



HANBAT NATIONAL UNIVERSITY

COURSE: DESIGN OF COASTAL ENGINEERING

REPORT III

NOVEMBER 29, 2022

Author

Student ID

Jonghyeok Kim

20201967

Report III

Jonghyeok Kim

November 29, 2022

Contents

1	Introduction	3
2	Questions	6
2.1	Question #0	7
2.2	Answer #0	7
2.3	Question #1	9
2.4	Answer #1	10
2.5	Question #2	18
2.6	Answer #2	18
2.7	Question #3	23
2.8	Answer #3	23
3	Conclusion	29
3.1	Conclusion	29
4	Acknowledgement	33
5	References	33
A	Question #0	34
B	Question #1	34
C	Question #2	43
D	Question #3	49

1 Introduction

기본적으로 아래의 분산관계식(Dispersion Relationship)을 이용하여 본 보고서에 수록된 문제들을 해결할 수 있다.

$$\sigma^2 = gktanh(kh) \quad (1.1)$$

만약 kh 의 값이 조건으로 제시가 되었다면, 위의 분산 관계식을 이용하여 σ^2 으로부터 σ 값을 산정하고, 아래의 각진동수 σ 에 대한 관계식으로부터,

$$\sigma = \frac{2\pi}{T} \quad (1.2)$$

다음과 같이 주기 T 에 대한 식으로 정리하여 주기 T 를 산정해낼 수 있다.

$$T = \frac{2\pi}{\sigma} \quad (1.3)$$

또한, 주어진 kh 의 값을 이용하여 아래의 파수 k 에 대한 관계식으로부터,

$$k = \frac{2\pi}{L} \quad (1.4)$$

다음과 같이 파장 L 에 대한 식으로 정리하여 파장 L 을 산정해낼 수 있다.

$$L = \frac{2\pi}{k} \quad (1.5)$$

반면에, kh 가 제시되어 있지 않거나, 파수 k 의 값을 모르는 경우에는 위의 분산관계식에 대해 양해법(Explicit Method)과 음해법(Implicit Method)을 이용하여 k 값을 산정할 수 있다.

본 보고서에서 필자는 음해법으로서 이분법(Bisection Method)을 이용하였고, Boundary Points를 결정함에 있어서 양해법 중 하나인 아래의 Eckart 식을 이용하였다.

$$\sigma^2 = gk\sqrt{\tanh(\frac{\sigma^2}{g}h)} \quad (1.6)$$

반드시는 아니나, 분산관계식에서 파수 k 를 구함에 있어 음해법(Implicit Method)을 사용하는 경우에는 양해법

(Explicit Method)을 이용하여 구한 k 값을 초깃값으로 하는 경우가 일반적이다.

또한, 위의 양해법에 대한 Eckart 식 외에도 다음과 같은 Hunt의 식도 존재한다.

$$(kh)^2 = y^2 + \frac{y}{1 + \sum_{n=1}^6 d_n y^n} \quad (1.7)$$

where,

$$y = \frac{\sigma^2 h}{g}$$

$d_1 = 0.6666666666, d_2 = 0.3555555555, d_3 = 0.1608465608, d_4 = 0.0632098765, d_5 = 0.0217540484, d_6 = 0.0076507983$

특히, You(2003)에 따르면, 여러 개의 파랑분산식의 양해(Hunt, 1979; Nielsen, 1982; Fenton과 McKee, 1990; Guo, 2002; Nielsen, 2002; You, 2003)를 비교하였을 때, Hunt의 해가 $\nu \leq \pi$ 의 수심범위에서 가장 정확하다는 것을 밝혀 내었다. 이러한 사실로부터, 초깃값 산정의 방법에 있어서 Eckart 식 외에 Hunt의 식 사용을 고려해보는 것도 음해법을 이용한 해의 수렴속도를 증가시키는 한 가지 요인으로 작용할 수 있을 것으로 기대해볼 수 있겠다.

물론 양해법에는 Eckart의 방법과 Hunt의 방법 외에도 여러 개의 다른 양해법이 존재한다.

본 과제에서는 위의 기본적인 해안공학에서의 개념에 기반하여, 규칙파의 변형 중 파의 굴절에 대해서 살펴볼 예정이다.

파의 굴절은 파 속도의 차이로 발생하고, 기본적으로 Snell의 법칙을 이용하여 일정 수심(해당 구역에서는 주어진 수심으로 일정하다는 의미)에서의 그 굴절 각도를 산정할 수 있으며, 다음과 같이 유도한다.

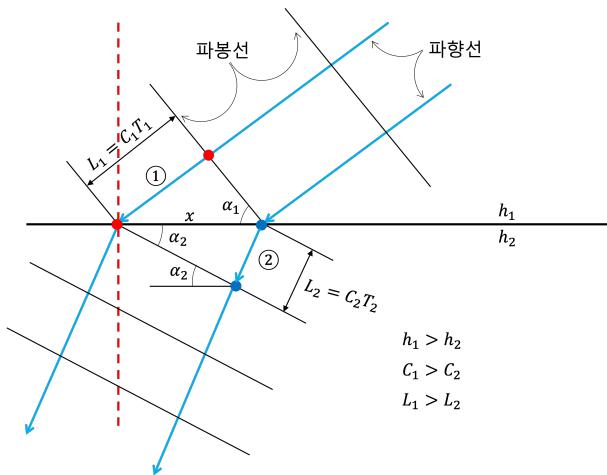


Figure 1: Wave Refraction

파향선과 파봉선이 이루는 각도가 90° 라는 사실과 위의 Figure 1 상에서 도시되어있듯, 파가 화살표 방향으로 ① 구간을 통과하는 데 걸리는 시간 T_1 과 파가 화살표 방향으로 ② 구간을 통과하는 데 걸리는 시간 T_2 가 T 로 서로 같다는 사실($T_1 = T_2 = T$)로부터 Snell의 법칙을 유도할 수 있다.

삼각함수의 정의로부터 다음과 같은 관계식을 유도할 수 있다.

$$\sin(\alpha_1) = \frac{L_1}{x} \quad (1.8)$$

$$\sin(\alpha_2) = \frac{L_2}{x} \quad (1.9)$$

또한, 파 속도 C 는 다음과 같이 정의된다.

$$C = \frac{L}{T} \quad (1.10)$$

(1.10) 식으로부터 (1.8)과 (1.9)의 식을 다음과 같이 작성할 수 있다.

$$\sin(\alpha_1) = \frac{L_1}{x} = \frac{C_1 \cdot T_1}{x} \quad (1.11)$$

$$\sin(\alpha_2) = \frac{L_2}{x} = \frac{C_2 \cdot T_2}{x} \quad (1.12)$$

그런데, $T_1 = T_2 = T$ 이므로, 위의 식 (1.11)과 식 (1.12)은 다시 다음과 같이 작성할 수 있다.

$$\sin(\alpha_1) = \frac{L_1}{x} = \frac{C_1 \cdot T_1}{x} = \frac{C_1 \cdot T}{x} \quad (1.13)$$

$$\sin(\alpha_2) = \frac{L_2}{x} = \frac{C_2 \cdot T_2}{x} = \frac{C_2 \cdot T}{x} \quad (1.14)$$

위의 식 (1.13)과 식 (1.14)을 $\sin(\alpha_1)$ 에 대한 $\sin(\alpha_2)$ 에 대한 식으로 정리하면 다음과 같다.

$$\frac{\sin(\alpha_2)}{\sin(\alpha_1)} = \frac{\frac{C_2 \cdot T}{x}}{\frac{C_1 \cdot T}{x}} = \frac{C_2}{C_1} \quad (1.15)$$

다음으로, 위의 식 (1.15)을 $\sin(\alpha_2)$ 에 대한 식으로 정리한다.

$$\sin(\alpha_2) = \sin(\alpha_1) \cdot \frac{C_2}{C_1} \quad (1.16)$$

마지막으로, 양변에 \sin^{-1} 을 취하면 α_2 에 대한 식으로 정리할 수 있다.

$$\alpha_2 = \sin^{-1}(\sin(\alpha_1) \cdot \frac{C_2}{C_1}) \quad (1.17)$$

결론적으로, 파속 C_1, C_2 를 알고, 파랑의 굴절각 α_1 을 알 때, 식 (1.17)을 이용하여 미지의 굴절각 α_2 를 산정할 수 있다.

2 Questions

수심 $h_0 = 10m$ 에서 $kh = \pi$ 를 만족하는 파가 입사하고 있다.

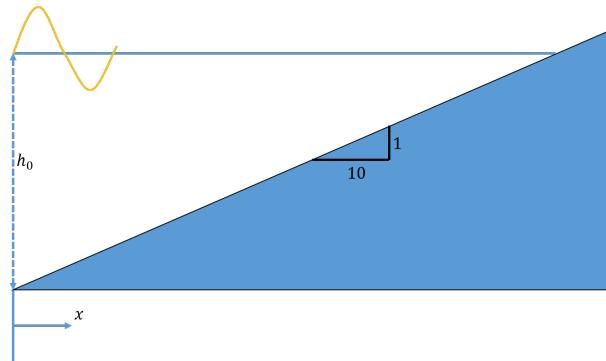


Figure 2: Condition #1

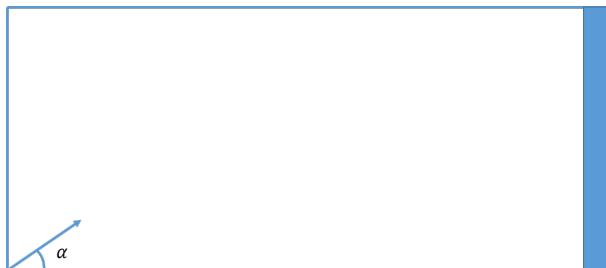


Figure 3: Condition #2

2.1 Question #0

수심 h_0 에서 $kh = \pi$ 를 만족하는 주기 T_0 와 파장 L_0 를 구하시오. 또한, 음해법(Implicit Method)과 양해법(Explicit Method)인 Eckart's Method와 Hunt's Method로 산정한 파수 k 와 파속 C 를 서로 비교하시오.

2.2 Answer #0

먼저 먼 바다에서 파가 진행하면 해당 파에 대해서는 항상 주기가 일정하므로, 식 (1.17)의 Snell의 법칙을 활용하여 파의 파랑의 굴절각 α_2 를 산정함에 있어 변수 1개를 줄일 수 있다.

이러한 주기 T 의 산정을 위해 Questions 섹션에서 주어진 수심 $h_0 = 10m$ 과 $kh = \pi$ 를 이용하여 k 값을 아래와 같이 산정한다.

$$kh = k \times 10m = \pi \quad (2.1)$$

$$\therefore k = \frac{\pi}{10m} = 0.3141592653589793(\text{rad}/\text{m}) \quad (2.2)$$

위에서 도출한 k 값으로부터, 분산관계식 (1.1)을 이용하여 다음과 같이 각진동수 σ^2 을 산정한다.

$$\sigma^2 = 9.81m/\text{sec}^2 \times \frac{\pi}{10m} \times \tanh\left(\frac{\pi}{10m} \times 10m\right) \quad (2.3)$$

$$\therefore \sigma^2 = 3.0704133(\text{rad}^2/\text{sec}^2) \quad (2.4)$$

위의 식 (2.3)로부터 도출한 σ^2 의 값으로부터, 각진동수 σ 와 주기 T 의 관계식 (1.2)에서 주기 T 를 산정하기 위해 (2.4)의 값에 제곱근을 취하여 σ 의 값을 다음과 같이 산정한다.

$$\sigma = \sqrt{\sigma^2} = \sqrt{3.0704133(\text{rad}^2/\text{sec}^2)} = 1.7522594830545946(\text{rad/sec}) \quad (2.5)$$

위의 (2.5)에서 구한 σ 값을 이용하여 각진동수 σ 와 주기 T 의 관계식 (1.3)을 통해 주기 T 를 다음과 같이 산정한다.

$$T = T_0 = \frac{2\pi}{\sigma} = \frac{2\pi}{1.7522594830545946(\text{rad/sec})} \approx 3.58(\text{sec}) \quad (2.6)$$

또한, Question #0 섹션에서 주어진 $kh = \pi$ 라는 조건으로부터 위의 (2.2)에서 구한 k 값을 이용하여 파수 k 와 파장 L 의 관계식 (1.5)을 통해 파장 L 을 다음과 같이 산정한다.

$$L = L_0 = \frac{2\pi}{k} = \frac{2\pi}{0.3141592653589793(\text{rad}/\text{m})} = 20.0(\text{m}) \quad (2.7)$$

다음으로, 음해법(Implicit Method)인 이분법(Bisection Method)과 양해법(Explicit Method)인 Eckart와 Hunt의 방법으로부터 파수 k 를 산정하고, 이들의 그래프 비교 및 차이에 대한 평균 백분율을 분석한 다음, 각 방법에서 구한 파수 k 로부터 파속 C 역시 서로 비교하고자 한다.

먼저 파수 k 를 수심 $h_0 = 10\text{m}$ 일 때의 파수를 포함하여, 수심 $h_1, h_2, h_3, \dots, h_{99}$ 까지에 대해 Eckart 식 (1.6)과 Hunt 식 (1.7)으로부터 구한 다음, 위에서 이분법에 의해 도출한 파수 k 를 수심 h 가 $[0, 1]$ 의 범위에 대해 그래프로 비교해보면 다음과 같다.

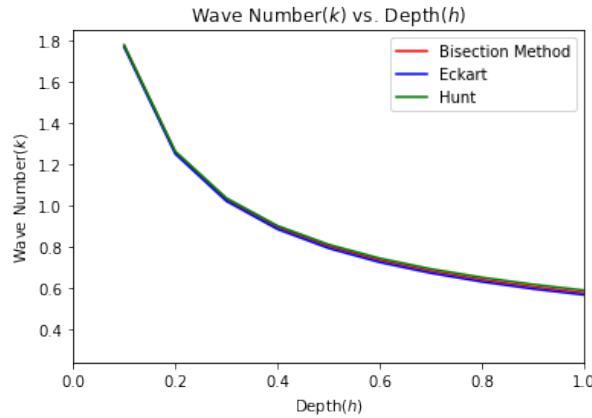


Figure 4: Graph of Wave Number(k) vs. Depth(h)

Figure 4에서 Bisection Method와 Hunt's Method의 파수 k 는 큰 차이를 보이고 있지 않지만, Eckart's Method의 파수 k 는 상대적으로 큰 차이를 보이면서 나머지 두 방법에 의해 산정한 수심 h 에 대한 파수 k 와는 다르게 불연속적으로 직선의 개형을 보임도 확인할 수 있다.

Bisection Method로 산정한 파수 k 에 대해 나머지 두 방법에 의해 산정한 파수 k 와의 차이의 평균의 백분율 값을 비교하여 표로 정리하면 다음과 같다.

Table 2.2.1 Bisection Method vs. Eckart Method vs. Hunt's Method for k

	k (Using Eckart's Method)	k (Using Hunt's Method)
k (Using Bisection Method)	0.64951202%	0.37930637%

Bisection Method로 산정한 파수 k 를 기준으로 이 값과의 차이의 평균의 백분율 값으로서 Eckart's Method

에서는 약 0.65%로 약 0.379%인 Hunt's Method에 비해 상대적으로 큰 차이를 보이고 있음을 확인해볼 수 있다.

또한, 각 방법에서 도출해낸 파수 k 값으로부터 식 (2.17)을 이용하여 파속 C 를 산정하여 이를 수심 h 에 대해 그래프를 작도하여 서로 비교해보면 다음과 같다.

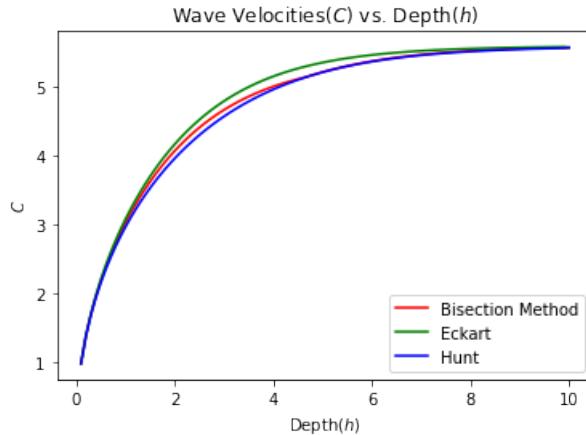


Figure 5: Graph of Wave Speed(C) vs. Depth(h)

Figure 5에서 Bisection Method로 산정한 C 와 Eckart's Method와 Hunt's Method에서 산정한 C 가 어느정도 차이가 발생하고 있음을 그래프상으로 확인해볼 수 있다.

마찬가지로, Bisection Method로 산정한 파속 C 에 대해 나머지 두 방법에 의해 산정한 파속 C 와의 차이의 평균의 백분율 값을 비교하여 표로 정리하면 다음과 같다.

Table 2.2.2 Bisection Method vs. Eckart Method vs. Hunt's Method for C

	C (Using Eckart's Method)	C (Using Hunt's Method)
C (Using Bisection Method)	7.72149906%	3.43992334%

Bisection Method로 산정한 파속 C 값을 기준으로 이 값과의 차이의 평균의 백분율 값을 통해서 Eckart's Method에서는 약 7.72%로 상당히 큰 차이를 보여주고 있음을 확인할 수 있고, Hunt's Method에서도 3.44%로 Eckart's Method에 비해서는 작은 차이이지만, Bisection Method로 산정한 파속 C 값과 어느정도 차이가 발생하고 있음을 확인해볼 수 있다.

2.3 Question #1

입사파의 각도가 30° 인 경우에 대하여 파의 진행경로를 예측하시오.

2.4 Answer #1

먼저 (1.17) 식의 C_1 을 앞서 Question #0 섹션에서 산정한 파수 k 와 각진동수 σ 를 다음의 파속 C 에 대한 식에 대입하여 C_1 을 산정한다.

$$C = C_1 = \frac{L}{T} = \frac{\frac{2\pi}{k}}{\frac{2\pi}{\sigma}} = \frac{\sigma}{k} \quad (2.8)$$

$$C_1 = \frac{1.7522594830545946}{0.3141592653589793} \quad (2.9)$$

$$\therefore C_1 = 5.577615166155759(m/sec) \quad (2.10)$$

입사파의 각도가 30° 이므로, 앞서 Introduction 섹션에서 살펴본 파의 파랑의 굴절각을 계산하는 식 (1.17)의 $\alpha_1 = 30^\circ$ 임을 알 수 있다.

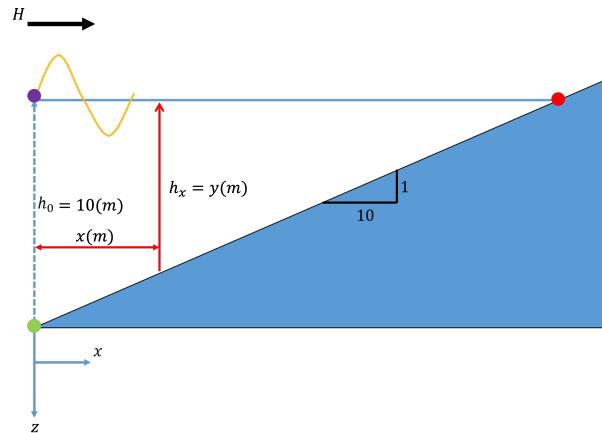


Figure 6: Condition #1-1

다음으로, 위 Figure 6에서 도시한 보라색 점의 좌표를 $(0, 0)$ 으로 설정하였을 때, 연두색 점의 좌표는 $(0, 10)$ 이 되고, 기울기는 $-\frac{1}{10}$ 이므로, 이로부터 직선의 방정식을 다음과 같이 유도하여 임의로 원점에서 x 만큼 이동하였을 때, 원점에서 해당 지점까지 이동한 거리에서의 수심 $y = h_x$ 를 산정한다.

$$y = h_x = -\frac{1}{10}(x - 0) + 10 \quad (2.11)$$

유도한 직선의 방정식 (2.11)에서 $y = h_x = 0m$ 일 때, 즉 수심이 $0m$ 일 때의 원점에서 이동한 거리 x 를 계산하면

다음과 같다.

$$y = 0 = -\frac{1}{10}(x) + 10 \quad (2.12)$$

$$\therefore x = 100m \quad (2.13)$$

그러므로, $x = 0m$ 에서 $x = 99m$ 까지 $x_0, x_1, x_2, \dots, x_{99}$ 각각에 대해 총 100개의 수심 $h_0, h_1, h_2, \dots, h_{99}$ 을 산정한다.

산정한 각 수심에 대해 음해법(Implicit Method)인 이분법(Bisection Method)을 이용하여 미지의 파수 k 값을 산정하기 위해, 그 초기값을 양해법(Explicit Method) 중 하나인 Eckart 식 (1.6)을 이용하여 산정한다.

예를 들어, $x_1 = 1m$ 에서 수심 h_1 의 값은 $9.9m$ 이고, 면 바다에서 진행되어 온 동일 파에 대해서는 주기 T 가 동일하므로, 앞서 Question #0 섹션에서 산정한 T_0 와 그 값이 동일하다. 즉, $T = T_0$ 이다. 또한, Eckart 식의 σ^2 값 역시 동일 주기이므로, (1.2) 관계식으로부터 이것 역시 앞서 Question #0 섹션에서 산정한 각진동수 σ 의 제곱한 값 $\sigma^2 = 3.0704133(rad^2/sec^2)$ 과 동일하다. 마지막으로, 중력 가속도 $g = 9.81m/sec^2$ 이라 하면, Eckart 식 (1.6)을 k 에 대한 식으로 정리하여, 위의 값을 대입하여 초기 k 값을 산정해낼 수 있다.

$$k = \frac{\sigma^2}{g \sqrt{\tanh(\frac{\sigma^2}{g} h)}} = \frac{3.0704133}{9.81 \sqrt{\tanh(\frac{3.0704133}{9.81} 9.9)}} \quad (2.14)$$

$$\therefore k = 0.31362574(rad/m) \quad (2.15)$$

그때의 k 의 초기값은 $0.31362574(rad/m)$ 로 도출된다.

산정한 초기값을 이용하여, 음해법(Implicit Method) 중 하나인 이분법(Bisection Method)을 통해 컴퓨터를 이용하여 수치해를 구해보자.

이를 위해, 우선 분산관계식(Dispersion Relationship) (1.1)의 좌변이 0이 되도록 모든 항을 이항하여 정리하면 다음과 같다.

$$0 = \frac{gk \tanh(kh)}{\sigma^2} - 1 \quad (2.16)$$

위의 식 (2.16)에 대해 오차(Error)를 0.5×10^{-6} 으로 설정하여, 위에서 산정한 (2.14)의 초기 k 값의 ± 0.1 만큼을

이분법의 해 탐색 구간으로 설정하여 k 값을 산정한다.

예를 들어, 위에서와 같이 수심 $h = 9.9m$ 에 대해 이분법(Bisection Method)을 이용하여 k 값을 산정하면 그 값은 $k = 0.31423335(rad/m)$ 으로, 초기값과의 백분율 차이는 약 0.06% 수준이다.

다음의 파속 C 식에 수치해석 기법인 이분법(Bisection Method)으로 도출한 k 값과 Question #0에서 산정한 σ 값을 대입하여 (1.17)의 파속 C_2 를 산정한다.

$$C = C_2 = \frac{L}{T} = \frac{\frac{2\pi}{k}}{\frac{2\pi}{\sigma}} = \frac{\sigma}{k} \quad (2.17)$$

$$C_2 = \frac{1.7522594830545946}{0.31423335} \quad (2.18)$$

$$\therefore C_2 = 5.57630018(m/sec) \quad (2.19)$$

산정한 파속 C_2 의 값과 C_1 의 값 그리고 입사파의 각도 α_1 의 값을 앞서 Introduction 섹션에서 유도한 Snell의 법칙에 대한 식 (1.17)에 대입함으로써, 임의지점에서의 파의 굴절각도 α_2 를 산정한다.

$$\alpha_2 = \sin^{-1}(\sin(\alpha_1)) \cdot \frac{C_2}{C_1} = \sin^{-1}(\sin(30^\circ)) \cdot \frac{5.57630018(m/sec)}{5.577615166155759(m/sec)} \quad (2.20)$$

$$\therefore \alpha_2 = 0.52346266(rad) \approx 29.992^\circ \quad (2.21)$$

위의 원리를 그대로 적용하여 나머지 수심 $h_2, h_3, h_4, \dots, h_{99}$ 에서의 파랑의 굴절각 α_2 를 구한다.

다음으로 진행하기에 앞서, 주어진 조건에 대한 Figure 3 상을 기준으로 하는 파랑의 굴절각 α 에 대한 그래프를 산정하기 위해 해당 조건에 대한 Figure 3에 기준축과 원점을 설정해야한다.

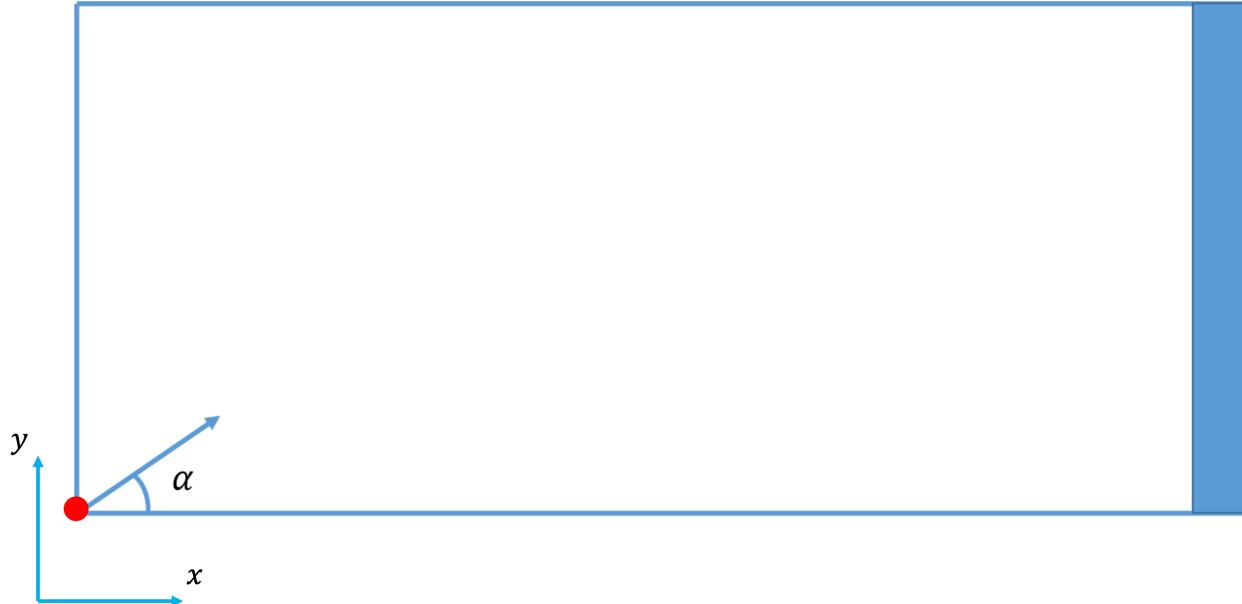


Figure 7: Modified Condition #2-1

위 Figure 7을 통해 축의 방향과 각 축이 무엇을 의미하는지 확인할 수 있다.

즉, 축의 방향은 x 축의 경우 오른쪽으로 진행할 수록 증가하고, y 축의 경우 위쪽으로 진행할 수록 증가한다.

또한, Figure 7에서 y 축은 연직방향으로의 이동거리(Distance)이고, x 축의 경우 수평방향으로의 이동거리(Distance)를 나타낸다.

추가적으로, Figure 7에서 확인할 수 있듯, 좌표가 원점 $(0, 0)$ 인 위치를 빨간점으로 도시하였다.

마지막으로, 수심 h 를 Figure 7에 도시해보면, 다음과 같다.

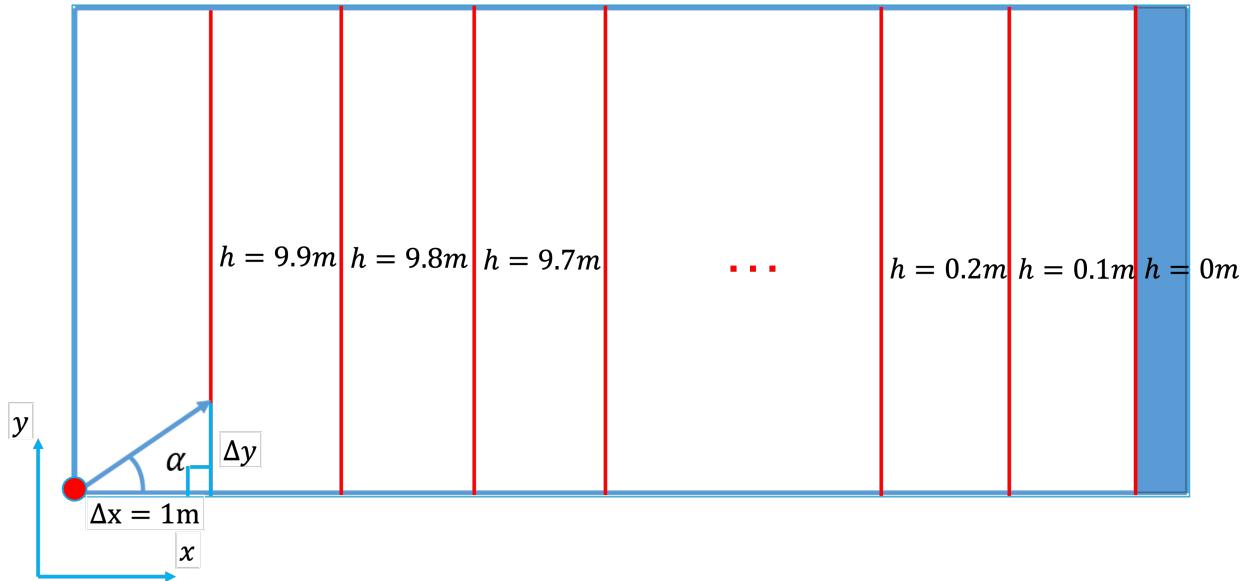
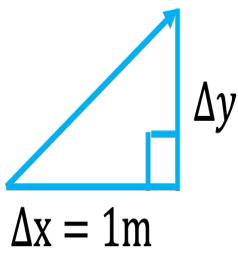
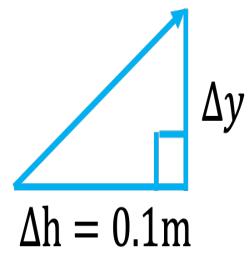


Figure 8: Modified Condition #2-2

다음으로, 앞서 식 (2.11)에 의하여 동일 기준점 하에서 수심(h)의 변화량인 $\Delta h = 0.1m$ 과 이동거리(x)의 변화량인 $\Delta x = 1m$ 에서 산정한 파랑의 굴절각 α 는 서로 같을 것이다. 이러한 사실로부터 수심(h)의 변화량인 $\Delta h = 0.1m$ 과 이동거리(x)의 변화량 $\Delta x = 1m$ 을 밑변으로 하는 직각삼각형을 도시해보면 아래 Figure 9과 같다.



(a) Ref #1-1



(b) Ref #1-2

Figure 9: Ref #1

위 Figure 9에 도시한 직각삼각형에 대해 삼각함수인 \tan 함수의 정의를 이용하여 y 방향으로의 이동거리에 대한 변화량 Δy 를 구한다.

즉 위에서 수심 $h = 9.9m$ 에서 구한 (2.21)의 파랑의 굴절각 $\alpha_2 = 0.52346266(rad) = 29.992^\circ$ 에 대한 \tan 값은 다음과 같고,

$$\tan(\alpha_2) = \tan(0.52346266(\text{rad})) = 0.5771688 \quad (2.22)$$

\tan 함수의 정의로부터, Figure 9a 상에서는 아래와 같이 정의되고,

$$\tan(\alpha_2) = \frac{\Delta y}{\Delta x} = \frac{\Delta y}{1m} \quad (2.23)$$

Figure 9b 상에서는 아래와 같이 정의된다.

$$\tan(\alpha_2) = \frac{\Delta y}{\Delta h} = \frac{\Delta y}{0.1m} \quad (2.24)$$

또한, 수심 $h = 9.9m$ 즉 원점으로부터의 이동거리 $x = 1m$ 일 때 (2.22)에서 구한 \tan 값은 0.5771688이 있으므로, 이를 이용하여 (2.23)와 (2.24)으로부터 연직방향으로의 이동거리에 대한 변화량 Δy 를 각각 다음과 같이 산정해낼 수 있다.

$$\tan(\alpha_2) = \frac{\Delta y}{1m} = 0.5771688 \quad (2.25)$$

$$\therefore \Delta y = 0.5771688 \times 1m = 0.5771688m \quad (2.26)$$

$$\tan(\alpha_2) = \frac{\Delta y}{0.1m} = 0.5771688 \quad (2.27)$$

$$\therefore \Delta y = 0.5771688 \times 0.1m = 0.05771688m \quad (2.28)$$

Figure 7 상에서는 x 방향으로의 이동거리의 변화량 $\Delta x = 1m$ 마다 수심의 변화량이 $\Delta h = 0.1m$ 이었으나, 단순히 수평방향으로의 이동거리의 변화량 Δx 와 수심의 변화량 Δh 가 0.1m로 동일하다고 가정($\Delta x = \Delta h = 0.1m$)하고, 해당 그림상의 좌표축과 원점을 차용하여 수심 h 에 대해 y 방향으로의 이동거리에 대한 변화량 Δy 를 누적하여, y 방향으로의 이동거리에 대한 좌표를 산정하고, 이를 그래프로 도출하면 다음과 같다.

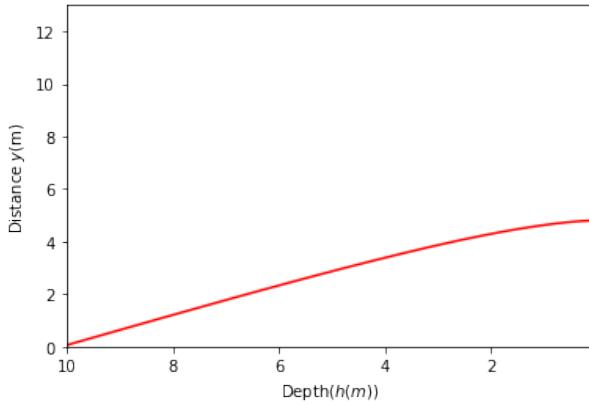


Figure 10: Wave Directions at each Depth h when the change in x is $0.1m$ ($\Delta x = 0.1m$) and the angle of the incoming wave is 30°

위 Figure 10는 $\Delta x = \Delta h = 0.1m$ 인 상황에서 연직방향으로의 이동거리에 대한 변화량 Δy 를 누적하여 y 방향으로의 이동거리에 대한 좌표를 산정함으로써, 각 수심 h 에서의 파향(Wave Directions)을 나타낸다.

이때, 우리는 앞서 $\Delta x = 0.1m$ 로 한다고 가정하였으므로, 해당 구간 내에서는 이전 경계 수심 h 로 동일하다.

다시 말해서, 이전 Figure 8과는 다르게, $\Delta x = 0.1m$ 마다 변화할 때 수심 역시 $\Delta h = 0.1m$ 마다 변화하는 아래의 상황을 살펴보자.

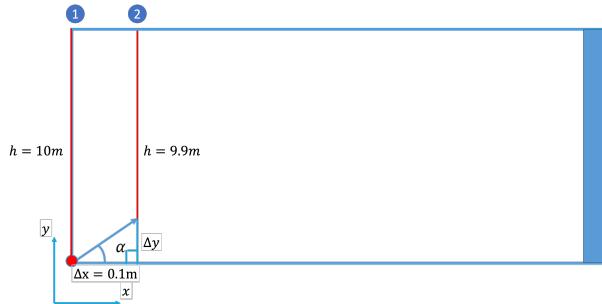


Figure 11: Modified Condition #2-3

위 Figure 11에 도시한 것과 같이 1번과 2번 구간 사이는 $\Delta x = 0.1m$ 로서, 해당 구간에서는 수심이 $h = 10m$ 로 1번 경계의 수심 $h = 10m$ 로 동일한 것을 확인할 수 있다.

우리는 앞서, $\Delta x = \Delta h = 0.1m$ 상황에서 수심 h 에 대해 y 방향으로의 이동거리 그래프를 산정하여 파향(Wave Directions)을 확인하였다.

파향(Wave Directions)은 $\Delta x = \Delta h = 0.1m$ 라고 가정한 위의 Figure 10 상과 동일하겠으나, 실제 좌표의 위치와 파의 경로는 상이할 것이므로, 실제 8의 상황 즉 $\Delta x = 1.0m$, $\Delta h = 0.1m$ 상황에서 파향과 파의 진행경로를

확인해보자.

이를 위해 Figure 7 상에서 x 방향으로의 이동거리의 변화량 $\Delta x = 1m$ 마다 수심의 변화량이 $\Delta h = 0.1m$ 이었으므로, 좌표축과 원점에 기반하여, 구한 y 방향으로의 이동거리에 대한 변화량 Δy 를 누적하여 원점으로부터의 이동거리 $\Delta x = 1m$ 마다 파향(Wave Directions)을 그래프로 그려보면 다음과 같다.

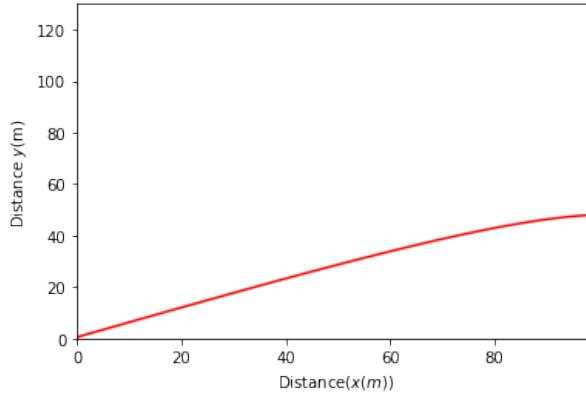


Figure 12: Wave Directions at each Distance x from Origin $(0,0)$ when the change in x is $1m(\Delta x = 1m)$ and the angle of the incoming wave is 30°

즉, 위의 Figure 12를 Figure 3 위에 겹쳐 표시하면 다음과 같고, 이는 파향 뿐만 아니라 파의 진행경로도 확인할 수 있다.

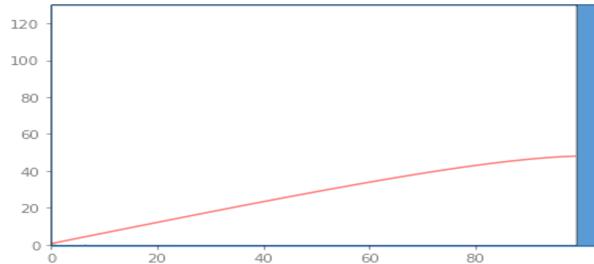
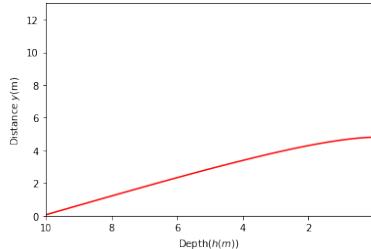
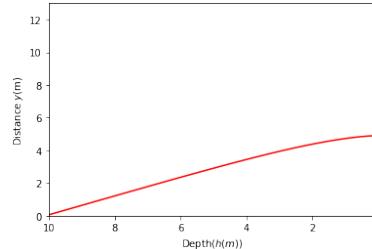


Figure 13: Wave Directions and Paths at each Distance x from Origin $(0,0)$ when the change in x is $1m(\Delta x = 1m)$ and the angle of the incoming wave is 30°

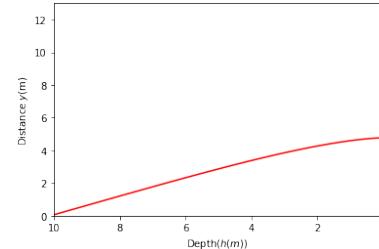
마지막으로, 위에서 Bisection Method로 산정한 파랑의 굴절각 α 로부터 $\Delta x = \Delta h = 0.1m$ 상황에서 작도한 수심 h 에 대해 y 방향으로의 이동거리를 나타낸, 파향(Wave Directions)에 대한 그래프와 나머지 두 방법을 통해 산정한 파랑의 굴절각 α 로부터 $\Delta x = \Delta h = 0.1m$ 상황에서 작도한 수심 h 에 대해 y 방향으로의 이동거리를 나타낸, 파향(Wave Directions)에 대한 그래프를 서로 비교해보자.



(a) Graph of Wave Directions at each Depth h when the change in x is $0.1m(\Delta x = 0.1m)$ and the angle of the incoming wave is 30° (Using Bisection Method)



(b) Graph of Wave Directions at each Depth h when the change in x is $0.1m(\Delta x = 0.1m)$ and the angle of the incoming wave is 30° (Using Eckart's Method)



(c) Graph of Wave Directions at each Depth h when the change in x is $0.1m(\Delta x = 0.1m)$ and the angle of the incoming wave is 30° (Using Hunt's Method)

Figure 14: Graphs of Wave Directions at each Depth h when the change in x is $0.1m(\Delta x = 0.1m)$ and the angle of the incoming wave is 30°

위의 Figure 14 상에서 확인할 수 있듯이, 세 방법에서의 파향(Wave Directions)에 대한 개형이 거의 유사한 것을 확인할 수 있다.

엄밀한 분석을 위해, Bisection Method로 산정한 파랑의 굴절각 $\alpha(rad)$ 에 대해 나머지 두 방법에 의해 산정한 파랑의 굴절각 $\alpha(rad)$ 와의 차이의 평균의 백분율 값을 비교하여 표로 정리하면 다음과 같다.

Table 2.4.1 Bisection Method vs. Eckart Method vs. Hunt's Method for angle α of refraction

	α (Using Eckart's Method)	α (Using Hunt's Method)
α (Using Bisection Method)	0.77287975%	0.33519741%

위 표에서 확인할 수 있듯이, Bisection Method를 기준으로 파랑의 굴절각 $\alpha(rad)$ 와의 차이의 평균의 백분율 값으로서 Eckart's Method에서는 약 0.773%가, Hunt's Method에서는 약 0.335%로 Hunt's Method에서 그 차이가 작음을 확인할 수 있다.

2.5 Question #2

입사파의 각도가 45° 인 경우에 대하여 파의 진행경로를 예측하시오.

2.6 Answer #2

앞서 Question #1 섹션에서 입사파의 각도가 30° 인 경우에서와 동일한 원리를 적용하여 입사파의 각도가 45° 인 경우에 파의 진행경로를 예측할 수 있다.

먼저 (1.17) 식의 C_1 을 앞서 Question #0 섹션에서 산정한 파수 k 와 각진동수 σ 를 식 (2.8)에 대입하여 C_1 을 산정하면 (2.10)의 $C_1 = 5.577615166155759(m/sec)$ 이다.

그 다음, 입사파의 각도가 45° 이므로, 앞서 Introduction 섹션에서 살펴본 파의 파랑의 굴절각을 계산하는 식 (1.17)의 $\alpha_1 = 45^\circ$ 임을 알 수 있다.

임의지점과 그때의 수심을 선정하기 위하여, 시작점인 수심 $h_0 = 10m$ 를 기준으로 하여, Figure 2에 도시한 x 축의 양의 방향으로 $\Delta x = 1m$ 간격으로 잘라낸 다음, Figure 6에서 도시한 보라색 점의 좌표를 $(0, 0)$ 으로 설정하였을 때, 연두색 점의 좌표는 $(0, 10)$ 이 되고, 기울기는 $-\frac{1}{10}$ 이므로, 이로부터 직선의 방정식을 다음과 같이 유도하여 임의로 원점에서 x 만큼 이동하였을 때, 원점에서 해당 지점까지 이동한 거리에서의 수심 $y = h_x$ 를 산정한다.

유도한 직선의 방정식 (2.11)에서 $y = h_x = 0m$ 일 때, 즉 수심이 $0m$ 일 때의 원점에서 이동한 거리 x 를 계산하면 (2.13)의 $x = 100m$ 이었다.

그러므로, $x = 0m$ 에서 $x = 99m$ 까지 $x_0, x_1, x_2, \dots, x_{99}$ 각각에 대해 총 100개의 수심 $h_0, h_1, h_2, \dots, h_{99}$ 을 산정한다.

산정한 각 수심에 대해 음해법(Implicit Method)인 이분법(Bisection Method)을 이용하여 미지의 파수 k 값을 산정하기 위해, 그 초기값을 양해법(Explicit Method) 중 하나인 Eckart 식 (1.6)을 이용하여 산정한다.

예를 들어, $x_1 = 1m$ 에서 수심 h_1 의 값은 $9.9m$ 이고, 먼 바다에서 진행되어 온 동일 파에 대해서는 주기 T 가 동일하므로, 앞서 Question #0 섹션에서 산정한 T_0 와 그 값이 동일하다. 즉, $T = T_0$ 이다. 또한, Eckart 식의 σ^2 값 역시 동일 주기이므로, (1.2) 관계식으로부터 이것 역시 앞서 Question #0 섹션에서 산정한 각진동수 σ 의 제곱한 값 $\sigma^2 = 3.0704133(rad^2/sec^2)$ 과 동일하다. 마지막으로, 중력 가속도 $g = 9.81m/sec^2$ 이라 하면, Eckart 식 (1.6)을 k 에 대한 식으로 정리하여, 위의 값을 대입하여 초기 k 값을 산정해낼 수 있다.

그때의 k 의 초기값은 (2.15)의 $0.31362574(rad/m)$ 로 도출된다.

산정한 초기값을 이용하여, 음해법(Implicit Method) 중 하나인 이분법(Bisection Method)을 통해 컴퓨터를 이용하여 수치해를 구해보자.

이를 위해, 우선 분산관계식(Dispersion Relationship) (1.1)의 좌변이 0이 되도록 모든 항을 이항하여 정리한 식 (2.16)에 대해 오차(Error)를 0.5×10^{-6} 으로 설정하여, 위에서 산정한 (2.14)의 초기 k 값의 ± 0.1 만큼을 이분법의 해 탐색 구간으로 설정하여 k 값을 산정한다.

예를 들어, 위에서와 같이 수심 $h = 9.9m$ 에 대해 이분법(Bisection Method)을 이용하여 k 값을 산정하면 그 값은 $k = 0.31423335(rad/m)$ 으로, 초기값과의 백분율 차이는 약 0.06% 수준이다.

다음의 파속 C 식에 수치해석 기법인 이분법(Bisection Method)으로 도출한 k 값과 Question #0에서 산정한 σ 값을 대입하여 (1.17)의 파속 C_2 를 산정하면 (2.19)의 $C_2 = 5.57630018(m/sec)$ 이다.

산정한 파속 C_2 의 값과 C_1 의 값 그리고 입사파의 각도 α_1 의 값을 앞서 Introduction 섹션에서 유도한 Snell의 법칙에 대한 식 (1.17)에 대입함으로써, 임의지점에서의 파의 굴절각도 α_2 를 산정한다.

$$\alpha_2 = \sin^{-1}(\sin(\alpha_1)) \cdot \frac{C_2}{C_1} = \sin^{-1}(\sin(45^\circ)) \cdot \frac{5.57630018(m/sec)}{5.577615166155759(m/sec)} \quad (2.29)$$

$$\therefore \alpha_2 = 0.78516243(rad) \approx 44.986^\circ \quad (2.30)$$

위의 원리를 그대로 적용하여 나머지 수심 $h_2, h_3, h_4, \dots, h_{99}$ 에서의 파랑의 굴절각 α_2 를 구한다.

이전 섹션에서 살펴보았듯이, 주어진 조건에 대한 Figure 3 상을 기준으로 하는 파랑의 굴절각 α 에 대한 그래프를 산정하기 위해 해당 조건에 대한 Figure 3에 기준축과 원점을 설정했고, 이는 Figure 7에서 확인할 수 있다.

또한, Figure 7에서 축의 방향은 x 축의 경우 오른쪽으로 진행할 수록 증가하고, y 축의 경우 위쪽으로 진행할 수록 증가하는 것으로 설정하였다.

더불어, Figure 7에서 y 축은 연직방향으로의 이동거리(Distance)이고, x 축의 경우 수평방향으로의 이동거리(Distance)를 나타낸다는 사실을 알고 있다.

추가적으로, 좌표의 원점 $(0, 0)$ 인 위치를 빨간점으로 도시하였다.

마지막으로, 수심 h 를 Figure 7에 도시해보면 Figure 8과 같았다.

다음으로, 앞서 식 (2.11)에 의하여 동일 기준점 하에서 수심(h)의 변화량인 $\Delta h = 0.1m$ 과 이동거리(x)의 변화량인 $\Delta x = 1m$ 에서 산정한 파랑의 굴절각 α 는 서로 같을 것이다. 이러한 사실로부터 수심(h)의 변화량인 $\Delta h = 0.1m$ 과 이동거리(x)의 변화량 $\Delta x = 1m$ 을 밑변으로 하는 직각삼각형을 도시하면 Figure 9과 같았다.

앞서 이전 섹션에서 Figure 9에 도시한 직각삼각형에 대해 삼각함수인 \tan 함수의 정의를 이용하여 y 방향으로의 이동거리에 대한 변화량 Δy 를 구했다.

즉 위에서 수심 $h = 9.9m$ 에서 구한 (2.30)의 파랑의 굴절각 $\alpha_2 = 0.78516243(rad) = 44.986^\circ$ 에 대한 \tan 값은 다음과 같고,

$$\tan(\alpha_2) = \tan(0.78516243(rad)) = 1.73041914 \quad (2.31)$$

\tan 함수의 정의로부터, Figure 9a 상에서는 식 (2.23)과 같이 정의되었고, Figure 9b 상에서는 식 (2.24)로 정의된 바 있다.

또한, 수심 $h = 9.9m$ 즉 원점으로부터의 이동거리 $x = 1m$ 일 때 (2.31)에서 구한 \tan 값은 1.73041914 이었으므로, 이를 이용하여 (2.23)와 (2.24)으로부터 연직방향으로의 이동거리에 대한 변화량 Δy 를 각각 다음과 같이 산정해낼 수 있다.

$$\tan(\alpha_2) = \frac{\Delta y}{1m} = 1.73041914 \quad (2.32)$$

$$\therefore \Delta y = 0.5771688 \times 1m = 1.73041914m \quad (2.33)$$

$$\tan(\alpha_2) = \frac{\Delta y}{0.1m} = 1.73041914 \quad (2.34)$$

$$\therefore \Delta y = 0.5771688 \times 0.1m = 0.173041914m \quad (2.35)$$

Figure 7 상에서는 x 방향으로의 이동거리의 변화량 $\Delta x = 1m$ 마다 수심의 변화량이 $\Delta h = 0.1m$ 이었으나, 단순히 수평방향으로의 이동거리의 변화량 Δx 와 수심의 변화량 Δh 가 $0.1m$ 로 동일하다고 가정($\Delta x = \Delta h = 0.1m$)하고, 해당 그림상의 좌표축과 원점을 차용하여 수심 h 에 대해 y 방향으로의 이동거리에 대한 변화량 Δy 를 누적하여, y 방향으로의 이동거리에 대한 좌표를 산정하고, 이를 그래프로 도출하면 다음과 같다.

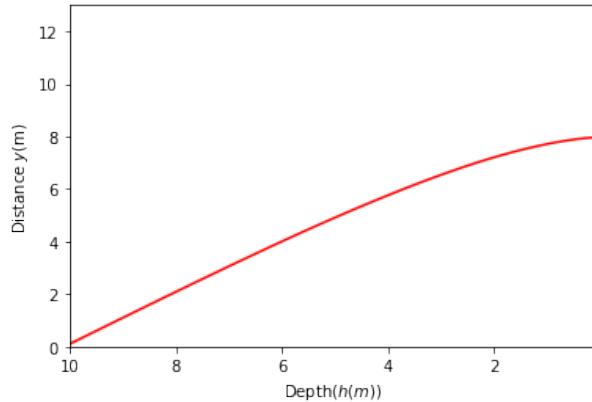


Figure 15: Wave Directions at each Depth h when the change in x is $0.1m$ ($\Delta x = 0.1m$) and the angle of the incoming wave is 45°

위 Figure 15는 $\Delta x = \Delta h = 0.1m$ 인 상황에서 연직방향으로의 이동거리에 대한 변화량 Δy 를 누적하여 y 방향으로의 이동거리에 대한 좌표를 산정함으로써, 각 수심 h 에서의 파향(Wave Directions)을 나타낸다.

이때, 우리는 앞서 $\Delta x = 0.1m$ 로 한다고 가정하였으므로, 해당 구간 내에서는 이전 경계 수심 h 로 동일하다.

다시 말해서, 이전 Figure 8과는 다르게, $\Delta x = 0.1m$ 마다 변화할 때 수심 역시 $\Delta h = 0.1m$ 마다 변화하는 Figure 11에서 1번과 2번 구간 사이는 $\Delta x = 0.1m$ 로서, 해당 구간에서는 수심이 $h = 10m$ 로 1번 경계의 수심 $h = 10m$

로 동일한 것을 확인할 수 있었다.

또한, 우리는 앞서, $\Delta x = \Delta h = 0.1m$ 상황에서 수심 h 에 대해 y 방향으로의 이동거리 그래프를 산정하여 파향(Wave Directions)을 확인하였다.

파향(Wave Directions)은 $\Delta x = \Delta h = 0.1m$ 라고 가정한 위의 Figure 15 상과 동일하겠으나, 실제 좌표의 위치와 파의 경로는 상이할 것이므로, 실제 8의 상황 즉 $\Delta x = 1.0m$, $\Delta h = 0.1m$ 상황에서 파향과 파의 진행경로를 확인해보자.

이를 위해 Figure 7 상에서 x 방향으로의 이동거리의 변화량 $\Delta x = 1m$ 마다 수심의 변화량이 $\Delta h = 0.1m$ 이었으므로, 좌표축과 원점에 기반하여, 구한 y 방향으로의 이동거리에 대한 변화량 Δy 를 누적하여 원점으로부터의 이동거리 $\Delta x = 1m$ 마다 파향(Wave Directions)을 그래프로 그려보면 다음과 같다.

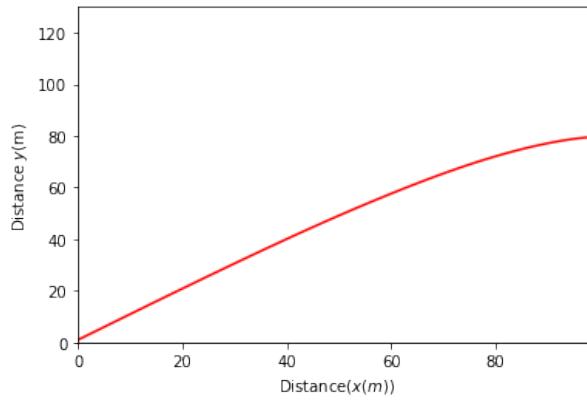


Figure 16: Wave Directions at each Distance x from Origin $(0,0)$ when the change in x is $1m$ ($\Delta x = 1m$) and the angle of the incoming wave is 45°

즉, 위의 Figure 20를 Figure 3 위에 겹쳐 표시하면 다음과 같고, 이는 파향 뿐만 아니라 파의 진행경로도 확인할 수 있다.

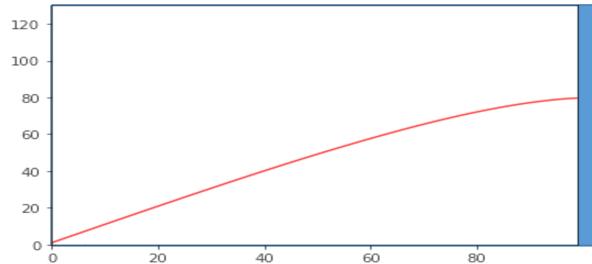
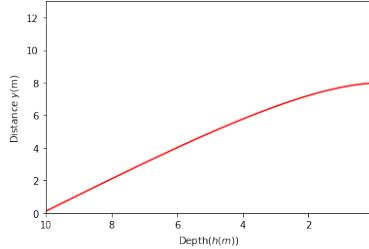


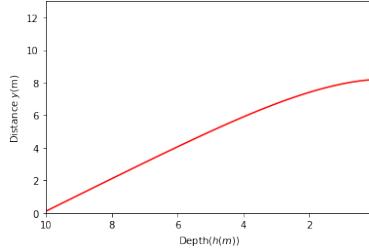
Figure 17: Wave Directions and Paths at each Distance x from Origin $(0,0)$ when the change in x is $1m$ ($\Delta x = 1m$) and the angle of the incoming wave is 45°

마지막으로, 위에서 Bisection Method로 산정한 파랑의 굴절각 α 로부터 $\Delta x = \Delta h = 0.1m$ 상황에서 작도한

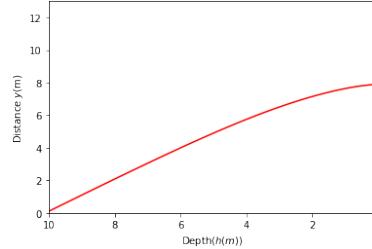
수심 h 에 대해 y 방향으로의 이동거리를 나타낸, 파향(Wave Directions)에 대한 그래프와 나머지 두 방법을 통해 산정한 파랑의 굴절각 α 로부터 $\Delta x = \Delta h = 0.1m$ 상황에서 작도한 수심 h 에 대해 y 방향으로의 이동거리를 나타낸, 파향(Wave Directions)에 대한 그래프를 서로 비교해보자.



(a) Graph of Wave Directions at each Depth h when the change in x is $0.1m(\Delta x = 0.1m)$ and the angle of the incoming wave is 45° (Using Bisection Method)



(b) Graph of Wave Directions at each Depth h when the change in x is $0.1m(\Delta x = 0.1m)$ and the angle of the incoming wave is 45° (Using Eckart's Method)



(c) Graph of Wave Directions at each Depth h when the change in x is $0.1m(\Delta x = 0.1m)$ and the angle of the incoming wave is 45° (Using Hunt's Method)

Figure 18: Graphs of Wave Directions at each Depth h when the change in x is $0.1m(\Delta x = 0.1m)$ and the angle of the incoming wave is 45°

위의 Figure 18 상에서 확인할 수 있듯이, 세 방법에서의 파향(Wave Directions)에 대한 개형이 거의 유사한 것을 확인할 수 있다.

엄밀한 분석을 위해, Bisection Method로 산정한 파랑의 굴절각 $\alpha(rad)$ 에 대해 나머지 두 방법에 의해 산정한 파랑의 굴절각 $\alpha(rad)$ 와의 차이의 평균의 백분율 값을 비교하여 표로 정리하면 다음과 같다.

Table 2.6.1 Bisection Method vs. Eckart Method vs. Hunt's Method for angle α of refraction

	α (Using Eckart's Method)	α (Using Hunt's Method)
α (Using Bisection Method)	1.2659064%	0.52660238%

위 표에서 확인할 수 있듯이, Bisection Method를 기준으로 파랑의 굴절각 $\alpha(rad)$ 와의 차이의 평균의 백분율 값으로서 Eckart's Method에서는 약 1.266%이고, Hunt's Method에서는 약 0.527%로 Hunt's Method에서 그 차이가 작음을 확인할 수 있다.

2.7 Question #3

입사파의 각도가 60° 인 경우에 대하여 파의 진행경로를 예측하시오.

2.8 Answer #3

앞서 Question #1, Question #2 섹션에서 입사파의 각도가 $30^\circ, 45^\circ$ 인 경우에서와 동일한 원리를 적용하여 입사파의 각도가 60° 인 경우에 파의 진행경로를 예측할 수 있다.

먼저 (1.17) 식의 C_1 을 앞서 Question #0 섹션에서 산정한 파수 k 와 각진동수 σ 를 식 (2.8)에 대입하여 C_1 을 산정하면 (2.10)의 $C_1 = 5.577615166155759(m/sec)$ 이다.

그 다음, 입사파의 각도가 60° 이므로, 앞서 Introduction 섹션에서 살펴본 파의 파랑의 굴절각을 계산하는 식 (1.17)의 $\alpha_1 = 60^\circ$ 임을 알 수 있다.

임의지점과 그때의 수심을 설정하기 위하여, 시작점인 수심 $h_0 = 10m$ 를 기준으로 하여, Figure 2에 도시한 x 축의 양의 방향으로 $\Delta x = 1m$ 간격으로 잘라낸 다음, Figure 6에서 도시한 보라색 점의 좌표를 $(0, 0)$ 으로 설정하였을 때, 연두색 점의 좌표는 $(0, 10)$ 이 되고, 기울기는 $-\frac{1}{10}$ 이므로, 이로부터 직선의 방정식을 다음과 같이 유도하여 임의로 원점에서 x 만큼 이동하였을 때, 원점에서 해당 지점까지 이동한 거리에서의 수심 $y = h_x$ 를 산정한다.

유도한 직선의 방정식 (2.11)에서 $y = h_x = 0m$ 일 때, 즉 수심이 $0m$ 일 때의 원점에서 이동한 거리 x 를 계산하면 (2.13)의 $x = 100m$ 이었다.

그러므로, $x = 0m$ 에서 $x = 99m$ 까지 $x_0, x_1, x_2, \dots, x_{99}$ 각각에 대해 총 100개의 수심 $h_0, h_1, h_2, \dots, h_{99}$ 을 산정한다.

산정한 각 수심에 대해 음해법(Implicit Method)인 이분법(Bisection Method)을 이용하여 미지의 파수 k 값을 산정하기 위해, 그 초기값을 양해법(Explicit Method) 중 하나인 Eckart 식 (1.6)을 이용하여 산정한다.

예를 들어, $x_1 = 1m$ 에서 수심 h_1 의 값은 $9.9m$ 이고, 먼 바다에서 진행되어 온 동일 파에 대해서는 주기 T 가 동일하므로, 앞서 Question #0 섹션에서 산정한 T_0 와 그 값이 동일하다. 즉, $T = T_0$ 이다. 또한, Eckart 식의 σ^2 값 역시 동일 주기이므로, (1.2) 관계식으로부터 이것 역시 앞서 Question #0 섹션에서 산정한 각진동수 σ 의 제곱한 값 $\sigma^2 = 3.0704133(rad^2/sec^2)$ 과 동일하다. 마지막으로, 중력 가속도 $g = 9.81m/sec^2$ 이라 하면, Eckart 식 (1.6)을 k 에 대한 식으로 정리하여, 위의 값을 대입하여 초기 k 값을 산정해낼 수 있다.

그때의 k 의 초기값은 (2.15)의 $0.31362574(rad/m)$ 로 도출된다.

산정한 초기값을 이용하여, 음해법(Implicit Method) 중 하나인 이분법(Bisection Method)을 통해 컴퓨터를 이용하여 수치해를 구해보자.

이를 위해, 우선 분산관계식(Dispersion Relationship) (1.1)의 좌변이 0이 되도록 모든 항을 이항하여 정리한 식 (2.16)에 대해 오차(Error)를 0.5×10^{-6} 으로 설정하여, 위에서 산정한 (2.14)의 초기 k 값의 ± 0.1 만큼을 이분법의 해 탐색 구간으로 설정하여 k 값을 산정한다.

예를 들어, 위에서와 같이 수심 $h = 9.9m$ 에 대해 이분법(Bisection Method)을 이용하여 k 값을 산정하면 그 값은 $k = 0.31423335(rad/m)$ 으로, 초기값과의 백분율 차이는 약 0.06% 수준이다.

다음의 파속 C 식에 수치해석 기법인 이분법(Bisection Method)으로 도출한 k 값과 Question #0에서 산정한 σ 값을 대입하여 (1.17)의 파속 C_2 를 산정하면 (2.19)의 $C_2 = 5.57630018(m/sec)$ 이다.

산정한 파속 C_2 의 값과 C_1 의 값 그리고 입사파의 각도 α_1 의 값을 앞서 Introduction 섹션에서 유도한 Snell의 법칙에 대한 식 (1.17)에 대입함으로써, 임의지점에서의 파의 굴절각도 α_2 를 산정한다.

$$\alpha_2 = \sin^{-1}(\sin(\alpha_1)) \cdot \frac{C_2}{C_1} = \sin^{-1}(\sin(45^\circ)) \cdot \frac{5.57630018(m/sec)}{5.577615166155759(m/sec)} \quad (2.36)$$

$$\therefore \alpha_2 = 1.04678935(rad) \approx 59.977^\circ \quad (2.37)$$

위의 원리를 그대로 적용하여 나머지 수심 $h_2, h_3, h_4, \dots, h_{99}$ 에서의 파랑의 굴절각 α_2 를 구한다.

이전 섹션에서 살펴보았듯이, 주어진 조건에 대한 Figure 3 상을 기준으로 하는 파랑의 굴절각 α 에 대한 그래프를 산정하기 위해 해당 조건에 대한 Figure 3에 기준축과 원점을 설정했고, 이는 Figure 7에서 확인할 수 있다.

또한, Figure 7에서 축의 방향은 x 축의 경우 오른쪽으로 진행할 수록 증가하고, y 축의 경우 위쪽으로 진행할 수록 증가하는 것으로 설정하였다.

더불어, Figure 7에서 y 축은 연직방향으로의 이동거리(Distance)이고, x 축의 경우 수평방향으로의 이동거리(Distance)를 나타낸다는 사실을 알고 있다.

추가적으로, 좌표의 원점 $(0, 0)$ 인 위치를 빨간점으로 도시하였다.

마지막으로, 수심 h 를 Figure 7에 도시해보면 Figure 8과 같았다.

다음으로, 앞서 식 (2.11)에 의하여 동일 기준점 하에서 수심(h)의 변화량인 $\Delta h = 0.1m$ 와 이동거리(x)의 변화량인 $\Delta x = 1m$ 에서 산정한 파랑의 굴절각 α 는 서로 같을 것이다. 이러한 사실로부터 수심(h)의 변화량인 $\Delta h = 0.1m$ 과 이동거리(x)의 변화량 $\Delta x = 1m$ 을 밑변으로 하는 직각삼각형을 도시하면 Figure 9과 같았다.

앞서 이전 섹션에서 Figure 9에 도시한 직각삼각형에 대해 삼각함수인 \tan 함수의 정의를 이용하여 y 방향으로의 이동거리에 대한 변화량 Δy 를 구했다.

즉 위에서 수심 $h = 9.9m$ 에서 구한 (2.30)의 파랑의 굴절각 $\alpha_2 = 1.04678935(rad) = 59.977^\circ$ 에 대한 \tan 값은 다음과 같고,

$$\tan(\alpha_2) = \tan(1.04678935(rad)) = 1.73041914 \quad (2.38)$$

\tan 함수의 정의로부터, Figure 9a 상에서는 식 (2.23)과 같이 정의되었고, Figure 9b 상에서는 식 (2.24)로 정의된 바 있다.

또한, 수심 $h = 9.9m$ 즉 원점으로부터의 이동거리 $x = 1m$ 일 때 (2.38)에서 구한 \tan 값은 1.73041914

이었으므로, 이를 이용하여 (2.23)와 (2.24)으로부터 연직방향으로의 이동거리에 대한 변화량 Δy 를 각각 다음과 같이 산정해낼 수 있다.

$$\tan(\alpha_2) = \frac{\Delta y}{1m} = 1.73041914 \quad (2.39)$$

$$\therefore \Delta y = 0.5771688 \times 1m = 1.73041914m \quad (2.40)$$

$$\tan(\alpha_2) = \frac{\Delta y}{0.1m} = 1.73041914 \quad (2.41)$$

$$\therefore \Delta y = 0.5771688 \times 0.1m = 0.173041914m \quad (2.42)$$

Figure 7 상에서는 x 방향으로의 이동거리의 변화량 $\Delta x = 1m$ 마다 수심의 변화량이 $\Delta h = 0.1m$ 이었으나, 단순히 수평방향으로의 이동거리의 변화량 Δx 와 수심의 변화량 Δh 가 $0.1m$ 로 동일하다고 가정($\Delta x = \Delta h = 0.1m$)하고, 해당 그림상의 좌표축과 원점을 차용하여 수심 h 에 대해 y 방향으로의 이동거리에 대한 변화량 Δy 를 누적하여, y 방향으로의 이동거리에 대한 좌표를 산정하고, 이를 그래프로 도출하면 다음과 같다.

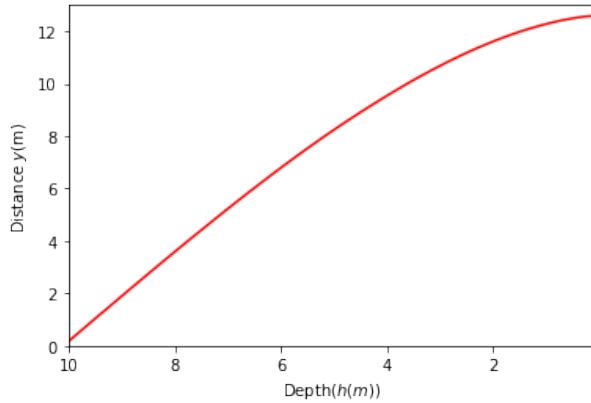


Figure 19: Wave Directions at each Depth h when the change in x is $0.1m$ ($\Delta x = 0.1m$) and the angle of the incoming wave is 60°

위 Figure 19는 $\Delta x = \Delta h = 0.1m$ 인 상황에서 연직방향으로의 이동거리에 대한 변화량 Δy 를 누적하여 y 방향으로의 이동거리에 대한 좌표를 산정함으로써, 각 수심 h 에서의 파향(Wave Directions)을 나타낸다.

이때, 우리는 앞서 $\Delta x = 0.1m$ 로 한다고 가정하였으므로, 해당 구간 내에서는 이전 경계 수심 h 로 동일하다.

다시 말해서, 이전 Figure 8과는 다르게, $\Delta x = 0.1m$ 마다 변화할 때 수심 역시 $\Delta h = 0.1m$ 마다 변화하는 Figure 11에서 1번과 2번 구간 사이는 $\Delta x = 0.1m$ 로서, 해당 구간에서는 수심이 $h = 10m$ 로 1번 경계의 수심 $h = 10m$ 로 동일한 것을 확인할 수 있었다.

또한, 우리는 앞서, $\Delta x = \Delta h = 0.1m$ 상황에서 수심 h 에 대해 y 방향으로의 이동거리 그래프를 산정하여 파향 (Wave Directions)을 확인하였다.

파향(Wave Directions)은 $\Delta x = \Delta h = 0.1m$ 라고 가정한 위의 Figure 19 상과 동일하겠으나, 실제 좌표의 위치와 파의 경로는 상이할 것이므로, 실제 8의 상황 즉 $\Delta x = 1.0m$, $\Delta h = 0.1m$ 상황에서 파향과 파의 진행경로를 확인해보자.

이를 위해 Figure 7 상에서 x 방향으로의 이동거리의 변화량 $\Delta x = 1m$ 마다 수심의 변화량이 $\Delta h = 0.1m$ 이었으므로, 좌표축과 원점에 기반하여, 구한 y 방향으로의 이동거리에 대한 변화량 Δy 를 누적하여 원점으로부터의 이동거리 $\Delta x = 1m$ 마다 파향(Wave Directions)을 그래프로 그려보면 다음과 같다.

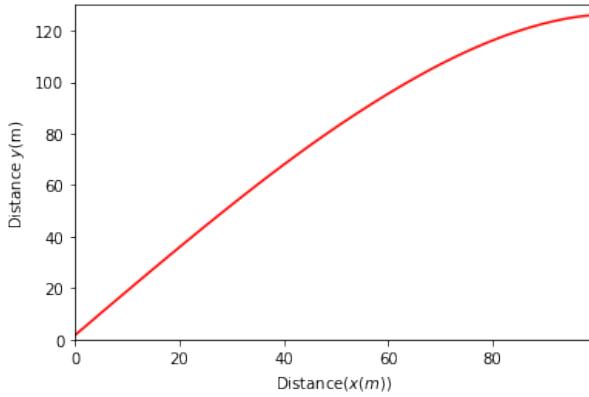


Figure 20: Wave Directions at each Distance x from Origin $(0, 0)$ when the change in x is $1m$ ($\Delta x = 1m$) and the angle of the incoming wave is 60°

즉, 위의 Figure 20를 Figure 3 위에 겹쳐 표시하면 다음과 같고, 이는 파향 뿐만 아니라 파의 진행경로도 확인할 수 있다.

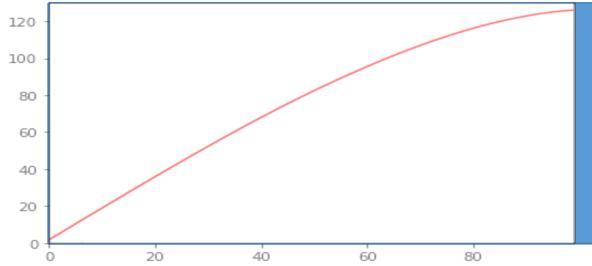
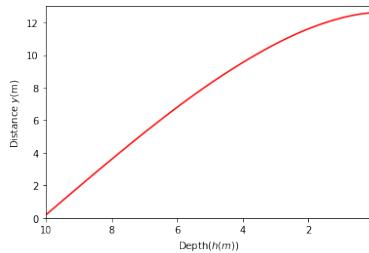
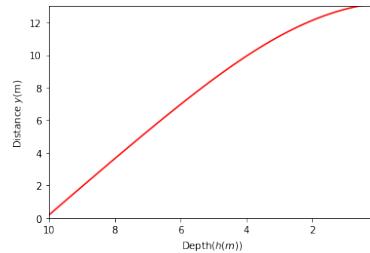


Figure 21: Wave Directions and Paths at each Distance x from Origin $(0, 0)$ when the change in x is $1m(\Delta x = 1m)$ and the angle of the incoming wave is 60°

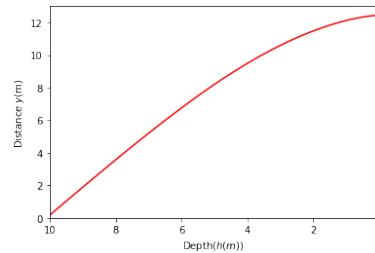
마지막으로, 위에서 Bisection Method로 산정한 파랑의 굴절각 α 로부터 $\Delta x = \Delta h = 0.1m$ 상황에서 작도한 수심 h 에 대해 y 방향으로의 이동거리를 나타낸, 파향(Wave Directions)에 대한 그래프와 나머지 두 방법을 통해 산정한 파랑의 굴절각 α 로부터 $\Delta x = \Delta h = 0.1m$ 상황에서 작도한 수심 h 에 대해 y 방향으로의 이동거리를 나타낸, 파향(Wave Directions)에 대한 그래프를 서로 비교해보자.



(a) Graph of Wave Directions at each Depth h when the change in x is $0.1m(\Delta x = 0.1m)$ and the angle of the incoming wave is 60° (Using Bisection Method)



(b) Graph of Wave Directions at each Depth h when the change in x is $0.1m(\Delta x = 0.1m)$ and the angle of the incoming wave is 60° (Using Eckart's Method)



(c) Graph of Wave Directions at each Depth h when the change in x is $0.1m(\Delta x = 0.1m)$ and the angle of the incoming wave is 60° (Using Hunt's Method)

Figure 22: Graphs of Wave Directions at each Depth h when the change in x is $0.1m(\Delta x = 0.1m)$ and the angle of the incoming wave is 60°

위의 Figure 22 상에서 확인할 수 있듯이, 세 방법에서의 파향(Wave Directions)에 대한 개형이 거의 유사한 것을 확인할 수 있다.

엄밀한 분석을 위해, Bisection Method로 산정한 파랑의 굴절각 $\alpha(rad)$ 에 대해 나머지 두 방법에 의해 산정한 파랑의 굴절각 $\alpha(rad)$ 와의 차이의 평균의 백분율 값을 비교하여 표로 정리하면 다음과 같다.

Table 2.6.1 Bisection Method vs. Eckart Method vs. Hunt's Method for angle α of refraction

	α (Using Eckart's Method)	α (Using Hunt's Method)
α (Using Bisection Method)	1.94232869%	0.74774457%

위 표에서 확인할 수 있듯이, Bisection Method를 기준으로 파랑의 굴절각 $\alpha(rad)$ 와의 차이의 평균의 백분율

값으로서 Eckart's Method에서는 약 1.942%이고, Hunt's Method에서는 약 0.748%로 Hunt's Method에서 그 차이가 작음을 확인할 수 있다.

3 Conclusion

3.1 Conclusion

본 보고서에서는 서론에서 언급하였듯이, 기본적으로 아래와 같이 정의되는 분산관계식(Dispersion Relationship)을 이용하여 모든 문제를 해결하였다.

$$\sigma^2 = gktanh(kh)$$

첫 과제로서, kh 가 주어지는 경우에는 단순히 위의 분산관계식에 그 값을 대입 또는 주어진 수심 h 로부터 k 값을 도출하여 이를 대입함으로써, 각진동수 σ 를 구하였고, $T = \frac{2\pi}{\sigma}$ 의 관계식으로부터 주기 T 를 산정할 수 있었다.

또한, 두 번째 이후의 과제에서는 첫 번째 과제에서와는 다르게, k 값이 주어지지 않은 상황에서 k 값을 산정하고 이로부터 파속 C 를 구하는 과정이 수반되었다.

특히, Question #1, #2, #3에서와 같이 파향(Wave Directions)을 계산하기 위하여 미지인 k 를 산정해야하는 경우에는 Eckart나 Hunt 식과 같은 양해법(Explicit Method)을 이용하여 k 값 산정 및 음해법(Implicit)에 대한 초깃값으로 이를 이용하여, 더 정확하고 정밀한 해를 찾아보는 과정을 살펴보았다. 또한, 파수 k 를 산정한 이후에는 파속 C 에 대한 식 (2.17)을 이용하여 이것을 구하는 절차를 밟았다.

그 다음, 주어진 조건으로부터 산정한 파속 C_1 과 입사파의 굴절각도 α_1 , 그리고 산정한 파속 C_2 와 식 (1.17)을 이용하여 파랑의 굴절각도 α_2 를 산정하고, 이에 대한 \tan 값으로부터 \tan 의 정의를 이용하여 수심의 변화량 ($\Delta h(m)$)이나 수평방향으로의 이동거리 변화량 $\Delta x(m)$ 에 대한 연직방향으로의 이동거리 변화량 $\Delta y(m)$ 를 산정하였으며, 이를 누적시킴으로써 y 좌표를 산정하여 이를 통해 Figure 3 상에 파의 진행경로를 그릴 수 있었다.

또한, 입사하는 하나의 파에 대해서는 그 주기(Periods) T 가 일정하다는 조건을 이용할 수 있기 때문에 $kh = \pi$ 라는 조건과 분산관계식(Dispersion Relationship) (1.1)을 이용하여 σ^2 의 값을 산정하고, 이로부터 식 (1.3)을 통해 주기 T 를 도출하여 임의지점에서의 파랑의 굴절각도 α_2 를 산정함에 있어 이를 고정값으로 사용할 수 있었다.

추가적으로, 음해법(Implicit Method)인 이분법(Bisection Method)으로 파수 k 와 파속 C 을 양해법(Explicit Method)인 Eckart의 방법(Eckart's Method)과 Hunt의 방법(Hunt's Method)과 비교하였다.

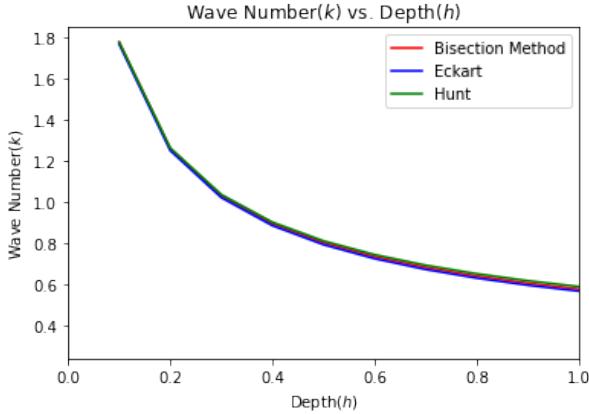


Figure 23: Graph of Wave Number(k) vs. Depth(h)

Table 2.2.1 Bisection Method vs. Eckart Method vs. Hunt's Method for k

	k (Using Eckart's Method)	k (Using Hunt's Method)
k (Using Bisection Method)	0.64951202%	0.37930637%

위의 결과로부터, Eckart의 방법은 Bisection Method보다 상대적으로 그 차이가 크게 발생하였으나, Hunt의 방법은 Eckart의 방법에 비해 상대적으로 작은 차이를 보임을 확인할 수 있었다.

이러한 결과는, You(2003)에 따르면, 여러 개의 파랑분산식의 양해(Hunt, 1979; Nielsen, 1982; Fenton과 McKee, 1990; Guo, 2002; Nielsen, 2002; You, 2003)를 비교하였을 때, Hunt의 해가 $\nu \leq \pi$ 의 수심범위에서 가장 정확하다는 것을 밝혀 내었는데, 수심 h 에 대한 $y = \nu = \frac{\sigma^2 h}{g}$ 의 그래프를 도출하고, 그때의 최댓값 역시 도시해보면 다음과 같고,

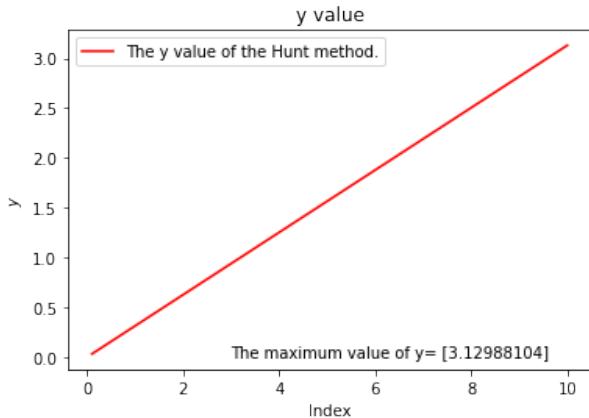


Figure 24: Graph of $y = \nu = \frac{\sigma^2 h}{g}$ vs. Depth(h)

그때의 최댓값이 3.12988104으로 이는 π 보다 작기 때문에 즉, $\nu \leq \pi$ 범위에 존재하기 때문에 양해법(Explicit

Method) 중에서 가장 정확한 해가 도출되었다는 결론을 내릴 수 있다.

즉, 위의 연구결과가 위에서 Hunt's Method가 Bisection Method와 크게 차이가 발생하지 않다는 사실을 뒷받침하고 있음도 확인할 수 있다.

다음으로, 앞서 우리는 개별적인 입사파의 굴절각도 α_1 에 따라 음해법(Implicit Method) 중 이분법(Bisection Method)을 이용하여 산정한 파수 k 에 대해 수심 $h(m)$ 에 따른 파향(Wave Directions)에 대한 그래프를 각각 별도의 Figure로 구성하여 비교한 바 있는데, 이를 하나의 Figure 내에서 살펴보면 다음과 같다.

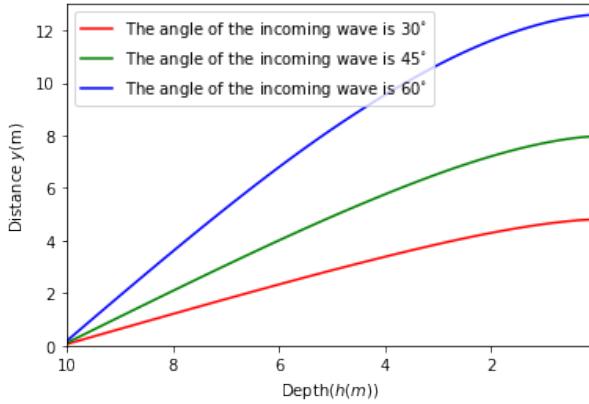


Figure 25: Graphs of Wave Directions at each Depth h when the change in x is $0.1m(\Delta x = 0.1m)$ and angles of the incoming waves are $30^\circ, 45^\circ, 60^\circ$

마찬가지로, 개별적인 입사파의 굴절각도 α_1 에 따라 음해법(Implicit Method) 중 이분법(Bisection Method)을 이용하여 산정한 파수 k 에 대해 수평방향으로의 이동거리 x 에 따른 파향(Wave Directions)에 대한 그래프를 각각 별도의 Figure로 구성하여 비교한 바 있는데, 이것 역시 하나의 Figure 내에서 살펴보면 다음과 같다.

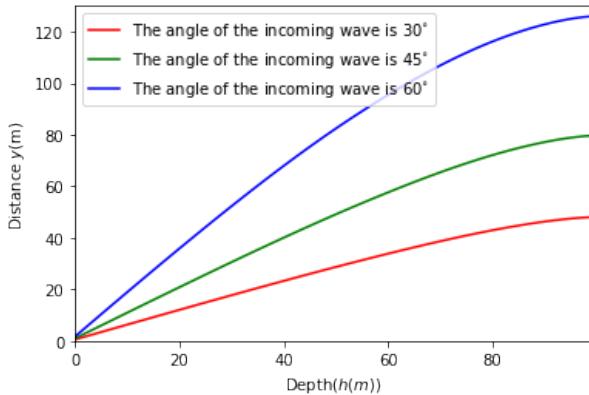


Figure 26: Graphs of Wave Directions at each Distance x from Origin (0, 0) when the change in x is $1m(\Delta x = 1m)$ and angles of the incoming waves are $30^\circ, 45^\circ, 60^\circ$

또한, 입사파의 굴절각도 α_1 마다 파랑의 굴절각도 α 를 두 방법에 의하여 추가 산정하여 수평방향으로의 이동거리의 변화량 Δx 와 수심의 변화량 Δh 가 $0.1m$ 로 동일하다고 가정($\Delta x = \Delta h = 0.1m$)한 상황에서 Figure 7 상의 좌표축과 원점을 차용하여 수심 h 에 대해 y 방향으로의 이동거리에 대한 변화량 Δy 를 누적하여, 연직방향으로의 이동거리 y 에 대한 좌표를 산정한 파향(Wave Directions)에 대한 그래프를 개별적으로 산정하여 비교한 바 있는데, 입사파에 따른 파향을 하나의 Figure내에서 이들을 비교해보면 다음과 같다.

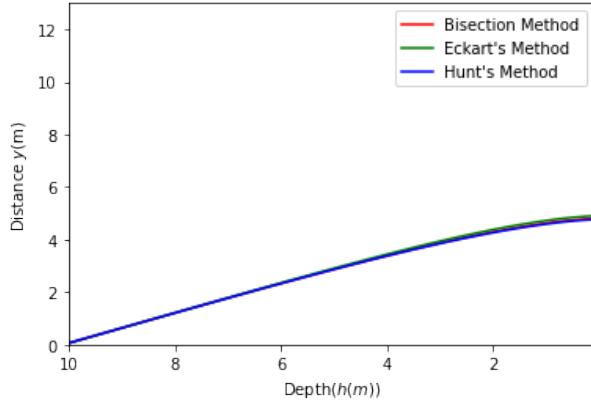


Figure 27: Graphs of Wave Directions at each Depth h when the change in x is $0.1m$ ($\Delta x = 0.1m$) and the angle of the incoming wave is 30°

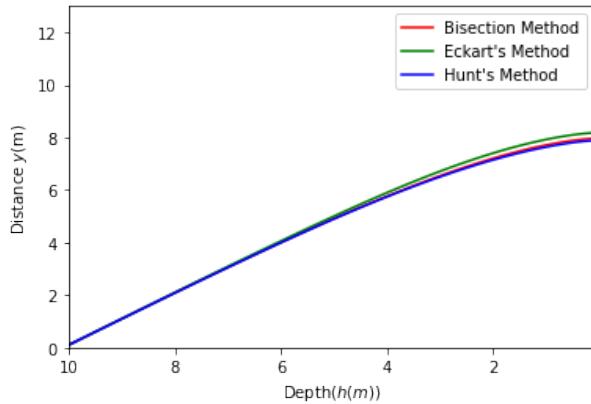


Figure 28: Graphs of Wave Directions at each Depth h when the change in x is $0.1m$ ($\Delta x = 0.1m$) and the angle of the incoming wave is 45°

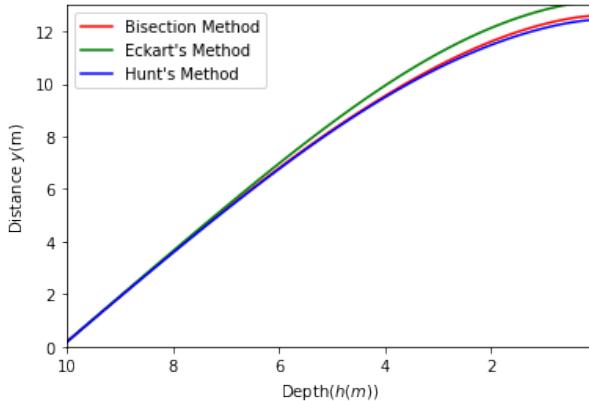


Figure 29: Graphs of Wave Directions at each Depth h when the change in x is $0.1m(\Delta x = 0.1m)$ and the angle of the incoming wave is 60°

위의 Figure 27, 28, 29을 통해 입사파의 각도가 커짐에 따라 세 가지 방법에 의한 파향에 큰 차이가 발생한다는 사실을 확인해볼 수 있다.

결론적으로, 본 보고서에서는 다양한 방법을 이용하여 분산관계식을 통해 파수 k 를 산정한 이후에는 파속 C 를 산정하고, 식 (1.17)식을 이용하여 임의지점에서의 파랑의 굴절각도 α 를 산정하는 과정과 이를 주어진 Figure 위에 그려봄으로써, 파의 진행경로를 예측해보는 과정이 수록되었다. 또한, 기본적인 해안공학의 개념들을 컴퓨터(Computers)내에서 코딩(Coding)을 통해 직접 구현해보며 이론적인 내용들을 재정립함과 동시에 실무적 활용 능력 또한 배양할 수 있었다.

4 Acknowledgement

본 과제를 부여해주신 Prof. Jung, T.H. 교수님께 감사의 말씀을 드립니다.

5 References

- [1] Jonghyeok Kim. *Code Files Written in Python by Kim, J.H. For Report III in Design of Coastal Engineering*. 2022. URL: <https://github.com/enfycius/Coastal-Engineering/blob/main/Assignment3.ipynb> (visited on 11/24/2022).
- [2] 정태화 이창훈. “파랑의 굴절 정도를 예측하는 해석기법”. In: 한국해안 · 해양공학회논문집 (2018), pp. 217–222.
- [3] 이창훈 장호철. “순환 관계에 의한 파랑분산식의 양해”. In: 대한토목학회 논문집 (2008), pp. 111–114.

Appendix A

Code Snippets

A Question #0

```
1 import math
2 import numpy as np

1 h0 = 10
2 g = 9.81

1 kh = math.pi
2 k = kh / h0

1 k0 = k

1 sqr_sigma = g * k * np.tanh([kh])

1 sqr_sigma

1 sigma = math.sqrt(sqr_sigma)

1 sigma

1 T0 = 2 * math.pi / sigma

1 T0

1 k0

1 L0 = 2 * math.pi / k0

1 L0
```

B Question #1

```
1 init_angle = 30

1 C1 = sigma / k0
```

```

1 C1

2 angle_bisection = []
3 angle_eckart = []
4 angle_hunt = []
5 degree_angle = []
6 tanv_bisection = []
7 tanv_hunt = []
8 tanv_eckart = []

9 h = []

10 bisection_k = []
11 eckart_k = []
12 hunt_k = []
13 hunt_y_30 = []

```

```

1 a1 = init_angle * np.pi / 180

```

```

1 h.append(h0)
2 angle_bisection.append(a1)
3 angle_eckart.append(a1)
4 angle_hunt.append(a1)
5 degree_angle.append(a1 * 180 / math.pi)
6 tanv_bisection.append(np.tan(a1))
7 tanv_eckart.append(np.tan(a1))
8 tanv_hunt.append(np.tan(a1))
9 bisection_k.append(k0)
10 eckart_k.append(k0)
11 hunt_k.append(k0)

```

```

1 d = [0.6666666666, 0.3555555555, 0.1608465608, 0.0632098765, 0.0217540484, 0.0076507983]

```

```

1 def f(x, sqr_omega, h1):
2     return g * (x / sqr_omega) * np.tanh([x*h1]) - 1

```

```

1 def bisection_method(k_init, sqr_omega, h1):
2     error=0.5 * 10**(-6)
3
4     a = -0.01 + k_init
5     b = 0.01 + k_init
6
7     while (b - a) / 2 > error:
8         c = (b + a) / 2

```

```

9     if f(c, sqr_omega, h1) == 0:
10        break
11    elif f(a, sqr_omega, h1)*f(c, sqr_omega, h1) > 0:
12        a = c
13    else:
14        b = c
15    c = (b + a) / 2
16
17    return c
18
19 hunt_y_30.append(sqr_sigma*h0/g)
20
21 for i in range(0, 99):
22     h1 = round(10 - (1/10)*(i+1), 2)
23     k_init = sqr_sigma / (g * math.sqrt(np.tanh(sqr_sigma / g * h1)))
24     k2 = abs(bisection_method(k_init, sqr_sigma, h1))
25     C2 = sigma / k2
26     a2_bisection = np.arcsin(C2 / C1 * np.sin([init_angle * np.pi / 180]))
27     y = sqr_sigma*h1/g
28     hunt_y_30.append(y)
29
30     dny = 0
31
32     for j in range(0, len(d)):
33         dny += d[j] * (y ** (j+1))
34
35     k1 = math.sqrt(((y) ** 2 + ((y) / (1 + dny))) / (h1 ** 2))
36     C2_Hunt = sigma / k1
37     C2_Eckart = sigma / k_init
38     a2_hunt = np.arcsin(C2_Hunt / C1 * np.sin([init_angle * np.pi / 180]))
39     a2_eckart = np.arcsin(C2_Eckart / C1 * np.sin([init_angle * np.pi / 180]))
40
41     hunt_k.append(k1)
42     eckart_k.append(k_init)
43     bisection_k.append(k2)
44     h.append(h1)
45     angle_bisection.append(a2_bisection)
46     angle_eckart.append(a2_eckart)
47     angle_hunt.append(a2_hunt)
48
49     tanv_bisection.append(np.tan(a2_bisection))
50     tanv_eckart.append(np.tan(a2_eckart))

```

```

31 tanv_hunt.append(np.tan(a2_hunt))

1 abs(eckart_k[1] - bisection_k[1])*100

1 len(h)

1 X = np.arange(0, 100)

1 DeltaX = round(abs(X[0] - X[1]), 1)

1 DeltaX

1 Deltah = round(abs(h[0] - h[1]), 1)

1 Deltah

1 Y_30 = []
2 x_Y_30 = []

1 added_x_Y_b_30 = []
2 added_x_Y_e_30 = []
3 added_x_Y_h_30 = []

1 added_h_Y_b_30 = []
2 added_h_Y_e_30 = []
3 added_h_Y_h_30 = []

1 added_x_Y_b_v_30 = 0
2 added_x_Y_e_v_30 = 0
3 added_x_Y_h_v_30 = 0

1 added_h_Y_b_v_30 = 0
2 added_h_Y_e_v_30 = 0
3 added_h_Y_h_v_30 = 0

1 len(tanv_bisection)

1 for i in range(0, len(tanv_bisection)):
2     added_x_Y_b_v_30 += (tanv_bisection[i] * DeltaX).item()
3     added_x_Y_e_v_30 += (tanv_eckart[i] * DeltaX).item()
4     added_x_Y_h_v_30 += (tanv_hunt[i] * DeltaX).item()
5     added_x_Y_b_30.append(added_x_Y_b_v_30)
6     added_x_Y_e_30.append(added_x_Y_e_v_30)
7     added_x_Y_h_30.append(added_x_Y_h_v_30)

```

```

1 for i in range(0, len(tanv_bisection)):
2     added_h_Y_b_v_30 += (tanv_bisection[i] * Deltah).item()
3     added_h_Y_e_v_30 += (tanv_eckart[i] * Deltah).item()
4     added_h_Y_h_v_30 += (tanv_hunt[i] * Deltah).item()
5     added_h_Y_b_30.append(added_h_Y_b_v_30)
6     added_h_Y_e_30.append(added_h_Y_e_v_30)
7     added_h_Y_h_30.append(added_h_Y_h_v_30)

```

```
1 added_h_Y_h_30
```

```

1 import matplotlib.pyplot as plt
2 import numpy as np
3 import math
4
5 # Plotting both the curves simultaneously
6 plt.plot(h, added_h_Y_b_30, color='r', label="y(m)")
7
8 # Naming the x-axis, y-axis and the whole graph
9 plt.xlabel("Depth(h(m))")
10 plt.ylabel("Distance y(m)")
11
12 plt.axis([max(h), min(h), min(added_h_Y_b_30), max(added_h_Y_b_30)])
13
14 plt.ylim(0, 13)
15
16 # To load the display window
17 plt.show()

```

```

1 import matplotlib.pyplot as plt
2 import numpy as np
3 import math
4
5 # Plotting both the curves simultaneously
6 plt.plot(X, added_x_Y_b_30, color='r', label="y(m)")
7
8 # Naming the x-axis, y-axis and the whole graph
9 plt.xlabel("Distance(x(m))")
10 plt.ylabel("Distance y(m)")
11
12 plt.axis([min(X), max(X), min(added_x_Y_b_30), max(added_x_Y_b_30)])
13
14 plt.ylim(0, 130)

```

```
15  
16 # To load the display window  
17 plt.show()
```

```
1 import matplotlib.pyplot as plt  
2 import numpy as np  
3 import math  
4  
5 # Plotting both the curves simultaneously  
6 plt.plot(X, added_x_Y_b_30, color='r', label="y$(m)$")  
7  
8 plt.axis([min(X), max(X), min(added_x_Y_b_30), max(added_x_Y_b_30)])  
9  
10 plt.ylim(0, 130)  
11  
12 # To load the display window  
13 plt.show()
```

```
1 import matplotlib.pyplot as plt  
2 import numpy as np  
3 import math  
4  
5 # Plotting both the curves simultaneously  
6 plt.plot(h, bisection_k, color='r', label='Bisection Method')  
7 plt.plot(h, eckart_k, color='b', label='Eckart')  
8 plt.plot(h, hunt_k, color='g', label='Hunt')  
9  
10 # Naming the x-axis, y-axis and the whole graph  
11 plt.xlabel("Depth($h$)")  
12 plt.ylabel("Wave Number($k$)")  
13 plt.title("Wave Number($k$) vs. Depth($h$)")  
14  
15 # Adding legend, which helps us recognize the curve according to it's color  
16 plt.legend()  
17 plt.xlim(0, 1)  
18  
19 # To load the display window  
20 plt.show()
```

```
1 abs_Hunt_k = 0  
2 abs_Eckart_k = 0
```

```

1 for i in range(0, len(hunt_k)):
2     abs_Hunt_k += abs(hunt_k[i] - bisection_k[i])
3     abs_Eckart_k += abs(eckart_k[i] - bisection_k[i])
4
5 mean_abs_Hunt_k = abs_Hunt_k / len(bisection_k)
6 mean_abs_Eckart_k = abs_Eckart_k / len(bisection_k)
7
8 mean_abs_Hunt_k * 100
9
10 mean_abs_Eckart_k * 100
11
12 Bisection_C = []
13 Eckart_C = []
14 Hunt_C = []
15
16 for i in range(0, len(bisection_k)):
17     Bisection_C.append((2 * math.pi / bisection_k[i])/T0)
18
19 for i in range(0, len(eckart_k)):
20     Eckart_C.append((2 * math.pi / eckart_k[i])/T0)
21
22 for i in range(0, len(hunt_k)):
23     Hunt_C.append((2 * math.pi / hunt_k[i])/T0)
24
25 import matplotlib.pyplot as plt
26 import numpy as np
27 import math
28
29 # Plotting both the curves simultaneously
30 plt.plot(h, Bisection_C, color='r', label='Bisection Method')
31 plt.plot(h, Eckart_C, color='g', label='Eckart')
32 plt.plot(h, Hunt_C, color='b', label='Hunt')
33
34 # Naming the x-axis, y-axis and the whole graph
35 plt.xlabel("Depth($h$)")
36 plt.ylabel("$C$")
37 plt.title("Wave Velocities($C$) vs. Depth($h$)")
38
39 # Adding legend, which helps us recognize the curve according to it's color
40 plt.legend()
41
42 # To load the display window
43 plt.show()

```

```

1 abs_Eckart_C = 0
2 abs_Hunt_C = 0

1 for i in range(0, len(Bisection_C)):
2     abs_Eckart_C += abs(Bisection_C[i] - Eckart_C[i])
3     abs_Hunt_C += abs(Bisection_C[i] - Hunt_C[i])

1 mean_abs_Eckart_C = abs_Eckart_C / len(Eckart_C)
2 mean_abs_Hunt_C = abs_Hunt_C / len(Hunt_C)

1 mean_abs_Eckart_C * 100

1 mean_abs_Hunt_C * 100

1 import matplotlib.pyplot as plt
2 import numpy as np
3 import math
4
5 # Plotting both the curves simultaneously
6 plt.plot(h, added_h_Y_e_30, color='r', label="$y$(m)")
7
8 # Naming the x-axis, y-axis and the whole graph
9 plt.xlabel("Depth($h$(m))")
10 plt.ylabel("Distance $y$(m)")
11
12 plt.axis([max(h), min(h), min(added_h_Y_e_30), max(added_h_Y_e_30)])
13
14 plt.ylim(0, 13)
15
16 # To load the display window
17 plt.show()

```

```

1 import matplotlib.pyplot as plt
2 import numpy as np
3 import math
4
5 # Plotting both the curves simultaneously
6 plt.plot(h, added_h_Y_h_30, color='r', label="$y$(m)")
7
8 # Naming the x-axis, y-axis and the whole graph
9 plt.xlabel("Depth($h$(m))")
10 plt.ylabel("Distance $y$(m)")
11

```

```

12
13 plt.axis([max(h),min(h),min(added_h_Y_h_30),max(added_h_Y_h_30)])
14
15 plt.ylim(0, 13)
16
17 # To load the display window
18 plt.show()

```

```

1 import matplotlib.pyplot as plt
2 import numpy as np
3 import math
4
5 # Plotting both the curves simultaneously
6 plt.plot(h, added_h_Y_b_30, color='r', label="Bisection Method")
7 plt.plot(h, added_h_Y_e_30, color='g', label="Eckart's Method")
8 plt.plot(h, added_h_Y_h_30, color='b', label="Hunt's Method")
9
10 # Naming the x-axis, y-axis and the whole graph
11 plt.xlabel("Depth($h(m)$)")
12 plt.ylabel("Distance $y$(m)")
13
14 plt.axis([max(h),min(h),min(added_h_Y_h_30),max(added_h_Y_h_30)])
15
16 plt.ylim(0, 13)
17
18 plt.legend()
19
20 # To load the display window
21 plt.show()

```

```

1 abs_Eckart_angle = 0
2 abs_Hunt_angle = 0

```

```

1 len(angle_bisection)

```

```

1 len(angle_eckart)

```

```

1 len(angle_hunt)

```

```

1 angle_bisection

```

```

1 angle_eckart

```

```

1 for i in range(0, len(angle_bisection)):
2     abs_Eckart_angle += abs(angle_bisection[i] - angle_eckart[i])
3     abs_Hunt_angle += abs(angle_bisection[i] - angle_hunt[i])
4
5 mean_abs_Eckart_angle = abs_Eckart_angle / len(angle_eckart)
6 mean_abs_Hunt_angle = abs_Hunt_angle / len(angle_hunt)
7
8 mean_abs_Eckart_angle * 100
9
10 mean_abs_Hunt_angle * 100

```

C Question #2

```

1 init_angle = 45
2
3 C1 = sigma / k0
4
5 C1
6
7 angle_bisection = []
8 angle_eckart = []
9 angle_hunt = []
10 degree_angle = []
11 tanv_bisection = []
12 tanv_hunt = []
13 tanv_eckart = []
14
15 h = []
16
17 bisection_k = []
18 eckart_k = []
19 hunt_k = []
20 hunt_y_45 = []
21
22 a1 = init_angle * np.pi / 180
23
24 h.append(h0)
25 angle_bisection.append(a1)
26 angle_eckart.append(a1)
27 angle_hunt.append(a1)
28 degree_angle.append(a1 * 180 / math.pi)
29 tanv_bisection.append(np.tan(a1))
30 tanv_eckart.append(np.tan(a1))
31 tanv_hunt.append(np.tan(a1))

```

```

9 bisection_k.append(k0)
10 eckart_k.append(k0)
11 hunt_k.append(k0)

1 d = [0.6666666666, 0.3555555555, 0.1608465608, 0.0632098765, 0.0217540484, 0.0076507983]

1 def f(x, sqr_omega, h1):
2     return g * (x / sqr_omega) * np.tanh([x*h1]) - 1

1 def bisection_method(k_init, sqr_omega, h1):
2     error=0.5 * 10**(-6)

3
4     a = -0.01 + k_init
5     b = 0.01 + k_init
6
7     while (b - a) / 2 > error:
8         c = (b + a) / 2
9         if f(c, sqr_omega, h1) == 0:
10            break
11        elif f(a, sqr_omega, h1)*f(c, sqr_omega, h1) > 0:
12            a = c
13        else:
14            b = c
15        c = (b + a) / 2
16
17    return c

1 hunt_y_45.append(sqr_sigma*h0/g)

1 for i in range(0, 99):
2     h1 = round(10 - (1/10)*(i+1), 2)
3     k_init = sqr_sigma / (g * math.sqrt(np.tanh(sqr_sigma / g * h1)))
4     k2 = abs(bisection_method(k_init, sqr_sigma, h1))
5     C2 = sigma / k2
6     a2_bisection = np.arcsin(C2 / C1 * np.sin([init_angle * np.pi / 180]))
7     y = sqr_sigma*h1/g
8     hunt_y_45.append(y)
9
10    dny = 0
11
12    for j in range(0, len(d)):
13        dny += d[j] * (y ** (j+1))
14

```

```

15     k1 = math.sqrt(((y) ** 2 + ((y) / (1 + dny))) / (h1 ** 2))
16     C2_Hunt = sigma / k1
17     C2_Eckart = sigma / k_init
18     a2_hunt = np.arcsin(C2_Hunt / C1 * np.sin([init_angle * np.pi / 180]))
19     a2_eckart = np.arcsin(C2_Eckart / C1 * np.sin([init_angle * np.pi / 180]))
20
21     hunt_k.append(k1)
22     eckart_k.append(k_init)
23     bisection_k.append(k2)
24     h.append(h1)
25     angle_bisection.append(a2_bisection)
26     angle_eckart.append(a2_eckart)
27     angle_hunt.append(a2_hunt)
28
29     tanv_bisection.append(np.tan(a2_bisection))
30     tanv_eckart.append(np.tan(a2_eckart))
31     tanv_hunt.append(np.tan(a2_hunt))

```

```
1 abs(eckart_k[1] - bisection_k[1])*100
```

```
1 angle_bisection
```

```
1 angle_bisection[1] * 180 / math.pi
```

```
1 tanv_bisection[1]
```

```
1 len(h)
```

```
1 X = np.arange(0, 100)
```

```
1 DeltaX = round(abs(X[0] - X[1]), 1)
```

```
1 DeltaX
```

```
1 Deltah = round(abs(h[0] - h[1]), 1)
```

```
1 Deltah
```

```
1 Y_45 = []
```

```
2 x_Y_45 = []
```

```
1 added_x_Y_b_45 = []
```

```
2 added_x_Y_e_45 = []
```

```
3 added_x_Y_h_45 = []
```

```
1 added_h_Y_b_45 = []
2 added_h_Y_e_45 = []
3 added_h_Y_h_45 = []
```

```
1 added_x_Y_b_v_45 = 0
2 added_x_Y_e_v_45 = 0
3 added_x_Y_h_v_45 = 0
```

```
1 added_h_Y_b_v_45 = 0
2 added_h_Y_e_v_45 = 0
3 added_h_Y_h_v_45 = 0
```

```
1 len(tanv_bisection)
```

```
1 for i in range(0, len(tanv_bisection)):
2     added_x_Y_b_v_45 += (tanv_bisection[i] * DeltaX).item()
3     added_x_Y_e_v_45 += (tanv_eckart[i] * DeltaX).item()
4     added_x_Y_h_v_45 += (tanv_hunt[i] * DeltaX).item()
5     added_x_Y_b_45.append(added_x_Y_b_v_45)
6     added_x_Y_e_45.append(added_x_Y_e_v_45)
7     added_x_Y_h_45.append(added_x_Y_h_v_45)
```

```
1 for i in range(0, len(tanv_bisection)):
2     added_h_Y_b_v_45 += (tanv_bisection[i] * Deltah).item()
3     added_h_Y_e_v_45 += (tanv_eckart[i] * Deltah).item()
4     added_h_Y_h_v_45 += (tanv_hunt[i] * Deltah).item()
5     added_h_Y_b_45.append(added_h_Y_b_v_45)
6     added_h_Y_e_45.append(added_h_Y_e_v_45)
7     added_h_Y_h_45.append(added_h_Y_h_v_45)
```

```
1 added_h_Y_h_45
```

```
1 import matplotlib.pyplot as plt
2 import numpy as np
3 import math
4
5 # Plotting both the curves simultaneously
6 plt.plot(h, added_h_Y_b_45, color='r', label="$y$(m)")
7
8 # Naming the x-axis, y-axis and the whole graph
9 plt.xlabel("Depth($h$(m))")
10 plt.ylabel("Distance $y$(m)")
11
```

```

12 plt.axis([max(h),min(h),min(added_x_Y_b_45),max(added_x_Y_b_45)])
13
14 plt.ylim(0, 13)
15
16 # To load the display window
17 plt.show()

```

```

1 import matplotlib.pyplot as plt
2 import numpy as np
3 import math
4
5 # Plotting both the curves simultaneously
6 plt.plot(X, added_x_Y_b_45, color='r', label="$y$(m)")
7
8 # Naming the x-axis, y-axis and the whole graph
9 plt.xlabel("Distance($x$(m))")
10 plt.ylabel("Distance $y$(m)")
11
12 plt.axis([min(X),max(X),min(added_x_Y_b_45),max(added_x_Y_b_45)])
13
14 plt.ylim(0, 130)
15
16 # To load the display window
17 plt.show()

```

```

1 import matplotlib.pyplot as plt
2 import numpy as np
3 import math
4
5 # Plotting both the curves simultaneously
6 plt.plot(X, added_x_Y_b_45, color='r', label="$y$(m)")
7
8 plt.axis([min(X),max(X),min(added_x_Y_b_45),max(added_x_Y_b_45)])
9
10 plt.ylim(0, 130)
11
12 # To load the display window
13 plt.show()

```

```

1 h

```

```

1 import matplotlib.pyplot as plt

```

```

2 import numpy as np
3 import math
4
5 # Plotting both the curves simultaneously
6 plt.plot(h, added_h_Y_e_45, color='r', label="$y$(m)")
7
8 # Naming the x-axis, y-axis and the whole graph
9 plt.xlabel("Depth($h$(m))")
10 plt.ylabel("Distance $y$(m)")
11
12 plt.axis([max(h), min(h), min(added_h_Y_e_45), max(added_h_Y_e_45)])
13
14 plt.ylim(0, 13)
15
16 # To load the display window
17 plt.show()

```

```

1 import matplotlib.pyplot as plt
2 import numpy as np
3 import math
4
5 # Plotting both the curves simultaneously
6 plt.plot(h, added_h_Y_h_45, color='r', label="$y$(m)")
7
8 # Naming the x-axis, y-axis and the whole graph
9 plt.xlabel("Depth($h$(m))")
10 plt.ylabel("Distance $y$(m)")
11
12 plt.axis([max(h), min(h), min(added_h_Y_h_45), max(added_h_Y_h_45)])
13
14 plt.ylim(0, 13)
15
16 # To load the display window
17 plt.show()
18

```

```

1 import matplotlib.pyplot as plt
2 import numpy as np
3 import math
4
5 # Plotting both the curves simultaneously
6 plt.plot(h, added_h_Y_b_45, color='r', label="Bisection Method")

```

```

7 plt.plot(h, added_h_Y_e_45, color='g', label="Eckart's Method")
8 plt.plot(h, added_h_Y_h_45, color='b', label="Hunt's Method")
9
10 # Naming the x-axis, y-axis and the whole graph
11 plt.xlabel("Depth($h(m)$)")
12 plt.ylabel("Distance $y$(m)")
13
14 plt.axis([max(h),min(h),min(added_h_Y_h_45),max(added_h_Y_h_45)])
15
16 plt.ylim(0, 13)
17
18 plt.legend()
19
20 # To load the display window
21 plt.show()

```

```

1 abs_Eckart_angle = 0
2 abs_Hunt_angle = 0

1 len(angle_bisection)

1 len(angle_eckart)

1 len(angle_hunt)

1 angle_bisection

1 angle_eckart

1 for i in range(0, len(angle_bisection)):
2     abs_Eckart_angle += abs(angle_bisection[i] - angle_eckart[i])
3     abs_Hunt_angle += abs(angle_bisection[i] - angle_hunt[i])

1 mean_abs_Eckart_angle = abs_Eckart_angle / len(angle_eckart)
2 mean_abs_Hunt_angle = abs_Hunt_angle / len(angle_hunt)

1 mean_abs_Eckart_angle * 100

1 mean_abs_Hunt_angle * 100

```

D Question #3

```

1 init_angle = 60

```

```

1 C1 = sigma / k0

1 C1

1 angle_bisection = []
2 angle_eckart = []
3 angle_hunt = []
4 degree_angle = []
5 tanv_bisection = []
6 tanv_hunt = []
7 tanv_eckart = []

8

9 h = []

10 bisection_k = []
11 eckart_k = []
12 hunt_k = []
13 hunt_y_60 = []

```

```

1 a1 = init_angle * np.pi / 180

1 h.append(h0)
2 angle_bisection.append(a1)
3 angle_eckart.append(a1)
4 angle_hunt.append(a1)
5 degree_angle.append(a1 * 180 / math.pi)
6 tanv_bisection.append(np.tan(a1))
7 tanv_eckart.append(np.tan(a1))
8 tanv_hunt.append(np.tan(a1))
9 bisection_k.append(k0)
10 eckart_k.append(k0)
11 hunt_k.append(k0)

```

```

1 d = [0.6666666666, 0.3555555555, 0.1608465608, 0.0632098765, 0.0217540484, 0.0076507983]

```

```

1 def f(x, sqr_omega, h1):
2     return g * (x / sqr_omega) * np.tanh([x*h1]) - 1

```

```

1 def bisection_method(k_init, sqr_omega, h1):
2     error=0.5 * 10**(-6)

3

4     a = -0.01 + k_init
5     b = 0.01 + k_init
6

```

```

7     while (b - a) / 2 > error:
8         c = (b + a) / 2
9         if f(c, sqr_omega, h1) == 0:
10            break
11        elif f(a, sqr_omega, h1)*f(c, sqr_omega, h1) > 0:
12            a = c
13        else:
14            b = c
15        c = (b + a) / 2
16
17    return c

1 hunt_y_60.append(sqr_sigma*h0/g)

1 for i in range(0, 99):
2     h1 = round(10 - (1/10)*(i+1), 2)
3     k_init = sqr_sigma / (g * math.sqrt(np.tanh(sqr_sigma / g * h1)))
4     k2 = abs(bisection_method(k_init, sqr_sigma, h1))
5     C2 = sigma / k2
6     a2_bisection = np.arcsin(C2 / C1 * np.sin([init_angle * np.pi / 180]))
7     y = sqr_sigma*h1/g
8     hunt_y_60.append(y)

9
10    dny = 0
11
12    for j in range(0, len(d)):
13        dny += d[j] * (y ** (j+1))
14
15    k1 = math.sqrt(((y) ** 2 + ((y) / (1 + dny))) / (h1 ** 2))
16    C2_Hunt = sigma / k1
17    C2_Eckart = sigma / k_init
18    a2_hunt = np.arcsin(C2_Hunt / C1 * np.sin([init_angle * np.pi / 180]))
19    a2_eckart = np.arcsin(C2_Eckart / C1 * np.sin([init_angle * np.pi / 180]))
20
21    hunt_k.append(k1)
22    eckart_k.append(k_init)
23    bisection_k.append(k2)
24    h.append(h1)
25    angle_bisection.append(a2_bisection)
26    angle_eckart.append(a2_eckart)
27    angle_hunt.append(a2_hunt)
28
```

```
29     tanv_bisection.append(np.tan(a2_bisection))
30     tanv_eckart.append(np.tan(a2_eckart))
31     tanv_hunt.append(np.tan(a2_hunt))
```

```
1 tanv_bisection
```

```
1 angle_bisection
```

```
1 bisection_k
```

```
1 abs(eckart_k[1] - bisection_k[1])*100
```

```
1 len(h)
```

```
1 X = np.arange(0, 100)
```

```
1 DeltaX = round(abs(X[0] - X[1]), 1)
```

```
1 DeltaX
```

```
1 Deltah = round(abs(h[0] - h[1]), 1)
```

```
1 Deltah
```

```
1 Y_60 = []
```

```
2 x_Y_60 = []
```

```
1 added_x_Y_b_60 = []
```

```
2 added_x_Y_e_60 = []
```

```
3 added_x_Y_h_60 = []
```

```
1 added_h_Y_b_60 = []
```

```
2 added_h_Y_e_60 = []
```

```
3 added_h_Y_h_60 = []
```

```
1 added_x_Y_b_v_60 = 0
```

```
2 added_x_Y_e_v_60 = 0
```

```
3 added_x_Y_h_v_60 = 0
```

```
1 added_h_Y_b_v_60 = 0
```

```
2 added_h_Y_e_v_60 = 0
```

```
3 added_h_Y_h_v_60 = 0
```

```
1 len(tanv_bisection)
```

```

1 for i in range(0, len(tanv_bisection)):
2     added_x_Y_b_v_60 += (tanv_bisection[i] * DeltaX).item()
3     added_x_Y_e_v_60 += (tanv_eckart[i] * DeltaX).item()
4     added_x_Y_h_v_60 += (tanv_hunt[i] * DeltaX).item()
5     added_x_Y_b_60.append(added_x_Y_b_v_60)
6     added_x_Y_e_60.append(added_x_Y_e_v_60)
7     added_x_Y_h_60.append(added_x_Y_h_v_60)

```

```

1 for i in range(0, len(tanv_bisection)):
2     added_h_Y_b_v_60 += (tanv_bisection[i] * Deltah).item()
3     added_h_Y_e_v_60 += (tanv_eckart[i] * Deltah).item()
4     added_h_Y_h_v_60 += (tanv_hunt[i] * Deltah).item()
5     added_h_Y_b_60.append(added_h_Y_b_v_60)
6     added_h_Y_e_60.append(added_h_Y_e_v_60)
7     added_h_Y_h_60.append(added_h_Y_h_v_60)

```

```
1 added_h_Y_h_60
```

```

1 import matplotlib.pyplot as plt
2 import numpy as np
3 import math
4
5 # Plotting both the curves simultaneously
6 plt.plot(h, added_h_Y_b_60, color='r', label="$y$(m)")
7
8 # Naming the x-axis, y-axis and the whole graph
9 plt.xlabel("Depth($h$) $m$")
10 plt.ylabel("Distance $y$(m)")
11
12 plt.axis([max(h), min(h), min(added_h_Y_b_60), max(added_h_Y_b_60)])
13
14 plt.ylim(0, 13)
15
16 # To load the display window
17 plt.show()

```

```

1 import matplotlib.pyplot as plt
2 import numpy as np
3 import math
4
5 # Plotting both the curves simultaneously
6 plt.plot(X, added_x_Y_b_60, color='r', label="$y$(m)")

```

```

7
8 # Naming the x-axis, y-axis and the whole graph
9 plt.xlabel("Distance($x(m)$)")
10 plt.ylabel("Distance $y$(m)")
11
12 plt.axis([min(X),max(X),min(added_x_Y_b_60),max(added_x_Y_b_60)])
13
14 plt.ylim(0, 130)
15
16 # To load the display window
17 plt.show()

```

```

1 import matplotlib.pyplot as plt
2 import numpy as np
3 import math
4
5 # Plotting both the curves simultaneously
6 plt.plot(X, added_x_Y_b_60, color='r', label="$y$(m)")
7
8 plt.axis([min(X),max(X),min(added_x_Y_b_60),max(added_x_Y_b_60)])
9
10 plt.ylim(0, 130)
11
12 # To load the display window
13 plt.show()

```

```

1 h
2
3
4
5 # Plotting both the curves simultaneously
6 plt.plot(h, added_h_Y_e_60, color='r', label="$y$(m)")
7
8 # Naming the x-axis, y-axis and the whole graph
9 plt.xlabel("Depth($h(m)$)")
10 plt.ylabel("Distance $y$(m)")
11
12 plt.axis([max(h),min(h),min(added_h_Y_e_60),max(added_h_Y_e_60)])
13
14 plt.ylim(0, 13)

```

```

15
16 # To load the display window
17 plt.show()

1 import matplotlib.pyplot as plt
2 import numpy as np
3 import math
4
5 # Plotting both the curves simultaneously
6 plt.plot(h, added_h_Y_h_60, color='r', label="$y$(m)")
7
8
9 # Naming the x-axis, y-axis and the whole graph
10 plt.xlabel("Depth($h$(m))")
11 plt.ylabel("Distance $y$(m)")
12
13 plt.axis([max(h), min(h), min(added_h_Y_h_60), max(added_h_Y_h_60)])
14
15 plt.ylim(0, 13)
16
17 # To load the display window
18 plt.show()

```

```

1 import matplotlib.pyplot as plt
2 import numpy as np
3 import math
4
5 # Plotting both the curves simultaneously
6 plt.plot(h, added_h_Y_b_60, color='r', label="Bisection Method")
7 plt.plot(h, added_h_Y_e_60, color='g', label="Eckart's Method")
8 plt.plot(h, added_h_Y_h_60, color='b', label="Hunt's Method")
9
10 # Naming the x-axis, y-axis and the whole graph
11 plt.xlabel("Depth($h$(m))")
12 plt.ylabel("Distance $y$(m)")
13
14 plt.axis([max(h), min(h), min(added_h_Y_h_60), max(added_h_Y_h_60)])
15
16 plt.ylim(0, 13)
17
18 plt.legend()
19

```

```

20 # To load the display window
21 plt.show()

1 import matplotlib.pyplot as plt
2 import numpy as np
3 import math
4
5 # Plotting both the curves simultaneously
6 plt.plot(h, added_h_Y_b_30, color='r', label="The angle of the incoming wave is $30^{\circ}$")
7 plt.plot(h, added_h_Y_b_45, color='g', label="The angle of the incoming wave is $45^{\circ}$")
8 plt.plot(h, added_h_Y_b_60, color='b', label="The angle of the incoming wave is $60^{\circ}$")
9
10 # Naming the x-axis, y-axis and the whole graph
11 plt.xlabel("Depth($h$(m))")
12 plt.ylabel("Distance $y$(m)")
13
14 plt.axis([max(h),min(h),min(added_h_Y_h_60),max(added_h_Y_h_60)])
15
16 plt.ylim(0, 13)
17
18 plt.legend()
19
20 # To load the display window
21 plt.show()

```

```

1 import matplotlib.pyplot as plt
2 import numpy as np
3 import math
4
5 # Plotting both the curves simultaneously
6 plt.plot(X, added_x_Y_b_30, color='r', label="The angle of the incoming wave is $30^{\circ}$")
7 plt.plot(X, added_x_Y_b_45, color='g', label="The angle of the incoming wave is $45^{\circ}$")
8 plt.plot(X, added_x_Y_b_60, color='b', label="The angle of the incoming wave is $60^{\circ}$")
9
10 # Naming the x-axis, y-axis and the whole graph
11 plt.xlabel("Depth($h$(m))")
12 plt.ylabel("Distance $y$(m)")
13
14 plt.axis([min(X),max(X),min(added_x_Y_h_60),max(added_x_Y_h_60)])
15
16 plt.ylim(0, 130)
17

```

```

18 plt.legend()
19
20 # To load the display window
21 plt.show()

1 abs_Eckart_angle = 0
2 abs_Hunt_angle = 0

1 len(angle_bisection)

1 len(angle_eckart)

1 len(angle_hunt)

1 angle_bisection

1 angle_eckart

1 for i in range(0, len(angle_bisection)):
2     abs_Eckart_angle += abs(angle_bisection[i] - angle_eckart[i])
3     abs_Hunt_angle += abs(angle_bisection[i] - angle_hunt[i])

1 mean_abs_Eckart_angle = abs_Eckart_angle / len(angle_eckart)
2 mean_abs_Hunt_angle = abs_Hunt_angle / len(angle_hunt)

1 mean_abs_Eckart_angle * 100

1 mean_abs_Hunt_angle * 100

1 import matplotlib.pyplot as plt
2 import numpy as np
3 import math
4
5 # Plotting both the curves simultaneously
6 plt.plot(h, hunt_y_30, color='r', label='The y value of the Hunt method.')
7
8 # Naming the x-axis, y-axis and the whole graph
9 plt.xlabel("Index")
10 plt.ylabel("$y$")
11 plt.title("y value")
12
13 ymax = max(hunt_y_30)
14
15 s = 'The maximum value of y= ' + str(ymax) + ', '

```

```
16
17 plt.annotate(s, (3, 0))
18
19 # Adding legend, which helps us recognize the curve according to it's color
20 plt.legend()
21
22 # To load the display window
23 plt.show()
```