응용수치해석

전산금융공학 (Computational Finance Engineering)

전산유체역학 (Computational Fluid Dynamics)

PC 클러스터 구축 및 병렬계산 (PC Cluster Building and Parallel Computing)

정다래 ◦ 김성기 ◦ 김준석

2010-01-26

머리말

응용수치해석은 기초적인 수치해석을 활용하여 전산금융공학과 전산유체역학을 학습한다. 또한 PC 클러스터를 직접 구축하여 병렬계산을 수행한다. 본 저서는 다음과 같이 구성되어있다.

제 1장은 MATLAB 기초로 본 저서를 학습하는데 있어서 필요한 기본적인 명령어들을 학습한다.

제 2장은 KOSPI200 스톡옵션 가격 결정 모델에 대한 유한차분법에 관해서 학습한다.

제 3장은 유체 유동의 지배 방정식인 Navier-Stokes 방정식에 대한 수치해석이다.

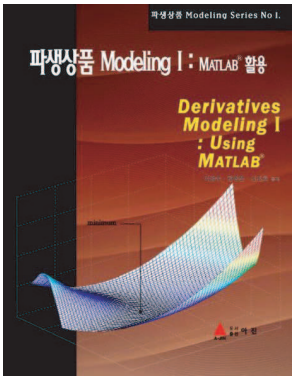
제 4장은 두개의 PC에 병렬계산을 할수있는 시스템 구축이다. 전세계 각국의 기상예보에는 수퍼컴퓨터를 사용한다. 수퍼컴퓨터의 기본은 병렬계산으로 한번에 대용량을 계산을 수행하는데 적합하다. 간단한 예제를 통해서 병렬계산이 어떻게 이루어 지는지 학습한다.

정다래 (tinayoyo@korea.ac.kr)
김성기 (zeqcads@naver.com)
김준석 (cfdkim@korea.ac.kr, `http://math.korea.ac.kr/~cfdkim`)

This work was supported by Seoul R&BD Program(10551). 본 저서는 또한 고려대학교 계량 금융 기술 연구소의 지원을 받았습니다.

참고도서

계산재무론 (최병선)
계산재무론의 필요성을 시작으로 금융파생상품의 가치평가, 계산재무기법들, 나무모형을 사용한 가치평가, 편미분방정식을 사용한 가치평가, 몬테카를로법을 사용한 가치평가 등을 차례대로 설명한다.



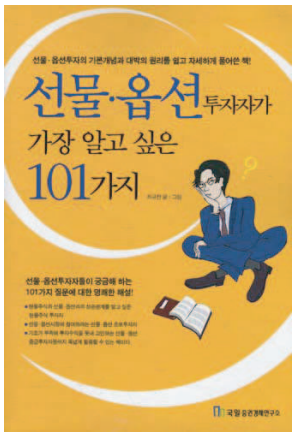
파생상품 Modeling I MATLAB활용 (이경수 권명은 신진호)
파생상품 평가와 리스크측정에 대한 전문지식을 체계적으로 설명하고 이를 MATLAB을 이용하여 실무에 구현하는 방법을 소개하였다.

금융 증권을 위한 블랙숄즈의 편미분방정식 (김완세 옮김)

금융 증권을 위한 블랙 숄즈 미분 방정식 입문서. 미분과 편미분, 테일러 급수 전개, 적분과 무한적분, 푸리에 해석과 편미분방정식해의 공식 등의 내용을 담았다.

선물옵션 (이필상)

선물, 옵션, 스왑과 같은 파생 금융상품에 대한 전반 적이고 체계적인 내용을 다룬 전공서. 선물의 개념, 선물시장의 구조와 운영, 금리/통화/주가지수 선물, 스왑, 옵션시장의 구조와 운영 등 14개 장으로 설명하고 각 장마다 연습문제를 실었다.



선물,옵션 투자자가 가장 알고 싶은 101가지 (최규찬)

투자자들이 알고 싶어하는 선물 옵션거래의 핵심사항들을 간단명료하게 해설하고 있다. 투자자들이 궁금해 하는 점들을 구체적으로 예시하고 그에 대한 답변을 제시함으로써 선물 옵션거래의 원리와 기법을 쉽고 정확하게 이해할 수 있도록 했다.

차 례

제 1 장

MATLAB 기초

MATLAB(www.mathworks.com)은 미국의 Math Works에서 만들어진 프로그램으로, 1984년도에 소개된 이후로 오늘날 전 세계 50만 이상이 사용하고 있다. MATLAB은 `MATrix+LABoratory`로서 행렬을 기본으로 최적화되어진 프로그램으로 알고리즘 개발, 데이터 수치분석이나 시각화를 위한 컴퓨터 언어이다. C언어에 비해 사용하기가 편리하다는 장점이 있으나 실행속도가 느리다는 단점을 안고 있다.

새로운 장외파생상품들의 출현으로 주로 공학계통에서 사용되던 MATLAB은 편리한 사용방법으로 최근 모델링을 위한 프로그래밍 언어로 인기를 얻어가고 있으며 이 책에서도 모든 코드는 MATLAB언어로 구현하였다.

특히 MATLAB의 Financial Derivatives Toolbox는 금융 데이터 분석, 모델링, 시뮬레이션 및 최적화를 위한 MATLAB 및 툴박스로 더 자세한 내용은 홈페이지를 참조하기 바란다.

제 1 절 MATLAB 기초

```
>> a = 1   Enter          a에 1 대입
a = 1
```

```
>> a=1;b=2,c=3;   Enter
b =
     2
```

$a = 1$, $b = 2$, $c = 3$값이 대입되지만 세미콜론(;)이 붙은 a와 c는 화면에 출력 되지 않고 ;이 붙지 않은 b만 화면에 출력된다.

```
>> d=[1 2 3;4 5 6]   Enter
d =
     1     2     3
     4     5     6
```

d는 2행 3열의 행렬이 된다.

```
>> d(1,3)   Enter
ans =
     3
```

d의 1행 3열 원소값을 보여준다.

```
>> d(2,3)   Enter
ans =
     6
```

d의 2행 3열 원소값을 보여준다.

```
>> 2*3   Enter
ans =
     6
```

2×3의 계산값을 보여준다.

```
>> ans   Enter
ans =
     6
```

마지막으로 계산된 ans값을 보여준다. ans는 마지막 값을 가지고 있다.

```
>> ans+6   Enter
ans =
     12
>> clear   Enter
```

마지막으로 계산된 ans값에 $+6$ 연산을 한다. 이 후, `clear`하면 ans값은 사라진다.

```
>> a=2, A = 3;      Enter
>> a
a =
     2
```

MATLAB은 대소문자를 구분한다.

```
>> A=[1 2 3;4 5 6;7 8 9]; Enter
```

```
>> sum(A)  Enter
ans =
    12    15    18
```

A행렬의 각 열의 원소들의 합.

```
>> sum(A')  Enter
ans =
     6    15    24
```

A행렬의 각 행의 원소들의 합. 즉, A의 transpose행렬의 각 열의 원소들의 합.

```
>> sum(sum(A))  Enter
ans =
     45
```

`sum(A)`의 각 원소들의 합.

```
>> b=[1 2 3];  Enter
>> sum(b)  Enter
ans =
     6
```

b행렬의 각 원소들의 합.

```
>> A=[1 2;3 4]
A =
     1     2
     3     4
>> B=[5 6;7 8]
B =
     5     6
     7     8
```

A행렬과 B행렬의 정의

다음은 각각의 행렬에 대한 사칙연산이다.

MATLAB 명령어	연산 의미
`A*B`	$AB = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \begin{pmatrix} 5 & 6 \\ 7 & 8 \end{pmatrix} = \begin{pmatrix} 19 & 22 \\ 43 & 50 \end{pmatrix}$
`A/B` or `A*inv(B)`	$AB^{-1} = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \begin{pmatrix} -4 & 3 \\ 3.5 & 2.5 \end{pmatrix} = \begin{pmatrix} 3 & -2 \\ 2 & -1 \end{pmatrix}$
`A\B` or `inv(A)*B`	$A^{-1}B = \begin{pmatrix} -2 & 1 \\ 1.5 & -0.5 \end{pmatrix} \begin{pmatrix} 5 & 6 \\ 7 & 8 \end{pmatrix} = \begin{pmatrix} -3 & -4 \\ 4 & 5 \end{pmatrix}$
`A^2` or `A*A`	$A^2 = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} = \begin{pmatrix} 7 & 10 \\ 15 & 22 \end{pmatrix}$

위에서 행렬과 행렬사이에 사용한 * 연산자는 내적(Inner Product)연산자이다. 따라서 곱하는 두 행렬의 크기가 $(n,m) \times (m,k) = (n,k)$이어야 한다. 또한 / 연산자는 역행렬을 곱하는 것으로 연산하고자 하는 두 행렬의 크기는 같으며 정방행렬이어야 한다. 이제 위에서 본 연산자 앞에 .을 찍으면 어떤 결과가 나오는지 살펴보자.

MATLAB 명령어	연산 의미
`A.*B`	$\begin{pmatrix} 1 \cdot 5 & 2 \cdot 6 \\ 3 \cdot 7 & 4 \cdot 8 \end{pmatrix} = \begin{pmatrix} 5 & 12 \\ 21 & 32 \end{pmatrix}$
`A.^B`	$\begin{pmatrix} 1^5 & 2^6 \\ 3^7 & 4^8 \end{pmatrix} = \begin{pmatrix} 1 & 64 \\ 2187 & 65536 \end{pmatrix}$

연산자 앞에 .이 붙게 되면 행렬의 같은 위치에 있는 각각의 원소끼리 연산을 수행하라는 의미이다. 원소끼리의 연산이므로 반드시 두 행렬의 크기는 정확하게 일치해야 한다.

항목	명령어	기능
명령어 나열	,	두 개 이상의 명령어를 한 줄에 표현하려면 콤마(,)를 이용하여 구분

줄넘기기	...	명령어가 너무 길어 다음 줄로 넘어가고 싶을 때 사용 (Command Window에서도 사용가능)
		(예) `plot(x,y,'--rs','LineWidth',2,...`
		`'MarkerEdgeColor','k',...`
		`'MarkerSize',10)`

출력여부	;	Command Window상에서 명령어를 실행하면 화면에 결과가 출력되지만, 세미콜론(;)을 입력하여 실행하면 출력되지 않고 workspace에만 저장된다. (예) `>> a=1;b=2;c=3;`
주석	%	명령어의 앞에 %를 입력하면 주석으로 처리된다.
화면정리	clc	clc 명령어를 입력하고 엔터를 치면 Command Window에 표시되었던 모든 내용들이 지워짐
화면정리	clf	clf 명령어를 입력하고 엔터를 치면 Figure에 나타난 모든 그림이 지워짐
변수삭제	clear	변수 및 배열에 할당된 값들 모두 삭제. (whos로 확인 가능) • 모든 변수를 삭제 : `clear all` • 특정 변수만을 삭제 : `clear 특정변수`

제 2 절 M-file 만들기

MATLAB을 이용하여 원하는 기능을 수행하는 방법은 크게 두 가지로 구분된다.

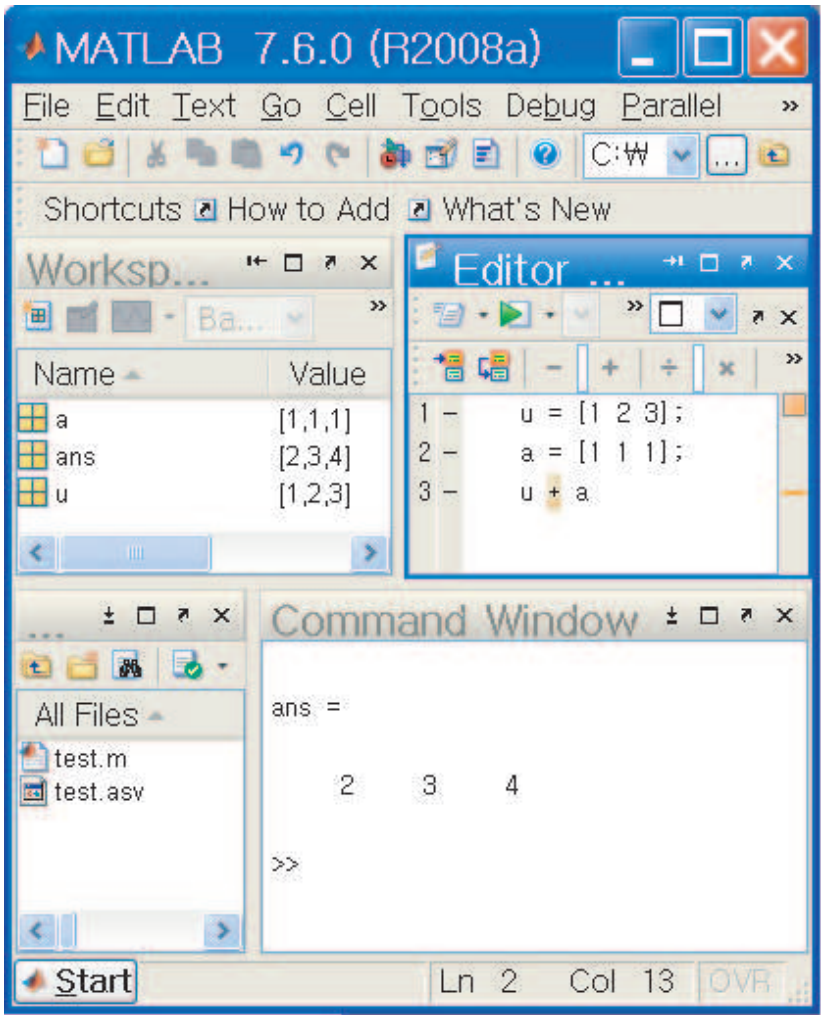


그림 1.1: MATLAB창

첫 번째는 Command Window에 직접 명령어를 입력하는 방법이고, 두 번째는 Script파일을 이용하는 것이다.

MATLAB에서 사용하는 파일을 보통 M-file이라고 부르며 파일의 확장

제 3 절 for ∼ end 문

'for' 문은 'end'문과 짝을 이루어 사용된다. 'for'문과 같은 행에 있는 변수의 값을 초기값부터 증분의 크기만큼 누적시키면서 최종값에 도달된 때까지 'for'문과 'end'문 사이 문장의 명령을 수행한다. 증분이 '1'인 경우에는 '증분:'을 생략해도 무방하다.

```
for 변수명=초기값:(증분:)최종값
    문장
end
```

[예제] for ∼ end 프로그램

```
for x=0:0.5:1
  a=2^x
end
for k=5:-2:1
  b=k
end
```

[프로그램 실행결과]

```
a = 1
a = 1.4142
a = 2
b = 5
b = 3
b = 1
```

수행한다.

while 조건

문장

end

[예제] while ∼ end 문 프로그램

```
a=1;
while a<4
    a=a+1
end
```

[프로그램 실행결과]

```
a =  2
a =  3
a =  4
```

제 6 절 linspace 문

'linspace'는 a와 b사이의 간격이 동일한 n개의 벡터를 만드는 데 사용한다. 다음과 같이 이용하며 이에 대한 예제를 살펴보자.

linspace(a,b,n)

linspace(시작점,끝점,점의 총수)

[예제] linspace 문 프로그램

색상		모양		라인	
b	Blue	.	Point	-	Solid
g	Green	o	Circle	:	Dotted
r	Red	x	x mark	-.	Dashdot
c	Cyan	+	Plus	–	Dashed
m	Magenta	*	Star	(none)	No line
y	Yellow	s	Square		
k	Black	d	Diamond		
w	white	v	Triangle(down)		
		^	Triangle(up)		

그림 1.2: Plot 명령어의 옵션

예를 들어, `plot(x,sin(x),'k--',x, cos(x),'ko')`를 실행하면, 다음 결과를 얻게 된다. 이는 y축의 값을 $\sin(x)$, $\cos(x)$로 하는 두 개의 그래프

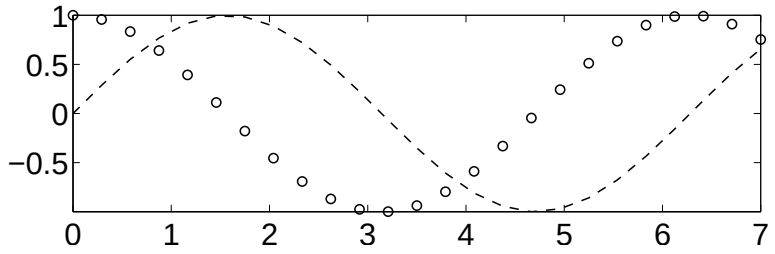


그림 1.3: plot문 옵션을 이용한 프로그램 실행결과

를 나타내며, 첫번째 $\sin(x)$는 검은 색의 점선으로, $\cos(x)$는 검은색 원으로 표현된다. (그림 1.3 참고)

제 2 장

전산금융공학 (Computational Finance Engineering)

제 1 절 파생금융상품

파생금융상품은 기초자산의 가격변동으로 인한 손실위험을 제거하기 위해 탄생했다. 옵션계약의 대상이 되는 상품을 **기초자산**이라 한다. **옵션** (option)이란 특정 자산을 미리 정해진 가격으로 미래의 일정한 시기에 매수 또는 매도할 권리가 내재된 계약을 일컫는다. 이를 유러피언 옵션(European option)이라 한다. 이 때 특정자산을 매수할 권리를 **콜옵션**(call option), 매도할 수 있는 권리를 **풋옵션**(put option)이라고 한다.

옵션매입자(option buyer)는 옵션발행자(option seller)에게 일정한 프리미엄 (premium)을 지급하고 권리를 매입한 사람을 말한다. 옵션매입자는 옵션권리의 취득에 의해 옵션 매도자에게 정해진 기간에 옵션계약의 내용에 대한 이행을 청구할 수 있으며, 반대로 자신에게 불리한 경우 권리의 행사를 포기할 수 있다.

반면에 옵션발행자는 옵션매입자에게 권리를 양도하고 일정한 프리미엄을 받는다. 그에 대한 의무로 만기에 옵션매입자가 미리 정해진 조건에 자산을 매입하고자 하는 경우(콜옵션)나 매도하고자 하는 경우(풋옵션), 거래의 상대방이 되어야 할 의무가 발생한다.

우는 1,000원, 즉 0.01포인트이다. 옵션거래는 가격제한폭이 없지만 시장안정을 위해서 호가가격이 전일의 대상자산 가격대비 ±15%를 벗어나는 경우 거래소에서 호가접수를 거부하는 호가한도 가격제도를 두고 있다.

우리나라 주가지수 옵션거래는 주식시장의 개장 시간인 9시에 거래를 시작하며 주식시장 종료시점보다 15분 늦은 3시 15분에 거래가 종료된다. 그러나 결제일에는 주식시장보다 10분 일찍 2시 50분에 거래를 종료하도록 되어 있다.

매수호가와 매도호가가 일치하면 거래가 체결된다. A가 10계약 매수주문을 내고 동일한 금액으로 B가 30계약 매도주문을 내게 되면, 주문이 일치된 10계약이 이루어지는데 이 때 거래가 체결된 10계약이 거래량이 된다.

행사가격은 전날의 KOSPI 200 종가에 가장 가까운 행사가격과 2.5포인트씩 높인 행사가격 4개 그리고 2.5포인트씩 낮춘 행사가격 4개를 제시함으로써 모두 9개 행사가격을 가진 옵션이 거래된다. 단, 3, 6, 9, 12월이 결제일인 옵션은 5포인트 간격으로 5개의 행사가격이 설정된다.

예를 들어 2010년 3월 11일의 KOSPI 200 지수가 109.30이었다면 3월 12일 새로이 설정되는 9월 결제(6개월 만기)옵션 행사가격의 종류는 110.0과 115.0, 120.0 그리고 105.0, 100.0의 5개가 된다. 이후 주가의 상승 또는 하락에 따라 계속해서 새로운 행사가격의 종목이 추가 설정된다.

호가방법은 회원인 증권회사가 각 지점의 주문내용을 증권사의 전산시스템을 이용하여 거래소 옵션거래시스템에 전달하거나 증권회사의 영업장소에 설치된 호가입력단말기에 직접 입력하는 방법을 사용한다. 호가의 내용에는 지정가 또는 시장가주문인지를 표시하는 주문유형과 매수 혹은 매도의 구분, 가격, 수량, 위탁매매 혹은 자기매매의 구분, 고객계좌번호, 투자자구분 등이 기록된다.

우리나라의 옵션시장은 전산시스템에 의한 개별경쟁매매방식을 채택하고 있다. 매매방식에는 가격우선, 시간우선, 수량우선의 원칙이 적용된다.

증권선물거래소는 옵션거래 전산시스템에 장애가 발생하여 10분 이상 매매거래가 불가능하거나, KOSPI 200 구성종목의 절반 이상이 거래가 이루어지지 않는 경우에 옵션의 매매거래를 일시 중단할 수 있다. 또한 선물

회사는 임의로 반대매매를 하거나 예탁된 대용증권을 매각할 수 있다.

미결제약정이란 옵션거래가 성립된 이후 만기일까지 반대매매, 권리행사, 또는 최종결제 등으로 청산되지 않은 약정을 말한다. 매수 미결제 약정을 보유하고 있는 것을 "매수 포지션(Long Position)을 취하고 있다." 라고 말하고 매도 미결제 약정을 보유하고 있는 것을 "매도 포지션(Short Position)을 취하고 있다." 라고 말한다.

증권회사는 투자자의 위탁주문에 대한 매매거래가 성립하거나 최종결제 또는 권리행사에 의한 결제가 발생한 때 투자자로부터 증권회사가 정한 위탁수수료를 징수한다. 구입한 옵션을 만기일까지 보유하지 않고, 도중에 반대 매매하여 손익을 확정짓는 것이 중간 정산입니다.

1.1 KOSPI 200 옵션 시세표 읽는 법

그림 2.1는 야후금융 옵션시세에서 발췌한 것으로, 다음의 사이트를 방문하여 정보를 얻었다.

http://kr.stock.yahoo.com/sise/idx202.html

이제, 그림 2.1을 보면서 각각의 용어들의 의미를 정리해보자.

- 현재가 (207.44) : 현재 지수
- 전일비 (▼ 2.30) : 전일 종가에 비교한 값으로 (현재가 - 전일종가)
- 전일지수는 207.44 + 2.30 = 209.74
- 등락률 (1.09%) : 전일 종가를 기준으로 오르고 내린 정도로

$$\frac{\text{전일비}}{\text{전일종가}} \times 100\% = \frac{2.30}{209.74} \times 100\% = 1.0966\%$$

- 시가 (209.43) : 당일 최초로 형성된 지수
- 고가 (209.99) : 하루 중 가장 높은 지수
- 저가 (205.80) : 하루 중 가장 낮은 지수
- 행사가격 (210.00) : 옵션매입자가 만기일 또는 그 이전에 권리를 행사할 때 적용되는 가격

행사가격(210.00)에 대한 콜옵션 매수가는 9.40이다. 예를 들어, A가 7계

과 같이 정의하자.

$$\mathcal{E}[X] = \int_{-\infty}^{\infty} xf(x)dx = \int_{-\infty}^{\infty} \frac{x}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} dx. \tag{2.1}$$

한편, $\mu = 0$이고 $\sigma = 1$인 경우를 확률변수 X가 표준정규분포를 따른다고 하며, 이때 확률밀도함수는 다음과 같다.

$$\phi(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}.$$

또한 $X \sim N(0,1)$로 표기 할 수도 있다. 확률밀도함수(probability density function)를 간단히 기호 pdf로 표시한다.

만일 $X \sim N(\mu, \sigma^2)$이면, 다음 식들이 성립한다.

$$\mathcal{E}[X] = \mu \tag{2.2}$$

$$Var[X] = \sigma^2 \tag{2.3}$$

$$M_X(t) = \exp\left(\mu t + \frac{\sigma^2}{2} t^2\right) \tag{2.4}$$

Proof. 다음 식을 정의하자.

$$M_X(t) = \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(tx - \frac{(x-\mu)^2}{2\sigma^2}\right) dx.$$

이 식에서 $\exp(\cdot)$ 함수의 지수 항은 다음 식을 만족한다.

$$tx - \frac{(x-\mu)^2}{2\sigma^2} = -\frac{1}{2\sigma^2}\left(x - (\sigma^2 t + \mu)\right)^2 + \mu t + \frac{\sigma^2}{2} t^2.$$

여기서 $y = x - (\sigma^2 t + \mu)$라고 하자. 그러면, 다음 식을 만족한다.

$$\int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{\{x - (\sigma^2 t + \mu)\}^2}{2\sigma^2}\right] dx = \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{y^2}{2\sigma^2}\right) dy.$$

이제 I를 다음과 같이 정의하자.

$$I = \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{y^2}{2\sigma^2}\right) dy.$$

Brown 운동

다음 조건들을 만족하는 확률과정 $\{S(t)|t \leq 0\}$을 Brown운동이라고 부른다.

- $S(0) = 0$
- 임의의 $0 \leq t_1 < t_2 < t_3 < \cdots$에 대해서 $S(t_1), S(t_2)-S(t_1), S(t_3)-S(t_2), \cdots$는 서로 독립이다.
- 만일 $0 \leq \alpha \leq \beta$이면, $S(\beta) - S(\alpha)$는 평균이 0이고 분산이 $\sigma^2(\beta - \alpha)$인 정규분포를 따른다. 이 σ를 변동성(volatility)라 한다.
- 확률과정 $\{S(t)\}$에서 실현된 표본경로는 t의 연속함수이다.

σ가 1이면, 이 확률과정을 표준 Brown운동이라 한다.

일반화 Brown 운동

만일 확률변수 $S(t)$가 위의 Brown 운동의 첫번째, 두번째, 그리고 세번째 조건을 만족하고 평균이 μt이고 분산이 $\sigma^2 t$인 정규분포를 따르면 일반화 Brown 운동이라 부른다. 이 μ를 추세모수(drift parameter)라 한다.

일반화된 Brown 운동에서 파생된 기하 Brown 운동은 다음과 같이 정의된다.

기하 Brown 운동(Geometric Brownian Motion, GBM)

일반화된 Brown 운동 $\{S(t)|t \leq 0\}$에 대해서, 식 $Z(t) = e^{S(t)}$로 정의된 $\{Z(t)|t \leq 0\}$를 기하 Brown운동이라 한다.

효율적 시장가설(effective market hypothesis)은 시장에서 기본적 정보와 기술적 정보를 포함한 과거의 모든 정보가 금융자산의 가격에 반영되어 있다고 가정한다. 먼저, 자산 가격에서 절대적인 변화는 그 자체로서 유

확률미분방정식(stochastic differential equation)

$$\frac{dS}{S} = \mu dt + \sigma dX \tag{2.5}$$

을 얻게 된다. 자산 가격의 특징인 무작위성(randomness)을 포함하는 dX는 위너과정(Wiener process)을 따르고 다음의 성질을 갖는다;

- dX는 정규분포로부터 나온 확률 변수이다.
- dX의 평균은 0이다.
- dX의 분산은 dt이다.

ϕ가 평균 0 분산 1의 표준정규분포 $N(0,1)$을 따른다면 dX는 정규분포 $N(0,dt)$를 따르므로 $dX = \sqrt{dt}\phi$로 표현된다. $X(t)$는 미분가능이 아니므로 dX을 정의할 수는 없지만 $dt \to 0$으로 했을 때

$$dX = \sqrt{dt}$$

로 된다.

시계열 $S(t)$의 변화량 dS가 다음의 식에 따라 움직이고 있다고 하자.

$$dS = a(S,t)dt + b(S,t)dX.$$

일반화된 위너과정의 상수 a와 b를 S와 t에 대한 함수 $a(S,t), b(S,t)$로 일반화한 것을 이또과정이라고 하며, 따라서, 이 시계열 $S(t)$의 움직임을 이또과정이라 할 수 있다. 이또과정을 이용하면 이또의 보조정리(Itô lemma)를 이끌 수 있다.

이또의 보조정리

S가 이또과정

$$dS = a(S,t)dt + b(S,t)dX$$

를 따를 때, S와 t의 함수 $V(S,t)$의 동향은

$$dV = \left(\frac{\partial V}{\partial t} + \frac{1}{2}\frac{\partial^2 V}{\partial S^2}b^2(S,t) + \frac{\partial V}{\partial S}a(S,t)\right)dt + \frac{\partial V}{\partial S}b(S,t)dX$$

를 따른다.

이 되고, 이를 식 (2.7)에 대입하면

$$dV = \left(\mu - \frac{\sigma^2}{2}\right) dt + \sigma dX = \left(\mu - \frac{\sigma^2}{2}\right) dt + \sigma\sqrt{dt}\phi$$

를 얻게 된다.

제 2 절 옵션 가격 결정 모형의 Black-Scholes 편미분 방정식

Black-Scholes 편미분 방정식(partial differential equation:PDE) [**?**] 은 미국의 Fisher Black 교수와 Myron Scholes 교수에 의해 개발된 옵션 가격 결정 모형으로서 옵션 이론 가격을 산출할 때 일부 수정된 모델이 현재 널리 이용되고 있다. 이 모델을 이용하면 기초자산가격(S), 행사가격(E), 잔존기간(T), 무위험이자율(r), 기초자산가격의 변동성(σ)의 값들로 콜옵션과 풋옵션의 이론가격을 직접 계산할 수 있다.

옵션 이론가격를 구하기 위해서는 다음의 다섯 가지 변수를 반드시 알아야 한다.

(1) 기초자산 가격, S
(2) 행사가격, E
(3) 이자율, r
(4) 잔존기간, T
(5) 변동성, σ

Black-Scholes는 계산을 쉽게 하기 위해서 다음과 같은 몇 가지 가정을 했다.

- 잔존기간 동안 가격의 변동성과 무위험 이자율은 변하지 않는다.
- 거래비용과 세금은 일체 고려하지 않으며, 배당은 없는 것으로 본다.
- 기준물의 거래는 연속적으로 일어난다.

기초자산가격 S가 이또과정

$$dS = \mu S dt + \sigma S dX \tag{2.8}$$

을 도출하게 된다. 여기에 옵션의 만기시점에서 지불되는 지불금액함수를 경계조건으로 하면 다음의 식(2.14)을 만족하며, 그림 2.3를 따르게 된다.

$$V(S,T) = \begin{cases} S - E & \text{if } S \geq E \\ 0 & \text{if } S < E \end{cases} \tag{2.14}$$

식 (2.13)은 식(2.14)의 만기조건을 가지는 Black-Scholes의 편미분 방정식이다.

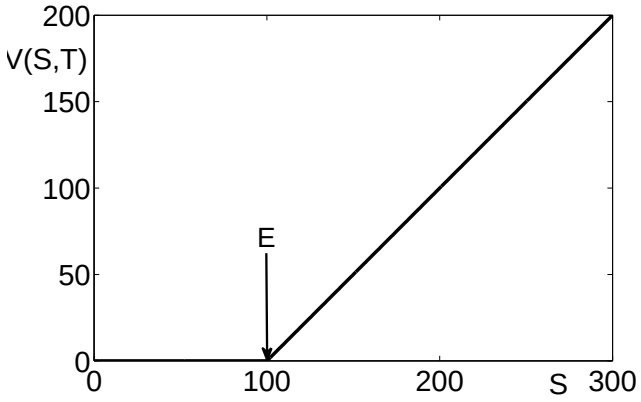


그림 2.3: 옵션의 만기시 지불금액함수

2.1 Black-Scholes 편미분방정식의 공식

이 절에서는 Black-Scholes 편미분방정식의 해를 구한다.[2]

다음과 같은 형태의 상미분 방정식을 생각해 보자.

$$g(y)\frac{dy}{dx} = f(x). \tag{2.15}$$

위 식의 양변에 x에 관해서 적분을 수행하면 다음을 얻는다.

[2]이 절에서 더 자세한 내용은 **금융증권을 위한 블랙숄즈의 편미분방정식(김완세 옮김)**의 책을 참고하길 바란다.

우선 방정식을 열 방정식(heat equation)형태로 전환하고 해를 구한 후 다시 치환을 통해서 원래 방정식인 Black-Scholes 편미분방정식의 해를 구한다. Black-Scholes 편미분방정식 (2.13)에 대해서 두 개의 변수 x와 τ를 도입하자

1번째 변수변환

$$x = \log\frac{S}{E} + \left(r - \frac{\sigma^2}{2}\right)(T-t) \tag{2.24}$$

$$\tau = T - t \tag{2.25}$$

2개의 변수 x와 τ를 사용하여, $V(S,t)$를 다음과 같이 표현하자.

$$V(S,t) = e^{-r\tau}u(x,\tau),$$

여기서 함수 $V(S,t)$를 이와 같이 표현하면 블랙 숄즈의 편미분방정식을 매우 간단한 편미분방정식으로 고쳐 쓸 수 있게 된다. $V_S(S,t)$, $V_{SS}(S,t)$, $V_t(S,t)$를 각각 계산하여 보자. 연쇄법칙(chain rule)을 사용하여

$$\begin{aligned}
V_S(S,t) &= V_x(S,t)x_S + V_\tau(S,t)\tau_S = \frac{\partial}{\partial x}\left(e^{-r\tau}u(x,\tau)\right)x_S \\
&= e^{-r\tau}u_x(x,\tau)S^{-1}. \\
V_{SS}(S,t) &= \frac{\partial}{\partial S}\left(e^{-r\tau}u_x(x,\tau)S^{-1}\right) = e^{-r\tau}\left(u_{Sx}(x,\tau)S^{-1} - u_x(x,\tau)S^{-2}\right) \\
&= \frac{e^{-r\tau}}{S^2}\left(u_{Sx}(x,\tau)S - u_x(x,\tau)\right) \\
&= \frac{e^{-r\tau}}{S^2}\left(\left[u_{xx}(x,\tau)x_S + u_{\tau x}(x,\tau)\tau_S\right]S - u_x(x,\tau)\right) \\
&= \frac{e^{-r\tau}}{S^2}\left(u_{xx}(x,\tau) - u_x(x,\tau)\right).
\end{aligned}$$

식 (2.30)로부터 다음 식이 성립한다.

$$\frac{\partial \widehat{u}(\lambda,\tau)}{\partial \tau} + \frac{\sigma^2\lambda^2}{2}\widehat{u}(\lambda,\tau) = 0. \tag{2.31}$$

식 (2.31)은 $\widehat{u}(\lambda,\tau)$의 τ에 관한 상미분 방정식이다. 식 (2.31)은 분리가능 방정식이라고도 하며, 이는 다음과 같이 나타낼 수 있다.

$$\frac{1}{\widehat{u}(\lambda,\tau)}\partial\widehat{u}(\lambda,\tau) + \frac{\sigma^2\lambda^2}{2}\partial\tau = 0.$$

c를 임의의 상수라 하고, 이제 위 식을 적분하면 다음의 식을 얻게 된다.

$$\begin{aligned}\int \frac{1}{\widehat{u}(\lambda,\tau)}\partial\widehat{u}(\lambda,\tau) + \int \frac{\sigma^2\lambda^2}{2}\partial\tau &= 0\\ \ln\widehat{u}(\lambda,\tau) + \frac{\sigma^2\lambda^2}{2}\tau &= c\\ \widehat{u}(\lambda,\tau) &= e^{-\frac{\sigma^2\lambda^2}{2}\tau + c}\end{aligned}$$

적분상수 c의 계산을 위해 초기조건을 적용해보자. 먼저, 초기조건 $u(x,0) = g(x)$을 $\widehat{u}$의 초기조건으로 바꿀 수 있다.

$$\widehat{u}(\lambda,0) = \widehat{g}(\lambda) = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\infty} g(x)e^{-i\lambda x}dx. \tag{2.32}$$

초기조건 (2.32)를 만족하는 상미분 방정식 (2.31)의 해가 다음과 같음을 쉽게 알 수 있다.

$$\widehat{u}(\lambda,\tau) = \widehat{g}(\lambda)e^{-\frac{\sigma^2\lambda^2\tau}{2}}. \tag{2.33}$$

식 (2.33)에 역 Fourier 변환을 적용해서 $u(x,\tau)$의 해를 구할 수 있다.

$$\begin{aligned}u(x,\tau) &= \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\infty}\widehat{u}(\lambda,\tau)e^{i\lambda x}d\lambda = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\infty}\widehat{g}(\lambda)e^{i\lambda x-\frac{\sigma^2\lambda^2\tau}{2}}d\lambda\\ &= \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\infty}\left[\frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\infty}g(y)e^{-i\lambda y}dy\right]e^{i\lambda x-\frac{\sigma^2\lambda^2\tau}{2}}d\lambda\\ &= \frac{1}{2\pi}\int_{-\infty}^{\infty}g(y)\left[\int_{-\infty}^{\infty}e^{-i\lambda(x-y)-\frac{\sigma^2\lambda^2\tau}{2}}d\lambda\right]dy.\end{aligned} \tag{2.34}$$

2번째 변수변환

$$v = \frac{y-x}{\sigma\sqrt{\tau}} \rightarrow y = x + \sigma\sqrt{\tau}v \tag{2.42}$$

이 때 경계조건은 다음과 같이 쓸 수 있다.

$$g(y) = g(x+\sigma\sqrt{\tau}v) = \begin{cases} E(e^{x+\sigma\sqrt{\tau}v} - 1) & \text{if } \ v \geq \frac{-x}{\sigma\sqrt{\tau}} \\ 0 & \text{else} \end{cases}$$

2번째 변수변환을 하게 되면,

$$u(x,\tau) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} g(y) e^{-\frac{v^2}{2}} dv.$$

$g(y)$의 경계조건에 따라 $v < -\frac{x}{\sigma\sqrt{\tau}}$인 부분의 적분 값들은 모두 0의 값을 갖게 되므로, $v \geq \frac{-x}{\sigma\sqrt{\tau}}$의 부분의 적분 값만 생각하면 된다. $g(y)$의 경계조건에 주목하면,

$$\begin{aligned} u(x,\tau) &= \frac{1}{\sqrt{2\pi}} \int_{-\frac{x}{\sigma\sqrt{\tau}}}^{+\infty} g(y) e^{-\frac{v^2}{2}} dv = \frac{1}{\sqrt{2\pi}} \int_{-\frac{x}{\sigma\sqrt{\tau}}}^{+\infty} (Ee^{x+\sigma\sqrt{\tau}v} - E) e^{-\frac{v^2}{2}} dv \\ &= \frac{1}{\sqrt{2\pi}} \int_{-\frac{x}{\sigma\sqrt{\tau}}}^{+\infty} Ee^{x+\sigma\sqrt{\tau}v} e^{-\frac{v^2}{2}} dv - \frac{1}{\sqrt{2\pi}} \int_{-\frac{x}{\sigma\sqrt{\tau}}}^{+\infty} Ee^{-\frac{v^2}{2}} dv \end{aligned} \tag{2.43}$$

첫 번째 변수변환(2.25)에 의해 식(2.43)의 첫 번째 항은 다음을 만족한다.

$$\begin{aligned} u(x,\tau) &= \frac{1}{\sqrt{2\pi}} \int_{-\frac{x}{\sigma\sqrt{\tau}}}^{+\infty} Se^{r\tau - \frac{\sigma^2}{2}\tau} e^{\sigma\sqrt{\tau}v} e^{-\frac{v^2}{2}} dv - \frac{1}{\sqrt{2\pi}} \int_{-\frac{x}{\sigma\sqrt{\tau}}}^{+\infty} Ee^{-\frac{v^2}{2}} dv \\ &= \frac{Se^{r\tau}}{\sqrt{2\pi}} \int_{-\frac{x}{\sigma\sqrt{\tau}}}^{+\infty} e^{-\frac{1}{2}(v-\sigma\sqrt{\tau})^2} dv - \frac{1}{\sqrt{2\pi}} \int_{-\frac{x}{\sigma\sqrt{\tau}}}^{+\infty} Ee^{-\frac{v^2}{2}} dv. \end{aligned} \tag{2.44}$$

여기서 $z = v - \sigma\sqrt{\tau}$로 변수변환을 하게 되면,

$$\begin{aligned} u(x,\tau) &= Se^{r\tau} \frac{1}{\sqrt{2\pi}} \int_{-\frac{x}{\sigma\sqrt{\tau}} - \sigma\sqrt{\tau}}^{+\infty} e^{-\frac{z^2}{2}} dz - \frac{1}{\sqrt{2\pi}} \int_{-\frac{x}{\sigma\sqrt{\tau}}}^{+\infty} Ee^{-\frac{v^2}{2}} dv \\ &= Se^{r\tau} N\left(\frac{x}{\sigma\sqrt{\tau}} + \sigma\sqrt{\tau}\right) - EN\left(\frac{x}{\sigma\sqrt{\tau}}\right), \end{aligned}$$

```
d1 = (log(S/E) + (r+ 0.5*sigma^2)*T)/(sigma*sqrt(T));
d2 = d1 -(sigma*sqrt(T));
CallPrice = S * normcdf(d1) - E * exp(-r * T)*normcdf(d2)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

위의 MATLAB 코드`eurocall.m`을 실행하면 다음의 결과를 얻는다.

```
>> eurocall
>> CallPrice =11.6105
```

제 3 절 유한 차분법 (Finite Difference Method)

유한차분법은 미분방정식 (differential equation)을 차분방정식 (difference equation)으로 이산화 시켜서 수치적인 해를 구하는 방법이다. 이 장에서는 유한차분법을 사용하여 열방정식 (heat equation)과 블랙 숄즈 편미분방정식의 근사해를 구할 것이다. 먼저 Taylor의 정리[3]를 바탕으로 하고 있는 유한차분법의 기본 원리를 살펴보자. Taylor의 정리를 이용하면 함수 $u(x+h,t)$는 다음과 같이 (x,t) 에서의 u 함수값과 미분값들의 무한급수로 나타낼 수 있다.

$$u(x+h,t) = u(x,t) + u_x(x,t)h + \frac{u_{xx}(x,t)}{2}h^2 + \frac{u_{xxx}(x,t)}{3!}h^3 + \cdots \quad (2.45)$$

$u_x(x,t)$에 대해서 정리하면, 1차미분에 대한 차분식을 얻는다.

$$u_x(x,t) = \frac{u(x+h,t) - u(x,t)}{h} + O(h). \quad (2.46)$$

이것이 전방 차분법 (forward difference method)이다. 마찬가지로, 변수 t에 관하여 전방차분을 하면

$$u_t(x,t) = \frac{u(x,t+k) - u(x,t)}{k} + O(k) \quad (2.47)$$

[3]Taylor 정리: 함수 $f(x)$가 $x = x_0$에서 n번 미분가능하다고 하자.

$$p_n(x) = u(x_0) + u'(x_0)(x-x_0) + \frac{u''(x_0)}{2!}(x-x_0)^2 + \ldots + \frac{u^{(n)}(x_0)}{n!}(x-x_0)^n$$

을 $x = x_0$에서 $u(x)$의 n번째 Taylor 다항식이라 한다.

이 때 경계조건은 $u(0,t) = u(1,t) = 0 \quad (t > 0)$이고 초기조건은 $u(x,0) = \sin(\pi x) \quad (0 \le x \le 1)$을 만족한다. 해석해는 $u(x,t) = \sin(\pi x)\mathrm{e}^{-\pi^2 t}$이며 그림2.5와 그림2.6처럼 그래프로 나타낼 수 있다. 이 방정식에 대한 근사해를 명시적, 함축적, 그리고 크랭크-니콜슨 유한차분법을 이용하여 구해보자.

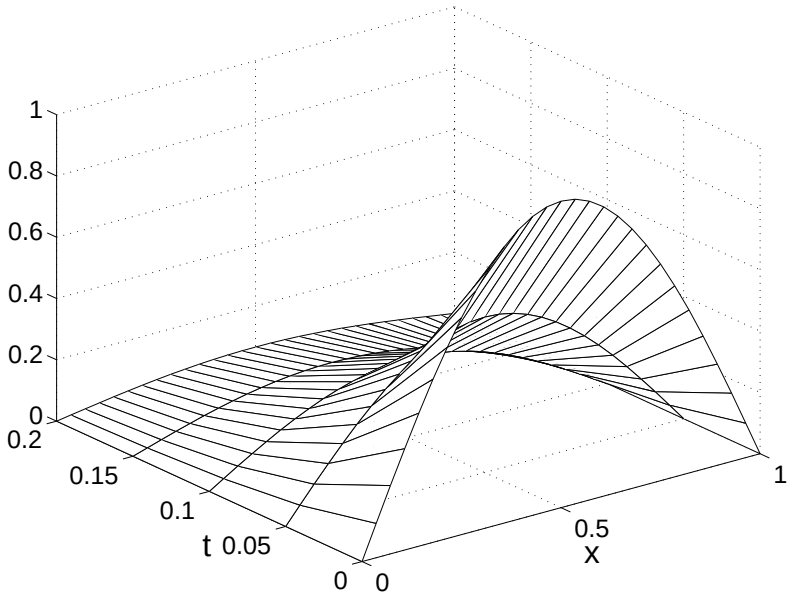


그림 2.5: 열방정식의 해석해

4.1 명시적 (Explicit) 유한 차분법

먼저 정수 $N_x > 0$을 선택하고 $h = 1/(N_x - 1)$이라 정의하면, 식(2.47)와 (2.51)을 이용하여, 열방정식(2.52)에 대해 다음과 같이 유한차분법을 적용할 수 있게 된다. 시간에 대해서 u_t을 유한 전방차분 그리고 공간에 대해서 u_{xx}에 대한 중앙을 이용하여 열방정식을 다음과 같이 이산화 시켜서 나타낼 수 있다.

$$\frac{u_i^{n+1} - u_i^n}{k} + O(k) = \frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{h^2} + O(h^2) \tag{2.53}$$

$$\text{for} \quad i = 2, \ldots, N_x - 1 \quad \text{and} \quad n = 1, 2, \ldots, N_t.$$

```
for n=1:Nt
    for i=2:Nx-1
        u(i,n+1) = u(i,n)+alpha*(u(i-1,n)-2*u(i,n)+u(i+1,n));
      exu(i,n+1) = sin(pi*x(i))*exp(-pi^2*(k*n));
    end
end
plot(x,u(:,1),'k*',x,u(:,50),'kd',x,u(:,100),'ks',...
     x,u(:,Nt+1),'ko');
hold
plot(x,exu(:,1),'k',x,exu(:,50),'k',x,exu(:,100),'k',...
    x,exu(:,Nt+1),'k')
legend('initial','n=50','n=100','n=Nt+1','exact
solution',-1)
xlabel('x','FontSize',20); ylabel('u(x,t)','FontSize',20)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

4.2 함축적 (Implicit) 유한 차분법

명시적 유한차분법의 안정조건인 $0 < \alpha \leq \frac{1}{2}$의 제약을 피하기 위해 함축적 유한 차분법을 이용한다. 함축적 방법은 시간간격을 작게 취하지 않고도 많은 수의 격자점들을 이용할 수 있다. 다만 함축적 방법은 유한차분 연립방정식의 해를 구해야 한다. 보통 함축적 유한차분법이라고 알려진 완전 함축적 유한차분법은 u_t에 대한 후방 유한차분근사와 u_{xx}에 대한 중앙차분 근사를 이용한다. 따라서 다음과 같은 함축적 유한차분 방정식을 이끌어 낼 수 있다.

$$\frac{u_i^{n+1} - u_i^n}{k} = \frac{u_{i-1}^{n+1} - 2u_i^{n+1} + u_{i+1}^{n+1}}{h^2}.$$

다시 정리하면 다음의 함축적 유한차분방정식

$$-\alpha u_{i-1}^{n+1} + (1+2\alpha)u_i^{n+1} - \alpha u_{i+1}^{n+1} = u_i^n,\ \alpha = \frac{k}{h^2} \quad \text{for each } i = 2, \cdots, N_x - 1 \tag{2.55}$$

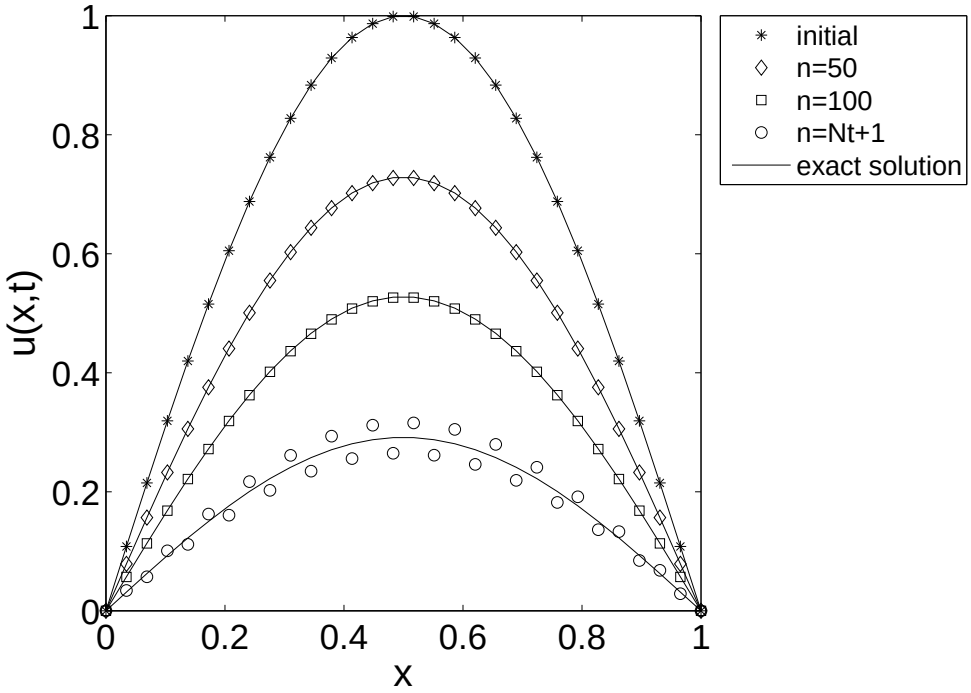


그림 2.8: $\alpha = 0.55$인 불안정한 상태

$$\begin{pmatrix} 1+2\alpha & -\alpha & 0 & \dots & 0 \\ -\alpha & 1+2\alpha & -\alpha & & 0 \\ 0 & -\alpha & \ddots & \ddots & \vdots \\ \vdots & & \ddots & \ddots & -\alpha \\ 0 & 0 & & -\alpha & 1+2\alpha \end{pmatrix} \begin{pmatrix} u_2^{n+1} \\ u_3^{n+1} \\ \vdots \\ \vdots \\ u_{N_x-1}^{n+1} \end{pmatrix}$$

$$= \begin{pmatrix} \alpha u_1^{n+1} + u_2^n \\ u_3^n \\ \vdots \\ \vdots \\ u_{N_x-1}^n + \alpha u_{N_x}^{n+1} \end{pmatrix} = \begin{pmatrix} b_2^n \\ b_3^n \\ \vdots \\ \vdots \\ b_{N_x-1}^n \end{pmatrix}. \qquad (2.56)$$

4.2.1 토마스 알고리즘 (Thomas Algorithm)

영이 아닌 원소를 가지는 다음 행렬을 살펴보자.

$$\begin{pmatrix} d_1 & c_1 & & & & & & & \\ a_1 & d_2 & c_2 & & & & & & \\ & a_2 & d_3 & c_3 & & & & & \\ & & \ddots & \ddots & \ddots & & & & \\ & & & a_{i-1} & d_i & c_i & & & \\ & & & & \ddots & \ddots & \ddots & & \\ & & & & & a_{N_x-2} & d_{N_x-1} & c_{N_x-1} & \\ & & & & & & a_{N_x-1} & d_{N_x} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_i \\ \vdots \\ x_{N_x-1} \\ x_{N_x} \end{pmatrix}$$

$$= \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_i \\ \vdots \\ b_{N_x-1} \\ b_{N_x} \end{pmatrix}, \tag{2.59}$$

여기서 표시되지 않은 원소들은 모두 0이다. 삼중대각(Tridiagonal) 행렬은 $|i-j| \geq 2$일 때, $a_{ij} = 0$가 되는 특징을 가지고 있다. 이제 삼중대각 행렬의 해를 구하는 알고리즘을 구해보자.

1행에 a_1/d_1을 곱한 값을 2행에서 뺀다. 그러면 a_1의 자리에는 0이 위

전방소거의 결과로, 식(2.59)은 다음의 형태를 갖게 된다.

$$\begin{pmatrix} d_1 & c_1 & & & & & & \\ & d_2 & c_2 & & & & & \\ & & d_3 & c_3 & & & & \\ & & & \ddots & \ddots & & & \\ & & & & d_i & c_i & & \\ & & & & & \ddots & \ddots & \\ & & & & & & d_{N_x-1} & c_{N_x-1} \\ & & & & & & & d_{N_x} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_i \\ \vdots \\ x_{N_x-1} \\ x_{N_x} \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_i \\ \vdots \\ b_{N_x-1} \\ b_{N_x} \end{pmatrix}.$$

여기서 b_i와 d_i들은 처음 값과는 다른 값을 가지고 있지만 c_i의 값들은 처음 값과 동일하다. 이제 후방대입을 통해 $x_{N_x},\ x_{N_x-1},\ \cdots,\ x_1$을 차례로 구할 수 있다:

$$\begin{aligned} x_{N_x} &= \frac{b_{N_x}}{d_{N_x}}, \\ x_i &= \frac{1}{d_i}(b_i - c_i x_{i+1}), \qquad i = N_x - 1,\ N_x - 2,\ \cdots,\ 1. \end{aligned}$$

`heatim.m`은 열방정식의 함축적 유한 차분법을 이용한 수치해를 토마스 알고리즘을 이용하여 구하는 MATLAB 코드이다.

```
%%%%%%%%%%%%%%%%%%%%%%% heatim.m %%%%%%%%%%%%%%%%%%%%%%%%%
clear; clc; clf; Nx=12; x=linspace(0,1,Nx); h=x(2)-x(1);
T=0.1; alpha=2; k=alpha*(h^2); Nt=round(T/k);
u(:,1)=sin(pi*x);
for i=1:Nx-2
    dd(i)= 1 + 2*alpha; c(i)= - alpha; a(i)= - alpha;
end
```

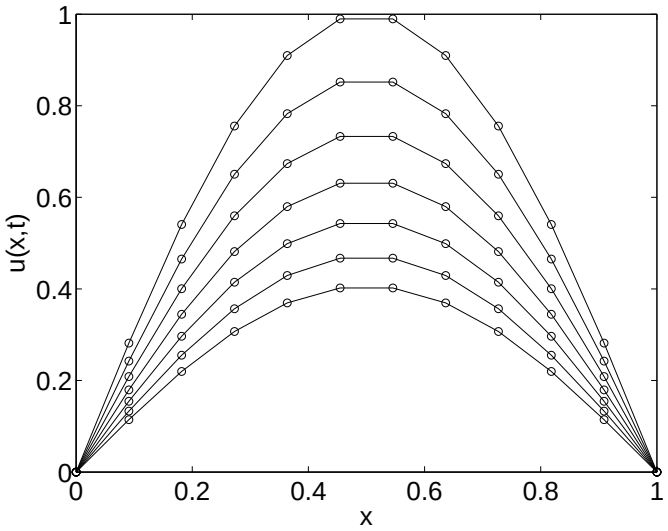


그림 2.9: 함축적 열방정식 $\alpha = 2$인 안정한 상태

이제 노드(x_i, t^n)에서 테일러전개를 하면 각각의 항을 다음과 같이 나타낼 수 있다.

$$\begin{aligned} T(x_i,t^n) &= u_t(x_i,t^n) + \frac{k}{2}u_{tt}(x_i,t^n) + \mathcal{O}(k^2) \\ &\quad -u_{xx}(x_i,t^n) + \frac{h^2}{12}u_{xxxx}(x_i,t^n) + \mathcal{O}(h^4). \end{aligned}$$

여기서 $u(x_i, t^n)$은 열방정식을 만족하므로 다음이 성립한다.

$$\begin{aligned} T(x_i,t^n) &= \frac{k}{2}u_{tt}(x_i,t^n) + \frac{h^2}{12}u_{xxxx}(x_i,t^n) + \mathcal{O}(k^2) + \mathcal{O}(h^4) \\ &= \mathcal{O}(k) + \mathcal{O}(h^2). \end{aligned} \tag{2.60}$$

수치적 해가 수렴하기 위해 필요한 조건은 수치기법의 국소절단오차가 공간간격과 시간간격을 줄일수록 0에 근사해야 한다는 것이다. 이럴 경우에, 수치기법이 일관적 (consistent)이라고 한다. 정확도의 차수 (order of accuracy)는 절단오차항에서 h와 k의 승수의 차수로 정의된다. 절단오차항을 $\mathcal{O}(k^l + h^m)$로 가정하면 수치기법이 l차 시간 정확 (lth order time accurate)하고 m차 공간정확 (mth order space accurate)하다고 한다. 식(2.60)으로부터 명시적 유한차분법은 1차 시간 정확하고 2차 공간 정확함을 알 수 있다.

```
end
fprintf('-----------------------------------------------\n')
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

초기조건은 $u(x,0) = \sin(x)$, $T = 0.1$, $\alpha = 0.1$, $h = 1/N$, $\Delta t = \alpha h^2$. MATLAB 코드 heatex_convergence_test을 실행하면 다음의 결과를 얻을 수 있다.

```
>> heatex_convergence_test
-----------------------------------------------
    h          dt         max_error       order
-----------------------------------------------
 0.10000    0.001000     0.001220
 0.05000    0.000250     0.000303      2.008865
 0.02500    0.000063     0.000076      2.002218
 0.01250    0.000016     0.000019      2.000555
 0.00625    0.000004     0.000005      2.000139
-----------------------------------------------
```

4.3.2 함축적 유한차분법

열방정식의 함축적 유한 차분법의 수렴성을 알아보기 위해 다음의 테스트를 수행해보자.

```
%%%%%%%%%%%%%%% heatim_convergence_test.m %%%%%%%%%%%%%%%
clear; clc; T=0.1; alpha=0.1;
for iter=1:5
   N=10*2^(iter-1)+1; x=linspace(0,1,N); h=x(2)-x(1);
   k=alpha*h^2; Nt=round(T/k); u(1:N,1:Nt+1)=0;
   u(:,1)=sin(pi*x); exact=u(:,1)*exp(-pi^2*T);
   for i=1:N-2
```

```
for iter = 2:5
fprintf('%8.5f    %8.6f     %8.6f      %8.6f \n', ...
    hh(iter),tt(iter), err(iter),Order(iter-1))
end
fprintf('-----------------------------------------------\n')
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

초기조건은 $u(x,0) = \sin(x)$, $T = 0.1$, $\alpha = 0.1$, $h = 1/N$, $\Delta t = \alpha h^2$. MATLAB 코드 `heatim_convergence_test`을 실행하면 다음의 결과를 얻을 수 있다.

```
>> heatim_convergence_test
-----------------------------------------------
    h          dt        max error       order
-----------------------------------------------
 0.10000    0.001000     0.004820
 0.05000    0.000250     0.001209      1.995470
 0.02500    0.000063     0.000302      1.998865
 0.01250    0.000016     0.000076      1.999716
 0.00625    0.000004     0.000019      1.999929
-----------------------------------------------
```

제 5 절 Black-Scholes 편미분방정식에 대한 유한 차분법

유러피언 콜 옵션의 값을 구하기 위해서 Black-Scholes 편미분방정식을 유한차분법으로 풀어서 구한다. 편미분방정식은 Dirichlet 경계조건을 갖는 포물선형 편미분방정식이다. 특히 초기조건보다 만기시의 조건이 주어진다. $\tau = T - t$를 잔존기간으로 놓음으로써, 더 자연스러운 시간의 방정식

```
%%%%%%%%%%%%%%%%%%%%%% BSex.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%
clf; clear; E=230; L=800; sigma=0.5; r=0.03; T=1; Nx=50;
Nt=1000; k=T/Nt; x=linspace(0,L,Nx); h=x(2)-x(1);
u(1:Nx,1:Nt+1)=0;
for i=1:Nx
    if x(i)<= E
        u(i,1)=0;
    else
        u(i,1)=x(i)-E;
    end
end
for n=2:Nt+1
    u(Nx,n)=L-E*exp(-r*k*(n-1));
end
for n=1:Nt
    for i=2:Nx-1
        u(i,n+1)=u(i,n) + k*((1/2)*(sigma^2)*((i-1)*h)^2*...
            ((u(i+1,n)-2*u(i,n)+u(i-1,n))/(h^2)) +...
            r*(i-1)*h*((u(i+1,n)-u(i-1,n))/(2*h))-r*u(i,n));
    end
end
plot(x,u(:,1:200:Nt+1),'ko-')
axis image; axis([0 L 0 600])
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

5.2 함축적 방법에 의한 옵션 가격 결정

시간에 대한 전방 차분을 이용한 명시적 방법에서 일어날 수 있는 불안정성 문제를 해결하기 위해서 후방차분을 이용하여 함축적 방법을 적용하면

```
%%%%%%%%%%%%%%%%%%%%% BSim.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%
clf; clear; E=230; L=800; sigma=0.5; r=0.03; T=1; Nx=50;
Nt=100; k=T/Nt; x=linspace(0,L,Nx); h =x(2)-x(1);
u(1:Nx,1:Nt+1)=0; N=Nx-2;
for i=1:Nx
    if x(i) < E
        u(i,1)= 0;
    else
        u(i,1)= x(i)-E;
    end
end
for i=1:N
   dd(i)=1/k+(sigma*i)^2+r; c(i)=-r*i/2-((sigma*i)^2)/2;
   a(i)=r*(i+1)/2-((sigma*(i+1))^2)/2;
end
for n=1:Nt
    d=dd;
    for i=1:N-1
         b(i)=u(i+1,n)/k;
    end
     u(Nx,n+1)=L - E*exp(-r*k*n);
     b(N)=u(N+1,n)/k - c(N)*u(Nx,n+1);
    for i = 2:N
        xmult= a(i-1)/d(i-1);
        d(i) = d(i) - xmult*c(i-1);
        b(i) = b(i) - xmult*b(i-1);
    end
    u(N+1,n+1) = b(N)/d(N);
    for i = N-1:-1:1
```

제 3 장

전산유체역학(Computational Fluid Dynamics)

전산 유체 역학(CFD, Computational Fluid Dynamics)은 비선형 미분 방정식 Navier-Stoke Equation을 수치 기법(numerical methods)의 알고리듬을 사용하여 유체 유동 문제를 풀고 해석하는 것이다.

제 1 절 비압축 점성 유체 유동

Navier-Stokes 방정식이란 점성을 가진 유체의 운동을 기술하는 비선형 편미분 방정식이다. 예를 들면, 배의 몸통 주위를 흐르는 물이나 비행기 날개 위로 흐르는 공기와 같은 유체와 기체의 흐름을 기술하는 방정식이다. 이러한 Navier-Stokes 방정식는 오늘날까지 그 누구도 해의 공식의 존재 여부조차 밝혀내지 못하고 있으며 100만달러가 걸려있는 밀레니엄 문제(클레이 수학연구소 [1]) 중 하나이기도 하다.

[1] **클레이 수학연구소, Clay Mathematics Institute** 케임브리지 지방에 있는 사설 비영리 재단으로 수학을 발전시키고자 여러 활동을 하며 유망한 수학자들에게 여러상을 수여하기도 한다. 그 중 Navier-Stokes 방정식는 클레이 수학연구소가 100만달러의 현상금을 내 건 7가지 밀레니엄 문제 중 하나이기도 하다. 더 자세한 내용은 클레이 수학연구소 홈페이지(http://www.claymath.org)를 참고하기 바란다.

1.1 Navier-Stokes 방정식

Navier-Stokes 방정식는 다음과 같이 표현할 수 있다.

$$\rho\left(\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u}\cdot\nabla\mathbf{u}\right) = -\nabla p + \mu\Delta\mathbf{u} + \rho\mathbf{g}, \tag{3.1}$$

$$\nabla\cdot\mathbf{u} = 0, \tag{3.2}$$

위의 방정식에서 사용된 기호들은 다음의 의미를 갖는다. ρ, $\mathbf{u}$, p, 그리고 μ는 유체의 밀도, 속도, 압력, 그리고 점성을 의미한다. 또한 속도 $\mathbf{u} = \mathbf{u}(x, y, t)$와 압력 $p = p(x, y, t)$는 공간과 시간의 함수이다. 편의상 ρ, μ, 그리고 $\mathbf{g}$는 상수 또는 상수벡터로 간주한다. 속도 벡터 $\mathbf{u}$에 부과할 경계 조건은 no-slip 경계 조건이다. 이 조건은 고체 경계면에 접하고 있는 유체는 경계에 붙어있고 미끄러지지 않는다는 뜻이다.

1.2 Navier-Stokes 방정식 유도과정

2차원 유동의 Navier-Stokes 방정식 유도 과정을 살펴보자. 유도과정은 코시(Cauchy) 방정식에 응력-변형률 관계식을 대입하여 유도하였으며 Navier-Stokes 방정식을 처음 접하는 독자들도 이해하기 쉽도록 정리하였다. 편의상 중력과 같은 체적력은 고려하지 않으면 유체입자에 작용하는 모든 힘의 합은 Newton의 제 2법칙에 의해 질량과 가속도의 곱($\mathbf{F} = m\mathbf{a}$)으로 표현가능하므로 2차원 유동의 코시 방정식은 다음과 같다.

$$\underbrace{dm}_{①}\underbrace{a_x}_{②} = \underbrace{dF_x}_{③} \tag{3.3}$$

$$\underbrace{dm}_{①}\underbrace{a_y}_{②} = \underbrace{dF_y}_{③} \tag{3.4}$$

이제 위 식에 나온 순서대로 알아보자.

① 질량은 밀도와 부피의 곱으로 표현되므로, 다음과 같이 나타낼 수 있다.

$$dm = \rho dxdy$$

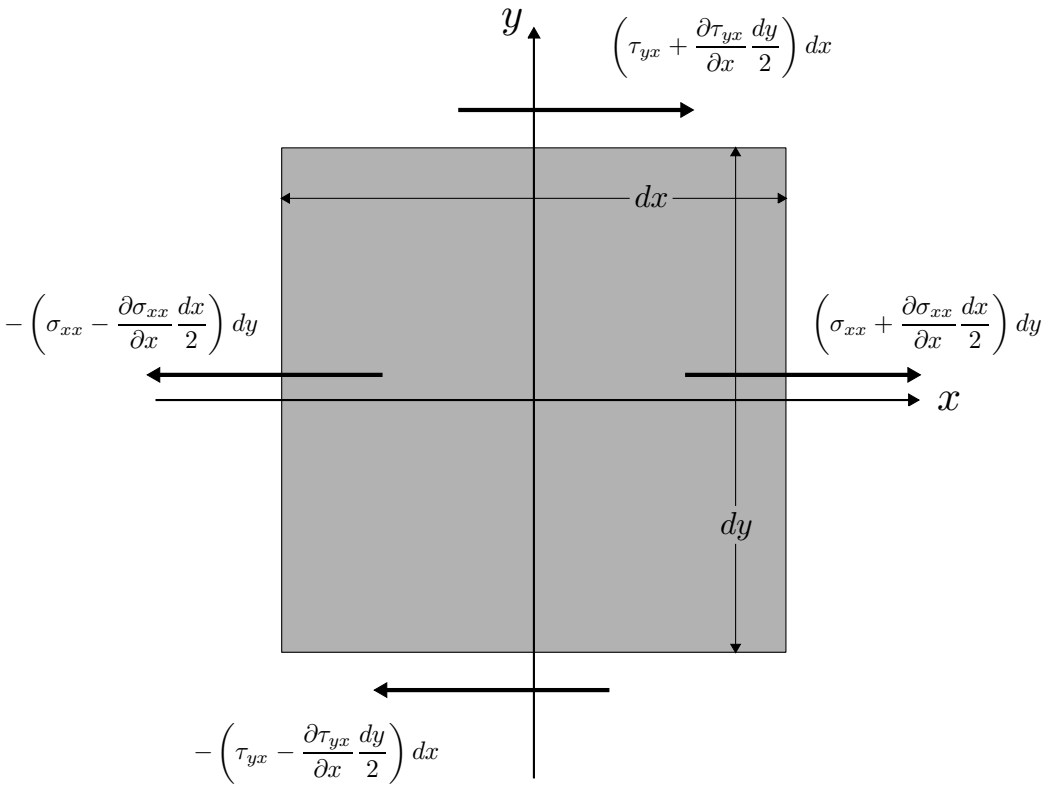


그림 3.2: x축에 대한 응력분포

이제 위 식을 정리하면 다음 식을 얻을 수 있다.

$$\rho\left(\frac{\partial u}{\partial t}+u\frac{\partial u}{\partial x}+v\frac{\partial u}{\partial y}\right) = \frac{\partial \sigma_{xx}}{\partial x}+\frac{\partial \tau_{yx}}{\partial y}, \tag{3.8}$$

$$\rho\left(\frac{\partial v}{\partial t}+u\frac{\partial v}{\partial x}+v\frac{\partial v}{\partial y}\right) = \frac{\partial \sigma_{yy}}{\partial y}+\frac{\partial \tau_{xy}}{\partial x}. \tag{3.9}$$

• 응력과 변형률 관계식

유체의 점성계수를 μ라고 하면

(전단응력) = (유체의 점성계수) × (유체의 속도변화율)

위 식으로 유체와 평판 사이의 전단응력을 구할 수 있다.

$$\tau_{xy}=\tau_{yx}=\mu\left(\frac{\partial u}{\partial y}+\frac{\partial v}{\partial x}\right)$$

또한, 인장응력은 양쪽으로 생기므로 다음을 만족한다.

$$\sigma_{xx}=-p+2\mu\frac{\partial u}{\partial x},\ \sigma_{yy}=-p+2\mu\frac{\partial v}{\partial y}$$

이제, 정리된 코시(Cauchy) 방정식 (3.8)과 (3.9)에 응력과 변형률 사이의 관계식을 대입해보자.

$$\begin{aligned}
&\rho\left(\frac{\partial u}{\partial t}+u\frac{\partial u}{\partial x}+v\frac{\partial u}{\partial y}\right)=-\frac{\partial p}{\partial x}+\frac{\partial \sigma_{xx}}{\partial x}+\frac{\partial \tau_{yx}}{\partial y}\\
&=-\frac{\partial p}{\partial x}+\frac{\partial}{\partial x}\left(2\mu\frac{\partial u}{\partial x}\right)+\frac{\partial}{\partial y}\left(\mu\left(\frac{\partial u}{\partial y}+\frac{\partial v}{\partial x}\right)\right)\\
&=-\frac{\partial p}{\partial x}+\mu\left(\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}\right)+\mu\frac{\partial}{\partial x}\left(\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}\right)\\
&=-\frac{\partial p}{\partial x}+\mu\left(\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}\right),
\end{aligned}$$

써 얻는 것이다. 이 장에서는 독자들의 이해를 돕기 위해 2차원상에서 기술하지만, 이를 기반으로 3차원으로의 확장도 어렵지 않게 접근할 수 있으리라 생각된다.

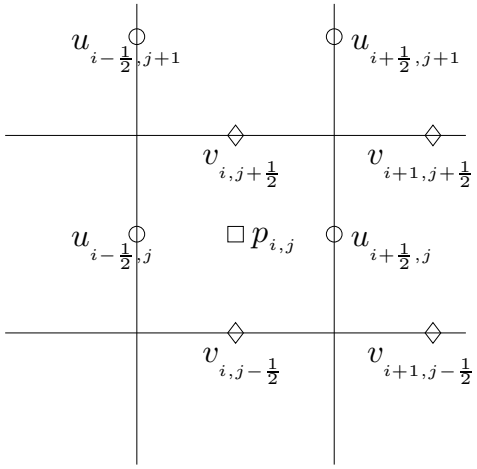


그림 3.4: 압력은 셀 중앙 그리고 속도는 셀 경계에 정의되어 있다.

계산 영역은 격자간격이 h인 균등격자이다. 각각의 셀, Ω_{ij}의 중심은 $(x_i, y_j) = ((i-0.5)h, (j-0.5)h)$ $i = 1, \cdots, M$ 그리고 $j = 1, \cdots, N$ 에 위치해 있다. 여기서 M과 N은 각각 x와 y-방향의 셀의 개수를 의미한다. 이제, Navier-Stokes 방정식를 풀기 위해 Staggered Marker-and-Cell (MAC) 격자를 이용할 것이다. Harlow and Welch [3] 의 MAC 격자는 압력은 셀 중앙에 속도는 셀 경계에 정의하는 격자를 의미한다.

$$u^n_{i+\frac{1}{2},j} \approx u(x_{i+\frac{1}{2}}, y_j, t^n), \tag{3.17}$$

$$v^n_{i,j+\frac{1}{2}} \approx v(x_i, y_{j+\frac{1}{2}}, t^n), \tag{3.18}$$

$$p^n_{ij} \approx p(x_i, y_j, t^n). \tag{3.19}$$

그림 3.4은 MAC 격자를 나타내는 그림이다. $u^n_{i+\frac{1}{2},j}$은 동그라미 심볼이 있는 곳에 $v^n_{i,j+\frac{1}{2}}$은 다이아몬드 심볼이 있는 곳에 p^n_{ij}은 정사각형 심볼이 있는 지역에 각각 정의 되어진다.

이제 $(n+1)$ 시점에서 압력장에 대하여 다음 방정식을 푼다.

$$\begin{aligned} \frac{\mathbf{u}^{n+1}-\tilde{\mathbf{u}}}{\Delta t} &= -\nabla_d p^{n+1}, && (3.24)\\ \nabla_d \cdot \mathbf{u}^{n+1} &= 0. && (3.25) \end{aligned}$$

식 (3.24)에서 divergence operator($\nabla_d \cdot$)와 식 (3.25)을 이용하면 압력에 대한 포아송 방정식(Poisson equation)을 얻을 수 있게 된다:

$$\Delta_d p^{n+1} = \frac{1}{\Delta t}\nabla_d \cdot \tilde{\mathbf{u}}, \tag{3.26}$$

위의 식에서 각각의 항들은 다음과 같이 정의되어진다.

$$\begin{aligned} \Delta_d p_{ij}^{n+1} &= \frac{p_{i-1,j}^{n+1}+p_{i+1,j}^{n+1}-4p_{ij}^{n+1}+p_{i,j-1}^{n+1}+p_{i,j+1}^{n+1}}{h^2},\\ \nabla_d \cdot \tilde{\mathbf{u}}_{ij} &= \frac{\tilde{u}_{i+\frac{1}{2},j}-\tilde{u}_{i-\frac{1}{2},j}}{h}+\frac{\tilde{v}_{i,j+\frac{1}{2}}-\tilde{v}_{i,j-\frac{1}{2}}}{h}. \end{aligned}$$

압력에 대한 경계조건은 다음과 같다.

$$\mathbf{n}\cdot\nabla_d p^{n+1} = \mathbf{n}\cdot\left(-\frac{\mathbf{u}^{n+1}-\mathbf{u}^n}{\Delta t}-(\mathbf{u}\cdot\nabla_d\mathbf{u})^n+\frac{1}{Re}\Delta_d\mathbf{u}^n\right), \tag{3.27}$$

여기서 $\mathbf{n}$은 영역 경계면에서의 단위법선벡터(unit normal vector)이다.

식 (3.26)로 얻어진 선형시스템은 Gauss-Seidel 반복계산법을 이용하여 근사해를 구할 수 있다. p^{n+1}을 구한 다음 divergence-free 속도는 다음과 같이 정의되어진다.

$$\mathbf{u}^{n+1} = \tilde{\mathbf{u}} - \Delta t \nabla_d p^{n+1}.$$

즉,

$$\begin{aligned} u_{i+\frac{1}{2},j}^{n+1} &= \tilde{u}_{i+\frac{1}{2},j}-\frac{\Delta t}{h}(p_{i+1,j}^{n+1}-p_{ij}^{n+1}),\\ v_{i,j+\frac{1}{2}}^{n+1} &= \tilde{v}_{i,j+\frac{1}{2}}-\frac{\Delta t}{h}(p_{i,j+1}^{n+1}-p_{ij}^{n+1}). \end{aligned}$$

이로써, 한번의 time step이 끝나게 된다.

주어진 초기조건은 $(x^0, y^0, z^0) = (0, 0, 0)$을 이용하여 반복적으로 해를 구해보면 다음의 결과를 얻게 된다.

$k = 0$ 일 때
$$x^1 = (3 - 2y^0 + z^0)/4 = \frac{3}{4},$$
$$y^1 = (-1 - x^0 - z^0)/3 = -\frac{1}{3},$$
$$z^1 = (4 - x^0 - y^0)/4 = 1.$$

$k = 1$ 일 때
$$x^2 = (3 - 2y^1 + z^1)/4 \approx 0.6667,$$
$$y^2 = (-1 - x^1 - z^1)/3 \approx -0.9167,$$
$$z^2 = (4 - x^1 - y^1)/4 \approx 0.8958.$$

실제 해 $x = 1, y = -1, z = 1$로 조금씩 근접해가는 것이다. 다음은 MATLAB을 이용하여 위의 예제의 해를 구하는 코드이다.

```
%%%%%%%%%%%%%%%%% Jacobi_iteration.m %%%%%%%%%%%%%%%%%%%
clear; clc; x(1) = 0; y(1) = 0; z(1) = 0; Max_iter = 20;
fprintf('----------------------------------------\n')
fprintf('    k         x          y         z \n')
fprintf('----------------------------------------\n')
for n = 1:Max_iter
    x(n+1) = (1/4)*(3 - 2*y(n) - z(n));
    y(n+1) = (1/3)*(-1-x(n)-z(n));
    z(n+1) = (1/4)*(4-x(n)-y(n));
    fprintf('%5.0d  %10.4f  %9.4f  %8.4f  \n',...
        n, x(n+1), y(n+1), z(n+1))
end
fprintf('----------------------------------------\n')
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

MATLAB코드 `Jacobi_iteration.m`을 실행해보면 다음의 결과를 얻을 수 있다.

예제

다음의 주어진 연립방정식을 Gauss-Seidel 반복계산법을 이용하여 $x,\ y,\ z$의 근사해를 구해보자. 초기 조건은 $x^0 = 0,\ y^0 = 0,\ z^0 = 0$으로 가정하자.

$$\begin{aligned} 4x + 2y + z &= 3, \\ x + 3y + z &= -1, \\ x + y + 4z &= 4. \end{aligned}$$

위의 연립방정식을 다음과 같이 변형하여 나타내보자.

$$\begin{aligned} 4x &= 3 - 2y + z, \\ 3y &= -1 - x - z, \\ 4z &= 4 - x - y. \end{aligned}$$

Gauss-Seidel 반복계산법을 이용하면 위의 식은 다음과 같이 정리된다.

$$\begin{aligned} x^{k+1} &= \frac{1}{4}(3 - 2y^k + z^k), \\ y^{k+1} &= \frac{1}{3}(-1 - x^{k+1} - z^k), \\ z^{k+1} &= \frac{1}{4}(4 - x^{k+1} - y^{k+1}). \end{aligned}$$

MATLAB코드 `GaussSeidel_iteration.m`을 실행해보면 다음의 결과를 얻을 수 있다.

```
---------------------------------------
   k         x          y         z
---------------------------------------
   1      0.7500    -0.5833    0.9583
   2      0.8021    -0.9201    1.0295
   3      0.9527    -0.9941    1.0103
   4      0.9944    -1.0016    1.0018
   5      1.0004    -1.0007    1.0001
   6      1.0003    -1.0001    1.0000
   7      1.0001    -1.0000    1.0000
   8      1.0000    -1.0000    1.0000
---------------------------------------
```

위의 결과를 보면 각각의 열이 k, x, y, z를 나타낸다. 위의 코드는 총 8번의 반복을 시행하였는데 이로 연립 방정식의 해 $x = 1, y = -1, z = 1$로 수렴함을 확인해볼 수 있다. 또한 앞서 배운 Jacobi 반복계산법보다 훨씬 빠르게 해에 수렴함도 알 수 있을 것이다.

제 4 절 포아송 방정식(Poisson equation)

Poisson 방정식은 정전기학(electrostatics), 기계공학(mechanical engineering)과 이론물리학(theoretical physics)에서 널리 쓰이는 편미분 방정식이다. 포아송 방정식은 다음과 같다.

$$\Delta u = f, \tag{3.28}$$

여기서 Δ는 라플라스 연산자(Laplace operator)라고 하고 1차원에서는 $\Delta u = u_{xx}$, 2차원에서는 $\Delta u = u_{xx} + u_{yy}$이다. 이제, 1차원과 2차원에서 포아송 방정식을 수치적으로 풀어보도록 하자.

$$\begin{aligned} u_3 - 2u_2 + u_1 &= h^2 f_2 \\ u_4 - 2u_3 + u_2 &= h^2 f_3 \\ &\vdots \\ -u_N + u_{N-1} &= h^2 f_N \end{aligned}$$

위 시스템을 Gauss-Seidel 반복계산법에 따라 풀기위해 다시 정리하면, 다음을 얻을 수 있다.

$$\begin{aligned} u_2^{k+1} &= (u_3^k - h^2 f_2)/2 \\ u_3^{k+1} &= (u_4^k + u_2^{k+1} - h^2 f_3)/2 \\ &\vdots \\ u_N^{k+1} &= u_{N-1}^k - h^2 f_N \end{aligned}$$

위 시스템을 더욱 간단히 정리하면,

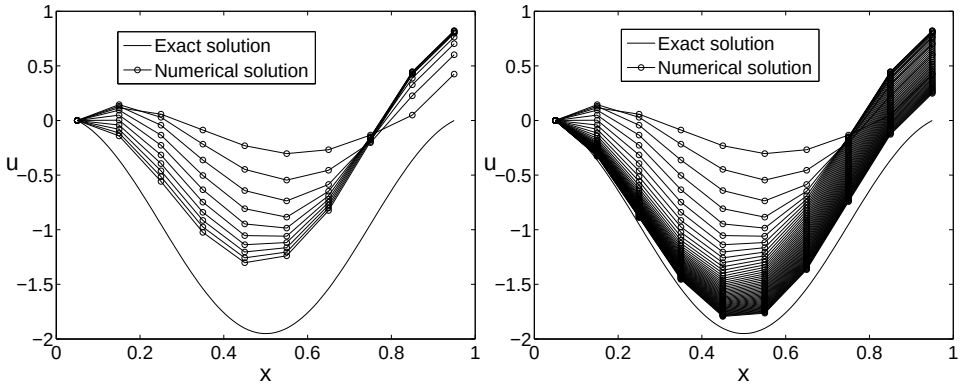
$$\begin{aligned} u_2^{k+1} &= (u_3^k - h^2 f_2)/2 \\ u_i^{k+1} &= (u_{i+1}^k + u_{i-1}^{k+1} - h^2 f_2)/2 \qquad (3 \le i \le N-1) \\ u_N^{k+1} &= u_{N-1}^k - h^2 f_N \end{aligned}$$

여기서, u^{k+1}의 값들이 유일하게 결정되도록 하기 위해서 모든 k에 대해서 u_1^k을 임의의 상수 0으로 고정시켰다. 이제, 이 과정을 MATLAB코드로 만들어보자.

$$f(x) = -4\pi^2 \cos(2\pi x) \tag{3.31}$$

로 주어졌을때 식 (3.29)을 만족하는 수치해를 구하시오.

프로그램 코드



(a) (b)

그림 3.6: Gauss Seidel방법에 의해 푼 포아송 방정식의 결과. (a) 10번 반복 (b) 50번 반복.

을 적용시키자. 위 방정식 (3.32)을 차분방정식으로 바꾸면,

$$\frac{u_{i+1,j} + u_{i-1,j} - 4u_{ij} + u_{i,j+1} + u_{i,j-1}}{h^2} = f_{ij} \qquad (3.34)$$
$$\text{for } 1 \le i \le N_x,\ 1 \le j \le N_y$$

이제 이산 경계조건은 식 (3.33)을 이용하면 다음과 같이 나타낼수 있다.

$$u_{i0} = u_{i1}, \qquad u_{i,N_y+1} = u_{i,N_y} \qquad (1 \le i \le N_x)$$
$$u_{0j} = u_{1j}, \qquad u_{N_x+1,j} = u_{N_x j} \qquad (1 \le j \le N_y)$$

을 적용하면 다음과 같다. 포아송 방정식 (3.28)의 $u(x,y)$가 해라면, 임의의 상수 C에 대하여 $u(x,y)+C$도 해가 된다. 따라서, 해의 유일성을 보장받기 위해 $u_{11} = 0$으로 두도록 하자. 식(3.34)을 Gauss-Seidel 반복계산법에 따라 풀기위해 다시 정리하면, 다음을 얻을 수 있다.

$$u_{ij}^{k+1} = (h^2 f_{ij} - u_{i+1,j}^{k} - u_{i-1,j}^{k+1} - u_{i,j+1}^{k} - u_{i,j-1}^{k+1})/(-4).$$

```
        end
    end
end
mesh(xx,yy,exu-exu(1,1));
mesh(xx,yy,u);
xlabel('x','fontsize',25); ylabel('y','fontsize',25);
zlabel('u','fontsize',25,'rotation',0);
set(gca,'fontsize',20); colormap([0 0 0]);
axis image; view(26,34); axis([0 1 0 1 -1.2 0.6])
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

그림 3.7를 보면, $u(1,1) = 0$인 값으로 고정했을 경우 반복 횟수에 따라 유일한 해 (a)로 수렴해감을 확인해 볼 수 있다. (b), (c), (d), (e), 그리고 (f)는 2, 10, 50, 5000, 그리고 100000번 반복 결과이다.

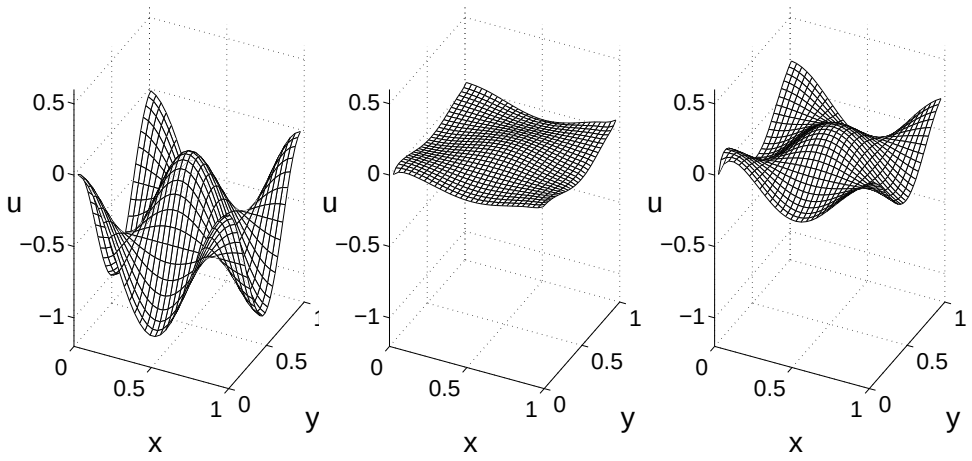
제 5 절 결과

MAC격자에서 비압축성의 Navier-Stokes 방정식의 수치해를 그림 3.8에서 확인할 수 있다. 이 결과는 전체 영역 $\Omega = (0,1) \times (0,1)$에서 $N_x = N_y = 16$이고, 시간간격 $\Delta t = 0.001$, 레이놀즈 수 $Re = 100$인 경우에 초기 속력장과 이 때 주어진 경계조건, 그리고 일정시간이 흐른 후에 속력장을 확인할 수 있다.

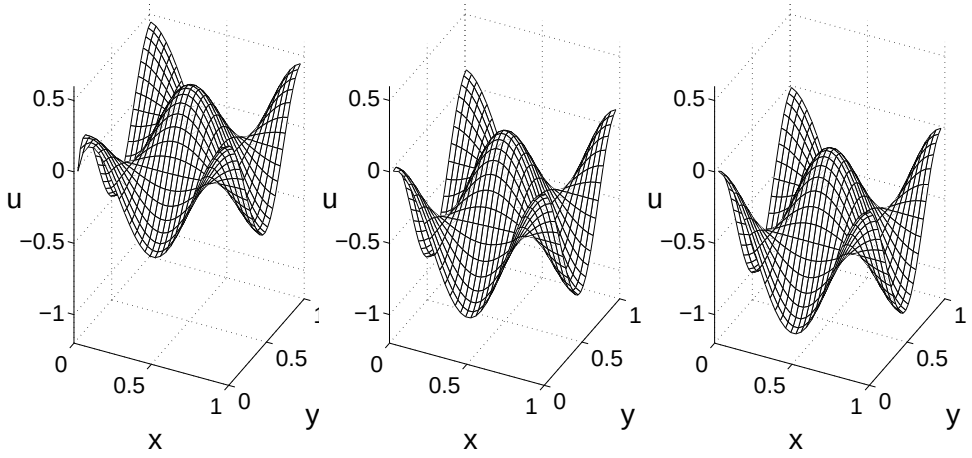
```
%%%%%%%%%%%%%%%%%%%%%%%% NS_GS.m %%%%%%%%%%%%%%%%%%%%%%%%%%
clear; clc; clf;
xleft = 0.0; xright = 1.0;  yleft = 0.0; yright = 1.0;
nx = 16; ny = nx; h = xright/nx; h2 = h^2;
max_it = 50; dt = 0.001; Re = 100.0; a = 1.0;
u(1:nx+1,1:ny+2) = 0.0;  nu = u;
v(1:nx+2,1:ny+1) = 0.0;  nv = v; p(1:nx+2,1:ny+2) = 0.0;
```

```
  + 0.25*(u(i-1,j)+u(i-1,j+1))*0.5*(-v(i,j)-v(i-1,j))/h;
    else
     adv_v(i,j) = 0.5*v(i,j)*(v(i,j+1)-v(i,j-1))/h ...
  + 0.25*(u(i-1,j)+u(i,j)+u(i-1,j+1)+u(i,j+1)) ...
    *0.5*(v(i+1,j)-v(i-1,j))/h;
            end
        end
  end
  for i = 2:nx
   for j = 2:ny+1
    tu(i,j) = u(i,j) + dt*((u(i+1,j)+u(i-1,j)-4.0*u(i,j)
...
        +u(i,j+1)+u(i,j-1))/(Re*h*h) - adv_u(i,j));
   end
  end
    tu(1,2:ny+1) = 0.0; tu(nx+1,2:ny+1) = 0.0;
 for i = 2:nx+1
  for j = 2:ny
   tv(i,j) = v(i,j)+dt*((v(i+1,j)+v(i-1,j)-4.0*v(i,j) ...
       +v(i,j+1)+v(i,j-1))/(Re*h*h)-adv_v(i,j));
  end
 end
 tv(2:nx+1,1) = 0.0; tv(2:nx+1,ny+1) = 0.0;
 tol = 1.0e-4; oldp(1:nx+2,1:ny+2) = p; resid = 1.0;
 while (resid > tol)
  oldp = p;
   for i = 2:nx+1
    for j = 2:ny+1
     coef = 0.0;
```

```
    for i = 1:nx
        for j = 1:ny+1
            if (j==1)
                dpdy(i,j) = 0.0;
            elseif (j==ny+1)
                dpdy(i,j) = 0.0;
            else
                dpdy(i,j) = (p(i+1,j+1)-p(i+1,j))/h;
            end
        end
    end
    for i = 2: nx+1
        for j = 2: ny+1
            nu(i,j) = tu(i,j) - dt*dpdx(i,j-1);
        end
    end
    for i = 2:nx+1
        for j = 2:ny+1
            nv(i,j) = tv(i,j) - dt*dpdy(i-1,j);
        end
    end
    nv(1:nx+2,1) = 0.0;
    nv(1:nx+2,ny+1) = 0.0;
    u = nu; v = nv;
end
x=linspace(0.5*h,1-0.5*h,nx);
y=linspace(0.5*h,1-0.5*h,ny);
[xx,yy]=meshgrid(x,y);
for k2=1:ny
```



(a) Exact solution (b) 2번 반복 (c) 10번 반복



(d) 50번 반복 (e) 5000번 반복 (f) 100000번 반복

그림 3.7: Gauss Seidel 반복계산법에 의해 푼 포아송 방정식의 결과.

제 4 장

PC 클러스터 구축 및 병렬계산

제 1 절 병렬계산에 대해서

1.1 병렬계산(Parallel computing)이란?

컴퓨터에서 병렬계산이란 프로그램 명령어를 여러 프로세서에 분산시켜 동시에 수행함으로써 빠른 시간 내에 원하는 답을 구하는 작업을 일컫는다.

제 2 절 PC 클러스터 구축

클러스터(Cluster)란 네트워크를 통해 여러 대의 컴퓨터를 연결하여 하나의 단일 컴퓨터처럼 동작하도록 제작한 컴퓨터를 말한다.

Cluster 구축 과정 요약 [1]

1. 관리(Master)컴퓨터에 리눅스를 설치한 후 MPI(Message Passing Interface) 라이브러리를 설치한다.
2. 계산(Slave)컴퓨터에 리눅스를 설치 후 MPI 라이브러리를 설치한다.
3. 완성된 클러스터를 테스트한다.

*국내 벤처회사로는 clunix가 있다. http://www.clunix.com

[1] 본 책에서는 • Master 컴퓨터 ⟶ 관리 컴퓨터 • Slave 컴퓨터 ⟶ 계산 컴퓨터라고 하겠다.

♠ **스위칭 허브(Switching Hub)**

스위칭 허브(그림 4.3)는 근거리통신망(LAN)을 구성할 때 단말기간 집선 장치로 스위칭 기능을 가진 장치를 가리킨다.

그림 4.3: Route

♠ **KVM 스위치**

클러스터를 사용할때에는 모니터, 마우스, 그리고 키보드를 사용하지 않고 네트워크를 이용하여 프로그램을 실행하게 된다. 하지만 처음에 클러스터를 구축할 때에는 모니터를 보고 마우스와 키보드를 사용하여 운영체제와 필요한 라이브러리를 설치한다. 따라서 하나의 키보드, 모니터, 마우스를 공유하면 비용절감의 효과가 있다. 이 때 쓰이는 장비를 KVM(Keyboard, Video, Mouse) 스위치라고 한다.

그림 4.4: Switch

Fedora Download 방법

1. KAIST FTP Server에서 Download

   주소 입력란에

   ftp://ftp.kaist.ac.kr/fedora/linux/releases/11/Fedora/i386/iso/를 입력하고 Fedora-11-i386-DVD.iso를 다운받는다.

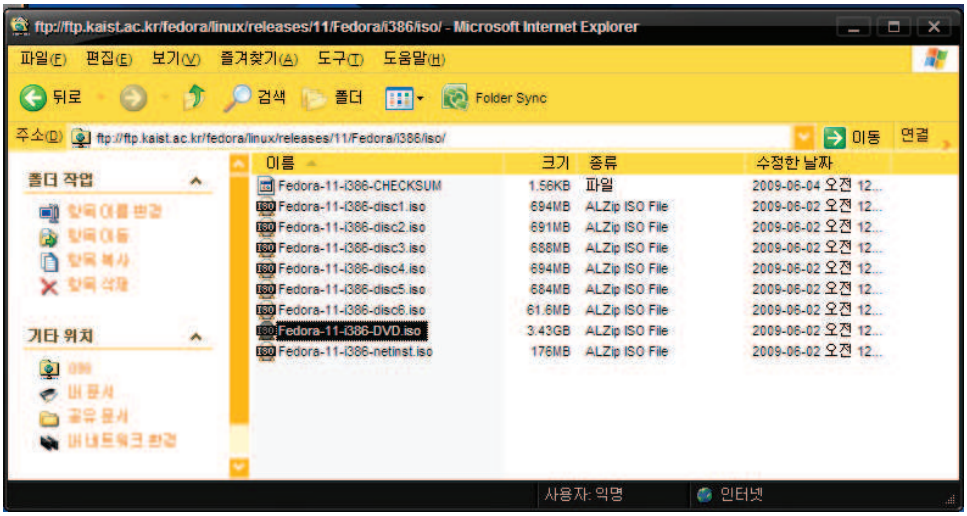


그림 4.5: KAIST FTP Server

2. Fedora홈페이지에 있는 Mirror Server를 이용

   Fedora Mirror Server로 이동하여 FTP서버를 선택한다.

   releases Directroy로 이동

   11 Directroy로 이동

   Fedora Directroy로 이동

   i386 Directroy로 이동

   iso Directroy로 이동

   Fedora-11-i386-DVD.iso 를 다운받는다.

그림 4.8: 11 Directroy

그림 4.9: Fedora Directroy

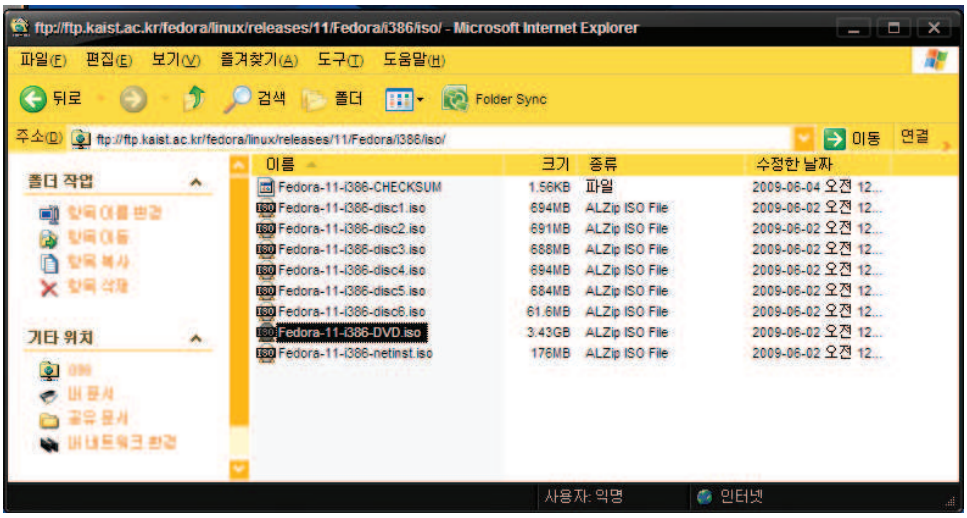


그림 4.12: Fedora iso Download

제 3 절 관리 컴퓨터(Master computer) Linux 설치 (Fedora 11 기준)

두 개의 랜카드를 구비한 컴퓨터에 설치한다.

♠ **BIOS 설정**

Fedora11 DVD를 CDROM에 넣고 재부팅을 한다. CDROM로 부팅해야 하기 문에 BIOS에서 부팅순서를 CDROM이 첫 순서로 되도록 해 놓는다. BIOS로 들어가는 방법은 부팅 하는 화면에서 F2키를 누르면 된다. CDROM이 가장 부팅 우선순위로 되면 F10키를 눌러 저장하고 BIOS화면에서 나간다.

♠ **Select Install Menu**

설치는 Install or upgrade an existing system을 선택한다.

♠ **Media Test**

부팅을 하면 CD 매체를 검사할지 물어보는데 Tab 키를 사용하여 [Skip]을 선택한 후 Enter 키를 누른다. 이후에 Next 클릭한다.

그림 4.14: Media Test

♠ **Language**

리눅스 상에서 쓰일 언어를 선택한다. English(English)를 선택 후 Next 클릭

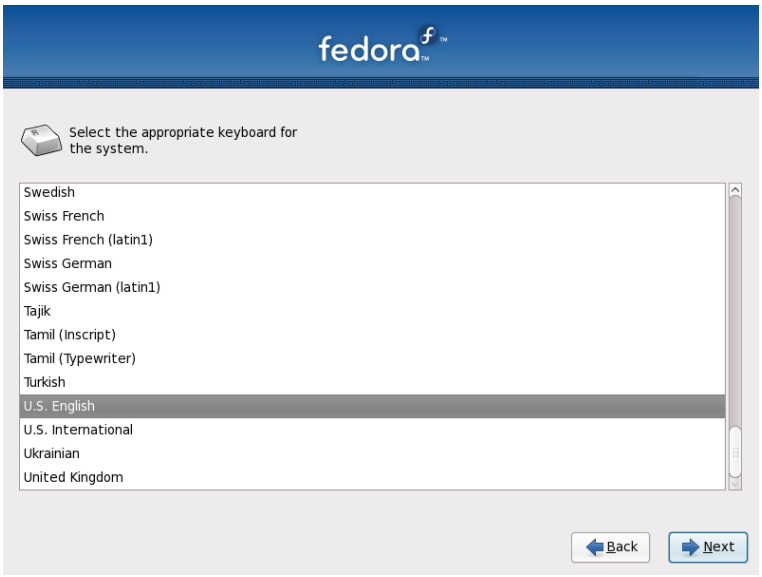


그림 4.16: Keyboard

그림 4.17: 관리 컴퓨터

그림 4.19: Nearest City

그림 4.20: Root Password

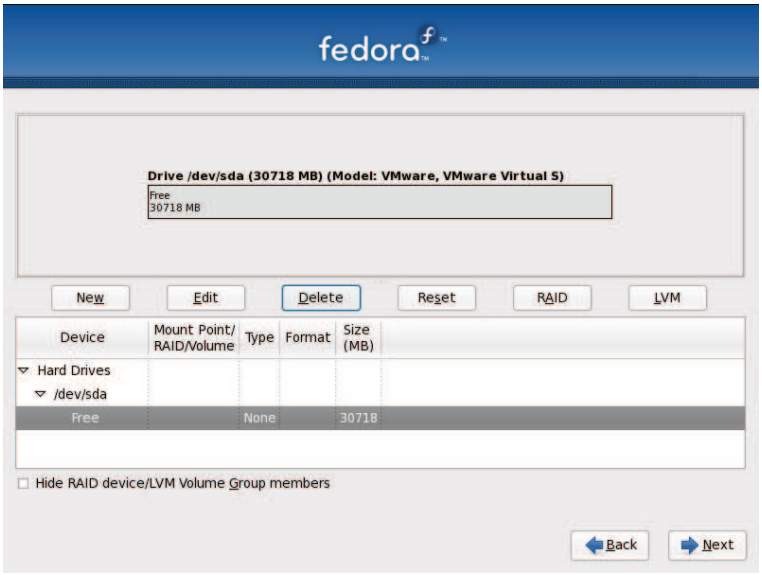


그림 4.22: Partition

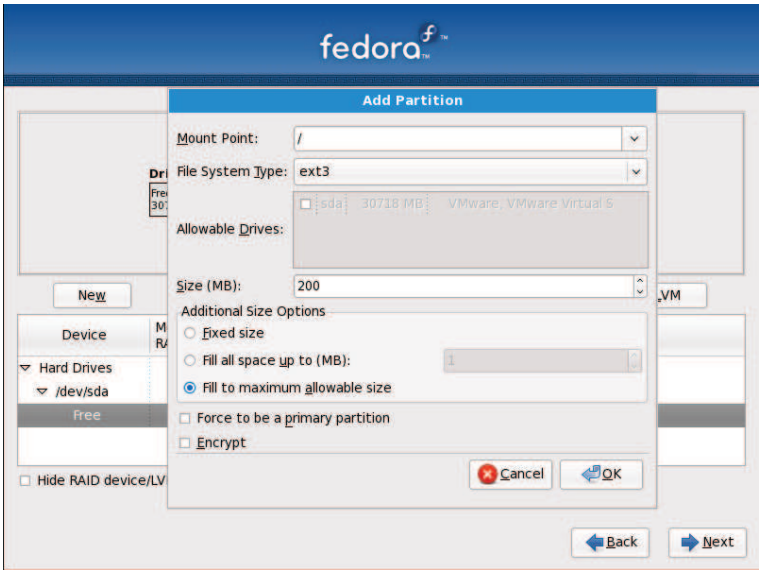


그림 4.23: Partition Add

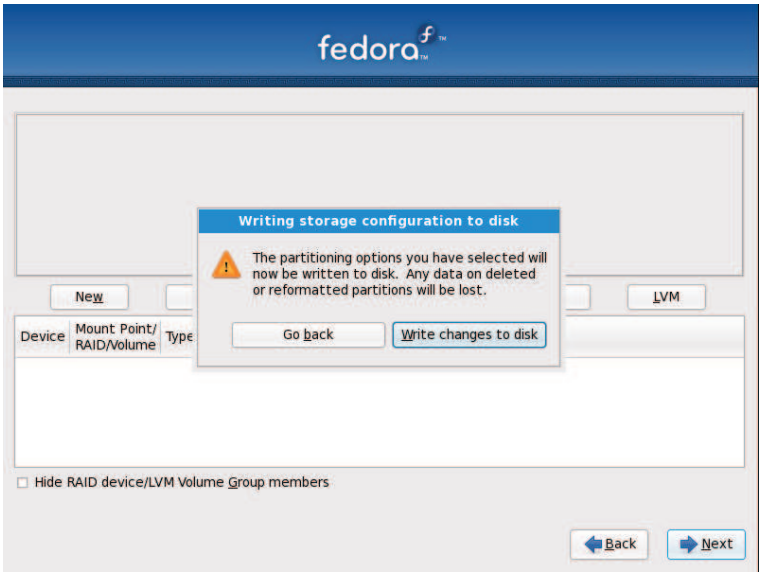


그림 4.25: Partition WriteDisk

그림 4.26: Boot System

(a) (b)

그림 4.28: (a) Install (b) Reboot

♠ **Welcome Fedora**

Reboot을 하고 나면 다음과 같은 화면이 나오는데 Forward를 클릭한다.

(a) (b)

그림 4.29: (a) Welcome Fedora (b) License

♠ **License**

License정보가 나온다. Forward를 클릭한다.

♠ **Create User**

계정의 이름은 어떤 이름도 상관없지만 반드시

• 계산 컴퓨터와 관리 컴퓨터의 계정 이름은 같아야 한다. 본 예제에서는 korea라는 계정 이름을 사용한다. 비밀번호도 편의상 111111으

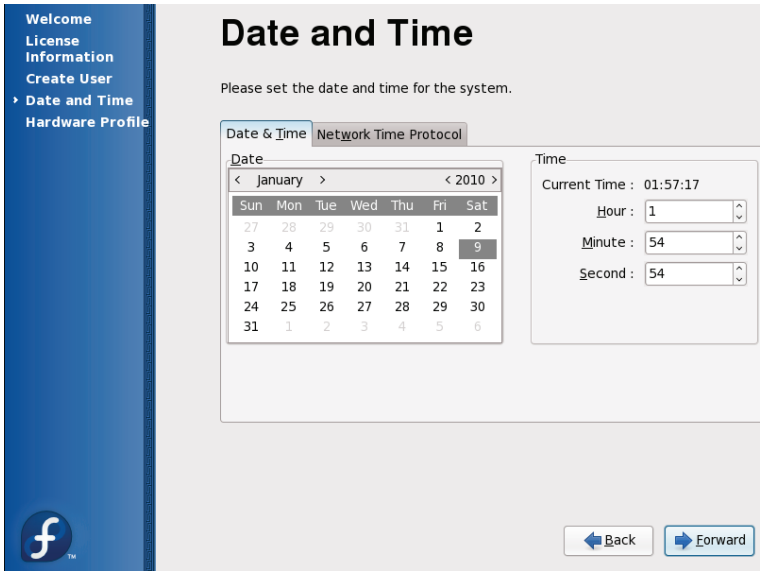


그림 4.31: Date And Time

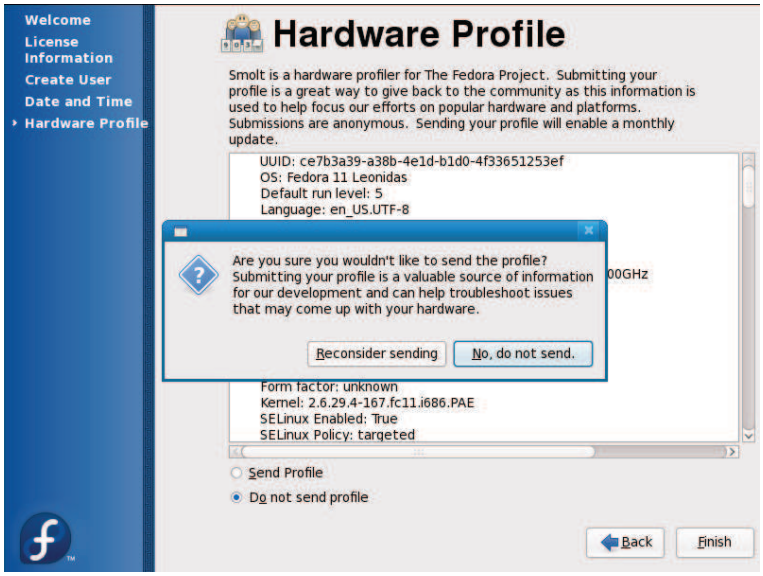


그림 4.32: Hardware Profile

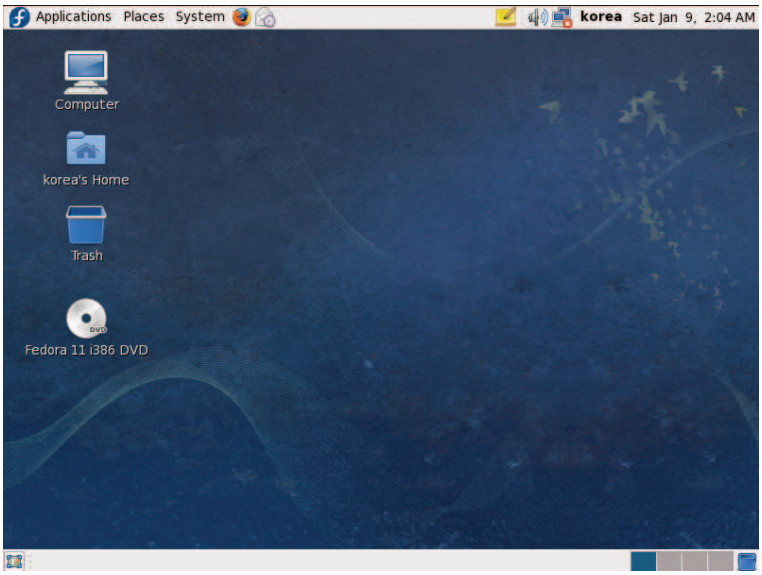


그림 4.34: Main

♠ **외부로 연결된 네크워크 장치 설정**

(본 책에서는 eth1이 외부로 연결되어 있다. 확인을 위해서는 Network configuration에서 Hardware를 보면 외부 랜카드 Description을 보면 알수있다. 계산 컴퓨터인 경우 설치하지 않는다.)

- Terminal로 들어가 `system-config-network`를 입력한다.
- 그 다음 외부로 연결된 장치(eth1)를 더블 클릭
- Controlled by NetworkManager와 Activate device when computer starts를 체크하고
- Statically set IP address항목을 선택하여 Address, Subnet mask, Default gateway address, Primary DNS, 그리고 secondary DNS를 입력한다. 반드시 공인 IP이여야한다. OK를 클릭하고 File > Save를 클릭한다. 그리고 Quit를 하고 나온다.

본 예제에서는

Address : 163.152.197.67

Subnet mask : 255.255.255.0

Default gateway address : 163.152.62.1

Primary DNS : 163.152.1.1

Secondary DNS : 163.152.11.6

으로 한다.

♠ **내부로 연결된 네크워크 장치 설정**

(본 책에서는 eth0이 내부로 연결되어 있다.)

- Terminal로 들어가 `system-config-network`를 입력한다.
- 그 다음 내부로 연결된 장치(et0)를 더블클릭
- Controlled by NetworkManager와 Activate device when computer starts를 체크하고
- Statically set IP address항목을 선택하여 Address, Subnet mask를 입력한다. 내부로 연결된 컴퓨터만 통신을 하기 때문에 어떤 IP를 주소를 입력해도 된다. 단 다른 계산 컴퓨터와 같은 IP는 불가능하다. OK를 클릭하고 File > Save를 클릭한다. 그리고 Quit를 하고 나온다.

본 예제에서는

- 관리 컴퓨터인 경우(그림 4.37)

Address : 192.168.111.111

Subnet mask : 192.168.0.100

- 계산 컴퓨터인 경우(그림 4.38)

Address : 192.168.111.222

Subnet mask : 192.168.0.100

으로 한다.

♠ **Network 환경 적용**

앞서 설정해놓은 network 환경을 적용하기 위해

- Terminal로 들어가 `service network restart`를 입력한다.

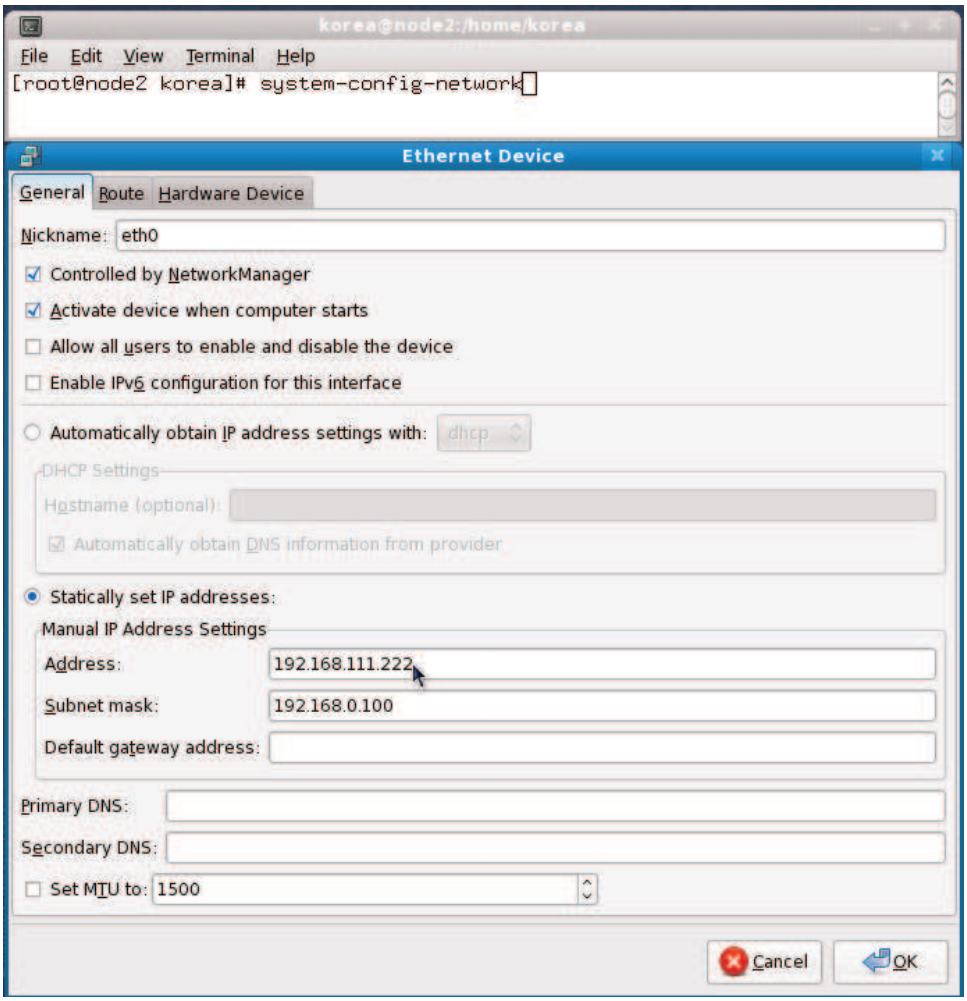


그림 4.38: 계산 컴퓨터 내부로 통신 할 Network 장치 설정

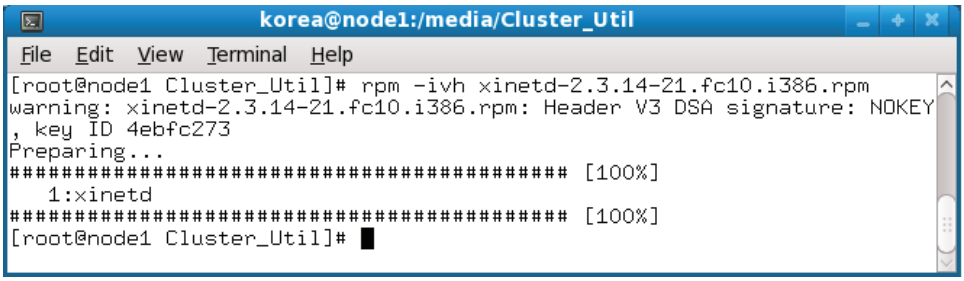


그림 4.40: Rpm Xinetd 설치

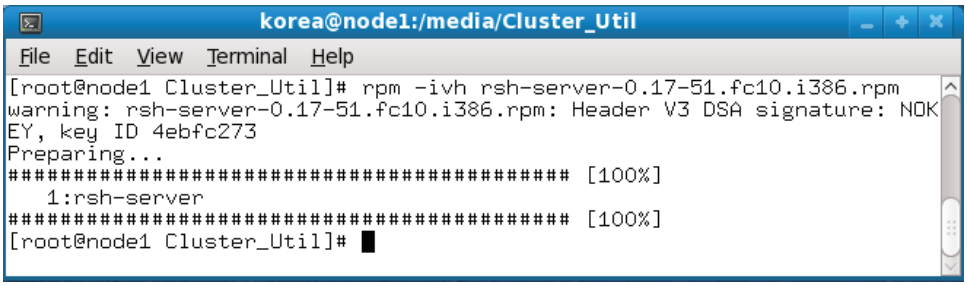


그림 4.41: Rpm Rsh-Server 설치

♠ LAM-MPI 설치 및 설정

`tar -C /home/korea/Download -zxvf lam-7.1.5b2.tar.gz` 을 입력한다. -C 옵션은 압축을 풀 장소를 지정하는 옵션이다. korea는 계정이름이므로 만든 계정에 따라 달라질수 있다.

- 압축을 푼 곳 `lam-7.1.5b2` 폴더로 이동한다.

  본 책에서는 `cd /home/korea/Download/lam-7.1.5b2/`를 입력하면 된다.

♠ Configure

현재 OS의 종류나 컴파일러 위치, 종류 등을 파악하고, 사용자가 컴파일이 완료된 프로그램의 위치를 지정하거나, 기타 등등 환경을 맞춰서 자신이 원하는 makefile을 만들어내는 과정이다.

- 앞서 이동한 `lam-7.1.5b2` 폴더에서

  `./configure --prefix=/usr/local/lam`을 입력하고 엔터키를 친다.

--prefix옵션은 설치를 할 곳을 지정하는 것이다. 컴퓨터에 따라 다르겠지만, 약 5분정도 소요된다.

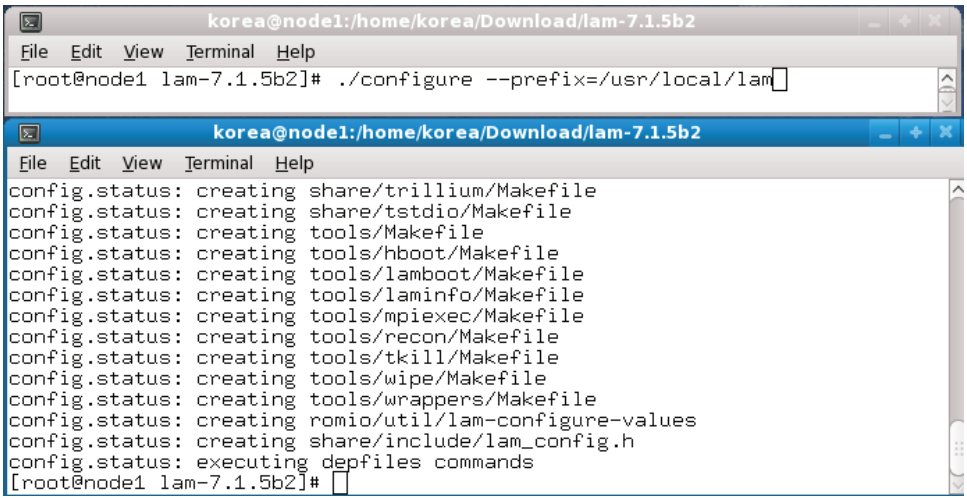


그림 4.44: configure

♠ **make**

make 명령어는 LAMMPI를 Build한다.

- Terminal에서 `make`를 입력한다.

♠ **make install**

make 명령어로 만들어진 binary파일을 지정된 Directory로 위치시키는 과정이다.

- Terminal에서 `make install`를 입력후 엔터키를 친다.

이제, /etc/bashrc파일을 수정하여 LAMMPI 환경설정을 한다.

- Terminal에서 `vi /etc/bashrc`를 입력후 엔터키를 친다. 그 다음

```
LAMHOME=/usr/local/lam
PATH=$PATH:/usr/local/lam/bin
export  LAMHOME PATH
```

을 그림 4.47과 같이 마지막 줄에 추가한다.

vi에서 입력은 키보드 i키를 누르고 시작하면 되고, 저장하고 나갈려면 ESC키 를 누른 다음 Shift키와 :키를 동시에 누른 다음 wq를 쓰고 엔터키를 친다. 마지막줄에서 export 다음에 간격은 Tab키를 사용한다.

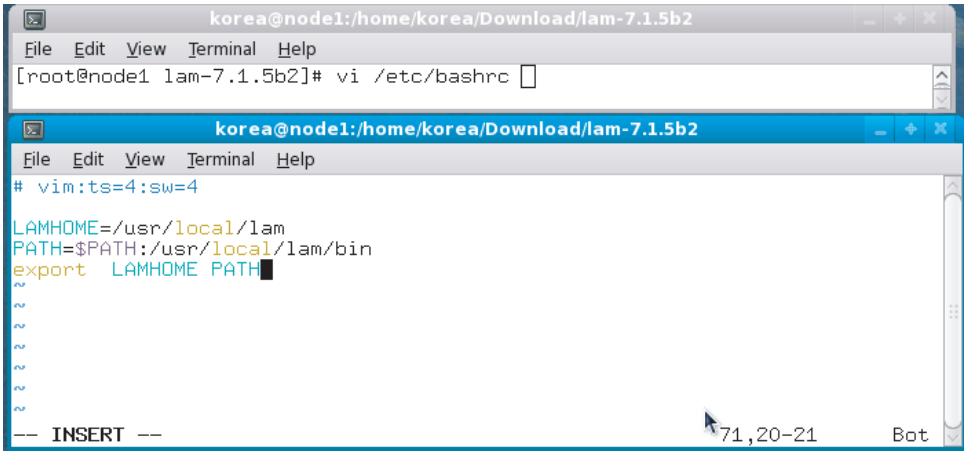


그림 4.47: LAMMPI 환경 설정

bashrc파일을 적용하기 위해

- Terminal에서 `source ~/.bahsrc`를 입력한다.

그림 4.48: 수정한 bashrc 파일 적용

♠ ntsysv 설정

- Terminal 에서 `ntsysv`을 입력한다. 그러면 부팅시 데몬을 자동으로 실행 할 수 있는 항목들이 나온다. **nfs, rexec, rlogin, rpcbind, rsh**가 체크되었는지 확인하고, 체크표시가 없다면스페이스바를 이용

인증된 원격 호스트로부터 허용된 포트를 이용하는 인증된 사용자의 로그인을 허용하는 서비스이다. 즉, 원격 로그인 서비스이다.

rsh

원격쉘(remote shell)로서 r-commands 의 일종이다. 즉, 인증된 원격 호스트로부터 허용된 포트를 이용하는 인증된 사용자의 원격쉘을 사용할 수 있는 서비스이다.

rpcbind

시스템이 내장된 RPC를 사용하기 위하여, 준비된 서비스를 반드시 등록되어야 한다. rpc데몬은 시스템에서 RPC서비스를 관리한다.

이제, ntsysv로 변경된 내용을 적용하자.

- Terminal 에서 `service xinetd restart`을 입력한다. 그림 4.53에서 볼 수 있듯이 Starting xinetd항목만 OK이면 성공적으로 된 것이다.

그림 4.50: Xinetd Restart

- Exports 설정 (관리 컴퓨터만 설정)

  Terminal에서 `vi /etc/exports`를 입력후 엔터키를 친다.
  /etc/exports파일을 수정하면 파일에 마운트를 허가할 디렉토리와 마운트를 허가할 호스트 목록을 설정할 수 있다.

```
/home  node1(rw,async,no_root_squash)
/home  node2(rw,async,no_root_squash)
```

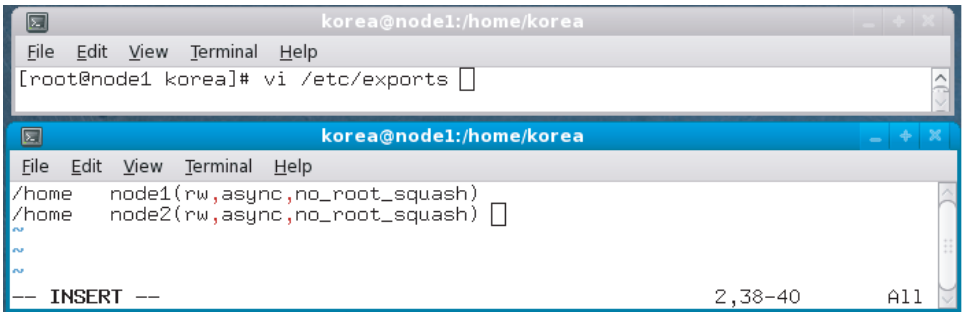


그림 4.52: Exports

Terminal에서 `service nfs restart`를 입력한다. Starting NFS항목들만 OK로 나오면 성공한 것이다.

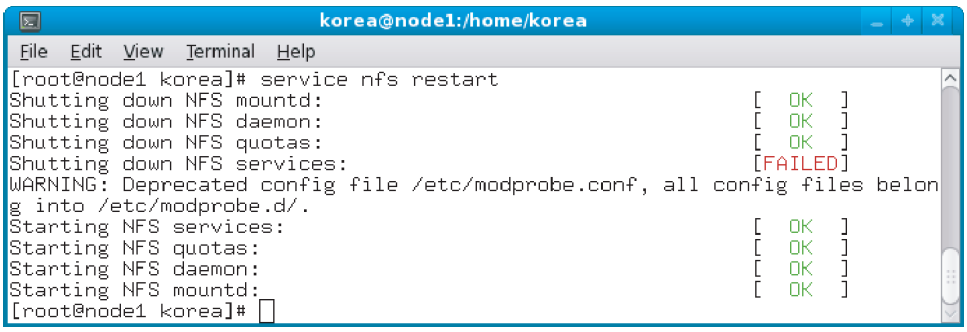


그림 4.53: NFS Restart

portmapper 데몬 확인

RPC portmapper은 RPC 프로그램 번호를 TCP/IP(혹은 UDP/IP) 프로토콜 포트 번호로 변환하는 서버이다. 이것은 머신상의 RPC 서버들을

접근할 Host 등록

서로 접근할 호스트를 /etc/hosts.equiv에 등록한다. 물론 host이름으로 사용하려면 /etc/hosts에 미리 등록해놓아야 한다.

- Terminal에서 `vi /etc/hosts.equiv`를 입력한다.
  본 예제에서는 node컴퓨터 2대 존재하며, host를 node1과 node2로 정의했으므로, 그림 4.56과 같이 **node1, node2**를 입력한 후 저장하자.

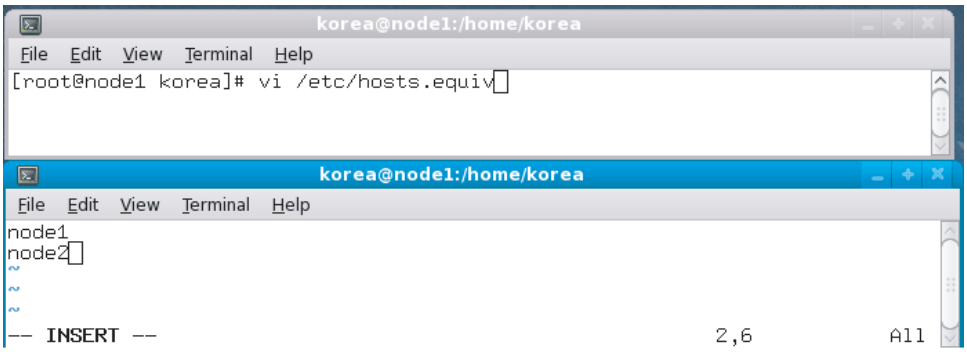


그림 4.56: 접근할 Hosts

♠ 부팅시 자동 mount 설정

- Terminal에서 `vi /etc/rc.local`을 입력한다. 마지막 줄에 다음을 입력하자.

```
sleep 7
mount -a
```

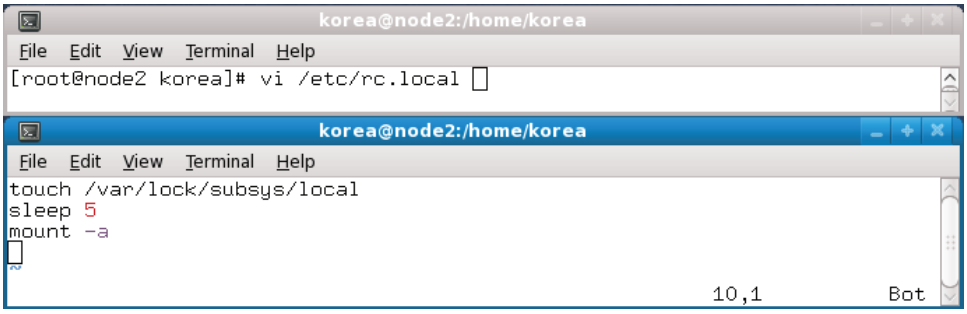


그림 4.58: 부팅시 자동 Mount 설정

♠ 수동 mount 설정

자동 mount 설정이 되지 않을 경우, 수동으로 mount 설정을 해야 한다.

- Terminal에서 `mount -t nfs node1:/home /home`을 입력한다.

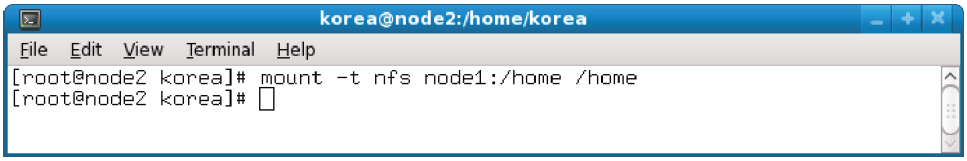


그림 4.59: 수동 Mount 설정

lamboot

LAM/MPI를 이용하여 MPI Job을 실행 시키기 위해서는 계산에 참여하는 모든 Node에 대해 lamd(LAM demon)가 실행되어야 하며, lamboot이라는 명령이 이러한 역할을 한다. 또한 사용할 Cluster Node에 대한 정보가 있어야 한다.

- 관리 컴퓨터 Terminal에서 root계정으로 `vi lamhosts`를 입력한다. 파일이름은 어떠한 이름이어도 상관없다.

  `node1`

  `node2`

  를 입력한다.

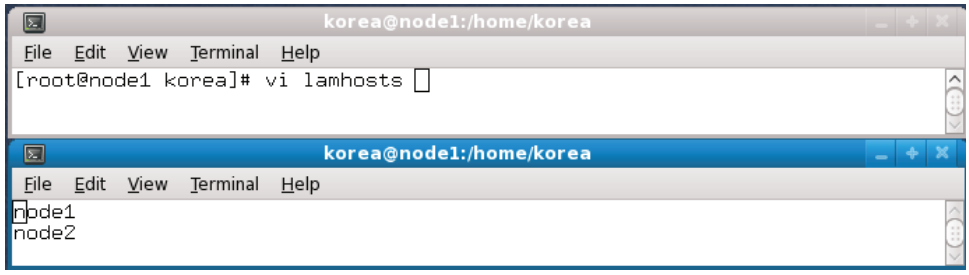


그림 4.62: Cluster Node 정의

그 다음 lamboot명령어로 lamd를 실행시킨다.

- Terminal에서 일반계정으로 `lamboot -v lamhosts`(위에서 생성한 파일)를 입력한다.

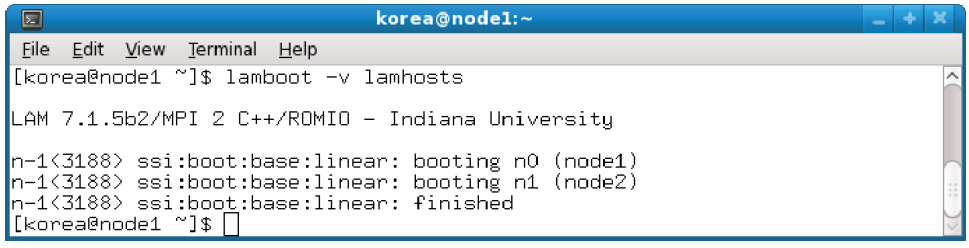


그림 4.63: Node컴퓨터 정의

그림 4.65: greetings.c Compile

그림 4.66: greets 실행

이 있다면, rank로 $0, 1, \cdots, p-1$을 갖게 될 것이다. 그리고 여기서 프로세스 0에게 메세지 전달은 0을 제외한 다른 프로세스들이 시행한다.

`MPI_Send`와 `MPI_Recv`를 이용한 간단한 프로그램을 살펴보자.

```
#include <stdio.h>

main()
{
printf("Hello World!\n");
}
```

MPI Compiler인 mpicc로 Compile한다.

- Terminal에서 일반 계정으로
  `mpicc -o greetp greetingp.c`를 입력한다.

‘

Parallel Program 실행

mpirun명령어로 compile해서 생긴파일을 실행한다.

```
{
  int id, p;
  double wtime;
  MPI_Init ( &argc, &argv );
  MPI_Comm_rank ( MPI_COMM_WORLD, &id );
  MPI_Comm_size ( MPI_COMM_WORLD, &p );

  printf ("Solve the 1D time-dependent heat equation from
        node %d.\n",id);
  update(id, p );
  MPI_Finalize ( );
  return 0;
}
void update (int id, int p)
{
  int i, it, it_max = 100, n = 5, tag;
  double pi, time_delta, x_delta,
         x_max = 1.0, x_min = 0.0, *h, *h_new;
  FILE *x_file, *h_file;
  MPI_Status status;

  pi=4.0*atan(1.0);
  time_delta = 0.001;
  x_delta=(x_max-x_min)/(double)(p*n+1);

  h=(double*)malloc((n+2)*sizeof(double));
  h_new=(double*)malloc((n+2)*sizeof(double));

  for (i=0; i<=n; i++) {
```

```
  printf("node %d %f\n",id,h[i+id]);}
  free ( h );
  free ( h_new );
  return;
}
```

프로그램의 실행은 현재 heat1d.c 파일이 있는 위치에서 앞서 설명한 순서대로 동일하게 하면된다.

```
mpicc -o heat heat_mpi.c -lm
```

그림 4.69: heat_mpi.c Compile

위의 명령어에 의해 heat이라는 실행파일이 만들어진다. 그 실행파일을 2개의 프로세스에 의해 실행되도록 명령하자.

```
mpirun -np 2 heat
```

이 프로그램의 결과는 그림 4.70과 같이 화면에 나타난다.

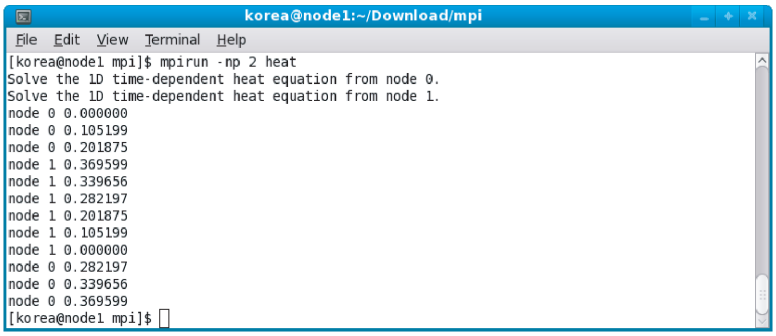


그림 4.70: heat 실행

참고 문헌

[1] J. Bell, P. Collela, and H. Glaz, A second-order projection method for the incompressible Navier-Stokes equations, J. Comput. Phys. 85 (1989) 257-283.

[2] A.J. Chorin, A numerical method for solving incompressible viscous flow problems, J. Comput. Phys. 2 (1967) 12-26.

[3] F.H. Harlow and J.E. Welch, Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface, Phys. Fluids 8 (1965) 2182-2189.

[4] J. Li and Y. Renardy, Numerical study of flows of two immiscible liquids at low Reynolds number, SIAM Rev. 42 (2000) 417-439.

[5] N.A. Patankar, P. Singh, D.D. Joseph, R. Glowinski, T.W. Pan, A new formulation of the distributed Lagrange multiplier/fictitious domain method for particulate flows, Int. J. Multiphase Flow 26 (2000) 1509-1524.

[6] 서용권, 강상모, 서이수 저, **전산유체역학**, 동아대학교 출판부.

[7] Brandimarte P., *Numerical Methods in Finance and Economics*, Wiley, 2/E, 2006

[8] Clewlow L., Strickland C., *Implementing Derivatives Models*, John Wiley & Sons, 1998

찾아보기