Modelling a phase stepping interferometer (PSI)

You will already be familiar with interferometry as a way of measuring the shape of test objects, perhaps lenses or mirrors, as they are being manufactured. In general the interferometer produces a fringe pattern, but how do we make use of that fringe pattern to get quantitative data out on the shape of the test object? You may already have thought about some of the problems with simply looking at the fringes and it basically comes down to how do you know which way a slope is going as both positive and negative slopes will give the same fringe pattern? The answer to this simple problem is that you move either the test object or the reference mirror and you look at the direction in which the fringes move. A fringe denotes a constant path difference and so as you move the test or reference you can track the path difference. Visually that works, but it still doesn't get you a simple quantitative answer.

Phase stepping interferometry

What the concept of changing the interferometer path length does highlight is that you might be able to get a quantitative answer buy moving, or phase stepping, the reference or test object. To understand how this might work, start by considering the interference between a reference wave $E_R = A\exp(i\phi_R)$ and a test wave $E_T = B\exp(i\phi_T)$ to give an intensity $I(\phi_R, \phi_T)$:

$$\begin{aligned} I(\phi_R, \phi_T) = |E_R + E_T|^2 &= (A\exp(i\phi_R) + B\exp(i\phi_T))(A\exp(-i\phi_R) + B\exp(-i\phi_T)) \\ &= A^2 + B^2 + AB\exp\big(i(\phi_T - \phi_R)\big) + AB\exp\big(-i(\phi_T - \phi_R)\big) \\ &= A^2 + B^2 + 2AB\cos(\phi_T - \phi_R) \end{aligned}$$

Mathematically we need to work out the phase difference, which appears in the sinusoidal interference term. With 3 unknowns (A, B and $(\phi_T - \phi_R)$), we need a minimum of 3 equations to do this, which we can get by adding 3 different phases to the reference and measuring the resulting intensity for each. However, to make things slightly simpler for now let's take 4 samples and step the phase of the reference between each interferogram by $\pi/2$:

$I_1 = I(\phi_R, \phi_T) = A^2 + B^2 + 2AB\cos(\phi_T - \phi_R))$
$I_2 = I(\phi_R + \pi/2, \phi_T) = A^2 + B^2 + 2AB\cos(\phi_T - \phi_R - \pi/2)) = A^2 + B^2 + 2AB\sin(\phi_T - \phi_R))$
$I_3 = I(\phi_R + \pi, \phi_T) = A^2 + B^2 - 2AB\cos(\phi_T - \phi_R))$
$I_4 = I(\phi_R + 3\pi/2, \phi_T) = A^2 + B^2 - 2AB\sin(\phi_T - \phi_R))$

Then subtracting the intensities in pairs we get:

$$\begin{aligned} I_1 - I_3 &= 4AB\cos(\phi_T - \phi_R) \\ I_2 - I_4 &= 4AB\sin(\phi_T - \phi_R) \end{aligned}$$

The phase difference can then be extracted simply by calculating:

$$\phi_T - \phi_R = \tan^{-1}\left(\frac{I_2 - I_4}{I_1 - I_3}\right)$$

This is the basis of phase stepping interferometry, and many different algorithms exist with various numbers of steps that each have their own advantages and disadvantages. You might like to look up some others on the internet, or in the classic textbook "Optical Shop Testing" by Daniel Malacara.

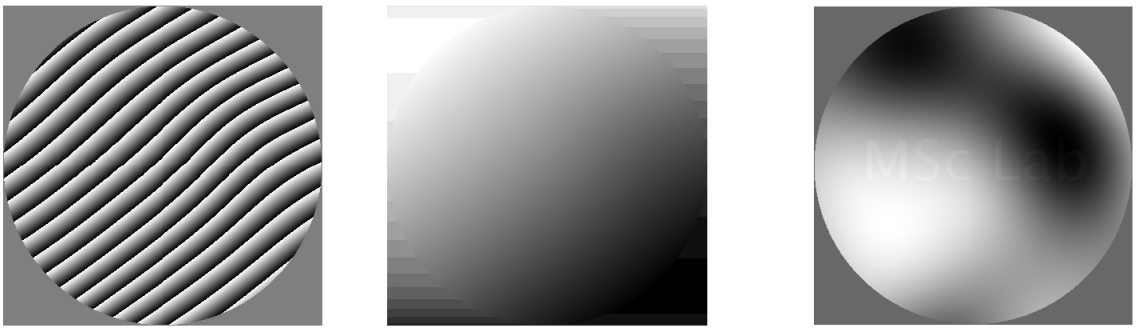
The aim of this experiment is to build a computer model of a phase stepping interferometer that will allow you to demonstrate and assess its performance. As part of this you will need

to deal with unwrapping the phase that comes out of the arctangent, and fitting your measured aberration to Zernike modes. You should also then look at what happens if the phase steps are incorrect. How robust are the algorithms in the presence of noise? What if the camera has a non-linear response to intensity?

Preliminaries

If you have already completed the "Introduction to modelling an interferometer" demonstration experiment then you should already be on the way to building the software tools that you will require. You should use the script for that experiment and your work on it to help you get started on this one. In addition, Appendix 1 gives a Matlab code listing that goes quite a long way to calculating the sort of things you will need to look at here. The code will run standalone as a script in Matlab or line by line in interactive mode, but perhaps you can do better buy writing a set of functions or even some object oriented code! Also the code is fairly lightly commented, so you will have to look up what it is doing, and in particular do try to understand the way arrays are being indexed. Add your own comments as you look through this code to help build an understanding of how it and Matlab works. Initially try to follow the steps below.

1. Start by generating a test aberration that you can feed into your interferometer. Examples given include a random but slowly varying function or a specific known shape like some text.
2. Calculate a set of interferograms for that test aberration where you change the phase of the reference between each by the required amount.
3. Perform the phase extraction calculation for your set of interferograms to get the phase back. At this stage you should end up with something like that shown in figure 1(a).



(a) Raw phase (b) unwrapped phase (c) tilt removed

Figure 1: Phase extracted from PSI algorithm

If you have already displayed your input test phase pattern then you will notice that this looks nothing like your input test aberration. Firstly, the extracted phase is wrapped to the range $-\pi$ to $+\pi$, because of the multivalued nature of the arctangent function used [note you do need to use something like the `atan2(y,x)` function to get the full 2π range]. As a result the phase is what is called wrapped and has steps in it. This is a common issue when dealing with phases in all sorts of areas and there are algorithms to *unwrap* the phase. These work by stepping through the values in a list and adding or subtracting 2π whenever a step greater than a certain threshold value (usually π) is seen. Matlab implements this sort of function but it operates only on 1d arrays or along only 1 dimension of 2d arrays. To fully unwrap a 2d array you need to apply the function once along each dimension. However if you try this then you will still get steps, because unwrapping functions do not work well in the presence of real phase steps such as those at the edge of your circular aperture. The Matlab code in the appendix demonstrates a simple fix by unwrapping the phase from the centre line of the image outwards. You still get problems at the edge, but these are now outside the region of interest (figure 1(b)).

So back to your code:

4. Unwrap the phase to give a continuous phase across your circular aperture.
5. Remember that the phase is actually the difference between your test phase and the reference, so if you have added a tilt to the reference then you need to remove that tilt from the unwrapped phase in order to see your test wavefront (figure 1(c)).
6. Analyse the test wavefront. Perhaps look at what Zernike modes it has present (remember the orthogonality relationship you demonstrated in the demonstration experiment), or simply measuring the RMS phase in the wavefront. In a real interferometer you might remove certain aberrations (piston, tilt and defocus) to see what is left afterwards – think about why you might want to do this.

Analysing performance of a non-ideal phase shifting interferometer

Your model so far has essentially described an ideal PSI. You could imagine a number of things going wrong in a real system such as:

- The phase shifts in the reference are not correct or the test object moves.
- The intensity of the laser changes between images that you collect for the different interferograms.
- The camera has a non-linear response to intensity (look up gamma factor).
- The camera produces noise on the captured images e.g. additive read-out noise and/or fundamental shot noise.
- There are thermal air currents in the system that introduce an extra variable phase.

You now have the tools to examine all of these phenomena and the effect that they can have on your final extracted phase. Start by looking at the sort of effects that each produces and try to assess their significance.

Other PSI algorithms

It was mentioned above that there are many PSI algorithms in addition to the 4-step one show above and the 3-step one in the appendix. Some of these other algorithms are specifically designed to overcome some of the issues highlighted above. Can you demonstrate that this is the case? Are there any drawbacks to using these more complex algorithms? Again Malacara's book, particularly chapter 14 of the 3rd edition, will be helpful here. You can find it and the 2nd edition in the central library or view or download a time restricted pdf copy of the 3rd edition via the central library website.

Your report

The power of a computer model is that you can produce results for a range of test scenarios very quickly. Do not be tempted to simply churn out images of extracted phases and fill your report with those. By all means fill your lab book with results from your model, but your report should be a more critical/analytical document that shows something useful. Your report should include:

- A description of your model including a demonstration of extraction of Zernike mode content and analysis of any inaccuracies in the results that your model produces.
- A demonstration and analysis of the inaccuracies that result from the sort of physical problems that can arise in a real interferometer (like those listed above).
- A demonstration of one or more, more sophisticated algorithms that can mitigate the sort of effects you have just shown above.

Appendix 1: Example code in Matlab

```
% set up arrays and plot some simple figures
[x y] = meshgrid(-1:0.005:1,-1:0.005:1);
r=sqrt(x.^2+y.^2);
theta=atan2(y,x);
circ = (r<1);
coma=2*sqrt(2)*(3*r.^3-2*r).*cos(theta).*circ;
figure(1);
imshow(coma,[]);
figure(3);
intens=abs(exp(1i*(30*x+40*y))-circ.*exp(1i*coma)).^2;
imshow(intens,[]);

% create a random image with low spatial frequency content
ri = rand( [50 50])-0.5;
kern = imresize(exp(-r.^2/(2*0.25^2)),[30 30]);
ri = conv2(ri,kern);
ri = imresize(ri,20,'bicubic');
ri = ri(300:700,300:700).*circ;
figure(20);
imshow(ri,[]);
% what is RMS aberration and how much coma is in it?
rms = sqrt(sum(ri(:).^2)/sum(circ(:)))
c = sum(ri(:).*coma(:))/sum(circ(:))

% create a test wavefront
blank = zeros(size(coma));
textInserter = vision.TextInserter('MSc Lab','Color', 0.1,'FontSize', ...
       72, 'Location', [150 60]);
textim = step(textInserter,blank);
testphase = coma + textim;
testwf = circ.*exp(1i*testphase);

% 3 step PSI algorithm
I1=abs(exp(1i*(30*x+40*y))+testwf).^2;
I2=abs(exp(1i*(30*x+40*y+2*pi/3))+testwf).^2;
I3=abs(exp(1i*(30*x+40*y+4*pi/3))+testwf).^2;
p=circ.*atan2(sqrt(3)/2*(I2-I3),(I1-(I2+I3)/2));
figure(5);
imshow(p,[]);

% try to unwrap the phase
figure(6);
imshow(unwrap(unwrap(p,[],2),[],1),[]);

% try to unwrap phase from middle out half of image at a time
q=p;
q(200:end,:)=unwrap(p(200:end,:),[],1);
q=flipud(q);
q(200:end,:)=unwrap(q(200:end,:),[],1);
q(:,200:end)=unwrap(q(:,200:end),[],2);
q=fliplr(q);
q(:,200:end)=unwrap(q(:,200:end),[],2);
q=flipud(fliplr(q));
figure(7);
imshow(q,[]);

% take off tilt to see what is left
figure(8);
q=circ.*(q+30*x+40*y);
imshow(q,[]);

% take off aberration to see what is left
figure(9);
c = sum(q(:).*coma(:))/sum(circ(:));
imshow(q-c*coma,[]);
```