The Asian Journal of TEX, Volume 1, No. 1, April 2007
Article revision 2007/04/01

PSTricks를 이용한 함수의 플로팅: 효용극대화 모형을 중심으로

Plotting Functions with PSTricks: A New Macro \uMaxCD for the Utility Maximization Problem

조인성 In-Sung Cho
공주대학교 경제통상학부 ischo@ktug.or.kr

ABSTRACT Though the PostScript language is very powerful to plot functions with PSTricks, its Reverse Polish Notation(RPN) convention is somewhat difficult to get used to. This article examines how to plot functions with PSTricks using general algebraic notation. Some examples of plotting functions in economics are provided. This article also introduces a new macro \uMaxCD, based on RPN, which provides an intuitive tool to plot graphics related to the utility maximization problem.

1 서론

LaTeX이 자체로 그래픽 기능을 가지고 있지만, LaTeX의 그래픽 기능은 매우 제한적이다. PSTricks는 포스트스크립트(PostScript)를 기반으로 LaTeX의 제한적인 그래픽 기능을 강화시키는 TeX 매크로의 집합체이다. PSTricks는 프로그래밍 언어인 포스트스크립트를 TeX에서 직접 사용할 수 있는 환경을 제공한다는 점에서 보통의 그래픽 패키지와는 다르다. 또한 PSTricks를 이용하면 TeX 소스에 그림을 그리기 위한 코드를 직접 쓸 수 있으므로, 외부의 그래픽 패키지로 그린 그림을 TeX 파일 내로 불러들이거나 따로 관리하는 번거로움을 피할 수 있다.

TeX에서 포스트스크립트 언어를 직접 사용할 수 있도록 연결해주는 기본적인 패키지는 `pstricks` 패키지이다. `pstricks` 패키지는 일종의 기지(base)이며, 이를 기반으로 하여 그래픽 기능을 개선하거나 확장하는 수십 개의 부가 패키지가 개발되어 있다. 부가 패키지는 `pst-plot`처럼 `pst-`␣␣␣␣ 형식의 이름을 가지는 경우가 보통이다. PSTricks 패키지라 하면 `pstricks` 패키지를 포함하여 관련된 많은 부가 패키지를 통칭하는 것이라 할 수 있다. PSTricks 관련 패키지 중에서는 `multido` 패키지와 같이 `pstricks`와 함께 쓸 목적으로 만들어졌지만, `pstricks`와 독립적으로 쓸 수 있는 패키지도 있다.

이 글에서는 PSTricks가 제공하는 그래픽 기능 중에서 함수를 플롯하는 기능에 집중하고자 한다. `pst-plot` 패키지를 로드하면 포스트스크립트 언어로 함수를 정의하여 정확성이 담보된 함수의 그래프를 그릴 수 있다. 포스트스크립트는 RPN(Reverse Polish

Notation, 역 폴란드식 표기법)이라 불리는 방식으로 연산을 한다. RPN 방식이란 연산자(operator)가 인자(argument)의 뒤에 표기 되는 방식이다. 이를 후위(postfix) 표기라 하기도 한다. '$x \times y$'를 표현하는 것을 예로 들면, 일반적인 삽입(infix) 표기 또는 대수적(algebraic) 표기로는 'x*y'와 같이 쓰지만, 포스트스크립트에서는 'x y mul'과 같이 쓴다. 또 다른 예를 들면, '$\ln x$'를 대수적 표기로는 'ln(x)'로 쓰지만, 포스트스크립트에서는 'x ln'으로 쓴다.

pst-plot 패키지를 이용하면 포스트스크립트 언어를 직접 이용하여 함수를 플롯하는 강력함을 얻을 수 있는 반면, RPN 방식이 익숙하지 않아 불편하다는 점이 단점으로 지적되고 있다. pstricks-add 패키지는 \psplot 명령과 함께 algebraic 옵션을 이용하여 대수적 표기로 함수를 정의하고 플롯할 수 있는 매우 편리한 기능을 제공한다.[1] pstricks-add 패키지는 이 이외에도 임의의 점에서 함수의 접선을 플롯할 수 있게 하는 \psplotTangent 매크로를 포함하여 매우 다양한 기능을 제공하고 있다.

pst-func 패키지는 자주 쓰는 복잡한 함수를 간편하게 플롯할 수 있는 매크로를 제공한다. 특히 다항함수를 간편하게 플롯할 수 있게 하는 \psPolynomial 매크로를 다양한 옵션과 함께 제공하고 있다. 또한 \psGauss 매크로를 포함하여 다양한 특수함수를 간편하게 플롯할 수 있는 매크로를 제공한다.

pst-func 패키지는 pstricks-add 패키지를 자동검색하여 로드하고, pstricks-add 패키지는 pst-plot 패키지를, pst-plot 패키지는 다시 pstricks 패키지를 자동으로 검색하여 로드한다. 따라서 pst-func를 로드하면, 앞에서 언급한 모든 기능을 다 사용하여 함수를 플롯할 수 있다.

한편, pst-math 패키지도 여러 가지 특수한 함수를 간편하게 플롯할 수 있는 기능을 제공한다. pst-math 패키지는 다른 패키지와 병행해서 로드하여야 한다. 다만, 매뉴얼에 의하면, pstricks-add 패키지는 모든 PSTricks 관련 패키지의 맨 마지막에 로드하는 것이 안전하므로, pstricks-add 패키지를 자동검색하여 로드하는 pst-func도 맨 마지막에 로드하여야 한다. 이 글은 다음과 같은 PSTricks 관련 패키지들을 로드하여 작성되었다.

```
1 \usepackage{pst-math}
2 \usepackage{pst-func}
```

이 글의 첫째 목적은 PSTricks를 이용하여 함수를 플롯하는 다양한 기법을 소개하는 것이다. 위에서 언급한 패키지에서 제공하는 대수적 표기의 플로팅 기법을 중심으로 논의한다. 또한 이러한 기본 기법들이 경제학에서 빈번히 볼 수 있는 함수를 플롯하는 데에 어떻게 적용되는지를 살펴본다. 이 글의 논의 과정에서 나타나는 모든 '그림'에 대해서는 '소스'가 함께 제공됨으로써 내용을 이해하는 데에 도움이 되도록 하였다.

함수의 플로팅 기법에 대한 논의를 기반으로, 효용극대화 모형을 간편하게 플롯하는 새로운 매크로를 제시하는 것이 이 글의 두번째 목적이다. 효용극대화(ulility maxmiza-

1. pst-infixplot 패키지에서 \psPlot 매크로를 이용하면 삽입 표기로 함수를 정의하여 플롯할 수 있다. 그러나 이 패키지는 다른 부가기능을 제공하지 않아 이 패키지의 기능을 모두 포함하는 pstricks-add 패키지만 논의하는 것으로 충분하다. pst-infixplot 패키지의 기능에 대해서는 매뉴얼 [4]를 참고할 수 있다.

tion) 모형을 플롯하는 새로운 도구로 실험적인 \uMaxCD 매크로를 제시한다. 새로운 매크로를 정의하는 데 있어서는 RPN 방식을 사용한다. 이는 프로그래밍 언어로서의 포스트스크립트의 장점을 이용하기에는 RPN 방식이 효과적이기 때문이다. 콥더글러스(Cobb-Douglas) 효용함수를 기반으로 한 \uMaxCD 매크로는 효용극대화의 균형상황을 직관적으로 쉽게 플롯할 수 있게 하며 비교정태분석(comparative statics) 등 관련 내용을 플롯하는 데에 다양하게 응용될 수 있다.

이 글의 구성은 다음과 같다. 2절에서 PSTricks의 기초 개념을 소개하고, 3절에서는 함수 플로팅의 기본 기법을 논의하고, 4절에서는 기존 패키지에서 편리하게 정의한 매크로의 사용법을 논의한다. 5절에서는 효용극대화 모형을 간편하게 플롯할 수 있는 새로운 매크로 \uMaxCD를 제시하고 그 사용법을 논의한다. 6절에서는 결론을 맺는다.

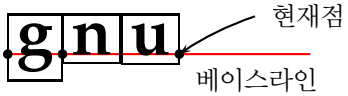
2 PSTricks 기본

현재점

PSTricks를 이용하여 그리는 선이나 원 등의 그래픽 객체(graphics object)는 현재점[2]에 0-차원 박스를 형성한다. pstricks 코드가 텍스트 상에 식자될 때의 현재점은 TEX이 배열할 박스의 베이스라인(baseline)이 자리잡을 곳이다.[3] PSTricks가 그리는 그래픽 객체는 0-차원 박스를 형성하므로 현재점을 변경시키지 않는다. 따라서, 현재점을 공유하는 여러 개의 그래픽 객체를 모아 하나의 그림(picture)을 만들어 갈 수 있는 것이다 [8, p. 41].

아래 그림에서 TEX은 \fbox로 만들어진 세 개의 박스를 배열하고 있다. g의 전과 후에 표시된 점은 TEX이 g를 배열하기 전과 후의 현재점을 나타내고 있다. TEX이 u까지 모두 배열한 후의 현재점도 표시되어 있다. 수평선은 베이스라인의 위치를 나타내고 있다. 세 개의 박스 이외의 점과 선 등은 모두 PSTricks 코드로 그려져 있지만 0-차원이므로 자리를 차지하지 않으며, 따라서 TEX이 배열하는 박스의 위치에 영향을 주지 않는다.

```
{\Huge\bfseries
  \psline[linecolor=red](0,0)(4,0)% baseline
  \psdot(0,0)\fbox{g}\psdot(0,0)%
  \fbox{n}\fbox{u}\psdot(0,0)%
  \pcarc{<-}(0,0)(1,.5)% pointing arrow
}
\uput[r](1,.5){현재점}
\uput[d](1,0){베이스라인}
```



2. "현재점(current point)"은 포스트스크립트가 그림을 그리기 위해 유령 펜(phantom pen)의 끝을 놓는 점이다 [2, p. 18].

3. pstricks 코드가 식자된 곳에서, TEX이 다음 박스를 정렬하는 기준이 되는 정렬점(reference point)이 현재점이 된다. 이 글에서는 편의를 위해 TEX의 정렬점과 현재점을 구별하지 않고 사용한다. 그러나 pspicture 환경 안에서 \moveto 등의 명령으로 현재점을 이동시킬수 있다는 점에서 TEX의 정렬점과 현재점은 서로 다른 역할을 가진다.

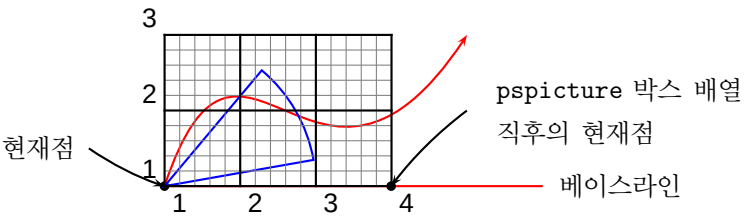
pspicture 환경

PSTricks 코드가 그리는 그래픽 객체는 자리를 차지하지 않으므로, 그래픽 객체를 텍스트와 겹쳐지지 않게 그리기 위해서는 일정한 영역을 확보하여 그 곳에 그래픽 객체가 위치하도록 할 필요가 있다. pspicture 환경은 박스의 크기를 정하고 그 크기만큼의 자리를 차지한다. 이 환경 안에 PSTricks 코드를 넣으면, 텍스트와 겹치지 않는 그림을 그릴 수 있다. pspicure 환경의 구문(syntax)은 다음과 같다.

```
\begin{pspicture}[<options>](xMin,yMin)(xMax,yMax)
  <pstricks codes>
\end{pspicture}
```

pspicture 환경은 왼쪽 아래의 좌표 (xMin,yMin)과 오른쪽 위의 좌표 (xMax,yMax)를 명시하여 박스의 크기를 정한다. 왼쪽 아래의 좌표 (xMin,yMin)은 현재점에 주어지는 좌표이다. 왼쪽 아래의 좌표가 생략되면 기본값인 $(0,0)$을 왼쪽 아래의 좌표로 간주한다. TEX은 pspicture가 정하는 크기의 박스를 현재점에 배열한다. 즉 TEX의 입장에서 pspicture 환경은 가로 $(\texttt{xMax}-\texttt{xMin})$, 세로 $(\texttt{yMax}-\texttt{yMin})$의 크기를 가진 박스에 불과하다.

그림 1에는, 현재점을 공유하는 \psdot, \psbezier, 그리고 \pswedge가 그려져 있으며, 이 그래픽 객체들은 0-차원으로 현재점을 변화시키지 않는다. 또한 이 그래픽 객체들과 현재점을 공유하는 pspicture 환경 박스가 그려져 있다. 이 박스는 왼쪽 아래의 좌표 (1,1)과 오른쪽 위의 좌표 (4,3)에 의해 형성되는 사각형만큼의 자리를 차지한다. pspicture 박스를 시각적으로 표현하기 위해 박스의 크기와 일치하는 격자(grid)를 \psgrid로 그렸다. pspicture 박스가 그려지면 현재점이 이동한다. 그림 1에서는 박스가 그려진 이후의 현재점이 점으로 표시되어 있다.




```
1 \psdot(0,0)% current point
2 \psbezier[linecolor=red]{->}(0,0)(1,3)(2,-1)(4,2)
3 \pswedge[linecolor=blue]{2}{10}{50}
4 \begin{pspicture}(1,1)(4,3)
5   \psgrid
6 \end{pspicture}
7 \psdot(0,0)% dot after pspicture box
```

그림 1. pspicture 환경과 현재점

크기가 큰 그래픽 객체는 pspicture 박스의 영역 밖으로 나가서 표시될 수 있다. pspicture 환경이 차지하는 영역 밖의 그림을 잘라내려면 *버전인 pspicture* 환경을 사용한다. 103 쪽의 그림 16에 pspicture* 환경이 사용되고 있다.

\psgrid 매크로

그림 1에서 사용된 \psgrid 매크로에 대해 살펴보기로 하자.[4] \psgrid의 구문은 다음과 같다.

```
\psgrid[<options>](x0,y0)(xMin,yMin)(xMax,yMax)
```

격자의 크기는 좌표 (xMin,yMin)과 (xMax,yMax)에 의해 결정되며 (x0,y0)에서 격자 라벨이 교차된다. (x0,y0)이 생략되면 (xMin,yMin)와 같은 것으로 간주되고 둘 다 생략되면 두 좌표 모두 기본값인 (0,0)으로 대체된다. pspicture 환경 안에서 \psgrid의 모든 좌표가 생략되면 pspicture 환경이 지정하는 (xMin,yMin)과 (xMax,yMax)의 값을 그대로 이용한다. pspicture 환경 밖에서 모든 좌표가 생략되면 (xMax,yMax)=(10,10)이 기본값이다.

\psgrid를 배경으로 표시하면, 그래픽 객체의 위치를 입력하거나 확인하는 데 편리하다. 격자와 부(sub)격자의 색이나 굵기 등은 옵션으로 설정을 조절할 수 있다. 격자의 설정은 \newpsstyle로 정의해 두고 반복하여 사용할 수도 있다. 이 글에서는, 편의상, 배경의 격자를 표시하기 위해 \newpsstyle 명령으로 gridstyle을 프리앰블(preamble)에서 다음과 같이 정의하여 두기로 한다.

```
1 \newpsstyle{gridstyle}{griddots=10,subgriddiv=0,%
2                        gridcolor=lightgray,gridlabels=0pt}
```

새로 정의한 스타일은 필요한 곳에서 \psgrid[style=gridstyle]과 같이 사용하면 된다. 이 글의 그림들에서 \psgrid가 사용된 예를 쉽게 찾아볼 수 있다.

전형적 매크로의 형태

위에서 소개한 \psgrid 매크로의 구문에서 볼 수 있듯이 PSTricks의 전형적인 매크로는 다음과 같은 구조를 가진다.

```
\macro_command[<options>](<coordinates>){<stuff>}
```

- <options> : 옵션은, 예를 들어, linewidth=1.5pt와 같이 '파라미터=값'의 형태를 가진다. 파라미터의 값을 지정하면 기본값에 우선하여 적용된다. 참고로 linewidth의 기본값은 0.8pt이다.

4. \psgrid 매크로에 대한 자세한 설명은 pstricks 매뉴얼[8, p. 17]을 참조하라.

- <coordinates> : 데카르트 직교좌표(Cartesian coordinate) (x, y)가 주로 쓰인다. \SpecialCoor가 선언된 경우, (r, θ) 형태의 극좌표나 포스트스크립트 코드로 표현되는 특수한 좌표 (!postscript code)도 사용된다.[5] pstricks-add 패키지나 pst-func 패키지가 로드된 경우에는 \SpecialCoor 선언을 생략하여도 특수한 좌표를 쓸 수 있다.
- <stuff> : \rput 또는 \uput 등의 매크로가 배치하려고 하는 텍스트 등의 대상을 말한다. \psline이나 \pscurve 등 <stuff>를 갖지 않는 매크로도 있다.

단일 매크로에 [<options>]로 옵션을 설정하면 설정내용이 그 매크로에만 영향을 미친다. \psset 매크로를 이용하면 여러 매크로에 영향을 미치도록 옵션을 설정할 수 있다.

\psset 매크로

\psset 매크로로 옵션을 설정하면 \psset이 사용된 이후의 모든 명령에 유효한 옵션이 된다. \psset을 pspicture 밖에서 사용하면, 새로 옵션을 지정하지 않는 한, 문서 전 영역에서 유효하고, pspicture 환경 내에서 쓰면 환경 안에서만 유효하다. 물론, 설정된 옵션은 [<options>] 또는 \psset을 이용하여 설정을 다시 바꿀 수 있다.

PSTricks 패키지가 사용하는 기본적인 길이단위는 1cm이고 각도 단위는 도(degree)이다. \psset 매크로를 이용하여 길이단위인 unit을 다르게 설정할 수 있다. 길이단위는 가로단위와 세로단위를 xunit과 yunit으로 따로따로 설정할 수 있다. 또한 길이단위를 절대적 크기로 지정할 수도 있고 상대적인 크기를 조절할 수도 있다.

- \psset{unit=10cm}와 같이 절대적으로 설정한 후 \psset{unit=.5}로 축소시키면 절대단위를 \psset{unit=5cm}로 설정하는 것과 같은 효과를 갖는다.
- \psset{xunit=.7,yunit=1.2cm}로 입력하면 현재의 가로단위를 70%로 축소시키고 세로단위를 1.2cm로 설정한다.

3 함수 플로팅의 기본 기법

이 절에서는 함수를 플롯하는 기본 기법을 소개한다. 결론부터 말하면, 일반적인 대수적 표기 방식으로 편리하게 함수를 플롯하려면 pstricks-add 패키지에서 algebraic 옵션과 함께 \psplot 매크로를 사용하여 플롯한다.

3.1 pstricks 패키지

\pscurve 매크로

pstricks 패키지를 로드하고 \pscurve 명령을 사용하면 평면상의 점들을 연결하여 곡선을 그릴 수 있다. 점은 순서쌍 (x, y)로 나타내는 것이 보통이지만, \SpecialCoor 명령을 선언한 후에는 $(r; \theta)$ 형식의 극좌표나 (!<postscript code>) 형식의 특수 좌표도

5. \SpecialCoor와 특수 좌표에 대해서는 pstricks 패키지 매뉴얼[8, pp. 71–73]을 참조하라.

함께 쓸 수 있다. 단, pstricks-add 패키지가 로드된 상태에서는 \SpecialCoor를 선언하지 않아도 특수 좌표를 쓸 수 있다는 것을 기억할 필요가 있다. pstricks 패키지에서 \pscurve 매크로를 사용하는 구문은 다음과 같다.

```
\pscurve[<options>](x0,y0)(x1,y1)...(xn,yn)
```

```
\usepackage{pstricks} % preamble
%%
\begin{pspicture}(-.5,-.5)(5,4)
  \psgrid[style=gridstyle]
  \pscurve[showpoints=true](0,0)(1,.4)(3,3)(4,2)
  \psline{->}(-.5,0)(5,0)   % x-axis
  \psline{->}(0,-.5)(0,3.5) % y-axis
\end{pspicture}
```

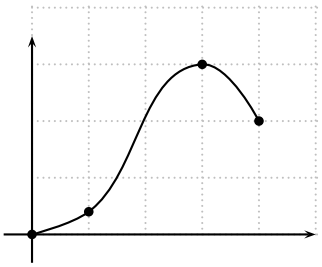


그림 2. \pscurve로 그리기

그림 2는 \pscurve 매크로로 네 점을 연결한 곡선을 보여주고 있다. 네 점을 시각적으로 확인하기 위하여 옵션 showpoints=true를 설정하였다.

\dataplot 매크로로

좌표를 연결하여 곡선을 플로팅하는 경우, 좌표를 외부의 파일로부터 읽어 들여서 플롯할 수도 있다. 그림 3에서와 같이 \readdata 매크로로 자료를 읽어 들이고 \dataplot 매크로로 좌표를 연결하는 선을 그리는 것이 한 가지 방법이다.[6]

```
\begin{pspicture}(-.5,-.5)(5,3)
  \psgrid[style=gridstyle]
  \psset{xunit=.5,yunit=8}
  \psaxes[dx=2,Dx=2,dy=.1,Dy=.1]
         {->}(0,0)(-1,-.05)(11,.35)
  \readdata{\unemp}{unemppdf.dat}
  \dataplot[linewidth=1.2pt]{\unemp}
\end{pspicture}
```

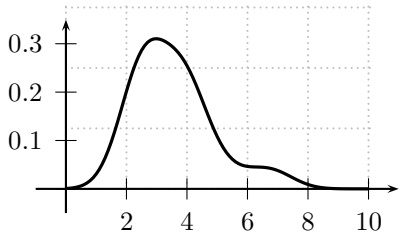


그림 3. \dataplot으로 그리기

그림 3은 실업률 자료로부터 Mathematica를 이용하여 실업률의 확률밀도함수를 추정하고 이 추정치로부터 좌표를 추출하여 파일 unemppdf.dat로 저장하고 이를 \readdata 명령으로 읽어 들여서 플롯한 것이다.[7]

6. \dataplot 매크로 이 외에 외부의 자료를 플롯하는 방법으로 \fileplot과 \listplot이 있다. 이에 대해서는 pstricks 매뉴얼[8, p. 20]과 pstricks-add 매뉴얼[6]을 참고하라.
7. 김우영 · 조인성[1] 참조.

표 1. RPN 방식과 대수적 표기

RPN 방식	대수적 표기	의미
`1 2 div`	`1/2`	$1/2$
`x 1 add`	`x+1`	$x+1$
`1 2 div x 1 add mul`	`(1/2)*(x+1)`	$1/2 \cdot (x+1)$
`x 1 sub`	`x-1`	$x-1$

3.2 pst-plot 패키지

pstricks 패키지에서 그래프 상의 좌표를 일일이 입력하지 않고 함수를 수학적으로 정의하여 플롯할 수 있다. 이를 위해서는 부가 패키지 pst-plot를 로드해야 한다. pst-plot 패키지는 pstricks 패키지를 자동검색하여 로드해준다.

\psplot 매크로

pst-plot 패키지를 로드한 상태에서 \psplot 명령을 이용하여 정의된 함수의 그래프를 플롯하는 구문은 다음과 같다.

```
\psplot[<options>]{xStart}{xEnd}{<function>}
```

\psplot 매크로는 <function>으로 정의된 함수의 그래프를 xStart에서 xEnd까지 플롯한다. PSTricks에서는 포스트스크립트가 함수를 표현하는 방식, 즉 RPN 방식으로 함수를 표현한다. 따라서 함수의 정의 <function>은 포스트스크립트 방식인 RPN 방식으로 입력해야 한다. 식 (1)을 플롯하는 문제를 예로 들어 설명하기로 하자.

$$f(x) = \frac{1}{2}(x+1)(x-1)(x-2) \tag{1}$$

포스트스크립트 코드로 식 (1)을 표현하면 다음과 같다.

```
%% 포스트스크립트 코드 (RPN 방식)
1 2 div x 1 add mul x 1 sub mul x 2 sub mul
```

위 포스트스크립트 코드를 일반적으로 많이 쓰는 삽입 표기 또는 대수적 표기로 해석하면 표 1과 같다.[8] 그림 4는 식 (1)을 플롯한 것이다.

\psaxes 매크로

pst-plot 패키지에서는 \psaxes 매크로를 사용하여 그래프의 축을 편리하게 그릴 수 있다. pst-plot 패키지의 기능은 pstricks-add 패키지를 통해 큰 폭으로 개선·확장되고

8. 포스트스크립트에서 사용하는 RPN 방식의 연산자가 의미하는 바에 대해서는 pstricks-add 매뉴얼[6]의 부록 또는 Casselman[3, 2005] 참조.

```
\usepackage{pst-plot} % preamble
\begin{pspicture}(-2,-1)(3,2)
  \psgrid[style=gridstyle]
  \psaxes[ticks=none]{->}(0,0)(-2,-1)(3,2)
  \psplot{-1.25}{2.6}{%
    1 2 div x 1 add mul x 1 sub mul x 2 sub mul}
\end{pspicture}
```

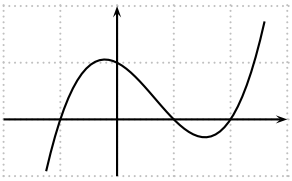


그림 4. \psplot으로 그리기

있는데, \psaxes 매크로의 경우에도 유용한 옵션이 추가되는 등 그 기능이 향상되고 있다. 따라서, \psaxes 매크로를 사용할 때에는 pstricks-add 패키지를 로드하여 개선된 기능을 사용하는 것이 효과적이다. \psaxes의 구문은 다음과 같다.

```
\psaxes[<options>]{<arrow>}(x0,y0)(xMin,yMin)(xMax,yMax)
```

\psaxes 매크로는 89 쪽에서 살펴본 \psgrid와 유사하게 동작한다. 즉 \psaxes 매크로는 (xMin,yMin)과 (xMax,yMax)에 의해 이루어지는 사각형의 영역 내에서 (x0,y0)에서 교차하는 좌표축을 그려준다. 이를 이용하면, 그림 5에서와 같이 교환경제(exchange economy)의 Edgeworth Box를 간단하게 그릴 수 있다.

```
\begin{pspicture}(-2,-1)(6,3)
  \psgrid[style=gridstyle,gridlabels=5pt]
  \psaxes[ticks=none]{->}(0,0)(0,0)(5,2.5)
  \psaxes[ticks=none]{<-}(4,2)(-1,-.5)(4,2)
  \uput{3pt}[dl](0,0){$O^A$}
  \uput{3pt}[ur](4,2){$O^B$}
  \uput[r](5,0){$x^A$}
  \uput[u](0,2.5){$y^A$}
  \uput[l](-1,2){$x^B$}
  \uput[d](4,-.5){$y^B$}
\end{pspicture}
```

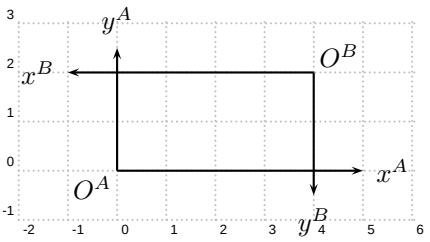


그림 5. Edgeworth Box: \psaxes

그림 5를 플롯할 때 사용된 매크로 \uput은 좌표의 라벨을 간편하게 표시하는 데 매우 유용하다. \uput의 구문은 다음과 같다.

```
\uput{<labelsep>}[<direction>]{<rotation>}(x,y){<stuff>}
```

<labelsep>의 기본값은 5pt로 정해져 있다. <direction>은 도(degree) 단위로 표시하며, r(ight), l(eft), u(p), d(own) 등 미리 정의된 각으로 간편하게 지정할 수도 있다. <labelsep>과 <rotation>은 생략될 수 있다.[9]

9. \uput 매크로에 대한 자세한 설명은 pstricks 매뉴얼[8, p. 44]을 참조하라.

표 2. 함수 정의하기

함수	RPN 정의	대수적 정의
$\sqrt{x}$	`x sqrt`	`sqrt(x)`
πx^2	`Pi x dup mul mul`	`Pi*x^2`
$\ln x$	`x ln`	`ln(x)`
$\log x$	`x log`	`log(x)`
e^x	`Euler x exp`	`Euler^x`
$\sin x$	`x sin` (degrees)	`sin(x)` (radians)
$\cos x$	`x cos` (degrees)	`cos(x)` (radians)

3.3 pstricks-add 패키지

함수를 플롯하기 위해 사용하는 RPN 방식의 포스트스크립트 언어가 강력한 도구이지만, RPN 방식이 익숙하지 않아 불편하다는 단점이 있다. 반면, 함수의 그래프를 플롯하는 데 있어서 일반적으로 익숙한 대수적 표기로 함수를 정의할 수 있으면 매우 편리할 것이다. pstricks-add 패키지에서, algebraic 옵션과 함께 \psplot 매크로를 쓰면, 포스트스크립트 방식이 아니라 대수적 표기로 함수를 정의할 수 있다. 구문은 다음과 같다.

```
\psplot[algebraic]{xStart}{xEnd}{<algebraic notation>}
```

algebraic 옵션은 algebraic=true와 같이 써도 된다. algebraic 옵션을 끄려면 algebraic=false로 쓰면 된다. 식 (1)을 포스트스크립트 코드와 대수적 표기로 표현하면 다음과 같다.

```
1 1 2 div x 1 add mul x 1 sub mul x 2 sub mul % 포스트스크립트 코드(RPN)
1/2*(x+1)*(x-1)*(x-2)                          % 대수적 표기
```

그림 4에서와 같이 식 (1)의 그래프를 그리기 위해서는 다음 두 가지 중 어느 방식을 사용해도 된다.

```
\psplot{-1.1}{2.5}{1 2 div x 1 add mul x 1 sub mul x 2 sub mul} % RPN
\psplot[algebraic]{-1.1}{2.5}{1/2*(x+1)*(x-1)*(x-2)} % 대수적 표기
```

표 2는 흔히 접할 수 있는 함수를 RPN으로 정의하는 방식과 대수적 표기로 정의하는 방식을 대비시킨 것이다.[10] 그림 6은 몇 가지 함수를 플롯하는 예를 보여주고 있다.

포스트스크립트 헤더 파일인 pstricks.pro에 포스트스크립트 방식으로 사용할 수 있는 상수 Euler와 Pi가 각각 $e = 2.71828182846$과 $\pi = 3.14159265359$의 값을 가지도록 정의되어 있으며, pstricks-add.tex에는 \def\psPi{3.14159265}로 정의되어 있다.

10. $\tan x$는 $\sin x / \cos x$로 $\log_a x$는 $\ln x / \ln a$와 같이 응용하여 표현할 수 있다.

```
\begin{pspicture}(-3,-1)(5,3)
  \psset{algebraic}
  \psaxes[Dx=\psPi]{->}(0,0)(-3,-1)(5,3.2)
  \psset{plotstyle=curve,linewidth=1.5pt}
  \psplot[linecolor=red]{-3}{4}{sin(2*x)}
  \psplot[linecolor=blue]{-3}{1}{Euler^x}
  \psplot[linestyle=dashed]{.35}{4}{ln(x)}
  \uput[r](1,2.7){$e^x$}
  \uput[r](4,1.5){$\ln x$}
  \uput[r](4,1){$\sin 2x$}
\end{pspicture}
```

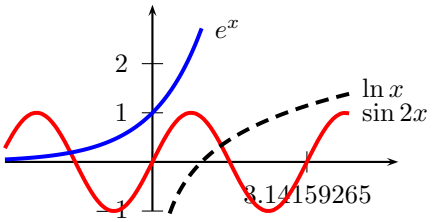


그림 6. 몇 가지 함수

4 편리한 매크로 이용하기

복잡한 함수의 경우 앞 절에서 논의한 플로팅의 기본 기법을 반복하여 적용하는 것이 매우 번거로울 수 있다. 이 절에서는 자주 쓰는 복잡한 함수를 간편하게 플로팅할 수 있도록 제공되는 몇 가지를 매크로를 다루고자 한다. 빠르게 업데이트되고 있는 pstricks-add 패키지는 \pst-plot의 기능을 보완하고 강화하는 매크로를 포함하여 여러 가지 간편한 매크로를 추가하고 있다. pst-func 패키지와 pst-math 패키지는 특정한 함수를 플롯하기 위한 간편한 매크로를 제공한다. 이들 패키지에서 제공하는 간편한 매크로 중 많이 응용될 수 있는 매크로 몇 가지를 소개하기로 한다.

4.1 pstricks-add 패키지

\psplotTangent 매크로

\psplotTangent 매크로는 RPN 또는 대수적 표기로 정의된 함수 $f(x)$의 접선을 간편하게 그려준다. 구문은 다음과 같다.

```
\psplotTangent{<a>}{<dx>}{<f(x)>}
```

\psplotTangent는 그래프상의 점 $(a, f(a))$를 중심으로 양쪽 방향으로 각각 dx의 길이만큼 접선(tangent line)을 그려준다. 즉 dx는 접선의 길이의 절반을 나타낸다. 그림 7은 \psplotTangent 매크로와 \multido 매크로를 이용하여 무차별곡선의 기울기의 변화를 나타낸 것이다.[11]

\psplotTangent 매크로에 Derive 옵션을 쓰면, 점 $(a, f(a))$에서 임의의 기울기를 가지는 선을 그릴 수 있다. 접선에 수직인 선을 그리려면 Derive의 값을 $-1/f'(x)$로 두면 된다. 그림 8에는 $x = a$에서의 접선과 접선에 수직하는 선, 그리고 같은 점에서 기울기 -1인 직선이 그려져 있다.[12]

11. \multido 매크로에 대한 자세한 설명은 multido 패키지 매뉴얼[9]를 참조하라.
12. \psplotTangent의 옵션 Derive와 pstricks-add 패키지에서 제공하는 함수 Derive는 다르다. 함수 Derive(n,f(x))는 n-차 도함수를 나타내며 다른 함수와 마찬가지로 \psplot으로 플롯한다. pstricks-add

```
\begin{pspicture}(0,0)(5,4)
  \psaxes[ticks=none]{->}(0,0)(5,4)
  \def\Fxi{2/x}
  \psplot[algebraic,linewidth=1.5pt,%
          linecolor=red]{.5}{5}{\Fxi}
  \multido{\nx=1+1}{3}{%
    \psplotTangent[algebraic,linecolor=blue]%
                  {\nx}{1.5}{\Fxi}}
\end{pspicture}
```

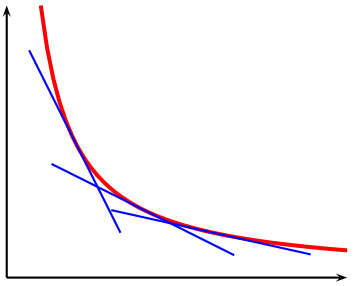


그림 7. 무차별 곡선과 기울기: \psplotTangent

```
\begin{pspicture}(0,-.5)(5,3)
  \psaxes[ticks=none]{->}(0,0)(5,3)
  \def\Fxi{2/x}
  \psplot[algebraic,linewidth=1.5pt,%
    linecolor=red]{.67}{5}{\Fxi}
  \psplotTangent[algebraic,%
    linecolor=blue]{2}{1.5}{\Fxi}
  \psplotTangent[algebraic,arrows=<-,%
    Derive=x^2/2]{2}{1}{\Fxi}
  \psplotTangent[algebraic,arrows=<->,%
    Derive=-1]{2}{1}{\Fxi}
  \psline[showpoints=true](2,0)(2,1)(0,1)
  \uput[d](2,0){$a$}
  \uput[l](0,1){$f(a)$}
\end{pspicture}
```

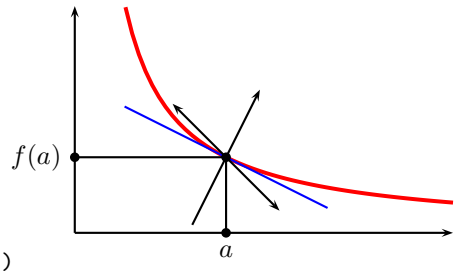


그림 8. Derive 옵션: \psplotTangent

IfTE 함수

IfTE 함수는 조건에 따라 함수를 플롯하기 위해 사용된다. IfTE 함수의 구문은 다음과 같다.

```
IfTE(<If>,<True>,<Else>)
```

IfTE 함수는 If 에서 제시된 조건이 만족되면 True 부분과 같은 값을 갖고, 그렇지 않으면 Else 부분과 같은 값을 가지는 함수이다. 예를 들어, $x > 0$ 이면 $f(x)$ 그리고 그렇지 않으면 $g(x)$와 같은 값을 가지는 함수 IfTE는 다음과 같이 플롯한다.

```
\psplot{xStart}{xEnd}{IfTE(x>0,f(x),g(x))}
```

그림 9는 IfTE함수를 이용하여 플롯하는 예를 보여주고 있다.

IfTE 함수는 중첩해서 사용할 수 있다. 예를 들어, $h(x) < f(x)$ 이면 $h(x)$를 취하고, 그렇지 않으면 다시 $f(x)$와 $g(x)$를 비교하여 조건에 따라 $f(x)$와 $g(x)$를 차례로 취하는 함수 IfTE는 다음과 같은 방법으로 입력한다.

매뉴얼[6] 참조.

```
\begin{pspicture}[showgrid=false](-.5,-.5)(5,3.5)
  \psaxes{->}(0,0)(-.5,-.5)(5,3.5)
  \def\Fai{-x+3}      \uput[45](1,2){Fa}
  \def\Fbi{-1/2*x+2} \uput[45](3,.5){Fb}
  \psplot[algebraic,linestyle=dotted,dotsep=1pt]
         {0}{3}{\Fai}
  \psplot[algebraic,linestyle=dotted,dotsep=1pt]
         {0}{4}{\Fbi}
  \psset{plotpoints=121,linecolor=blue,%
         linewidth=1.5pt}
  \psplot[algebraic]{0}{4}%
                    {IfTE(\Fai>\Fbi,\Fai,\Fbi)}
\end{pspicture}
```

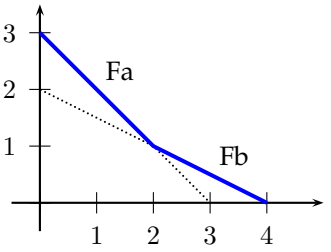


그림 9. IfTE 함수로 그리기

```
IfTE(h(x)<f(x),h(x),IfTE(f(x)<g(x),f(x),g(x)))
```

경제학 문서를 작성할 때 IfTE 함수를 응용할 수 있는 예는 많이 있다. 포락선(envelope curve)이 그러한 예 중의 하나이다. 그림 10은 단기비용함수의 포락선인 장기비용함수(long-run cost curve)를 플롯하는 예이다.

```
\begin{pspicture}(-.5,-.5)(9,6)
  \psaxes[ticks=none]{->}(0,0)(-.5,-.5)(9,6)
  \psset{xunit=3,plotpoints=1000,algebraic}
  \def\Fai{2*x^3+.3} \uput[u](1.3,5){$SC_1$}
  \def\Fbi{.4*x^3+1} \uput[u](2,5){$SC_2$}
  \def\Fci{.2*x^3+2} \uput[u](2.7,5){$SC_3$}
  \psclip{\psframe[linestyle=none](0,0)(8,5)}
    \psplot[linewidth=.4pt]{0}{2}{\Fai}
    \psplot[linewidth=.4pt]{0}{4}{\Fbi}
    \psplot[linewidth=.4pt]{0}{4}{\Fci}
    \psset{linecolor=blue,linewidth=1.5pt}
    \psplot{0}{3}{IfTE(\Fai<\Fbi,\Fai,%
                  IfTE(\Fbi<\Fci,\Fbi,\Fci))}
    \rput(2,1){장기비용함수($LC$)}
  \endpsclip
\end{pspicture}
```

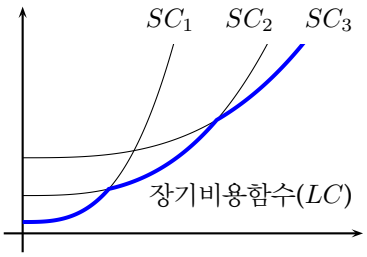


그림 10. Envelope Curve: IfTE 함수

4.2 pst-func 패키지

\psPolynomial 매크로

\psPolynomial 매크로를 이용하면 올림차순으로 정리된 다항함수

$$f(x) = a_0 + a_1x + a_2x^2 + \cdots + a_nx^n$$

의 그래프를 편리하게 플롯할 수 있다. \psPolynomial 매크로를 이용하여 위 다항함수를 플롯하려면 다음과 같이 다항함수의 계수를 옵션 coeff 의 값으로 설정하면 된다.

```
\psPolynomial[coeff=<a0 a1 ... an>]{xStart}{xEnd}
```

\psPolynomial 매크로에서 옵션 coeff 와 함께 옵션 Derivaion=<n> 을 쓰면, n-차 도함수를 플롯할 수 있다.

예를 들어 보자. 식 (1)을 전개하여 정리하면 식 (2)를 얻는다.

$$f(x) = 1 - \frac{1}{2}x - x^2 + \frac{1}{2}x^3 \tag{2}$$

그림 11에는 식 (2)가 실선으로, 그 도함수 $f'(x) = -\frac{1}{2} - 2x + \frac{3}{2}x^2$는 점선으로 그려져 있다.

```
\usepackage{pst-func} % preamble
\begin{pspicture}(-2,-1)(3,2)
  \psgrid[style=gridstyle]
  \psaxes[ticks=none]{->}(0,0)(-2,-1)(3,2)
  \psPolynomial[coeff=1 -0.5 -1 0.5]{-1.25}{2.6}
  \psPolynomial[coeff=1 -0.5 -1 0.5,Derivation=1,
                linestyle=dashed]{-0.67}{2}
\end{pspicture}
```

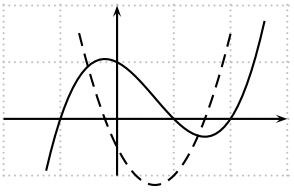


그림 11. \psPolynomial 매크로와 Derivation 옵션

그림 12는 \psPolynomial 매크로와 그 옵션들을 가지고 다항함수의 극대점과 극소점을 표현하는 예를 보이고 있다. \psPolynomial 매크로의 다양한 옵션들에 대해서는 pst-func 매뉴얼[7, p. 3]을 참고할 수 있다.

```
\begin{pspicture}(-2,-1)(3,2)
  \psaxes[Dy=2,labelFontSize=\tiny](0,0)(-2,-1)(3,2)
  \psset{plotpoints=1000,linewidth=1.2pt}
  \psPolynomial[coeff=1 -0.5 -1 0.5,linecolor=blue]
               {-1.25}{2.6}
  \psset{zeroLineStyle=dotted,dotsep=1pt,zeroLineTo=0}
  \psset{coeff=1 -0.5 -1 0.5,markZeros}
  \psPolynomial[Derivation=1,linestyle=none]{-0.67}{2}
\end{pspicture}
```

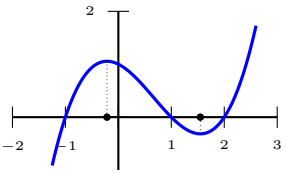


그림 12. 극대점과 극소점: \psPolynomial

정규분포함수

평균이 μ이고 표준편차가 σ인 정규분포함수(normal distribution function)

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

를 간편하게 플롯할 수 있는 매크로로 pst-func 패키지에서 제공하는 \psGauss가 있다. \psGauss 매크로는 평균(mue)과 표준편차(sigma)의 값을 옵션에서 설정하여 플롯한다.

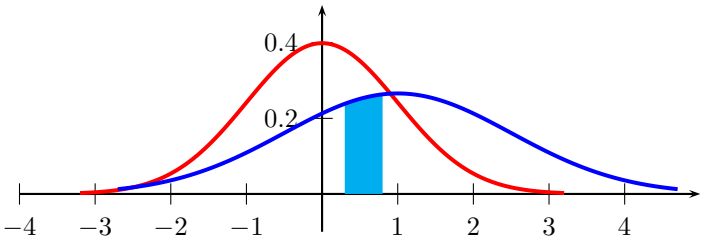
```
\psGauss[mue=<mean>,sigma=<std>]{xStart}{xEnd}
```

기본값은 mue=0이고 sigma=0.5이다.

정규분포함수를 간편하게 플롯할 수 있게 하는 또 하나는 pst-math 패키지에서 제공하는 새로운 포스트스크립트 함수 GAUSS이다. GAUSS 함수는 변수 x, 그리고 평균과 표준편차—이렇게 세 개의 인자를 차례로 RPN 방식으로 취한다. GAUSS 함수를 이용하여 정규분포함수를 플롯하는 구문은 다음과 같다.

```
\psplot{xStart}{xEnd}{x <mue> <sigma> GAUSS}
```

그림 13은 GAUSS 함수와 \psGauss 명령으로 정규분포함수를 플롯하는 예를 보여주고 있다. 그림 13에서 사용된 \pscustom 매크로는 두 함수의 그래프가 둘러싸고 있는 영역을 색칠하거나 하는 등에 응용될 수 있는 유용한 매크로이다.[13]




```
1 \begin{pspicture}(-4,-.1)(5,.5)
2   \psaxes[Dy=0.2]{->}(0,0)(-4,-.1)(5,.5)
3   \psset{plotpoints=1000,linewidth=1.5pt}
4   \psplot[linecolor=red]{-3.2}{3.2}{x 0 1 GAUSS} % pst-math
5   \pscustom[fillstyle=solid,fillcolor=cyan,linestyle=none]{%
6     \psGauss[mue=1,sigma=1.5,linecolor=green]{.3}{.8} % pst-func
7     \psline[liftpen=1](.8,0)(.3,0)}
8   \psGauss[mue=1,sigma=1.5,linecolor=blue]{-2.7}{4.7} % pst-func
9 \end{pspicture}
```

그림 13. 정규분포

\psPrintValue 매크로

\psPrintValue 매크로는 포스트스크립트의 연산결과를 숫자로 표현해준다. 구문은 다음과 같다.

```
\psPrintValue[<options>]{postscript code}
```

13. \pscustom과 옵션 liftpen에 대한 자세한 설명은 pstricks 매뉴얼[8, pp. 32–36]을 참고하라.

예로, 다음의 계산을 고려하자.

$$\sqrt{2} + \ln 3 = 1.41421 \; + 1.09861 \; = 2.51283 \tag{3}$$

식 (3)의 숫자는 다음과 같이 코딩하여 얻은 것이다. `\psPrintValue`는 포스트스크립트에게 계산하도록 하여 그 결과를 가져오므로, TEX의 입장에서는 0-차원 박스에 지나지 않는다. 따라서, 계산 결과를 표시해주기 위해서는 `\makebox` 명령 등을 이용하여 숫자가 들어갈 자리를 확보해 주어야 할 필요가 있다.

```
\sqrt{2}+\ln 3 = \makebox[3.4em][l]{\psPrintValue{2 sqrt}}
               + \makebox[3.4em][l]{\psPrintValue{3 ln}}
               = \makebox[3.4em][l]{\psPrintValue{2 sqrt 3 ln add}}
```

`\psPrintValue`는 포스트스크립트 방식으로 정의된 함수 f의 함수값 $f(x)$를 표시하는 데 응용될 수 있다는 점에서 함수의 플로팅에 있어서 긴요한 매크로이다. 108 쪽의 그림 20에서 `\psPrintValue` 매크로가 사용된 예를 볼 수 있다.

5 효용극대화 모형

5.1 Cobb-Douglas 효용함수와 효용극대

두 재화 x_1, x_2를 소비하는 어떤 소비자의 선호관계를 다음의 콥더글러스(Cobb-Douglas) 효용함수로 나타낼 수 있다고 하자.

$$u(x_1, x_2) = x_1^a \, x_2^{1-a} \qquad (0 < a < 1)$$

두 재화의 가격이 각각 p_1과 p_2이고 이 소비자의 소득이 m일 때 이 소비자의 예산제약은 예산선으로 규정된다. 예산선은 식 (4)로 표현된다.

$$p_1 x_1 + p_2 x_2 = m \tag{4}$$

주어진 가격과 소득 (p_2, p_2, m) 하에서 효용을 극대화하는 소비점 (x_1, x_2)을 선택하는 문제, 즉 효용극대화 모형을 쓰면 다음의 식 (5)와 같다.

$$\begin{aligned} &\max_{x_1, x_2} \; u(x_1, x_2) = x_1^a \, x_2^{1-a} \\ &\text{s. t.} \;\; p_1 x_1 + p_2 x_2 = m \end{aligned} \tag{5}$$

이 문제의 해 $x^* = (x_1^*, x_2^*)$는, 효용함수의 파라미터 a가 주어졌을 때, 예산제약의 파라미터 (p_1, p_2, m)에 의존하는 함수로 표현된다. 실제로 해를 구하면 식 (6)과 같다.

$$x_1^* = x_1(p_1, p_2, m) = a \, \frac{m}{p_1}, \qquad x_2^* = (1 - a) \, \frac{m}{p_2} \tag{6}$$

$x_i^* = x_i(p_1, p_2, m)$를 이 소비자의 i재에 대한 수요함수라 한다.

극대화된 효용수준은 식 (7)과 같이 쓸 수 있다.

$$v(p_1, p_2, m) \equiv u(x_1^*, x_2^*) = \left(\frac{am}{p_1}\right)^a \left(\frac{(1-a)m}{p_2}\right)^{1-a} \tag{7}$$

극대화된 효용수준을 나타내는 함수 v를 간접효용함수(indirect utility function)라 한다.

이제, 효용극대화 모형을 그림으로 표현하고자 한다. 먼저, 무차별곡선(indifference curve) 즉 효용함수의 등고선(level curve)을 플롯할 수 있으면, 경제학에 익숙한 사람들에게는 직관적으로 이해하기 쉽다는 이점이 있다. 함수값 k를 나타내는 효용함수의 등고선은

$$u(x_1, x_2) = x_1^a \, x_2^{1-a} = k \tag{8}$$

이며, 함수값 k를 나타내는 제약함수 g의 등고선은 다음과 같다.

$$g(x_1, x_2) \equiv p_1 x_1 + p_2 x_2 = k \tag{9}$$

`pst-func` 패키지가 제공하는 `\psplotImp` 매크로는 zero-level을 나타내는 등고선을 그려준다.[14] 그림 14는 $a = .5$이고 $(p_1, p_2, m) = (3, 4, 12)$일 때 두 등고선 (8)과 (9)를 `\psplotImp` 매크로를 이용하여 플롯한 것이다.

```
\begin{pspicture}(5,4)
  \psgrid[style=gridstyle]
  \psset{algebraic=true}
  \multido{\Iu=1+1}{4}{%
    \psplotImp(0,0)(5,4){sqrt(x*y)-sqrt(\Iu)}}
  \psplotImp(0,0)(5,4){3*x+4*y-12}
  \psdots(2,1.5)
  \psaxes[ticks=none]{->}(5,4)
\end{pspicture}
```

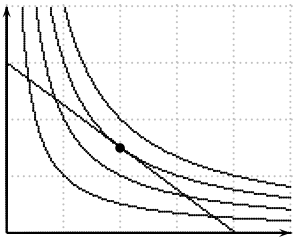


그림 14. 효용극대화: `\psplotImp`

그러나, 현재까지는, `\psplotImp` 매크로로 그린 그림에서 만족할만한 해상도를 기대하기는 어렵다.

이제 `\psplot` 매크로에서 포스트스크립트 코드로 함수를 정의하여 플롯하는 방법을 살펴보기로 하자. 역시 $a = .5$이고 $(p_1, p_2, m) = (3, 4, 12)$라 하자.

먼저, 식 (4)의 예산선을 다음과 같이 다시 쓸 수 있다.

$$x_2 = -\frac{p_1}{p_2} x_1 + \frac{m}{p_2} = -\frac{3}{4} x_1 + \frac{12}{4} \tag{10}$$

$a = .5$일 때 효용수준 $u = 1$을 나타내는 등고선의 식을 x_1에 의존하는 함수로 바꾸어 쓰면 다음과 같다.

$$x_2 = \left[1/x^a\right]^{\frac{1}{1-a}} \tag{11}$$

$a = .5$이고 가격과 소득이 (p_1, p_2, m)일 때, 식 (10)의 예산선과 식 (11)의 무차별곡선을 다음과 같이 코딩할 수 있다.

14. `pst-func` 패키지 매뉴얼[7, p. 27] 참조.

```
\psplot{0}{4}{3 4 div x mul neg 12 4 div add}     % budget line
\psplot{.1}{5}{1 x .5 exp div 1 1 .5 sub div exp} % indifference curve(IC)
```

이 코드를 반영하여 그림 14와 같은 내용의 그림을 플롯하면 그림 15와 같다. 그래프의 해상도는 plotpoints 옵션의 설정을 조정하면 된다. plotpoints 옵션의 기본값은 50이다.

```
\begin{pspicture*}(5,4)
  \psgrid[style=gridstyle]
  \psset{plotpoints=2000}
  \multido{\Iu=1+1}{4}{%
    \psplot{.1}{5}{\Iu\space
      sqrt x .5 exp div 1 1 .5 sub div exp}}
  \psplot{0}{4}{3 4 div x mul neg 12 4 div add }
  \psdots(2,1.5)
  \psaxes[ticks=none]{->}(5,4)
\end{pspicture*}
```

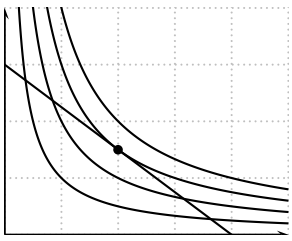


그림 15. 효용극대화: \psplot

5.2 새로운 매크로 \uMaxCD

a가 주어졌을 때, 효용극대점 x^*는 파라미터 (p_1, p_2, m)에 의존한다. 따라서 가격과 소득에 따라서 효용극대상황을 그림으로 나타낼 수 있다면 편리할 것이다. 이를 위해 다음과 같은 실험적인 매크로 \uMaxCD를 새롭게 제시하고자 한다.

새로운 매크로 \uMaxCD의 정의에서는 (!ps) 형태의 특수좌표가 사용되고 있으므로, 이를 실행하기 위해서는 pstricks 패키지를 로드하고 \SpecialCoor를 선언하면 된다. 하지만, \psPrintValue 등 다른 플롯기능과 함께 사용하기 위해서 pst-func 패키지를 로드하는 것이 권장된다. pst-func 패키지를 로드하는 경우에는 \SpecialCoor 선언을 생략해도 된다.

```
%% 새로운 매크로 \uMaxCD 정의
\newcommand\uMaxCD[4]{%
  % 파라미터
  \def\px{#1 } % x재 가격
  \def\py{#2 } % y재 가격
  \def\mm{#3 } % 소득(m)
  \def\aa{#4 } % Cobb-Douglas 함수의 지수
  % 수요(demands)
  \def\solx{\aa \mm \px div mul }       % x재의 수요량
  \def\soly{1 \aa sub \mm \py div mul } % y재의 수요량
  % 함수의 정의(RPN)
  \def\Bgt{/x exch def \px \py div x mul neg \mm \py div add }    % 예산선
  \def\uVal{\solx \aa exp \soly 1 \aa sub exp mul }              % 간접효용수준
  \def\Util{/x exch def \uVal x \aa exp div 1 1 \aa sub div exp } % 무차별곡선
  % 함수의 플로팅
  \psplot[plotpoints=2000]{.1}{100}{x \Util} % 무차별곡선
```

```
    \psplot{0}{100}{x \Bgt}                    % 예산선
    % 주요 좌표 저장하기
    \pnode(!\solx \soly){EQM}
    \pnode(EQM|0){xSTAR}
    \pnode(0|EQM){ySTAR}
    \pnode(!\mm \px div 0){xALL}
    \pnode(!0 \mm \py div){yALL}
    % 균형점과 점선의 표시
    \psdots(EQM)
    \psline[linestyle=dotted,dotsep=1.5pt](EQM|0)(EQM)(0|EQM)
}
```

새로운 매크로 \uMaxCD를 사용하는 구문은 다음과 같다.

```
\uMaxCD{p_1}{p_2}{m}{a}
```

\uMaxCD 매크로가 가지는 네 개의 인자 중 처음 세 인자는 파라미터 (p_1, p_2, m)의 값이고 마지막 인자는 a의 값이다. 그림 16은 새로운 명령 \uMaxCD를 이용하여 효용극대화 모형을 간편하게 플롯하는 예이다. pspicture 박스의 크기는 시행착오를 거쳐 적절히 조절하면 된다. pspicture 박스의 크기를 조절하는 문제에 대해서는 아래에서 다시 논의될 것이다. 그림 16에서는 pspicture 박스 밖의 그림을 잘라내기 위해 *버전을 사용하였다. 따라서 좌표축을 표시하기 위해서는 \psaxes 매크로를 pspicture 환경 밖에 쓰는 것이 좋다. 효용극대점과 이를 연결한 점선을 표시하지 않으려면 \uMaxCD를 정의하는 코드의 마지막 두 줄을 주석처리하면 된다.

```
\begin{pspicture*}(4,3)
  \uMaxCD{4}{6}{12}{.3}
\end{pspicture*}%
\psaxes[origin={-4,0},ticks=none]{->}(4,3)
```

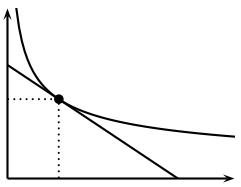


그림 16. 효용극대화: \uMaxCD와 pspicture* 환경

\uMaxCD를 정의하는 코드에는 유용한 몇 가지 좌표를 지정해두었는데, 표 3에 나타난 바와 같다. \uMaxCD가 처리되는 과정에서 이 좌표값들이 기억된다. 따라서 \uMaxCD가 선언된 직후에 이 좌표를 이용하여 그래프상에 필요한 정보를 표시할 수 있다. 예를 들어, 균형점의 45° 방향으로 x^*를 label하기 위해서는 다음과 같이 입력한다.

```
\uMaxCD{4}{6}{12}{.3}
\uput[45](EQM){$x^*$}
```

다른 파라미터 값이 적용된 \uMaxCD가 처리되면 이 좌표들이 기억하는 값도 달라진다는 점에 유의할 필요가 있다.

표 3. \uMaxCD에서 정의된 좌표

좌표	좌표의 내용
(EQM)	균형점의 좌표 (x_1^*, x_2^*)
(xSTAR)	균형점의 x-축 좌표 $(x_1^*, 0)$
(ySTAR)	균형점의 y-축 좌표 $(0, x_2^*)$
(xALL)	예산선의 x-절편의 좌표 $(m/p_1, 0)$
(yALL)	예산선의 y-절편의 좌표 $(0, m/p_2)$

pspicture* 환경으로 박스 밖의 그림을 잘라내는 경우에는 축과 축에 관한 정보 등을 표시하는 데에 불편한 점이 있다. 그림 16의 코드에서와 같이 pspicture* 환경을 사용하는 경우에는, 예를 들어, 절편을 나타내기 위한 코드를 환경내에 입력하면 잘라져 그림에 나타나지 않는다. 반면에 \psclip 매크로를 이용하면 보다 편리하게 축의 정보를 표현할 수 있다.[15] 그림 17은 그림 16과 본질적으로 같은 그림을 \psclip 매크로를 이용하여 그린 것이다. 그림 17의 소스에는 절편의 정보 등을 나타내기 위한 코드를 pspicture 환경내에 표시하고 있음에 유의하자.

```
\begin{pspicture}(0,-.5)(4,3)
  \psclip{\psframe[linestyle=none](0,0)(4,3)}
    \uMaxCD{4}{6}{12}{.3}
  \endpsclip
  \uput[l](yALL){$m/p_2$}
  \uput[45](EQM){$x^*$}
  \uput[d](xSTAR){$x_1^*$}
  \psaxes[ticks=none]{->}(4,3)
\end{pspicture}%
```

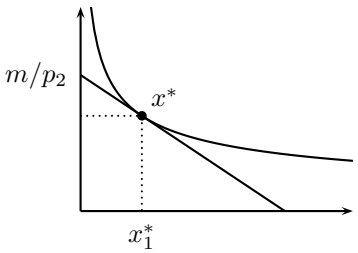


그림 17. 효용극대화: \uMaxCD & \psclip

비교정태분석

\uMaxCD 매크로를 이용하면 비교정태분석도 간편하게 플롯할 수 있다. 먼저 가격변화의 경우를 보자. 그림 18은 1재의 가격이 차례로 $p_1 = 9$, $p_1' = 6$, $p_1'' = 4$로 변할 때의 소비점의 변화를 나타낸 것이다. 단순히 첫 번째 인자만 바꾸며 \uMaxCD 매크로를 반복하여 사용하면 가격변화에 의한 비교정태분석 결과를 플롯할 수 있다.

다음으로 소득변화의 경우를 보자. 가격변화의 경우와 마찬가지로, \uMaxCD 매크로를 반복적으로 사용함으로써 소득변화에 따르는 비교정태분석의 결과를 간단하게 플롯할 수 있다. 그림 19는 소득수준이 차례로 $m = 10$, $m' = 12$, $m'' = 14$로 증가하는 경우의 소비점을 나타내고 있다.

15. \psclip 매크로에 관한 자세한 설명은 pstricks 매뉴얼[8, pp. 54–55] 참조.

```
\begin{pspicture}(0,-.5)(4,3)
  \psclip{\psframe[linestyle=none](0,0)(4,3)}
    \uMaxCD{9}{6}{12}{.5}
    \uMaxCD{6}{6}{12}{.5}
    \uMaxCD{4}{6}{12}{.5}
  \endpsclip
  \psaxes[ticks=none]{->}(4,3)
\end{pspicture}%
```

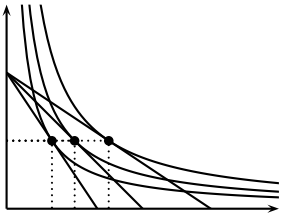


그림 18. 비교정태분석: 가격변화

```
\begin{pspicture}(0,-.5)(4,3)
  \psclip{\psframe[linestyle=none](0,0)(4,3)}
    \multido{\nm=10+2}{3}{%
      \uMaxCD{4}{6}{\nm}{.6}}
  \endpsclip
  \psaxes[ticks=none]{->}(4,3)
\end{pspicture}%
```

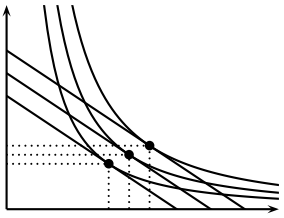


그림 19. 비교정태분석: 소득변화

5.3 \uMaxCD 매크로의 응용

picture 크기의 조절을 위한 템플릿

\uMaxCD 매크로는 충분히 넓은 범위에서 함수를 플롯하도록 정의되어 있으므로, 적절한 크기의 박스로 잘라주어야 한다. 예산선의 절편을 고려하면, 자를 박스의 크기는 가로와 세로가 각각 m/p_1과 m/p_2보다 커야 한다.

\psclip 매크로를 이용하여 \uMaxCD로 플롯한 그림을 사각형 모양으로 잘라내는 간단한 템플릿은 다음과 같다. <가로>는 m/p_1보다 크고 <세로>는 m/p_2보다 크도록 설정해야 한다.

```
%% 템플릿: \uMaxCD 그림 재단하기
\psclip{\psframe[linestyle=none](<가로>,<세로>)}
  \uMaxCD{p_1}{p_2}{m}{a}
\endpsclip
```

위와 같은 \psclip 매크로로 만들어진 박스는 TEX의 입장에서는 0-차원 박스이므로 이를 pspicture 환경 안에 넣어야 한다. 다음은 이를 위한 템플릿이다. 잘라진 박스가 pspicture 박스 바깥으로 나가지 않아야 하므로, xMax $\geq$ 가로이고 yMax $\geq$ 세로이어야 한다.

```
%% 템플릿: \uMaxCD 그림 재단하여 pspicture 박스에 넣기
\begin{pspicture}[showgrid=true](xMin,yMin)(xMax,yMax)
  \psclip{\psframe[linestyle=none](<가로>,<세로>)}
    \uMaxCD{p_1}{p_2}{m}{a}
  \endpsclip
\end{pspicture}\psset{unit=1cm}
```

그림 20은 이러한 모든 아이디어가 복합적으로 구현된 예이다. 그림 20의 소스를 템플릿으로 이용하는 경우, \xx, \yy, \Uscale의 세 가지 매크로의 값을 조절하여 쓰면 된다. \xx와 \yy는 pspicture 박스의 xMax와 yMax값을 대체하는 값이다. 이 값을 \psclip 박스의 <가로>와 <세로>의 크기로 배정하였지만, 그림 21의 경우처럼 실제 사용에 있어서는 <가로>와 <세로>의 값을 적절히 바꾸어 쓸 수 있다. \Uscale은 예산선의 절편이 박스 바깥으로 나가지 않게 하기 위하여 그래픽 객체의 크기를 조절하기 위한 것으로, pspicture 환경 내에서만 유효하다. 당연히 그림 전체의 크기를 조절하기 위해서는 환경 밖에서 \psset 매크로로 길이단위를 조절하면 된다. 한편, 그림 20에서는 \psPrintValue 매크로를 이용하여 균형소비점의 값과 극대화된 효용수준의 값을 표시하고 있다.

오퍼곡선

\uMaxCD 매크로는 콥더글러스 효용함수를 기반으로 하고 있다. 효용함수가 콥더글러스 함수이면 가격소비곡선(price consumption curve) 또는 오퍼곡선(offer curve)이 수평선이거나 수직선이다. 또한 소득소비곡선(income consumption curve) 또는 소득확장경로(income expansion path)는 원점을 지나는 직선이다.

그럼에도 불구하고, \uMaxCD 매크로를 이용하여 일반적인 형태의 오퍼곡선이나 소득확장경로를 플롯할 수 있다. 핵심 아이디어는 a를 조절하는 것이다. a를 변화시키고 균형점 EQM을 \pscurve로 연결하면 임의의 오퍼곡선을 그릴 수 있다. 그림 21에 이 아이디어가 구현되어 있다.

6 결론

PSTricks를 이용하면 포스트스크립트 언어로 함수를 정의하여 정확성이 담보된 함수를 플롯할 수 있다. 그러나 포스트스크립트에서 사용하는 RPN(Reverse Polish Notation) 방식이 익숙하지 않아서 불편한 점이 있다. 반면, 일반적으로 많이 사용하는 대수적 표기로 함수를 정의하여 플롯하면 매우 편리한데, 최근의 PSTricks는 이러한 기능을 포함하는 패키지를 제공하고 있다. 함수를 일반적인 대수적 표기 방식으로 손쉽게 플롯하는 가장 효율적인 방법은 pstricks-add 패키지를 로드하여 algebraic 옵션과 함께 \psplot 매크로를 이용하는 것이다. pstricks-add 패키지는 pst-plot 패키지를 부르고 이는 다시 pstricks 패키지를 부른다.

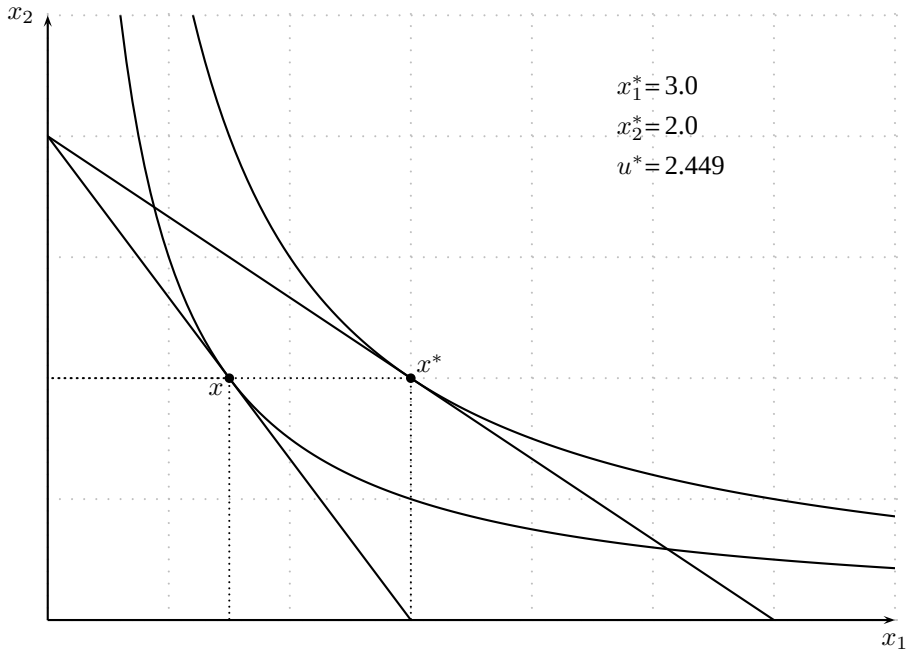
pstricks-add 패키지는 \psplot 매크로 이외에도 함수의 플로팅과 관련한 다양한 기능을 제공하고 있다. pst-func 패키지도 특정한 함수를 간편하게 플롯하는 매크로를 제공한다. pst-func을 로드하면 pstricks-add를 자동검색하여 로드해주므로, 프리앰블(preamble)에서 pst-func를 로드하는 것으로 지금까지 개발된 거의 모든 함수의 플로팅 기능을 구현할 수 있다. 다만, GAUSS 함수를 포함하여 역시 복잡한 함수를 간편하게 플롯할 수 있는 기능을 제공하는 pst-math 패키지를 보완적으로 사용하려면 이를 따로 로드하여야 한다. PSTricks 관련 패키지를 로드함에 있어서 pstricks-add 패키지를 맨

마지막에 로드하여야 하며, 따라서 `pst-func`을 대신 로드할 때에도 맨 마지막에 로드하여야 한다.

한편, 포스트스크립트 언어를 이용하면 매우 유연한 코딩을 할 수 있어서 다양한 응용을 할 수 있다. 이 글에서는 효용극대화 모형을 직관적으로 간편하게 플롯할 수 있는 새로운 매크로 `\uMaxCD`가 제시되어 있다. 이 새로운 매크로는 콥더글러스 효용함수를 기반으로 정의되었지만 일반적인 효용극대모형의 결과들을 플롯하는 데에도 손쉽게 응용될 수 있다.

참고 문헌

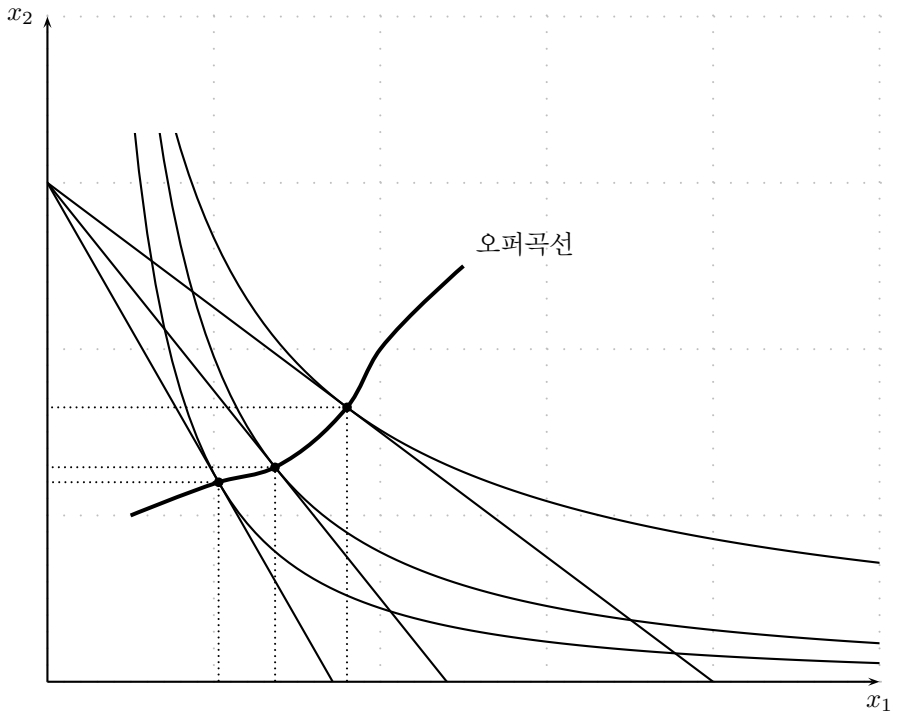
1. 김우영·조인성, 고용보험 실업급여사업의 적정변동요율에 관한 연구, 『한국경제연구』 15권 (2005), 35–86.
2. Adobe Systems Inc., *PostScript Language Tutorial and Cookbook*, Addison-Wesley, 1985.
3. Bill Casselman, *Mathematical Illustrations: A Manual of Geometry and PostScript*, Cambridge University Press, 2005.
4. Jean-Côme Charpentier and Chirstophe Jorssen, *'infix-RPN' — 'pst-infixplot'*, ver. 0.11, May 16, 2005. `CTAN:graphics/pstricks/contrib/pst-infixplot/pst-infixplot.pdf`
5. Christophe Jorssen, *'pst-math': A PSTricks package for enhancing mathematical operators in PSTricks*, ver. 0.2, July 14, 2004.
   `CTAN:graphics/pstricks/contrib/pst-math/pst-math.pdf`
6. Dominique Rodriguez and Herbert Voß, *pstricks-add: addtionals Macros for pstricks*, ver. 2.84, March 13, 2007.
   `CTAN:graphics/pstricks/contrib/pstricks-add/pstricks-add-doc.pdf`
7. Herbert Voß, *pst-func: plotting special mathematical functions*, ver. 0.46, September 16, 2006.
   `CTAN:graphics/pstricks/contrib/pst-func/pst-func-doc.pdf`
8. Timothy Van Zandt, *PSTricks: PostScript macros for Generic TeX. User's Guide*, ver. 97, July 25, 2003. `CTAN:graphics/pstricks/base/doc/pstricks-doc.pdf`
9. _____, *Documentation for multido.tex: A loop macro for Generic TEX*, May 18, 2004. `CTAN:macros/generic/multido/multido.pdf`




```
%% determining the box size
\def\xx{7 } %% box width
\def\yy{5 } %% box height

\begin{pspicture}(0,0)(\xx,\yy)\psgrid[style=gridstyle]
  \def\Uscale{1 }%% local scaling
  \psclip{\psframe[linestyle=none](0,0)(\xx,\yy)}
    \psset{unit=\Uscale}
    \uMaxCD{4}{3}{12}{.5} %% utility maximization parameters
    \uput{3pt}[225](EQM){$x$}
    \uMaxCD{2}{3}{12}{.5} %% utility maximization parameters
    \uput{3pt}[45](EQM){$x^*$}
    %% printing values
    \pnode(!.7 \xx mul \Uscale div .9 \yy mul \Uscale div){NUM}
    \uput{0pt}[d](NUM){$x_1^*=\,\psPrintValue{\solx}}
    \uput{15pt}[d](NUM){$x_2^*=\,\psPrintValue{\soly}}
    \uput{30pt}[d](NUM){$u^*=\,\psPrintValue{\uVal 1000 mul round 1000 div}}
  \endpsclip
  \psaxes[ticks=none]{->}(\xx,\yy)
  \uput[d](\xx,0){$x_1$}
  \uput[l](0,\yy){$x_2$}
\end{pspicture}%
```

그림 20. pspicture 상자의 크기 조절




```
%% determining the box size
\def\xx{5 } %% box width
\def\yy{4 } %% box height

\begin{pspicture}(0,0)(\xx,\yy)\psgrid[style=gridstyle]
  \def\Uscale{1 }%% local scaling
  \psclip{\psframe[linestyle=none](0,0)(\xx,3.3)}
    \psset{unit=\Uscale}
    \uMaxCD{7}{4}{12}{.6}
    \pnode(EQM){E1}
    \uMaxCD{5}{4}{12}{.57}
    \pnode(EQM){E2}
    \uMaxCD{3}{4}{12}{.45}
    \pnode(EQM){E3}
    \pscurve[plotpoints=1000,plotstyle=curve,linewidth=1.5pt]%
            (.5,1)(E1)(E2)(E3)(2,2)(2.5,2.5)
  \endpsclip
  \uput[45](2.5,2.5){오퍼곡선}
  \psaxes[ticks=none]{->}(\xx,\yy)
  \uput[d](\xx,0){$x_1$}
  \uput[l](0,\yy){$x_2$}
\end{pspicture}
```

그림 21. 오퍼곡선