[CSE4152] 고급소프트웨어 실습 I

2020.09.28

서강대학교 공과대학 컴퓨터공학과
교수 임 인 성

본 강의에서 제작하여 제공하는 **PDF 파일, 동영상, 그리고 예제 코드 등의 강의 자료**의 저작권은 특별히 명기되어 있지 않은 한 서강대학교에 있습니다. 본인의 학습 목적 외에 공개된 장소에 올리거나 타인에게 배포하는 등의 행위를 금합니다. 협조 부탁합니다.

실험 개요

- **이산공간(discrete space)**과 **연속공간(continuous space)**
  - 지금까지 배운 컴퓨터공학 주제들은 주로 **이산공간(discrete space)에서 정의**된 문제들임.
  - 한편 현실 세계에서 부딪치는 대부분의 문제들은 **연속공간(continuous space)에서 정의**가 됨.
  - 후자의 효과적인 해결을 위하여 **다양한 형태의 수치 알고리즘(numerical algorithm)의 이해와 활용 능력**이 필요함.
- **실수 연산**과 **부동소수점 연산**(floating-point operation)
  - 연속공간의 기본 수체계인 **실수(real number)**를 컴퓨터는 **부동 소수점 숫자(floating-point number)**를 통하여 **근사적으로 표현**하여 연산을 수행함.
  - 우리가 머리속으로 생각하는 **수학 계산과 실제로 컴퓨터가 수행하는 연산에는 상당한 괴리가 발생**할 수 있다는 사실에 대처할 능력을 필요로 함.
- **General-Purpose GPU Computing(GPGPU)**
  - 대부분의 수치 계산은 방대한 양의 계산을 요구하는데, 자주 CUDA/OpenCL 등의 툴 기반의 GPU 컴퓨팅을 통하여 손쉽게 가속할 수 있음.

The Patriot Missile Failure

- On February 25, 1991, during the Gulf War, an American Patriot Missile battery in Dharan, Saudi Arabia, **failed to track and intercept an incoming Iraqi Scud missile**. The Scud struck an American Army barracks, killing 28 soldiers and injuring around 100 other people. … It turns out that **the cause was an inaccurate calculation of the time since boot due to computer arithmetic errors**.

- Specifically, the time in tenths of second as measured by the system's internal clock **was multiplied by 1/10** to produce the time in seconds. This calculation was performed using **a 24 bit fixed point register**. In particular, **the value 1/10, which has a non-terminating binary expansion**, was chopped at 24 bits after the radix point.
- **The small chopping error, when multiplied by the large number giving the time in tenths of a second, led to a significant error**.

By courtesy of D. N. Arnold
(http://www.ima.umn.edu/~arnold/disasters/patriot.html)

- Indeed, the Patriot battery had been **up around 100 hours**, and an easy calculation shows that **the resulting time error due to the magnified chopping error was about 0.34 seconds**.
  - The binary approximation to 0.1
    - $0.1 = 0.0001100110011001100110011001100_2 = 209715/2097152$
  - The roundoff error
    - `1/10 - 209715/2097152 = 1/10485760` (about 0.0001%)
  - Integral values of its internal clock were converted to decimal by multiplying the binary approximation to 0.1
  - After 100 hours, the error becomes
    - `(1/10 - 209715/2097152)(100*60*60*10) = 5625/16384` (about 0.3433 second)

- A **Scud** travels at about 1,676 meters per second, and so **travels more than half a kilometer in this time**. This was far enough that the incoming Scud was outside the "range gate" that the Patriot tracked.
- **Ironically, the fact that the bad time calculation had been improved in some parts of the code, but not all, contributed to the problem, since it meant that the inaccuracies did not cancel.**

Additional Examples of Roundoff Errors

- An egregious example of roundoff error is provided by **a short-lived index devised at the Vancouver stock exchange** (McCullough and Vinod 1999). At its inception in 1982, the index was given a value of 1000.000. **After 22 months of recomputing the index and truncating to three decimal places at each change in market value**, the index stood at **524.881**, despite the fact that its "true" value should have been **1009.811**.
- Other sorts of roundoff error can also occur. A notorious example is the fate of the **Ariane rocket** launched on June 4, 1996 (European Space Agency 1996). In the 37th second of flight, the inertial reference system **attempted to convert a 64-bit floating-point number to a 16-bit number**, but instead triggered **an overflow error** which was interpreted by the guidance system as flight data, **causing the rocket to veer off course and be destroyed**.

By courtesy of Wolfram MathWorld (http://mathworld.wolfram.com/RoundoffError.html)

Why Numerical Computing?

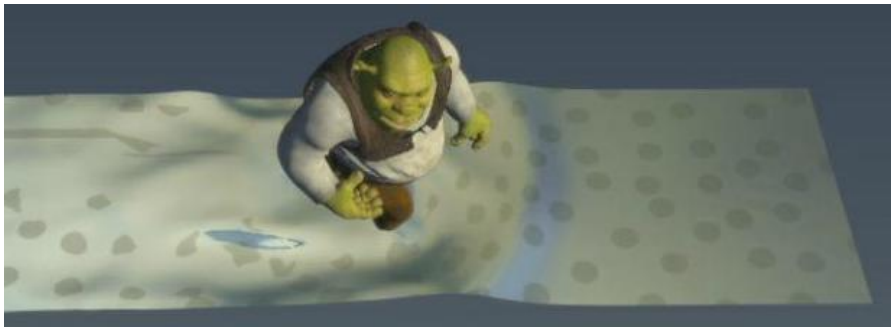
- **Camera tracking for AR/VR, 3D Graphics and Vision**

Lenovo Phab 2 Pro smartphone

- To implement the presented camera tracking method, you need to solve a **nonlinear minimization problem**, which is a variation of Iterative Closest Point(ICP).

$$\mathbf{E}(\mathrm{T}_{g,k}) = \sum_{\substack{\mathbf{u}\in\mathscr{U} \\ \Omega_k(\mathbf{u})\neq\text{null}}} \left\| \left(\mathrm{T}_{g,k}\dot{\mathbf{V}}_k(\mathbf{u}) - \hat{\mathbf{V}}^g_{k-1}(\hat{\mathbf{u}}) \right)^\top \hat{\mathbf{N}}^g_{k-1}(\hat{\mathbf{u}}) \right\|_2$$

Shrek and Differential Equations



Conservation of momentum

$$\frac{\partial u}{\partial \mathrm{t}} = \nu\nabla\cdot(\nabla u) - (u\cdot\nabla)u - \frac{1}{\rho}\nabla \mathbf{p} + g$$

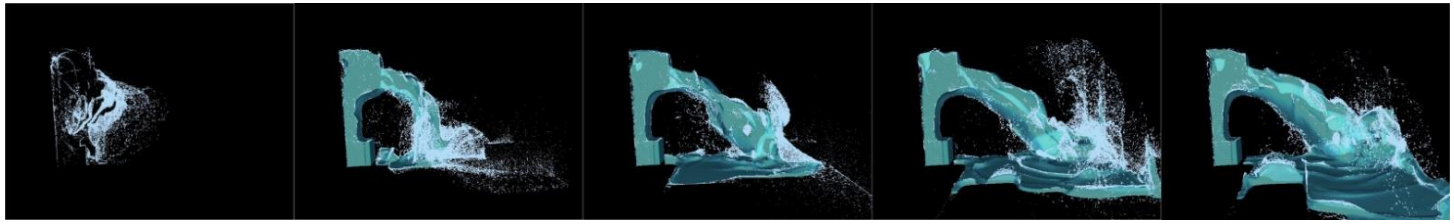
viscosity convection pressure

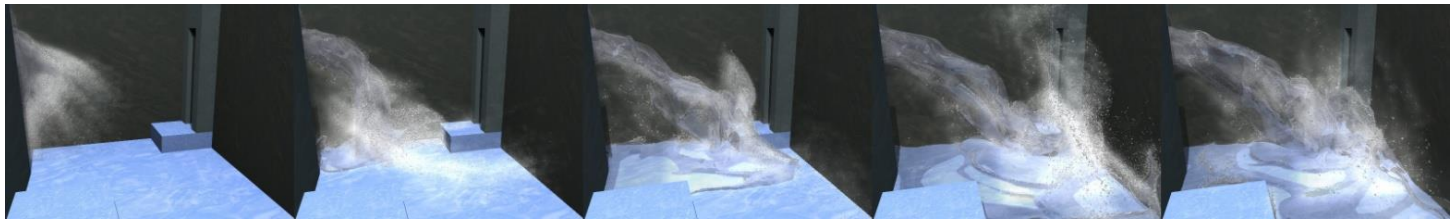
Conservation of mass

$$\nabla\cdot u = 0 \quad \nabla = \{\partial/\partial x,\ \partial/\partial y,\ \partial/\partial z\}$$

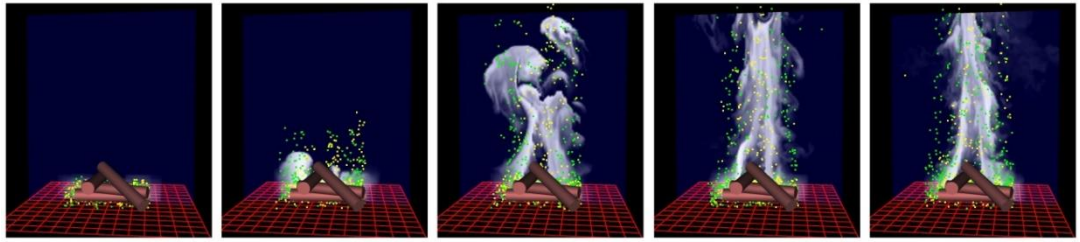
Physically Based Fluid Animation

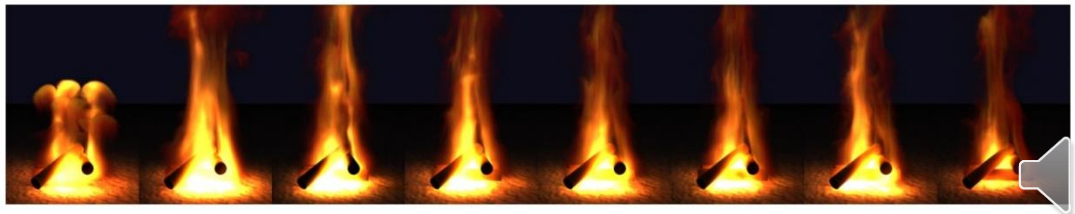
J. Kim, D. Cha, B. Chang, B. Koo, I. Ihm, "Practical Animation of Turbulent Splashing Water", *ACM SIGGRAPH/Eurographics Symp. on Computer Animation*, 2006.





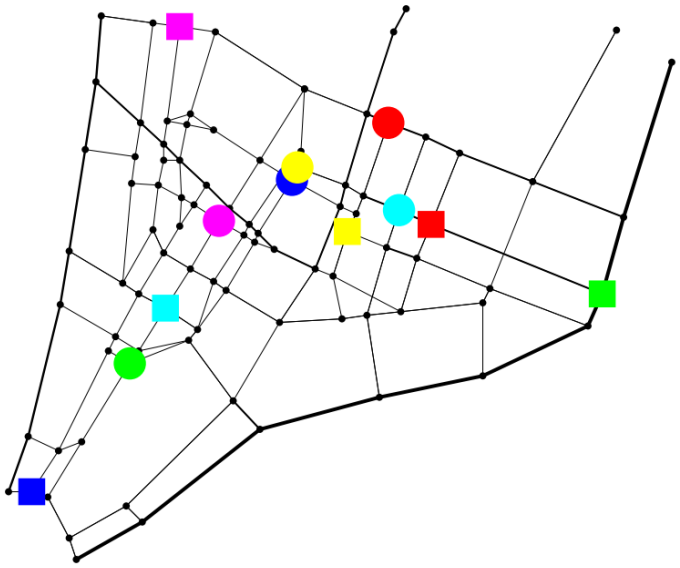
B. Kang, Y. Jang, and I. Ihm, "Animation of Chemically Reactive Fluids Using a Hybrid Simulation Method", *ACM SIGGRAPH/Eurographics Symp. on Computer Animation*, 2007.





Autonomous Vehicles and Minimization

Routing autonomous vehicles in congested transportation networks (F. Rossi et al.)



$$\underset{f_m(\cdot,\cdot),f_R(\cdot,\cdot)}{\text{minimize}} \sum_{m\in\mathcal{M}} \sum_{(u,v)\in\mathcal{E}} t(u,v) f_m(u,v)$$

$$+\rho \sum_{(u,v)\in\mathcal{E}} t(u,v) f_R(u,v) \tag{1a}$$

$$\text{subject to} \quad \sum_{u\in\mathcal{V}} f_m(u,s_m) + \lambda_m = \sum_{w\in\mathcal{V}} f_m(s_m,w) \quad \forall m \in \mathcal{M} \tag{1b}$$

$$\sum_{u\in\mathcal{V}} f_m(u,t_m) = \lambda_m + \sum_{w\in\mathcal{V}} f_m(t_m,w) \quad \forall m \in \mathcal{M} \tag{1c}$$

$$\sum_{u\in\mathcal{V}} f_m(u,v) = \sum_{w\in\mathcal{V}} f_m(v,w)$$

$$\forall m \in \mathcal{M}, v \in \mathcal{V} \setminus \{s_m, t_m\} \tag{1d}$$

$$\sum_{u\in\mathcal{V}} f_R(u,v) + \sum_{m\in\mathcal{M}} 1_{v=t_m}\lambda_m$$

$$= \sum_{w\in\mathcal{V}} f_R(v,w) + \sum_{m\in\mathcal{M}} 1_{v=s_m}\lambda_m \quad \forall v \in \mathcal{V} \tag{1e}$$

$$f_R(u,v) + \sum_{m\in\mathcal{M}} f_m(u,v) \le c(u,v) \quad \forall (u,v) \in \mathcal{E} \tag{1f}$$

Data Mining and Matrix Computation

- Problem: **Search books about *Baking Bread.***

The $t = 6$ **terms**:

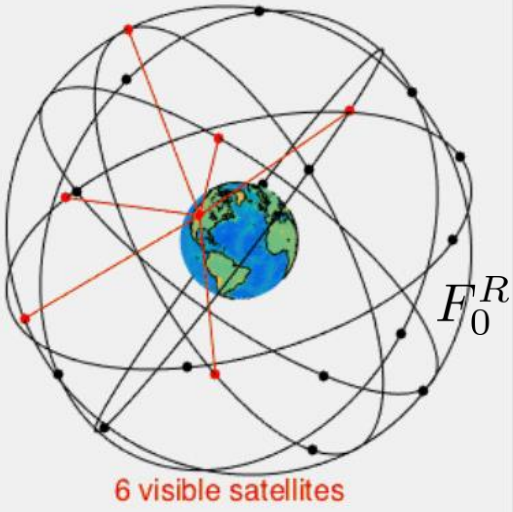
T1: bak(e,ing)
T2: recipes
T3: bread
T4: cake
T5: pastr(y,ies)
T6: pie

$$A_k^t q = (V_k D_k)(U_k^t q)$$

The $d = 5$ **document** titles:

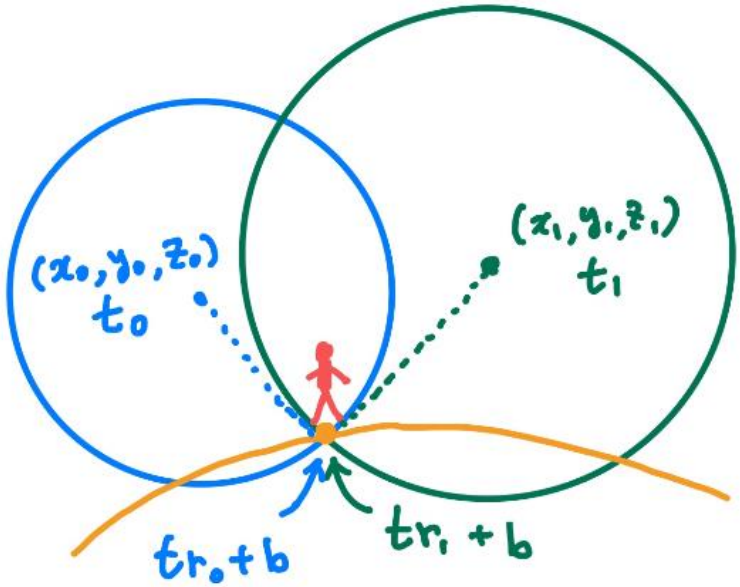
D1: How to Bake Bread Without Recipes
D2: The Classic Art of Viennese Pastry
D3: Numerical Recipes: The Art of Scientific Computing
D4: Breads, Pastries, Pies and Cakes: Quantity Baking Recipes
D5: Pastry: A Book of Best French Recipes

GPS 수신기 위치 계산 문제



A visual example of a 24 satellite GPS constellation in motion with the earth rotating. Notice how the number of *satellites in view* from a given point on the earth's surface, in this example in Golden CO (39.7469° N, 105.2108° W), changes with time.

Wikipedia



$$
\begin{aligned}
(x-x_0)^2 + (y-y_0)^2 + (z-z_0)^2 &= \{C(tr_0 + b - t_0)\}^2 \\
(x-x_1)^2 + (y-y_1)^2 + (z-z_1)^2 &= \{C(tr_1 + b - t_1)\}^2 \\
(x-x_2)^2 + (y-y_2)^2 + (z-z_2)^2 &= \{C(tr_2 + b - t_2)\}^2 \\
(x-x_3)^2 + (y-y_3)^2 + (z-z_3)^2 &= \{C(tr_3 + b - t_3)\}^2
\end{aligned}
$$

System of Nonlinear Equations

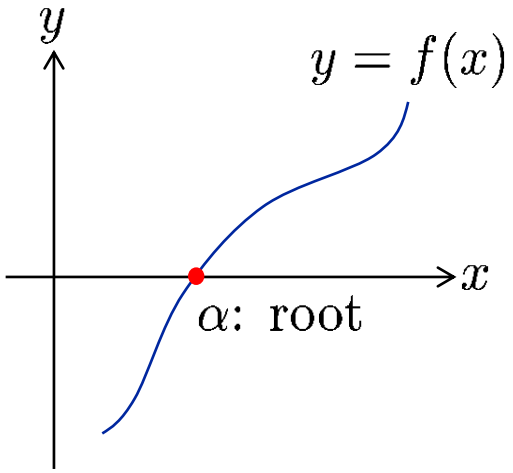
$$
\begin{aligned}
(x-x_0)^2+(y-y_0)^2+(z-z_0)^2 &= \{\,C(tr_0+b-t_0)\,\}^2 \\
(x-x_1)^2+(y-y_1)^2+(z-z_1)^2 &= \{\,C(tr_1+b-t_1)\,\}^2 \\
(x-x_2)^2+(y-y_2)^2+(z-z_2)^2 &= \{\,C(tr_2+b-t_2)\,\}^2 \\
(x-x_3)^2+(y-y_3)^2+(z-z_3)^2 &= \{\,C(tr_3+b-t_3)\,\}^2
\end{aligned}
$$

$$
\begin{aligned}
f_0(x,y,z,b) &= (x-x_0)^2+(y-y_0)^2+(z-z_0)^2-\{\,C(tr_0+b-t_0)\,\}^2 = 0 \\
f_1(x,y,z,b) &= (x-x_1)^2+(y-y_1)^2+(z-z_1)^2-\{\,C(tr_1+b-t_1)\,\}^2 = 0 \\
f_2(x,y,z,b) &= (x-x_2)^2+(y-y_2)^2+(z-z_2)^2-\{\,C(tr_2+b-t_2)\,\}^2 = 0 \\
f_3(x,y,z,b) &= (x-x_3)^2+(y-y_3)^2+(z-z_3)^2-\{\,C(tr_3+b-t_3)\,\}^2 = 0
\end{aligned}
$$

Root Finding of Nonlinear Equations

- **Root finding problem**

Given a function $f : R \rightarrow R$, find the value(s) of α for which $f(\alpha) = 0$.



- A user needs to be able to evaluate f (and possibly f') for given x.
- Each evaluation may be very expensive.
- A user may need some prior information on the root.

2 비선형 방정식의 풀이 방법

앞절에서 제시한 방정식의 풀이 방법을 이해하기 전에 좀 더 쉬운 형태의 방정식의 풀이 문제에 대하여 알아보자.

> 임의의 함수 $f : \mathbb{R} \to \mathbb{R}$에 대해, $f(\alpha) = 0$을 만족하는 값 α를 구하라.

여기서 실수 공간 R에 대해 정의된 함수 $f(x)$는 다음과 같이 일반적인 형태를 가지는 비선형 함수 (nonlinear equation)이며,

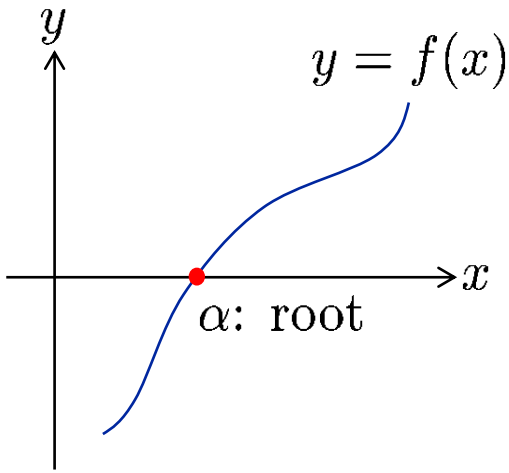
$$
\begin{aligned}
f(x) &= x^2 - 3x - 2 = 0 \\
f(x) &= x^5 + 2x^2 - 7x + 5 = 0 \\
f(x) &= x - a\sin x - b = 0 \\
f(x) &= e^x - e^{-x} - 3x = 0
\end{aligned}
$$

위에서 $x = \alpha$와 같이 함수 값을 0으로 만들어 주는 값을 방정식의 근 (root)이라 한다 (그림 2 참조). 또한 방정식 $f(x) = 0$을 푸는 것을 근 찾기 (root finding)라고 한다.

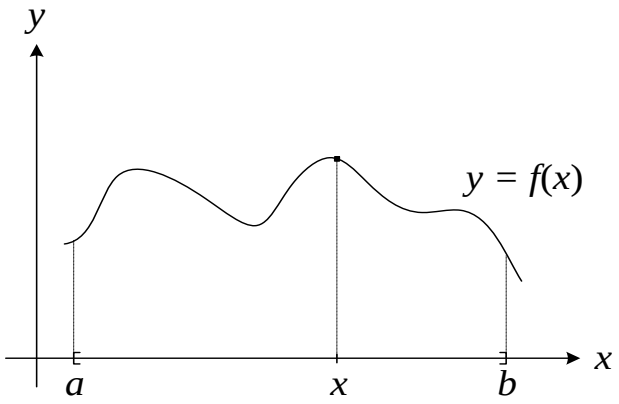
Given a function $f : R \rightarrow R$, find the value(s) of α for which $f(\alpha) = 0$.

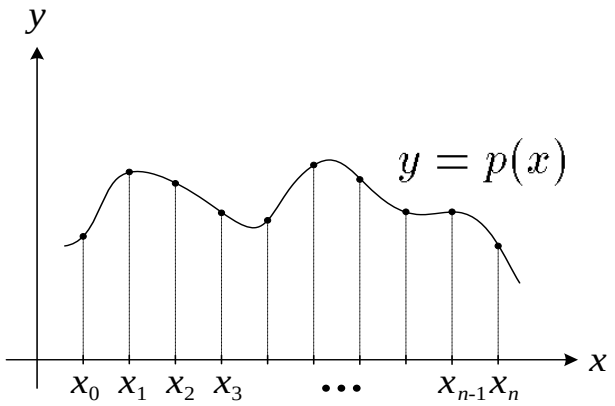
- **Mathematical functions at school**

$$
\begin{aligned}
f_1(x) &= x^2 - 3x - 2 = 0 \\
f_2(x) &= x^5 + 2x^2 - 7x + 5 = 0 \\
f_3(x) &= x - a\sin x - b = 0 \\
f_4(x) &= e^x - e^{-x} - 3x = 0
\end{aligned}
$$



- **Mathematical functions in real life**





x	x_0	x_1	x_2	$\cdots$	x_{n-1}	x_n
y	y_0	y_1	y_2	$\cdots$	y_{n-1}	y_n

- **Iterative methods**

  Given an initial guess x_0, find a sequence $x_0, x_1, x_2, x_3, \cdots$
  such that $\lim_{n\to\infty} x_n = \alpha$ where $f(\alpha) = 0$.

- **Rates of convergence**

  1. $\{x_n\}_{n=0}^{\infty}$ converges to α, at least, *linearly* if $|x_n - \alpha| \le \epsilon_n$, where ϵ_n is a positive sequence such that $\lim_{n\to\infty} \frac{\epsilon_{n+1}}{\epsilon_n} = c$, $0 < c < 1$.
  2. $\{x_n\}_{n=0}^{\infty}$ converges to α, *with an order of*, at least, p (≥ 1), if $|x_n - \alpha| \le \epsilon_n$, where ϵ_n is a positive sequence such that $\lim_{n\to\infty} \frac{\epsilon_{n+1}}{\epsilon_n^p} = c$, $c > 0$.

  (If $p = 1$, $0 < c < 1$.)

  Example: quadratic convergence $(p = 2,\ c = 10)$

$$
\begin{aligned}
\epsilon_0 &\approx 0.01 = 10^{-2} \\
\epsilon_1 &\approx 10\epsilon_0^2 = 10^{-3} \\
\epsilon_2 &\approx 10\epsilon_1^2 = 10^{-5} \\
\epsilon_3 &\approx 10\epsilon_2^2 = 10^{-9} \\
&\vdots
\end{aligned}
$$

Newton-Rapson Method

Assumption

$f(x) \in C^2$

Idea

α : root of $f(x) = 0$

x : α에 대한 근사값 $\Rightarrow \alpha = x + h$ for some h (h를 알경우 ...)

Taylor's series of f about x

$$0 = f(\alpha) = f(x+h) = f(x) + \frac{f'(x)}{1!}(x+h-x) + o(h^2)$$

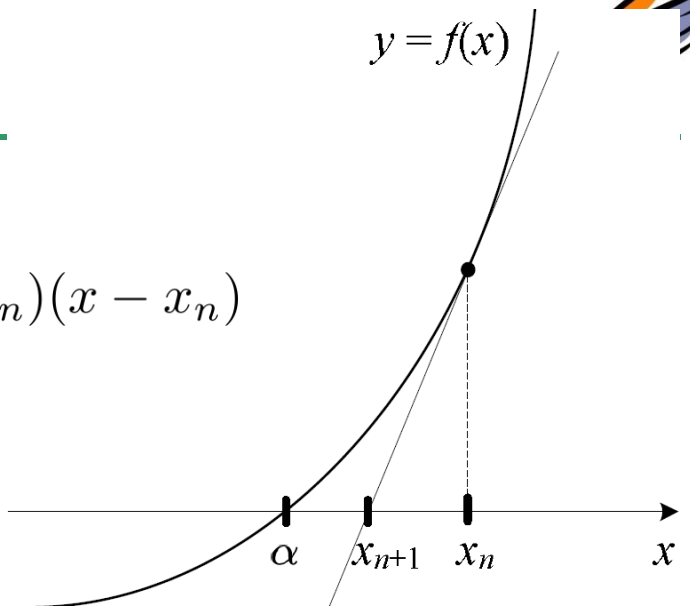
$$\Rightarrow 0 = f(x+h) = f(x) + h \cdot f'(x) + o(h^2)$$

$$\approx f(x) + h \cdot f'(x)$$

$$\Rightarrow h \approx -\frac{f(x)}{f'(x)}$$

$$\alpha = x + h \approx \overbrace{x - \frac{f(x)}{f'(x)}}^{x^*}$$

※ x^* is a better approximation to α !



$$y = f(x_n) + f'(x_n)(x - x_n)$$

Newton Iteration

For an initial guess x_0,

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)},\ n = 0, 1, 2, \ldots$$

When to stop: three stopping criteria

(a) 현재 구한 x_{n+1} 에 대해 함수 값이 충분히 작은가? ⟶ $|f(x_{n+1})| < \delta$ [1)]

(b) 충분히 많은 회수만큼 반복문을 수행하였는가? ⟶ $n \geq N_{max}$ [2)]

(c) 현재 구한 x_{n+1} 이 직전에 구한 x_n 에 비해 더 이상 의미 있는 진전을 하지 않는가? ⟶ $|x_{n+1} - x_n| < \varepsilon$ [3)]

$f(x) = 0$ $\quad x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)},\ n = 0, 1, 2, \ldots$

- **Example**

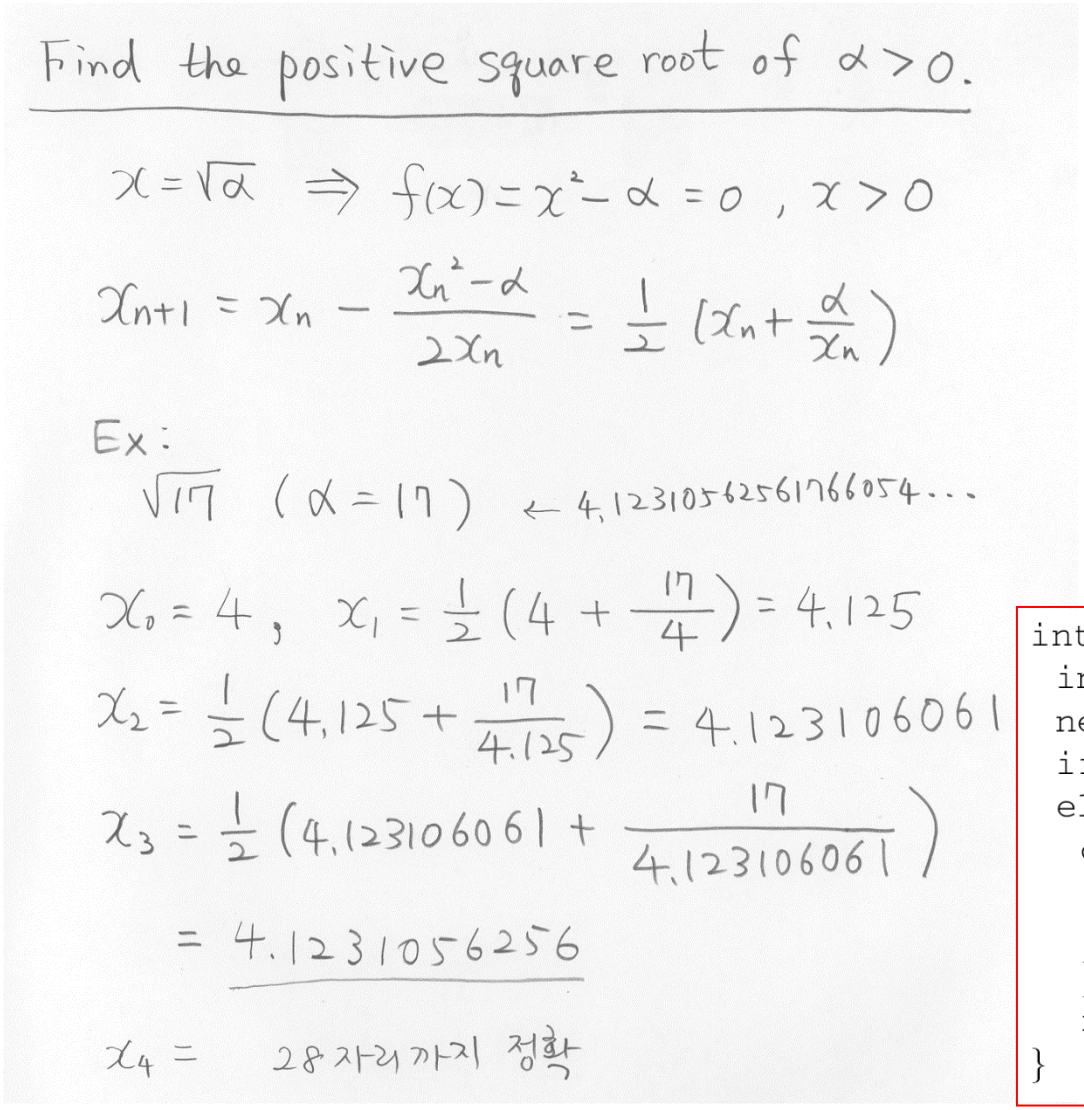
$$f(x) = x^3 - 2x^2 + x - 3$$

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} = x_n - \frac{x_n^3 - 2x_n^2 + x_n - 3}{3x_n^2 - 4x_n + 1}$$

$$x_0 = 3$$

$$x_{root} = 2.1745594102929799$$

```
C:₩Windows₩system32₩cmd.exe
  i              xn1                 |f(xn1)|     |xn1-xroot|
  0   3.0000000000000000e+000   9.0000e+000   8.2544e-001
  1   2.4375000000000000e+000   2.0369e+000   2.6294e-001
  2   2.2130327163151096e+000   2.5636e-001   3.8473e-002
  3   2.1755549387214881e+000   6.4634e-003   9.9553e-004
  4   2.1745601006664459e+000   4.4791e-006   6.9037e-007
  5   2.1745594102933126e+000   2.1574e-012   3.3262e-013
계속하려면 아무 키나 누르십시오 . . .
```




```
int find_integer_square_root(int n) {
 int next, current;
 next = n/2;
 if (n <= 1) return 1;
 else {
  do {
    current = next;
    next = (next + (A) )/2;
   } while (next < current);
   return current;
}
```

Secant Method

Newton 방법의 문제점

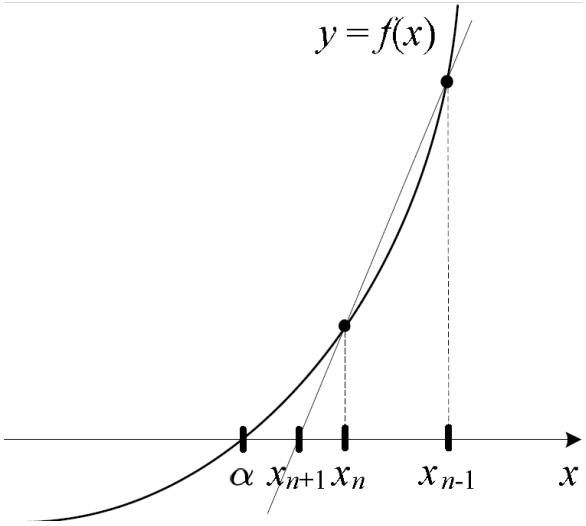
$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \longrightarrow$ 1차 미분값 $f'(x_n)$ 값이 필요함.

아이디어

$f'(x_n)$을 $\frac{f(x_n)-f(x_{n-1})}{x_n-x_{n-1}}$으로 근사함.

방법

$x_{n+1} = x_n - f(x_n)\frac{x_n-x_{n-1}}{f(x_n)-f(x_{n-1})}$, $n = 0, 1, 2, \ldots$



Convergence Rates

$$f(x) = 0: \quad x_0, x_1, x_2, x_3, \cdots \longrightarrow x = \alpha$$

Error: $\epsilon_n = |x_n - \alpha|$

- **Newton-Raphson method**

$$\epsilon_{n+1} \approx c\epsilon_n^2$$

$$\begin{aligned}
\varepsilon_0 &\approx 0.01 = 10^{-2} \\
\varepsilon_1 &\approx 10\varepsilon_0^2 = 10^{-3} \\
\varepsilon_2 &\approx 10\varepsilon_1^2 = 10^{-5} \\
\varepsilon_3 &\approx 10\varepsilon_2^2 = 10^{-9} \\
\varepsilon_4 &\approx 10\varepsilon_3^2 = 10^{-17} \\
&\vdots
\end{aligned}$$

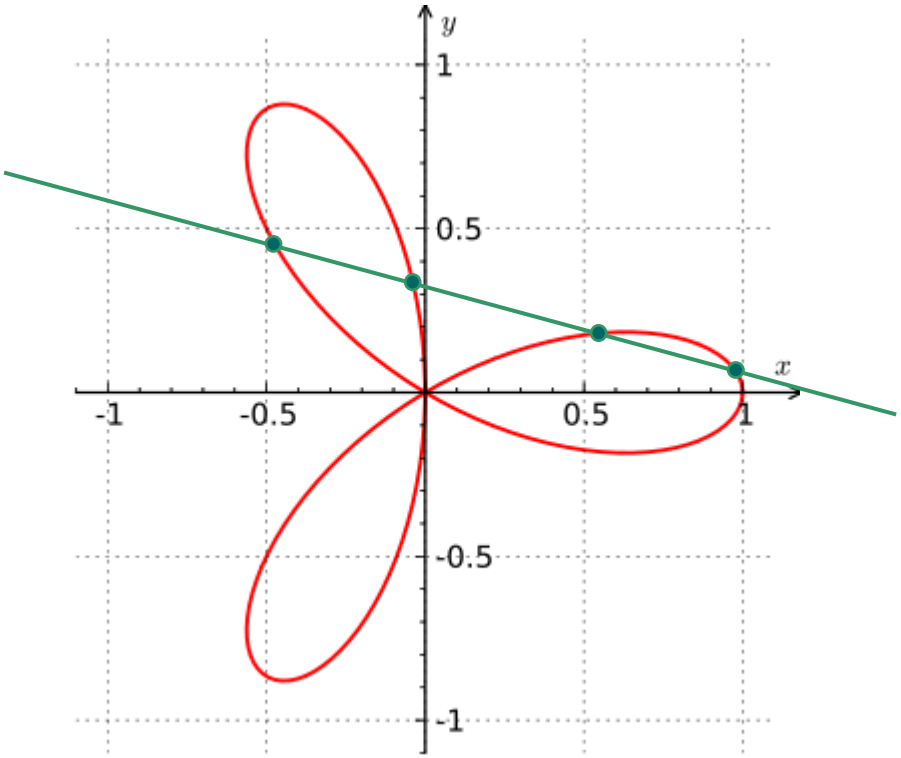
- **Secant method**

$$\epsilon_{n+1} \approx c\epsilon_n^{1.62}$$

$$\begin{aligned}
\varepsilon_0 &\approx 0.01 = 10^{-2} \\
\varepsilon_1 &\approx 10\varepsilon_0^{1.62} = 5.76 \cdot 10^{-3} \\
\varepsilon_2 &\approx 10\varepsilon_1^{1.62} = 2.35 \cdot 10^{-3} \\
\varepsilon_3 &\approx 10\varepsilon_2^{1.62} = 5.51 \cdot 10^{-4} \\
\varepsilon_4 &\approx 10\varepsilon_3^{1.62} = 5.53 \cdot 10^{-5} \\
\varepsilon_5 &\approx 10\varepsilon_4^{1.62} = 1.17 \cdot 10^{-6} \\
\varepsilon_6 &\approx 10\varepsilon_5^{1.62} = 2.46 \cdot 10^{-9} \\
&\vdots
\end{aligned}$$

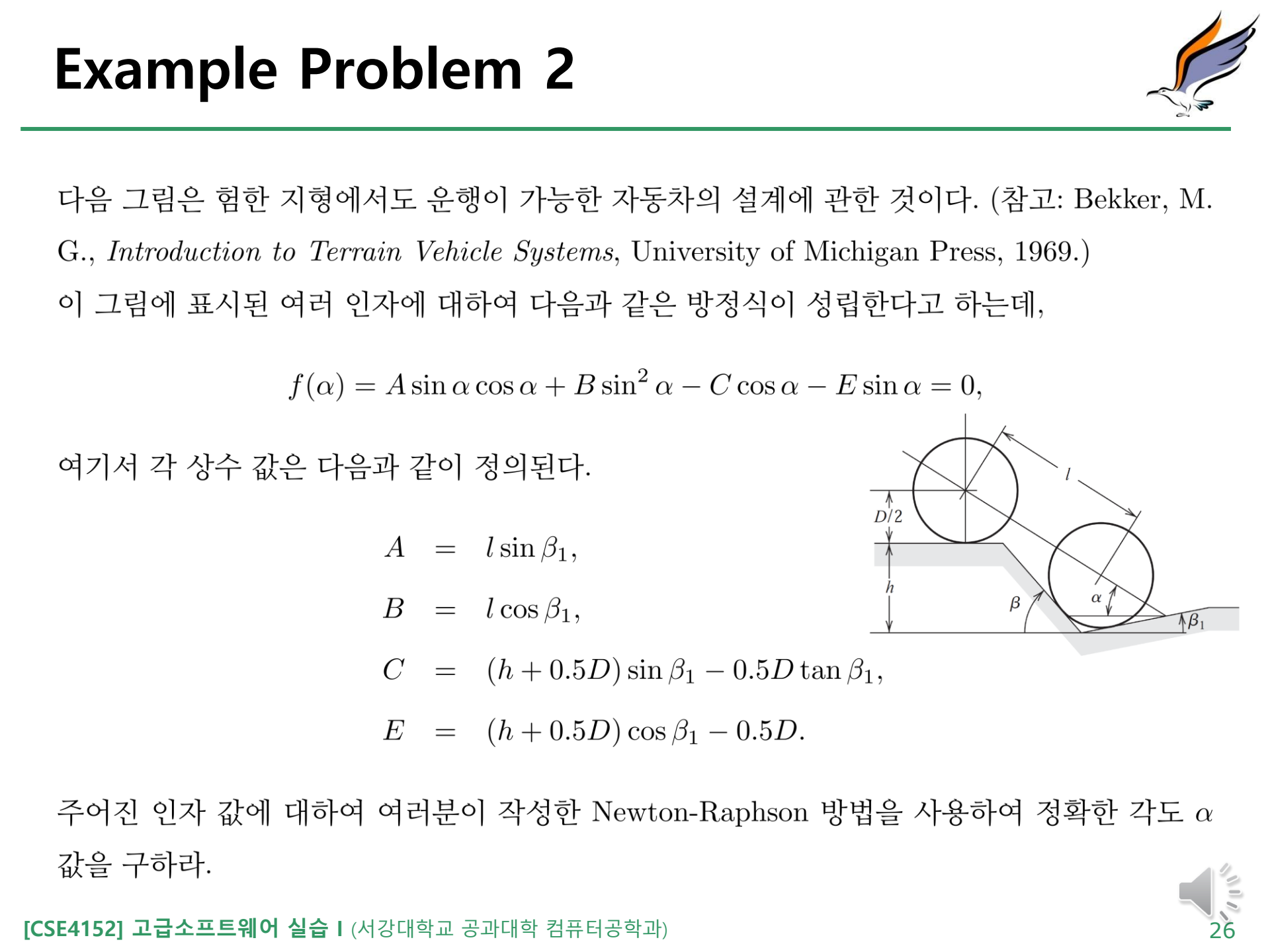
Example Problem 1

Find all intersections (x, y) of an arbitrary line $y = ax + b$ and the curve $g(x, y) = x^4 + 2x^2y^2 + y^4 - x^3 + 3xy^2 = 0$.



$$\longrightarrow f(x) = g(x, ax + b) = 0$$

Example Problem 2



다음 그림은 험한 지형에서도 운행이 가능한 자동차의 설계에 관한 것이다. (참고: Bekker, M. G., *Introduction to Terrain Vehicle Systems*, University of Michigan Press, 1969.)
이 그림에 표시된 여러 인자에 대하여 다음과 같은 방정식이 성립한다고 하는데,

$$f(\alpha) = A\sin\alpha\cos\alpha + B\sin^2\alpha - C\cos\alpha - E\sin\alpha = 0,$$

여기서 각 상수 값은 다음과 같이 정의된다.

$$\begin{aligned} A &= l\sin\beta_1, \\ B &= l\cos\beta_1, \\ C &= (h+0.5D)\sin\beta_1 - 0.5D\tan\beta_1, \\ E &= (h+0.5D)\cos\beta_1 - 0.5D. \end{aligned}$$

주어진 인자 값에 대하여 여러분이 작성한 Newton-Raphson 방법을 사용하여 정확한 각도 α 값을 구하라.