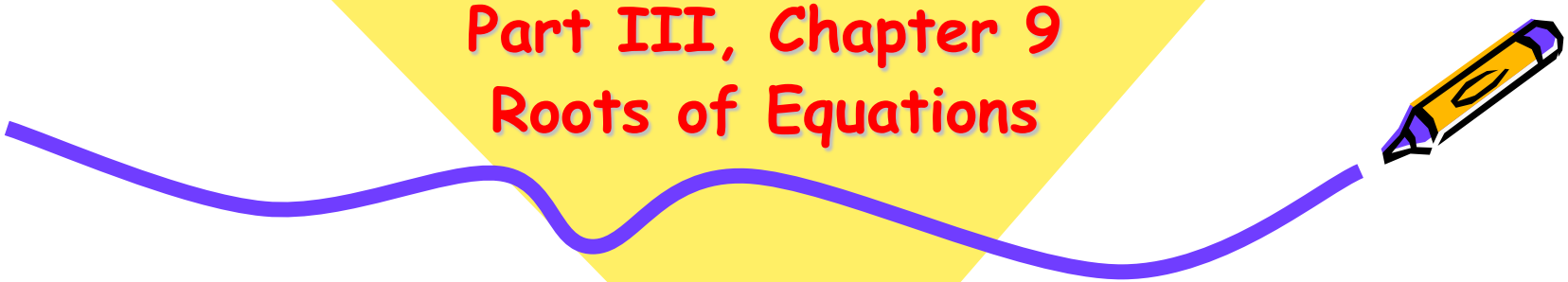
CPE 332
Computer Engineering Mathematics II

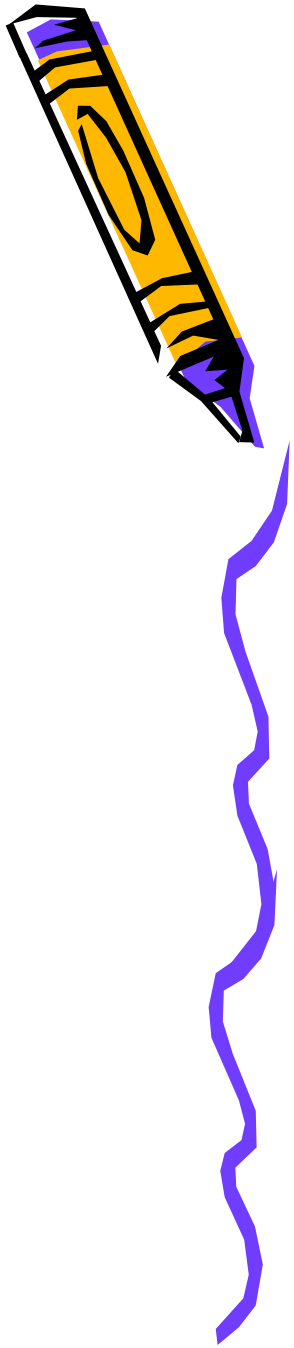
Week 11

Part III, Chapter 9

Roots of Equations



Today Topics



- Roots of Equation
- Bracketing Method
  - Bisection Method
  - False Position Method
- Open Method
  - Simple One-Point Iteration
  - Newton-Ralphson Method
  - Secant Method
  - Multiple Roots Problem
  - Modified Method

Roots of Equations

ในบทนี้เราจะศึกษา Algorithm ที่ใช้ในการหาราก หรือ Root ของ Function โดยที่เราจะกล่าวเฉพาะ Function ที่ประกอบด้วยหนึ่งตัวแปรเท่านั้น ซึ่งเขียนได้ในรูปของ $y = f(x)$ และรากของ Function $f(x)$ สามารถหาได้จากการแก้สมการ $f(x) = 0$

พึงเข้าใจว่ารากของสมการอาจจะมีได้มากกว่าหนึ่งตัว และอาจจะมีรากที่ซ้ำกันได้หรือที่เรียก Multiple Roots อย่างไรก็ตาม ในการศึกษาขั้นต้นนี้ เราจะสมมุติว่า Root ทุกตัวจะมีค่าไม่ซ้ำกัน และเราพอจะรู้ว่ารากที่เราต้องการอยู่ที่บริเวณไหน(การหาค่าประมาณของรากของสมการสามารถทำได้โดยการ Plot Graph และดูที่เมื่อไรเส้น $f(x)$ ตัดกับแกน x หรือมีค่าเท่ากับศูนย์)

Zeroes of Functions

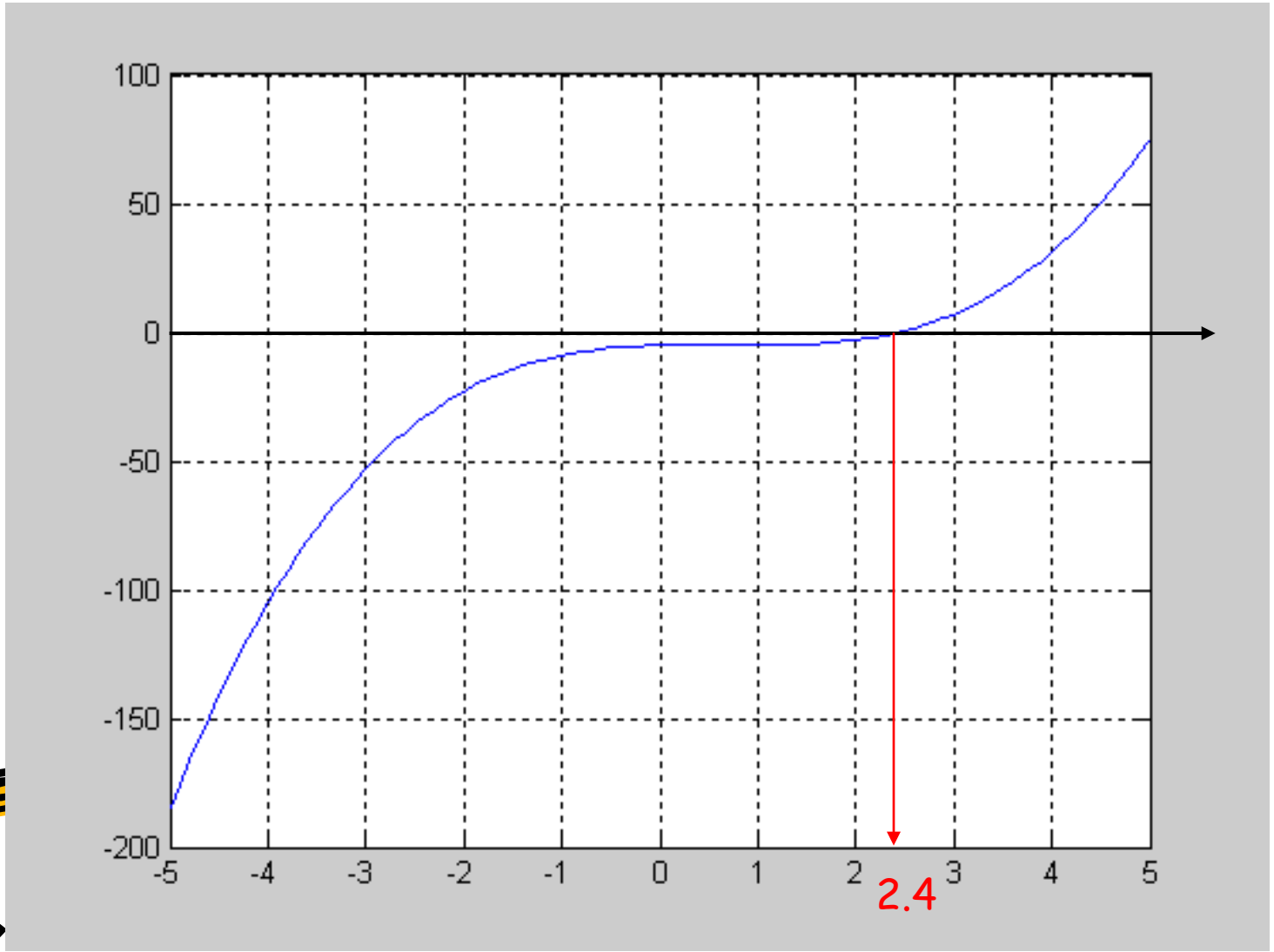
- กำหนด Function y=f(x)
  - Zeroes ของ Function คือค่าของ x ที่ทำให้ f(x)=0 หรือค่า y=0
  - คือราก(root) ของสมการ f(x)=0 นั่นเอง
- ตัวอย่าง $y=x^3-2x^2+x-5$
  - สมการ Polynomial, degree = 3
  - ถ้าเกิน Quadratic (Degree = 2) การหารากจะทำได้ยาก, ไม่มีวิธีทาง Analytic Method โดยตรง เหมือน $y=Ax^2+Bx+C=0$

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

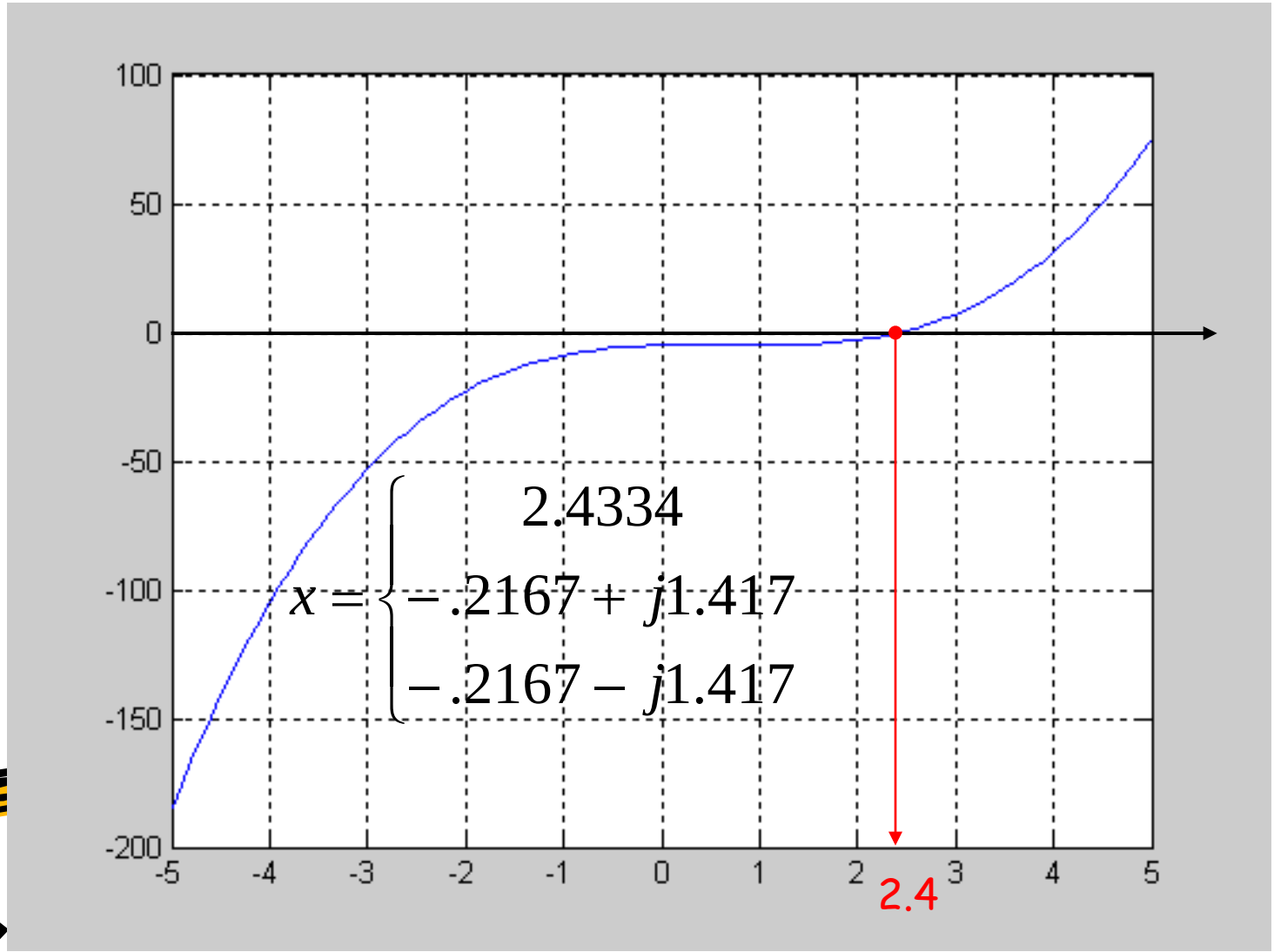
Zeroes of Functions

- ตัวอย่าง $y=x^3-2x^2+x-5$
  - วิธีทาง Numerical ที่เราเคยเรียนมาในวิชาเรขาคณิตย์ คือ Plot Graph และหารากของสมการจาก Graph
    - 1. แต่วิธีนี้จะให้ค่าที่หยาบ
    - 2. กรณีที่เป็น Complex Root จะหาไม่ได้ จาก Graph

Zeroes of Functions
$y=x^3-2x^2+x-5$



Zeroes of Functions
$y=x^3-2x^2+x-5$

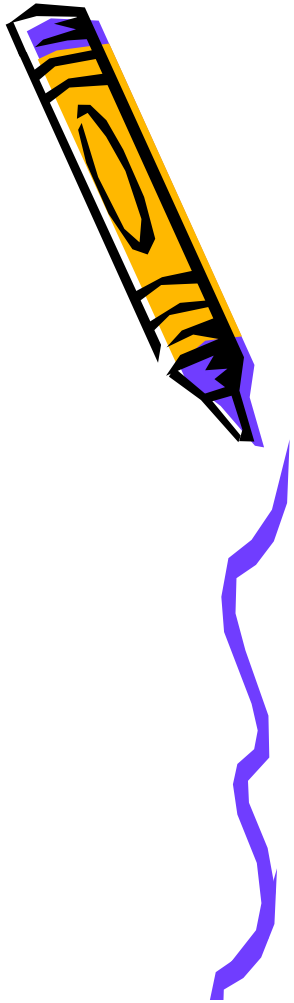


Roots of Equation

- วิธีที่จะกล่าวต่อไป
  - Numerical Method = Algorithm
  - หา Zero Crossing (รากที่เป็นค่าจริง)
  - สามารถได้คำตอบให้ถูกต้องด้วย Significant Digit มากเท่าที่เราต้องการ
    - ต้อง Run Algorithm นานขึ้น
    - Algorithm จะต้อง Converge
  - เป็น Iterative Method
    - ถ้า Algorithm Converge, แต่ละ Iteration ที่ Run จะให้คำตอบที่ถูกต้องขึ้นเรื่อยๆ

      จะช้าหรือเร็วขึ้นอยู่กับ Convergence Rate ของแต่ละ Algorithm

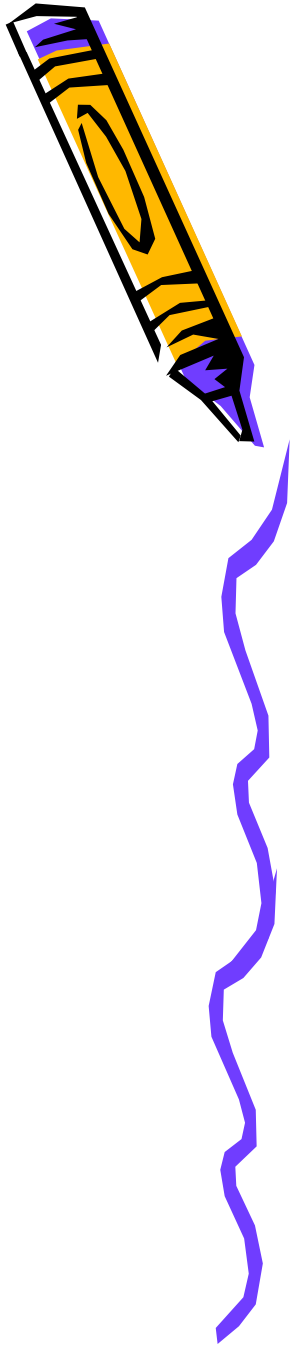
Methods



- Bracketing Method
  - Bisection
  - False Position
- Open Method
  - Simple one-point Iteration
  - Newton-Ralphson
  - Secant Method

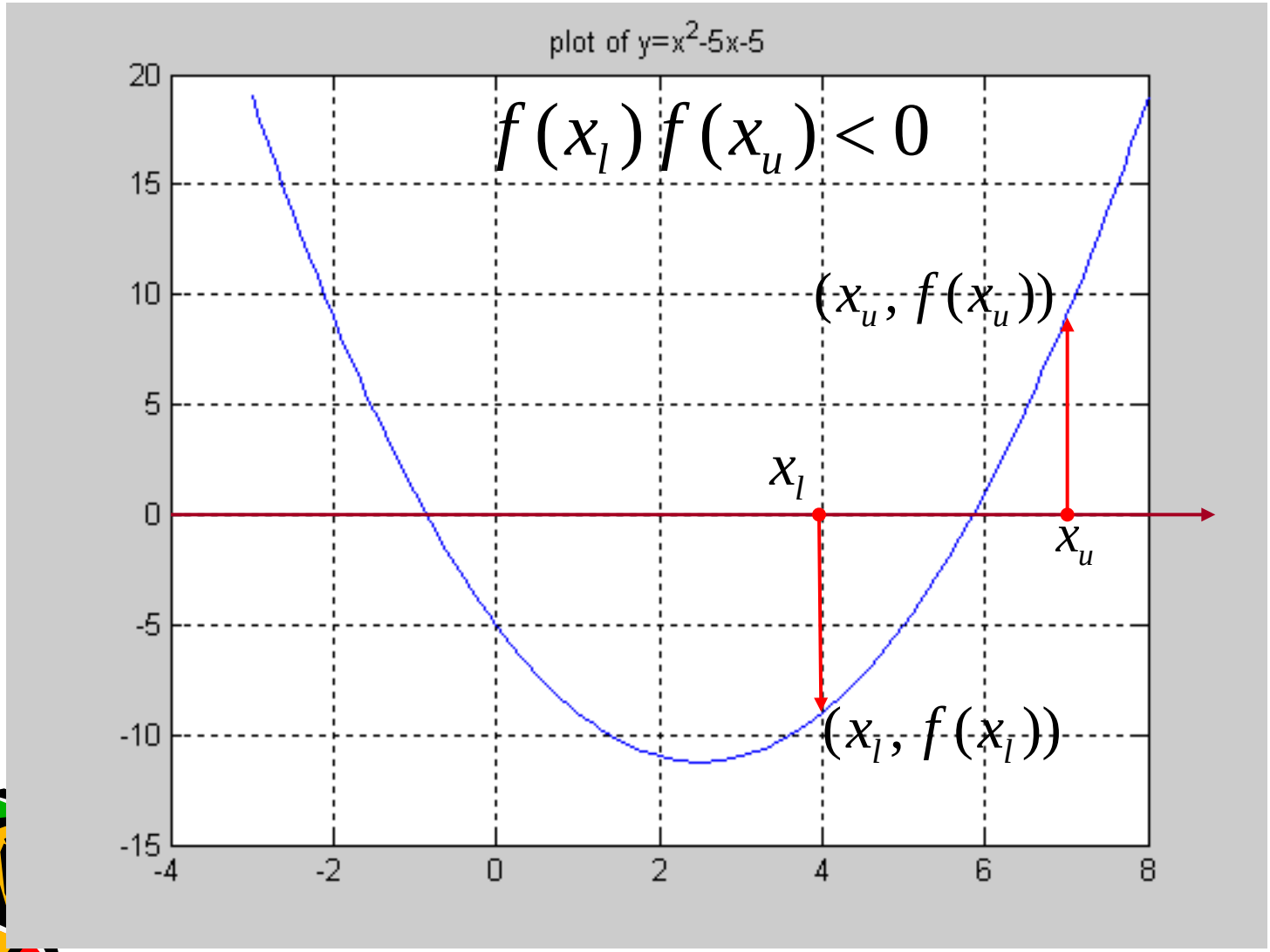
กรรมวิธีในการหารากของสมการจะแบ่งออกเป็นสองกลุ่มใหญ่ๆ กลุ่มแรกเรียก Bracketing Method เป็นวิธีที่เมื่อเรารู้ว่าคำตอบจะต้องอยู่ในช่วงใดช่วงหนึ่งของค่า x ดังนั้นวิธีนี้จะมีการกำหนดค่าสองค่า คือช่วงที่เราจะหารากของสมการ วิธีนี้เราสามารถแน่ใจได้ว่าโปรแกรมจะ Converge อีกวิธีหนึ่งเรียก Open Method ซึ่งเราไม่จำเป็นจะต้องรู้ล่วงหน้าถึงช่วงที่คำตอบจะต้องอยู่ เราอาจจะสมมุติค่าตั้งต้นขึ้นมาค่าหนึ่งสำหรับ Algorithm และ Algorithm มันจะเริ่มทำงานจากค่าตั้งต้นนี้ จนถึงคำตอบที่ต้องการ อย่างไรก็ตาม วิธีนี้จะไม่ Guarantee ว่าโปรแกรมจะ Converge ขึ้นอยู่กับการสมมุติค่าตั้งต้นของเรา

Bracketing Method

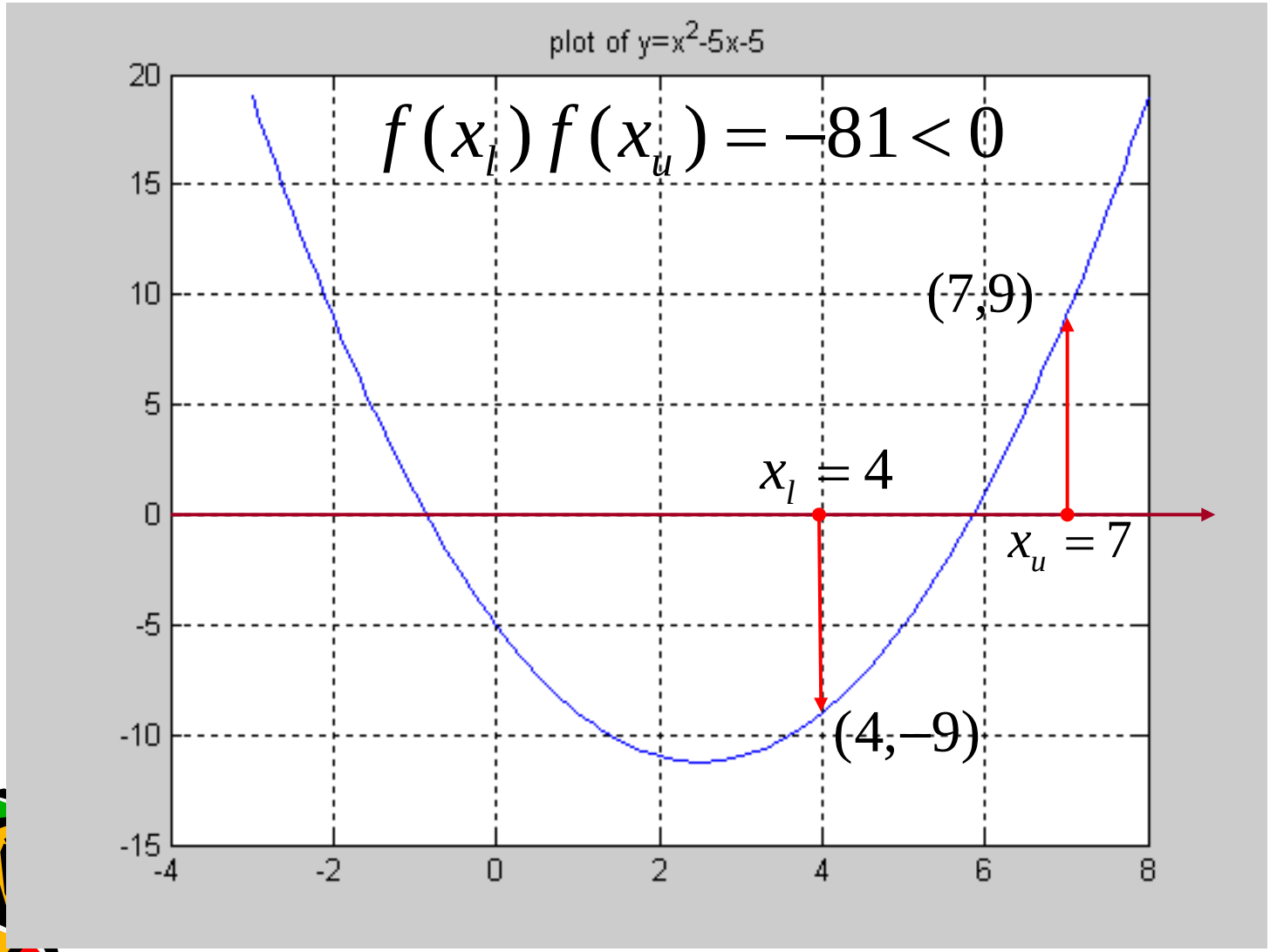


- ใช้หลักความจริงที่ว่า เมื่อ $f(x)=0$ มันจะต้องเปลี่ยนเครื่องหมาย
  - ดังนั้น $f(x^-)f(x^+) < 0$ เมื่อ $f(x)=0$
- เราเริ่มจาก สองค่าของ x คือ x_l และ x_u ที่มีคุณสมบัติ $f(x_l)f(x_u) < 0$
  - อย่างน้อยต้องมีคำตอบหนึ่งอยู่ในช่วงนี้
- *Algorithm* จะ *Search* หาคำตอบในช่วง Bracket นี้ โดยจะลดขนาดของ Bracket ลงเรื่อยๆ

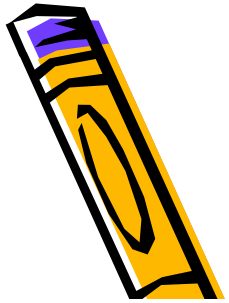
Bracketing Method



Bracketing Method



Bracketing Method:Bisection



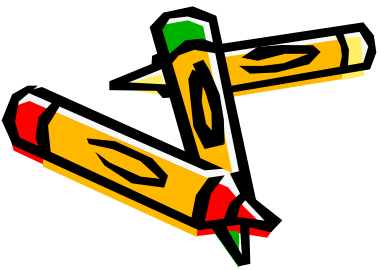
จากที่กล่าวมาแล้ว รากของสมการที่จุด x ที่ค่าของ $f(x)=0$ ดังนั้นที่จุดนี้ $f(x)$ จะมีค่าเปลี่ยนจากบวกเป็นลบ หรือเปลี่ยนจากลบเป็นบวก ถ้าเรากำหนดสองจุดคือ x_l และ x_u จากนั้นคำนวณหา $f(x_l)$ และ $f(x_u)$ และถ้า

$$f(x_l)f(x_u)<0$$

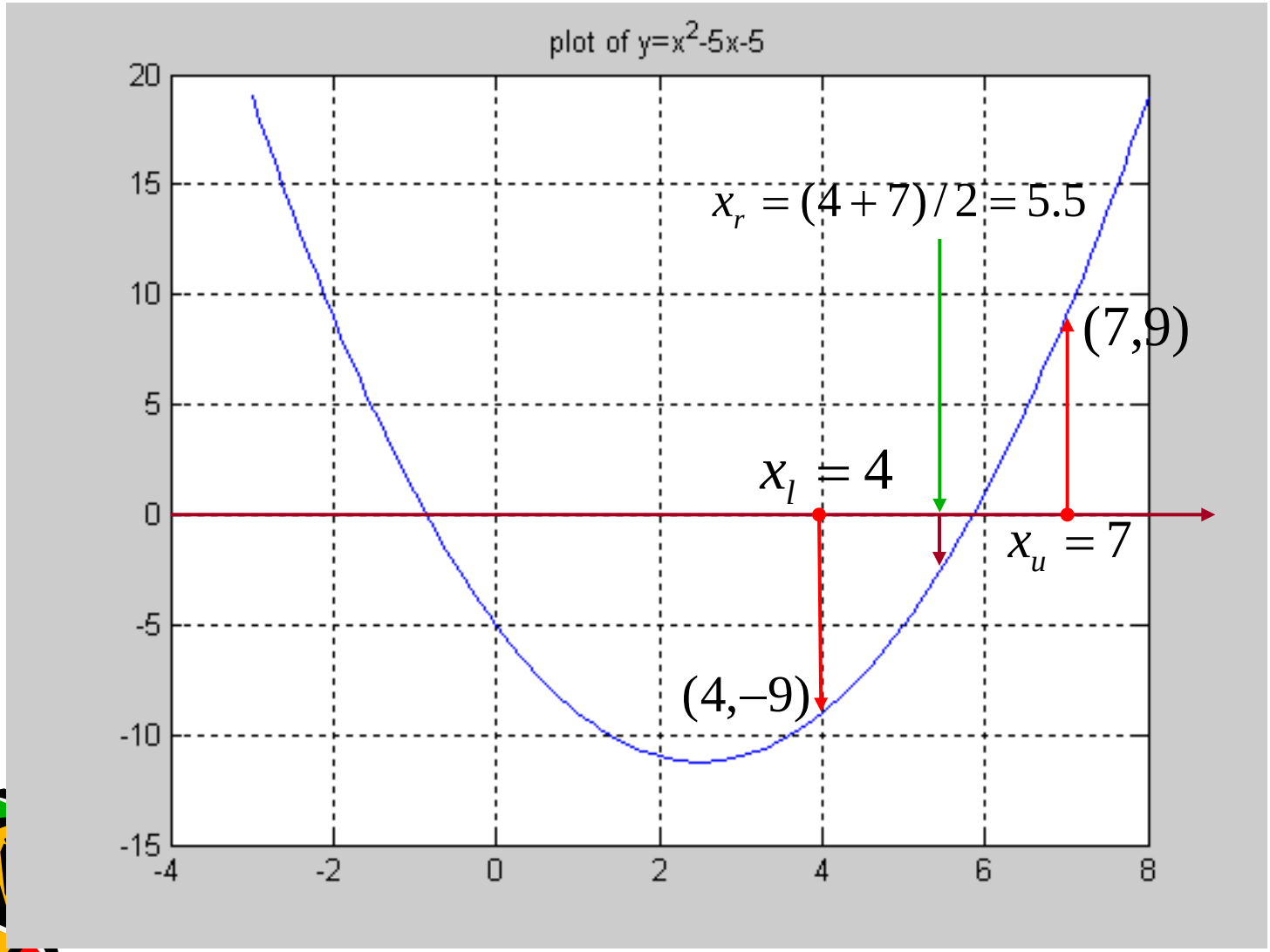
หมายถึงว่าในช่วง $[x_l, x_u]$ Function จะมีการเปลี่ยนเครื่องหมาย และจะต้องผ่านจุด $f(x)=0$ กล่าวอีกนัยหนึ่งก็คือจะต้องมีอย่างน้อยหนึ่งรากของสมการในช่วงนี้

จากนั้นเราลดขนาดช่วงให้เล็กลงโดยแบ่งครึ่งที่จุด $x_r=\dfrac{x_l+x_u}{2}$ และหาว่า Function มีการเปลี่ยนเครื่องหมายในช่วงไหน เราจะลดขนาดของช่วงที่ต้องการหาไปได้ครึ่งหนึ่ง และทำเช่นนี้ต่อไปเรื่อยๆจนได้คำตอบที่พอใจ

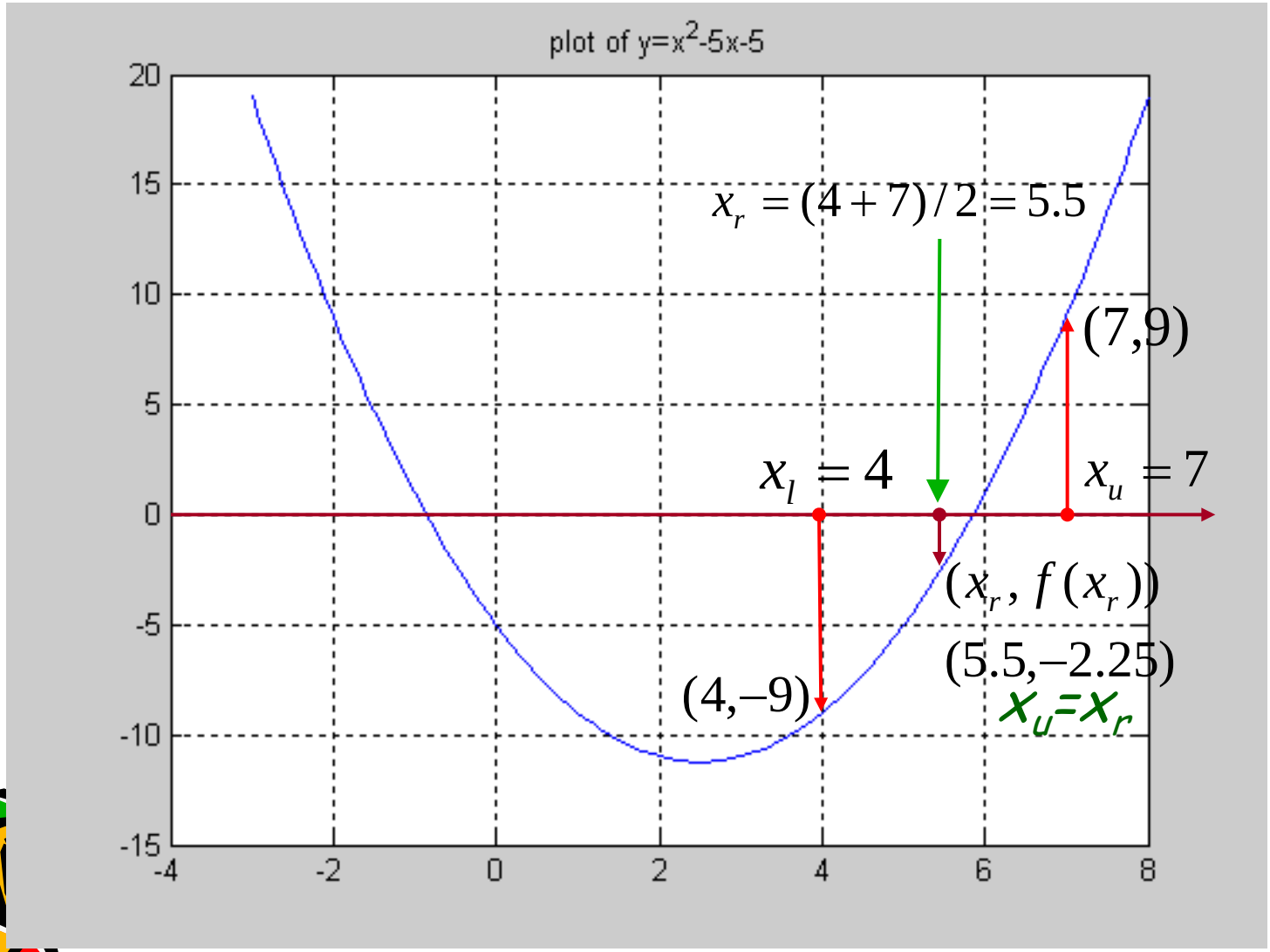
หลักการดังกล่าวก็คือหลักการของ Bisection Method หรือ Bolzano's Method และสามารถสรุปเป็น Algorithm ได้สามขั้นตอนดังนี้



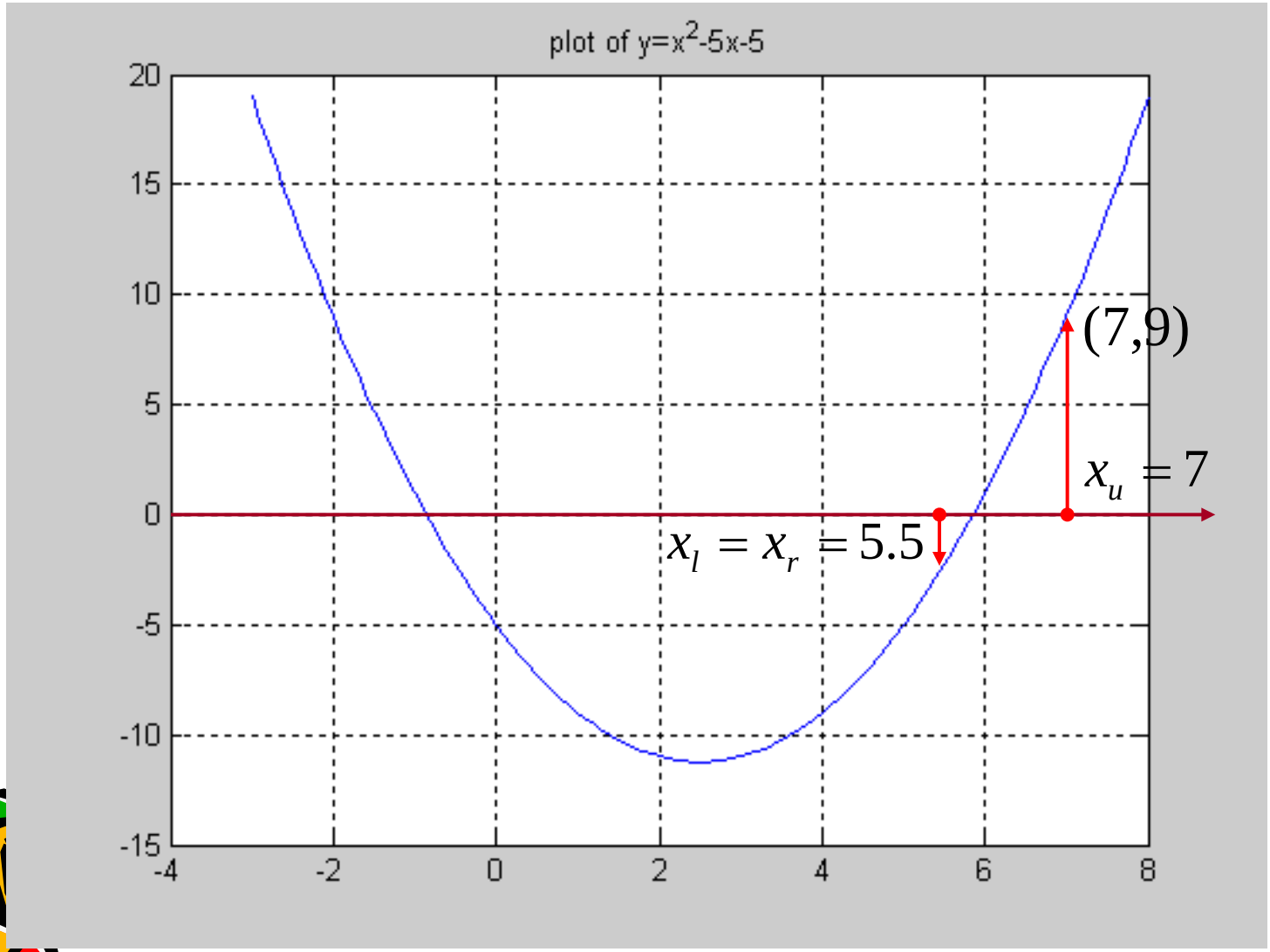
Bracketing Method



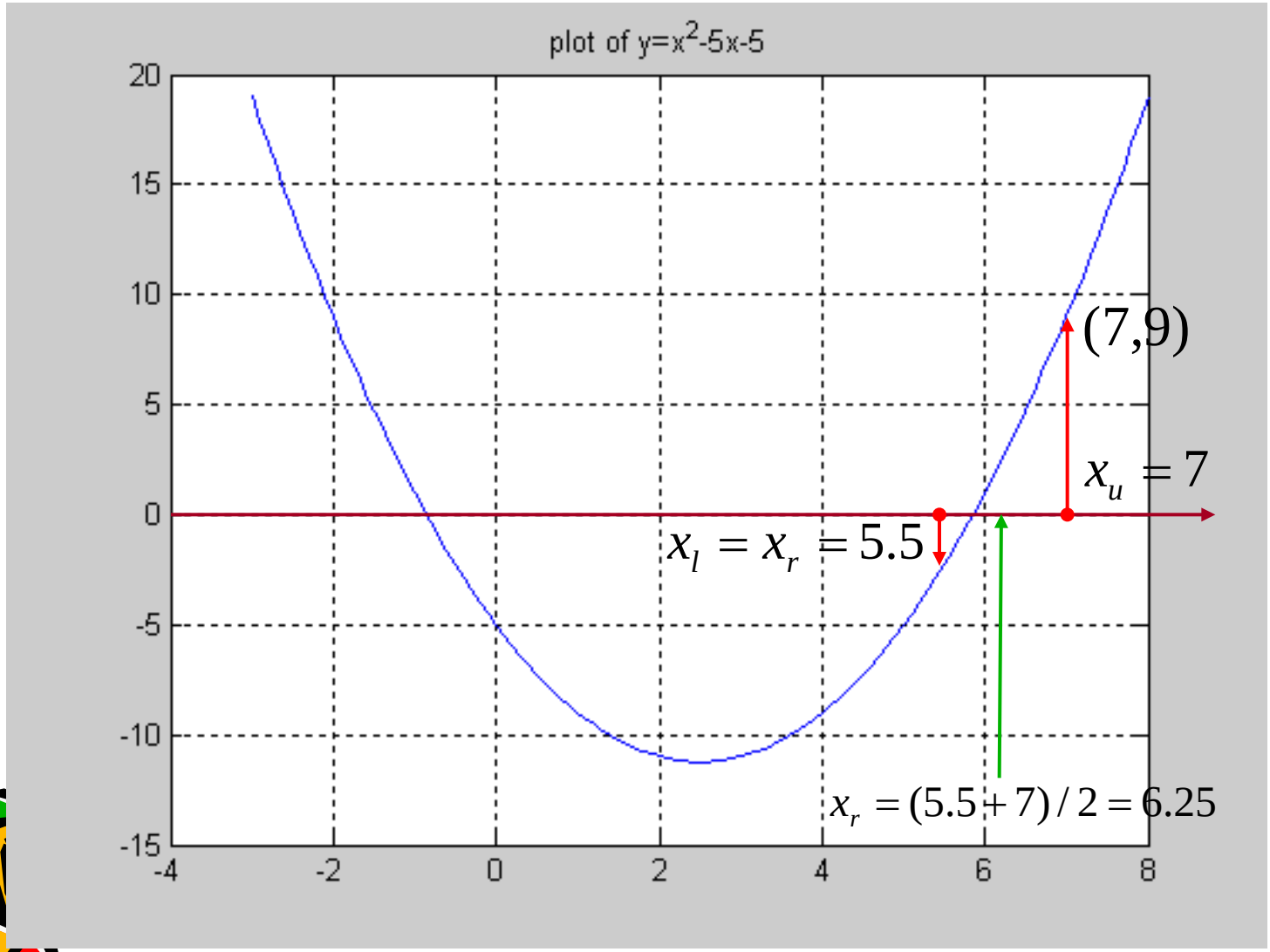
Bracketing Method



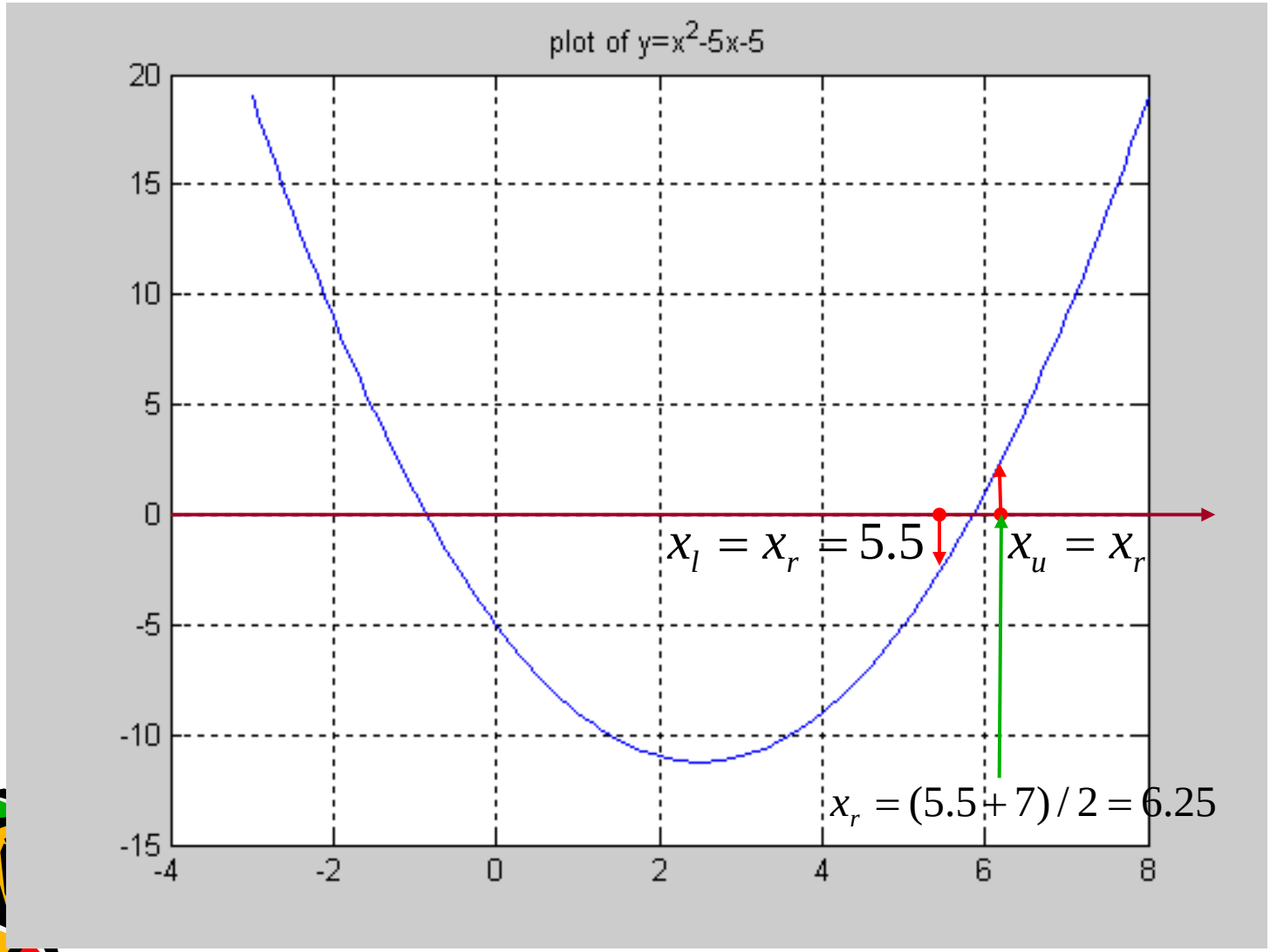
Bracketing Method



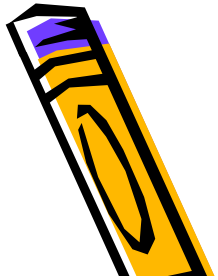
Bracketing Method



Bracketing Method



Bracketing Method:Bisection



Algorithm:

Step 1: ทำการเดาจุดสองจุดคือค่า x_l และค่า x_u สมมุติว่าค่า x_l เป็นค่าที่ต่ำกว่า จากนั้นทดสอบว่า $f(x_l)f(x_u)<0$ ถ้าไม่ใช่ให้หาจุดใหม่ ซึ่งจากขั้นตอนนี้ เรารู้ว่ารากจะต้องอยู่ในช่วงนี้

Step 2: ทำการประมาณค่า Root ที่ต้องการที่จุดกึ่งกลางระหว่างสองค่าใน Step 1 โดยคำนวณ $x_r = \frac{x_l + x_u}{2}$

Step 3: หาว่าตอนนี้ Root ที่ต้องการอยู่ในครึ่งไหนดังนี้

3.1 ถ้า $f(x_l)f(x_r)<0$ แสดงว่า Root ที่ต้องการอยู่ในครึ่งล่าง ให้ตั้ง $x_u = x_r$ และกลับไปทำ Step 2

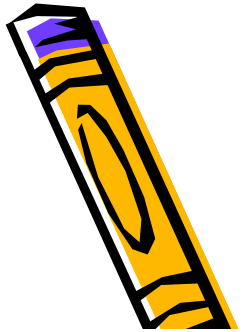
3.2 ถ้า $f(x_l)f(x_r)>0$ แสดงว่า Root ที่ต้องการอยู่ในครึ่งบน ให้ตั้ง $x_l = x_r$ และกลับไปทำ Step 2

3.3 ถ้า $f(x_l)f(x_r)=0$ แสดงว่าคำตอบที่ต้องการเท่ากับ x_r

จากสมการของ Error Approximation เราสามารถหาค่า Error Estimate ของ Bisection Method ได้จาก

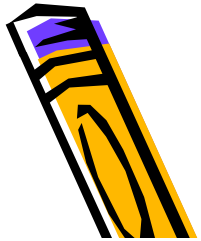
$$|e_a| = \left|\frac{x_r^{New} - x_r^{Old}}{x_r^{New}}\right| \times 100\%$$

Bracketing Method:Bisection



พึงสังเกตว่า แม้ว่าวิธีนี้จะให้ค่า $|e_a|$ ที่ลดลงเรื่อยๆ แต่ค่า $|e_t|$ จะมีการแปรผันขึ้นลงได้ ดังนั้นค่า Approximate Error จะไม่สอดคล้องกับ True Error ทีเดียวนัก ตามปกติเวลาเรา Run โปรแกรม เราจำเป็นต้องตั้ง Condition ที่โปรแกรมจะหยุดทำงาน ซึ่งมักจะกำหนดเป็นค่า Threshold Error, e_s โดยที่แต่ละ Iteration จะมีการตรวจสอบค่า e_a และเมื่อใดมันมีค่าต่ำกว่า e_s (ค่า Absolute) โปรแกรมจะหยุดทำงาน และให้คำตอบออกมา อีกอย่างหนึ่งก็คือ $|e_a|$ ที่ได้จะสูงกว่า $|e_t|$ เสมอ

Bracketing Method:Bisection



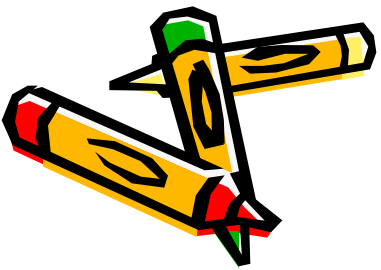
Example 7.1: สมการสำหรับหาความเร็วของนักโดดร่มสามารถแสดงได้ดังนี้ $v = \frac{gm}{c}[1 - e^{-(c/m)t}]$ โดยที่ v คือ ความเร็ว เป็น Dependent Variable, t คือเวลา เป็น Independent Variable, g เป็น Gravitational Constant, c เป็นค่าคงที่ของการฉุดของร่ม(Drag Coefficient) และ m คือมวลของนักโดดร่มและร่ม

ในการออกแบบร่มชูชีพ เราต้องการหาค่า Drag Coefficient ที่จะคงความเร็วของนักโดดร่มให้ได้ในช่วงเวลาที่กำหนด ซึ่งการแก้สมการหาค่า c จะไม่สามารถทำได้ทางคณิตศาสตร์ ในการนี้เราจะเอาคอมพิวเตอร์เข้ามาช่วย โดยการแก้สมการ

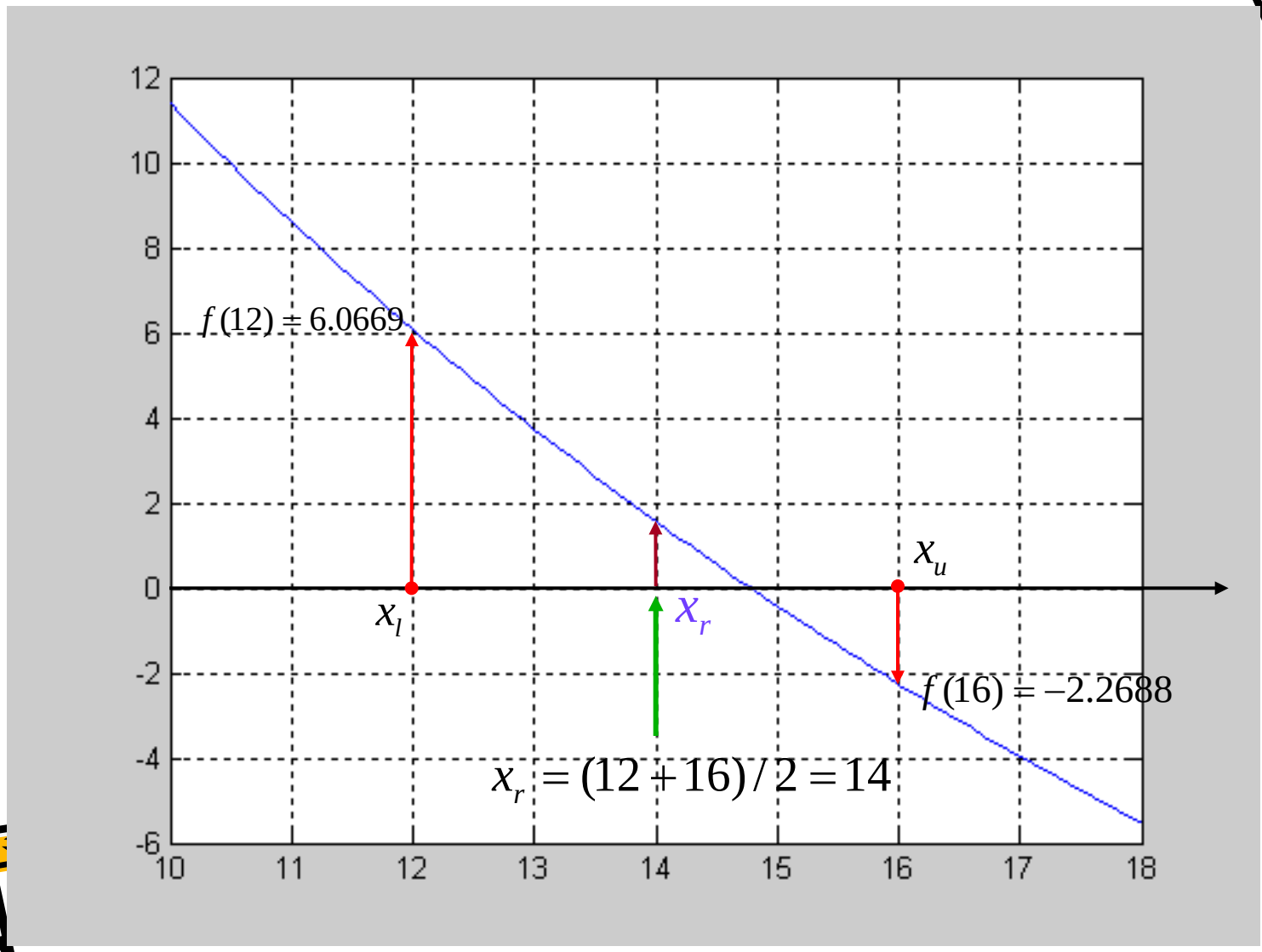
$$f(c) = \frac{gm}{c}[1 - e^{-(c/m)t}] - v = 0$$

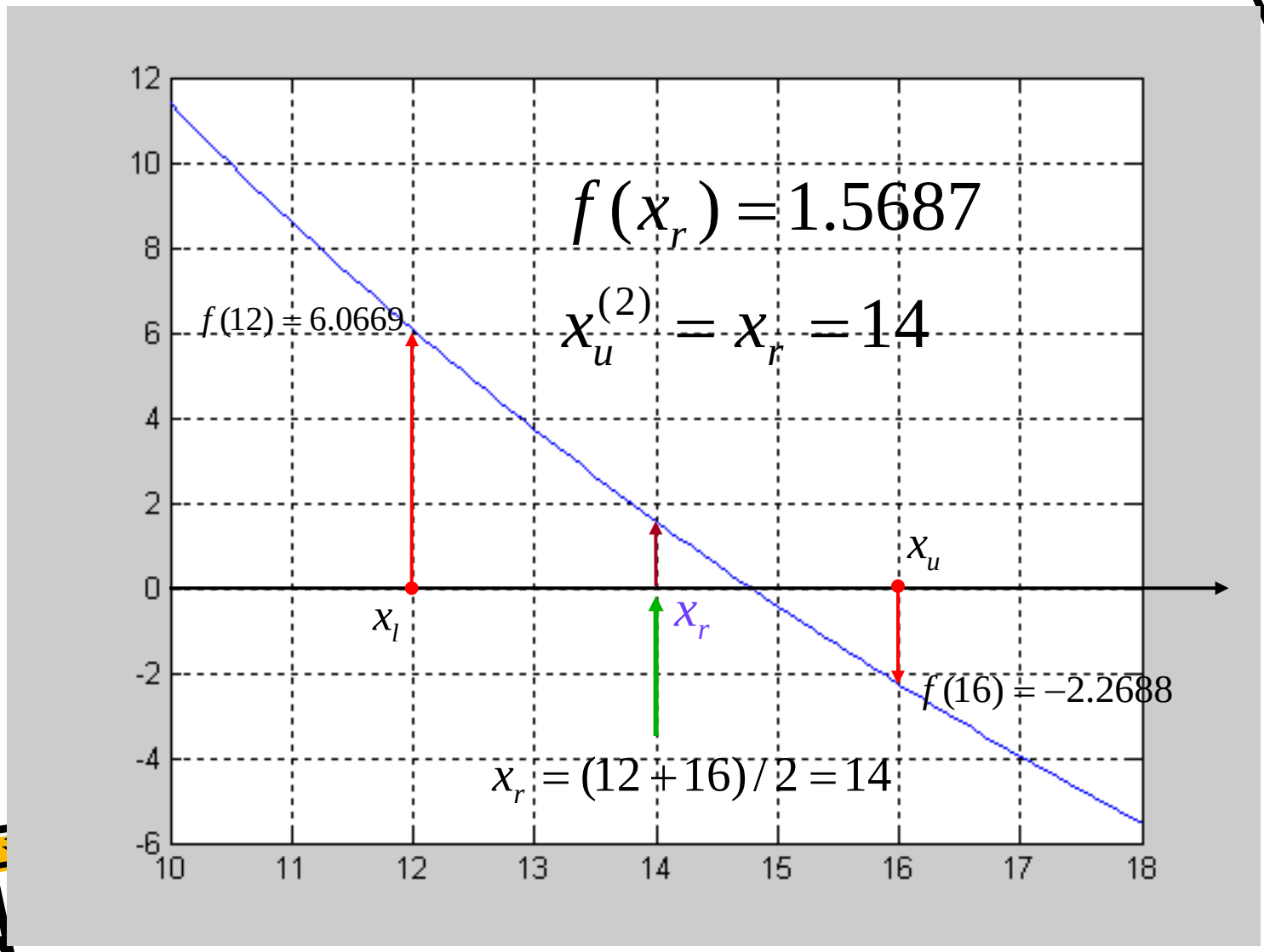
และหาค่า c ออกมา ซึ่งก็คือรากของสมการดังกล่าว

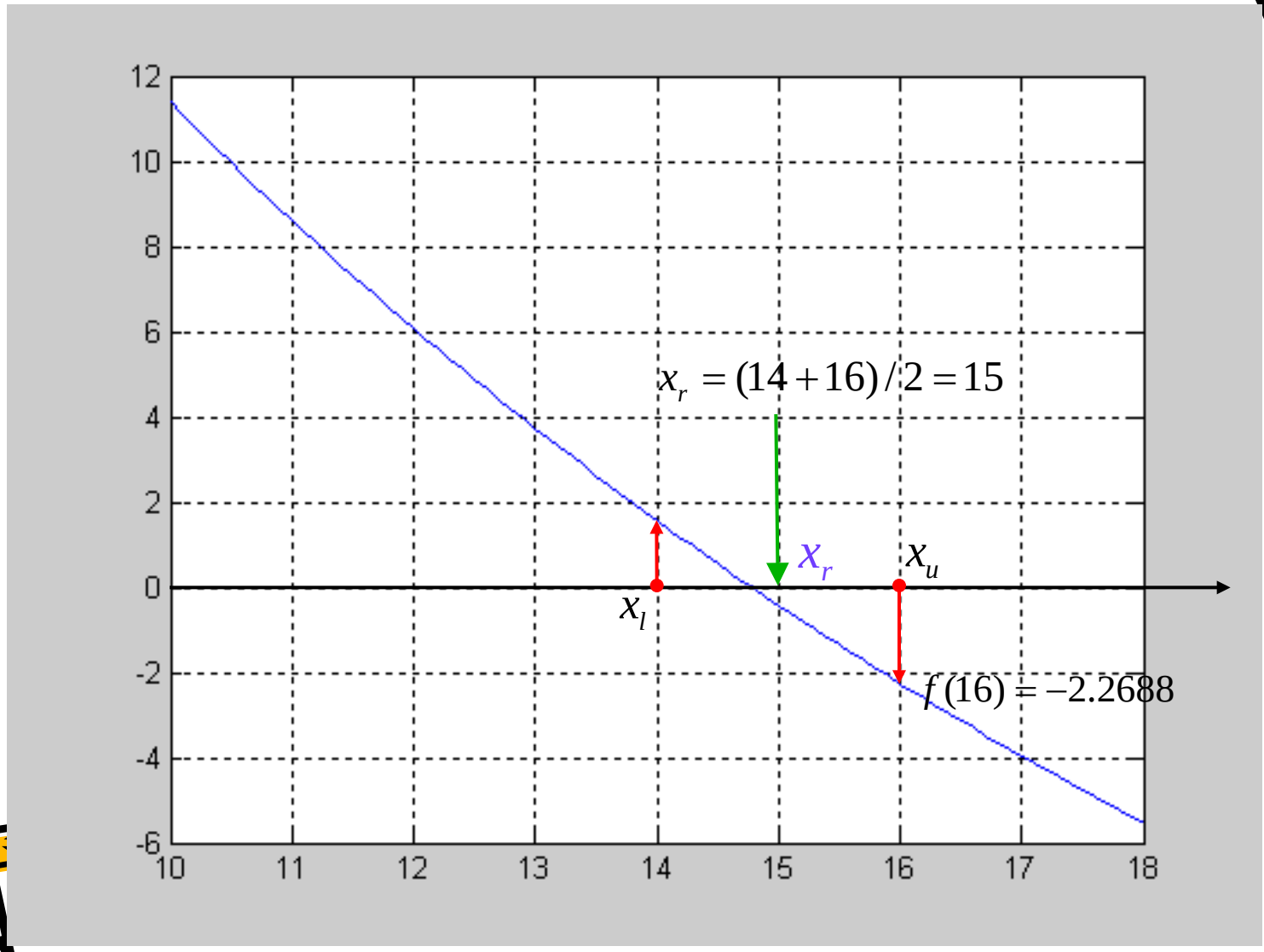
สมมุติว่า $m = 68.1\ kg$, $g = 9.8\ m/s^2$ เราต้องการหาค่า c ที่จะให้นักโดดร่มคงความเร็วอยู่ที่ $40\ m/s$ ในเวลา 10 วินาที และค่า c ควรมีค่าอยู่ระหว่าง 12 และ 16



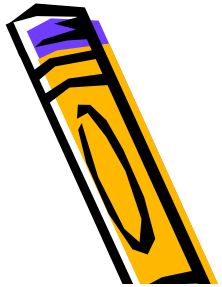
$$f(c) = \frac{667.38}{c}(1 - e^{-10c/68.1}) - 40 = 0$$


12
10
8
6
4
2
0
-2
-4
-6
10
11
12
13
14
15
16
17
18
f(12) = 6.0669
x_l
x_r
x_u
f(16) = −2.2688
x_r = (12 + 16) / 2 = 14


$f(x_r) = 1.5687$
$x_u^{(2)} = x_r = 14$
$f(12) = 6.0669$
x_l
x_r
x_u
$f(16) = -2.2688$
$x_r = (12 + 16)/2 = 14$
12
10
8
6
4
2
0
-2
-4
-6
10
11
12
13
14
15
16
17
18


12
10
8
6
4
2
0
-2
-4
-6
10
11
12
13
14
15
16
17
18
$x_r = (14+16)/2 = 15$
x_r
x_u
x_l
$f(16) = -2.2688$

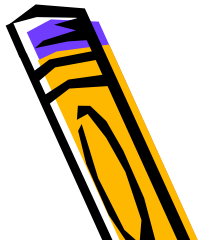
Bracketing Method: Bisection



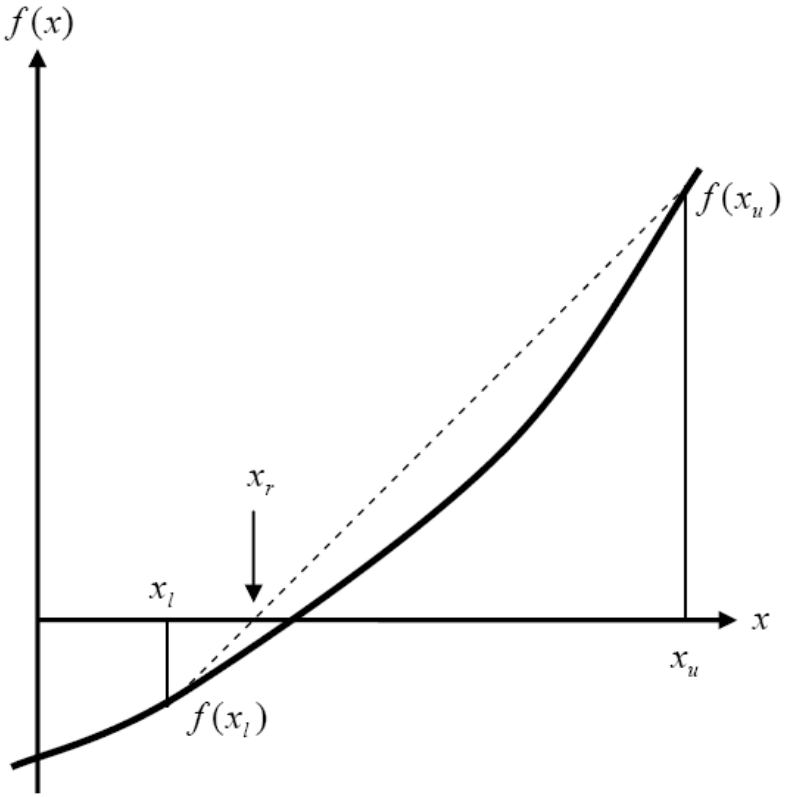
สังเกตว่า $f(12)f(16) < 0$ และคำตอบรวมทั้งค่า Error สรุปได้ดังตาราง(ค่า e_t คำนวณจากค่า True Value)

Iteration	x_l	x_u	x_r	$\lvert e_a \rvert$, %	$\lvert e_t \rvert$, %
1	12	16	14		5.279
2	14	16	15	6.667	1.487
3	14	15	14.5	3.448	1.896
4	14.5	15	14.75	1.695	0.204
5	14.75	15	14.875	0.840	0.641
6	14.75	14.875	14.8125	0.422	0.219

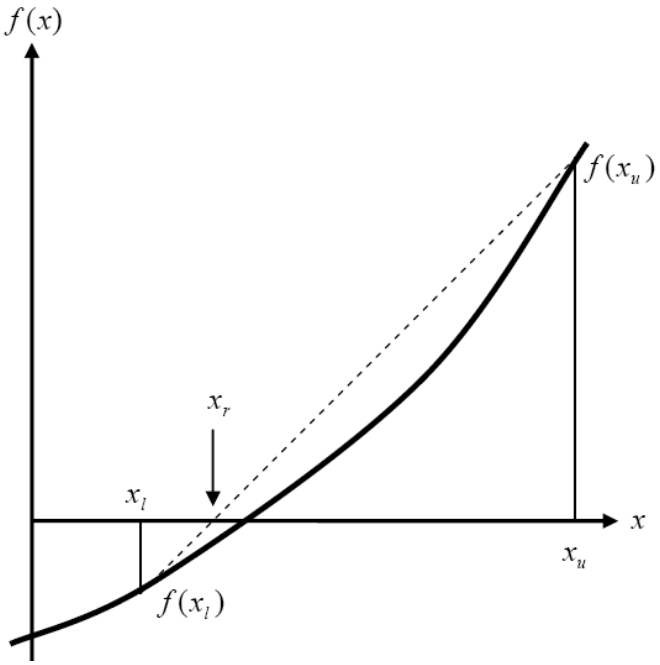
False-Position Method



แม้ว่า Bisection Method จะเป็นวิธีการที่ดี แต่ประสิทธิภาพสามารถปรับปรุงได้เพิ่มขึ้น เนื่องจากมันจะ Estimate ค่า Root ใหม่ที่กึ่งกลางช่วงเสมอ ซึ่งค่า Root ที่แท้จริงอาจจะอยู่ใกล้จุดปลายด้านใดด้านหนึ่ง ทำให้วิธีนี้ Converge ช้ากว่าที่ควรจะเป็น ดังนั้นเราควรจะมีวิธีในการประมาณค่า Root ใหม่ วิธีที่ดีกว่าคือใช้สมการเส้นตรง หรือที่เรียก Linear Interpolation(ดูรูป) และกรรมวิธีนี้เรียก False-Position Method หรือ Linear Interpolation Method



False-Position Method



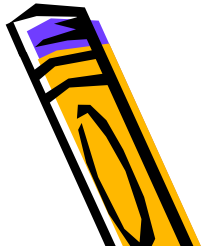
จากสามเหลี่ยมคล้ายในรูป จุดที่เส้นตรงตัดกับแกน x สามารถคำนวณได้จากสมการ

$$\frac{f(x_l)}{x_r - x_l} = \frac{f(x_u)}{x_r - x_u}$$

ดังนั้นเราสามารถแก้สมการหาค่า x_r ได้ดังนี้

$$x_r = x_u - \frac{f(x_u)(x_l - x_u)}{f(x_l) - f(x_u)} \qquad \text{(False-Position Method)}$$

False-Position Method



Example 7.2: จากตัวอย่างเดิม $f(c)=\frac{gm}{c}[1-e^{-(c/m)t}]-v=0$ ให้จะลองใช้ False-Position Method ทำการแก้ปัญหา

$$x_l=12, \quad x_u=16$$

Iteration 1:

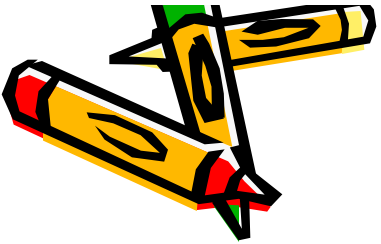
$$x_l=12, \quad x_u=16$$

$$f(x_l)=6.0669, \quad f(x_u)=-2.2688$$

$$x_r=16-\frac{-2.2688(12-16)}{6.0669-(-2.2688)}=14.9113, \quad e_t=0.89\%$$

Iteration 2:

$$f(x_l)f(x_r)=-1.5426 \Rightarrow x_u=x_r=14.9113$$



False-Position Method

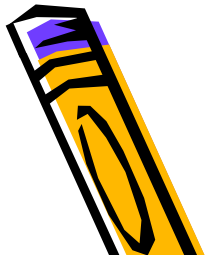
$$x_l = 12, \quad x_u = 14.9113$$

$$f(x_l) = 6.0669, \quad f(x_u) = -0.2543$$

$$x_r = 14.9113 - \frac{-0.2543(12 - 14.9113)}{6.0669 - (-0.2543)} = 14.7942, \quad e_t = 0.09\%, e_a = 0.79\%$$

จากตัวอย่างข้างบน จะดูเหมือนว่า Convergence Rate ของ False-Position Method จะดีกว่า Bisection Method มาก ทั้งนี้เนื่องจากเราได้ค่า Estimate ของ Solution ที่ใกล้เคียงกับค่าจริงมากกว่า และปกติก็เป็นเช่นนั้น ทำให้วิธีนี้นิยมใช้มากว่าวิธีของ Bisection อย่างไรก็ตาม ในบางกรณี วิธีของ False-Position จะ Converge ได้ช้ากว่า Bisection Method มาก ดังตัวอย่างถัดไป

False-Position Method



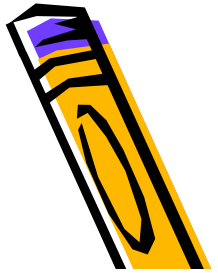
Example 7.3: จงเปรียบเทียบการทำงานของวิธี Bisection Method และ False-Position Method ในการหารากของสมการ $f(x) = x^{10} - 1$ โดย Root ที่จะหามีค่าอยู่ระหว่าง $[0,1.3]$

วิธีของ Bisection Method จะสรุปเป็นตารางได้ดังนี้

Iteration	x_l	x_u	x_r	$\lvert e_t \rvert$, %	$\lvert e_a \rvert$, %
1	0	1.3	0.65	35	100.0
2	0.65	1.3	0.975	2.5	33.3
3	0.975	1.3	1.1375	13.8	14.3
4	0.975	1.1375	1.05625	5.6	7.7
5	0.975	1.05625	1.015625	1.6	4.0

สังเกตได้ว่า หลังจาก Iteration ที่ 5 ค่า Error จะลดลงไม่ถึง 2%

False-Position Method

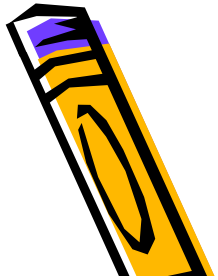


วิธีของ False-Position Method จะสรุปเป็นตารางได้ดังนี้

Iteration	x_l	x_u	x_r	$\lvert e_t \rvert$, %	$\lvert e_a \rvert$, %
1	0	1.3	0.09430	90.6	
2	0.09430	1.3	0.18176	81.8	48.1
3	0.18176	1.3	0.26287	73.7	30.9
4	0.26287	1.3	0.33811	66.2	22.3
5	0.33811	1.3	0.40788	59.2	17.1

พึงสังเกตอีกว่าในกรณีนี้ $|e_a| < |e_t|$ ซึ่งกลับกับวิธีก่อน ทำให้การพิจารณาจากค่า e_a ทำให้เราเข้าใจผิดได้ เหตุผลก็คือการ Interpolation Fail เพราะตำแหน่งที่ได้แย่กว่าวิธีของ Bisection Method เนื่องจาก Function มีการเปลี่ยนแปลงอย่างกะทันหันในช่วง Bracket

Open Method: Simple One-Point Iteration



ในกรรมวิธีของ Bracket Method ค่าของ Root จะอยู่ในช่วงที่กำหนดไว้ ดังนั้นในแต่ละ Iteration เราจะทำให้ช่วงนี้แคบลง และผลลัพธ์จะ Converge อย่างไรก็ตาม ใน Open Method จะกำหนดแค่ค่าตั้งต้น หรือช่วงที่ค่าของ Root อาจจะไม่อยู่ในช่วงนี้ก็ได้ ดังนั้นบางครั้งโปรแกรมจะ Diverge แต่ถ้าเราเลือกค่าที่เหมาะ โปรแกรมจะ Converge และปกติจะ Converge ได้รวดเร็วกว่า Bracket Method มาก

วิธีการง่ายที่สุดในการหารากของสมการ ทำได้โดยการจัดเรียงสมการใหม่ในรูป $x = g(x)$ คือยามยามย้ายค่า x บางส่วนมาอยู่ด้านซ้ายของสมการ และทำการประมาณค่าใหม่จากค่าตั้งต้น ทำเช่นนี้เรื่อยไปจนได้คำตอบที่มีค่า Precision ที่ต้องการ สังเกตว่าบางครั้งการจัดเรียงสมการใหม่สามารถกระทำได้หลายวิธี และจะมีผลต่อการ Converge และผลลัพธ์ที่ได้

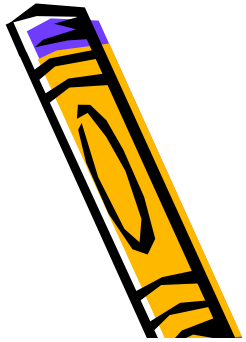
ยกตัวอย่างสมการ $x^2 - 3x + 3 = 0$ สามารถจัดเรียงใหม่ในรูป $x = g(x)$ ได้เป็น

$$x = \frac{x^2 + 3}{3} \qquad \text{หรือ} \qquad x = \sqrt{3x - 3}$$

ถ้าเราทำ Iteration ของสองสมการนี้ จะได้การ Converge ที่ต่างกัน

บางครั้งสมการสามารถสร้างได้โดยการบวกด้วย x ทั้งสองข้างเช่นจากสมการ $\sin x = 0$ เราได้สมการในรูป $x = g(x)$ คือ $x = \sin x + x$ เป็นต้น

Open Method: Simple One-Point Iteration



การทำงานของ Algorithm จะประมาณค่าใหม่ x_{i+1} จากค่าเดิม x_i ใน Iteration ที่ $i+1$ โดยให้ x_0 เป็นค่าตั้งต้นดังนี้

$$x_{i+1} = g(x_i)$$

ดังนั้นค่า Estimate Error, e_a สามารถคำนวณได้จาก

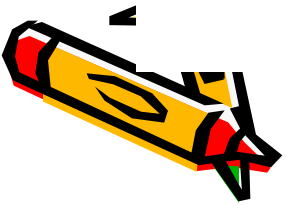
$$|e_a| = \left|\frac{x_{i+1} - x_i}{x_{i+1}}\right| \times 100\%$$

Open Method: Simple One-Point Iteration

Example 7.4: ใช้ Simple One-Point Iteration หารากของสมการ $f(x) = e^{-x} - x$

Solution: ในกรณีนี้เราเขียนสมการใหม่ดังนี้ $x = e^{-x}$ และกำหนด $x_0 = 0$ จากนั้นทำ Iteration

Iteration	x_i	$\lvert e_t \rvert$, %	$\lvert e_a \rvert$, %
0	0	100	
1	1.000000	76.3	100.0
2	0.367879	35.1	171.8
3	0.692201	22.1	46.9
4	0.500473	11.8	38.3
5	0.606244	6.89	17.4
6	0.545396	3.83	11.2
7	0.579612	2.20	5.90



Open Method: Simple One-Point Iteration

Iteration	x_i	$\lvert e_t \rvert$, %	$\lvert e_a \rvert$, %
8	0.560115	1.24	3.48
9	0.571143	0.705	1.93
10	0.564879	0.399	1.11

โดยค่าที่แท้จริงของ Root คือ 0.56714329

สังเกตว่า Error ที่ได้ในแต่ละ Iteration จะเป็นประมาณ 50-60% เมื่อเทียบกับ Iteration ก่อน และโปรแกรมจะ Converge เข้าสู่ค่าจริงในกรณีนี้ การ Converge เช่นนี้เราเรียกว่าเป็น Linear Convergence

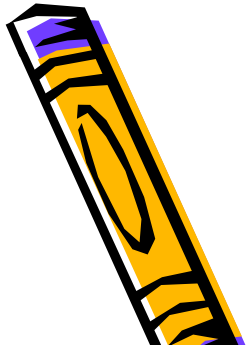
จากที่กล่าวมาแล้วว่าวิธีของ Open Method อาจจะได้โปรแกรมที่ไม่ Converge ยกตัวอย่างสมการที่คล้ายกันในตัวอย่างถัดไป

Example 7.5: ใช้ Simple One-Point Iteration หารากของสมการ $f(x) = e^{-x} - x/4$

Solution: ในกรณีนี้เราเขียนสมการใหม่ดังนี้ $x = 4e^{-x}$ และกำหนด $x_0 = 0$ จากนั้นทำ Iteration

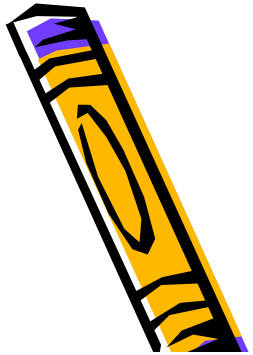
Iteration	x_i	$\lvert e_t \rvert$, %	$\lvert e_a \rvert$, %
0	0	100	
1	4	232.7322	100.0
2	.0733	93.9058	5.3598e+003
3	3.7174	209.2270	98.0292
4	0.0972	91.9158	3.7251e+003
5	3.6296	201.9171	97.3224
6	0.1061	91.1732	3.3205e+003
7	3.5973	199.2339	97.0502
8	0.1096	90.8839	3.1825e+003
9	3.5848	198.1948	96.9429
10	0.1110	90.7693	3.1305e+003
⋮	⋮	⋮	⋮
26261	3.5766	197.5121	96.8718
26262	0.1119	90.6392	3.0967e+003

Open Method: Simple One-Point Iteration



คำตอบที่แท้จริงคือ 1.20216787319704 และในกรณีนี้คำตอบจะ Oscillate และจะไม่ Converge ถ้านักศึกษาลองเปลี่ยนเลข 4 เป็นเลขอื่นที่มีค่ามากกว่านี้ คำตอบอาจจะไม่ Converge เช่นกัน ในกรณีนี้โปรแกรมจะ Diverge และ Algorithm จะ Fail (ในกรณีนี้ ตราบใดที่ค่า Absolute ของ Slope ของ $y_2 = g(x)$ น้อยกว่าค่า Alsolute ของ Slope $y_1 = x$ หรืออีกนัยหนึ่ง เมื่อ $|g'(x)| < 1$ โปรแกรมจะ Converge)

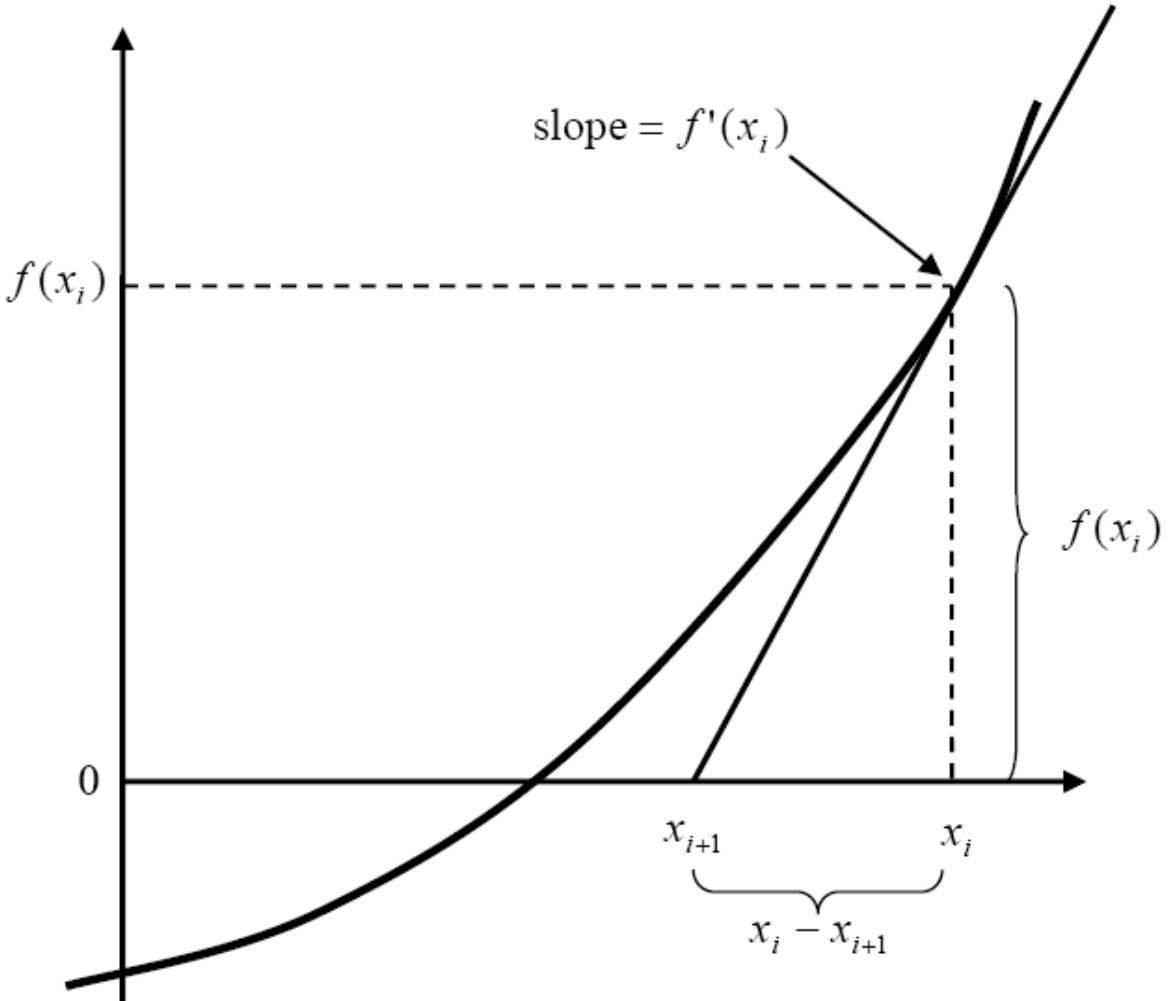
Open Method: Newton-Ralphson Method



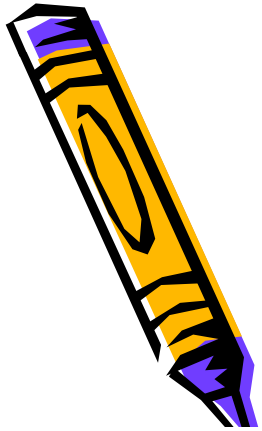
เป็นวิธีที่นิยมมากที่สุดในการหา Root ของสมการ เนื่องจากรวดเร็ว เราสามารถพิสูจน์ได้(จาก Taylor Series Expansion) ว่า ถ้าโปรแกรม Converge แล้ว Error ใน Iteration ใหม่ จะมีค่าประมาณเท่ากับกำลังสองของ Error ใน Iteration ก่อนหน้านี้ และในกรณีนี้เราเรียกว่าเป็น Quadratic Convergence

Open Method: Newton-Ralphson Method

วิธีการของ Newton-Ralphson สามารถอธิบายได้จากรูป



Open Method: Newton-Ralphson Method



ในกรรมวิธีนี้ เราจะ Estimate ค่า x_{i+1} ที่ดีกว่า โดยใช้ค่า Tangent ที่จุด $[x_i, f(x_i)]$ ตัดกับแกน x ซึ่งเขียนเป็นสมการค่า Tangent ได้ดังนี้

$$f'(x_i) = \frac{f(x_i) - 0}{x_i - x_{i+1}}$$

และเมื่อจัดเรียงใหม่ เราได้

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)} \qquad \text{(Newton-Ralphson Formula)}$$

Open Method: Newton-Ralphson Method

Example 7.6: จงใช้กรรมวิธีของ Newton-Ralphson หารากของสมการ $e^{-x} - x$

Solution:

เราได้ $f(x) = e^{-x} - x$ และดังนั้น $f'(x) = -e^{-x} - 1$

ดังนั้นสมการของ Newton-Ralphson จะเป็น

$$x_{i+1} = x_i - \frac{e^{-x_i} - x_i}{-e^{-x_i} - 1}$$

เริ่มจากค่า $x_0 = 0$ เราได้ Iteration ดังนี้

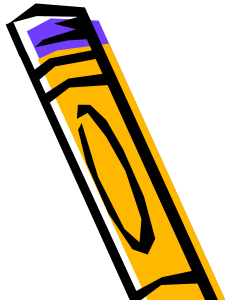
Open Method: Newton-Ralphson Method

Iteration	x_i	$\lvert e_t \rvert$, %
0	0	100
1	0.500000000	11.8
2	0.566311003	0.147
3	0.567143165	0.0000220
4	0.567143290	$<10^{-8}$

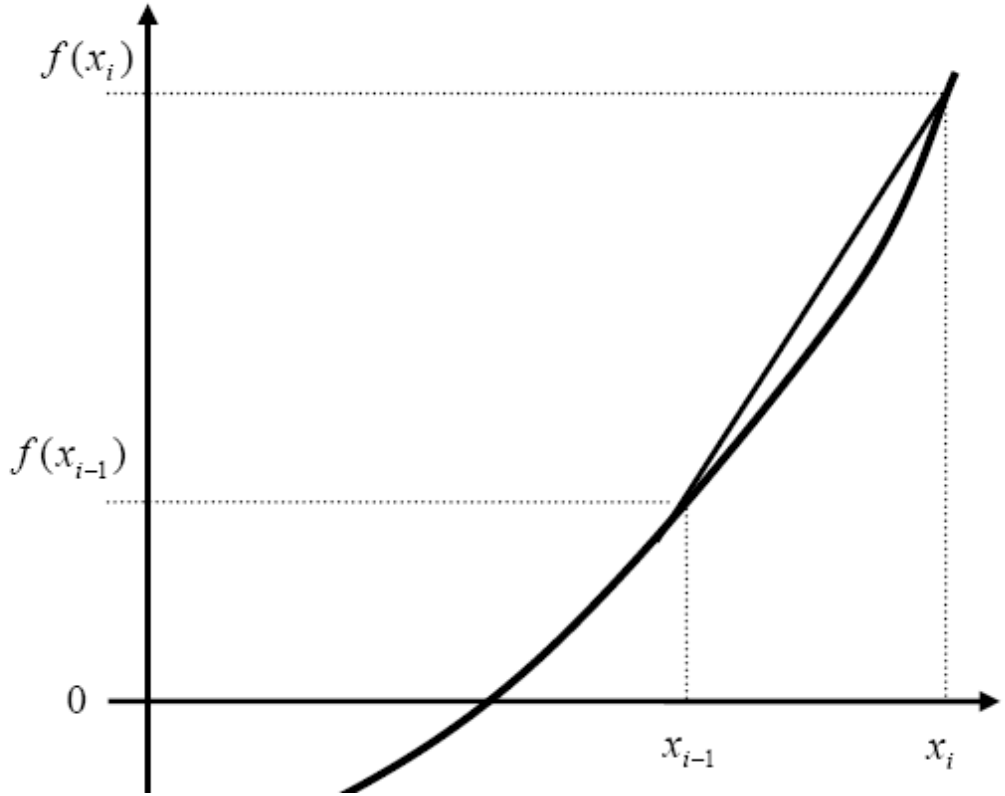
จะเห็นได้ว่าการ Converge ของ Newton-Ralphson เร็วกว่าวิธีก่อนมาก เพียงแค่ 2 Iteration ก็ดีกว่าวิธีของ Simple-One Point Iteration ที่ทำ 10 Iteration อย่างไรก็ตาม วิธีของ Newton-Ralphson นั้นมีข้อเสียที่ว่าจะให้ผลลัพธ์ที่แย่ในกรณีของ Multiple Root และแม้แต่ Simple Root บางครั้งก็มีปัญหา เช่นกรณีของ $f(x) = x^{10} - 1$ โดยเริ่มจาก $x_0 = 0.5$ จะพบว่า Converge ได้ช้ามาก และอัตราการ Converge จะขึ้นกับค่า x_0 ที่เลือก

นอกจากนี้แล้ว วิธีการนี้จะต้องใช้การหา Derivative ของ Function ซึ่งบางครั้งไม่สามารถหาได้ง่ายๆ การแก้ไขก็คือใช้วิธีการประมาณค่า Derivative ซึ่งเป็นวิธีการของ Secant Method ดังจะกล่าวในหัวข้อต่อไป

Open Method: Secant Method



จากที่กล่าวมาแล้ว ในการใช้วิธีของ Newton-Ralphson Method นั้น เราจะต้องหา Derivative ของ Function ซึ่งบาง Function จะหาค่า Derivative ได้ยากมาก ในกรณีเช่นนี้ เราอาจจะใช้วิธีการประมาณค่า Derivative ดังนี้(ดูรูป)



Open Method: Secant Method

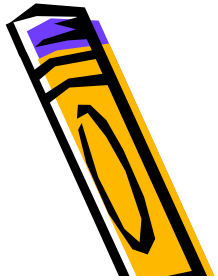
$$f'(x_i) \cong \frac{f(x_{i-1}) - f(x_i)}{x_{i-1} - x_i}$$

เมื่อนำสมการข้างบนไปแทนค่าในสมการของ Newton-Ralphson เราจะได้

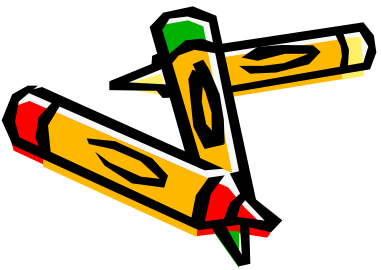
$$x_{i+1} = x_i - \frac{f(x_i)[x_{i-1} - x_i]}{f(x_{i-1}) - f(x_i)}$$ (Secant Method Formula)

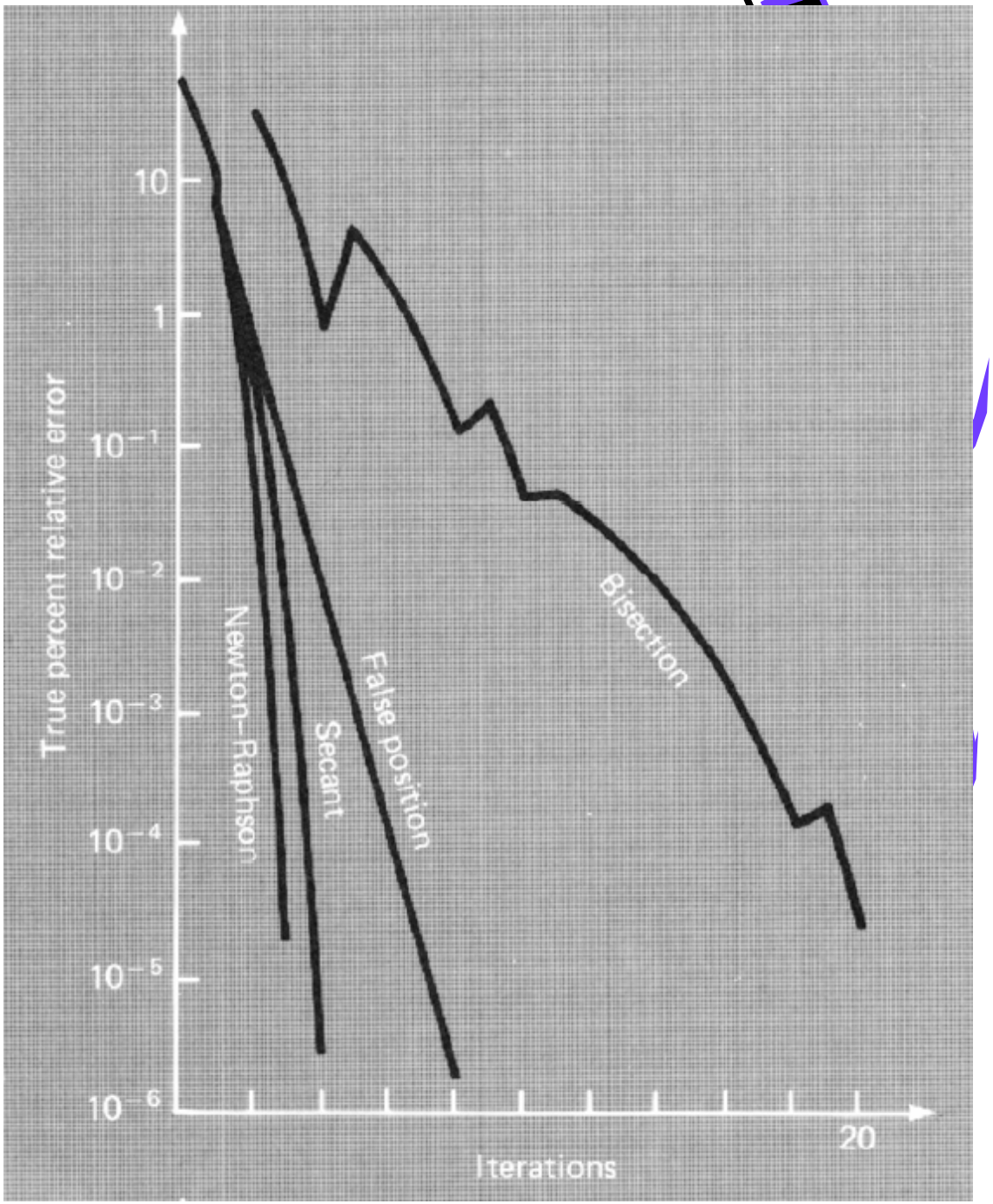
สังเกตว่าวิธีการนี้จำต้องใช้ค่า Estimate ของ x จำนวนสองค่า

Open Method: Secant Method



Secant Method อาจจะไม่ Converge ถ้าเราเลือกสองจุดที่ไม่เหมาะสม แต่ถ้ามัน Converge แล้ว มันจะ Converge ได้เร็วเกือบเท่าๆ Newton-Ralphson Method อย่างไรก็ตาม การ Convergence ขึ้นอยู่กับ Function และจุดเริ่มต้นที่เลือก รูปข้างล่างแสดงการเปรียบเทียบการ Convergence ของ $f(x) = e^{-x} - x$ (Simple One-Point Iteration เป็น Linear Convergence และจะ Converge ช้าสุด ไม่ได้แสดงไว้ ขอให้นักศึกษาลอง Plot เอง โดยใช้ข้อมูลจากตัวอย่างที่ 7.4)



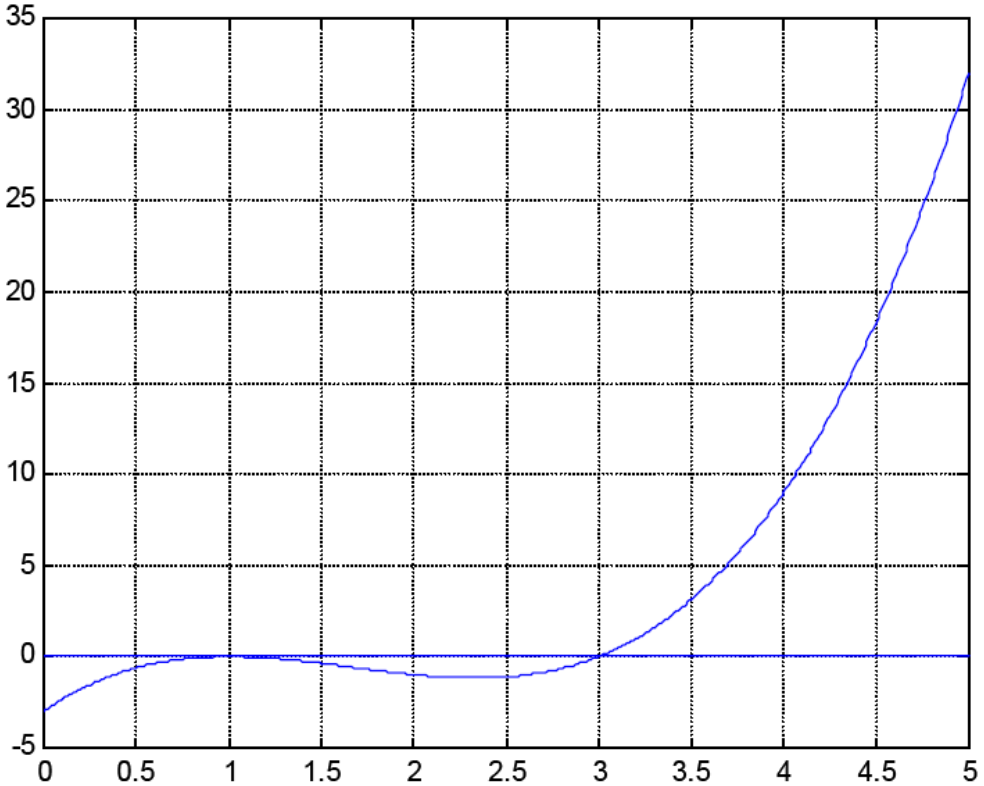

True percent relative error
10
1
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
10^{-6}
Newton–Raphson
Secant
False position
Bisection
20
Iterations

Multiple Roots

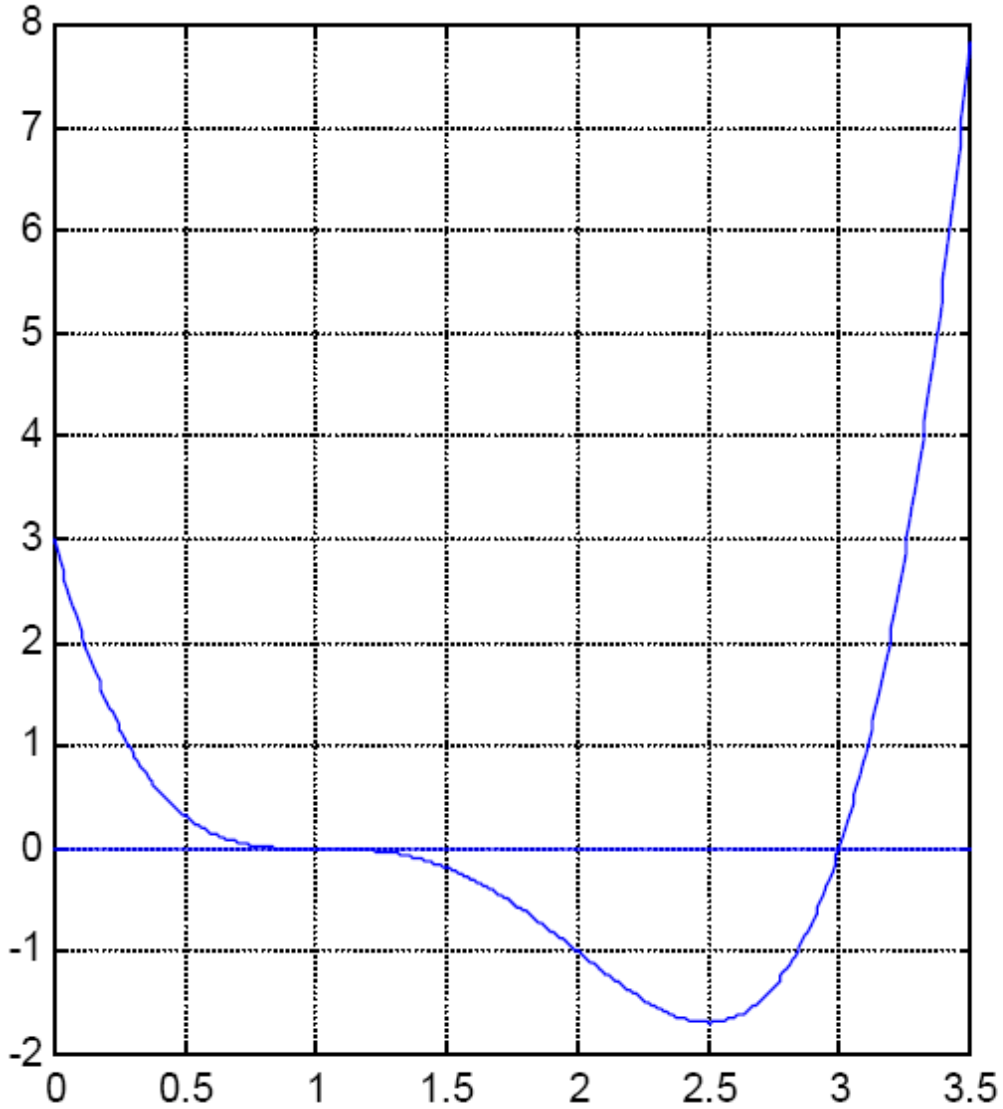
Multiple Root จะเป็นจุดที่ Function สัมผัสกับแกน x กล่าวคือค่า Slope จะเป็นศูนย์ ยกตัวอย่างเช่นสมการ

$$f(x) = x^3 - 5x^2 + 7x - 3 = (x-3)(x-1)(x-1)$$

ในกรณีเช่นนี้เรากล่าวว่า Function มี Double Root ที่ $x = 1$ ดังรูป



หรือในกรณีของ Triple Root เช่น $f(x) = x^4 - 6x^3 + 12x^2 - 10x + 3 = (x-3)(x-1)^3$ ดังรูป



Multiple Roots

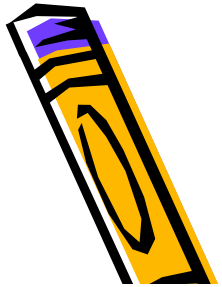
กรณีของ Multiple Root จะทำให้วิธีของ Numerical Method ที่กล่าวมาเกิดปัญหา เนื่องจาก

Function ไม่มีการเปลี่ยนเครื่องหมายที่จุดของ Root

1. ที่จุดนี้ นอกจาก $f(x)$ จะเท่ากับ ศูนย์แล้ว ค่า $f'(x)$ จะเท่ากับศูนย์ด้วย จะทำให้เกิดปัญหาใน Newton-Ralphson Method และ Secant Method อย่างไรก็ตาม $f(x)$ จะเข้าใกล้ศูนย์ก่อน $f'(x)$ เสมอและเราสามารถจะเพิ่มส่วนของโปรแกรมเพื่อจะตรวจสอบค่า $f(x)$ และหยุดโปรแกรมก่อนที่ค่า $f'(x)$ จะเป็นศูนย์ ซึ่งจะทำให้โปรแกรมเกิด “Divide by Zero Overflow”

ในกรณีของ Multiple Root วิธีของ Newton-Ralphson และ Secant Method จะมีการ Converge แบบ Linear แทนที่จะเป็น Quadratic วิธีการแก้มีหลายวิธี ที่แนะนำคือวิธีที่เสนอโดย Ralston and Rabinowitz(1978) โดยการให้นิยาม Function ใหม่ดังนี้

Multiple Roots



$$u(x) = \frac{f(x)}{f'(x)} \text{ และ } u'(x) = \frac{f'(x)f'(x) - f(x)f''(x)}{[f'(x)]^2}$$

และใช้สมการในการทำ Iteration เป็น

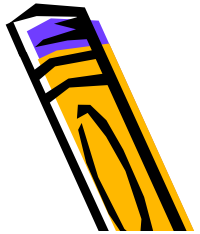
$$x_{i+1} = x_i - \frac{u(x_i)}{u'(x_i)}$$

ดังนั้นเราได้

$$x_{i+1} = x_i - \frac{f(x_i)f'(x_i)}{[f'(x_i]^2 - f(x_i)f''(x_i)} \qquad \text{(Modified Newton-Ralphson Method)}$$

ซึ่ง Algorithm ข้างบนจะเป็น Quadratic Convergence ทั้ง Simple Root และ Multiple Root แต่จะใช้การคำนวณมากกว่าวิธีการปกติของ Newton-Ralphson สำหรับแต่ละ Iteration

Multiple Roots



Example 7.7: เปรียบเทียบ Newton-Ralphson และ Modified Newton-Ralphson ในกรณีของการหา Root ของ

$$f(x) = x^3 - 5x^2 + 7x - 3$$

Solution:

$$f'(x) = 3x^2 - 10x + 7$$

$$f''(x) = 6x - 10$$

ดังนั้นเราได้ Newton-Ralphson: $x_{i+1} = x_i - \dfrac{x_i^3 - 5x_i^2 + 7x_i - 3}{3x_i^2 - 10x_i + 7}$

และสำหรับ Modified Newton-Ralphson Method เราได้

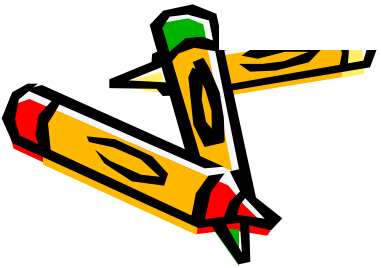
$$\text{Modified Newton-Ralphson: } x_{i+1} = x_i - \frac{(x_i^3 - 5x_i^2 + 7x_i - 3)(3x_i^2 - 10x_i + 7)}{(3x_i^2 - 10x_i + 7)^2 - (x_i^3 - 5x_i^2 + 7x_i - 3)(6x_i - 10)}$$

เมื่อ Run Iteration เราจะได้คำตอบดังตารางข้างล่าง ($x_0 = 0$)

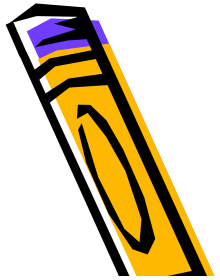
Multiple Roots

Normal Newton-Ralphson:

Iteration	x_i	$\lvert e_t \rvert$, %
0	0	100
1	0.428571429	57
2	0.685714286	31
3	0.832865400	17
4	0.913328983	8.7
5	0.955783293	4.4
6	0.977655101	2.2



Multiple Roots

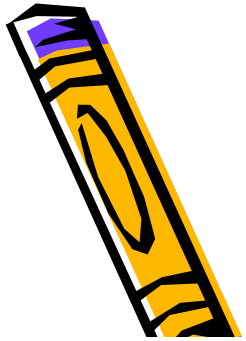


Modified Newton-Ralphson:

Iteration	x_i	$\lvert e_t \rvert$, %
0	0	100
1	1.105263158	22
2	1.003081664	0.31
3	1.000002382	0.00024

เช่นเดียวกัน เราสามารถปรับปรุงวิธีของ Secant Method ได้เช่นเดียวกัน โดยใช้การ Estimate ของ Function $u(x)$ และ $u'(x)$ แต่ในกรณีนี้จะไม่กล่าวถึง

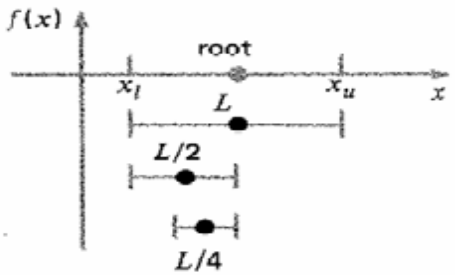
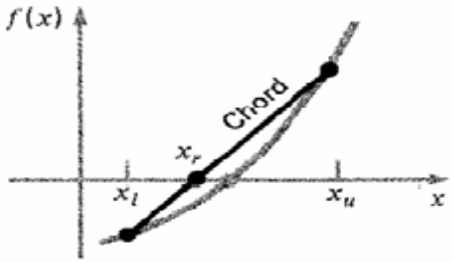
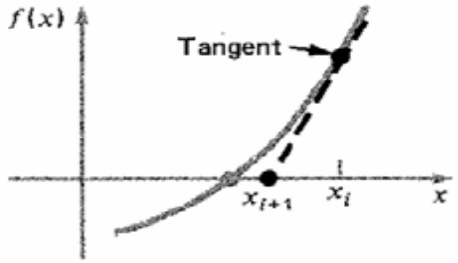
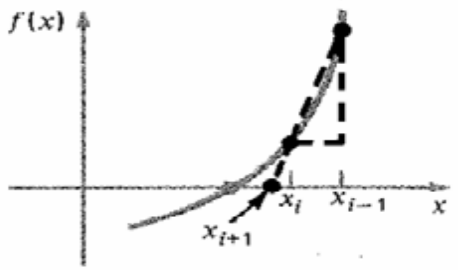
Comparison



การหา Root ของ Function เป็นเรื่องที่สำคัญสำหรับวิศวกรในการแก้ปัญหาทางคณิตศาสตร์ เนื่องจากการแก้ปัญหาแบบ Analytical Method ไม่สามารถกระทำได้ทุกกรณี เราจึงหันมาใช้วิธีทาง Numerical Method

การนำวิธีทาง Numerical Method จะต้องคำนึงถึง Error เป็นสำคัญ ทั้ง Truncation Error และ Round Off Error นอกจากนี้แล้วจะต้องคำนึงถึงการ Convergence ของ Algorithm ด้วย ดังนั้นการเลือกกรรมวิธีที่จะนำมาใช้จะเป็นสิ่งที่ควรจะพิจารณาเป็นอันดับแรก ตารางข้างล่างเป็นตารางสรุปของกรรมวิธี และข้อดีข้อเสียของแต่ละวิธี

Method	Initial Guest	Rate of Convergence	Stability	Accuracy	Breadth of Application	Program	Comments
Direct	-	-	-	-	Very Limited	-	
Graphical	-	-	-	Poor	General	-	May Take Much Time
Bisection	2	Slow	Always Converges	Good	General	Easy	
False Position	2	Medium	Always Converges	Good	General	Easy	
One-Point Iteration	1	Slow	May Not Converge	Good	General	Easy	
Newton-Ralphson	1	Fast	May Not Converge	Good	Limited if $f'(x) = 0$	Easy	Requires Evaluation of $f'(x)$
Modified Newton-Ralphson	1	Fast for Multiple Roots, Medium for Single Roots	May Not Converge	Good	Specifically Designed for Multiple Roots	Easy	Requires Evaluation of $f'(x)$ and $f''(x)$
Secant	2	Medium to Fast	May Not Converge	Good	General	Easy	Initial Guesses Do Not Have to Bracket The Roots

Method	Formulation	Graphical Interpretation	Errors and Stopping Criteria
Bracketing methods:			
Bisection	$x_r = \frac{x_l + x_u}{2}$ If $f(x_l)f(x_r) < 0$, $x_u = x_r$ $f(x_l)f(x_r) > 0$, $x_l = x_r$		Stopping criterion: $\left\|\frac{x_r^{new} - x_r^{old}}{x_r^{new}}\right\| 100\% \leq \epsilon$
False Position	$x_r = x_u - \frac{f(x_u)(x_l - x_u)}{f(x_l) - f(x_u)}$ If $f(x_l)f(x_r) < 0$, $x_u = x_r$ $f(x_l)f(x_r) > 0$, $x_l = x_r$		Stopping criterion: $\left\|\frac{x_r^{new} - x_r^{old}}{x_r^{new}}\right\| 100\% \leq \epsilon$
Open methods:			
Newton-Raphson	$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$		Stopping criterion: $\left\|\frac{x_{i+1} - x_i}{x_{i+1}}\right\| 100\% \leq \epsilon_s$ Error: $E_{i+1} = O(E_i^2)$
Secant	$x_{i+1} = x_i - \frac{f(x_i)(x_{i-1} - x_i)}{f(x_{i-1}) - f(x_i)}$		Stopping criterion: $\left\|\frac{x_{i+1} - x_i}{x_{i+1}}\right\| 100\% \leq \epsilon_s$

Homework 8

- นักศึกษาต้องเขียนโปรแกรมช่วยคำนวณ หรือใช้ Spreadsheet (MS Excel) ช่วยคำนวณ
  - แนะนำให้ใช้ MATLAB
    - เขียน Function หรือ Scratch File ก็ได้
    - หรือคำนวณจาก Workspace โดยตรง
- Download คำถามและตอบคำถาม