PRIFYSGOL ABERYSTWYTH UNIVERSITY

Preliminary Results

Fitting of polynomials to optical surfaces

3rd year project

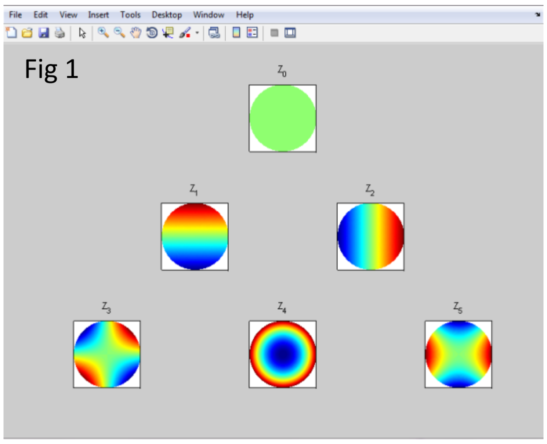
Martin Lee

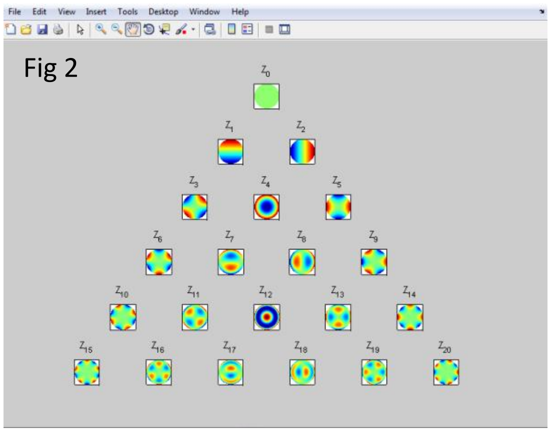
Aberystwyth University

mal36@aber.ac.uk

The first and perhaps most important task for acquisition of data during this project was to create an efficient Zernike Polynomial (ZP) generating function. This function needs to be able to generate any ZP required and ideally it needs to be able to calculate the ZP's quickly so as to not bottleneck the fitting algorithm.

Using Matlab a code has been produced to generate any ZP using the ANSI ordering scheme. Fig 1 and Fig 2 show the first 6 and 20 ANSI ZP's respectively. These have been checked against both diagrams from literature and hand calculations to ensure they are correct.




File Edit View Insert Tools Desktop Window Help
Fig 2
Z_0
Z_1 Z_2
Z_3 Z_4 Z_5
Z_6 Z_7 Z_8 Z_9
Z_{10} Z_{11} Z_{12} Z_{13} Z_{14}
Z_{15} Z_{16} Z_{17} Z_{18} Z_{19} Z_{20}

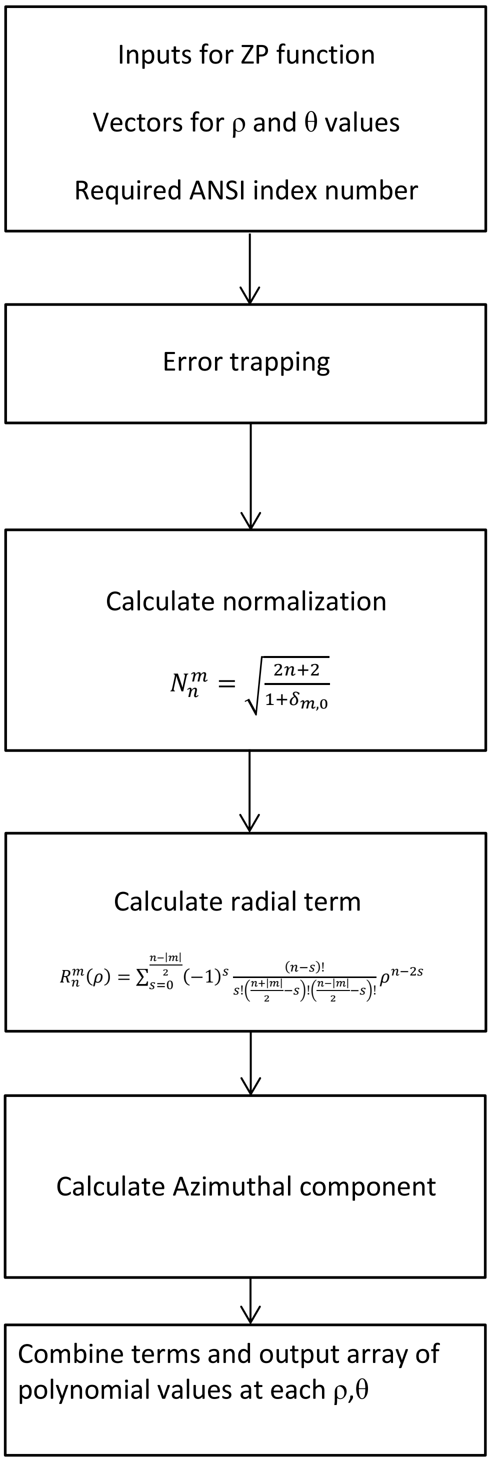
These plots represent a 200X200 array of values across a unit circle. The code takes an N*N array, reshapes this array into a single column vector. Matlab can then work on the whole vector simultaneously instead of having to repeat a nested loop over the N*N array.

This approach results in the plots shown in Fig 1 and 2 being produced almost instantaneously.

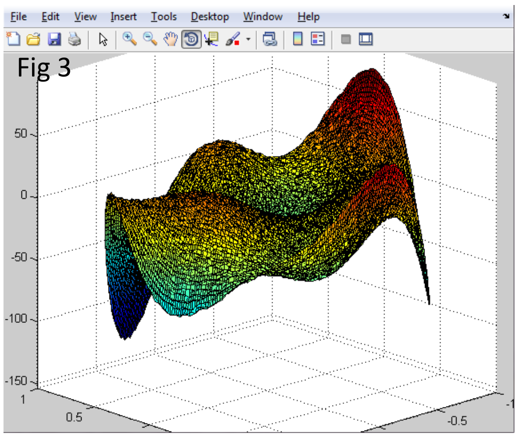
A flow diagram for the generating function is as follows.

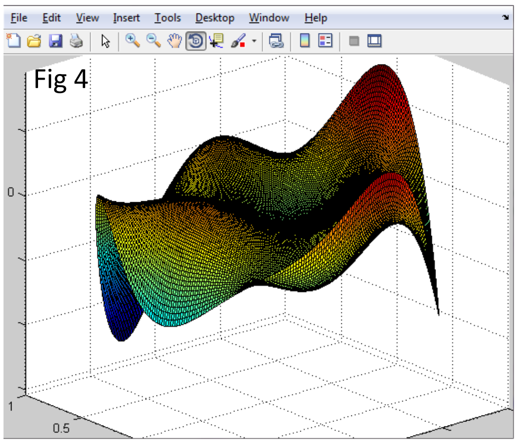
Mathematical description for the ZP's

$$Z_n^m(\rho,\varphi)=\begin{cases}N_n^m R_n^m(\rho)\cos(m\varphi)\\ N_n^m R_n^m(\rho)\sin(m\varphi)\end{cases}$$


Inputs for ZP function
Vectors for ρ and θ values
Required ANSI index number
Error trapping
Calculate normalization
$N_n^m=\sqrt{\frac{2n+2}{1+\delta_{m,0}}}$
Calculate radial term
$R_n^m(\rho)=\sum_{s=0}^{\frac{n-|m|}{2}}(-1)^s\frac{(n-s)!}{s!\left(\frac{n+|m|}{2}-s\right)!\left(\frac{n-|m|}{2}-s\right)!}\rho^{n-2s}$
Calculate Azimuthal component
Combine terms and output array of polynomial values at each ρ,θ

The main analysis of this project will be to look at fitting the ZP's too noisy surface data. In Fig 3 I show one such surface in 3D so that you can easily identify the addition of noise. The surface in Fig 4 is constructed from the first 20 ANSI ZP's(as shown in Fig 2) with coefficients selected from Matlabs build in uniform random number generator.





The plot in Fig 4 was produced using the coefficients outputted from the fitting algorithm. It is clear to see that the algorithm has successfully reproduced the surface and the full numerical results are shown in the appendix to this report.

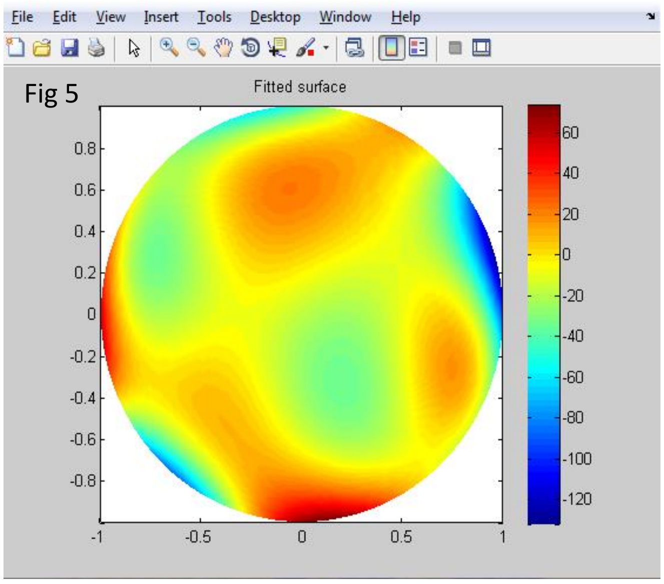
As a comparison between the real, the unweighted and the weighted coefficients we can look at the RMS of the surface given by.

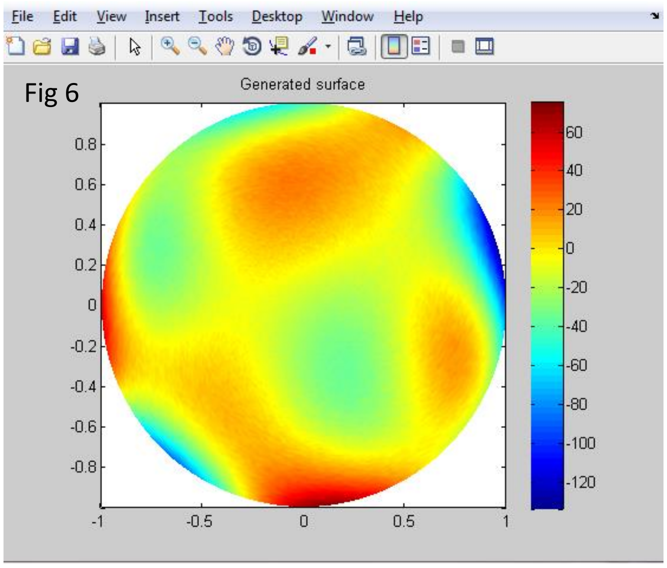
$$RMS = \sqrt{\sum_{j=0} a_j^2}$$

Surface	RMS
Real	22.813
Weighted	22.8129
Unweighted	22.8038

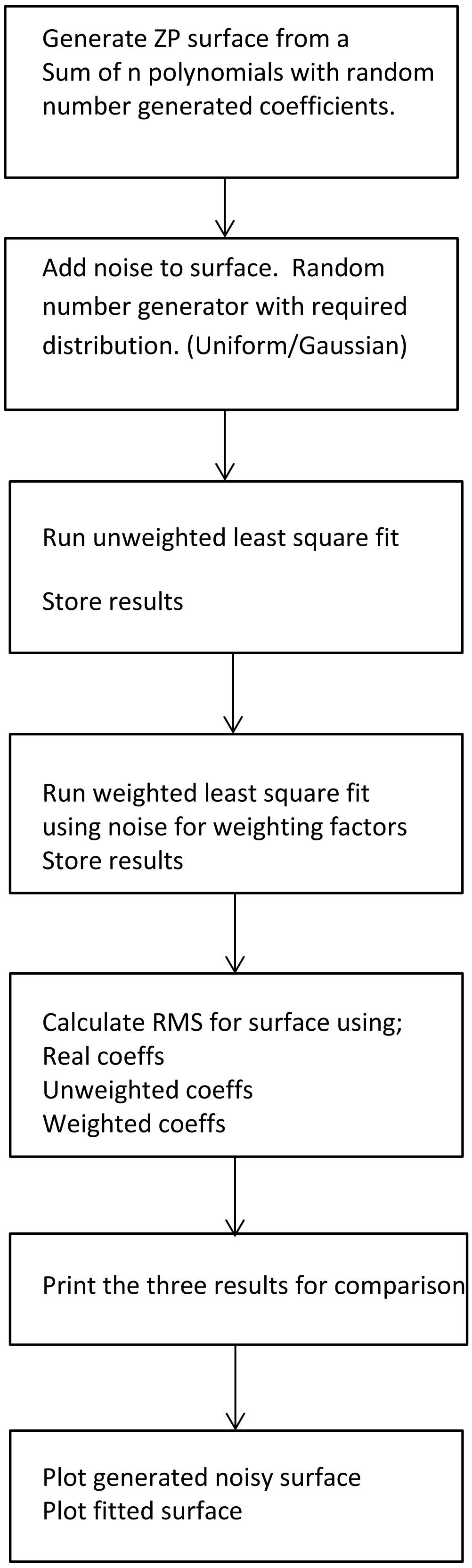
Both fitting procedures produce a good result in terms of RMS values but the weighted fit shows a clear improvement over the unweighted. This is what I would expect to happen when using a uniform distribution for the noise.

Fig 5 and 6 show the surface on a 2D gradient map.





Fitting comparison algorithm

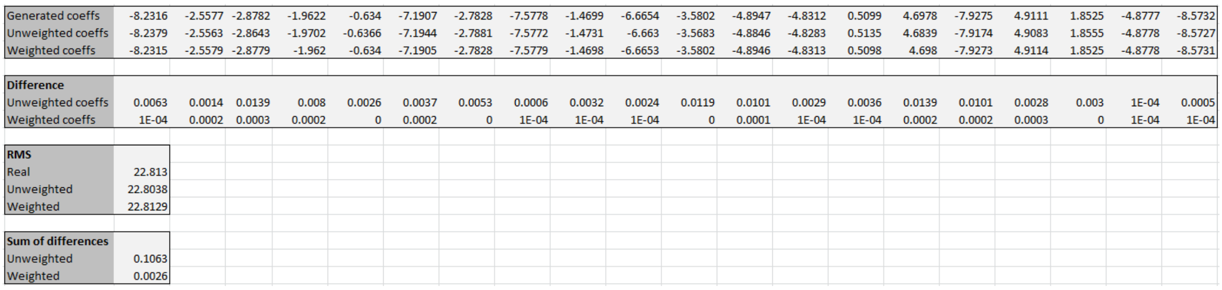

Generate ZP surface from a Sum of n polynomials with random number generated coefficients.
Add noise to surface. Random number generator with required distribution. (Uniform/Gaussian)
Run unweighted least square fit
Store results
Run weighted least square fit using noise for weighting factors
Store results
Calculate RMS for surface using;
Real coeffs
Unweighted coeffs
Weighted coeffs
Print the three results for comparison
Plot generated noisy surface
Plot fitted surface

As is clear from these initial results the fitting algorithm is working as expected. Giving the weighted least square fitting the exact noise array as was added to the surface, we see a significant improvement on the unweighted fit. The code is able to return many of the coefficients to their real value down to 10^{-4} decimal places.

Detailed analysis of how the algorithm performs with Gaussian error distributions can now be investigated.

Appendix

Raw data



Generated coeffs	-8.2316	-2.5577	-2.8782	-1.9622	-0.634	-7.1907	-2.7828	-7.5778	-1.4699	-6.6654	-3.5802	-4.8947	-4.8312	0.5099	4.6978	-7.9275	4.9111	1.8525	-4.8777	-8.5732
Unweighted coeffs	-8.2379	-2.5563	-2.8643	-1.9702	-0.6366	-7.1944	-2.7881	-7.5772	-1.4731	-6.663	-3.5683	-4.8846	-4.8283	0.5135	4.6839	-7.9174	4.9083	1.8555	-4.8778	-8.5727
Weighted coeffs	-8.2315	-2.5579	-2.8779	-1.962	-0.634	-7.1905	-2.7828	-7.5779	-1.4698	-6.6653	-3.5802	-4.8946	-4.8313	0.5098	4.698	-7.9273	4.9114	1.8525	-4.8778	-8.5731

Difference																				
Unweighted coeffs	0.0063	0.0014	0.0139	0.008	0.0026	0.0037	0.0053	0.0006	0.0032	0.0024	0.0119	0.0101	0.0029	0.0036	0.0139	0.0101	0.0028	0.003	1E-04	0.0005
Weighted coeffs	1E-04	0.0002	0.0003	0.0002	0	0.0002	0	1E-04	1E-04	1E-04	0	0.0001	1E-04	1E-04	0.0002	0.0002	0.0003	0	1E-04	1E-04

RMS	
Real	22.813
Unweighted	22.8038
Weighted	22.8129

Sum of differences	
Unweighted	0.1063
Weighted	0.0026