Coefficient-of-determination Fourier Transform CFT

Matthew Marko
matthew.marko@navy.mil

Abstract

This algorithm is designed to perform Discrete Fourier Transforms (DFT) to convert temporal data into spectral data. What is unique about this DFT algorithm is that it can produce spectral data at any user-defined resolution; existing DFT methods such as FFT are limited in resolution proportional to the temporal resolution. This algorithm obtains the Fourier Transforms by studying the Coefficient of Determination of a series of artificial sinusoidal functions with the temporal data, and normalizing the variance data into a high-resolution spectral representation of the time-domain data with a finite sampling rate.

Introduction

The Fourier Transform [1–6] is one of the most widely used mathematical operators in all of engineering and science [7–9]. The Fourier Transform can take a temporal function and convert it into a series of sinusoidal functions. While the original Fourier Transform is an analytical mathematical operator, Discrete Fourier Transform (DFT) methods are overwhelmingly used to take incoherent temporal measurements and convert them into understandable spectral plots. One of the limitations of the existing DFT methods, including the popular Fast Fourier Transform (FFT), is that the spectral resolution is proportional to the temporal resolution of the data to be transformed. If the temporal information of the resolution is limited, it will be impossible to accurately determine the frequencies of the results with great confidence.

The author proposes a numerical algorithm to perform a highly-resolved Fourier Transform of a temporal function of limited resolution. Rather than the spectral domain being proportional to the time step, the user defines exactly which frequencies are necessary to investigate. The spectral domain can be as large or as resolved as is necessary, regardless of the temporal resolution; the resolution possible is limited only by the abilities of the computer performing the transform.

Algorithm

The transform starts by first determining the peak total range of the data in the temporal domain, this range will become the base amplitude of the spectral series. The computer then generates a series of sine and cosine functions at each frequency within the spectral domain, and compares each of these sinusoidal functions to the temporal data to be transformed. In the comparison, a correlation coefficient is found and saved. To accommodate fluctuations in phase, each frequency generates both a sine and cosine function; this ultimately results in real and imaginary spectral components. Finally, the magnitudes of the correlation factor data is normalized, and the results is an accurate spectral representation of the temporal function.

The Fourier Transform is one of the most utilized mathematical transforms in science and engineering. By definition, a Fourier Transform will take a given function and represent it by a series of sinusoidal functions of varying frequencies and amplitudes. Analytically, the Fourier Transform is represented as [1, 2]

$$F(\omega) = \int_{-\infty}^{\infty} f(t) \cdot e^{-2\pi \cdot i \cdot t \cdot \omega} dt, \tag{1}$$

where i is the imaginary term ($i = \sqrt{-1}$), *f(t)* is any temporal function of t to be transformed, and ω (rad/s) represents the frequency of each sinusoidal function. The inverse of this function is

$$f(t) = \int_{-\infty}^{\infty} F(\omega) \cdot e^{2\pi \cdot i \cdot t \cdot \omega} d\omega. \tag{2}$$

Conceptually, the spectral function $F(\omega)$ represents the amplitudes of a series of sinusoidal functions of frequency ω (rad/s)

$$f(t) = \Sigma_{n=0}^{\infty} F(\omega_n) \cdot sin(\omega_n \cdot t). \tag{3}$$

Often in practical application, one does not have an exact analytical function, but a series of discrete data points. If it is necessary to convert this discrete data into the spectral domain, the traditional approach has been to use the Discrete Fourier Transform, often known as Fast Fourier Transform (FFT). The FFT algorithm is, by definition [10, 11]

$$F_k = \Sigma_{n=0}^{N-1} x_n \cdot e^{-2\pi \cdot i \cdot k \cdot n/N}. \tag{4}$$

where F_k is a discrete spectral data point, and x_n is a discrete data point in the temporal domain. With DFT, the spectral resolution is proportional to the temporal resolution, and it is often the case that the limited temporal data will not be sufficient to obtain the spectral resolution desired.

This algorithm is an approach to obtain greater spectral resolution; the full spectral domain, or any frequency range or resolution desired, is determined by the user. Greater resolution or a larger domain will inherently take longer to solve, depending on the computer resources available. One advantage of this approach is that the spectral domain can also have varying resolutions, for enhanced resolution at points of interest without dramatically increasing the computation cost of each Fourier Transform.

At each discrete point in the spectral domain, the algorithm generates two sinusoidal functions

$$\begin{aligned} F_n(t) &= A{\cdot}sin(2\pi\omega_n t), \\ \hat{F}_n(t) &= A{\cdot}cos(2\pi\omega_n t), \end{aligned} \tag{5}$$

where $F_n(t)$ is to represent the real spectral components, $\hat{F}_n(t)$ is to represent the imaginary spectral components, ω_n is the discrete frequency of interest, t is the independent variable of the data of interest, and A is the amplitude of the function,

$$A = max\{f(t)\} - min\{f(t)\}, \tag{6}$$

defined and the total range within the temporal data.

The next step is to take each of these functions, and find the *Coefficient of Determination* (CoD) between the function and the temporal data, all with the same temporal domain and resolution [8, 12–15]. The CoD is a numerical representation of how much variance can be expected between two functions. To find the CoD between two equal-length discrete functions $F_n(t)$ and $f_n(t)$, three coefficients are first calculated

$$\begin{aligned} SS_t &= \Sigma_{n=1}^{N}(F_n(t) - \bar{F_n}){\cdot}(f_n(t) - \bar{f_n}), \\ SS_1 &= \Sigma_{n=1}^{N}(F_n(t) - \bar{F_n})^2, \\ SS_2 &= \Sigma_{n=1}^{N}(f_n(t) - \bar{f_n})^2, \end{aligned}$$

where N is the discrete length of the two functions, and $\bar{F_n}$ and $\bar{f_n}$ represent the arithmatic mean value of functions F_n and f_n. The CoD is then determined as

$$CoD = \frac{SS_t}{\sqrt{SS_1 \cdot SS_2}}, \tag{7}$$

and the closer the two functions match, the closer the value of the CoD reaches 1. If there is no match at all, the CoD will be equal to 0, and if the two functions are perfectly opposite of each other ($F_n = -f_n$), the CoD goes up to -1. In practice, the CoD is often represented as the R^2 value,

$$R^2 = \frac{SS_t^2}{SS_1 \cdot SS_2}. \tag{8}$$

This process is repeated for every sine and cosine function generated with each frequency within the spectral domain. The coefficients of determinations can be used to represent the spectral values, both real and imaginary, for the given discrete frequency point. These functions of R^2 values for the real and imaginary components are then normalized to the maximum real and imaginary values, and multiplied by the amplitude A determined in equation 6. The final outcome is a phase-resolved spectral transformation of the input function, but with a spectral domain as large or resolved as desired.

Finally, this spectral transformation can easily be converted back to the temporal domain. By definition, the temporal domain is merely the sum of the series of sinusoidal waves, and thus the inverse Fourier transform can simply be defined as

$$f(t_n) = \Sigma_{m=1}^{N}\{real(F_m) \cdot cos(\omega_m \cdot t_n)\} + \{imag(F_m) \cdot sin(\omega_m \cdot t_n)\}. \tag{9}$$

Parametric Study

A parametric study of this transform was conducted, to demonstrate that it can be used for high resolution measurements of the spectral frequency with a limited temporal resolution. To demonstrate this, 15 random frequencies were selected, ranging from 2 to 17 cycles over the duration of the measured window. Both the independent and dependent temporal variables are arbitrary values to demonstrate the transform function; the independent scale ranges from 0 to 1 and has 180 data points. The arbitrary dependent data had random averages between -1000 and 1000, with an amplitude of 200 and random noise to represent the typical randomness found in typical test data. Each of these 15 random frequencies was phase shifted by three random phases. All forty-five arbitrary functions were transformed into the spectral domain with this transform, with a frequency domain

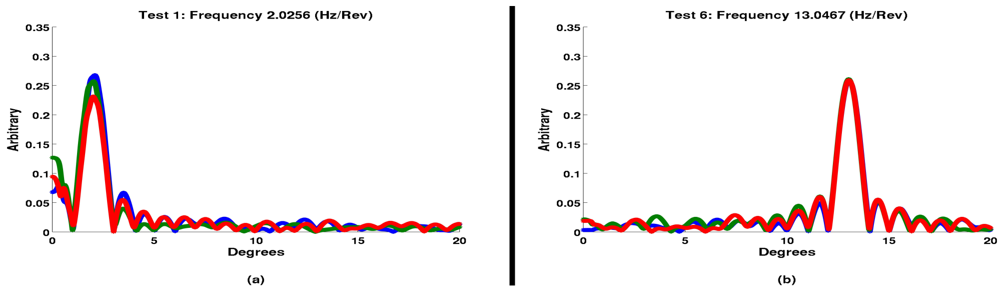

Figure 1: Spectral results of the randomly generated functions, for frequencies of (a) 2.0256 (Hz/Rev) and (b) 13.0467 (Hz/Rev), but for different phases, magnitudes, and random noises.

ranging from 0 to 20 cycles per unit time duration, and a frequency resolution of 1 mHz; two examples of these spectral results are presented in Figure 1. As a further test of the robustness of the transform, the spectral data was then converted back to the temporal domain, and the new temporal function was compared to the original function with the coefficient of determination method to ascertain errors from the transform.

This Fourier transform was remarkably effective at finding the peak primary frequency, often with accuracy's down to tens of mHz. The functions of the peak frequencies (Figure 2), both which was used for the initial function and the peak of the Fourier transform, matches with an R^2 value of 0.999991; effectively identical. The functions of the random phase angle at the peak frequencies (Figure 3), both which was used for the initial function and the phase of the Fourier transform at the peak frequency, matches with an R^2 value of 0.9982; demonstrating that this transform can be used to capture both spectral magnitude and phase with great accuracy.

Finally, the inverse of this Fourier transform was conducted for each spectral output, and the errors between the original functions and the transformed-inverse-transformed function are minimal. As expected, not all of the fine random noise is captured; this would require a near infinite spectral domain, which would further increase computational costs, but the overarching shapes, magnitudes, and phases of the functions are consistently captured. Taking the coefficient of determination squared of each function pair, the value of R^2 is never less than 0.92. Two examples of the original function (lines) and the transformed-inverse-transformed function (stars) are represented in Figure 4. The tabulated results of all fifteen studies, for each of the three phase magnitude shifts, are demonstrated in Tables 1 - 3.

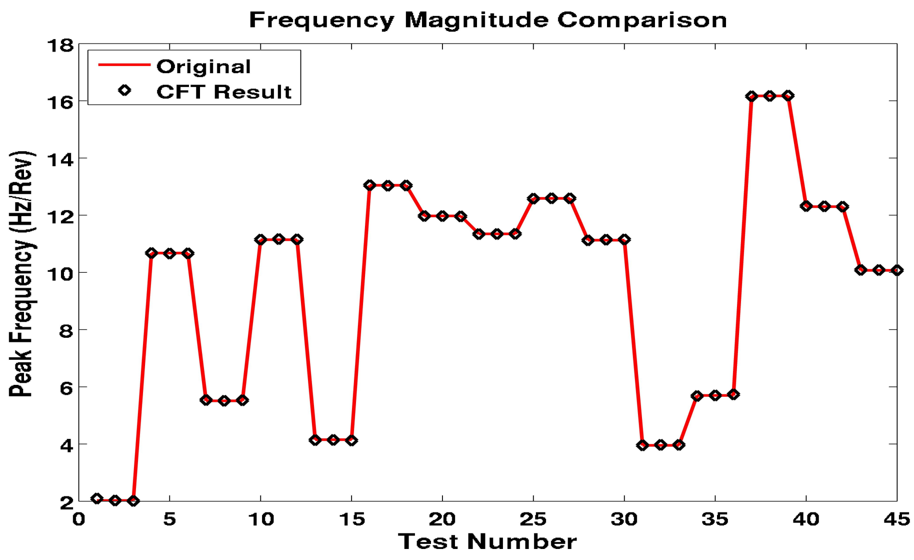

Figure 2: Frequency Prediction Results, $R^2 = 0.999991$

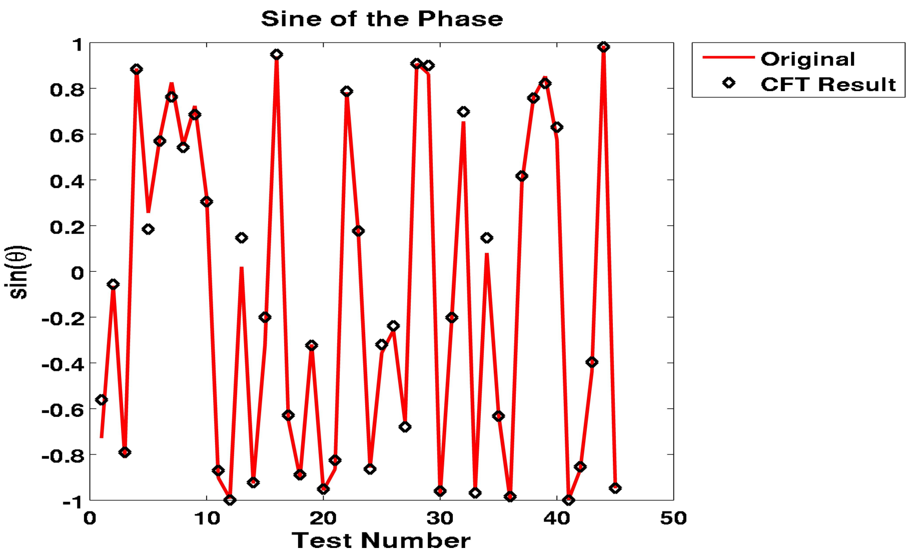

Figure 3: Angle Prediction Results, $R^2 = 0.9982$

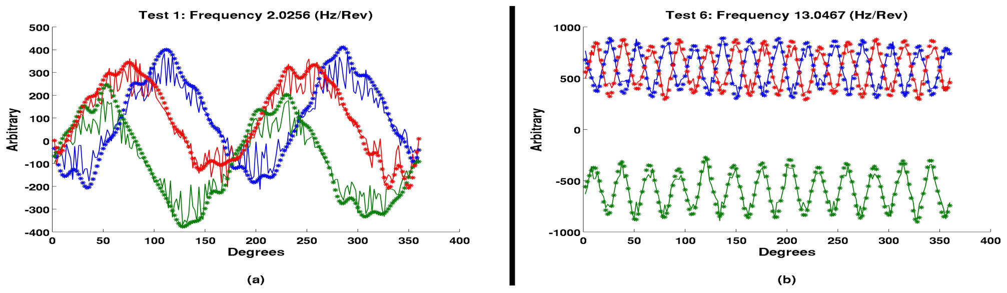

Figure 4: Time results of the randomly generated functions, for frequencies of (a) 2.0256 (Hz/Rev) and (b) 13.0467 (Hz/Rev), but for different phases, magnitudes, and random noises.

Conclusion

This effort has demonstrated a practical, working, novel method of numerically conducting a Fourier Transform when there is limited temporal resolution. All that is necessary for this transform to be effective is a temporal domain that can capture, to some extent, the individual cycles. While the transform inherently is more computationally expensive than traditional DFT methods, any desired spectral resolution and spectral domain can be used to characterize the input data; the transform can even convert the function to a spectral domain of varying resolution, so that peaks can be accurately identified without too much computational expense. The algorithm was tested at fifteen different random frequencies, all with three different random phases, all with random noises and errors, and consistently the transform was able to characterize the peak frequency and phase angle remarkably, with a higher degree of accuracy than one can expect with traditional DFT methods.

Acknowledgments

The author would like to acknowledge Lou Vocaturo and Kevin Larkins for useful discussions.

[1] Nagel RK, Saff EB, Snider AD. Fundamentals of Differential Equations, 5^{th} Edition. 75 Arlington Street, Suite 300 Boston, MA 02116: Addison Wesley; 1999.

[2] Haberman R. Applied Partial Differential Equations With Fourier Series and Boundary Value Problems, 4^{th} Edition. Upper Saddle River, New Jersey: Prentice Hall; 2003.

Test	Max Freq Original	Max Freq CFT Result	sin(Phase) Original	sin(Phase) CFT Result	R^2 (Temporal)
1	2.0256	2.095	-0.72837	-0.56129	0.92791
2	10.6771	10.678	0.88707	0.88349	0.94843
3	5.5118	5.537	0.82648	0.76277	0.94092
4	11.1441	11.146	0.32585	0.30499	0.93117
5	4.1527	4.137	0.020562	0.14696	0.93132
6	13.0467	13.05	0.94015	0.94829	0.93359
7	11.9742	11.973	-0.31888	-0.32338	0.92769
8	11.3472	11.339	0.78318	0.78725	0.93398
9	12.5892	12.578	-0.35645	-0.31945	0.93956
10	11.128	11.122	0.90871	0.90812	0.92836
11	3.9531	3.954	-0.20131	-0.20166	0.93636
12	5.6981	5.677	0.080619	0.14678	0.93363
13	16.1721	16.158	0.38625	0.41684	0.94243
14	12.2989	12.32	0.57446	0.63023	0.93483
15	10.0715	10.085	-0.43758	-0.3967	0.92398

Table 1: Comparison of Results, for Phase Shift Angle 1.

[3] Harris FJ. On the Use of Windows for Harmonic Analysis with the Discrete Fourier Transform. Proceedings of the IEEE. January 1978;66(1):51–83.

[4] Dorrer C, Belabas N, Likforman JP, Joffre M. Spectral resolution and sampling issues in Fourier-transform spectral interferometry. Journal of Optics Society of America B. 2000;17(19):1795–1802.

[5] Arfken GB, Weber HJ. Mathematical Methods for Physicists, Sixth Edition. 30 Corporate Drive, Suite 400, Burlington MA 01803: Elsevier; 2005.

[6] Zill DG, Cullen MR. Advanced Engineering Mathematics, Second Edition. Sudbury MA: Jones and Bartlett Publishers; 2000.

[7] Chen WH, Smith CH, Fralick SC. A Fast Computational Algorithm for the Discrete Cosine Transform. IEEE Transactions on Communications. September 1977;25(9):1004–1009.

[8] Agarwal RC. A New Least-Squares Refinement Technique Based on the Fast Fourier Transform Algorithm. Acta Cryst. 1978;34:791–809.

Test	Max Freq Original	Max Freq CFT Result	sin(Phase) Original	sin(Phase) CFT Result	R^2 (Temporal)
1	2.0256	2.026	-0.062681	-0.056313	0.93311
2	10.6771	10.658	0.25575	0.18466	0.93944
3	5.5118	5.511	0.54959	0.54223	0.9434
4	11.1441	11.166	-0.90216	-0.87023	0.93194
5	4.1527	4.141	-0.92686	-0.92268	0.92885
6	13.0467	13.031	-0.64986	-0.6287	0.92476
7	11.9742	11.97	-0.95715	-0.95146	0.93294
8	11.3472	11.344	0.16368	0.17658	0.92725
9	12.5892	12.593	-0.25732	-0.23808	0.94473
10	11.128	11.145	0.86293	0.90011	0.93841
11	3.9531	3.97	0.65591	0.69795	0.94498
12	5.6981	5.7	-0.62372	-0.63233	0.93151
13	16.1721	16.169	0.76013	0.75789	0.93039
14	12.2989	12.31	-0.99804	-1	0.93751
15	10.0715	10.084	0.9865	0.98098	0.9411

Table 2: Comparison of Results, for Phase Shift Angle 2.

[9] Finzel B. Incorporation of fast Fourier transforms to speed restrained least-squares refinement of protein structures. Journal of Applied Cryst. 1986;20:53–55.

[10] Garcia A. Numerical Methods for Physics, Second Edition. 75 Arlington Street, Suite 300 Boston, MA: Addison-Wesley; 1999.

[11] Poon TC, Kim T. Engineering Optics With Matlab. 27 Warren St, Hackensack, NJ 07601: World Scientific Publishing Co; 2006.

[12] Cameron AC, Windmeijer FAG. An R-squared measure of goodness of fit for some common nonlinear regression models. Journal of Econometrics. 1997;77:329–342.

[13] Magee L. R2 Measures Based on Wald and Likelihood Ratio Joint Significance Tests. The American Statistician. August 1990;44(3):250–253.

[14] Nagelkerke NJD. A note on a general definition of the coefficient of determination. Biomelrika. 1991;78(3):691–692.

[15] Strang G. Introduction to Linear Algebra, 3^{rd} Edition. 7 Southgate Rd, Wellesley, MA 02482: Wellesley-Cambridge Press; 2003.

Test	Max Freq Original	Max Freq CFT Result	sin(Phase) Original	sin(Phase) CFT Result	R^2 (Temporal)
1	2.0256	2.009	-0.81562	-0.79053	0.93367
2	10.6771	10.666	0.58339	0.5697	0.92987
3	5.5118	5.53	0.72393	0.68535	0.93461
4	11.1441	11.154	-0.99286	-0.99909	0.92839
5	4.1527	4.122	-0.3222	-0.20013	0.95071
6	13.0467	13.04	-0.90228	-0.88856	0.94272
7	11.9742	11.958	-0.86461	-0.82531	0.93705
8	11.3472	11.347	-0.85766	-0.86378	0.93519
9	12.5892	12.588	-0.66127	-0.67959	0.92307
10	11.128	11.152	-0.9832	-0.96008	0.9276
11	3.9531	3.969	-0.9574	-0.96818	0.9428
12	5.6981	5.734	-0.99907	-0.9842	0.94156
13	16.1721	16.183	0.85337	0.82142	0.93247
14	12.2989	12.296	-0.86634	-0.85394	0.92086
15	10.0715	10.075	-0.94753	-0.9473	0.94203

Table 3: Comparison of Results, for Phase Shift Angle 3.

MkRndDat.m

```
clear all
close all

t=2:2:360;
raw=zeros(180,55);
raw(:,1)=t;

RnFct=(rand(1,18)).*sin(2*pi*(1:18)/9)*(1e3);

for ii=1:18
    oo=((ii-1)*3)+(1:3)+1;
    Mag=RnFct(ii);
    for jj=1:3
        A=sin(2*pi*t/40);
        A=A+(sin(2*pi*((2*rand(1,180))-1))).*((2*rand(1,180))-1)/10;
        A=A.*(1+(((2*rand(1,180))-1)*0.5));
        A=A*(2e2);
        A=A+(Mag*((-1)^(jj-1)));
        raw(:,oo(jj))=A';
    end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%

save SGdat raw
```

Crunch.m

```
clear all
close all
tic

load SGdat

rr=54;
Sfct=0:0.01:20;
ct=length(Sfct);
t=raw(:,1)/360; ctT=length(t);

i=sqrt(-1);
R2=zeros(rr,1);
SpecFctParam=zeros(ct,rr);
FtestParam=zeros(ctT,rr);
for oo=1:rr
    FF0=raw(:,oo+1);
    FFavg=mean(FF0);
    FFstd=max(FF0)-min(FF0);

    SpecFct=zeros(ct,1);
    for ii=2:ct
        display([num2str(ii) '/' num2str(ct) '   ' 9 num2str(oo) '/' num2str(rr)]);
        sinfct=sin(2*pi*t*(Sfct(ii)));
        cosfct=cos(2*pi*t*(Sfct(ii)));
        corrR=R2fct(cosfct,FF0);
        corrI=R2fct(sinfct,FF0);
        SpecFct(ii)=corrR+(i*corrI);
    end
    SpecFct=FFstd*SpecFct/(sum(abs(SpecFct)));
    SpecFct(1)=FFavg;
    SpecFctParam(:,oo)=SpecFct;

    Ftest=zeros(ctT,1);
    for ii=1:ct
        Ftest=((real(SpecFct(ii)))*cos(2*pi*t*Sfct(ii)))+Ftest;
        Ftest=((imag(SpecFct(ii)))*sin(2*pi*t*Sfct(ii)))+Ftest;
    end
    FtestParam(:,oo)=Ftest;
    R2(oo)=(R2fct(Ftest,FF0))^2;
end
runtime=toc;
save Results
```

R2fct.m

```
function [corr]=R2fct(Exp,CFD)

ctx=length(Exp);

% Average Residual

R_avg=0;
for ii=1:ctx
    R_avg=(Exp(ii)-CFD(ii));
end
R_avg=R_avg/ctx;

R_std=0;
for ii=1:ctx
    R_std=R_std+(((Exp(ii)-CFD(ii))-(mean(R_avg)))^2);
end
R_std=sqrt(R_std/(ctx-1));

%%%%%%%%%%%%%%%%%%%%

% Correlation

foo=zeros(ctx,3);
for ii=1:ctx
    foo(ii,1)=(Exp(ii)-(mean(Exp)))*(CFD(ii)-(mean(CFD)));
    foo(ii,2)=(Exp(ii)-(mean(Exp)))^2;
    foo(ii,3)=(CFD(ii)-(mean(CFD)))^2;
end
foo=sum(foo);
corr=foo(1)/(sqrt((foo(2))*(foo(3)))); % Closer to 1 is best
R2=corr^2;

end
```