8 การแก้ปัญหาจุดตรึงและปัญหาไดนามิกโปรแกรมมิงในแบบจำลองทางเศรษฐศาสตร์ด้วยการแทนซ้ำ

ในบทนี้เราจะศึกษาการแก้ปัญหาจุดตรึง (fixed point) และปัญหาไดนามิกโปรแกรมมิง (dynamic programming) ด้วยการแทนซ้ำ ส่วนที่ 8.1 แสดงวิธีการแทนซ้ำในการแก้ปัญหาจุดตรึง ส่วนที่ 8.2 ประยุกต์ใช้การแทนซ้ำเพื่อแก้ปัญหาสมการฟังก์ชัน (functional equation) ส่วนที่ 8.3 ประยุกต์ใช้การแทนซ้ำในการหาดุลยภาพของแนช (Nash equilibrium) ส่วนที่ 8.4 -8.6 ศึกษาปัญหาไดนามิดโปรแกรมมิงและการใช้การแทนซ้ำในการปัญหาไดนามิกโปรแกรมมิงและสมการเบลแมน (Bellman equation) ส่วนที่ 8.7 และ 8.8 ศึกษาการแก้สมการเบลแมนในแบบจำลองสโตแคสติก

8.1 จุดตรึงและวิธีการแทนซ้ำ

แนวความคิดสำคัญหลายอย่างด้านเศรษฐศาสตร์ เช่น ดุลยภาพของตลาด (market equilibrium) ดุลยภาพของแนช (Nash equilibrium) สภาวะคงตัว และคำตอบสมการเบลแมน ล้วนเป็นจุดตรึงชนิดหนึ่งทั้งสิ้น

x^* เป็นจุดตรึงของฟังก์ชัน f ก็ต่อเมื่อ $f(x^*) = x^*$ วิธีการหาจุดตรึงของฟังก์ชันที่ง่ายที่สุด คือ การแทนค่าฟังก์ชันซ้ำๆ ด้วยวิธีการดังนี้

1. ให้ $n = 1$ เลือกค่าเริ่มต้น x_1
2. ให้ $x_{n+1} = f(x_n)$
3. ตรวจสอบว่าระยะห่างระหว่าง x_{n+1} กับ $f(x_{n+1})$ มีค่าน้อยพอยอมรับได้หรือไม่ หากมีค่าน้อยยอมรับได้ x_{n+1} คือจุดตรึงของฟังก์ชัน f หากยังยอมรับไม่ได้ ให้ $n = n + 1$ และกลับไปทำขั้นตอนที่ 2

ตารางที่ 8.1 แสดงการหาค่าจุดตรึงของ $f(x) = \frac{x}{2} - 1$ กำหนดให้ $x_1 = 0$ เราจะได้ผลดังตารางด้านล่างนี้ จากตารางที่ 8.1จะเห็นว่า ค่าของ x_n จะเข้าใกล้ -2 ซึ่งเป็นจุดตรึงของฟังก์ชันนี้เนื่องจาก $f(-2) = -2$

วิธีการแทนค่าฟังก์ชันซ้ำๆเป็นวิธีที่ง่ายและตรงไปตรงมาที่สุด แต่วิธีนี้มีข้อจำกัดอยู่สองประการคือ หากฟังก์ชันที่เราสนใจมีจุดตรึงมากกว่าหนึ่งจุด วิธีการนี้จะให้ค่าจุดตรึงไม่ครบทุกค่า และค่าที่ได้นี้อาจขึ้นอยู่กับค่าตั้งต้น x_1 ที่เราเลือก ข้อจำกัดอีกประหนึ่ง คือค่าของอนุกรม x_n อาจจะมีลักษณะลู่ออกและไม่เข้าสู่จุดตรึงของฟังก์ชัน f

ตารางที่ 8.1: การหาจุดตรึงของ $f(x)=\frac{x}{2}-1$ ด้วยการแทนซ้ำ

n	x_n	$f(x_n)$
1	0	-1
2	-1	-1.5
3	-1.5	-1.75
4	-1.75	-1.875
5	-1.875	-1.9375
6	-1.9375	-1.96875
7	-1.96875	-1.98438
8	-1.98438	-1.99219
9	-1.99219	-1.99609
10	-1.99609	-1.99805

ตารางที่ 8.2 แสดงตัวอย่างการหาจุดตรึงของ $f(x)=2x$ ด้วยวิธีการแทนซ้ำโดยใช้ค่า $x_1=1$ จะเห็นได้ว่าอนุกรม x_n มีลักษณะลู่ออก และไม่ได้ลู่เข้าสู่จุดตรึงของฟังก์ชันซึ่งมีค่าเท่ากับ 0

ตารางที่ 8.2: การหาจุดตรึงของ $f(x)=2x$ ด้วยการแทนซ้ำ

n	x_n	$f(x_n)$
1	1	2
2	2	4
3	4	8
4	8	16
5	16	32
6	32	64

ตารางที่ 8.3 แสดงวิธีการหาจุดตรึงของ $f(x)=\sqrt{x}$ โดยใช้ค่า $x_1=2$ จะเห็นค่าจุดตรึงที่เราได้จะมีค่าประมาณ 1 แต่ฟังก์ชันที่จริงแล้วฟังก์ชัน f มีจุดตรึงสองจุดคือ 1 และ 0

ตารางที่ 8.3: การหาจุดตรึงของ $f(x)=\sqrt{x}$ โดยการแทนซ้ำ

n	x_n	$f(x_n)$
1	2	1.414214
2	1.414214	1.189207
3	1.189207	1.090508
4	1.090508	1.044274
5	1.044274	1.021897
6	1.021897	1.010889

อย่างไรก็ตามสำหรับบางปัญหาอย่าง เช่น ปัญหาไดนามิกโปรแกรมมิง นักเศรษฐศาสตร์มักกำหนดข้อสมมติของแบบจำลองทางทฤษฎีบางอย่างเพื่อให้จุดตรึงของฟังก์ชันมีจุดเดียวและอนุกรม x_n ที่ได้จากการแทนซ้ำมีลักษณะลู่เข้าสู่จุดตรึงเสมอ

โปรแกรมด้านล่างแสดงวิธีการหาจุดของฟังก์ชัน $f(x) = \frac{x}{2} - 1$ ด้วยการแทนซ้ำ

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

```
function r=f(x)
 r = x/2 - 1;
endfunction
x = 2;
while norm(x-f(x))  > 0.00001
 x = f(x);
end
disp(x);
```

ผลที่ได้จากโปรแกรม

- 1.999984

8.2 สมการฟังก์ชัน

ปัญหาทางเศรษฐศาสตร์ที่สำคัญหลายปัญหาเป็นปัญหาสมการฟังก์ชัน สมการฟังก์ชัน คือ สมการที่มีคำตอบเป็นฟังก์ชัน ตัวอย่างปัญหาสมการฟังก์ชันได้แก่

1. จงหาฟังก์ชัน f ซึ่ง $f'(x) = 1$ และ $f'(0) = 1$

   คำตอบคือ $f(x) = x + 1$
2. จงหาฟังก์ชัน f ซึ่ง $f(x + y) = f(x).f(y)$ และ $f(1) = 2$

   คำตอบคือ $f(x) = 2^x$
3. จงหาฟังก์ชัน $V(x) = \frac{V(1-x)}{2} + \frac{x}{2}$ โดยที่ x อยู่ในช่วง [0, 1]

   คำตอบคือ $V(x) = \frac{x+1}{3}$

วิธีเชิงสัญลักษณ์ที่ใช้ในการแก้สมการฟังก์ชันคือวิธีการเดาและจับคู่สัมประสิทธิ์ (guess and coefficient matching) ตัวอย่างเช่น สำหรับปัญหาที่ 3 เราสามารถแก้ปัญหานี้โดยเริ่มเดาว่า V

เป็นฟังก์ชันเส้นตรง $V(x) = a + bx$ โดย a และ b เป็นค่าคงที่ เราสามารถหาค่าของ a และ b ได้ดังนี้ จาก

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

แทนค่า $V(x) = a + bx$ ในสมการจะได้

$$a + bx = \frac{a + b(1-x)}{2} + \frac{x}{2}$$

คูณสองทั้งสมการและจัดรูปจะได้

$$2a + 2bx = (a+b) + (1-b)x$$

จากการจับคู่สัมประสิทธิ์เราจะได้

$$2a = a + b$$

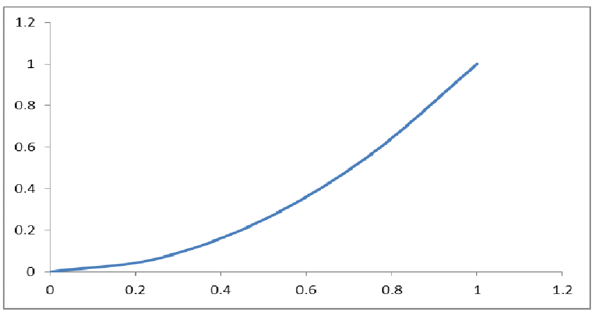
$$2b = 1 - b$$

แก้สมการสองสมการนี้จะได้ $b = a = \frac{1}{3}$ ดังนั้นจะได้ว่า

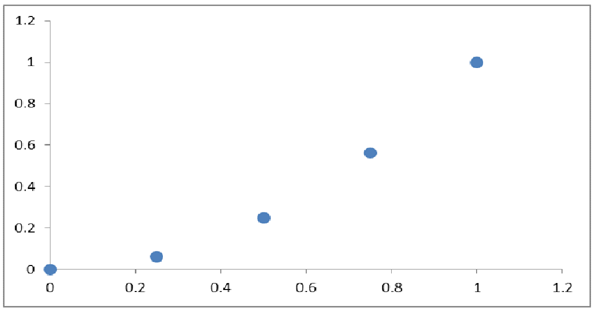
$$V(x) = \frac{x+1}{3}$$

อย่างไรก็ตามในกรณีทั่วไปเราจำเป็นต้องหาคำตอบของสมการฟังก์ชันด้วยด้วยวิธีการเชิงตัวเลข สิ่งที่เราต้องทำเป็นเพื่อแก้สมการฟังก์ชันด้วยวิธีการเชิงตัวเลขคือ การประมาณฟังก์ชัน (function approximation) ซึ่งโดยทั่วไปประกอบด้วยจุดเป็นจำนวนอนันต์ ให้เป็นเวกเตอร์ของจุดที่มีจำนวนจำกัด

ตัวอย่างเราสามารถประมาณฟังก์ชัน $f(x) = x^2$ สำหรับ x ในช่วง [0, 1] สามารถประมาณได้ด้วยจุด 5 จุดดังต่อไปนี้ (0,0) (1/4, 1/16) (1/2,1/4) (3/4,9/16) (1,1) ดังแสดงในรูปที่ 8.1 และ 8.2



รูปที่ 8.1: $f(x) = x^2$



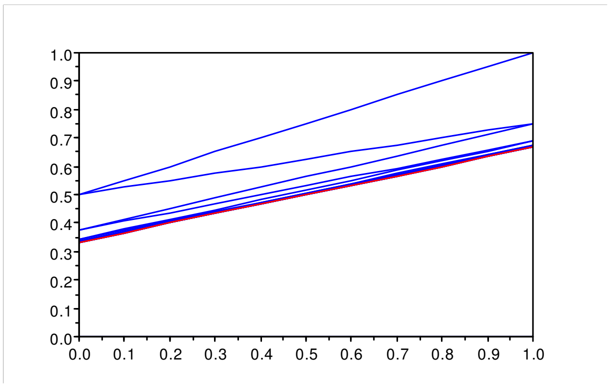
รูปที่ 8.2: การประมาณ $f(x)=x^2$ ด้วยจุด 5 จุด

หลังจากทำการประมาณฟังก์ชันด้วยเวกเตอร์ของจุดแล้ว เราสามารถหาคำตอบของสมการฟังก์ชันด้วยการแทนซ้ำ เหมือนกับการหาคำตอบของระบบสมการปกติ ตัวอย่างโปรแกรมด้านล่างแสดงการหาคำตอบของสมการ $V(x)=\frac{V(1-x)}{2}+\frac{x}{2}$ โดยการประมาณฟังก์ชัน V ด้วยจุด 11 จุด และใช้การแทนซ้ำ

การแก้สมการ $V(x)=V(1-x)+\frac{x}{2}$ ด้วยการแทนซ้ำ

```
clear; n=11;
V  = zeros(1,n);
x = 0:0.1:1;
clf;
Vp = ones(1,n);
V1x = Vp; //V1x is V(1-x)
plot(x,V);  Plot the initial function V
while norm(V-Vp) > 0.0001
  V = Vp;
  V1x = V(n:-1:1); //Find V(1-x)
  Vp = V1x/2 + x/2;
  plot(x,Vp);
end
plot(x,Vp,'red');
```

ผลที่ได้จากโปรแกรม



กราฟที่ได้จากโปรแกรมแสดงฟังก์ชันจากการแทนซ้ำแต่ละครั้ง กราฟเส้นล่างสุดกราฟที่ได้จากการแทนซ้ำครั้งสุดท้ายและคือคำตอบของสมการฟังก์ชัน จะเห็นได้ว่ากราฟเส้นล่างสุดใกล้เคียงกับฟังก์ชัน $V(x)=\dfrac{x+1}{3}$ ซึ่งเป็นคำตอบของสมการฟังก์ชัน

8.3 การหาดุลยภาพของแนช

ในทฤษฎีเกมดุลยภาพของแนช (Nash equilibrium) เป็นจุดตรึงของฟังก์ชันตอบสนองดีที่สุด (best response function) ดังนั้นเราจึงสามารถประยุกต์วิธีการแทนซ้ำเพื่อหาดุลยภาพของแนชได้ ส่วนที่ 8.3.1 แสดงการหาดุลยภาพของแนชในแบบจำลองคูโนด้วยการแทนซ้ำ ส่วนถัดไปแสดงการหาดุลยภาพของแนชในแบบจำลองคูโนที่มีความไม่สมบูรณ์ของข้อมูลด้วยการแทนซ้ำ

8.3.1 การหาดุลยภาพของแนชในแบบจำลองคูโน

พิจารณาแบบจำลองคูโนที่อุปสงค์ของตลาดคือ $p=10-q_1-q_2$ โดย p และ q_i คือราคาและปริมาณในตลาดของบริษัท $i=1,2$ บริษัท 1 มีต้นทุนการผลิตต่อหน่วย $c_1=0$ บริษัท 2 มีต้นทุนการผลิตต่อหน่วย $c_2=1$ บริษัททั้งสองแข่งขันกันโดยการกำหนดปริมาณการผลิตเพื่อให้ได้กำไรสูงสุด เราจะทำการหาปริมาณการผลิตของทั้งสองบริษัทในดุลยภาพแนชโดยใช้วิธีแทนซ้ำ

เนื่องจากดุลยภาพของแนชนั้นคือจุดตรึงของฟังก์ชันตอบสนองที่ดีที่สุด ดังนั้นเราจะเริ่มจากการหาฟังก์ชันตอบสนองที่ดีที่สุดก่อน จากฟังก์ชันกำไรของบริษัทที่ 1 (π_1)

$$\pi_1 = pq_1 - c_1q_1 = (10-q_1-q_2)q_1$$

ดิฟเฟอเรนชิทเอทเพื่อหาฟังก์ชันตอบสนองจะได้

$$\frac{d\pi_1}{dq_1}=0 \Rightarrow 10-2q_1-q_2=0 \Rightarrow q_1=\frac{10-q_2}{2}$$

ในทำนองเดียวกันสำหรับบริษัทที่ 2

$$\pi_2 = pq_2 - c_2 q_2 = (10 - q_1 - q_2)q_2 - q_2$$

ดิฟเฟอเรนชิทเอทเพื่อหาฟังก์ชันตอบสนองจะได้

$$\frac{d\pi_2}{dq_2} = 0 \Rightarrow q_2 = \frac{9 - q_1}{2}$$

ดังนั้นฟังก์ตอบสนองของเกมนี้คือ

$$B(q_1, q_2) = (\frac{10 - q_2}{2}, \frac{9 - q_1}{2})$$

ให้ (q_1^*, q_2^*) เป็นดุลยภาพแนชของเกม (q_1^*, q_2^*) จะมีคุณสมบัติเป็นจุดตรึงของฟังก์ชันตอบสนองหรือเขียนในรูปสมการได้เป็น

$$B(q_1^*, q_2^*) = (\frac{10 - q_2^*}{2}, \frac{9 - q_1^*}{2}) = (q_1^*, q_2^*)$$

เราสามารถหาดุลยภาพของแนชได้ด้วยวิธีการแทนฟังก์ชันซ้ำตามโปรแกรมข้างล่างนี้

การหาดุลยภาพของแนชในแบบจำลองคูโน

```
function r = B(q) //Best Response Function
 r(1) = (10 - q(2))/2;
 r(2) = (9 - q(1))/2;
endfunction
q = [0 ; 0];
qp = [1; 1];
while norm(q - qp) > 0.000001
q = qp;
qp = B(q);
end
disp(qp(1));
disp(qp(2));
```

ผลที่ได้จากโปรแกรม

```
3.67
2.67
```

ผลที่ได้จากโปรแกรมแสดงว่าในดุลยภาพของแนชบริษัทที่หนึ่งและบริษัทที่สองจะผลิตสินค้าจำนวน 3.67 และ 2.67 หน่วยตามลำดับ

8.3.2 การหาดุลยภาพแนชในแบบจำลองคูโนที่มีความไม่สมบูรณ์ของข้อมูล

ส่วนนี้เราพิจารณาแบบจำลองคูโนภายใต้สภาพแวดล้อมที่มีความไม่สมบูรณ์ของข้อมูล ด้านต้นทุนของบริษัท กำหนดให้อุปสงค์ของตลาดคือ $p = 10 - q_1 - q_2$ โดย p และ q_i คือราคาและปริมาณในตลาดของบริษัท i บริษัท 1 มีต้นทุนการผลิตต่อหน่วยคือ c_1 บริษัท 2 มีต้นทุนการผลิตต่อหน่วยคือ c_2 แต่ละบริษัทรู้ต้นทุนการผลิตของตัวเอง แต่ไม่รู้ต้นทุนการผลิตของอีกบริษัทหนึ่ง แต่ละบริษัทมีความเชื่อว่าต้นทุนของอีกบริษัทมีการกระจายตัวแบบยูนิฟอร์มในช่วง [0, 1]

เราจะหาดุลยภาพของแนชในเกมนี้ด้วยวิธีการแทนซ้ำ ดุลยภาพของแนชนี้จะประกอบด้วยกลยุทธ์ของบริษัทที่ 1 และ 2 ซึ่งเป็น ฟังก์ชันของราคาต่อหน่วยของแต่ละบริษัท $(q_1(c_1), q_2(c_2))$ เราเริ่มโดยการหาฟังก์ชันตอบสนองที่ดีที่สุดของบริษัทที่หนึ่ง จากฟังก์ชันกำไรคาดหวังของบริษัทที่ 1:

$$E[\pi_1] = E[pq_1 - c_1 q_1] = E[(10 - q_1 - q_2)q_1 - c_1 q_1]$$

จากสมการนี้จะได้เงื่อนไขอนุพันธ์อันดับหนึ่งในหารทำให้กำไรคาดหวังสูงสุดคือ

$$\frac{dE[\pi_1]}{dq_1} = 10 - 2q_1 - E[q_2] - c_1 = 0$$

แก้สมการจะได้ฟังก์ชันตอบสนองที่ดีที่สุดของบริษัทที่ 1 คือ

$$q_1(c_1) = \frac{10 - c_1 - E[q_2]}{2}$$

ในทำนองเดียวกันจะได้ฟังก์ชันตอบสนองที่ดีที่สุดของบริษัทที่ 2 คือ

$$q_2(c_2) = \frac{10 - c_2 - E[q_1]}{2}$$

ดังนั้นฟังก์ชันตอบสนองของเกมนี้คือ

$$B(q_1(c_1), q_2(c_2)) = \left(\frac{10 - c_1 - E[q_2]}{2}, \frac{10 - c_2 - E[q_1]}{2}\right)$$

จะเห็นได้ว่า $B(.)$ เป็นฟังก์ชันที่รับอินพุท (input) เป็นฟังก์ชันและให้ค่าเป็นฟังก์ชัน ดุลยภาพของแนชคือฟังก์ชัน $q_1^*()$ และ $q_2^*()$ ซึ่งเป็นจุดตรึงของฟังก์ชัน $B(.)$ และ

$$B(q_1^*(c_1), q_2^*(c_2)) = \left(\frac{10 - c_1 - E[q_2^*]}{2}, \frac{10 - c_2 - E[q_1^*]}{2}\right) = (q_1^*(c_1), q_2^*(c_2))$$

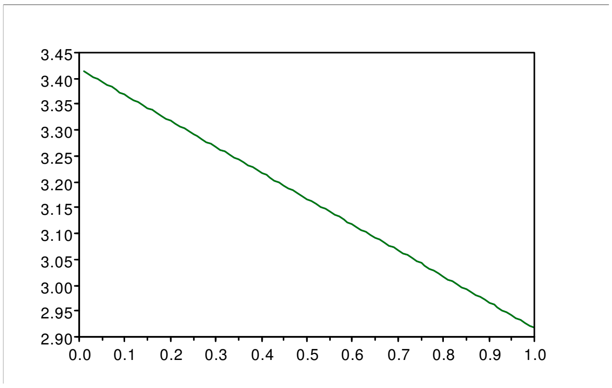
ดุลยภาพของแนชซึ่งคือจุดตรึงของฟังก์ชัน B สามารถหาได้ด้วยโปรแกรมด้านล่างนี้

การหาดุลยภาพของแนชในแบบจำลองคูโน

```
clear;
function r = B(q) //Best Response Function
 eq1 = mean(q(1,:)); //find expection of q1
 eq2 = mean(q(2,:));
 r(1,:) = (10 - gammas - eq2)/2;
 r(2,:) = (10 - gammas - eq1)/2;
endfunction

n = 11;
gammas = 0: 0.1: 1;
q(1,:) = zeros(1,n);
q(2,:) = zeros(1,n);
qp(1,:) = ones(1,n);
qp(2,:) = ones(1,n);
while norm(q - qp) > 0.000001
 q = qp;
 qp = B(q);
 end
plot(gammas, qp);
```

ผลที่ได้จากโปรแกรม



8.4 การแก้ปัญหาไดนามิกโปรแกรมมิงด้วยวิธีการเชิงตัวเลข

ในส่วนนี้เราจะศึกษาการแก้ปัญหาไดนามิกโปรแกรมมิงด้วยวิธีการเชิงตัวเลข ส่วนที่ 8.4.1 แนะนำการเปลี่ยนปัญหาผลบวกอนันต์ให้อยู่ในรูปปัญหาไดนามิกโปรแกรมมิง

8.4.1 การแก้ปัญหาผลบวกอนันต์ด้วยวิธีการไดนามิกโปรแกรมมิง

ปัญหาทางเศรษฐศาสตร์ที่สำคัญหลายปัญหาอยู่ในรูปการหาผลบวกอนันต์ (infinite sum) ในทางคณิตศาสตร์แล้วการหาค่าการหาผลบวกอนันต์นี้สามารถหาได้สองวิธีคือ 1. วิธีการตรง (direct method) และ 2. วิธีการไดนามิกโปรแกรมมิง

ตัวอย่างเช่นหากต้องการหาค่าของ

$$V = 1+\beta+\beta^2+\beta^3+\ldots;\ |\beta|<1$$

เราสามารถหาค่า V ด้วยวิธีการตรงดังนี้คือ

$$V = 1+\beta+\beta^2+\beta^3+\ldots$$
$$V = \lim_{T->\infty}(1+\beta+\beta^2+\beta^3+\ldots+\beta^T)$$
$$V = \lim_{T->\infty}\left(\frac{1-\beta^{T+1}}{1-\beta}\right) = \frac{1-0}{1-\beta} = \frac{1}{1-\beta}$$

นอกจากนี้เราสามารถหาค่า V ด้วยวิธีการแบบไดนามิกโปรแกรมมิงดังนี้

$$V = 1+\beta+\beta^2+\beta^3+\ldots$$
$$V = 1+\beta(1+1+\beta+\beta^2+\beta^3+\ldots)$$
$$V = 1+\beta V \rightarrow V = \frac{1}{1-\beta}$$

หลักการของไดนามิกโปรแกรมมิงคือการเปลี่ยนรูปแบบของปัญหาใหญ่ที่ต้องการแก้ให้อยู่ในรูปปัญหาย่อย (sub problem) ที่มีลักษณะคล้ายกัน ในตัวอย่างด้านบนจะเห็นได้ว่าวิธีการแบบไดนามิกโปรแกรมมิงง่ายและใช้ความรู้ทางด้านคณิตศาสตร์น้อยกว่าการแก้ผลบวกอนันต์โดยตรง

8.5 การแก้สมการของเบลแมนด้วยวิธีการเชิงสัญลักษณ์

ปัญหาในแบบจำลองด้านเศรษฐศาสตร์มหภาคเชิงพลวัตสมัยใหม่ส่วนใหญ่สามารถเขียนในอยู่ในรูปแบบปัญหาไดนามิกโปรแกรมมิงในรูปของสมการเบลแมน (Bellman equation) ได้ ตัวอย่างเช่น ปัญหาของผู้บริโภคในแบบจำลองด้านล่าง

$$\max_{c_t,k_{t+1}} \sum_{t=1}^{\infty} \beta^t u(c_t)$$

ภายใต้เงื่อนไข $k_{t+1} = k_t^{\alpha} - c_t$, $u(c_t) = \ln(c_t)$, k_1 ถูกกำหนดมาแล้ว

สามารถเปลี่ยนเป็นปัญหาไดนามิกโปรแกรมในรูปสมการของเบลแมนดังนี้ กำหนดให้

$$V(k_1) = \max_{c_t, k_{t+1}} \sum_{t=1}^{\infty} \beta^t u(c_t)$$

ภายใต้เงื่อนไข $k_{t+1} = k_t^{\alpha} - c_t$, $u(c_t) = \ln(c_t)$, k_1 ถูกกำหนดมาแล้ว

เราสามารถแสดงฟังก์ชัน V ดังนี้

$$V(k) = \max_{k'}(u(c) + \beta V(k'))$$

ภายใต้เงื่อนไข $k' = k^{\alpha} - c$, $u(c) = \ln(c)$

โดยที่ $k \equiv k_t$, $k' \equiv k_{t+1}$ และ $c \equiv c_t$ และ $V(k)$ คือ อรรถประโยชน์ตลอดชีวิตที่ได้จากสินค้าทุน k เมื่อแทน c ในรูปของ k เราสามารถเขียนสมการเบลแมนได้ดังนี้

$$V(k) = \max_{k'}(u(k^{\alpha} - k')) + \beta V(k'))$$

จะเห็นได้ว่าสมการเบลแมนก็คือสมการฟังก์ชันสมการหนึ่ง ดังนั้นเราสามารถหาฟังก์ชัน V ได้ด้วยการแทนซ้ำในรูปแบบสัญลักษณ์ดังนี้

1. ให้ $n = 1$ เลือกค่าฟังก์ชันตั้งต้น V_1
2. หาค่า $V_{n+1}(k) = \max_{k'}(u(k^{\alpha} - k')) + \beta V_n(k'))$ ซ้ำไปเรื่อยๆ

ตัวอย่างเช่นหากเราให้ $V_1(k) = 0$ จะได้ว่า

$$V_2(k) = \max_{k'}(\ln(k^{\alpha} - k')) + \beta V_1(k'))$$

$$V_2(k) = \max_{k'}(\ln(k^{\alpha} - k')) = \ln(k^{\alpha} - 0) = \alpha \ln k$$

จะได้ว่า $V_2(k) = \alpha \ln k$ ดังนั้นเราจะแทน

$$V_3(k) = \max_{k'}(\ln(k^{\alpha} - k')) + \beta V_2(k'))$$

$$V_3(k) = \max_{k'}(\ln(k^{\alpha} - k')) + \alpha \ln k')$$

ดิฟเฟอเรนชิเอท $\ln(k^{\alpha} - k') + \alpha \ln k'$ จะได้

$$0 = \frac{1}{k^{\alpha} - k'} + \frac{\alpha}{k'}$$

$$k' = \frac{\beta\alpha}{1 + \beta\alpha} k^{\alpha}$$

แทนค่า $k' = \dfrac{\beta\alpha}{1 + \beta\alpha} k^{\alpha}$ ใน $V_3(k) = \ln(k^{\alpha} - k')) + \beta V_2(k')$ จะได้

$$V_2(k) = \alpha(1 + \beta\alpha)\ln k + \beta\alpha \ln(\beta\alpha) - (1 + \beta\alpha)\ln(1 + \beta\alpha)$$

เมื่อทำแทนซ้ำไปเรื่อยๆ เราสามารถพิสูจน์ได้ว่า

$$V(k)=\lim_{n->\infty}V_n(k)=\frac{\alpha}{1-\beta\alpha}\ln k+\frac{1}{1-\beta}\ln(1-\beta\alpha)+\frac{1}{1-\beta}\frac{\beta\alpha}{1-\beta\alpha}\ln(\beta\alpha)\,.$$

จากตัวอย่างนี้จะเห็นได้ว่าในกรณีทั่วไปการแก้ปัญหาด้วยการแทนซ้ำเชิงลักษณ์มีความยุ่งยากมาก

8.6 การแก้สมการของเบลแมนด้วยวิธีการเชิงตัวเลข

นอกจากการแทนซ้ำด้วยวิธีเชิงสัญลักษณ์แล้ว เราสามารถหาคำตอบของสมการเบลแมนด้วยการแทนซ้ำเชิงตัวเลข ตัวอย่างเช่นหากเราต้องการหาค่าของฟังก์ชัน V ซึ่งเป็นคำตอบของสมการ

$$V(k)=\max_{k'}(u(k^{\alpha}-k'))+\beta V(k'))$$

ได้ด้วยขั้นตอดังนี้

ขั้นตอนที่ 1 คือ การกำหนดโดเมนของฟังก์ชัน$V(.)$ รอบๆค่าสภาวะคงตัว k^* เช่น เราอาจระบุให้โดเมนของฟังก์ชัน $V(.)$ คือช่วงตั้งแต่ $0.9k^*$ ถึง $1.1k^*$ (หากช่วงโดเมนที่กำหนดมีความกว้างมากหรือน้อย เกินไป เราสามารถปรับเปลี่ยนโดเมนได้ภายหลัง)

ขั้นตอนที่ 2 คือ การสร้างเวกเตอร์กริดที่ประกอบด้วยกริดเป็นจำนวน g จุดเพื่อประมาณโดเมนของฟังก์ชัน $V(.)$ ตัวอย่างเช่น หากเราให้โดเมนของฟังก์ชัน $V(.)$ คือช่วง $0.9k^*$ ถึง $1.1k^*$ เราสามารถเวกเตอร์กริด $\mathbf{k}$ จำนวน 11 จุดที่ประกอบด้วยค่า $0.9k^*, 0.91k^*, \ldots, 1.01k^*$ แทนโดเมนของฟังก์ชัน

ขั้นตอนที่ 3 คือการสร้างแมทริกซ์ระดับการบริโภค $\mathbf{C}$ ซึ่งมีมิติ $g\times g$ โดยที่ค่าของ c_{ij} (สมาชิกในแถวที่ i และหลักที่ j ของแมทริกซ์ $\mathbf{C}$) คือระดับการบริโภคที่ทำให้ประมาณสินค้าใน $k'=k_j$ เมื่อปริมาณสินค้าทุนในวันนี้เท่ากับ $k=k_i$ โดยที่ k_i คือค่าของสมาชิกตัวที่ i ของเวกเตอร์ $\mathbf{k}$ จากเงื่อนไข

$$c=k^{\alpha}-k'$$

จะได้ว่า

$$c_{ij}=k_i^{\alpha}-k_j$$

ขั้นตอนที่ 4 คือ การสร้างเวกเตอร์ศูนย์ $\mathbf{V}_1$ ขนาด $g\times 1$ ให้เป็นค่าเริ่มต้นของฟังก์ชัน V ในการแทนซ้ำ สร้างเวกเตอร์ $\mathbf{V}_{n+1}$ = $\max_{k'}\{u(c(k'))+\beta\mathbf{V}_n(k')\}$ ทำการแทนซ้ำในขั้นตอนที่สามจนการทั่งค่า $\mathbf{V}_{n+1}$ ที่ได้ลู่เข้า และเป็นคำตอบของสมการเบลแมน

ส่วนต่อไปนี้จะแสดงตัวอย่างโดยละเอียดของขั้นตอน 1 – 4 ในสำหรับกรณีที่เราใช้กริดเพียง 2 กริด $(g = 2)$ ให้เวกเตอร์กริดของ $\mathbf{k} = [k_1\ k_2]$ โดยที่ $k_1 = 0.9k^*$ และ $k_2 = 1.1k^*$ จากเวกเตอร์กริดนี้เราจะสามารถสร้างแมทริกซ์การบริโภค

$$\mathbf{C} = \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix}$$

โดยที่ $c_{ij} = k_i^{\alpha} - k_j$

ต่อมาในขั้นตอนที่ 3 เราสร้างแมทริกซ์ $\mathbf{V}_1 = [V_{11}\ V_{12}]'$ ซึ่งแทนของฟังก์ชัน V_1 และทำการหาค่า $\mathbf{V}_2$ จากการแทนค่าในสมการเบลแมนดังนี้ $V_2 = \max_{k'}\{u(c(k')) + \beta V_1(k')\}$ เป็นรูปแบบไซแลบดังนี้

$$V_2 = \begin{bmatrix} V_{21} \\ V_{22} \end{bmatrix} = \max\left(\begin{bmatrix} u(c_{11}) & u(c_{12}) \\ u(c_{21}) & u(c_{22}) \end{bmatrix} + \beta \begin{bmatrix} V_{11} & V_{12} \\ V_{11} & V_{12} \end{bmatrix}, 'c' \right)$$

โดยที่คำสั่ง max(., ‘c’) ของไซแลบจะให้ค่าเวกเตอร์ซึ่งที่มีจากตัวเลขที่มากที่สุดของแต่ละแถวเช่น ค่าของ $\max(\begin{bmatrix} 2 & 1 & 4 \\ 8 & 6 & 0 \end{bmatrix}, 'c')$ คือ $\begin{bmatrix} 4 \\ 8 \end{bmatrix}$ นอกจากนี้หากเราต้องการรู้ตำแหน่งของตัวเลขในแต่ละแถวซึ่งมีค่ามากที่สุดในแถวนั้นเราสามารถทำได้โดยใช้คำสั่งในรูปแบบ

$$[A, B] = \max(\begin{bmatrix} 2 & 1 & 4 \\ 8 & 6 & 0 \end{bmatrix}, 'c')$$

จากคำสั่งนี้เราจะได้ค่า A คือ $\begin{bmatrix} 4 \\ 8 \end{bmatrix}$ และค่าของ B คือ $\begin{bmatrix} 3 \\ 1 \end{bmatrix}$ เราจะเห็นได้ว่า 3 คือตำแหน่งของเลข 4 ในแถวที่หนึ่ง และ 1 คือตำแหน่งของแปดในแถวที่สอง

ตัวอย่างโปรแกรมด้านล่างแสดงการหาค่า V ซึ่งเป็นคำตอบของปัญหาไดนามิกโปรแกรมมิงพร้อมทั้งแสดงกราฟระดับสินค้าทุนกับเวลา

การสมการเบลแมนด้วยวิธีเชิงตัวเลข

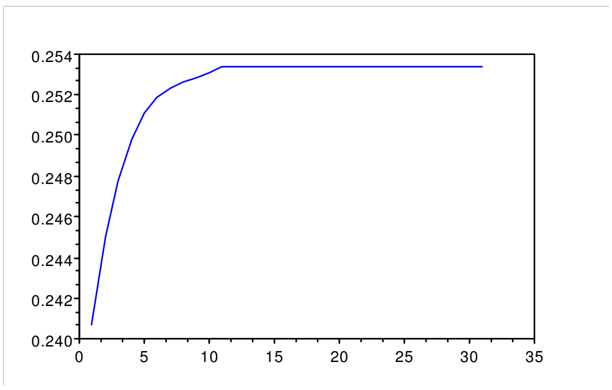
```
clear; alpha = 0.66; d = 1; beta = 0.95;
kstar =  ((1+d*beta-beta)/(alpha*beta))^(1/(-1+alpha));
g = 101;
k = ( 0.9*kstar: 0.1*kstar/(g-1):1.1*kstar)'; //discretize the state variables
c = zeros(g,g);  //generate consumption matrix
for i=1:g;
 for j=1:g;
    c(i,j) = k(i)^alpha - k(j);
```

```
    if c(i,j) > 0 then
      u(i,j) = log(c(i,j));
     else
       u(i,j) = -10000000000;
    end
  end
end

//value function iteration
Tv = ones(g, 1);
v = zeros(g,1);
o = ones(1,g);
while norm(Tv - v) > 0.000001
  v = Tv;
  [Tv kprime_id] = max(u + beta*(v*o)', 'c');
end

//the path of capital accumulation.
kt_id(1) = 1; // start from k0 = k(1)
kt(1) = k(1);  //note that kt(i) is the real value of capital stock at time i.
               //kt_id(i) is the capital stock index at time i.
for t=1 : 30
 kt_id(t+1) = kprime_id(kt_id(t));
 kt(t+1) = k(kt_id(t+1));
end
plot(kt);
```

ผลที่ได้จากโปรแกรม



8.6.1 การแทนซ้ำฟังก์ชันนโยบาย

การหาคำตอบของปัญหาไดนามิกโปรแกรมมิงโดยใช้วิธีการแทนฟังก์ชัน $V(.)$ ซ้ำโดยตรงนั้นมักให้ความเร็วในการลู่เข้า (speed of convergence) ค่อนข้างต่ำ วิธีที่ให้ความเร็วในการลู่เข้าสูงกว่า คือ วิธีการแทนค่าฟังก์ชันนโยบาย (policy function iteration) ซึ่งวิธีการนี้จะแตกต่างกับวิธีการแทนค่าซ้ำของฟังก์ชัน $V(.)$ ในขั้นตอนที่ 3 ดังนี้คือ ในขั้นตอนที่ 3 เราหาฟังก์ชันนโยบาย $k'(k)$ ซึ่งเป็นฟังก์ชันของ k ซึ่งทำให้ $u(c(k')) + \beta V_n(k')$ มีค่าสูงสุด

เมื่อได้ฟังก์ชันนโยบาย $k'(k)$ แล้ว เราจึงหาค่าฟังก์ชัน $V_{n+1}(.)$ ด้วยการแก้สมการ

$$V_{n+1}(k) = u(k'(k)) + \beta V_{n+1}(k'(k))$$
$$V_{n+1}(k) - \beta V_{n+1}(k'(k)) = u(k'(k))$$

เมื่อเขียนสมการในรูปแมทริกซ์จะได้

$$(\mathbf{I} - \beta \mathbf{k}')\mathbf{V_{n+1}} = \mathbf{u}(\mathbf{k}')$$
$$\mathbf{V_{n+1}} = (\mathbf{I} - \beta \mathbf{k}')^{-1}\mathbf{u}(\mathbf{k}')$$

ตัวอย่างการหาค่าฟังชัน $V(.)$ โดยวิธีการแทนค่าฟังก์ชันนโยบายอยู่ด้านล่างนี้ (หมายเหตุเนื่องจากส่วนอื่นๆของโปรแกรมไม่ได้เปลี่ยนแปลงไปจากเดิมจึงไม่ได้ถูกนำมาแสดงในที่นี้) จากการทดสอบวิธีการแทนซ้ำแบบแรกที่ไม่ใช้ฟังก์ชันโยบายและแบบที่ใช้ฟังก์ชันโยบายนี้ เราพบว่าวิธีการแรกจะมีการวนซ้ำถึง 284 ครั้งก่อนได้ค่าของฟังก์ชัน $V(.)$ ส่วนวิธีที่ใช้การแทนฟังก์ชันนโยบายจะใช้การวนซ้ำเพียง 11 ครั้ง และใช้เวลาน้อยกว่าวิธีแรกประมาณ 30 เปอร์เซ็นต์

การแก้สมการเบลแมนด้วยการแทนซ้ำฟังก์ชันนโยบาย

```
clear; alpha = 0.66; d = 1; beta = 0.95;
kstar =  ((1+d*beta-beta)/(alpha*beta))^(1/(-1+alpha));
g = 101;
k = ( 0.9*kstar: 0.1*kstar/(g-1):1.1*kstar)'; //discretize the state variables
c = zeros(g,g);  //generate consumption grid
for i=1:g;
 for j=1:g;
    c(i,j) = k(i)^alpha - k(j);
   if c(i,j) > 0 then
     u(i,j) = log(c(i,j));
    else
      u(i,j) = -10000000000;
   end
 end
end
```

```
Tv = ones(g, 1); v = zeros(g,1);
idg = eye(g,g); o = ones(1,g);
kprime_id = zeros(g,1);
ou = zeros(g,1); //optimal utility
while norm(Tv - v) > 0.000001
  v = Tv;
  [Tv kprime_id] = max(u + beta*(v*o)', 'c');
  op = zeros(1,g); //optimal capital stock for the next period
 for i = 1:g
   op(i,kprime_id(i)) = 1;
   ou(i) = u(i,kprime_id(i));
   end
  Tv = inv(idg - beta*op)*ou;
end
```

8.7 การแก้สมการเบลแมนในแบบจำลองสโตแคสติก

ส่วนนี้จะแสดงการแก้สมการเบลแมนในแบบจำลองแบบสโตแคสติก โดยภาวะสโตแคสติกของแบบจำลองเกิดการความไม่แน่นอนของระดับเทคโนโลยี ส่วนแรกจะศึกษากรณีที่ระดับของเทคโนโลยีเป็นตัวแปรสุ่มแบบไอไอดี (independent and identically distributed) และส่วนที่สองศึกษากรณีที่ระดับเทคโนโลยีเป็นตัวแปรสุ่มแบบมาคอฟ

8.7.1 การแก้สมการของเบลแมนในแบบจำลองที่มีช็อคแบบไอไอดี

พิจารณาแบบจำลองต่อไปนี้

$$\max_{c_t, k_{t+1}} E_1 \sum_{t=1}^{\infty} \beta^t u(c_t)$$

ภายใต้เงื่อนไข $k_{t+1} = a_t k_t^{\alpha} - c_t, \quad u(c_t) \equiv \ln(c_t)$

$$prob(a_t = 0.99) = prob(a_t = 1.01) = 0.5$$

แบบจำลองนี้เป็นแบบจำลองสโตแคสติกเนื่องจำระดับเทคโนโลยี a_t เป็นตัวแปรสุ่มแบบไอไอดี ปัญหานี้สามารถเขียนเปลี่ยนให้อยู่ในรูปสมการของเบลแมนได้ดังนี้

$$V(k,a) = \max_{k'} (u(ak^{\alpha} - k') + \beta(V(k',0.99) + V(k',1.01))/2)$$

ปัญหานี้ซับซ้อนกว่าปัญหาที่ผ่านมาเนื่องจากการที่ระดับเทคโนโลยี a เป็นตัวแปรสุ่มจะทำให้ $V(.)$ เป็นฟังก์ชันของ k และ a ดังนั้นเราต้องประมาณฟังก์ชัน $V(.)$ ด้วยเวกเตอร์สองเวกเตอร์ เวกเตอร์แรกสำหรับกรณีที่ $a = 0.99$ และเวกเตอร์แรกสำหรับกรณีที่ $a = 1.01$

การแก้สมการของเบลแมนในแบบจำลองที่มีช็อคแบบไอไอดี

```
clear; alpha = 0.66; d = 1; a(1) = 0.99;  a(2) = 1.01;  beta = 0.95;
kstar =  ((1+d*beta-beta)/(alpha*beta))^(1/(-1+alpha));
g = 51;
k = ( 0.9*kstar: 0.1*kstar/(g-1):1.1*kstar)'; //discretize the state variables
c = zeros(g,g,2);  //generate consumption matrix
for m=1:2 //m is the extra dimension for a
 for i=1:g;
  for j=1:g;
     u(i,j,m) = -10000000000;
     c(i,j,m) = a(m)*(k(i)^alpha) - k(j);
   if c(i,j,m) > 0 then
     u(i,j,m) = log(c(i,j,m));
   end
 end
 end
end

//value function iteration
Tv(:,1) = ones(g, 1);
Tv(:,2) = ones(g, 1);
v(:,1) = zeros(g,1);
v(:,2) = zeros(g,1);
o = ones(1,g);
while norm(Tv - v) > 0.000001
  v = Tv;
  Ev = 0.5*v(:,1) + 0.5*v(:,2);
  [temp1  temp2]  =  max(u(:,:,1) + beta*(Ev*o)', 'c');
  Tv(:,1) = temp1;
  kprime_id(:,1) = temp2;
  [temp1  temp2]  =  max(u(:,:,2) + beta*(Ev*o)', 'c');
  Tv(:,2) = temp1;
  kprime_id(:,2) = temp2;
end
```

```
//find variance of consumption using Monte-Carlo Simulation
kt_id(1) = (1+g)/2; // start from the steady state
kt(1) = k((1+g)/2);
shock = grand(5000,1,'uin',1,2);
for t=1 : 5000
  kt_id(t+1) = kprime_id( kt_id(t), shock(t));   //
  kt(t+1) = k(kt_id(t+1));
  ct(t) =  a(shock(t))*kt(t)^alpha - kt(t+1);
end
disp(stdev(ct)/mean(ct));
```

ผลที่ได้จากโปรแกรม

0.01990505

8.7.2 การแก้สมการของเบลแมนในแบบจำลองที่มีช็อคแบบมาคอฟ

พิจารณาแบบจำลองต่อไปนี้

$$\max_{c_t, k_{t+1}} E_1 \sum_{t=1}^{\infty} \beta^t u(c_t)$$

ภายใต้เงื่อนไข $k_{t+1} = a_t k_t^{\alpha} - c_t, \;\; u(c_t) \equiv \ln(c_t)$

$a_t \in \{0.99, 1.01\}$ และ ให้ $\theta_1 = 0.99$ และ $\theta_2 = 1.01$ $prob(a_{t+1} = \theta_j \mid a_t = \theta_i) = \pi_{ij}$

$$\mathbf{\Pi} = \begin{bmatrix} \pi_{11} & \pi_{12} \\ \pi_{21} & \pi_{22} \end{bmatrix} = \begin{bmatrix} 0.2 & 0.8 \\ 0.4 & 0.6 \end{bmatrix}$$

ในแบบจำลองนี้ระดับเทคโนโลยีเป็นตัวแปรสุ่มแบบมาคอฟซึ่งการกระจายของ a_{t+1} จะขึ้นอยู่กับค่าของ a_t และแมทริกซ์ทรานสิชัน (transition matrix) $\mathbf{\Pi}$

ปัญหาของผู้บริโภคในแบบจำลองด้านบนจะสามารถเปลี่ยนให้อยู่ในรูปสมการเบลแมนดังนี้

$$V(k,a) = \max_{k'}(u(c) + \beta(\pi_{i1}V(k',0.99) + \pi_{i2}V(k',1.01)) \;\; \text{สำหรับ} \;\; a = \theta_i$$

โปรแกรมด้านล่างแสดงวิธีการเชิงตัวเลขเพื่อแก้ปัญหาสมการฟังก์ชันนี้ และใช้ค่าฟังก์ชันนโยบายที่ได้ในการหาค่าความแปรปรวนของระดับการบริโภค

```
การแก้สมการของเบลแมนในแบบจำลองที่มีช็อคแบบมาคอฟ
clear;  alpha = 0.66; d = 1; beta = 0.95;  a(1) = 0.9;  a(2) = 1.1;
```