One-Dimensional Phase Unwrapping Problem

By Dr. Munther Gdeisat and Dr. Francis Lilley

1. Introduction to one-dimensional phase unwrapping

There are many digital signal-processing techniques that use the four quadrant arctangent function to calculate the phase of a signal. You should be aware that there are many algorithms that are available to calculate the phase of a signal, however the explanation of these algorithms is beyond the objectives of this tutorial which concentrates solely on the phase unwrapping process. The amplitude of the phase that is calculated can take any value and typically exceeds the range [-π,π] that is returned by the arctangent function. In cases where the phase exceeds this range of values, it will be wrapped so that it stays within the normal range [-π,π]. In such cases the wrapped phase will contain one, or more 2π jumps.

Suppose that we have a discrete signal *x(n)* whose amplitude exceeds the range [-π,π]. Sometimes here we shall refer to the function *x(n)* as simply *x* for brevity. Also, we shall refer to the four quadrant arctangent function as atan2. Please note that we are dealing here with discrete signals. We can wrap the signal *x(n)* as follows.

1. Calculate the sinusoidal value of *x*
2. Calculate the cosinusoidal value of *x*
3. Calculate the four quadrant arctangent (atan2) of $\sin(x)$ and $\cos(x)$.

The four quadrant arctangent function can be calculated using the following equation;

$$atan2(a,b) = \begin{cases} tan^{-1}\left[\frac{a}{b}\right] & a > 0 \; and \; b > 0 \quad first \; quadrant \\ tan^{-1}\left[\frac{a}{b}\right] + \pi & a > 0 \; and \; b < 0 \quad second \; quadrant \\ tan^{-1}\left[\frac{a}{b}\right] - \pi & a < 0 \; and \; b < 0 \quad third \; quadrant \\ tan^{-1}\left[\frac{a}{b}\right] & a < 0 \; and \; b > 0 \quad fourth \; quadrant \end{cases}$$

Where a and b are real numbers.

We can express the wrapping process mathematically as,

$$x_w(n) = \mathcal{W}[x(n)]$$

Where *x(n)* is the original continuous phase signal, $\mathcal{W}[\,]$is the phase wrapping operation and $x_w(n)$ is the wrapped phase signal.

We will use Matlab software to perform the required computer simulations. Let us construct a signal $x(n)$ whose amplitude exceeds the range [-π,π]. Then we will wrap this signal.

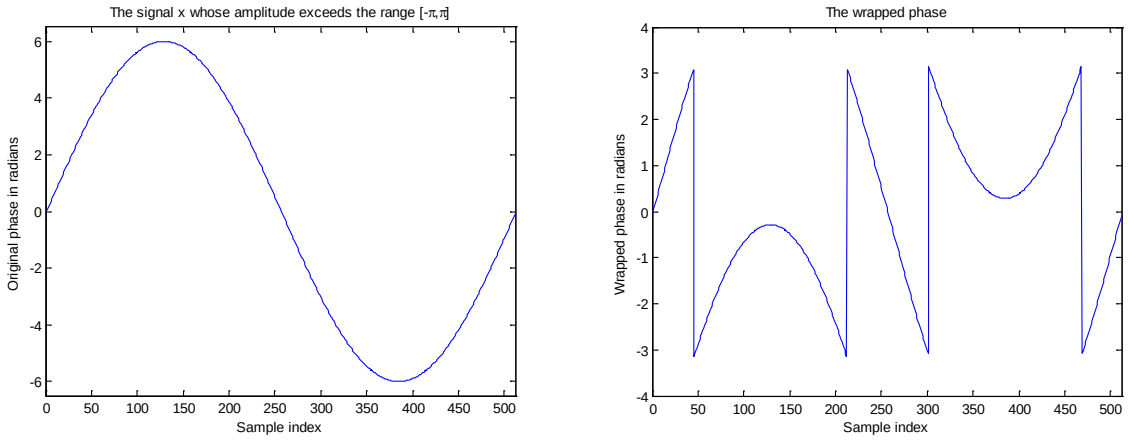
The Matlab code to perform this task is given below;

```
%This program is to simulate the wrapping process
clc; close all; clear
```

```
%The number of samples in the signal
N=512;
n=0:N-1;
fo=1/512;

% The signal x whose amplitude exceeds the range [-pi,pi]
x = 6*sin(2*pi*fo*n);
plot(x)
xlabel('Sample index')
ylabel('Original phase in radians')
axis([0 512 -6.5 6.5])
title('The signal x whose amplitude exceeds the range [-\pi,\pi]')

%Calculating the wrapped signal using the four quadrant arctangent function
xw = atan2(sin(x), cos(x));
figure, plot(xw)
xlabel('Sample index')
ylabel('Wrapped phase in radians')
axis([0 512 -4 4])
title('The wrapped phase')
```



(a) (b)

Figure 1: (a) Continuous phase, (b) wrapped phase.

The 2π jumps that are present in the wrapped phase signal that is shown in Figure 1 (b) must be removed in order to return the phase signal $x_w(n)$ to a continuous form and hence make the phase usable in any analysis or further processing. This process is called phase unwrapping and has the effect of returning a wrapped phase signal to a continuous phase signal that is free from 2π jumps. The basic phase unwrapping process can be explained by splitting the task down into the following steps.

1. Start with the second sample from the left in the wrapped phase signal $x_w(n)$.
2. Calculate the difference between the current sample and its directly adjacent left-hand neighbour.
3. If the difference between the two is larger than $+\pi$, then subtract 2π from this sample and also from all the samples to the right of it.

4. If the difference between the two is smaller than -π, then add 2π to this sample and also to all the samples to the right of it.
5. Have all the samples in $x_w(n)$ been processed? If **No** then go back to step 2. If **Yes** then stop.

The gradual progression of the phase unwrapping process is shown in Figure 2. The original wrapped phase signal $x_w(n)$ contains four wraps, as shown in Figure 2(a). The removal of the first wrap is shown in Figure 2(b). The removal of the second wrap is shown in Figure 2(c). The removal of the third wrap is shown in Figure 2(d). Once the fourth and final wrap is removed the signal has now been completely unwrapped, as shown in Figure 2(e).

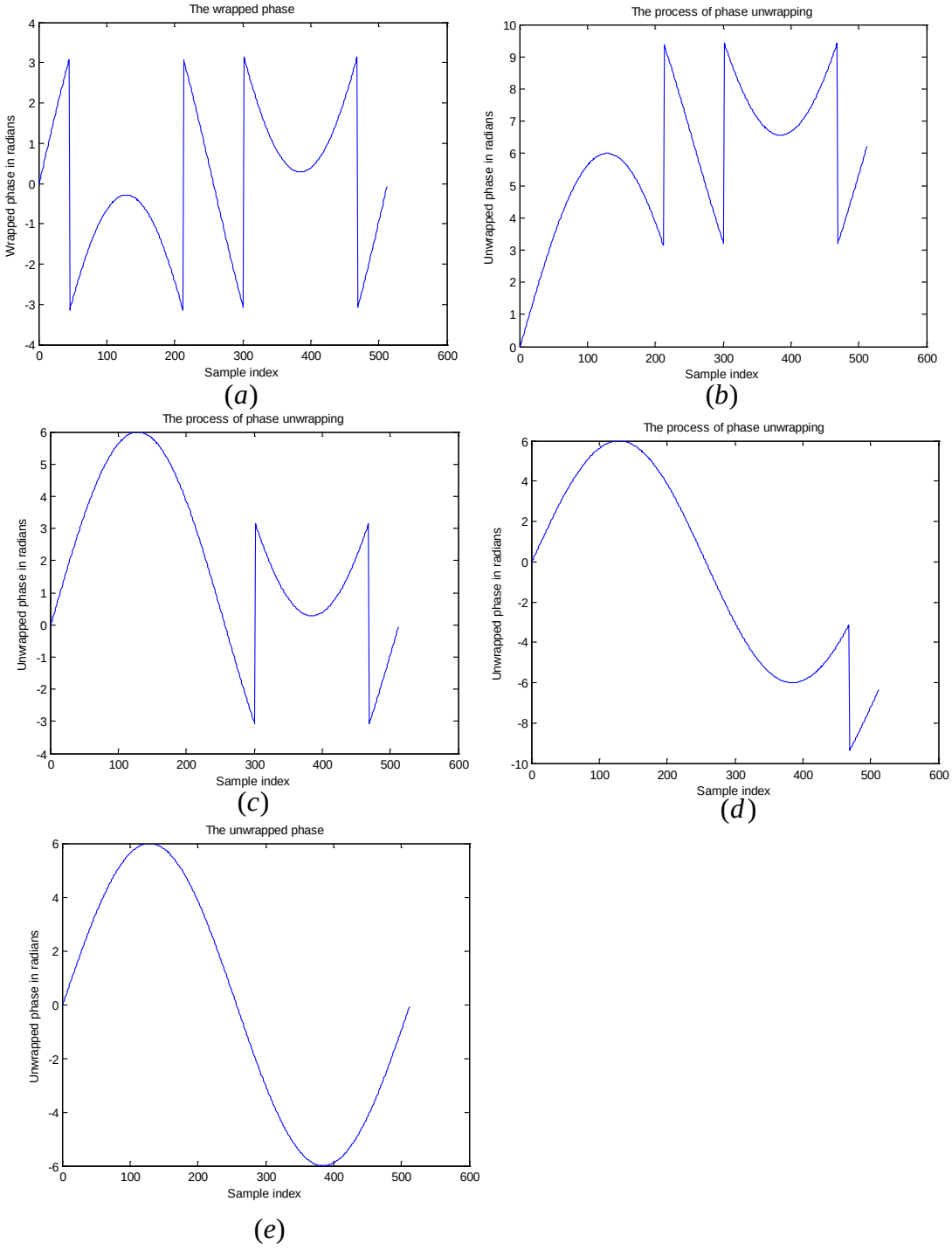


Figure 2: (a) Wrapped phase, (b)-(e) the process of phase unwrapping.

The Matlab code to unwrap $x_w(n)$ is shown below.

```
%Unwrap the signal xw(n)
xu = xw;
for i=2:length(xw)
    difference = xw(i)-xw(i-1);
    if difference > pi
        xu(i:end) = xu(i:end) - 2*pi;
    elseif difference < -pi
        xu(i:end) = xu(i:end) + 2*pi;
    end
end
figure, plot(xu)
xlabel('Sample index')
ylabel('Unwrapped phase in radians')
axis([0 512 -6.5 6.5])
title('The unwrapped phase')
```

We can express the unwrapping process mathematically as,

$$x_U(n) = \mathcal{U}[x_w(n)] = x_w(n) + 2\pi k$$

Where $\mathcal{U}[\,]$is the phase unwrapping operation, $x_U(n)$ is the unwrapped phase signal and k is an integer.

The wrapped phase signal that is shown in Figure 2(a) is a very simple signal to unwrap. This is because $x_w(n)$ is a simulated signal that does not contain any noise, and also because the sampling rate is sufficiently high here, as we have 512 samples per period.

Phase unwrapping for practical real-world applications is actually one of the most challenging tasks in digital signal processing. This is because of two main reasons, which are;

1. The wrapped signal may be **noisy**
2. The wrapped signal may be **under sampled**.

We will discuss these two cases in detail below;

As we have explained previously, in order to perform the phase unwrapping process, the difference between a sample and the preceding sample (directly adjacent on its left) is calculated. When this difference is larger than +π, or smaller than -π, a phase wrap is detected. Once a phase wrap is detected, the value of 2π is either added, or subtracted, to/from this sample and also from all the further samples to the right-hand side of it. Now consider that the phase wrap that has just been detected can be either a 'genuine phase wrap', or might also be a 'fake wrap' that has been produced by noise in the signal. It is this distinction between true phase wraps and apparent phase wraps that have been caused by noise that make the practical phase unwrapping problem such a challenging task. The four wraps that were shown in Figure 1(b) are all genuine phase wraps. Now consider the wrapped phase signal that is shown in Figure 4(a). This signal contains four genuine phase wraps and one fake wrap.

In the case where a genuine wrap has been detected, the phase unwrapping process continues its operation successfully. The existence of a fake wrap will affect the unwrapping of the sample where

the fake wrap is located and will also affect all the samples to the right of the wrap. Errors accumulate, as shown in Figure 4(c). This makes phase unwrapping such a difficult task, as an error in the determination of only a single phase wrap may affect a significant part of the entire signal because this error then propagates to the rest of the samples to the right-hand side of the problematic 'detected' wrap.

2. Effect of noise on one-dimensional phase unwrapping

Noise can have catastrophic effects on the phase unwrapping process. Let us use a computer simulation to illustrate this. Suppose that we have the discrete signal $x(n)$ and then we add white noise to this signal as follows;

$$x_{noise}(n) = x(n) + white\ noise$$

Then we shall 'wrap' the noisy signal;

$$x_{w_{noise}}(n) = \mathcal{W}[x_{noise}(n)]$$

After that, we shall attempt to unwrap the noisy wrapped phase signal $x_{w_{noise}}(n)$;

$$x_{U_{noise}}(n) = \mathcal{U}[x_{w_{noise}}(n)]$$

The Matlab code to perform the procedure that was outlined above is given below. Here we have set the variance of the noise to a value of 0.2.

```
%This program is to simulate the wrapping process
clc; close all; clear
%The number of samples in the signal
N=512;
n=0:N-1;
fo=1/512;
% The signal x whose amplitude exceeds the range [-pi,pi]
noise_variance = 0.2;
x = 6*sin(2*pi*fo*n) + noise_variance*randn(size(n));
plot(x)
xlabel('Sample index')
ylabel('Original phase in radians')
title('The signal x whose amplitude exceeds the range [-\pi,\pi]')
%Calculating the wrapped signal using the four quadrant arctangent function
xw = atan2(sin(x), cos(x));
figure, plot(xw)
xlabel('Sample index')
ylabel('Wrapped phase in radians')
title('The wrapped phase')
%Unwrap the signal xw(n)
xu = xw;
for i=2:length(xw)
    difference = xw(i)-xw(i-1);
    if difference > pi
        xu(i:end) = xu(i:end) - 2*pi;
    elseif difference < -pi
        xu(i:end) = xu(i:end) + 2*pi;
    end
end
```

```
figure, plot(xu)
xlabel('Sample index')
ylabel('Unwrapped phase in radians')
title('The unwrapped phase')
```

The result of running this Matlab program is shown below. In this case the noise does not adversely affect the phase unwrapping process.

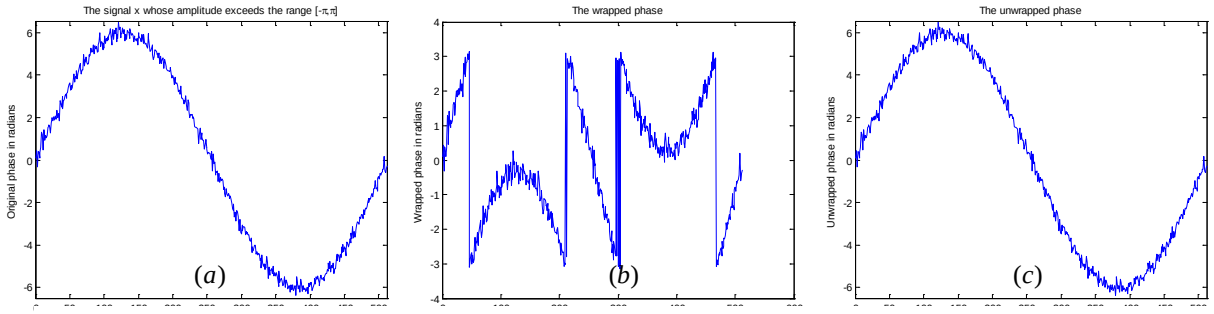


Figure 3: (a) Original noisy signal, (b) the phase wrapped signal, (c) the phase unwrapped signal. Here the noise variance has been set to a value of 0.2.

The noise variance is now set to a higher value of 0.8. In this case, you can see that the higher noise level has seriously affected the phase unwrapping process. The phase unwrapping of the signal that is shown in Figure 4(b) is a challenging task. This is due to the existence of a fake wrap in the signal.

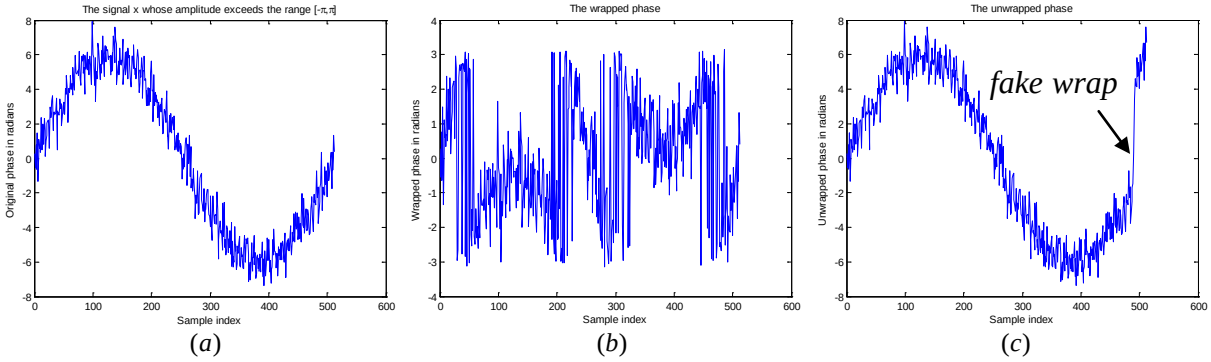


Figure 4: (a) Original noisy signal, (b) the phase wrapped signal, (c) the phase unwrapped signal. Here the noise variance has been increased to a value of 0.8.

The wrapped signal in Figure 4(a) contains four genuine phase wraps and one fake wrap. The existence of a fake wrap will affect not only the unwrapping of the sample where the fake wrap occurs, but all the subsequent samples that are located to the right-hand side of the wrap. Errors then accumulate, as shown in Figure 4(c), where all the unwrapped phase values to the right-hand side of the location of the fake wrap are incorrect. Let us re-iterate that it is this fact that makes phase unwrapping such a difficult task, because an error in determining only a single phase wrap may affect the majority of the signal due to propagation of errors.

3. Effect of under sampling on one-dimensional phase unwrapping

A phase unwrapper detects the existence of a wrap in a discrete signal by calculating the difference between two successive samples. If this difference is larger than $+\pi$, then the phase unwrapper considers that a wrap exists at this location. This could either be a genuine phase wrap, or it might be a fake wrap due to noise or under sampling. The effect of under sampling (phase aliasing) on one-dimensional phase unwrapping can be explained as follows.
Suppose that we have the discrete phase signal;

$$x(t) = 10sin10t, \qquad 0 \leq t \leq 1$$

The rate of phase change of the signal is given by differentiating the above equation to give;

$$\frac{dx(t)}{dt} = 100cos10t$$

The maximum value for the phase change here is 100. Remember that the difference between two successive samples should be less than π, or else an unwrapping algorithm will incorrectly think that there is a phase wrap at this location, when there isn't actually a wrap present. Therefore we can work out how many samples we must have in one cycle for this relation to hold true;

$$\frac{100}{N} < \pi$$

Where *N,* the sampling rate, is the number of samples in a cycle. Rearranging, this implies that sampling rate, i.e. the number of samples in a cycle;

$$N > \frac{100}{\pi} = 31.83$$

This means that for this example dataset, we must use at least 32 samples per cycle in order to avoid under-sampling. If we use less than this number of samples, then the difference between two successive samples of this signal may reach a value of pi (or higher) and hence be incorrectly regarded as being a true phase wrap, when there is actually no real phase wrap present, i.e. it would be a 'fake wrap'. Note that, for discrete signals of finite length, if you know the sampling rate and the number of cycles, then you can work out the total number of samples over the entire signal length. This will be covered in further detail in the 2D unwrapping tutorial.

Let us test this result using Matlab. The following Matlab program implements this concept.

```
clc; close all; clear
%N should be more than 32 to prevent under sampling effects on the phase
%unwrapping process
N=31;
t=0:1/N:1;

% The signal x whose amplitude exceeds the range [-pi,pi]
x = 10*sin(10*t);
plot(t,x,'*-')
xlabel('time')
ylabel('Original phase in radians')
title('The signal x whose amplitude exceeds the range [-\pi,\pi]')
```

```
%Calculating the wrapped signal using the four quadrant arctangent function
xw = atan2(sin(x), cos(x));
figure, plot(t,xw,'*-')
xlabel('time')
ylabel('Wrapped phase in radians')
title('The wrapped phase')

%Unwrap the signal xw(t)
xu = xw;
for i=2:length(xw)
    difference = xw(i)-xw(i-1);
    if difference > pi
        xu(i:end) = xu(i:end) - 2*pi;
    elseif difference < -pi
        xu(i:end) = xu(i:end) + 2*pi;
    end
end
figure, plot(t,xu,'*-')
xlabel('time')
ylabel('Unwrapped phase in radians')
title('The unwrapped phase')
```

Figure 5(a) shows the discrete signal

$$x(t) = 10sin10t, \qquad 0 \leq t \leq 1$$

With a sampling rate of *N*=32 samples per period. Then the phase signal is wrapped as is shown in Figure 5(b). After that the signal is unwrapped as shown in Figure 5(c). As can be seen in Figure 5(c) the phase unwrapper succeeds in processing the signal correctly when sampled at this rate.

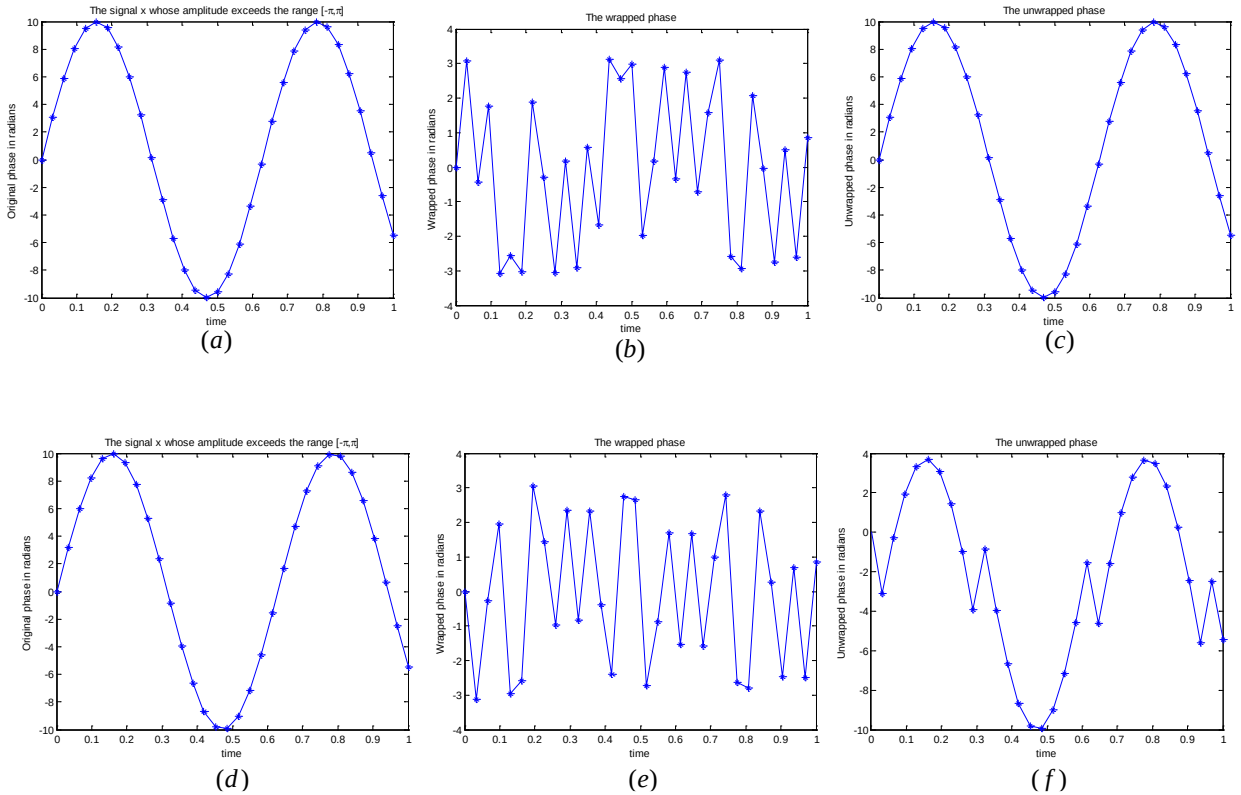


Figure 5: (a)-(c) Unwrapping a discrete signal that has an adequate sampling rate of N=32 sampes per period, (d)-(f) Unwrapping a slightly under sampled discrete signal where N=31.

The above experiment is repeated using a slightly lower sampling rate, with *N*=31, as is shown in Figures 5(d)-(f). As you can see the phase unwrapper fails to correctly retrieve the signal. This error is produced due to under sampling of the original signal.

As a result of this tutorial you should have an understanding of the one-dimensional phase unwrapping process and its basic mode of operation. You should also be familiar with the practical and theoretical limitations of phase unwrapping algorithms in terms of their performance in the presence of noise and due to variations in signal sampling rates.

Note: Matlab produces a different sequence of noise on each execution of the `randn` command – as it is by definition pseudo random. This is the reason that you will get very similar, but not exactly the same, results as those that are presented in this tutorial.

Exercises

1. There is a function in Matlab called unwrap. This function implements a one-dimensional phase unwrapper. The source code for this function is reproduced below. Study this code and compare it with the phase unwrapper that is implemented in this tutorial.

```
function up = unwrap(p)
%UNWRAP unwrap phase

  N = length(p);
  up = zeros(size(p));
  pm1 = p(1);
  up(1) = pm1;
  po = 0;
  thr = pi - eps;
  pi2 = 2*pi;
  for i=2:N
    cp = p(i) + po;
    dp = cp-pm1;
    pm1 = cp;
    if dp>thr
      while dp>thr
        po = po - pi2
        dp = dp - pi2;
      end
    end
    if dp<-thr
      while dp<-thr
        po = po + pi2
        dp = dp + pi2;
      end
    end
    cp = p(i) + po;
    pm1 = cp;
    up(i) = cp;
  end
```

2. Write a Matlab program to implement the one-dimensional phase unwrapping algorithm explained in Ref. 2.

3. A one-dimensional phase unwrapping algorithm can be implemented as shown in the Matlab program below. Explain the operation of this code.

```
%This program is to simulate the wrapping process
clc; close all; clear
%The number of samples in the signal
N=512;
n=0:N-1;
fo=1/512;
% The signal x whose amplitude exceeds the range [-pi,pi]
x = 6*sin(2*pi*fo*n);
plot(x)
xlabel('Sample index')
ylabel('Original phase in radians')
title('The signal x whose amplitude exceeds the range [-\pi,\pi]')
%Calculating the wrapped signal using the four quadrant arctangent function
xw = atan2(sin(x), cos(x));
figure, plot(xw)
xlabel('Sample index')
ylabel('Wrapped phase in radians')
title('The wrapped phase')

%Unwrap the signal xw(n)
tic   %start measuring the execution time
K=0;
increments = zeros(size(xw));
for i=2:length(xw)
    difference = xw(i)-xw(i-1);
    if difference > pi
        K = K - 2*pi;
    elseif difference < -pi
        K = K + 2*pi;
    end
    increments(i) = K;
end
xu = xw + increments;
toc      %finish measuring the execution time
figure, plot(xu)
xlabel('Sample index')
ylabel('Unwrapped phase in radians')
title('The unwrapped phase')
```

4. The Matlab implementation of the one-dimensional phase unwrapping algorithm that is presented in exercise 3 is faster than the other algorithms that were explained in this tutorial and in exercises 1 & 2. Confirm this claim by measuring the execution time for these four different phase unwrapping algorithms at different sampling rates. Enter your results in the table below. Hint: use the `tic` and `toc` Matlab functions to measure the execution time of the code.

Number of Samples N=	Phase unwrapper implemented in section 1 of this tutorial	Phase unwrapping function implemented in Matlab `unwrap`. (see exercise 1)	Phase unwrapping implemented using the Itoh algorithm. (see exercise 2)	Phase unwrapping implemented in exercise 3
512				
1024				
5000				

References

1. Dennis Ghiglia and Mark Pritt, Two-dimensional phase unwrapping theory, algorithms and applications, John Wiley & Sons, 1998.
2. K. Itoh, "Analysis of the phase unwrapping problem," Applied Optics, Vol. 21, No. 14, p. 2470, July 15, 1982.