# Introducing the New World of Polyterms: New Tools for Polynomial Regression 

By

Namir Clement Shammas

## Introduction

Polynomials make a significant contribution to algebra, calculus, numerical analysis algorithms, and curve fitting. This report focuses on the curve fitting aspects of polynomials. The popular linear regression represents fitting data with the best line--the simplest polynomial. In 1975, HP introduced the HP-55 programmable calculator as the first calculator with built-in linear regression feature. Fitting with linear, quadratic, and cubic polynomials have also become more popular built-in features in the last few decades in many handheld graphing calculators. Much earlier in the fifties and sixties, mainframe computers were used to perform higher-degree polynomial fitting to approximate many popular math and statistical functions. Among these are the gamma function, the inverse normal function, the inverse student-t function, and Bessel functions, just to name a few. Many such approximations can be found in Handbook of Mathematical Functions with Formulas Graphs and Mathematical Tables by Abramowitz \& Stegun. Several Statistics and Math application manuals for vintage HP calculators also included polynomial approximations for various functions. Curve fitting with polynomials typically offer a convenient and easy approximation for calculation-intensive functions.

What is typical of regular polynomials is that their terms have powers that increase in a strictly linear fashion. The typical power increments are 1 or 2 , and start at indices 1 or 2. Challenging the linear progression of polynomial powers was the out-of-the-box spark that gave birth to a new class of quasi-polynomials.

After over three decades of working with linearized and polynomial regression, I began to think of new types of quasi-polynomials in the early years of 2000. I tested how well these quasi-polynomials did in fitting financial stocks data, especially compared to similar polynomials. The results were very encouraging in favor of my new quasi-polynomials. The idea that came to me is to devise quasi-polynomials
whose power progression is slower than that of regular polynomials by using noninteger powers.

Since 2004, I have been attending annual HHC conferences for fans of vintage programmable HP calculators. A question that was frequently asked is "If mathematicians like Newton, Gauss, and Euler had programmable calculators how would numerical analysis algorithms look like today?" Funny, that no one asked about the effects if these same mathematicians had today's laptops! I have expressed my opinion in these meetings that the legacy numerical analysis algorithms were influenced by the computing resources available to the legacy mathematicians. My guess, is that these mathematicians hired educated people who could reliably do long calculations by hand-not to mention the painful task of rechecking these calculations! I regard the Gaussian Quadrature and linear regression as examples of algorithms that pushed hand calculations to the limit. Even doing linear regression using slide rules or four function calculators, in the seventies, was a real chore! This report attempts to give an answer of how computer power can offer resources for more advanced curve fitting using quasi-polynomials.

## The Advent of Shammas Polynomials

In the HHC2008 conference at the HP offices in Corvallis, Oregon, I presented my first quasi-polynomial which I called the Shammas Polynomials. I chose to use my last name to avoid potential naming conflicts with other mathematicians and statisticians who may end up using the same descriptive name. The Shammas Polynomials have x values raised to non-integer powers. These powers are defined by a function $p(i)=\alpha^{*} f(i)+\beta$ (such that $\left.p(i)<i\right)$, where $f(i)$ is a function of the term index $i$, and $\alpha$ and $\beta$ are parameters that can be integers or floating-point numbers. For regular polynomials the most typical values for $\alpha$ and $\beta$ are 1 and 0 , respectively. The general syntax for Shammas Polynomials is:
$\mathrm{P}_{\mathrm{n}}(\mathrm{x})=\mathrm{a}_{0}+\mathrm{a}_{1} * \mathrm{x}^{\mathrm{p}(1)}+\mathrm{a}_{2} * \mathrm{x}^{\mathrm{p}(2)}+\ldots+\mathrm{a}_{\mathrm{n}} * \mathrm{x}^{\mathrm{p}(\mathrm{n})}$
The most common version of $p(i)$ that $I$ have used is the simple $p(i)=\alpha * i+\beta$. One can use other equations for $p(i)$ such as $\alpha^{*} \ln (i+1)+\beta$, $\alpha / i+\beta$, and $\alpha^{*} \sqrt{i}+\beta$. In each case, the values of parameters $\alpha$ and $\beta$ are chosen to yield a good sequence of powers. The difference in powers between neighboring terms is less than 1 (as opposed to 1 or 2 in regular polynomials). One should avoid sequences of powers that are way too
close to each other. Using $p(i)=i / 2$ is a special case that allows you to use regular polynomial curve fitting tools by simply supplying the square root of the original independent variable, $x$, as the transformed independent variable, $x t$. You can likewise supply cube roots and other roots of x as the transformed independent variable. The rule of thumb for the $p(i)$ function is that it should generate a sequence of powers that are either consistently increasing or decreasing. Determining the best values for parameters $\alpha$ and $\beta$, and also the regression coefficients, involves using optimization and multiple/polynomial regression. You can replace the optimization step with using two nested loops (one that iterates over a range of values for parameter $\alpha$ and the other for the values of parameter $\beta$ ) to find the best optimized values of $\alpha$ and $\beta$. Using the function $p(i)$ essentially instills a new tempo for the progression of the sequence of powers in the Shammas Polynomials. This tempo is meant to slow down the increase in powers for the various terms. As such, the Shammas Polynomials are meant for raising numbers to lower powers and avoid possible rounding errors in regression calculations. I also have applied the notion of this new tempo to Fourier series.

Thus, the Shammas Polynomials represent the first example of polyterms. The creation of Shammas Polynomials easily led to the rational Shammas Padé polynomials which divide two Shammas Polynomials. Keep in mind that Polyterms are non-orthogonal quasi-polynomials where the powers used with the various terms have a pattern and are not chosen willy nilly.

While regular polynomials accept all values for the independent variable, the polyterms have some limitations in that regard. Some polyterms exclude zero and/or negative values for the independent variable. Other polyterms work strictly with positive values for the independent variable. In this case, you need to map the original values of the independent variable, that have non-positive values, to values in the range of $(1, \omega)$ where $\omega$ is 2 or higher, using the following equation:
$\mathrm{xt}=(\mathrm{x}-\min (\mathrm{x})) /(\max (\mathrm{x})-\min (\mathrm{x}))+\omega-1$
Keeping the value of $\omega$ low prevents higher power of the Shammas Polynomials from creating large numbers. For example, $2^{4}$ is 16 , while $10^{4}$ is 10000 -a big difference! Compressing the range of the independent variable has a down side. It amplifies how relatively small value changes of the independent variable affect the values of the dependent variable.

## The Advent of More Polyterms

More recently, I devised a new type of polyterms which I called the Quantum Shammas Polynomials. Quantum Shammas Polynomials are inspired by how quantum physics views the probabilistic orbits of the electrons in an atom. These non-orthogonal polynomials have nothing to do with the new art of quantum computing per se. Early on, scientists assumed that the electrons in an atom had distinct orbits that were thought to be fixed. This concept parallels the fixed powers of classical polynomials. By contrast, the Heisenberg uncertainty principle suggests that the orbits of the electrons are more probabilistic than fixed. This is the inspiration for Quantum Shammas Polynomials. While classical polynomials have the familiar fixed integer powers, the non-orthogonal Quantum Shammas Polynomials have random powers that typically vary closely above and below integer powers. For examples they can use ranges between $(i-1)+0.5$ to $i+0.4$ where $i$ is the term number. The general form of the Quantum Shammas Polynomial is:

$$
\begin{equation*}
y(x)=a_{0}+a_{1} * x^{R 1}+a_{2} * x^{R 2}+\ldots+a_{n} * x^{R n}, \text { for } x>=0 \tag{3}
\end{equation*}
$$

Where $0.5 \leq \mathrm{R}_{1} \leq 1.4,1.5 \leq \mathrm{R}_{2} \leq 2.4, \ldots$, and $(\mathrm{n}-1)+0.5 \leq \mathrm{R}_{\mathrm{n}} \leq \mathrm{n}+0.4$. Notice that the upper value of a random power is 0.1 less than the lower value of its successor. This gap ensures that no two random powers have the same exact value. I chose the above ranges for the random powers $r_{i}$ as arbitrary values (a kind of starting point or first run, if you will) and are by no means set in stone. You can narrow the range of random powers by using a scheme like ( $n-1$ ) $+0.6 \leq R_{n} \leq n+0.3$ to play it safer by using a wider gap between the powers of neighboring terms. The values of the random powers $\left(\mathrm{R}_{\mathrm{i}}\right)$ are chosen to minimize the sum of errors squared for some observed values of $\mathrm{y}(\mathrm{x})$. This minimization process involves optimization using either an optimization algorithm or random search. The latter method is feasible in the case of Quantum Shammas Polynomials because the ranges for the random powers are relatively small.

The creation of Quantum Shammas Polynomials easily led to the rational Quantum Shammas Padé polynomials which divide two Quantum Shammas Polynomials.

Another type of polyterms that I have studied is the symmetrical extended polynomials. It has an equal number of terms with positive and negative powers.

Thus, you can define symmetrical extended polynomials with a single order. Half of that number specifies the positive powers and the other half specifies negative powers. These powers can be:

1. Simple negative and positive integer powers. Here is an example of a fourth order symmetrical polynomial:

$$
y(x)=b_{2} / x^{2}+b_{1} / x+a_{0}+a_{1} * x+a_{2} * x^{2}
$$

2. Powers that use the Shammas Polynomial power function $p(i)$ with terms having positive and negative non-integer powers. Here is an example of a fourth order symmetrical Shammas polynomial using $p(i)=i+0.5$ :

$$
y(x)=b_{2} / x^{2.5}+b_{1} / x^{1.5}+a_{0}+a_{1} * x^{1.5}+a_{2} * x^{2.5}
$$

3. Powers that use the Quantum Shammas Polynomial random power value $R_{i}$ with terms having positive and negative non-integer powers. Here is an example of a fourth order symmetrical Quantum polynomial:

$$
y(x)=b_{2} / x^{1.9}+b_{1} / x^{1.1}+a_{0}+a_{1} * x^{1.1}+a_{2} * x^{1.9}
$$

Asymmetrical extended polyterms differ in that the number of terms with positive and negative powers are unequal. They require two orders to specify the number of terms with positive and negative powers. So, for each $\eta$ symmetrical extended polyterms there are $\eta^{2}$ asymmetrical extended polyterms.

Polyterms are not limited to just one independent variable. I am also studying polyterms with two and three independent variables that include cross product terms of two or three independent variables. The number of terms in these multivariate polyterms easily runs high. That is why I limit the orders of these multivariate polyterms to 2 and 3 . The simple form of multivariate polyterms uses a single order that is applied to each independent variable. A more advanced version applies a separate order to each independent variable.

The above types of polyterms are sample examples of polyterms in general. If polynomials are akin to impressive multi-stories huge cruise ships, then polyterms are the vast oceans these cruise ships navigate. Polyterms are non-orthogonal quasi-

## Introducing the New World of Polyterms

polynomials that are limited by our imagination! I hope my comments here open the door for more powerful curve fitting tools.

## A Question!

A question that poses itself. How well do polyterms fit data? The answer is neither regular polynomials nor polyterms are superior in fitting ALL functions and data. The nature of the data (whether calculated or observed) and the errors associated (whether from approximation or observation) will determine the winner. Functions and low-error data that are analytically approximated by polynomials will naturally favor regular polynomials. Polytems provide a worthwhile challenge to polynomials in curve fitting. One needs to test and compare a polyterm and its comparable polynomial (with the same order) to determine which of the two provides a better curve fit. Also, keep in mind that when polyterms or polynomials offer a better curve fit, the best adjusted coefficient of determination may still be disappointing (especially less than 0.9). Better may not always mean a spectacular fit! When approximating functions, one aims at getting adjusted coefficient of determination values with as many 9 s after the decimal place as possible. This is important especially when you want to replace a calculation-intensive function with a very good approximation using a polynomial or a polyterm.

My website (https://www.namirshammas.com/NEW/mainNEW.htm) should have several entries for various kinds of polyterms. Enjoy!

[^0]
[^0]:    Do not mix between polyterms and mathematical expressions (in a general sense). The following is an example of a mathematical expression (or we can call it an arbitrary polyterm):
    $\sqrt{x}+x^{2}+1 / x$
    While this is an example of a polyterm:
    $7 / \mathrm{x}^{2.1}+4 / \mathrm{x}^{1.1}+5+2^{*} \mathrm{x}^{1.1}+3 * \mathrm{x}^{2.1}$

