# Phase unwrapping* overview

(R. Mellors)

A) The problem and some answers**

1. Global solutions
    a) Transforms
    b) Matrix
2. Local solutions
    a) Residues
    b) Quality
3. Network methods

B) Reduce or avoid the problem***

A. Filter
B. Model phase beforehand

*phase unwrapping problems occurs in other areas as well
**many algorithms exist; I present only selected ones.
*** this is the best strategy in many cases.

# Phase unwrapping

Assume that we have two signals taken at different times:

$g_1 = a_1 \exp(i4\pi R_1/\lambda)$
$g_2 = a_2 \exp(i4\pi R_2/\lambda)$

a1,a2 = complex reflectivity
R1, R2 is range from antenna to surface
$\lambda$ = wavelength

At a given point, assume $a_1 = a_2 = a$

$(g1)(g2^*) = (a^2)\exp[i4\pi(R_1-R_2)] = s(t)$

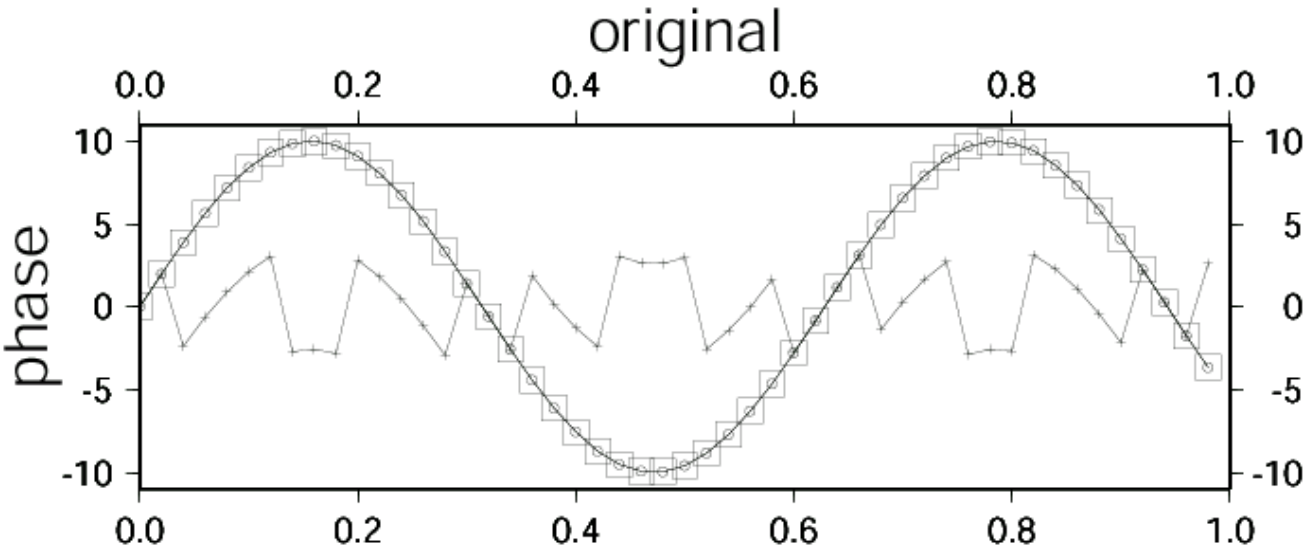
The phase of this function is proportional to the effective difference in range, which in turn depends on satellite geometry, topography, soil moisture, or maybe even *deformation*.

# Extracting the phase

We can get only the wrapped phase*
$\psi(t) = \arctan(I(s(t)),R(s(t)))$

where $-\pi < \psi(t) \leq \pi$

We would like the continuous phase.
This appears simple. Look for $2\pi$ jumps and then add the appropriate multiple of $2\pi$.




*If the data are good, phase unwrapping is straightforward*

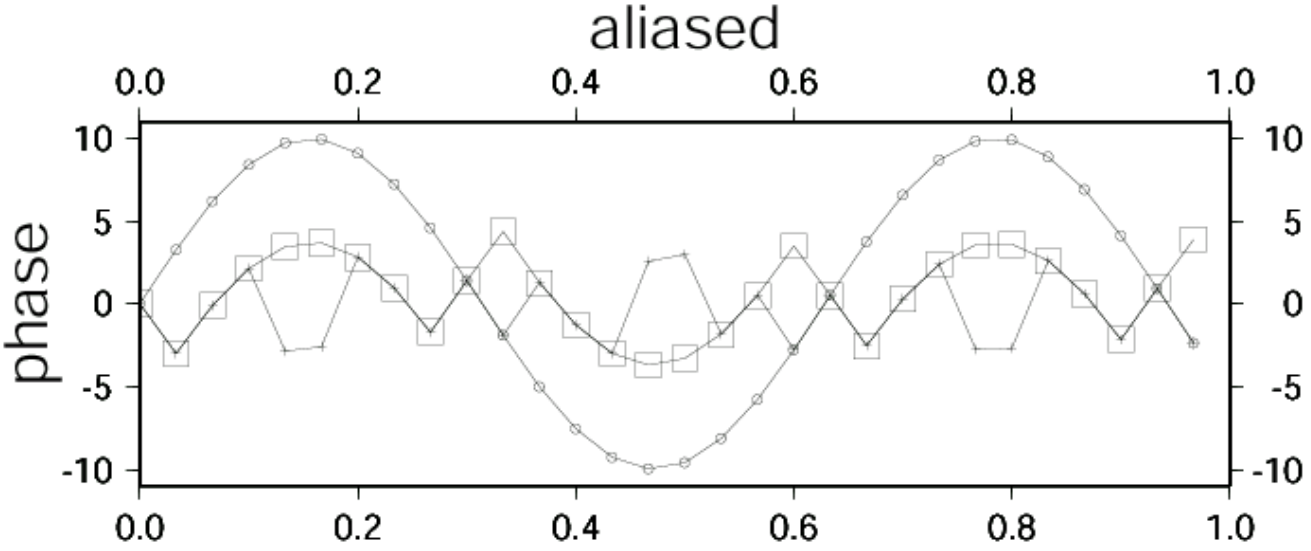
•We could take derivative in complex version (e.g. Sandwell & Price)

# Two major problems

*Aliasing*.

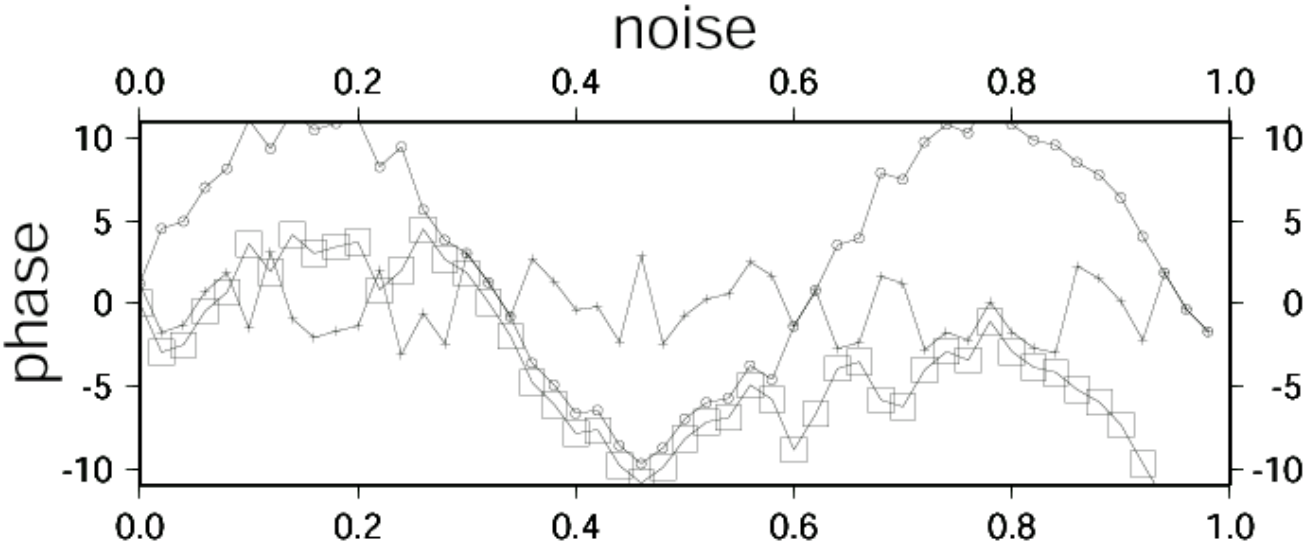
True phase changes by more than 1 cycle ($\pi$ radian) between samples.

Caused by long baselines, steep topography, or high deformation.




*Noise and/or gaps in the data.*

Changes in the surface may decorrelate the signal and introduce noise.



**For 2D data the same problems exist.**

How to unwrap?
Want to find a function that when wrapped, is “close” to the observed data (whatever “close” is).

Two basic approaches:

•Global methods that attempt to unwrap all pixels simultaneously.

•Local methods that solve along a path.

Note that the correlation data allows some estimation of how good the phase data is (maybe also amplitude/phase stability over time?)

# Global

*We want to find the function whose local derivatives "match" the observed derivatives given some measure :*

$$e^p = [(\phi_{i+1,k}-\phi_{i,k}) - \Delta^x_{i,k}]^p + [(\phi_{ik+1}- \phi_{ik})-\Delta^y_{ik})]^p$$

P = exponent

$\phi$ = unknown function

$\Delta$ = derivatives of the observed phase (can calculate from the complex phase).

For p = 2, this is equivalent to the discrete version of Poisson's Equation

Two basic ways to solve:

*Transform* : FFT, DCT (discrete cosine transform- be careful with b.c.'s)

*Matrix* (will allow weighting but now nonlinear and requires iteration)

Can vary the exponent (i.e. don't have to use 2)

•An elegant and easy solution, but….
doesn't work very well with noise.
•Tend to underestimate true phase when noise exists (it's a least-square fit).
•No easy way to add weighting short of iterating.

Matrix methods solve:

$$\mathbf{Ax} = \mathbf{b}$$

With weighting:

$$\mathbf{WAx} = \mathbf{Wb}$$

**W** = matrix of weights
**A** = operator
**B** = set of observed phase values
**x** = unknown function

These work better than the transform methods but like all global solutions, do not provide a good fit anywhere.

## Transform-based methods

Fast, but do not allow weighting
FFT requires periodic conditions
and extension of data.

Apply 2D Fourier transform:

$$\rho_{i,k} = (\Delta^x_{i,k} - \Delta^x_{i-1,k}) + (\Delta^y_{i,k} - \Delta^y_{i,k-1})$$

$$\Phi_{m,n} = \frac{P_{m,n}}{2\cos(\pi m / M) + 2\cos(\pi n / N) - 4}$$

$\Phi$ = Fourier transform of $\phi$
P = Fourier transform of $\rho$

1. Calculate the $\rho_{i,k}$ from the data
2. Calculate the 2D FFT of $\rho_{i,k}$
3. Calculate $\Phi_{m,n}$ from the transformed $\rho_{i,k}$
4. Do inverse FFT

## Local (path following)

Similar to the 1D approach

1.) Calculate the differences of the wrapped phase.

2.) Wrap the differences.

3.) Set the value of the first value.

3.) Integrate along all values.

Do this along a line throughout 2D area (in a zigzag back and forth along the rows, for example)
Works great if there is no noise.

With noise:

1) An error near the start of the path propagates along the whole path.

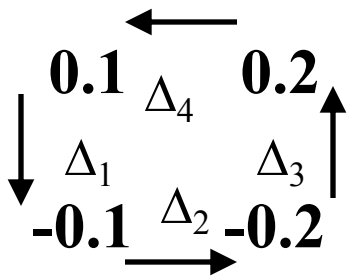
2) Answer may vary with path.

3) Need to identify bad pixels.

How?

**For 2D data integration around a closed loop should sum to zero.**

(in cycles – multiply by 2π to get radians)




$\Delta_1 = -0.2$
$\Delta_2 = -0.1$
$\Delta_3 = 0.4$
$\Delta_4 = -0.1$
Sum: ***0.0***

**-0.2** **-0.2** **-0.3**

If not:

$\Delta_1 = -0.4$
$\Delta_2 = -0.2$
$\Delta_3 = -0.3$
$\Delta_4 = -0.1$
Sum: ***-1.0***

**0.1** **0.2** **0.3**

*0.0* *-1.0*

**-0.1** **-0.2** **-0.4**

*0.0* *0.0*

**-0.2** **-0.2** **-0.3**

(i.e. non-conservative with a rotational component)

However, we know that topographic surfaces are conservative

•Any points that violate this should be avoided.

•These points are known as residues.

•Any integration path that circles a residue will contain errors – need to make “branch cuts”

•A residue is a property of phase differences, not a single pixel.

• can be positive or negative

**0.1** **0.2** **0.3**

*0.0* *-1.0*

**-0.1** **-0.2** **-0.4**

*0.0* *0.0*

**-0.2** **-0.2** **-0.3**

1.) Identify all residues in data (marked as the upper left pixel)
2) Draw lines (branch cuts) between them to eliminate possibility of drawing a circle around a residue.
3) Unwrap the rest of the data

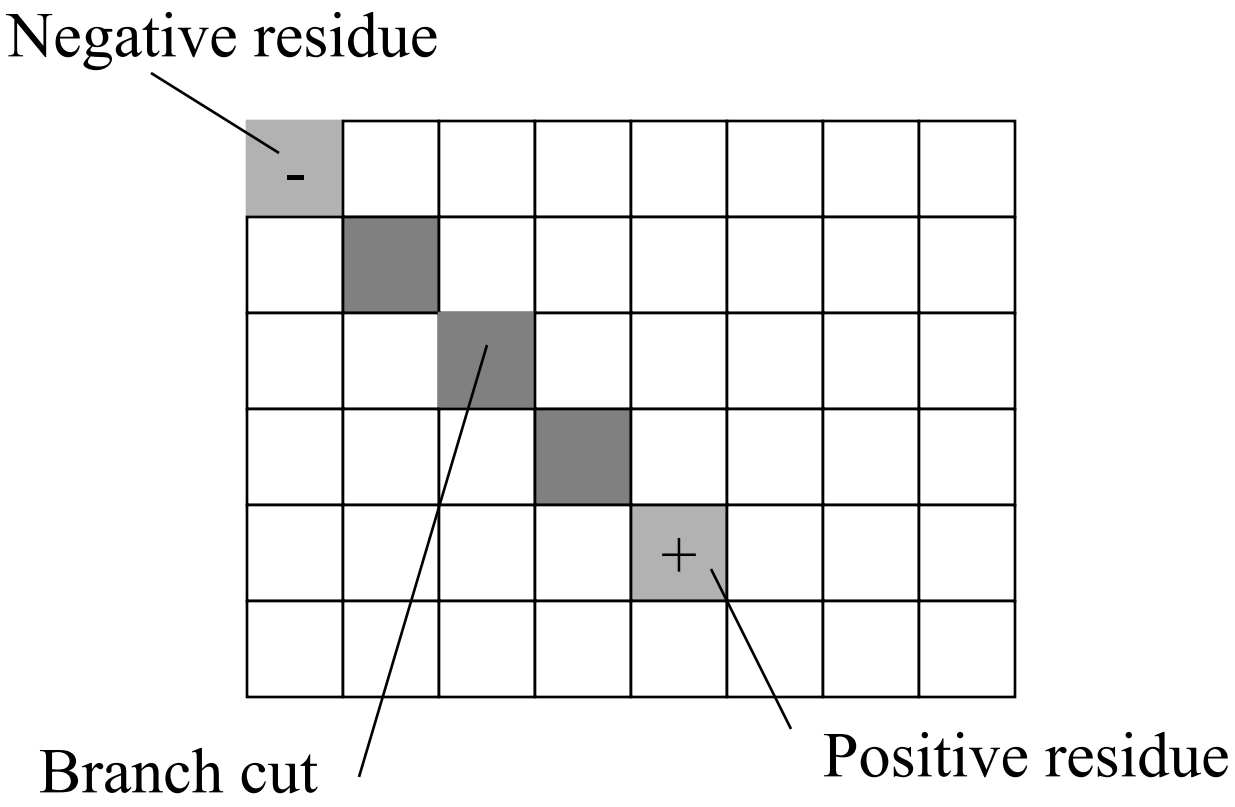
*Note: residues can be positive or negative. A positive linked to a negative cancel each other out.

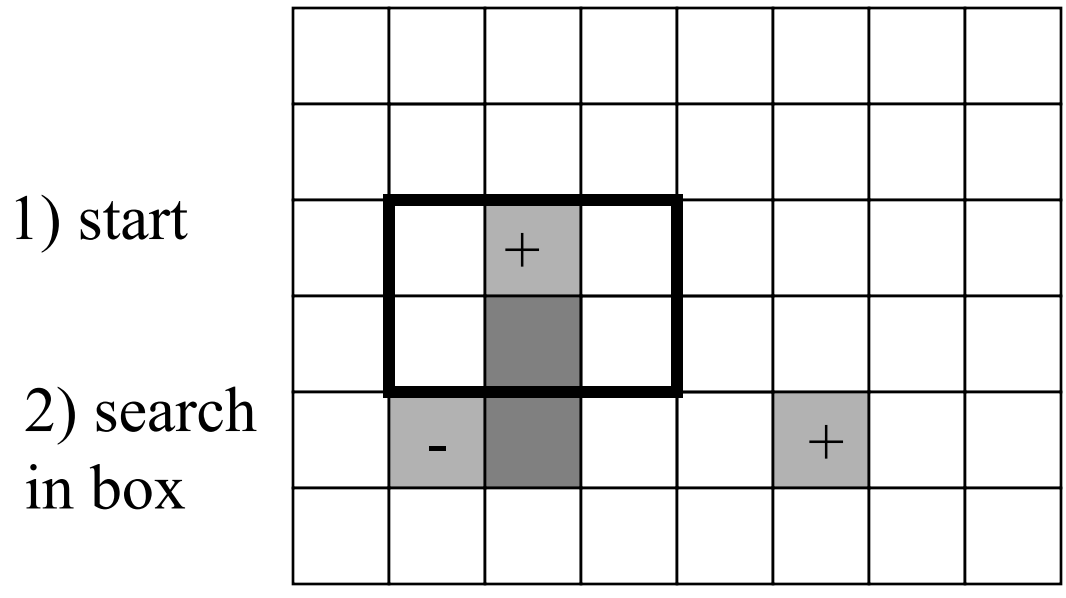
This is the basis of the Goldstein approach, (used in Roi_pac).

Often, poor data (with low correlation is masked out beforehand)

# Goldstein algorithm

1.) Calculate correlation for phase data.
2.) Mask out all areas with correlation less than a certain threshold value.
3.) Go through all pixels and identify residue locations (upper left of 4 pixels).
4) Start with first residue, look for nearest residue. Draw a “line” of marked pixels between the two.
- if residues cancel, go to next residue and start new “tree”
- otherwise, look for next nearest and draw line
- can also “cancel” by connecting to edge.
- connected lines are called a tree.
5) path-integrate along remaining pixels.




3) find second residue

4) connect with branch cut

•Fast
•Need to start integration (seed point) in area of good data.
•Residues often lie in areas of layover.
•Regions can get isolated from each other with dramatically different phase (by multiples of $2\pi$).
•Some implementations allows manual connecting of regions.
•Some implementations allow “pre-processing” to connect closely-spaced opposite residues (dipoles) first.

•Minimizes distance between residues; does not minimize number of cycles needed to “unwrap”.

•A similar algorithm uses quality rather than residues to define.

# Flynn's minimum discontinuity

1) Identify lines of discontinuity (fringe lines)
    1) Difference between adjacent pixels > $\pi$
    2) Magnitude of discontinuity defined by number of multiples of $2\pi$ needed to fix.
2) Add multiples of $2\pi$ to eliminate lines of discontinuities that form loops.
3) Checks to see if operation creates more discontinuities than removes.
4) Continues in an iterative fashion.
5) At end, no more discontinuities can be removed without adding more.
6) Complicated algorithm (i.e. I looked at it and got a headache)

**Comments:**

- Slow. Masking helps.
- Can be appended after other unwrapping algorithms.
- Good to run Goldstein or quality first, then Flynn.

## Chen and Zebker's (2000) [SNAPHU]

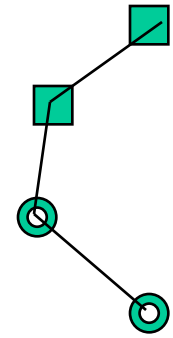
1) A branch-cut algorithm minimizes the total discontinuity length ($L^0$).
2) Flynn yields a $L^1$ solution.
3) Least-squares yield an $L^2$ solution.

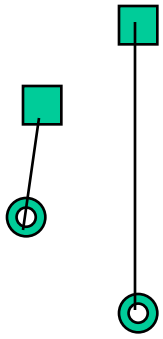
C&Z claim low norms are best.

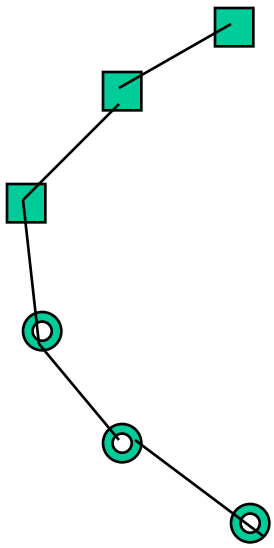
For interferograms corrupted by noise, $L^0$ and $L^1$ algorithms yield similar solutions.

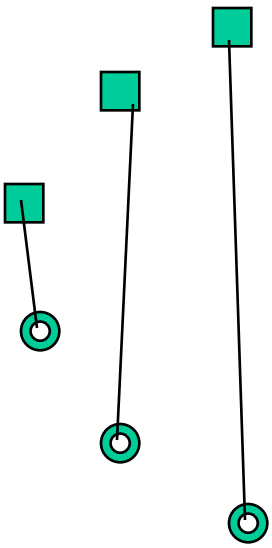
For layover, where discontinuities separate severe phase gradients, $L^1$ algorithms do not do well.

# Residues caused by topographic layover with illumination from one side









**Good – L0**

**Bad – L1**

Negative residue

Positive residue

The ultimate L0 algorithm would minimize the total cut length.

C&Z show that the L0 problem is equivalent to an NP-complete problem and therefore hard for efficient algorithms to solve completely.

- Therefore, the L0 branch-cut algorithm is a good place to start.
- Major problem is that cuts close on themselves.
- Total tree length is upper bound on total discontinuity length.

## Modifications

1) Minimum span (MST)
    1) Define cuts so that a tree cannot connect to itself.
    2) Connect all trees.
    3) Use knowledge of residues to guide integration.
    4) Complete unwrapping
2) Minimum cost (MCF)
    1) Uses flow to reduce cycles

# Removing topography (and deformation)

If an accurate topographic model is available, then many of these problems can be alleviated during calculation of the interferogram.

- Reduce need for unwrapping.
- Deformation model can also be included.
- Can be done iteratively.
- Also true for large deformations (maybe done in an iterative fashion)

# The future:
# persistent scatterers
# 3D unwrapping

•The PS technique leads to widely spaced pixels. Phase relationships between these pixels may be challenging to define.

•If we have a time series of interferograms, phase unwrapping becomes a 3d problem.

•I think the basic strategies for 3D unwrapping are similar to 2D unwrapping but I have never tried it.