1. 서론

원 신호 2\*sin(2 \* PI \* 100 \* t)을 0~0.1 sec 구간에 대하여 미분과 적분을 시행하는데, 이론적으로 계산되는 값과 컴퓨터가 계산하는 값을 비교하여 SNR(Signal to Noise Ratio)를 구해보고자 한다.

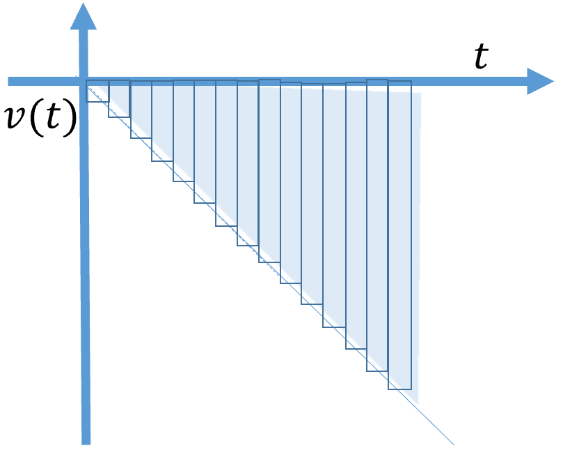
또한 dt값을 변화시켜가며, 다양한 dt값에 따른 SNR값을 비교해본다.

2. Integration

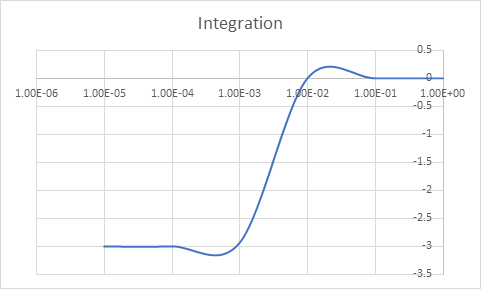
적분의 경우, 이론적으로 원 신호에 부정적분을 시행한 식은

(-1 / (100 \* PI))\*cos(2 \* PI \* 100 \* t) 이다.

그러나, C++을 통해 원 신호를 직접 적분하여 위의 식을 도출하기는 어려우므로, 구분 구적법

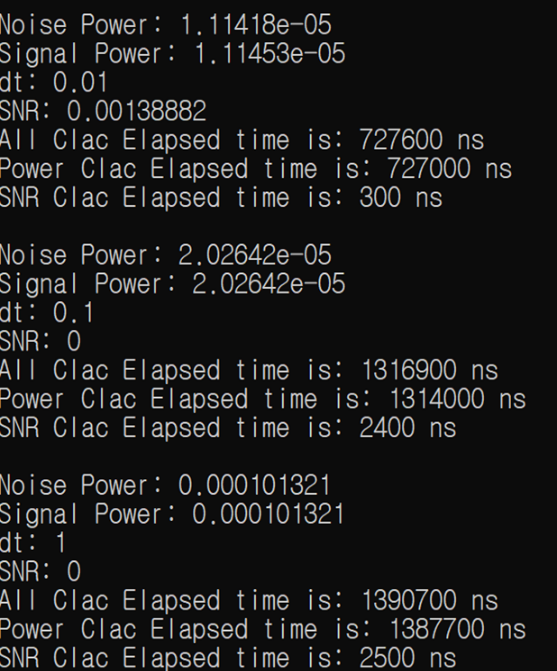
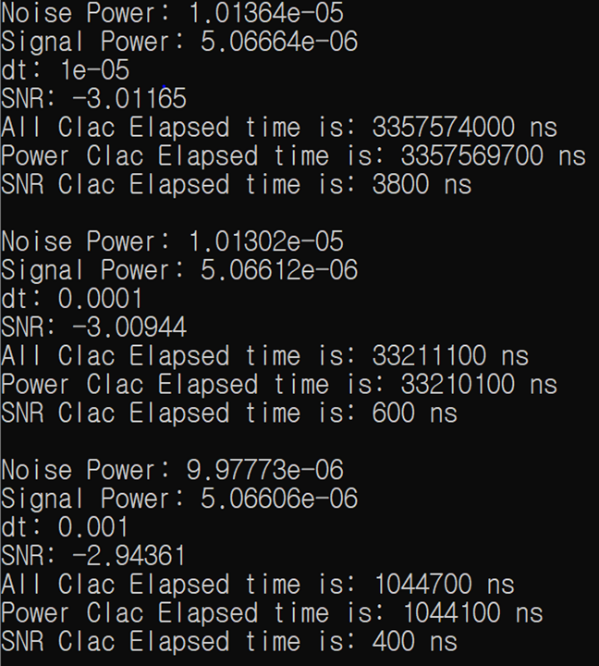
을 이용하여 유사한 값을 구하고자 한다.

0.00001~1까지 dt를 변화시켜가며 그에 따른 SNR값을 도표로 나타낸 것이다.

(더욱 다양한 dt에 대하여 실행해보고자 하였으나, 컴퓨터 성능의 문제인지 dt를 0.00001보다 작게 설정하였을 경우 콘솔 창에 아무것도 뜨지 않았다.)

(x축은 log scale)

-값을 가지던 SNR값이 dt값이 점차 증가하며 같이 증가하다가, 0으로 수렴하는, log함수 그래프의 형태를 보였다.



SNR값이 음수라는 것은 Signal power보다 Noise power가 더 크다는 의미이고, Noise에 다른 조치를 취하지 않는다면, 그 신호를 이용하는 통신에 대한 신뢰성은 매우 낮을 것이다.

위의 경우, 0~0.1sec에 대하여 적분을 시행했기 때문에 dt가 0.1이상일 경우에는 이론과 다른 결과가 도출될 가능성이 높을 것이다. dt가 0.1이상인 경우 SNR은 항상 0인 결과를 보였다.

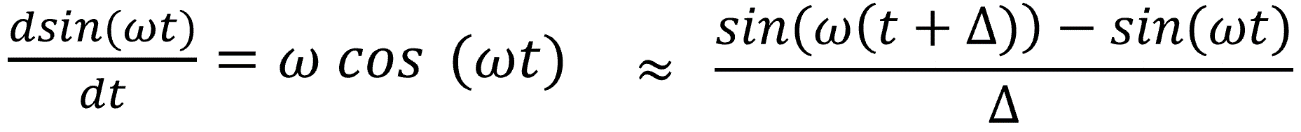
이론적으로 SNR이 0인 경우에는 Signal power와 Noise power가 같다는 것인데, 이런 경우 원 신호의 50%정도만 전달된다고 볼 수 있다.

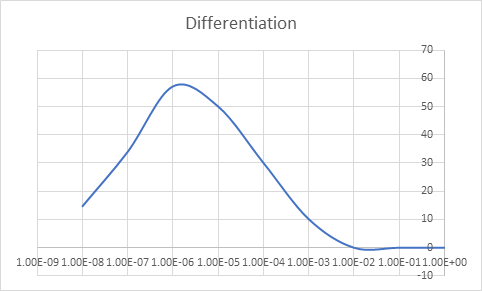
또한, chrono 라이브러리를 통해 각 과정의 Computation time도 구해보았다.

dt가 작아질수록 컴퓨터가 계산하는데 걸리는 시간은 증가하는 모습을 보였다. 이론적으로도 dt가 작아질수록 적분 과정에서 구해야 하는 넓이의 수가 급격히 증가하므로 계산에 걸리는 시간 역시 증가한다. 반면 dt가 0.1 이상인 경우에는 이러한 양상을 보이지 않았는데, 앞서 말했듯이 위의 경우 0~0.1sec 구간에 대해서 적분을 시행했기 때문일 것이다.

2. Differentiation

미분의 경우, 이론적으로 미분을 계산한 식은 400 \* PI\*cos(2 \* PI \* 100 \* t) 이다.

하지만, C++를 이용해 원 신호를 직접 미분하여 위의 식을 도출하기 어려우므로 difference를 이용하여 유사한 값을 구하도록 한다.

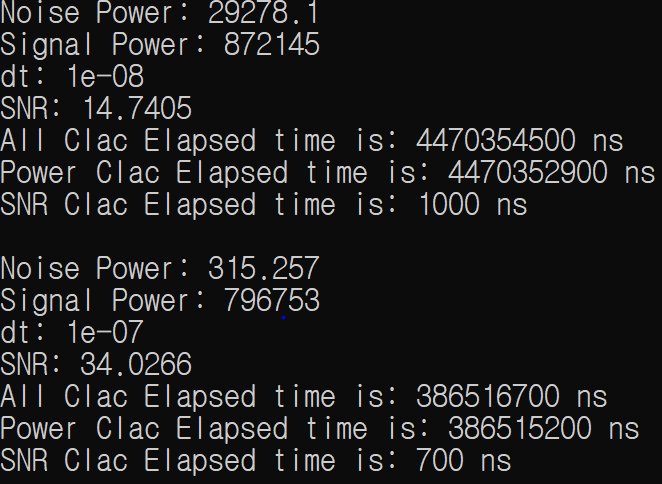
0.00000001부터 1까지 10배씩 dt를 변화시켜가며 그에 따른 SNR 값을 도표로 만든 것이다.

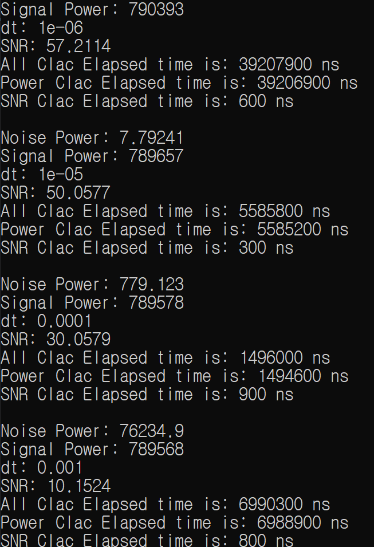
(적분과 달리 미분에서는 더욱 작은 dt에 대해서도 느리지만 콘솔 창에 출력이 되었다.)

(x축은 log scale)

1부터 dt가 10배씩 감소하면서 0.000001까지는 점점 SNR 값이 증가하였고, 이는 dt가 작아질수록 이론적인 값과 컴퓨터가 계산한 값의 차이가 작았다는 것을 의미한다.

그러나, dt가 이후로 더욱 작아짐에 따라 SNR 값도 감소하는 경향을 보였다.

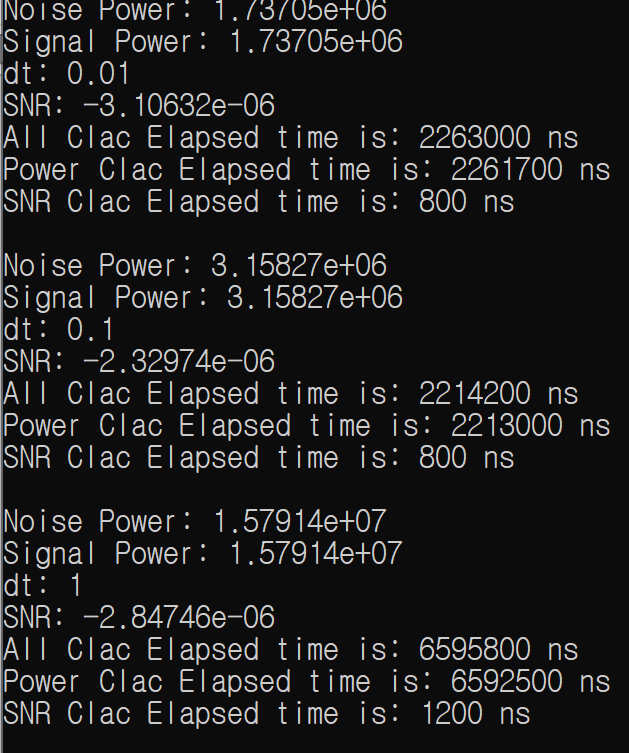




위의 두 사진에서 보면 알 수 있듯이, SNR이 증가하는 구간에서는 Signal Power가 약 790000정도의 값을 갖는 반면에, 감소하기 시작하는 구간에서는 급격히 증가하는 양상을 보인다. Signal Power와 Noise Power를 구하는 과정에서 적분을 하게 되는데, 이 적분을 C++을 이용해서 구분적분법으로 하다보니 일정 dt보다 더 작아지게 되면 오히려 왜곡된 값을 가진다고 추측할 수 있을 것 이다.

적분과 마찬가지로 chrono 라이브러리를 통해 각 과정의 Computation time을 구해보았다.

dt가 작아질수록 컴퓨터가 계산해야하는 과정이 복잡해지며 computation time이 증가하는 하여야 하는데, dt가 0.0001 이하일 경우에는 그러한 양상을 보였다.

하지만, 0.0001보다 클 경우에는 그러한 양상이 보이지 않았다. 그 이유를 고민해 보았지만 정확한 이유를 알 수가 없어, 추후 질문을 통해 해결하고자 한다.

4. 결론

지금까지 C++을 통해 신호를 미분, 적분해 보고 조건을 변화시켜가며 SNR값과 Computation time을 비교해 보았다.

저번 주에 다루었던 삼각함수와 달리 간단하게 함수를 이용해 계산을 할 수 없었는데, 근사값을 통해 유사한 결과를 얻는 방법을 익힐 수 있었다.

SNR 값을 구하는 것을 matlab을 이용하여 다른 강의에서 해보았는데, 전공과 관련된 수학과 거리가 멀어 보이던 C++을 통해 전공 내용들을 다루어보니 흥미로웠다.

dt의 값을 10배씩 변화 시키는 것을 가볍게 생각하고 있었는데 이번 과제에서 컴파일을 했을 때 결과가 늦게 뜨거나 아예 뜨지 않는 것과 dt에 따른 결과값의 변화를 보며 생각을 고치게 되었다. 그리고, 이전까지 프로그래밍 강의를 들으며 코딩을 간결하고 효율적으로 하는것의 중요성에 대해 배워왔지만, 수업시간에 하던 내용만으로 직접 경험해보지 못해 와닿는 느낌이 아니었는데, 그 부분에 대해서도 좀 더 주의하게 되었다.