Journal of Interpolation and Approximation in Scientific Computing 2013 (2013) 1-16

ISPACS
International Scientific Publications and Consulting Services

Available online at www.ispacs.com/jiasc

Volume 2013, Year 2013 Article ID jiasc-00046, 16 Pages

doi:10.5899/2013/jiasc-00046

Research Article

Journal of Interpolation and Approximation in Scientific Computing

Least-squares fit of a linear combination of functions

Niraj Upadhyay[1*], Debajyoti Gangopadhyay[2]

(1) *20 Digvijay Plot, Jamnagar - 361005, India.*

(2) *Annada College, Vinoba Bhabe University, Hazaribag, India.*

Abstract

We propose that given a data-set $S = \{(x_i, y_i)/i = 1,2,\ldots n\}$ and real-valued functions $\{f_\alpha(x)/\alpha = 1,2,\ldots m\}$, the least-squares fit vector $A = \{a_\alpha\}$ for $y = \sum_\alpha a_\alpha f_\alpha(x)$ is $A = (F^T F)^{-1} F^T Y$ where $[F_{i\alpha}] = [f_\alpha(x_i)]$. We test this formalism by deriving the algebraic expressions of the regression coefficients in $y = ax + b$ and in $y = ax^2 + bx + c$. As a practical application, we successfully arrive at the coefficients in the semi-empirical mass formula of nuclear physics. The formalism is *generic* – it has the potential of being applicable to any *type* of $\{x_i\}$ as long as there exist appropriate $\{f_\alpha\}$. The method can be exploited with a CAS or an object-oriented language and is excellently suitable for parallel-processing.

Keywords: "Algebraic regression", "least-squares".

1 Introduction

Science, as an organised body of thought, existed in many ancient civilisations. Some of these civilisations hypothesised that all matter was made up from a very small number, typically three or five, of fundamental "elements". It is known that Empedocles, Plato and Aristotle held such a view [1]. Many philosophers, notably Democritus [1], believed in the concept of an "atom" – the smallest indivisible entity. However, the modern scientific theory of an atom was proposed in 1911 by Rutherford [2] on the basis of α-scattering experiments. In 1913, Niels Bohr [3] developed a quantum theory of the atom which accounted for many of the properties of atomic spectra.

It came to be held that the atom had a tiny positively charged nucleus surrounded by an appropriate number of negatively charged electrons in orbits so that the atom, as a whole, would be an electrically neutral system. In 1929, George Gamow [1] proposed that the nucleus was a ball of α-particles and that this system could possess forces similar to those of a liquid-drop – an *incompressible* "fluid" of constant density, being spherical in shape, possessing a surface-tension and which can undergo fission (radioactive decay), much along the way of a drop of water.

After the discovery of the neutron [4], Werner Heisenberg suggested that the nucleus constituted of protons and neutrons and that these interact via "exchange forces" – a precursor of the modern idea of *gauge-bosons*. Heisenberg's student Weizsacker [5] derived an *empirical* equation that could yield the measure of the bond-strength of a nucleus, incorporating into the equation the various properties that characterise a liquid drop. Hans Bethe and Robert Bacher reworked the equation to include a "pairing energy". In 1939, Bohr and Wheeler [7], successfully applied the concept of such a liquid-drop to understand nuclear fission. This is the foundation of the liquid-drop model (LDM) of the nucleus. Weizsacker's equation, generally referred as the "semi-empirical mass formula" (SEMF) or "Bethe-Weizsacker

*Corresponding author. Email address: drniraj_jam@bsnl.in, Tel:+912882670665

formula", is an integral part of the model.
The LDM is useful in the theory of elements heavier than Iron, phenomenological analysis of nuclear fission, the deformations and details of the division of the nuclear-drop. Over the years, many refinements [8] [9] [10] have been included to accommodate new experimental results, and the LDM and the SEMF are still relevant for capturing certain aspects of nuclear physics such as interpolating or extrapolating to find the mass of previously unknown nuclides, explaining the correlation between isobaric-stability and β-decay [11]. It is an excellent example of a semi-classical treatment of an inherently complex quantum-mechanical system.
In Mendeleev's periodic table of chemical elements, each element has an atomic number Z and an atomic mass number A. This number A has nothing in common to the fit-vector A mentioned in the Abstract and which appears in Theorem (2.1). The symbol used to stand for "Atomic weight" and later for "Atomic Mass number".
The nucleus of every atom of an element has a Z positively-charged protons and $N = A - Z$ electrically-neutral neutrons. The proton and the neutron are customarily called nucleons. So, $A = N + Z$ is also called the "nucleon number". On the basis of this simple relation, nuclear properties are quoted either as functions of (N,Z) or of (A,Z).
Let m_p be the mass of the proton and m_n be that of a neutron. Experimentally it has been found that the *measured* mass $M(N,Z)$ of the nucleus of an element is *less* than the sum $Nm_n + Zm_p$ by an amount $\Delta M(N,Z)$ which is called the "mass deficit". By the mass-energy equality $E = mc^2$, this deficit can be expressed as the *Nuclear Binding Energy*

$$\begin{aligned} E_B(N,Z) &= c^2 \Delta M(N,Z) && (1.1) \\ &= Zm_p + Nm_n - M(N,Z) && (1.2) \end{aligned}$$

where we have cast m_p, m_n and $M(N,Z)$ into units of energy. The equation, without loss of too much accuracy, is written as

$$E_B(A,Z) = Zm_H + (A-Z)m_n - M_{atom}(A,Z) \tag{1.3}$$

with m_H being the mass of the Hydrogen atom and $M_{atom}(A,Z)$ is the atomic mass of the isotope under consideration. One can use the preceding equation to calculate nuclear binding-energy E_B from Atomic Mass Evaluations [17]. The most appealing aspect of the SEMF is that it can be written in a simple form [14] consisting of five terms:

1. *Volume term:* The plot of the measured binding energies $E_B(A)$ against the nucleon-number A increases monotonically and is almost linear. As per the LDM, the volume V of a nucleus is proportional to A. Therefore, there is a "volume-energy" term contributing to $E_B(A,Z)$:

$$E_V \propto A \tag{1.4}$$

2. *Surface term:* For an incompressible liquid-drop, the volume $V \propto A$, the radius $r \propto V^{1/3}$ and the surface-area $S \propto r^2$. The liquid-drop has a surface-tension, the corresponding energy being proportional to the surface area or, equivalently, to the number of nucleons on the surface. Thus, there is a corrective "surface-energy" contribution to $E_B(A,Z)$:

$$E_S \propto A^{2/3} \tag{1.5}$$

3. *Coulomb term:* Each of the Z protons repels the $(Z-1)$ others. Consequently, the total number of proton-proton pairs is $Z(Z-1)/2$. The nuclear radius is effectively $r \propto V^{1/3} \propto A^{1/3}$. The correct electrostatic Coulomb charge distribution and the pertinent geometrical factors have been established [6]. Thus, there is a corrective "Coulomb energy" contribution to $E_B(A,Z)$:

$$E_C \propto \frac{Z(Z-1)}{A^{1/3}} \tag{1.6}$$

4. *Asymmetry term:* It is observed that the lightest nuclei are stable when $N = Z$. In the heavier nuclei, there is a *neutron-excess* $T = N - Z$ which counters the increasing Coulomb repulsion. The nucleon-nucleon cohesion (exchange force) tends to provide stability against fragmentation. The energy-contribution due to such an exchange-force can be represented by $|T|$. However, it is *chosen* to be proportional to T^2 so that it is zero along the $N = Z$ line (in the N-Z plane), is symmetrical about this line (required to interpret experimental observations)

and the power two makes analytic calculations (e.g. differentiation) easy. Further, it is necessary to "normalise" the contribution against the distance from the "valley of stability". This is achieved by introducing $(N+Z)$ in the denominator which also makes the algebraic-expression symmetric in N and Z. Such a heuristically defined term, the "asymmetry term", corrects $E_B(A,Z)$ by an amount

$$E_A \propto \frac{(N-Z)^2}{N+Z} \tag{1.7}$$

$$E_A \propto \frac{(A-2Z)^2}{A} \tag{1.8}$$

5. *Pairing term:* Laboratory observations (Table 9-2, [12]) indicate that even-even nuclei are more stable than odd-even nuclei which, in turn, are more stable than odd-odd nuclei. Consequently, there has to be a corrective term. Unfortunately, in the absence of a "standard model" of nuclear physics, authors differ on the representation of this term [13]. There are three commonly used dependences in the literature: A^{-1} [12], $A^{-3/4}$ and $A^{-1/2}$ [14]. As we shall appreciate in section (4), these predict vastly differing values for the coefficient a_P without disagreeing upon the other coefficients or the relevant statistical deviation in the SEMF.

Finally, one may write down the SEMF [14] as a summation of the above five terms:

$$E_B(A,Z) = a_V A - a_S A^{2/3} - a_C \frac{Z(Z-1)}{A^{1/3}} - a_A \frac{(A-2Z)^2}{A} - \delta(A,Z) \tag{1.9}$$

where

$$\delta(A,Z) = \begin{cases} -\delta_0 & \text{even } N, \text{ even } Z \\ 0 & \text{otherwise} \\ +\delta_0 & \text{odd } N, \text{ odd } Z \end{cases} \tag{1.10}$$

with

$$\delta_0 = \frac{a_P}{A^{1/2}} \tag{1.11}$$

The coefficients a_V, a_S, a_C, a_A and a_P are usually determined by fitting experimental data to the formula and are typically of the order 10 MeV to 1 MeV. It has to be mentioned that reference [14] has an enlightening exposition of *why* these coefficients assume values of these orders.
In equation (1.9), one can "pull-out" a constant factor "$-a_P$" and cast the SEMF thus:

$$E_B(A,Z) = a_V A - a_S A^{2/3} - a_C \frac{Z(Z-1)}{A^{1/3}} - a_A \frac{(A-2Z)^2}{A} + a_P \delta_P(A,Z) \tag{1.12}$$

where

$$\delta_P(A,Z) = \begin{cases} +A^{-1/2} & \text{even } N, \text{ even } Z \\ 0 & \text{otherwise} \\ -A^{-1/2} & \text{odd } N, \text{ odd } Z \end{cases} \tag{1.13}$$

One can easily see that if one had an experimental data-base then the SEMF (1.12) is a candidate for least-squares fit to an equation like

$$y = \sum_{\alpha=1}^{m} a_\alpha f_\alpha(x) \tag{1.14}$$

Below, in section (2), we propose a formalism for doing the least-squares fit of such a linear combination of functions. The formalism can be used both to do fits in algebraic way by using CAS like Mathematica, Maxima or GiNaC and in numerical fashion by writing object-oriented code.
Once the formalism has been set, we *test* it on a previously known case. This is done in section (3) by applying the formalism to derive the algebraic expressions of the regression coefficients a and b of straight-line fit $y = ax + b$. Going one step further, we also derive the coefficients of the parabolic-fit $y = ax^2 + bx + c$.
In section (4), as a practical numerical application, we employ the formalism on the SEMF and compare the result against those in references [14] [15] [16].
In section (5), with parallel-processing in mind, we comment on the aspect of deploying this formalism on computers. Therein, wherever relevant, we also quote snippets of our functional source-code.

2 Least-squares fit

Theorem 2.1. *Let us consider the following:*

1. *A "database" set $\mathbb{X} \subset \mathbb{U}$ containing "records" from some suitable universal set.*
2. *$x_i \in \mathbb{X}$ and $y_i \in \mathbb{R}$ as part of input data-set $S = \{(x_i, y_i)/i = 1,2,\ldots n\}$. Let Y be the column-vector constructed from $\{y_i\}$.*
3. *$\{f_\alpha : \mathbb{X} \to \mathbb{R}/\alpha = 1,2,\ldots,m\}$ be provided as closed-form (machine-computable) functions.*
4. *The "functions matrix" $F_{n\times m} = [F_{i\alpha}] = [f_\alpha(x_i)]$ constructed from the input data. We need to assume that $G_{m\times m} = F^T F$ is* non-singular. *A template F, meant for future reference, appears thus :*

$$F = \begin{pmatrix} f_1(x_1) & . & . & . & f_m(x_1) \\ . & & & & . \\ . & & & & . \\ . & & & & . \\ f_1(x_n) & . & . & . & f_m(x_n) \end{pmatrix}_{n\times m} \tag{2.15}$$

5. *An unknown parameter-set $A = \{a_\alpha \in \mathbb{R}/\alpha = 1,2,\ldots,m\}$* to be determined *such that S is a least-squares fit to the relation*

$$y(x_i) = \sum_{\alpha=1}^{m} a_\alpha f_\alpha(x_i) \tag{2.16}$$

We need to assume that the a_α are uncorrelated*, i.e.,*

$$\frac{\partial a_\mu}{\partial a_\nu} = \delta_{\mu\nu} \tag{2.17}$$

Then the least-squares fit is achieved when

$$A = (F^T F)^{-1} F^T Y \tag{2.18}$$

Proof. Given a data-point (x_i, y_i), the error Δy_i in y is

$$\begin{aligned} \Delta y_i &= y(x_i) - y_i && (2.19)\\ &= \sum_{\alpha=1}^{m} a_\alpha f_\alpha(x_i) - y_i && (2.20) \end{aligned}$$

whose square is

$$\begin{aligned} [\Delta y_i]^2 &= \left[\sum_{\alpha=1}^{m} a_\alpha f_\alpha(x_i) - y_i\right]^2 && (2.21)\\ &= \left[\sum_{\alpha=1}^{m} a_\alpha f_\alpha(x_i)\right]^2 - 2\left[\sum_{\alpha=1}^{m} a_\alpha f_\alpha(x_i)\right] y_i + y_i^2 && (2.22)\\ &= \left[\sum_{\alpha=1}^{m} a_\alpha f_\alpha(x_i)\right]\left[\sum_{\beta=1}^{m} a_\beta f_\beta(x_i)\right] - 2\left[\sum_{\alpha=1}^{m} a_\alpha f_\alpha(x_i)\right] y_i + y_i^2 && (2.23) \end{aligned}$$

Within *summations*, let us choose the Greek indices to run from 1 to m and the Latin indices to run from 1 to n. Then equation (2.23) can be succinctly written as

$$[\Delta y_i]^2 = \sum_{\alpha\beta} a_\alpha f_\alpha(x_i) a_\beta f_\beta(x_i) - 2y_i \sum_{\alpha} a_\alpha f_\alpha(x_i) + y_i^2 \tag{2.24}$$

It then follows that the *sum* of these squared-errors is

$$\xi = \sum_i [\Delta y_i]^2 \tag{2.25}$$

$$= \sum_i \left[\sum_{\alpha\beta} a_\alpha f_\alpha(x_i) a_\beta f_\beta(x_i) \right] - 2 \sum_i \left[y_i \sum_\alpha a_\alpha f_\alpha(x_i) \right] + \sum_i y_i^2 \tag{2.26}$$

When one differentiates ξ with respect to any a_γ, the right-hand-most term in equation (2.26) will vanish. Also, the a_α are uncorrelated. Consequently, the first partial derivative of ξ with respect to any a_γ is

$$\frac{\partial \xi}{\partial a_\gamma} = \sum_i \frac{\partial}{\partial a_\gamma} \sum_{\alpha\beta} a_\alpha f_\alpha(x_i) a_\beta f_\beta(x_i) - 2 \sum_i y_i \frac{\partial}{\partial a_\gamma} \sum_\alpha a_\alpha f_\alpha(x_i) \tag{2.27}$$

$$= 2 \sum_i \left[\sum_\alpha a_\alpha f_\alpha(x_i) f_\gamma(x_i) - y_i f_\gamma(x_i) \right] \tag{2.28}$$

and the second (mixed) partial derivative is

$$\frac{\partial^2 \xi}{\partial a_\delta \partial a_\gamma} = 2 \sum_i \frac{\partial}{\partial a_\delta} \left[\sum_\alpha a_\alpha f_\alpha(x_i) f_\gamma(x_i) - y_i f_\gamma(x_i) \right] \tag{2.29}$$

$$= 2 \sum_i f_\delta(x_i) f_\gamma(x_i) \tag{2.30}$$

$$= \frac{\partial^2 \xi}{\partial a_\gamma \partial a_\delta} \tag{2.31}$$

We have implicitly assumed the $\{f_\alpha(x)\}$ to be continuous. But do they really need be so? A multi-variate extremum requires the second partial derivatives to be equal *and* continuous. In a given problem, the data-set S and the $\{f_\alpha\}$ are "held constant" – we are "varying" only the $\{a_\alpha\}$. We infer that the derivatives would be equal even if the $\{f_\alpha(x)\}$ are discontinuous, *but*, plausibly in a manner such that $\xi(a_\alpha)$ for a given problem is C^3. However, the question as to what is the nature of the *allowed* discontinuities – bounds and number of jumps – is a subject for further investigation. Picking up from equation (2.30), we find that

$$\frac{\partial^2 \xi}{\partial a_\gamma^2} = 2 \sum_i f_\gamma^2(x_i) \tag{2.32}$$

$$> 0 \tag{2.33}$$

Thus, under the given assumptions, the extremum solution is *guaranteed* to be a *minimum*. By equation (2.28), the requirement

$$\frac{\partial \xi}{\partial a_\gamma} = 0 \tag{2.34}$$

$$\Rightarrow \sum_i \left[\sum_\alpha a_\alpha f_\alpha(x_i) f_\gamma(x_i) \right] = \sum_i f_\gamma(x_i) y_i \tag{2.35}$$

$$\Rightarrow \sum_\alpha \sum_i f_\alpha(x_i) f_\gamma(x_i) a_\alpha = \sum_i f_\gamma(x_i) y_i \tag{2.36}$$

$$\Rightarrow \sum_\alpha \sum_i F_{\gamma i}^T F_{i\alpha} A_\alpha = \sum_i F_{i\gamma} Y_i \tag{2.37}$$

$$\Rightarrow \sum_\alpha \left(F^T F \right)_{\gamma\alpha} A_\alpha = \sum_i F_{\gamma i}^T Y_i \tag{2.38}$$

$$\Rightarrow \left(F^T F A \right)_\gamma = \left(F^T Y \right)_\gamma \tag{2.39}$$

Requiring an extremum with respect to variations of all a_γ,

$$F^T FA = F^T Y \tag{2.40}$$

Following our assumption that $G = F^T F$ is non-singular,

$$A = \left(F^T F\right)^{-1} F^T Y \tag{2.41}$$

□

3 Polynomial-fit

We shall now test this little discovery.

Example 3.1. *The least-squares fit of $y = ax + b$ on observations $S = \{(x_i, y_i)/i = 1, 2, \ldots, n\}$ is obtained when*

$$a = \frac{n\sum x_i y_i - \sum x_i \sum y_i}{n\sum x_i^2 - \left(\sum x_i\right)^2} \tag{3.42}$$

$$b = \frac{-\sum x_i \sum x_i y_i + \sum x_i^2 \sum y_i}{n\sum x_i^2 - \left(\sum x_i\right)^2} \tag{3.43}$$

Proof. Because theorem (2.1) works on indexed entities, let $a_1 = a$, $a_2 = b$ and

$$f_1(x_i) = x_i \tag{3.44}$$

$$f_2(x_i) = 1 \tag{3.45}$$

We can then re-write the fit-equation :

$$y = a_1 f_1(x) + a_2 f_2(x) \tag{3.46}$$

The matrices F and G are:

$$F = \begin{pmatrix} x_1 & 1 \\ . & . \\ . & . \\ . & . \\ x_n & 1 \end{pmatrix} \tag{3.47}$$

$$G = F^T F = \begin{pmatrix} \sum x_i^2 & \sum x_i \\ \sum x_i & n \end{pmatrix} \tag{3.48}$$

whose inverse can be calculated as

$$G^{-1} = \frac{\text{adj } G}{\det G} \tag{3.49}$$

which, by equation (2.18), generate the familiar formulae for a and b. □

One can similarly fit higher degree polynomials. We introduce a useful notation for this purpose.

Definition 3.1. *We shall denote an expression like $\sum_i^n x_i^p y_i^q$ by $(\mathbf{p}, \mathbf{q})$ and an expression like $\sum_i^n x_i^k$ by $\mathbf{k}$. In particular, a lone $\mathbf{0}$ is the number "n" of data-points.*

According to this terminology, the regression coefficients in equations (3.42) and (3.43) can be written as:

$$D_1 = \mathbf{2}\cdot\mathbf{0} - \mathbf{1}^2 \tag{3.50}$$

$$aD_1 = \mathbf{0}\cdot(\mathbf{1},\mathbf{1}) - \mathbf{1}\cdot(\mathbf{0},\mathbf{1}) \tag{3.51}$$

$$bD_1 = -\mathbf{1}\cdot(\mathbf{1},\mathbf{1}) + \mathbf{2}\cdot(\mathbf{0},\mathbf{1}) \tag{3.52}$$

Example 3.2. *Consider the parabola $y = ax^2 + bx + c$. The least-squares fit is obtained when*

$$D_2 = \mathbf{4}\{\mathbf{0}\cdot\mathbf{2} - \mathbf{1}^2\} - \mathbf{3}\{\mathbf{0}\cdot\mathbf{3} - \mathbf{1}\cdot\mathbf{2}\} + \mathbf{2}\{\mathbf{1}\cdot\mathbf{3} - \mathbf{2}^2\} \tag{3.53}$$

$$\begin{aligned} aD_2 &= \{\mathbf{0}\cdot\mathbf{2} - \mathbf{1}^2\}(\mathbf{2},\mathbf{1}) - \{\mathbf{0}\cdot\mathbf{3} - \mathbf{1}\cdot 2\}(\mathbf{1},\mathbf{1}) \qquad (3.54)\\ &+ \{\mathbf{3}\cdot\mathbf{1} - \mathbf{2}^2\}(\mathbf{0},\mathbf{1}) \\ bD_2 &= \{\mathbf{2}\cdot\mathbf{1} - \mathbf{0}\cdot\mathbf{3}\}(\mathbf{2},\mathbf{1}) + \{\mathbf{0}\cdot\mathbf{4} - \mathbf{2}^2\}(\mathbf{1},\mathbf{1}) \qquad (3.55)\\ &+ \{\mathbf{3}\cdot\mathbf{2} - \mathbf{4}\cdot\mathbf{1}\}(\mathbf{0},\mathbf{1}) \\ cD_2 &= \{\mathbf{1}\cdot 3 - \mathbf{2}^2\}(\mathbf{2},\mathbf{1}) + \{\mathbf{3}\cdot\mathbf{2} - \mathbf{4}\cdot\mathbf{1}\}(\mathbf{1},\mathbf{1}) \qquad (3.56)\\ &+ \{\mathbf{4}\cdot\mathbf{2} - \mathbf{3}^2\}(\mathbf{0},\mathbf{1}) \end{aligned}$$

Proof. For this case, the matrix G is

$$G = \begin{pmatrix} \mathbf{4} & \mathbf{3} & \mathbf{2} \\ \mathbf{3} & \mathbf{2} & \mathbf{1} \\ \mathbf{2} & \mathbf{1} & \mathbf{0} \end{pmatrix} \tag{3.57}$$

whose determinant D_2 is (3.53). Applying the formalism (2.18), one gets the proclaimed result for a, b, c. □

One can fit polynomials of degree three and higher. Without the aid of a CAS, the exercise can be tedious. A glance at equations (3.50) and (3.57) suggests that there is some "recursion" property involved and it should be possible to arrive at the fits of higher-degree polynomials merely by inspection.

4 SEMF

The heart of the current article, theorem (2.1), was motivated by an attempt to carry out a least-squares fit of the SEMF (1.12) on an appropriate data-base. Having successfully generated the regression-coefficients of polynomials in section (3), the aim of the exercise is to gauge the applicability of the theorem in a real-life *numerical* case for which there already exist independently evaluated results [14] [15] [16].
We performed such a fit using Atomic Mass Evaluation 2003 [17] [18] (AME03) to find the values of the coefficients $a_V, a_S, a_C, a_A,$ and a_P of the SEMF. Out of the 3179 nuclides in AME03, we chose a robust and reliable subset consisting of 1931 entries for applying the formalism. The nuclides in the subset range from H^3 to Md^{257} and are close to the "valley of stability". The result is given in Table.1. In order to be self-contained, we quote the entries of references [14] [15] [16] as well.

Table 1: The coefficients a_α of the SEMF.

(MeV)	Least Squares Fit	Wapstra	Rohlf	Our formalism
a_V	15.8	14.1	15.75	15.5
a_S	18.3	13	17.8	17.0
a_C	0.714	0.595	0.711	0.697
a_A	23.2	19	23.7	22.4
a_P	12	n/a	n/a	23.5

The formalism generates results concordant with Rohlf's result. The agreement with LSF is good, but for a_P. Why should two least-squares fit agree on all other coefficients but disagree so much on a_P? To address this issue, we need to explain how we had obtained our result. At the very outset, when we discovered discrepancy between the known result and our result, we carried out eight *runs* of our formalism and the results have been summarised in Table.2.

Table 2: SEMF in different regions with(out) pairing-term.

Run No.	Pairing term	Neutron excess	Samples	a_V	a_S	a_C MeV	a_A	a_P	rms keV
1	Absent	-ve	30	15.6	17.5	0.728	20.8	-	297
2		zero	26	13.4	12.9	0.434	-	-	634
3		+ve	1875	15.4	17.0	0.696	22.3	-	81.5
4		all	1931	15.5	17.1	0.698	22.4	-	79.9
5	Present	-ve	30	15.6	17.5	0.717	18.7	13.3	245
6		zero	26	13.8	13.5	0.465	-	22.6	260
7		+ve	1875	15.4	17.0	0.696	22.3	24.4	79.7
8		all	1931	15.5	17.0	0.697	22.4	23.5	77.8

In runs numbered 1 to 4 we had left out the pairing-term $a_P\delta_P(A,Z)$ but had retained it in runs numbered 5 through 8. For each of the two batches, we had considered four data sub-sets.

1. Only the (likely unstable) proton-rich nuclides which can be considered as possessing a negative "neutron excess". There are 30 such nuclides.
2. Only those nuclides which are on the $N = Z$ line. It is known that when such nuclides are considered, the Wigner-effect precludes accuracy by a big margin. There are 26 such nuclides.
3. Only the (likely stable) neutron-rich nuclides. There are 1875 such nuclides.
4. The entire set consisting of 1931 nuclides.

The coefficients a_α are quoted in MeV. For each run, we quote, in keV, the $\sigma_{n(n-1)}$ standard deviation of the difference $\Delta E = E_{AME03} - E_{SEMF}$ of binding energy between the experimental (AME03) and the SEMF-prediction. When required to quote "our result", it is only appropriate that we consider run number eight since it spans the entire subset and does not exclude the pairing-term. So, we have quoted the result of run 8 in Table.1.
The vast difference between the values a_P between LSF and our result in Table.1 can be understood if we compare the various forms of the pairing-term $\delta_P(A,Z)$ which are considered in the literature. We have come across three forms wherein $\delta_P(A,Z) = A^{-k}$ with $k \in \{1,\ 3/4,\ 1/2\}$. In Table.3 we compare all of the three forms by applying them on our subset (1931 nuclides) and the *entire* AME03 (3179 nuclides). It can be observed that the coefficient a_P changes dramatically between the three forms while the other coefficients and the errors do *not*. What was inadvertently quoted as our result in Table.1 is run numbered "8b". It should have been "8c" or, better, "8C".

Table 3: Various forms of the pairing-term.

Run No.	Samples	a_V	a_S	a_C	a_A	a_P	rms
				MeV			keV
$\delta_P(A,Z) = A^{-1}$							
8a	1931	15.5	17.0	0.697	22.4	38.4	78.5
8A	3179	15.2	16.2	0.687	22.2	25.9	70.5
$\delta_P(A,Z) = A^{-3/4}$							
8b	1931	15.5	17.0	0.697	22.4	23.5	77.8
8B	3179	15.2	16.2	0.687	22.2	19.6	70.0
$\delta_P(A,Z) = A^{-1/2}$							
8c	1931	15.4	17.0	0.697	22.4	11.0	77.3
8C	3179	15.2	16.2	0.687	22.2	10.1	69.5

5 Computerised calculations

The formalism lends itself excellently to parallel processing. The point to consider is the method by which G^{-1} is calculated. Section (*5.1*) contains discussion of how the numerical inverse can be calculated on SMP platforms. In section (*5.2*) we propose how one can use "parallel CAS" to employ the formalism of Theorem (2.1).

5.1 Numerical SMP

Arguably, the most common non-specialised technique used to numerically invert a matrix is Gauss-Jordan pivoted row-reduction to an echelon form followed by a "back substitution". A simple intuitive proof of this technique is given in [19] and we quote its statement:

Theorem 5.1. *If a sequence of elementary row operations will transform a square matrix M into the compatible identity matrix I, then this* same *sequence of elementary row operations will transform I into* M^{-1}.

There are two flavours of this technique and we reproduce the essential parts of the source-codes in sections (*5.1*.2) and (*5.1*.3). In the first flavour, we implement the algorithm for use on a *dual-core* workstation — one core does a row operation on *G* and the other does it concurrently on *I* — "one matrix, one CPU". In the second flavour, we implement the algorithm for use on a multi-core workstation with arbitrary many CPUs wherein one augments *G* with *I* and dispatches row-operations concurrently to the processors on the basis "one row, one CPU". Our implementation is in C++ and we have used OpenMP [20] [21].

5.1.*1* *Conventions and explanations*

Our basic mode of attack is *elementary row operations*. We have implemented a *Vector* with a *std::vector<double>* private container. The class *Matrix* contains a *vector<Vector*>* that holds the rows of a matrix as a "vector of vectors". The *numRows()* function is inlined and returns the number of rows of a Matrix. The *pivotIndex (int colnum, int sri)* member-function hunts down the most suitable pivot in column *colnum*, traversing downwards starting from row *sri*. The method *swapRows (int, int)* swaps the specified rows of the Matrix.
Most implementations of (higher-order) arrays provide access to individual elements using the indexing-style of either C/C++ or ForTran or both [22]. Instead, we have provisional interfaces *setElementAt (int r, int c, double datum)* and *getElementAt (int r, int c)* for accessing individual elements. This allows us to make significant changes in the underlying implementations of *Vector* or *Matrix* (iterators and OpenMP are mutually incompatible).
The method *tcs (int t, double c, int s)* is very special. It considers the row numbered *s*(ource) and adds its *c*(onstant) multiple to the row numbered *t*(arget). The method *divideRow (int r, double factor)* divides the r^{th} row by *factor*.

Given matrices G and I, the augmented matrix B is:

$$B = (G \mid I) \tag{5.58}$$

$$= \begin{pmatrix} g_{11} & g_{12} & \cdots & g_{1m} & 1 & 0 & \cdots & 0 \\ g_{21} & g_{22} & \cdots & g_{2m} & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \cdots & \vdots & \vdots & \vdots & \cdots & \vdots \\ g_{m1} & g_{m2} & \cdots & g_{mm} & 0 & 0 & \cdots & 1 \end{pmatrix}_{m\times 2m} \tag{5.59}$$

A Matrix instance can be augmented by some suitable Matrix by using the member interface *augment (Matrix)*. In our case, since we are interested in calculating the inverse, we augment by the identity matrix. Finally, the method *right_hand_half()* extracts out the $m\times m$ right-hand half of any such $m\times 2m$ "augmented" Matrix. In our case, once the processing has been done, the method returns the inverse matrix.
Though these functions and subroutines are functional, they can be *refined*. In the source-code reproduced here, we operate on *entire* rows. One can do better – trivially modify the member functions so that they take an extra parameter which tells them from *where* should they start operating in a given row.
For the current work, we used a dual-core AMD Athlon II X2 240 processor clocked at 2.8 GHz with 2 GB RAM and mounted on an Asus M2N68-AM-Plus mother-board. The system has 64K L1d cache, 64K L1i cache and 1024K L2 cache. The operating system used was the "amd64" port of Debian GNU/Linux 6.0 that offers *g++* version 4.4.5 and complies with the OpenMP standard 3.0 [23]. For symbolic manipulation, we used the Debian-default GiNaC V1.5.8 [24] [25].
We had a wish: comparing the two flavours of concurrency. However, since our computer has only *two* cores, both modes of concurrencies produced comparable timings. It would be interesting to test the concurrencies on a better platform.

5.1.2 *Dual-core*

```
Matrix*  Matrix::inverse_by_duality ()
{
    Matrix* I = Identity (numRows()); // Identity matrix.

    Matrix* RRE = new Matrix (*this); // Row-Reduced Echelon.

    int nrows = numRows();

    for (int i=0;   i < nrows;   i++)
        {
            int ipivot = RRE->pivotIndex (i, i);
            if (ipivot != i) //then and only then.
#pragma omp parallel sections
                {
#pragma omp section
                    RRE->swapRows (i, ipivot);
#pragma omp section
                    I->swapRows (i, ipivot);
                }

            double pivot = RRE->getElementAt (i,i);
            if (0 == pivot)
                continue;
            for (int j=i+1;   j < nrows;   j++)
                {
                    double mul = RRE->getElementAt(j, i);
```

```
                    mul /= pivot;
#pragma omp parallel sections
                    {
#pragma omp section
                        RRE->tcs (j, -mul, i);
#pragma omp section
                        I->tcs (j, -mul, i);
                    }

                }
#pragma omp parallel sections
            {
#pragma omp section
                RRE->divideRow (i, pivot);
#pragma omp section
                I->divideRow (i, pivot);
            }

        } //for (i=0 ; i<nrows)

      // Now, upwards elimination.

    for (int i=nrows-1;   i>=0;   i--)
        for (int j=i-1;   j>=0;   j--)
            {
                double mul = RRE->getElementAt (j, i);
#pragma omp parallel sections
                {
#pragma omp section
                    RRE->tcs (j, -mul, i);
#pragma omp section
                    I->tcs (j, -mul, i);
                }
            }

    return (I);
}
```

5.1.*3* *Multi-CPU*

```
Matrix*  Matrix::inverse_by_augmentation ()
{
    Matrix* I = Identity (numRows()); // Identity matrix.

    // Augmented matrix = "supplied + identity"
    Matrix* Aug = new Matrix (*this);
    Aug->augment (*I);

    int nrows = numRows();

    for (int i=0;   i < nrows;   i++)
        {
            int ipivot = Aug->pivotIndex (i, i);
```

```
            if (ipivot != i) //then and only then.
                {
                    Aug->swapRows (i, ipivot);
                }
            double pivot = Aug->getElementAt (i,i);
            if (0 == pivot)
                continue;

#pragma omp parallel for
            for (int j=i+1;   j < nrows;   j++)
                {
                    double mul = Aug->getElementAt(j, i);
                    mul /= pivot;
                    {
                        Aug->tcs (j, -mul, i);
                    }
                }
            {
                Aug->divideRow (i, pivot);
            }
        } //for (i=0 ; i<nrows)

    // Upwards elimination.

    for (int i=nrows-1;   i>=0;   i--)
        {
#pragma omp parallel for
            for (int j=i-1;   j>=0;   j--)
                {
                    double mul = Aug->getElementAt (j, i);
                    Aug->tcs (j, -mul, i);
                }
        }

    Matrix* Inv = Aug->right_hand_half ();

    return (Inv);
}
```

5.2 Parallel CAS

Deployed on various computing platforms, there is an overwhelming plethora of CAS tools, free as well as proprietary ones. Each software has its strengths and weaknesses. CAS have two main common drawbacks. Firstly, they differ in syntax and this renders them incompatible. Secondly, all are interactive though some do provide a limited batch-processing.

The biggest drawback with all the CAS tools is that they do *not* have provision for *parallel-processing*. This is a disappointment considering the amount of effort that goes into the research, development and deployment of parallel-processing technologies like LAM/MPI, OpenMP, POSIX threads, PVM and possibly more, targeting multi-core, cluster and grid architectures.

The only exceptional CAS, which gave us a pleasant surprise, is GiNaC [24] [25]. It is a free framework for symbolic computations within the C++ language. Though, per se, GiNaC does *not directly* support parallel-processing, one can write wrapper C++ code to achieve concurrency. Of the many concurrency frameworks available, we chose

OpenMP [20] [21]. As an introductory example, we quote the source-code which we used to implement equation (2.18).

```
#include <ginac/ginac.h>
#include <omp.h>

using namespace GiNaC;

matrix spinnrade ( matrix& F, matrix& Y )
{
    matrix Ft, G, Ginv, FtY;

    Ft = F.transpose();

#pragma omp parallel sections
    {
#pragma omp section
        {
            FtY = Ft.mul( Y);
        }
#pragma omp section
        {
            G  = Ft.mul( F);
            Ginv = G.inverse();
        }
    }

    matrix retval ( Ginv.mul( FtY);

    return retval;
}
```

We also have a *generic* driver `main()` that reads from the standard input, applies the formalism of Theorem (2.1) and outputs the fit-vector A to the standard output.

```
// Our "type"s and namespace.
#include "spinnrade_typedefs_ginac.hh"

#include <ginac/ginac.h>
#include <omp.h>

int main ()
{
    using namespace spinnrade_GiNaC; // A separate namespace.
    using namespace GiNaC;
    using namespace std;

    Liederkreis (); // Populates "f" with "f_\alpha"s.

    const int m = f.size();

    vector<x_data_t> x; // Store "X" objects.
    vector<y_data_t> y; // Store "Y" observations.
```

```
    x_data_t xd; // x datum.
    y_data_t yd; // y datum.
    string line; // Input one "object" (x_i or y_i) per new-line.
    GiNaC::parser reader; // Parses "line".

    while (cin)
        {
            try
                {
                    getline (cin, line);
                    xd = reader (line);
                    getline (cin, line);
                    yd = reader (line);
                }
            catch (exception& e)
                {
                    if (cin.eof())
                        break;
                }
            x.push_back (xd);
            y.push_back (yd);
        }

    //Assert (x.size()==y.size()).

    const int n = y.size();

    matrix F (n, m); // "Functions matrix".
    matrix Y (n, 1); // Observed "Y" column-vector.

    for (int r=0;   r<n;   r++)
        {
#pragma omp parallel for
            for (int alpha=0;   alpha<m;   alpha++)
                F(r, alpha) = f[alpha](x[r]);
            Y(r,0) = y[r];
        }

    matrix A = spinnrade (F,Y); // Fit-coefficients.

    // Optional symbolic manipulation of A here.

    cout << latex << A << endl << flush; // Output in LaTeX.

    exit (0); // Thread-safe. "return(0)" is not.
}
```

Tasks such as “How to do row-reduction of a matrix?”, “How to calculate the adjoint or determinant of a matrix?”, “How to calculate the inverse of a matrix?” ..., are done *innately* by GiNaC. For example, the `reduced_matrix()` method returns a minor. The `determinant()` method returns the determinant calculated by a specified algorithm, e.g., (pivoted) Gaussian elimination, Laplacian elimination (preserving intermediate expressions) and Bareiss elimi-

nation. The `solve()` method does row-elimination and arrives at the upper-triangular form.
We have made a simple deployment of parallel CAS for theorem(2.1) by introducing concurrency at the higher level, borrowing C++ code from the numerical section (*5.1*). We learned many things about GiNaC and C++ while doing so and conclude that mastering of the GiNaC framework is a worthy and fruitful endeavour.

References

[1] Alexander Hellemans, Bryan Bunch, The Timetables of Science, Simon & Schuster, ISBN 0-283-99926-8, (1988).

[2] E. Rutherford, The Scattering of α- and β-Particles by Matter and the Structure of the Atom, Phil. Mag, 21 (1911) 669-688.
http://dx.doi.org/10.1080/14786440508637080

[3] N. Bohr, On the Constitution of Atoms and Molecules, Phil. Mag, 26 (1) (1913).

[4] J. Chadwick, The Existence of a Neutron, Proc. Roy. Soc. (London), A136 (1932) 692-708.
http://dx.doi.org/10.1098/rspa.1932.0112

[5] Carl Friedrich von Weizsacker, On the theory of nuclear masses, Zeitschrift fur Physik, 96 (1935) 431.

[6] H. A. Bethe and R. F. Bacher, Nuclear Physics A. Stationary States of Nuclei, Rev. Mod. Phys, 8 (1936) 82.

[7] N. Bohr and J. A. Wheeler, The Mechanism of Nuclear Fission, Phys. Rev, 56 (1939) 426.
http://dx.doi.org/10.1103/PhysRev.56.426

[8] E. Feenberg, Semi-empirical Theory of the Nuclear Energy Surface, Revs. Mod. Phys, 19 (1947) 239258.
http://dx.doi.org/10.1103/RevModPhys.19.239

[9] A. E. S. Green, N. A. Engler, Mass Surfaces, Phys. Rev, 91 (1953) 40.
http://dx.doi.org/10.1103/PhysRev.91.40

[10] A. E. S. Green, D. F. Edwards, Discontinuities in the Nuclear Mass Surface, Phys. Rev, 91 (1953) 46.
http://dx.doi.org/10.1103/PhysRev.91.46

[11] N. Feather, Nuclear Stability Rules, Cambridge University Press, (1952).

[12] Irving Kaplan, Nuclear Physics, Addison-Wesley India, ISBN 81-85015-89-9, (1962).

[13] T. Duguet, P. Bonche, P. H. Heenen and J. Meyer, Pairing correlations II: Microscopic analysis of odd-even mass staggering in nuclei, Phys. Rev, 014311, C 65 (2001) 1-14.
http://dx.doi.org/10.1103/PhysRevC.65.014311

[14] www.en.wikipedia.org/wiki/Semi_empirical_mass_formula.
http://en.wikipedia.org/wiki/Semi_empirical_mass_formula

[15] A. H. Wapstra, Atomic Masses of Nuclides, in External Properties of Atomic Nuclei, ed. S. Flugge, part of series Encyclopedia of Physics,ISBN 978-3-540-02296-1, online ISBN 978-3-642-45901-6, 8/38/1 (1958) 1-37.

[16] James William Rohlf, Modern Physics from α to Z^0, Wiley, ISBN 978-0-471-57270-1, (1994).

[17] G. Audi, A. H.Wapstra and C. Thibault, The Ame2003 mass evaluation (II), Nucl. Phys, A729 (2003) 22.

[18] www.amdc.in2p3.fr/web/masseval.html.
http://amdc.in2p3.fr/web/masseval.html

[19] Ralph G. Stanton, Numerical Methods for Science and Engineering, Prentice-Hall of India Pvt. Ltd, ISBN-0-87692-306-6, (1985).

[20] Barbara Chapman, Gabrielle Jost and Rudd van der Pas, Using OpenMP, MIT Press, ISBN-10: 0-262-53302-2, ISBN-13: 978-0-262-53302-7, (2008).

[21] www.openmp.org.
http://openmp.org

[22] Bjarne Stroustrup, The C++ Programming Language, Pearson Education, ISBN 81-7808-126-1, (2000).

[23] www.openmp.org/wp/openmp-specifications.
http://openmp.org/wp/openmp-specifications

[24] Christian Bauer, Alexander Frink and Richard Kreckel, Introduction of the GiNaC Framework for Symbolic Computations within the C++ Programming Language, J. Symbolic Computation, 33 (2002) 1-12.
http://dx.doi.org/10.1006/jsco.2001.0494

[25] www.ginac.de.
http://www.ginac.de