v^2}{2}\right)\right]+p\frac{\partial\alpha_v}{\partial t}+\frac{\partial}{\partial x}\left[\alpha_v\rho_v u_v\left(h_v+\frac{u_v^2}{2}\right)\right]=Q_v^w+Q_v^i+\Gamma_v h_v^* \quad (2.5)$$

Energy equation for liquid phase

$$\frac{\partial}{\partial t}\left[\alpha_l\rho_l\left(e_l+\frac{u_l^2}{2}\right)\right]+p\frac{\partial\alpha_l}{\partial t}+\frac{\partial}{\partial x}\left[\alpha_l\rho_l u_l\left(h_l+\frac{u_l^2}{2}\right)\right]=Q_l^w+Q_l^i+\Gamma_l h_l^* \quad (2.6)$$

잘 알려져 있듯이 위와 같은 기본 이상유동 모델의 가장 중요한 문제점은 쌍곡선 방정식의 성질을 가지지 않는다는 것이다. 따라서 기본 이상유동 모델은 복소수 고유값을 포함하며, 불량 조건의 초기치 문제가 되므로 안정적인 결과를 얻을 수 없다. 이러한 현상의 원인은 액체와 기체의 압력은 동일한 반면, 속도는 서로 불연속적인 값을 가지고 있기 때문인 것으로 알려져 있다. 따라서 이러한 역학적인 불균형을 해결하기 위하여 추가로 수치적 감쇠 항을 필요로 한다는 어려움이 있다.

이러한 보존방정식의 수학적 특성은 물리적 첨가항에 의하여 개선될 수 있음이 많은 선행 연구들에 의하여 보고된 바 있다. Ransom과 Hicks[3]는 보존 방정식에 표면장력 방정식을 추가하였고, Stuhmiller[4]는 계면압력항을 고려함으로써 방정식 계의 고유값을 실수화 하였다. 이에 따라 복소수 고유값을 가짐으로써 발생하는 수치적인 불안정성을 제거할 수 있는 가능성이 제시되었다.

기본 이상유동 모델의 운동량 보존방정식에 계면압력항(interface pressure term)을 도입하여 쌍곡선 시스템을 얻을 수 있다. 여기서 표면장력에 의한 계면압력항은 운동량 보존방정식의 다른 항들에 비해 상대적으로 작은 값을 가지지만 가상 질량항(virtual mass term)이나 생성항(source term) 등이 없이도 2유체방정식의 계를

쌍곡선형(hyperbolic)으로 만들어 줌으로써 수치적 불안정성을 제거할 수 있다. 추가되는 계면압력항은 다음과 같다.

$$I = (p - p_i)\frac{\partial \alpha_v}{\partial x} \tag{2.7}$$

여기서

$$p - p_i = \alpha_v \rho_l \delta (u_v - u_l)^2 \tag{2.8}$$

이다. δ 는 압력계수(pressure coefficient)이다. 또한 이상기체에서 총에너지 및 총엔탈피는 다음과 같다.

$$E = \frac{p}{(\gamma - 1)\rho} + \frac{u^2}{2} \tag{2.9}$$

$$H = E + \frac{p}{\rho} \tag{2.10}$$

2.4 상태 방정식

기체상의 밀도는 다음과 같이 압력과 엔탈피의 함수로 나타낸다.

$$\rho_v = \rho_v (P, h_v) \tag{2.11}$$

따라서 압력은 다음과 같은 기체상의 밀도와 엔탈피의 함수로 나타낼 수 있다.

$$P = \frac{\gamma - 1}{\gamma}\rho_v h_v \tag{2.12}$$

3. 1차원 Two-Phase Two-Fluid Model의 Roe Scheme

이상모델을 수치적으로 해석하기 위해서는 다양한 수치 기법이 사용된다. 고속의 유동장을 해석할 경우 중앙차분의 형태로 이산화된 대류항은 수치해석에 있어 불안정성을 유발하고 수치해를 크게 왜곡하는 결과를 가져올 수 있다. 이러한 현상을 방지하기 위하여 적절한 인공 점성항을 추가하거나, 물리적인 상류차분법을 사용하는 것이 일반적이다.

Van Leer의 FVS(flux vector splitting), Roe의 FDS(flux difference splitting)와 같은 상류차분법은 수치적인 안정성과 충격파와 같은 불연속면을 안정적으로 포획하는 특성으로 인하여 전산유체역학에서 많이 사용되고 있다. 그리고 Toumi[1], Stadtke[2] 등은 이러한 상류차분법을 확장하여 다상유동 (multi-phase flow)에 적용하였다.

본 연구에서는 압축성유동의 해석에 있어서 가장 일반적으로 사용되는 Roe의 flux-difference splitting(FDS) 기법을 기반으로 확장된 Roe scheme을 이용하여 1차원 이상유동을 모사하는 것을 목적으로 한다. 편미분이 포함되지 않은 소스항의 경우, 중앙 차분에 의한 방법으로 이산화하여도 좋은 해를 얻을 수 있으므로 본 절에서는 편의상 소스 항을 생략한다.

본 절에서는 대류항의 계산을 위하여 확장된 Roe scheme을 이용하여 Roe-averaged matrix를 얻는 과정을 소개하고, 이를 이용하여 이상유동모델을 모사하는 코드를 개발하고자 한다. 기초 연구로서 단면적이 일정한 파이프 내부의 유동만 모사할 수 있도록 하였다.

3.1 지배방정식

질량 보존방정식

$$\frac{\partial \alpha_k \rho_k}{\partial t} + \frac{\partial \alpha_k \rho_k u_k}{\partial x} = 0 \tag{3.1}$$

운동량 보존방정식

$$\frac{\partial \alpha_k \rho_k u_k}{\partial t} + \frac{\partial \alpha_k \rho_k u_k^2}{\partial x} + \alpha_k \frac{\partial p}{\partial x} + (p - p_i)\frac{\partial \alpha_k}{\partial x} = 0 \tag{3.2}$$

에너지 보존방정식

$$\frac{\partial}{\partial t}\left[\alpha_k \rho_k \left(e_k + \frac{u_k^2}{2}\right)\right] + p\frac{\partial \alpha_k}{\partial t} + \frac{\partial}{\partial x}\left[\alpha_k \rho_k u_k \left(h_k + \frac{u_k^2}{2}\right)\right] = 0 \tag{3.3}$$

액상에서 밀도는 상수이므로 연속방정식을 이용하여 에너지 보존방정식의 시간 미분항을 다음과 같이 공간 미분항으로 바꾸어 나타낼 수 있다.

$$p\frac{\partial \alpha_l}{\partial t} = -\frac{p}{\rho_l}\frac{\partial \alpha_l \rho_l u_l}{\partial x} \tag{3.4}$$

따라서 이상모델의 보존형 유동 변수는 다음과 같이 정의된다.

$$\mathbf{U} = \begin{pmatrix} \alpha_v \rho_v \\ \alpha_l \rho_l \\ \alpha_v \rho_v u_v \\ \alpha_l \rho_l u_l \\ \alpha_v \rho_v \left(e_v + \frac{u_v^2}{2}\right) \\ \alpha_l \rho_l \left(e_l + \frac{u_l^2}{2}\right) \end{pmatrix} \tag{3.5}$$

본 연구에서는 편의상 ρ_l는 상수로 가정하였다. ρ_l이 상수가 아닌 경우에는 다음과 같은 방법을 통하여 각 항의 분리가 가능하다. 우선 전 단계에서

계산된 유동 변수를 이용하여 u_v, u_l, e_v, e_l을 구한다. 그러면 다음과 같은 식을 얻을 수 있다.

$$\alpha_v\rho_v = \alpha_v\rho_v(p,e_v) = C_1$$
$$(1-\alpha_v)\rho_l = (1-\alpha_v)\rho_l(p,e_l) = C_2 \qquad (3.6)$$

이 식에서 e_v, e_l, C_1, C_2는 기지수이며 α_v, p는 미지수이다. 두 개의 식에 두 개의 미지수가 존재하므로 적절한 수치해법을 사용하여 두 미지수를 계산할 수 있다.

3.2 비보존형 시스템에 대한 근사 Riemann 해법

보존형 시스템에서와는 달리, 비보존형 모델에 대한 Riemann 문제의 해는 알려져 있지 않으므로 근사적으로 해를 구하는 방법을 사용한다. Riemann 문제의 형태는 다음과 같다.

$$\frac{\partial \mathbf{u}}{\partial t} + \frac{\partial \mathbf{f}(\mathbf{u})}{\partial x} = 0$$
$$\mathbf{u}(x,0) = \mathbf{u}_L \ (x<0), \quad \mathbf{u}(x,0) = \mathbf{u}_R \ (x>0) \qquad (3.7)$$

Roe의 FDS(Flux-difference splitting) 기법은 상류 차분의 효과를 충격파관(shock tube) 문제와 같은 Riemann 문제의 해석을 통하여 얻고자 하는 Godunov 방법에서 출발한다.[5] 일반적인 Godunov 방법이 비선형의 Riemann 문제를 연속적으로 해석하여 많은 계산 시간이 소요되는 점에 반하여, Roe의 FDS 기법은 비선형 방정식의 해를 선형 쌍곡선형 방정식(linear hyperbolic equations)으로 근사하여 Riemann 문제의 근사해를 구하는 방법으로 계산량이 작으면서 충분히 정확한 상류 차분의 효과를 얻을 수 있어 널리 사용되는 수치해법이다. 쌍곡형 보존

시스템에서의 비선형 Riemann 문제를 계산하기 위하여 Roe는 국소 선형화(local linearization)를 제안하였다.[6]

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{A}(\mathbf{u}_L, \mathbf{u}_R)\frac{\partial \mathbf{u}}{\partial x} = 0 \tag{3.8}$$

여기서 $\mathbf{A}(\mathbf{u}_L, \mathbf{u}_R)$ 는 각각의 $\mathbf{u}_L, \mathbf{u}_R$ 에 대해 국소적으로 상수인 선형화 행렬로 Roe-averaged matrix라고 알려져 있다. 이 행렬은 다음과 같은 성질을 만족한다.

$$\mathbf{f}(\mathbf{u}_R) - \mathbf{f}(\mathbf{u}_L) = \mathbf{A}(\mathbf{u}_L, \mathbf{u}_R)(\mathbf{u}_R - \mathbf{u}_L) \tag{3.9}$$

이러한 성질은 충격파를 정확히 감지하는데 중요한 특성으로 수치유속이 보존형이 되기 위한 조건이다. $\mathbf{A}(\mathbf{u}_L, \mathbf{u}_R)$ 행렬은 Roe에 의하여 이상기체에 대한 Euler 방정식에 처음 적용되었으며, 실제기체에 대한 방정식에 적용할 수 있도록 확장하는 방법도 고안되었다. 그러나 이상유동을 모사한 비보존형 시스템에는 Roe scheme을 적용할 수 없으므로 weak formulation을 도입하여 Riemann 문제에 대한 근사해를 구할 수 있다.[6]

$$\begin{aligned} &\frac{\partial \mathbf{u}}{\partial t} + \mathbf{A}(\mathbf{u}_L, \mathbf{u}_R)_\Phi \frac{\partial \mathbf{u}}{\partial x} = 0 \\ &\mathbf{u}(x,0) = \mathbf{u}_L \ \ (x<0), \quad \mathbf{u}(x,0) = \mathbf{u}_R \ \ (x>0) \end{aligned} \tag{3.10}$$

여기서 $\mathbf{A}(\mathbf{u}_L, \mathbf{u}_R)_\Phi$ 행렬은 $\mathbf{u}_L, \mathbf{u}_R$ 의 값과 완만한 경로(smooth path) $\Phi(s; \mathbf{u}_L, \mathbf{u}_R)$에 의존하며 다음의 성질을 만족한다.

$$\int_0^1 \mathbf{A}(\Phi(s; \mathbf{u}_L, \mathbf{u}_R))\frac{\partial \Phi}{\partial s}(s; \mathbf{u_L}, \mathbf{u_R})ds = A(\mathbf{u_L}, \mathbf{u_R})_\Phi(\mathbf{u_L}, \mathbf{u_R}) \tag{3.11}$$

이 조건은 Roe condition을 일반화한 것이다. 이 조건에서는 유속함수가 필요하지 않으므로 비보존형 시스템에도 적용이 가능하다. 여기서 f_0 를 $\mathrm{f}_0(\mathbf{w}_L)=\mathbf{u}_L$, $\mathrm{f}_0(\mathbf{w}_R)=\mathbf{u}_R$ 인 함수라고 하고 $\mathbf{A}_0=\partial \mathrm{f}_0/\partial \mathbf{w}$ 라고 하면 다음과 같이 두 상태 $\mathbf{u}_L$ 과 $\mathbf{u}_R$ 를 연계하는 식으로 나타낼 수 있다.

$$\Phi(s;\mathbf{u}_L,\mathbf{u}_R)=f_0(\mathbf{w}_L+s(\mathbf{w}_R-\mathbf{w}_L)) \tag{3.12}$$

여기서

$$s\in[0,1]$$

이다. 따라서 Roe의 행렬 $\mathbf{A}(\mathbf{u}_L,\mathbf{u}_R)_\Phi$ 는 다음과 같이 정의될 수 있다.

$$\mathbf{A}(\mathbf{u}_L,\mathbf{u}_R)_\Phi=\mathbf{C}(\mathbf{u}_L,\mathbf{u}_R)_\Phi\mathbf{B}(\mathbf{u}_L,\mathbf{u}_R)_\Phi^{-1} \tag{3.13}$$

여기서

$$\mathbf{B}(\mathbf{u}_L,\mathbf{u}_R)_\Phi=\int_0^1\mathbf{A}_0(\mathbf{w}_L+s(\mathbf{w}_R-\mathbf{w}_L))ds \tag{3.14}$$

$$\mathbf{C}(\mathbf{u}_L,\mathbf{u}_R)_\Phi=\int_0^1\mathbf{A}(\mathrm{f}_0(\mathbf{w}_L+s(\mathbf{w}_R-\mathbf{w}_L)))\times\mathbf{A}_0(\mathbf{w}_L+s(\mathbf{w}_R-\mathbf{w}_L))ds \tag{3.15}$$

위에서 설명한 weak formulation을 이용하여 Euler 방정식에 대한 Roe-averaged Jacobian matrix를 두 상태 $\mathbf{U}_{j-1}^n,\mathbf{U}_j^n$에 대하여 다음과 같이 구성할 수 있다[7].

$$\mathbf{A}\left(\mathbf{U}_{j-1}^n,\mathbf{U}_j^n\right)_\Phi=\widetilde{\mathbf{C}}_{j-1/2}\widetilde{\mathbf{B}}_{j-1/2}^{-1} \tag{3.16}$$

여기서

$$\widetilde{\mathbf{B}}_{j-1/2} = \int_0^1 \mathbf{A_0}\left[\mathbf{W}_{j-1}^n + s\left(\mathbf{W}_j^n - \mathbf{W}_{j-1}^n\right)\right]ds \tag{3.17}$$

$$\widetilde{\mathbf{C}}_{j-1/2} = \int_0^1 \mathbf{A}\left[\Psi_0\left(\mathbf{W}_{j-1}^n + s\left(\mathbf{W}_j^n - \mathbf{W}_{j-1}^n\right)\right)\right]\mathbf{A_0}\left[\mathbf{W}_{j-1}^n + s\left(\mathbf{W}_j^n - \mathbf{W}_{j-1}^n\right)\right]ds \tag{3.18}$$

이다. 선형 행렬을 구성하기 위하여 parameter vector인 $\mathbf{W}$ 를 도입하였으며 $\mathbf{A}_0(\mathbf{W}) = \partial\Psi_0/\partial\mathbf{W}$ 이다. 이때 Ψ_0 는 $\Psi_0(\mathbf{W}) = \mathbf{U}$ 인 함수이다. Parameter vector, $\mathbf{W}$ 와 함수 $\Psi_0(\mathbf{W})$는 다음과 같다.

$$\mathbf{W} = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \\ w_5 \\ w_6 \end{bmatrix} = \begin{bmatrix} \sqrt{\alpha_v \rho_v} \\ \sqrt{\alpha_l \rho_l} \\ \sqrt{\alpha_v \rho_v} u_v \\ \sqrt{\alpha_l \rho_l} u_l \\ \sqrt{\alpha_v \rho_v}\left(h_v + \dfrac{u_v^2}{2}\right) \\ \sqrt{\alpha_l \rho_l}\left(h_l + \dfrac{u_l^2}{2}\right) \end{bmatrix} \tag{3.19}$$

$$\Psi_0(\mathbf{W}) = \begin{bmatrix} w_1^2 \\ w_2^2 \\ w_1 w_3 + w_2 w_4 \\ w_2 w_4 \\ w_1 w_5 - \alpha_v p \\ w_2 w_6 - \alpha_l p \end{bmatrix} \tag{3.20}$$

이로부터 $\widetilde{\mathbf{B}}_{j-1/2}^{-1}$ 을 행렬을 얻을 수 있다. 보존형의 경우 경로 Φ 에 대하여 독립적인데 비하여 비보존형 시스템은 경로 Φ 에 의존적이다. 이러한 경로에의 의존성을 없애기 위하여 α_k 와 p 의 평균값을 취함으로써 선형화된 보존형 시스템을 얻는다.

$$\frac{\partial \mathbf{U}}{\partial t}+\frac{\partial \mathbf{F}\left(\mathbf{U},\tilde{\alpha}_v,\tilde{\alpha}_l,\tilde{p}\right)}{\partial x}=0 \tag{3.21}$$

여기서

$$\mathbf{F}\left(\mathbf{U},\tilde{\alpha}_v,\tilde{\alpha}_l,\tilde{p}\right)=\begin{bmatrix}\alpha_v\rho_v u_v\\ \alpha_l\rho_l u_l\\ \alpha_v\rho_v u_v^2+\tilde{\alpha}_v p\\ \alpha_l\rho_l u_l^2+\tilde{\alpha}_l p\\ \alpha_v\rho_v u_v\left(h_v+\frac{u_v^2}{2}\right)+\tilde{p}\alpha_l u_l\\ \alpha_l\rho_l u_l\left(h_l+\frac{u_l^2}{2}\right)-\tilde{p}\alpha_l u_l\end{bmatrix} \tag{3.22}$$

이며 새로 도입된 항들은 다음과 같이 계산된다.

$$\frac{1}{\tilde{\alpha}_l}=\frac{1}{2}\left[\frac{1}{\alpha_l\left(\mathbf{U}_\mathbf{L}\right)}+\frac{1}{\alpha_l\left(\mathbf{U}_\mathbf{R}\right)}\right] \tag{3.23}$$

$$\tilde{\alpha}_v=1-\tilde{\alpha}_l \tag{3.24}$$

$$\tilde{p}=\frac{\alpha_l p\left(\mathbf{U}_\mathbf{L}\right)+\alpha_l p\left(\mathbf{U}_\mathbf{R}\right)}{\alpha_l\left(\mathbf{U}_\mathbf{L}\right)+\alpha_l\left(\mathbf{U}_\mathbf{R}\right)} \tag{3.25}$$

이제 경로 Φ에 대하여 독립적이므로 다음과 같은 선형화가 가능하다.

$$\mathbf{C}\left(\mathbf{W},\tilde{\alpha}_v,\tilde{\alpha}_l,\tilde{p}\right)=\frac{\partial \mathbf{F}}{\partial \mathbf{W}}\left(\mathbf{W},\tilde{\alpha}_v,\tilde{\alpha}_l,\tilde{p}\right) \tag{3.26}$$

유동함수 **F** 와 parameter vector **W** 에 대한 jacobian 행렬의 적분을 구함으로써 $\tilde{\mathbf{C}}_{-1/2}$을 얻을 수 있다. 쌍곡선형의 선형화된 시스템을 유도하기 위하여 interface pressure 를 도입한다. 이 항의 선형화된 형태는 다음과

같다.

$$\tilde{\mathbf{I}} = \mathbf{I}\left(\tilde{\mathbf{U}}\right) = \tilde{\alpha}_v \tilde{\delta}\left(\tilde{u}_v - \tilde{u}_l\right)^2 \partial_x \alpha_l \rho_l \tag{3.27}$$

각 상의 평균속도는 다음과 같다.

$$\tilde{u}_v = \frac{\overline{w}_3}{\overline{w}_1} = \frac{\left(\sqrt{\alpha_v \rho_v} u_v\right)_{j-1} + \left(\sqrt{\alpha_v \rho_v} u_v\right)_j}{\left(\sqrt{\alpha_v \rho_v}\right)_{j-1} + \left(\sqrt{\alpha_v \rho_v}\right)_j} \tag{3.28}$$

$$\tilde{u}_l = \frac{\overline{w}_4}{\overline{w}_2} = \frac{\left(\sqrt{\alpha_l \rho_l} u_l\right)_{j-1} + \left(\sqrt{\alpha_l \rho_l} u_l\right)_j}{\left(\sqrt{\alpha_l \rho_l}\right)_{j-1} + \left(\sqrt{\alpha_l \rho_l}\right)_j} \tag{3.29}$$

각 상의 평균 엔탈피는 각각 다음과 같다.

$$\tilde{H}_v = \frac{\overline{w}_5}{\overline{w}_1} = \frac{\left(\sqrt{\alpha_v \rho_v} H_v\right)_{j-1} + \left(\sqrt{\alpha_v \rho_v} H_v\right)_j}{\left(\sqrt{\alpha_v \rho_v}\right)_{j-1} + \left(\sqrt{\alpha_v \rho_v}\right)_j} \tag{3.30}$$

$$\tilde{H}_l = \frac{\overline{w}_6}{\overline{w}_2} = \frac{\left(\sqrt{\alpha_l \rho_l} H_l\right)_{j-1} + \left(\sqrt{\alpha_l \rho_l} H_l\right)_j}{\left(\sqrt{\alpha_l \rho_l}\right)_{j-1} + \left(\sqrt{\alpha_l \rho_l}\right)_j} \tag{3.31}$$

3.3 이상유동 모델에 대한 Roe-averaged matrix

계산을 단순화하기 위하여 기체상을 다음과 같은 상태식을 가지는 이상기체로 가정한다.

$$p = (\gamma - 1)\rho_v e_v \tag{3.32}$$

$\mathbf{A}\left(\mathbf{U}_{j-1}^n, \mathbf{U}_j^n\right)_\Phi = \tilde{\mathbf{C}}_{j-1/2} \tilde{\mathbf{B}}_{j-1/2}^{-1}$ 의 식에 위와 같은 이상기체 성질을 조합하면

결과적으로 다음과 같은 이상기체상에 관한 Roe-averaged matrix를 얻을 수 있다.[7]

$$\tilde{\mathbf{A}}_{j-1/2}=\begin{Bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ \frac{1}{2}(\gamma-3)\tilde{u}_v^2 & \frac{\tilde{p}}{\rho_l}-\tilde{\delta}\tilde{\alpha}_v(\tilde{u}_v-\tilde{u}_l)^2 & (3-\gamma)\tilde{u}_v & 0 & \gamma-1 & 0 \\ (\gamma-1)\frac{\tilde{\alpha}_l}{\tilde{\alpha}_v}\frac{\tilde{u}_v^2}{2} & \frac{\tilde{\alpha}_l}{\tilde{\alpha}_v}\frac{\tilde{p}}{\rho_l}-\tilde{u}_l^2+\tilde{\delta}\tilde{\alpha}_v(\tilde{u}_v-\tilde{u}_l)^2 & (1-\gamma)\frac{\tilde{\alpha}_l}{\tilde{\alpha}_v}\tilde{u}_v & 2\tilde{u}_l & (\gamma-1)\frac{\tilde{\alpha}_l}{\tilde{\alpha}_v} & 0 \\ \left[(\gamma-1)\frac{\tilde{u}_v^2}{2}-\tilde{H}_v\right]\tilde{u}_v & 0 & \tilde{H}_v-(\gamma-1)\tilde{u}_v^2 & \frac{\tilde{p}}{\rho_l} & \gamma\tilde{u}_v & 0 \\ (\gamma-1)\frac{\tilde{\alpha}_l}{\tilde{\alpha}_v}\frac{\tilde{u}_v^2}{2}\tilde{u}_v & \left(\frac{\tilde{p}}{\tilde{\alpha}_l\rho_l}-\tilde{H}_l\right)\tilde{u}_l & (1-\gamma)\frac{\tilde{\alpha}_l}{\tilde{\alpha}_v}\tilde{u}_v\tilde{u}_l & \tilde{H}_l-\frac{\tilde{p}}{\rho_l} & (\gamma-1)\frac{\tilde{\alpha}_l}{\tilde{\alpha}_v}\tilde{u}_l & \tilde{u}_l \end{Bmatrix} \tag{3.33}$$

또한 $\tilde{\delta}$ 값은 식 3.34와 같은 조건을 갖는다. $\tilde{\delta}$ 는 interface pressure 에 관련된 미지수로서 interface pressure coefficient라고 불린다. 이 계수의 최소값 δ_0 는 계산이 가능하며 $\delta=\sigma\delta_0$ 와 같이 σ 값으로 조절할 수 있다. 여기서 σ 는 1과 같거나 그보다 큰 값을 갖는다.

$$\tilde{\delta}\geq\delta_0=\gamma\frac{\tilde{\alpha}_l\tilde{p}}{\tilde{\alpha}_v\rho_l\tilde{\alpha}_m^2} \tag{3.34}$$

여기서

$$\tilde{\alpha}_m^2=(\gamma-1)\left(\tilde{H}_v-\frac{\tilde{u}_v^2}{2}\right)\gamma\frac{\tilde{\alpha}_l\tilde{p}}{\tilde{\alpha}_v\rho_l} \tag{3.35}$$

이다. 확장된 Roe scheme에 의한 새로운 시간단계의 계산은 다음과 같다.

$$\mathbf{U}_j^{n+1}=\mathbf{U}_j^n-\frac{\Delta t}{\Delta x}\left[\mathbf{A}\left(\mathbf{U}_{j-1}^n,\mathbf{U}_j^n\right)_\Phi^+\left(\mathbf{U}_j^n-\mathbf{U}_{j-1}^n\right)+\mathbf{A}\left(\mathbf{U}_j^n,\mathbf{U}_{j+1}^n\right)_\Phi^-\left(\mathbf{U}_{j+1}^n-\mathbf{U}_j^n\right)\right] \tag{3.36}$$

Roe-averaged matrix의 양과 음의 부분은 다음 식과 같이 정의된다.

$$\mathbf{A}\left(\mathbf{U}_{j-1}^{n},\mathbf{U}_{j}^{n}\right)_{\Phi}^{\pm}=R_{j-1/2}\Lambda_{j-1/2}^{\pm}R_{j-1/2}^{-1} \tag{3.37}$$

여기서

$$\Lambda_{j-1/2}^{\pm}=diag\left(\lambda_{j-1/2}^{1\pm},...,\lambda_{j-1/2}^{m\pm}\right) \tag{3.38}$$

이며 양과 음의 고유치들은 다음과 같다.

$$\lambda_{j-1/2}^{k-}=\min\left(0,\lambda_{j-1/2}^{k}\right) \tag{3.39}$$

$$\lambda_{j-1/2}^{k+}=\max\left(0,\lambda_{j-1/2}^{k}\right) \tag{3.40}$$

이상의 식은 비보존형에 대한 수치 기법이다. 또한 본 연구에서는 4 step Runge-Kutta 방법을 사용하였다.

$$\mathbf{U}^{i}=\mathbf{U}^{n}-\theta_{i}\Delta tR\left(\mathbf{U}^{(i-1)}\right)\quad \text{for } i=1,2,3,4 \tag{3.41}$$

여기서 θ_i는 i가 1에서 4의 순서대로 1/4, 1/3, 1/2, 1이며 R은 다음과 같이 정의되는 잔여항이다.

$$R\left(\mathbf{U}^{(i-1)},\mathbf{U}^{n}\right)=\left(\frac{\partial\mathbf{F}}{\partial x}\right)^{(i-1)} \tag{3.42}$$

$$\mathbf{U}^{(0)}=\mathbf{U}^{n},\quad \mathbf{U}^{n+1}=\mathbf{U}^{(4)} \tag{3.43}$$

4. Roe Scheme 의 예시 계산

본 연구에 도입된 해석 방법을 사용해 개발된 코드를 검증하기 위하여 충격파관(shock tube) 문제와 수도밸브(water faucet) 문제를 모사하였다. 두 문제 모두 1차원 관유동을 모사한 것으로 각각의 cell 번호는 그림 1과 같다. 그림에서 볼 수 있듯이 단면적이 일정한 관을 균등한 간격의 N개의 cell로 나누어 계산한다. 그림 1에서 1번과 (N+2)번은 각각 입구 및 출구 경계조건을 나타낸다. 충격파관의 경우 총 길이를 1m로 두었으며, 이를 간격이 일정한 96개의 cell로 나누었다. 수도밸브의 경우에는 총 길이를 12m로 두었으며, 이를 96개의 cell로 나누어 계산하였다. 문제의 개략도 및 상세한 계산 조건과 계산 결과는 표 1과 그림 2~9에 나타내었다.

4.1 충격파관(Shock tube)

충격파관(shock tube)은 주로 기체의 충격파 특성을 연구하기 위해 사용되는 실험설비이다. 이것은 초기 조건이 서로 다른 유체가 격막에 의해서 좌우로 분리되어 있다가, 어느 순간 격막이 제거되면서 여러 파동이 발생하여 전파되어 나가게 하는 장치이다. 몇몇의 연구자들은 단상유동에서 주로 사용되는 이러한 개념을 이상유동에서의 파동 전파특성을 연구하기 위해서 도입하여 왔다.[7,8] 본 연구에서는 2유체 모델의 수치적인 해법의 어려움을 개선한 수치기법을 이용하여 충격파관 문제를 해석하였다.

일정한 단면적을 가지는 관의 중앙에 격막이 존재한다. 기체는 이상기체로 가정하며 비열비는 1.0931이다. 액체는 앞 절에서 언급했던 바와 같이 비압축성 유체로 가정한다. 따라서 액체는 일정한 밀도 $\rho = 720kg/m^3$ 을 갖는다. 본 연구에서는 200개의 셀과 동일한 시간간격을 사용하였으며 관의 길이는 1m로 설정하였다. 자세한 초기 상태는 표 1에 나타내었다.

Left State $\mathbf{U}_L$	Right State $\mathbf{U}_R$
$\alpha_L = 0.25$	$\alpha_R = 0.25$
$P_L = 20\,\mathrm{MPa}$	$P_R = 15\,\mathrm{MPa}$
$u_v^L = 0\,\mathrm{m/s}$	$u_v^R = 0\,\mathrm{m/s}$
$u_l^L = 0\,\mathrm{m/s}$	$u_l^R = 0\,\mathrm{m/s}$
$h_v^L = 3092.7\,\mathrm{kJ/kg}$	$h_v^R = 3092.7\,\mathrm{kJ/kg}$
$h_l^L = 1338.2\,\mathrm{kJ/kg}$	$h_l^R = 1338.2\,\mathrm{kJ/kg}$

표 1 예시 계산 1의 초기조건

표 1에서 볼 수 있듯이, 초기에 격막 좌우의 압력에 차이가 있는 것을 제외하고는 격막 오른쪽과 왼쪽의 조건이 동일하다. 전체 관 내의 초기속도는 액체와 기체 모두 0m/s이며 기공률과 엔탈피 역시 동일한 초기값을 갖는다. 충격파관 문제에 대한 해석해를 구하는 것은 불가능하므로 본 문제에서는 기존에 계산된 다른 참고문헌의 값과 비교하는 것을 목적으로 한다.[2]

그림 3은 거리에 따른 압력 값의 변화를 보여준다. 여기에서 실선은 참고문헌의 값을 나타내며 점선은 본 연구에서 계산한 결과값을 나타낸다. 이때 시간은 0.23ms이다. 이 그래프에서 압력파는 불연속면으로부터 발생된다. 격막이 순간적으로 파괴되면 충격파는 저압인 오른쪽으로, 팽창파는 부채꼴모양으로 넓어지면서 고압인 왼쪽으로 진행하게 된다. 순수기체의 경우와는 다르게[9], 이상유동의 충격파관 문제를 모사한 그림 3에서는 불연속면 부근의 기울기가 평탄한 것을 알 수 있다. 이는 서로 다른 국소 상 속도에 의한 확산효과 때문이다.

그림 4는 t=0.23ms 에서의 거리에 따른 기체 속도 분포 그래프이다. 이때 시간 t는 0.23ms 이며, 실선은 참고문헌의 값이고 점선은 본 연구에서 계산한 결과값을 나타낸다. 그림 4 를 보면 불연속면 근처에서 속도가 갑자기 변하는데 이로부터 일반적인 상류차분법과 마찬가지로 본 연구에서

사용한 확장된 Roe의 기법 역시 충격파와 같은 불연속면을 포획하는 특성이 있다는 것을 알 수 있다.

그림 5에서는 시간에 따른 압력 분포를 나타내었다. 이 때 시간은 0ms, 0.0652ms, 0.1118ms, 0.1724ms, 0.2283ms이다. 이 그림에서는 불연속면을 기준으로 확산현상이 일어나는 과정을 볼 수 있다. 확산이 진행되는 부분을 세 개의 구간으로 나누어 각각 팽창팬(expansion fan), 불연속면(contact discontinuity), 충격파(shock wave)이다. 팽창팬은 높은 압력 구간에서의 완만한 팽창파를 말하며 이 때의 팽창파는 소리의 속도로 전파된다.

압력이나 속도 이외에 밀도나 체적비와 같은 다른 값들도 불연속면을 지나면서 변화된다. 그림 6에서는 체적비, 밀도, 속도, 엔트로피, 엔탈피, 압력을 시간 t=0.23ms에서 거리에 대하여 나타내었다. 이 중 체적비, 속도, 엔트로피, 엔탈피는 기체상과 액체상으로 구분하였다.

4.2 수도밸브(Water Faucet)

수도밸브(water faucet) 문제는 Ramsom 에 의해 제안된 문제이다[10]. 단면적이 일정한 원통형 수직 관에서 유체는 중력의 영향을 받으며 water jet 를 포함한다. 초기에 관은 정지해 있는 기체로 둘러싸인 물기둥 형태이다. 이 때, 기공률은 0.2 이며 물기둥은 10m/s 의 속도를 갖는다. 또한 초기압력은 10^5 Pa 이다.

수도밸브 문제는 적당한 조건을 이상화하여 해석적으로 계산하는 것이 가능하다. 본 연구에서는 확장된 Roe 의 기법을 사용하여 수도밸브 문제를 계산하였다. 계산된 결과는 해석적으로 계산된 엄밀해와 비교하여 검증한다. 수도밸브 문제의 개략적인 형태는 그림 7 과 같다.

그림 7(a)는 초기상태에서의 액체상과 기체상의 분포를 나타낸다. Water jet 가 관의 중앙에 위치하며 기체는 그 주위를 둘러싸고 있는 형태이다. 그림 7(b)는 수도밸브 문제에 대한 해를 해석적으로 구한 결과를 나타낸다. 그림 7(b)에서 알 수 있듯이, 시간이 지날수록 기공률은 점점 발달하여 액체 속도에 영향을 주게 된다.

입구에서의 속도는 액체의 경우 10m/s, 기체의 경우 0m/s 로 고정한다. 출구에서는 압력을 고정하며 이 때의 압력은 대기압과 같다. Void wave 가 관의 출구까지 확산되면 유동은 정상상태에 도달하게 된다. 그림 8 은 정상상태에 도달할 때까지의 기공률의 변화를 시간에 따라 나타낸 것이다. 이 때, 셀 수는 96 개이며, CFL 수는 0.9 이다. 계면 압력항에서의 σ 값은 1.01 을 사용하였다.

본 연구에 도입된 해석 방법의 안정성을 알아보기 위해 다양한 개수의 셀을 사용하여 계산을 수행하였다. 여기서는 각각 12, 24, 48, 96, 192, 768 개의 셀을 사용하여 계산하였다. CFL 수는 0.9 로 고정하였으며 나머지 조건들도 모두 같다. 그림 9 는 이러한 셀 개수에 따른 기공률의 형상을 보여준다. 이 때 시간 t=0.5s 이며, 같은 시간에서의 엄밀해와 비교하였다. 그림 9 에서 엄밀해는 굵은 실선으로 나타내었다. 셀의 개수가 많아질수록 엄밀해에 가까워지는 경향을 보이는 것을 알 수 있다. 그림 9 의 결과에서 주목할 만한 점은 셀 수가 적은 경우에도 기공률의 급격한 변화에 따른 진동이 일어나지 않는다는 것이다. 이는 곧 본 연구에 사용한 수치 기법이 불연속을 잘 포획하는 것을 나타낸다.

5. 결론

이상유동을 모사하기 위해 1차원 two-phase two-fluid model을 사용하였다. 기본 이상유동을 모사한 비보존형 시스템에서는 복소수 고유값을 가지며 불량 조건의 초기치 문제가 되어 Roe 방법을 그대로 사용할 수 없다. 따라서 상류차분법을 기반으로 만들어진 Roe scheme 에 weak formulation을 적용하여 얻어진 확장된 Roe scheme을 사용하였다. 이에 따라 Roe-averaged matrix를 얻음으로써 충격파관과 수도밸브 문제의 이상유동에 관한 두 가지 예시 계산을 수행하였다.

충격파관(shock tube) 문제에서 압력파는 불연속면으로부터 발생된다. 왼쪽으로부터 팽창파, 접촉 불연속면, 충격파의 순으로 전파되어 나가며, 불연속면 부근에서 그래프의 기울기가 정체되어 있는 결과를 얻을 수 있다. 이는 서로 다른 국소 상 속도에 의한 확산효과 때문인 것으로 추정된다. 또한 불연속면 근처에서 속도가 갑자기 변하는데 이로부터 일반적인 상류차분법과 마찬가지로 본 연구에서 사용한 확장된 Roe의 기법 역시 충격파와 같은 불연속면을 포획하는 특성이 있다는 것을 알 수 있다. 수도밸브(water faucet) 문제에서 셀의 개수가 늘어날수록 엄밀해에 가까워지는 것을 확인하였다. 여기에서 주목할 점은 셀 수가 적은 경우에도 기공률의 급격한 변화에 따른 진동이 일어나지 않는다는 것이다. 이는 곧 본 연구에 사용한 수치 기법이 불연속을 잘 포획하는 것을 나타낸다. 따라서 큰 진동이 없이 불연속면을 계산할 수 있다는 이점이 있다는 것을 알 수 있다.

일반적으로 이상유동 문제의 해를 해석적으로 구하기는 어려우므로, 앞으로는 본 연구에서 계산된 결과의 신뢰성 향상을 위해 확장된 Roe scheme 이외의 다른 방법을 사용한 해석 결과와 비교 검증하는 과정이 요구된다.

참고문헌

[1] I. Toumi, A. Kumbaro, “An Approximate Linearized Riemann Solver for a Two-Fluid Model”, Journal of Computational Physics, Vol. 124, pp. 286-300, 1996

[2] H. Stadtke, G. Franchello, B. Worth,“Numerical simulation of multi-dimensional two-phase flow based on flux vector splitting”, Nuclear Engineering and Design, Vol. 177, pp. 199-213, 1997

[3] V. H. RANSOM and D. L. Hicks, “Hyperbolic two-pressure models for two-phase flow”, Journal of Computational Physics, Vol. 53, pp. 124-151, 1984

[4] J. H. Stuhmiller, “The influence of interfacial pressure forces on the character of two-phase flow model equations”, International Journal of Multiphase Flow, Vol. 3, pp. 551-560, 1977

[5] P. L. Roe, “Characteristic-based schemes for the Euler equations”, Annual Review of Fluid Mechanics, Vol.18, pp. 337-365,1986

[6] I. Toumi, “A weak formulation of Roe’s approximate Riemann solver”, Journal of Computational Physics, Vol. 102, pp. 360-373, 1992

[7] I. Toumi, “An upwind numerical method for two-fluid two-phase flow models”, Nuclear Science and Engineering, Vol. 123, pp. 147-168, 1996

[8] Cortes, J., Debussche, A., and Toumi, I.,“A Density Perturbation Method to Study the Eigenstructure of Two-Phase Flow Equation Systems”, Journal of Computational Physics, Vol. 147, pp. 463-484, 1998

[9] Herbert Stadtke, “Gasdynamic Aspects of Two-Phase Flow”, WILEY-VCH, pp. 157-159, 2006

[10] V. H. RANSOM, “Numerical Benchmark Tests”, Multi-phase Science and Technology, Vol. 3, G. F. HEWITT, J. M. DELHAYE, and N. ZUBER, Eds., Hemisphere, New York, 1987

[11] Cambridge University, “Numerical recipes: the art of scientific computing(Fortran version)”, William H. Press, 1986

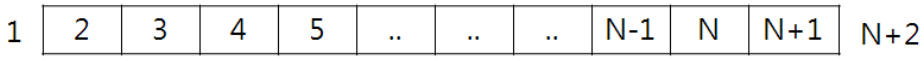


그림 1 예시문제의 cell 번호

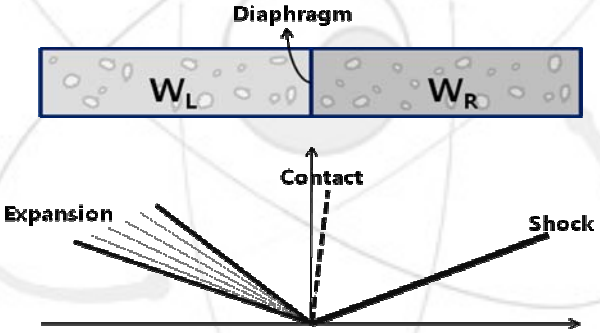


그림 2 충격파관(shock tube) 문제의 개략도

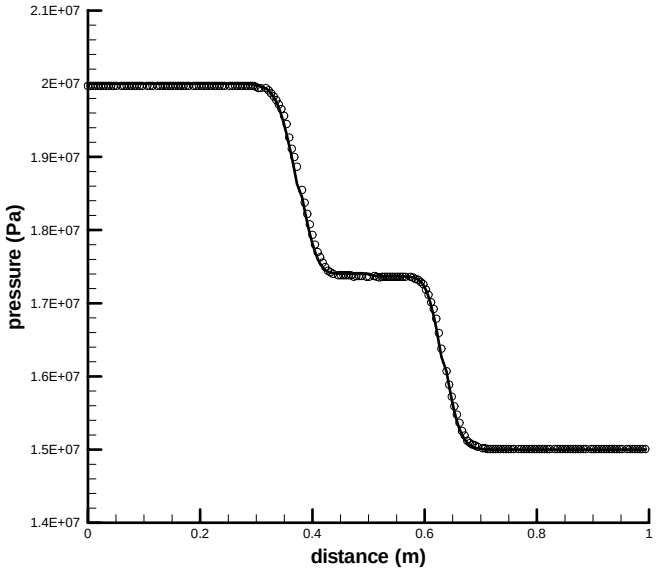


그림 3 충격파관 문제의 압력 분포 (t=0.23ms)

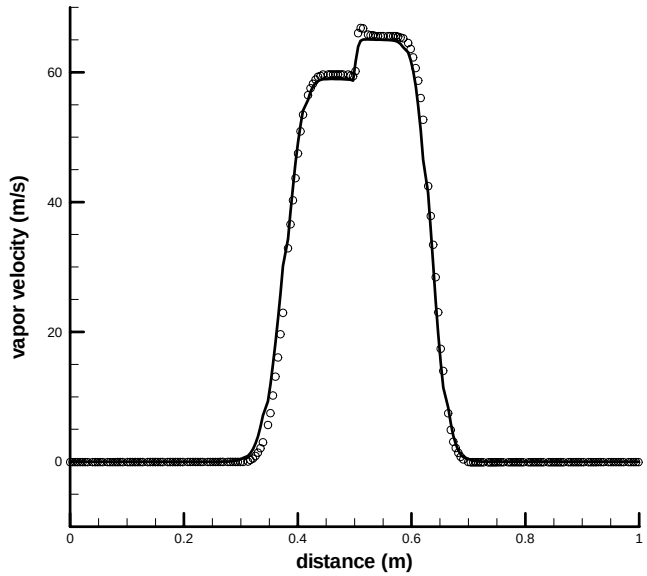


그림 4 충격파관 문제의 기체속도 분포 (t=0.23ms)

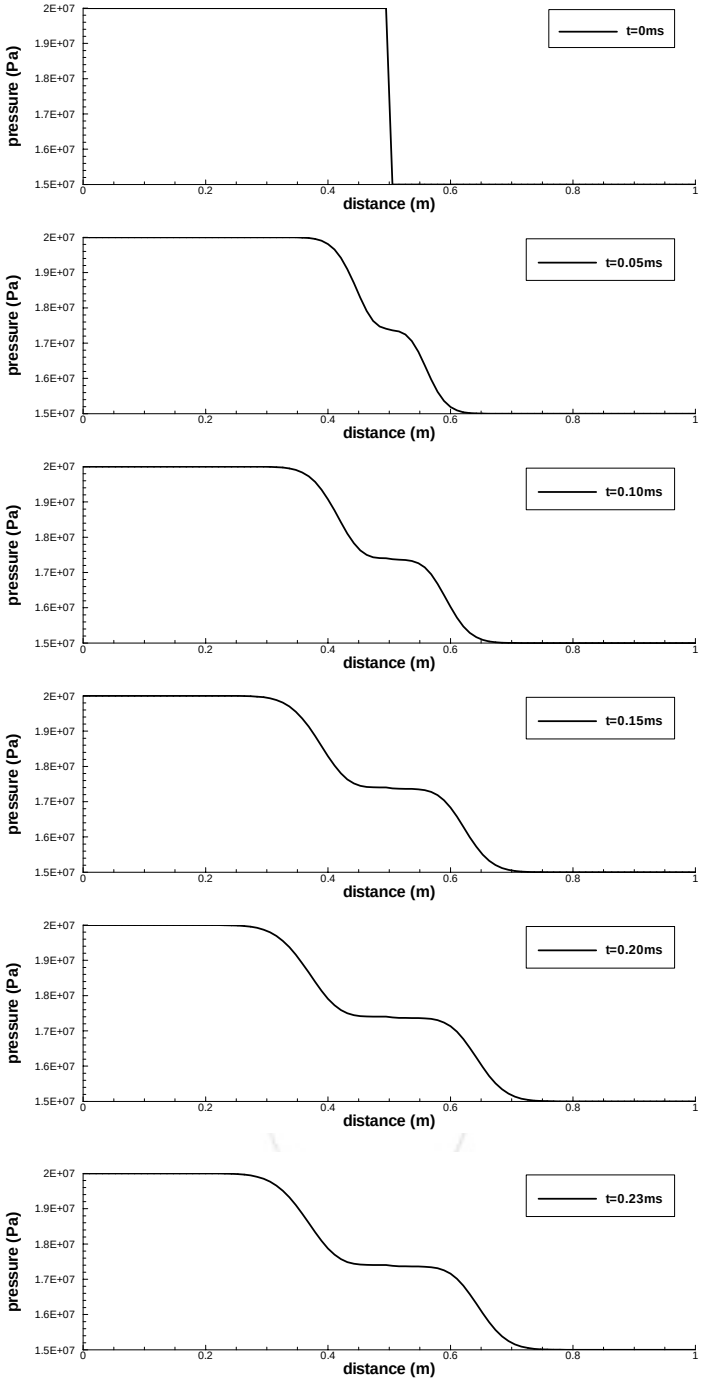
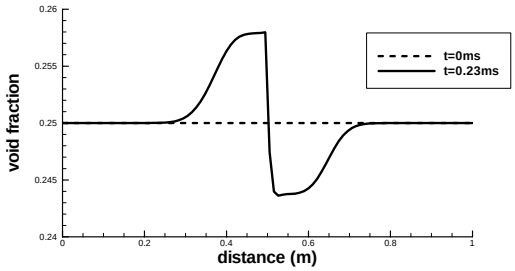
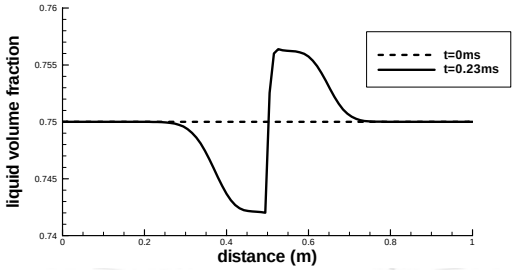


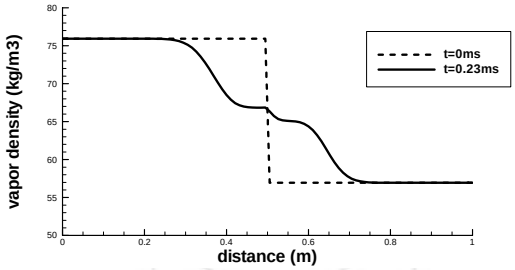
그림 5 시간에 따른 압력 분포



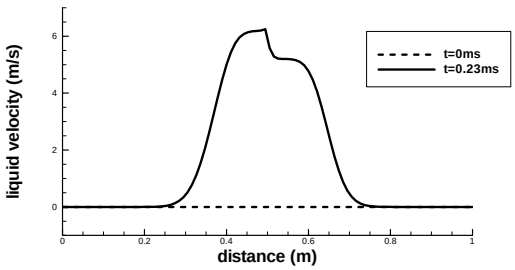
(a)



(b)



(c)



(d)

그림 6 예시 계산 1의 결과 (t=0.23ms)

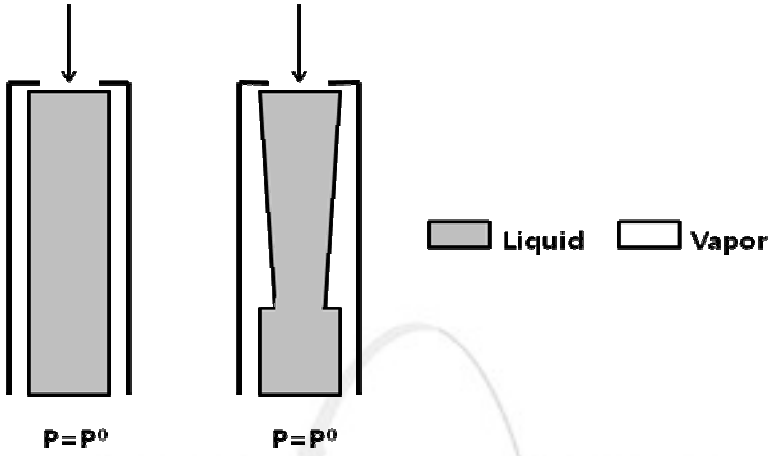


그림 7 수도밸브(water faucet) 문제의 개략도

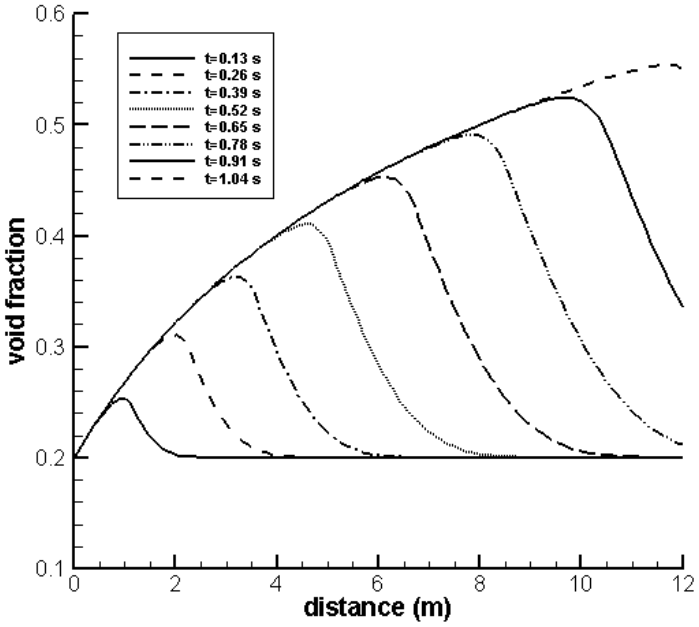


그림 8 시간에 따른 기공률의 변화

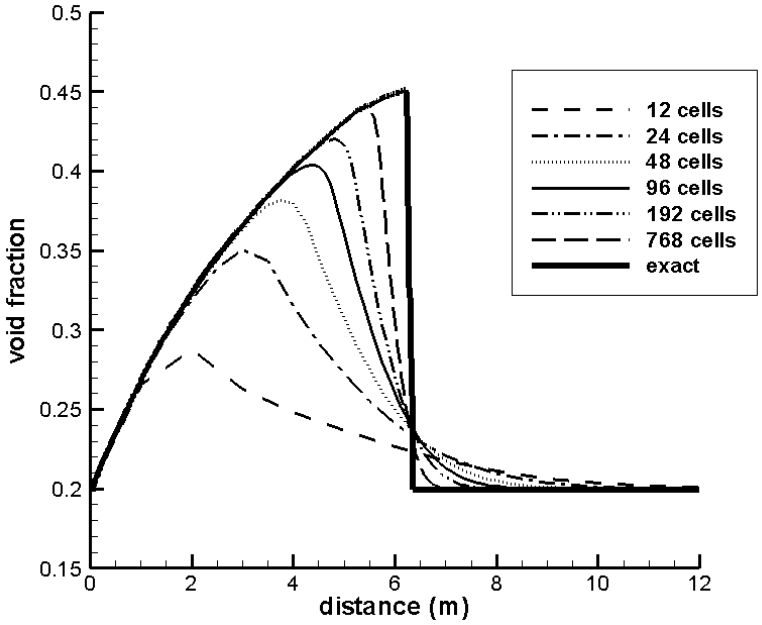


그림 9 cell 수에 따른 기공률 비교 (t=0.5s)

부록: 코드구조 및 Source List

A.1. 코드 구조

표 A1에서 코드 구조에 관한 내용을 살펴볼 수 있다. 또한 수학 관련 부프로그램[11]을 제외한 전체 코드를 다음 절에 첨부하였다.

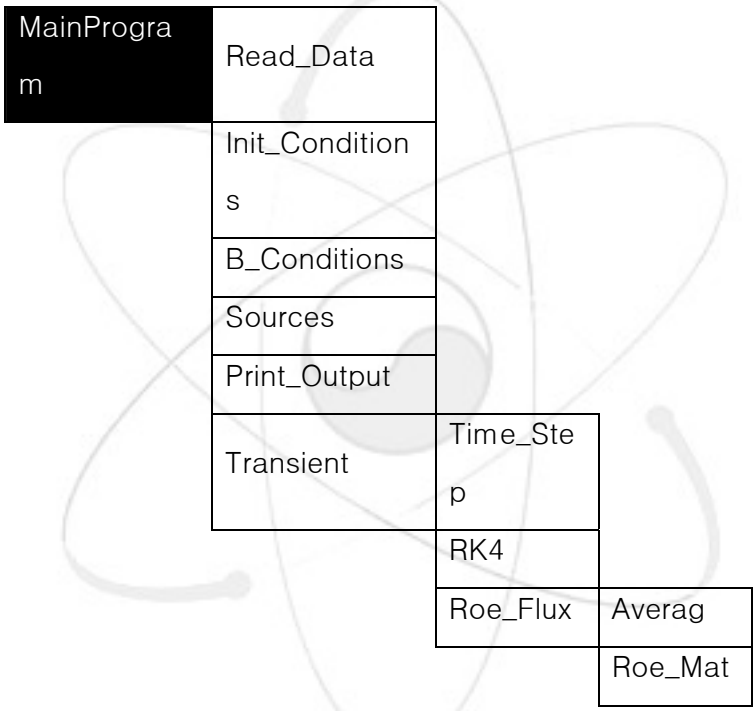


표 A1 코드 구조

A.2. Source List

```
!=====================================================
        MODULE mod_input
!=====================================================
!
        INTEGER::n,KD,imax,maxiter
        PARAMETER (n=6,KD=8,imax=96,maxiter=50000)
!
        INTEGER::err,dtcase,algo,src
        REAL(KIND=KD)::gamma,rhol,sigma,CFL,L
        REAL(KIND=KD)::alphaL,PL,uvL,ulL,hvL,hlL
        REAL(KIND=KD)::alphaR,PR,uvR,ulR,hvR,hlR
!
        INTEGER::bc_in,bc_out
        REAL(KIND=KD)::bc_uv,bc_ul,bc_p
!
        INTEGER::rtype
!
        END MODULE mod_input
!
!=====================================================
        MODULE mod_prop
!=====================================================
!
        USE mod_input,only:imax,maxiter,KD
!
        INTEGER::iter,vfs
        REAL(KIND=KD)::vf(9,imax)
        REAL(KIND=KD)::dx,dt(imax),t(maxiter)
        REAL(KIND=KD)::alphavL,alphalL,rhovL,rholL,evL,elL,etvL,etlL,htvL,htlL
        REAL(KIND=KD)::alphavR,alphalR,rhovR,rholR,evR,elR,etvR,etlR,htvR,htlR
!
        END MODULE
!
!=====================================================
        MODULE mod_cal
!=====================================================
!
        USE mod_input,ONLY:imax,maxiter,KD
        REAL(KIND=KD),DIMENSION(2,imax)::alpv,alpl,rhl,rhv,uv,ul,etv,etl,p
!
        END MODULE mod_cal
!
!=====================================================
        MODULE mod_averag
!=====================================================
        USE mod_input,ONLY:n,rhol,gamma,KD
!
        REAL(KIND=KD)::uvb,ulb,hvb,hlb
        REAL(KIND=KD)::allb,alvb,pb,amb,delb
```

```
!
        END MODULE mod_averag
!
!comment
!**********************************************************************
!
!       CODE FOR
!               One Dimensional Hyperbolic Coupled Solver
!               of a Two-phase Flow
!               with a Fourth Order Runge-Kutta Method
!               and an Extension of Roe's Method
!
!       Designed in Fortran90
!
!**********************************************************************
!
        PROGRAM main
!
        USE mod_input
        USE mod_prop
        USE mod_cal
!
        IMPLICIT NONE
!
        REAL(KIND=KD)::U(n,imax),dU(n,imax)
        REAL(KIND=KD)::cputime,time_start,time_end
        INTEGER::i,j,flag
!
!       Read input data
!
        CALL read_data
!
!       Set initial condition
!
        CALL IC(U)
!
!       Open output files
!
        OPEN(19,file='conv.txt')
!
!       Initial cpu_time clock
!
        CALL cpu_time(time_start)
!
!       Main loop
!
        DO iter=1,maxiter
!
!               Set time step
!
                CALL delta_t(U)
!
!               4th Runge-Kutta method
!
                CALL RK4(U,dU)
```

```
!
!                   Convergence
!
                    CALL Conv(iter,dU,U,flag)
                    IF (flag==1) EXIT
!
!                   save results
!
                    CALL dcmp(U)
                    CALL save_results
                    IF (vfs==8) EXIT
!
          ENDDO
!
!         print results
!
          CALL print_results
!
!         Final cpu_time clock
!
          CALL cpu_time(time_end)
          cputime=time_end-time_start
          WRITE(6,'(1a,1e15.5,1a)') 'total CPU time : ',cputime,' sec'
          WRITE(6,'(1a,1e15.5,1a)') 'total time : ',t(iter),' sec'
!
!         Close output files
!
          CLOSE(19)
!
          END PROGRAM
!
!=======================================================
          SUBROUTINE read_data
!=======================================================
          USE mod_input
!
          IMPLICIT NONE
!
          NAMELIST/property/gamma,rhol
          NAMELIST/solution/CFL,err,sigma
          NAMELIST/geometry/L
          NAMELIST/option/dtcase,src
          NAMELIST/left_state/alphaL,PL,uvL,ulL,hvL,hlL
          NAMELIST/right_state/alphaR,PR,uvR,ulR,hvR,hlR
          OPEN (unit=11,file='eui.in',status='old')
          READ(11,property)
          READ(11,solution)
          READ(11,geometry)
          READ(11,option)
          READ(11,left_state)
          READ(11,right_state)
          CLOSE(11)
!
          END SUBROUTINE
!
```

```
!=====================================================
        SUBROUTINE Read_BC_option
!=====================================================
!
                USE mod_input,ONLY:bc_in,bc_out
!
                NAMELIST/boundary_condition/bc_in,bc_out
                OPEN (unit=11,file='eui.in',status='old')
                READ(11,boundary_condition)
                CLOSE(11)
!
        END SUBROUTINE
!
!=====================================================
        SUBROUTINE Read_velocity_inlet
!=====================================================
!
                USE mod_input,ONLY:bc_uv,bc_ul
!
                NAMELIST/velocity_inlet/bc_uv,bc_ul
                OPEN (unit=11,file='eui.in',status='old')
                READ(11,velocity_inlet)
                CLOSE(11)
!
        END SUBROUTINE
!
!=====================================================
        SUBROUTINE Read_pressure_outlet
!=====================================================
!
                USE mod_input,ONLY:bc_p
!
                NAMELIST/pressure_outlet/bc_p
                OPEN (unit=11,file='eui.in',status='old')
                READ(11,pressure_outlet)
                CLOSE(11)
!
        END SUBROUTINE
!
!=====================================================
        SUBROUTINE Read_print_results
!=====================================================
!
                USE mod_input,ONLY:rtype
!
                NAMELIST/print_results/rtype
                OPEN (unit=11,file='eui.in',status='old')
                READ(11,print_results)
                CLOSE(11)
!
        END SUBROUTINE
!
!=====================================================
        SUBROUTINE IC(U)
!=====================================================
```

```
!
      USE mod_input
      USE mod_prop
!
      IMPLICIT NONE
      INTEGER::i,j
      REAL(KIND=KD)::UL(n),UR(n),U(n,imax)
!
      dx=L/(imax-1)
!
!     Left state
!
      alphavL=alphaL
      alphalL=1.-alphaL
      rholL=rhol
      htvL=hvL+uvL**2*0.5d0
      htlL=hlL+ulL**2*0.5d0
      etvL=(1./gamma)*(htvL+(gamma-1.)*uvL**2)
      etlL=(1./gamma)*(htlL+(gamma-1.)*ulL**2)
      rhovL=PL/(htvL-etvL)
!
      UL(1)=alphavL*rhovL
      UL(2)=alphalL*rholL
      UL(3)=alphavL*rhovL*uvL
      UL(4)=alphalL*rholL*ulL
      UL(5)=alphavL*rhovL*etvL
      UL(6)=alphalL*rholL*etlL
!
!     Right state
!
      alphavR=alphaR
      alphalR=1.-alphaR
      rholR=rhol
      htvR=hvR+uvR**2*0.5d0
      htlR=hlR+ulR**2*0.5d0
      etvR=(1./gamma)*(htvR+(gamma-1.)*uvR**2)
      etlR=(1./gamma)*(htlR+(gamma-1.)*ulR**2)
      rhovR=PR/(htvR-etvR)
!
      UR(1)=alphavR*rhovR
      UR(2)=alphalR*rholR
      UR(3)=alphavR*rhovR*uvR
      UR(4)=alphalR*rholR*ulR
      UR(5)=alphavR*rhovR*etvR
      UR(6)=alphalR*rholR*etlR
!
!     Set initial vector
!
      DO i=1,imax
            IF (i<=imax/2) THEN
                  DO j=1,n
                        U(j,i)=UL(j)
                  ENDDO
            ELSE
                  DO j=1,n
```

```
                                        U(j,i)=UR(j)
                                ENDDO
                        ENDIF
                ENDDO
!
!               save initial data
!
                CALL save_IC(U)
!
                END SUBROUTINE IC
!
!
!=====================================================
                SUBROUTINE save_IC(U)
!=====================================================
!
                USE mod_input,ONLY:n,KD,rtype
                USE mod_cal
                USE mod_prop,ONLY:t,vfs,vf,imax
!
                REAL(KIND=KD)::U(n,imax)
!
                CALL Read_print_results
                CALL dcmp(U)
                IF (rtype==1) THEN
                        vf(1,:)=p(2,:)
                        p(1,:)=p(2,:)
                        alpv(1,:)=alpv(2,:)
                        alpl(1,:)=alpl(2,:)
                        uv(1,:)=uv(2,:)
                        ul(1,:)=ul(2,:)
                        rhv(1,:)=rhv(2,:)
                        vfs=0
                ELSEIF (rtype==2) THEN
                        vf(1,:)=alpv(2,:)
                        vfs=0
                ENDIF
!
                END SUBROUTINE save_IC
!
!=====================================================
                SUBROUTINE delta_t(U)
!=====================================================
                USE mod_input,ONLY:CFL,n,imax,rhol,gamma,dtcase,KD
                USE mod_prop,ONLY:dt,dx,iter,t
!
                IMPLICIT NONE
                INTEGER::i
                REAL(KIND=KD)::U(n,imax)
                REAL(KIND=KD)::rhl(imax),alll(imax),alv(imax),rhv(imax),&
                                        uv(imax),ev(imax),p(imax),c(imax)
!
                DO i=imax,imax
                        rhl(i)=rhol                     ! Constant
                        alll(i)=U(2,i)/rhl(i)
```

```
                alv(i)=1-alll(i)
                rhv(i)=U(1,i)/alv(i)
                uv(i)=U(3,i)/U(1,i)
                ev(i)=U(5,i)/U(1,i)-uv(i)**2*0.5d0
                p(i)=ev(i)*rhv(i)*(gamma-1)
                c(i)=sqrt(gamma*p(i)/rhv(i))
        ENDDO
!
        IF (dtcase==1) THEN
!               Constant delta t case
                DO i=1,imax
                        dt(i)=CFL*dx/(uv(imax)+c(imax))
                ENDDO
        ELSEIF (dtcase==2) THEN
!               constant CFL case
                DO i=1,imax
                        dt(i)=CFL*dx/(uv(i)+c(i))
                ENDDO
        ENDIF
!
        IF (iter==1) THEN
                t(iter)=0
        ELSE
                t(iter)=t(iter-1)+dt(1)
        ENDIF
!
        END SUBROUTINE delta_t
!
!=======================================================
        SUBROUTINE RK4(U,dU)
!=======================================================
!
        USE mod_input,ONLY:n,imax,KD
        USE mod_prop,ONLY:dx,dt
!
        IMPLICIT NONE
        INTEGER::i,j,k
        REAL(KIND=KD)::U(n,imax),dU(n,imax)
        REAL(KIND=KD)::Uori(n,imax),Unew(n,imax)
!
        Uori=U
!
        DO k=1,4
                DO i=2,imax-1
                        CALL RoeSch(i,U,dU)
                        DO j=1,n
                                Unew(j,i)=Uori(j,i)+(1./(5.-k))*dU(j,i)
                        ENDDO
                ENDDO
!
                CALL BC(k,U,Unew,dU,Uori)
!
!       Update data
!
                U=Unew
```

```
!
        ENDDO
!
        END SUBROUTINE RK4
!
!=====================================================
        SUBROUTINE RoeSch(i,Q,dQ)
!=====================================================
        USE mod_input,ONLY:n,imax,KD,src
        USE mod_prop,ONLY:dx,dt
!
        IMPLICIT NONE
        INTEGER::i,j
        REAL(KIND=KD)::Q(n,imax),dQ(n,imax),H(n)
        REAL(KIND=KD)::Fp(n),Fm(n),dFdx(n)
!
        CALL RoeFlux(Q(:,i-1),Q(:,i),Fp,1,i)
        CALL RoeFlux(Q(:,i),Q(:,i+1),Fm,2,i)
        IF (src==0) THEN
                H=0
        ELSE
                CALL Source(Q(:,i),H)
        ENDIF
        DO j=1,n
                dFdx(j)=(Fm(j)+Fp(j))/dx
                dQ(j,i)=dt(i)*(-dFdx(j)+H(j))
        ENDDO
!
        END SUBROUTINE RoeSch
!
!=====================================================
        SUBROUTINE conv(iter,dQ,Q,flag)
!=====================================================
        USE mod_input,ONLY:n,imax,maxiter,err,KD
!
        IMPLICIT NONE
        INTEGER::iter,flag,i,k
        REAL(KIND=KD)::dQ(n,imax),Q(n,imax)
        REAL(KIND=KD)::dsumm(n),summ(n),ratio(n)
!
        dsumm(k)=0.
        summ(k)=0.
!
        DO k=1,n
                DO i=1,imax
                        dsumm(k)=dsumm(k)+dabs(dQ(k,i))
                        summ(k)=summ(k)+dabs(Q(k,i))
                ENDDO
                ratio(k)=dsumm(k)/summ(k)
        ENDDO
!
!       Print ratio of convergence
!
        WRITE(6,110) iter,ratio(1),ratio(2),ratio(3),ratio(4),ratio(5),ratio(6)
        WRITE(19,110) iter,ratio(1),ratio(2),ratio(3),ratio(4),ratio(5),ratio(6)
```

```
        110 FORMAT(1x,i5,6e12.4)
!
        IF (ratio(1)<=err .and. ratio(2)<=err .AND. &
                ratio(3)<=err .and. ratio(4)<=err .AND. &
                ratio(5)<=err .and. ratio(6)<=err) THEN
                flag=1
        ENDIF
!
        END SUBROUTINE conv
!
!=====================================================
        SUBROUTINE dcmp(U)
!=====================================================
!
        USE mod_input,ONLY:n,imax,KD
        USE mod_cal
!
        IMPLICIT NONE
        INTEGER::i
        REAL(KIND=KD)::U(n,imax),Q(n)
!
        DO i=1,imax
                CALL Qgen(U(:,i),Q)
                alpv(2,i)=Q(1)
                alpl(2,i)=1.-Q(1)
                rhv(2,i)=U(1,i)/Q(1)
                uv(2,i)=Q(2)
                ul(2,i)=Q(3)
                etv(2,i)=U(5,i)/U(1,i)
                etl(2,i)=U(6,i)/U(2,i)
                p(2,i)=Q(6)
        ENDDO
!
        END SUBROUTINE dcmp
!
!=====================================================
        SUBROUTINE save_results
!=====================================================
!
        USE mod_input,ONLY:rtype
!
        CALL Read_print_results
!
        IF (rtype==1) THEN
                CALL save_rtype1
        ELSEIF (rtype==2) THEN
                CALL save_rtype2
        ENDIF
!
        END SUBROUTINE save_results
!
!=====================================================
        SUBROUTINE save_rtype1
!=====================================================
!
```

```
        USE mod_cal
        USE mod_prop,ONLY:iter,t,vfs,vf
!
                IF (t(iter)>0.23E-3) THEN
                        vf(6,:)=p(2,:)
                        vfs=8
                ELSEIF (t(iter)>0.20E-3.AND.vfs<5) THEN
                        vf(5,:)=p(2,:)
                        vfs=4
                ELSEIF (t(iter)>0.15E-3.AND.vfs<4) THEN
                        vf(4,:)=p(2,:)
                        vfs=3
                ELSEIF (t(iter)>0.10E-3.AND.vfs<3) THEN
                        vf(3,:)=p(2,:)
                        vfs=2
                ELSEIF (t(iter)>0.05E-3.AND.vfs<2) THEN
                        vf(2,:)=p(2,:)
                        vfs=1
                ENDIF
!
        END SUBROUTINE save_rtype1
!
!=======================================================
        SUBROUTINE save_rtype2
!=======================================================
!
        USE mod_input,ONLY:KD,imax
        USE mod_cal,ONLY:alpv
        USE mod_prop,ONLY:iter,t,vfs,vf
!
                IF (t(iter)>1.04) THEN
                        vf(9,:)=alpv(2,:)
                        vfs=8
                ELSEIF (t(iter)>0.91.AND.vfs<7) THEN
                        vf(8,:)=alpv(2,:)
                        vfs=7
                ELSEIF (t(iter)>0.78.AND.vfs<6) THEN
                        vf(7,:)=alpv(2,:)
                        vfs=6
                ELSEIF (t(iter)>0.65.AND.vfs<5) THEN
                        vf(6,:)=alpv(2,:)
                        vfs=5
                ELSEIF (t(iter)>0.52.AND.vfs<4) THEN
                        vf(5,:)=alpv(2,:)
                        vfs=4
                ELSEIF (t(iter)>0.39.AND.vfs<3) THEN
                        vf(4,:)=alpv(2,:)
                        vfs=3
                ELSEIF (t(iter)>0.26.AND.vfs<2) THEN
                        vf(3,:)=alpv(2,:)
                        vfs=2
                ELSEIF (t(iter)>0.13.AND.vfs<1) THEN
                        vf(2,:)=alpv(2,:)
                        vfs=1
                ENDIF
```

```
!
        END SUBROUTINE save_rtype2
!
!=======================================================
        SUBROUTINE print_results
!=======================================================
!
        USE mod_input,ONLY:rtype
        USE mod_cal,ONLY:alpv,p
        USE mod_prop,ONLY:imax,iter,dx,vf
!
        CALL Read_print_results
        IF (rtype==1) THEN
                CALL print_rtype1
        ELSEIF (rtype==2) THEN
                CALL print_rtype2
        ENDIF
!
        END SUBROUTINE
!
!=======================================================
        SUBROUTINE print_rtype1
!=======================================================
!
        USE mod_cal,ONLY:p,alpv,alpl,uv,ul,rhv
        USE mod_prop,ONLY:imax,iter,dx,vf
!
        INTEGER::i
!
        OPEN(31,file='pressure.txt')
        OPEN(32,file='void_fraction.txt')
        OPEN(33,file='l_volume_fraction.txt')
        OPEN(34,file='v_velocity.txt')
        OPEN(35,file='l_velocity.txt')
        OPEN(36,file='v_desity.txt')
        OPEN(37,file='time_pressure.txt')
        DO i=1,imax
                WRITE(31,'(3e15.5)') (i-1)*dx,p(:,i)
                WRITE(32,'(3e15.5)') (i-1)*dx,alpv(:,i)
                WRITE(33,'(3e15.5)') (i-1)*dx,alpl(:,i)
                WRITE(34,'(3e15.5)') (i-1)*dx,uv(:,i)
                WRITE(35,'(3e15.5)') (i-1)*dx,ul(:,i)
                WRITE(36,'(3e15.5)') (i-1)*dx,rhv(:,i)
                WRITE(37,'(10e15.5)') (i-1)*dx,vf(:,i)
        ENDDO
        CLOSE(31)
        CLOSE(32)
        CLOSE(33)
        CLOSE(34)
        CLOSE(35)
        CLOSE(36)
        CLOSE(37)
!
        END SUBROUTINE
!
```

```
!=====================================================
      SUBROUTINE print_rtype2
!=====================================================
!
      USE mod_prop,ONLY:imax,dx,vf
!
      INTEGER::i
!
      OPEN(31,file='time_void_fraction.txt')
      DO i=1,imax
            WRITE(31,'(10e15.5)') (i-1)*dx,vf(:,i)
      ENDDO
      CLOSE(31)
!
      END SUBROUTINE
!
!=====================================================
      SUBROUTINE RoeFlux(UL,UR,F,op,inum)
!=====================================================
      USE mod_input,ONLY:n,KD
!
      IMPLICIT NONE
      INTEGER::op,inum,i,k,j,ierr,iV1(n)
      REAL(KIND=KD):: UL(n),UR(n),F(n)
      REAL(KIND=KD)::AL(n,n),AR(n,n),Aa(n,n),Aabs(n,n),FL(n),FR(n),dU(n),wave(n)
      REAL(KIND=KD)::gam(n,n),WR(n),WI(n),R(n,n),Rm1(n,n),FV1(n),C(n,n)
      REAL(KIND=KD)::nor(n),nor1,FF(n),ef(n),perc
!
!     Averaging
!
      CALL Averag(UL,UR)
      CALL RoeMat(Aa)
!
!     Find the eigenvalues and eigenvectors
!
      CALL RG(n,n,Aa,WR,WI,1,R,IV1,FV1,ierr)
!
!     Set the eigenvalue diagonal matrix
!
      CALL szero(n,gam)
      DO i=1,n
            IF (op==2) THEN
                  gam(i,i)=min(0.,WR(i))
            ELSEIF (op==1) THEN
                  gam(i,i)=max(0.,WR(i))
            ENDIF
      ENDDO
!
!     calculate inverse matrix of R(eigenvector matrix)
!
      CALL InvMat(R,n,Rm1)
!
!     Solve [R][|Λ|][R^(-1)]
!
      CALL MMM(n,R,gam,C)
```

```
        CALL MMM(n,C,Rm1,Aabs)
!
!       dU
!
        DO i=1,n
            dU(i)=UR(i)-UL(i)
        ENDDO
!
!       Wave
!
        CALL M1M(n,Aabs,dU,wave)
!
!       Calculate the flux vector at pt.(1-1/2)
        DO i=1,n
            F(i)=wave(i)
        ENDDO
!
        END SUBROUTINE RoeFlux
!
!=====================================================
        SUBROUTINE Averag(UL,UR)
!=====================================================
!
        USE mod_input,ONLY:sigma,KD,rhol
        USE mod_averag
!
        IMPLICIT NONE
        REAL(KIND=KD)::UL(n),UR(n)
        REAL(KIND=KD)::QL(n),QR(n)
        REAL(KIND=KD)::wb(n)
        REAL(KIND=KD)::allPL,allPR,Pbl
!
        CALL Qgen(UL,QL)
        CALL Qgen(UR,QR)
!
        allPL=QL(6)*(UL(2)/rhol)
        allPR=QR(6)*(UR(2)/rhol)
!
!       Averaging
!
        wb(1)=dsqrt(UL(1))+dsqrt(UR(1))
        wb(2)=dsqrt(UL(2))+dsqrt(UR(2))
        wb(3)=UL(3)/dsqrt(UL(1))+UR(3)/dsqrt(UR(1))
        wb(4)=UL(4)/dsqrt(UL(2))+UR(4)/dsqrt(UR(2))
        wb(5)=dsqrt(UL(1))*(UL(5)/UL(1)+QL(6)/(UL(1)/QL(1)))          &
              +dsqrt(UR(1))*(UR(5)/UR(1)+QR(6)/(UR(1)/QR(1)))
        wb(6)=dsqrt(UL(2))*(UL(6)/UL(2)+QL(6)/(UL(1)/QL(1)))          &
              +dsqrt(UR(2))*(UR(6)/UR(2)+QR(6)/(UR(1)/QR(1)))
!
        uvb=wb(3)/wb(1)
        ulb=wb(4)/wb(2)
        hvb=wb(5)/wb(1)
        hlb=wb(6)/wb(2)
!
        allb=2./(1./(UL(2)/rhol)+1./(UR(2)/rhol))
```

```
        alvb=1-allb
        pb=(allPL+allPR)/((UL(2)/rhol)+(UR(2)/rhol))
!
        Pbl=pb
        amb=(((gamma-1)*(hvb-uvb**2*0.5d0)+gamma*(allb/alvb)*(Pbl/rhol)))
        delb=sigma*gamma*(allb*Pbl)/(alvb*rhol*amb)
!
        END SUBROUTINE Averag
!
!=====================================================
        SUBROUTINE RoeMat(A)
!=====================================================
!
        USE mod_averag
!
        IMPLICIT NONE
        REAL(KIND=KD)::A(n,n)
!
        CALL SZERO(n,A)
!
        A(1,3)=1.
        A(2,4)=1.
!
        A(3,1)=(gamma-3.)*uvb**2
        A(3,2)=(Pb/rhol)-delb*alvb*(uvb-ulb)**2
        A(3,3)=(3.-gamma)*uvb
        A(3,5)=gamma-1.
!
        A(4,1)=(gamma-1.)*(allb/alvb)*(uvb**2*0.5d0)
        A(4,2)=(allb/alvb)*(Pb/rhol)-ulb**2+delb*alvb*(uvb-ulb)**2
        A(4,3)=(1.-gamma)*(allb/alvb)*uvb
        A(4,4)=2*ulb
        A(4,5)=(gamma-1.)*(allb/alvb)
!
        A(5,1)=((gamma-1.)*(uvb**2*0.5d0)-hvb)*uvb
        A(5,3)=hvb-(gamma-1.)*uvb**2
        A(5,4)=Pb/rhol
        A(5,5)=gamma*uvb
!
        A(6,1)=(gamma-1.)*(allb/alvb)*(uvb**2*0.5d0)*ulb
        A(6,2)=(Pb/(alvb*rhol)-hlb)*ulb
        A(6,3)=(1.-gamma)*(allb/alvb)*uvb*ulb
        A(6,4)=hlb-Pb/rhol
        A(6,5)=(gamma-1.)*(allb/alvb)*ulb
        A(6,6)=ulb
!
        END SUBROUTINE RoeMat
!
!=====================================================
        SUBROUTINE InvMat(P,np,Y)
!=====================================================
!
        IMPLICIT NONE
        INTEGER::i,j,n,np,indx(np)
        REAL*8,INTENT(in)::P(np,np)
```

```
        REAL*8::A(np,np),Y(np,np),d
!
        n=np
        A=P
!
!       identity matrix
!
        CALL SZERO(np,y)
        DO i=1,n
                y(i,i)=1.
        ENDDO
!
!       LU decomposition
!
        CALL ludcmp(A,n,np,indx,d)
!
        DO i=1,n
                CALL lubksb(A,n,np,indx,Y(:,i))
        ENDDO
!
        END SUBROUTINE InvMat
!
!=======================================================
        SUBROUTINE BC(k,U,Unew,dU,Uori)
!=======================================================
!
        USE mod_input,ONLY:imax,n,KD,bc_in,bc_out,bc_uv,bc_ul,bc_p
        USE mod_prop,ONLY:dx,dt
!
        IMPLICIT NONE
        INTEGER::i,j,k
        REAL(KIND=KD)::U(n,imax),dU(n,imax),Uin(n),Uout(n)
        REAL(KIND=KD)::Uori(n,imax),Unew(n,imax)
        REAL(KIND=KD)::Fp(n),Fm(n),dFdx(n),H(n),QR(n),QL(n)
!
        CALL Read_BC_option
!
!       Inlet
!
        i=1
!
        IF (bc_in==1) THEN
                Uin=U(:,i)
        ELSEIF (bc_in==2) THEN
                CALL Qgen(U(:,i),QL)
                CALL Read_velocity_inlet
                QL(2)=bc_uv
                QL(3)=bc_ul
                CALL Ugen(QL,Uin)
        ELSE
                PRINT *,'Inlet boundary condition is in error'
        ENDIF
!
        CALL RoeFlux(Uin,U(:,i),Fp,1,i)
        CALL RoeFlux(U(:,i),U(:,i+1),Fm,2,i)
```

```
        DO j=1,n
            dFdx(j)=(Fm(j)+Fp(j))/dx
            dU(j,i)=dt(i)*(-dFdx(j))
            Unew(j,i)=Uori(j,i)+(1./(5.-k))*dU(j,i)
        ENDDO
!
!       Outlet
!
        i=imax
!
        IF (bc_out==1) THEN
            Uout=U(:,i)
        ELSEIF (bc_out==2) THEN
            CALL Qgen(U(:,i),QR)
            CALL Read_pressure_outlet
            QR(6)=bc_p
            CALL Ugen(QR,Uout)
        ELSE
            PRINT *,'Outlet boundary condition is in error'
        ENDIF
!
        CALL RoeFlux(U(:,i-1),U(:,i),Fp,1,i)
        CALL RoeFlux(U(:,i),Uout,Fm,2,i)
        DO j=1,n
            dFdx(j)=(Fm(j)+Fp(j))/dx
            dU(j,i)=dt(i)*(-dFdx(j))
            Unew(j,i)=Uori(j,i)+(1./(5.-k))*dU(j,i)
        ENDDO
!
        END SUBROUTINE BC
!
!=====================================================
        SUBROUTINE Source(U,H)
!=====================================================
!
        USE mod_input,ONLY:n,rhol,KD
!
        IMPLICIT NONE
        REAL(KIND=KD):: U(n),H(n)
!
        H(1)=0.0D0
        H(2)=0.0D0
        H(3)=U(1)*9.81
        H(4)=U(2)*9.81
        H(5)=0.0D0
        H(6)=0.0D0
!
        END SUBROUTINE Source
!
!=====================================================
        SUBROUTINE Qgen(U,Q)
!=====================================================
!
        USE mod_input,only:n,KD,rhol,gamma
!
```

```
        IMPLICIT NONE
        REAL(KIND=KD)::U(n),Q(n)
!
        Q(1)=1.-U(2)/rhol
        Q(2)=U(3)/U(1)
        Q(3)=U(4)/U(2)
        Q(4)=U(5)/U(1)-Q(2)**2/2.
        Q(5)=U(6)/U(2)-Q(3)**2/2.
        Q(6)=(gamma-1)*Q(4)*(U(1)/Q(1))
!
        END SUBROUTINE Qgen
!
!=====================================================
        SUBROUTINE Ugen(Q,U)
!=====================================================
!
        USE mod_input,only:n,KD,rhol,gamma
!
        IMPLICIT NONE
        REAL(KIND=KD)::U(n),Q(n)
!
        U(1)=Q(1)*Q(6)/((gamma-1.)*Q(4))
        U(2)=(1-Q(1))*rhol
        U(3)=U(1)*Q(2)
        U(4)=U(2)*Q(3)
        U(5)=U(1)*(Q(4)+Q(2)**2/2.)
        U(6)=U(2)*(Q(5)+Q(3)**2/2.)
!
        END SUBROUTINE Ugen
!
```

A.3. Input Files

A.3.1. Shock Tube Problem

```
$property
    gamma=0.10931E+01, rhol=720.
$end
$solution
    CFL=0.2, err=0.001, sigma=1.01
$end
$geometry
    L=1.
$end
$option
    dtcase=1, src=0
$end
$left_state
    alphaL=0.25, PL=20e6, uvL=0., ulL=0., hvL=3.0927e6, hlL=1.3382e6
```

```
$end
$right_state
      alphaR=0.25, PR=15e6, uvR=0., ulR=0., hvR=3.0927e6, hlR=1.3382e6
$end
$boundary_condition
           bc_in=1, bc_out=1
$end
$velocity_inlet
           bc_uv=0.0D0, bc_ul=10.0D0
$end
$pressure_outlet
           bc_p=1E5
$end
$print_results
           rtype=1
$end
```

A.3.2. Shock Tube Problem

```
$property
      gamma=0.1327E+01, rhol=999.
$end
$solution
      CFL=0.9, err=0.001, sigma=1.01
$end
$geometry
      L=12.
$end
$option
      dtcase=1, src=1
$end
$left_state
      alphaL=0.2, PL=1e5, uvL=0., ulL=10., hvL=2.67546e6, hlL=0.41744e6
$end
$right_state
      alphaR=0.2, PR=1e5, uvR=0., ulR=10., hvR=2.67546e6, hlR=0.41744e6
$end
$boundary_condition
     bc_in=2, bc_out=2
$end
$velocity_inlet
     bc_uv=0.0D0, bc_ul=10.0D0
$end
$pressure_outlet
     bc_p=1E5
$end
$print_results
     rtype=2
$end
```

서 지 정 보 양 식			
수행기관보고서번호	위탁연구기관보고서번호	표준보고서번호	INIS 주제코드
KAERI/TR-3637/2008			

제 목 / 부 제	1차원 쌍곡선 방정식을 이용한 이상유동해석 코드 개발
주 저 자	김의진
공 저 자	김종태(열수력안전연구부), 정재준

발행지	대 전	발행 기관	한국원자력연구원	발행일	2008년 8월
면 수	47 p	도 표	유 (O), 무 ( )	크 기	A4
참고사항	2008 원자력연구개발과제; 고정밀 열수력 전산수치해석 기술개발				
비밀여부	공개 (O), 대외비 ( ), -----비밀			보고서종류	기술보고서
위탁연구기관	교육과학기술부			계약번호	

초 록

본 연구는 액체/기체 혼합체의 이상유동(two-phase flows)에 관한 기초적인 연구를 수행하여 이상유동 해석에 필요한 해석코드를 개발하는데 목적이 있다. 이를 위하여 상류차분법(upwind difference scheme) 중의 하나인 Roe Scheme을 채택하였다. 본 연구에서는 대류항의 계산을 위해 상류차분법 중 확장된 Roe Scheme을 이용하여 1차원 이상유동을 모사하였으며, 기초 연구로서 단면적이 일정한 경우의 유동만 계산할 수 있도록 하였다. 이에 따라 Roe-averaged matrix를 얻음으로써 충격파 관과 수도밸브 문제의 이상유동에 관한 두 가지 예시 계산을 수행하였다. 이를 통해 일반적인 상류차분법과 마찬가지로 본 연구에서 사용한 확장된 Roe의 기법 역시 충격파와 같은 불연속면을 포획하는 특성이 있다는 것을 알 수 있다.

주제명 키워드	이상유동, Roe 상류차분법

BIBLIOGRAPHIC INFORMATION SHEET					
Performing Org. Report No.		Sponsoring Organization Report No.		Standard Report No.	INIS Subject Code
KAERI/TR-3637/2008					
Title/Subtitle	Development of One Dimensional Hyperbolic Coupled Solver for Two-Phase Flows				
Main Author	Eoi-Jin Kim				
Author	Jongtae Kim(Division of thermal hydraulics and safety research), Jae June Jeong				
Pub. Place	Taejon	Pub. Org.	KAERI	Pub. Date	August 2008
Page	47 p	Fig. & Tab.	Yes (O), No ( )	Size	A4
Note	2008 Nuclear Mid-Long Term Project				
Classified	Open (O), Outside ( ), ----- Class			Report Type	Technical Report
Sponsoring Org.	Ministry of Education, Science and Technology			Contract No.	

Abstract

The purpose of this study is a code development for one dimensional two-phase two-fluid flows. In this study, the computations of two-phase flow were performed by using the Roe scheme which is one of the upwind schemes. The upwind scheme is widely used in the computational fluid dynamics because it can capture discontinuities clearly such as a shock. And this scheme is applicable to multi-phase flows by the extension methods which were developed by Toumi, Stadtke, etc. In this study, the extended Roe upwind scheme by Toumi for two-phase flow was implemented in the one-dimensional code. The scheme was applied to a shock tube problem and a water faucet problem. This numerical method seems efficient for non oscillating solutions of two phase flow problems, and also capable for capturing discontinuities.

Subject Keywords	Two phase flow, Roe upwind scheme