Conformable Fractional Models of the Stellar Helium Burning via Artificial Neural Networks

Emad A.-B. Abdel-Salam[1], Mohamed I. Nouh[2] and Yosry A. Azzam[2]

[1]Department of Mathematics, Faculty of Science, New Valley University, El-Kharja 72511, Egypt
[2]Astronomy Department, National Research Institute of Astronomy and Geophysics (NRIAG), 11421 Helwan, Cairo, Egypt

e-mail: mohamed.nouh@nriag.sci.eg

Abstract: The helium burning phase represents the second stage that the star used to consume nuclear fuel in its interior. In this stage, the three elements carbon, oxygen, and neon are synthesized. The present paper has two folds, the first is to develop an analytical solution to the system of the conformable fractional differential equations of the helium burning network, where we used for this purpose the series expansion method and obtained recurrence relations for the product abundances i.e. helium, carbon, oxygen, and neon. Using four different initial abundances, we calculated 44 gas models covering the range of the fractional parameter $\alpha = 0.5 - 1$ with step $\Delta\alpha = 0.05$. We found that the effects of the fractional parameter on the product abundances are small which coincides with the results obtained by a previous study. Second, we introduced the mathematical model of the neural network (NN) and developed a neural network algorithm to simulate the helium burning network using its feed-forward model that is trained by the back propagation (BP) gradient descent delta rule algorithm. A comparison between the NN and the analytical models revealed very good agreement for all gas models. We found that NN could be considered as a powerful tool to solve and model nuclear burning networks and could be applied to the other nuclear stellar burning networks.

Keywords: Conformable fractional derivative; Artificial neural network; Series expansion method; Stellar models: Helium burning phase

1. Introduction

Nowadays, applications of fractional calculus in physics, astrophysics, and related science are widely used. In astrophysics, many problems have been handled using fractional

models, examples of these studies are: El-Nabulsi [1] introduced an analytical solution to the fractional white dwarf equation; Bayin and Krisch [2] analyzed the fractional incompressible gas spheres; Abdel-Salam and Nouh [3] and Nouh and Abdel-Salam [4] introduced an analytical solution to the first and second types of Lane-Emden equation in the sense of modified Riemann-Liouville fractional derivative. Nouh [5] solved the fractional helium burning network using a series expansion method. Abdel-Salam and Nouh [6] and Yousif et al. [7] introduced analytical solutions
to the conformable polytropic and isothermal gas spheres.

Simulation of ordinary (ODE) and partial differential (PDE) equations using an artificial neural network (ANN) give very good accuracy when compared with both the numerical and analytical methods. Many authors deal with this issue and developed many neural algorithms to solve ODE and PDE. Dissanayake and Phan-Thien [8] first introduced the concept of approximating the solutions of differential equations with neural networks, where training was carried out by minimizing losses based on the satisfaction of the network with the boundary conditions and the differential equations themselves. Lagaris et al. [9] demonstrated that the network shape could be selected by construction to satisfy boundary conditions and that automatic differentiation could be used to determine the derivatives that appear in the loss function. This approach has been extended to irregular boundary systems [10,11], applied to the resolution of PDEs occurring in fluid mechanics [12], and software packages have been developed to facilitate their application [13-15]. Nouh et al. [16] and Azzam et al. [17] developed a neural network algorithm to solve the first and second types of Lane-Emden equations arising in astrophysics.

The helium burning stage (also known as the triple-alpha process) represents the second stage where the stars undergo transfer of nuclear energy from the interior to their surface. In this stage, nuclear energy is almost converted to light when passing through the stellar atmosphere. Helium burning (HB) releases energy per unit fuel of about 6×10^{23} MeV/g $\approx 10^{18}$ erg/g. The reaction equations govern HB network may be written as [5]

$$3He^4 \rightarrow C^{12} + \gamma + 7.281 \text{ Mev}$$
$$C^{12} + He^4 \rightarrow O^{16} + \gamma + 7.150 \text{ Mev}$$
$$O^{16} + He^4 \rightarrow Ne^{20} + \gamma + 4.750 \text{ Mev}$$

where the conversion process from helium to carbon needs 10^8 K^o.

Clayton [18] set up a model for the helium burning process by taking into account the above reactions. If the number of atoms per unit of stellar material mass for helium, carbon, oxygen, and neon is represented by x, y, z, and r respectively, then the next four equations (also may be called the kinetic equations) control the time-dependent change in abundance:

$$\begin{aligned} dx/dt &= -3ax^3 - bxy - cxz, \\ dy/dt &= ax^3 - bxy, \\ dz/dt &= bxy - cxz, \\ dr/dt &= cxz. \end{aligned} \tag{1}$$

where a, b, and c are the reaction rates.

The system of equations (1) represents the integer version of the helium burning network and solved simultaneously by computational or analytical methods [18-21]. The fractional kinetic equation (like helium burning network) has been solved by many authors. In terms of H-functions, [22] presented a solution to the fractional generalized kinetic equation. The generalized fractional kinetic equations have been solved by [23]. Chaurasia and Pandey [24] solved the fractional kinetic equations in a series form of the Lorenzo-Hartley function.

In the present article, we developed a neural network algorithm to solve the fractional system of differential equations describing the helium burning network. We use the principles of the conformable fractional derivative for the mathematical modeling of the ANN. We used in this research an architecture of ANN which is the feed-forward network having three-layers and trained using the algorithm of back-propagation (BP) based on the gradient descent delta rule.

The analytical solution is developed using the series expansion method and a comparison between the ANN and analytical models is performed to declare the efficiency and applicability of the ANN for solving the conformable helium burning network. The paper is organized as follows: section 2 introduces the details of the analytical solution of the conformable helium burning model using the series expansion method. Section 3 deals with the mathematical modeling of the neural network technique with its gradient computations and back propagation training algorithm. Section 4 discuss the results obtained and the comparison between the NN and analytical models. Section 5 gives the details of the conclusion.

2. Analytical Solution to the Conformable Helium Burning Model

The fractional form of Equation (1) is given by [5]

$$
\begin{aligned}
D_t^{\alpha} x &= -3a\,x^3 - b\,x\,y - c\,x\,z,\\
D_t^{\alpha} y &= a\,x^3 - b\,x\,y,\\
D_t^{\alpha} z &= b\,x\,y - c\,x\,z,\\
D_t^{\alpha} r &= c\,x\,z,
\end{aligned}
\tag{8}
$$

If $T = t^{\alpha}$, then *x, y, z, r* could be represented by

$$
\begin{aligned}
x = \sum_{m=0}^{\infty} X_m T^m &= X_0 + X_1 T + X_2 T^2 + X_3 T^3 + \ldots\\
&= X_0 + X_1 t^{\alpha} + X_2 t^{2\alpha} + X_3 t^{3\alpha} + \ldots,\\
y = \sum_{m=0}^{\infty} Y_m T^m &= Y_0 + Y_1 T + Y_2 T^2 + Y_3 T^3 + \ldots\\
&= Y_0 + Y_1 t^{\alpha} + Y_2 t^{2\alpha} + Y_3 t^{3\alpha} + \ldots,\\
z = \sum_{m=0}^{\infty} Z_m T^m &= Z_0 + Z_1 T + Z_2 T^2 + Z_3 T^3 + \ldots\\
&= Z_0 + Z_1 t^{\alpha} + Z_2 t^{2\alpha} + Z_3 t^{3\alpha} + \ldots,\\
r = \sum_{m=0}^{\infty} R_m T^m &= R_0 + R_1 T + R_2 T^2 + R_3 T^3 + \ldots\\
&= R_0 + R_1 t^{\alpha} + R_2 t^{2\alpha} + R_3 t^{3\alpha} + \ldots .
\end{aligned}
\tag{9}
$$

where $X_m,\ Y_m,\ Z_m,\ R_m$ are constants to be determined.

Using the series expansion method, we obtained the recurrence relation of the term x^3 by the following

Let $D_x^{\alpha} u^n = D_x^{\alpha} G$, with $u = \sum_{m=0}^{\infty} A_m x^{\alpha m},\ C = \sum_{m=0}^{\infty} Q_m x^{\alpha m}$

that is

$$n\,u^{n-1} D_x^{\alpha} u = D_x^{\alpha} G .$$

or

$$n\,u^{n}D_x^{\alpha}u = u\,D_x^{\alpha}G \quad . \tag{10}$$

Performing the fractional derivative to Equation (10) k timed we get

$$D^{\underbrace{\alpha\ \ldots\alpha}_{k\ \text{times}}}[nG\,D_x^{\alpha}u] = D^{\underbrace{\alpha\ \ldots\alpha}_{k\ \text{times}}}(u\,D_x^{\alpha}G),$$

or

$$n\sum_{j=0}^{k}\binom{k}{j}D^{\underbrace{\alpha\ \ldots\alpha}_{j+1\ \text{times}}}u\,D^{\underbrace{\alpha\ \ldots\alpha}_{k-j\ \text{times}}}G = \sum_{j=0}^{k}\binom{k}{j}D^{\underbrace{\alpha\ \ldots\alpha}_{j+1\ \text{times}}}G\,D^{\underbrace{\alpha\ \ldots\alpha}_{k-j\ \text{times}}}u,$$

put $X = 0$, we get

$$n\sum_{j=0}^{k}\binom{k}{j}D^{\underbrace{\alpha\ \ldots\alpha}_{j+1\ \text{times}}}u(0)D^{\underbrace{\alpha\ \ldots\alpha}_{k-j\ \text{times}}}G(0) = \sum_{j=0}^{k}\binom{k}{j}D^{\underbrace{\alpha\ \ldots\alpha}_{j+1\ \text{times}}}G(0)\,D^{\underbrace{\alpha\ \ldots\alpha}_{k-j\ \text{times}}}u(0),$$

Since $D^{\underbrace{\alpha\ \ldots\alpha}_{j+1\ \text{times}}}u(0) = \alpha^{(j+1)}A_{j+1}$, $D^{\underbrace{\alpha\ \ldots\alpha}_{k-j\ \text{times}}}G(0) = \alpha^{(k-j)}Q_{k-j}$, $D^{\underbrace{\alpha\ \ldots\alpha}_{j+1\ \text{times}}}G(0) = \alpha^{(j+1)}Q_{j+1}$,

$D^{\underbrace{\alpha\ \ldots\alpha}_{k-j\ \text{times}}}u(0) = \alpha^{(k-j)}Q_{j+1}$, so we have

$$n\sum_{j=0}^{k}\binom{k}{j}(j+1)!\alpha^{(j+1)}A_{j+1}(k-j)!\alpha^{(k-j)}Q_{k-j} = \sum_{j=0}^{k}\binom{k}{j}(j+1)!\alpha^{(j+1)}Q_{j+1}(k-j)!\alpha^{(k-j)}A_{k-j},\Rightarrow$$

$$n\sum_{j=0}^{k}\binom{k}{j}(j+1)!(k-j)!\alpha^{(k+1)}A_{j+1}Q_{k-j} = \sum_{j=0}^{k}\binom{k}{j}(j+1)!(k-j)!\alpha^{(k+1)}Q_{j+1}A_{k-j},$$

After some manipulations we get

$$k!(k+1)A_0Q_{k+1} = n\sum_{j=0}^{k}k!(j+1)A_{j+1}Q_{k-j} - \sum_{j=0}^{k-1}k!(j+1)A_{k-j}Q_{j+1},$$

(11)

put $i = j + 1$ and $i = k - j$ in Equation (11) we have

$$k!(k+1)A_0Q_{k+1} = n\sum_{i=1}^{k+1}k!(i)A_iQ_{k+1-i} - \sum_{i=1}^{k}k!(k+1-i)A_iQ_{k+1-i},$$

If $m = k + 1$, then

$$m!A_0Q_m = n\sum_{i=1}^{m}(m-1)!(i)A_iQ_{m-i} - \sum_{i=1}^{m-1}(m-1)!(m-i)A_iQ_{m-i},$$

Add the zero value $\{-(m-1)!(m-m)A_mQ_0\}$ to the second summation of the last equation, we get

$$m!A_0Q_m = n\sum_{i=1}^{m}(m-1)!(i)A_iQ_{m-i} - \sum_{i=1}^{m-1}(m-1)!(m-i)A_iQ_{m-i} - (m-1)!(m-m)A_mQ_0,$$

$$m!A_0Q_m = n\sum_{i=1}^{m}(m-1)!(i)A_iQ_{m-i} - \sum_{i=1}^{m}(m-1)!(m-i)A_iQ_{m-i},$$

From the last equation, we can write the coefficients Q_m as

$$Q_m = \frac{1}{m!A_0}\sum_{i=1}^{m}(m-1)!(in-m+i)A_iQ_{m-i}, \quad \forall\ m \quad 1, \tag{12}$$

Put $n = 3$ Equation (12), we have

$$Q_m = \frac{1}{m!X_0}\sum_{i=1}^{m}(m-1)!(4i-m)X_iQ_{m-i}, \quad \forall\ m \quad 1, \tag{13}$$

where $X_0 = A_0$ and $X_i = A_i$

and $Q_0 = X_0^3,\ Q_1 = \frac{3X_1Q_0}{X_0}, \ldots.$

taking fractional differentiation α -derivatives to Equation (9) we get

$$D_t^{\alpha}x = \sum_{n=1}^{\infty}\alpha\, n X_n T^{n-1},$$

$$D_t^{\alpha}y = \sum_{n=1}^{\infty}\alpha\, n Y_n T^{n-1},$$

$$D_t^{\alpha}z = \sum_{n=1}^{\infty}\alpha\, n Z_n T^{n-1},$$

$$D_t^{\alpha}r = \sum_{n=1}^{\infty}\alpha\, n R_n T^{n-1}, \tag{14}$$

inserting Equations (9) and (14) into Equation (8), the series coefficients $X_{n+1},\ Y_{n+1},\ Z_{n+1}$ and R_{n+1} could be obtained from

$$X_{n+1} = -\frac{1}{\alpha(n+1)}\left[3aQ_n + b\sum_{j=0}^{n} X_j Y_{n-j} + c\sum_{j=0}^{n} X_j Z_{n-j}\right],$$

$$Y_{n+1} = \frac{1}{\alpha(n+1)}\left[aQ_n - b\sum_{j=0}^{n} X_j Y_{n-j}\right],$$

(15)

$$Z_{n+1} = \frac{1}{\alpha(n+1)}\left[b\sum_{j=0}^{n} X_j Y_{n-j} - c\sum_{j=0}^{n} X_j Z_{n-j}\right],$$

and

$$R_{n+1} = \frac{c}{\alpha(n+1)}\sum_{j=0}^{n} X_j Z_{n-j}.$$

The recurrence relations corresponding to the integer model could be obtained by putting $\alpha = 1$ in the last four equations [21].

At *n=0* and with the initial values of the chemical composition, $X_0 = x_0; Y_0 = y_0;\ Z_0 = z_0;\ R_0 = r_0,$ where x_0, y_0, z_0, r_0 are arbitrary constants, we get

$$X_1 = -\frac{1}{\alpha}\left[3aQ_0 + bX_0Y_0 + cX_0Z_0\right] = -\frac{1}{\alpha}\left[3a\,x_0^3 + bx_0y_0 + cx_0y_0\right],$$

$$Y_1 = \frac{1}{\alpha}\left[aQ_0 - bX_0Y_0\right] = \frac{1}{\alpha}\left[a\,x_0^3 - bx_0y_0\right],$$

$$Z_1 = \frac{1}{\alpha}\left[bX_0Y_0 - cX_0Z_0\right] = \frac{1}{\alpha}\left[bx_0y_0 - cx_0z_0\right],$$

$$R_1 = \frac{c}{\alpha}X_0Z_0 = \frac{c}{\alpha}x_0z_0,$$

and

$$Q_0 = X_0^3 = x_0^3,\ \ Q_1 = \frac{3X_1Q_0}{X_0} = -\frac{3x_0^2}{\alpha}\left[3a\,x_0^3 + bx_0y_0 + cx_0y_0\right].$$

at n=1 we get

$$X_2 = -\frac{1}{2\alpha}\left[3aQ_1 + b(X_0Y_1 + X_1Y_0) + c(X_0Z_1 + X_1Z_0)\right]$$

$$= -\frac{1}{2\alpha}\left[\begin{array}{l} -\frac{9ax_0^2}{\alpha}\left[3a\,x_0^3 + bx_0y_0 + cx_0y_0\right] + b\left(\frac{x_0}{\alpha}\left[a\,x_0^3 - bx_0y_0\right] - \frac{y_0}{\alpha}\left[3a\,x_0^3 + bx_0y_0 + cx_0y_0\right]\right) \\ +c\left(\frac{x_0}{\alpha}\left[bx_0y_0 - cx_0z_0\right] - \frac{z_0}{\alpha}\left[3a\,x_0^3 + bx_0y_0 + cx_0y_0\right]\right) \end{array}\right]$$

$$Y_2 = \frac{1}{2\alpha}\left[aQ_1 - b(X_0Y_1 + X_1Y_0)\right]$$

$$= \frac{1}{2\alpha}\left[-\frac{9ax_0^2}{\alpha}\left[3a\,x_0^3 + bx_0y_0 + cx_0y_0\right] + b\left(\frac{x_0}{\alpha}\left[a\,x_0^3 - bx_0y_0\right] - \frac{y_0}{\alpha}\left[3a\,x_0^3 + bx_0y_0 + cx_0y_0\right]\right)\right],$$

$$Z_2 = \frac{1}{2\alpha}\left[b(X_0Y_1 + X_1Y_0) - c(X_0Z_1 + X_1Z_0)\right]$$

$$= \frac{1}{2\alpha}\left[b\left(\frac{x_0}{\alpha}\left[a\,x_0^3 - bx_0y_0\right] - \frac{y_0}{\alpha}\left[3a\,x_0^3 + bx_0y_0 + cx_0y_0\right]\right) - c\left(\frac{x_0}{\alpha}\left[bx_0y_0 - cx_0z_0\right] - \frac{z_0}{\alpha}\left[3a\,x_0^3 + bx_0y_0 + cx_0y_0\right]\right)\right],$$

$$R_2 = \frac{c}{2\alpha}(X_0Z_1 + X_1Z_0) = \frac{c}{2\alpha}\left(\frac{x_0}{\alpha}\left[bx_0y_0 - cx_0z_0\right] - \frac{z_0}{\alpha}\left[3a\,x_0^3 + bx_0y_0 + cx_0y_0\right]\right),$$

And by applying the same scheme we can determine the rest of the series terms. So, the series solution of Equation (8) is written as:

$$x = \sum_{m=0}^{\infty} X_m t^{\alpha m} = X_0 + X_1 t^{\alpha} + X_2 t^{2\alpha} + X_3 t^{3\alpha} + \ldots$$

$$= x_0 - \frac{1}{\alpha}\left[3a\,x_0^3 + bx_0y_0 + cx_0y_0\right]t^{\alpha}$$

$$-\frac{1}{2\alpha}\left[\begin{array}{l} -\frac{9ax_0^2}{\alpha}\left[3a\,x_0^3 + bx_0y_0 + cx_0y_0\right] + b\left(\frac{x_0}{\alpha}\left[a\,x_0^3 - bx_0y_0\right] - \frac{y_0}{\alpha}\left[3a\,x_0^3 + bx_0y_0 + cx_0y_0\right]\right) \\ +c\left(\frac{x_0}{\alpha}\left[bx_0y_0 - cx_0z_0\right] - \frac{z_0}{\alpha}\left[3a\,x_0^3 + bx_0y_0 + cx_0y_0\right]\right) \end{array}\right]t^{2\alpha} + \ldots,$$

$$y = \sum_{m=0}^{\infty} Y_m t^{\alpha m} = Y_0 + Y_1 t^{\alpha} + Y_2 t^{2\alpha} + Y_3 t^{3\alpha} + \ldots$$

$$= y_0 + \frac{1}{\alpha}\left[a\,x_0^3 - bx_0y_0\right]t^{\alpha}$$

$$+\frac{1}{2\alpha}\left[-\frac{9ax_0^2}{\alpha}\left[3a\,x_0^3 + bx_0y_0 + cx_0y_0\right] + b\left(\frac{x_0}{\alpha}\left[a\,x_0^3 - bx_0y_0\right] - \frac{y_0}{\alpha}\left[3a\,x_0^3 + bx_0y_0 + cx_0y_0\right]\right)\right]t^{2\alpha} + \ldots,$$

$$z = \sum_{m=0}^{\infty} Z_m t^{\alpha m} = Z_0 + Z_1 t^{\alpha} + Z_2 t^{2\alpha} + Z_3 t^{3\alpha} + \ldots$$

$$= z_0 + \frac{1}{\alpha}\left[bx_0 y_0 - cx_0 z_0 \right] t^{\alpha}$$

$$+ \frac{1}{2\alpha}\begin{bmatrix} b\left(\frac{x_0}{\alpha}\left[a x_0^3 - bx_0 y_0 \right] - \frac{y_0}{\alpha}\left[3a x_0^3 + bx_0 y_0 + cx_0 y_0 \right] \right) \\ -c\left(\frac{x_0}{\alpha}\left[bx_0 y_0 - cx_0 z_0 \right] - \frac{z_0}{\alpha}\left[3a x_0^3 + bx_0 y_0 + cx_0 y_0 \right] \right) \end{bmatrix} t^{2\alpha} + \ldots,$$

$$r = \sum_{m=0}^{\infty} R_m t^{\alpha m} = R_0 + R_1 t^{\alpha} + R_2 t^{2\alpha} + R_3 t^{3\alpha} + \ldots$$

$$= r_0 + \frac{c}{\alpha} x_0 z_0 t^{\alpha} + \frac{c}{2\alpha}\left(\frac{x_0}{\alpha}\left[bx_0 y_0 - cx_0 z_0 \right] - \frac{z_0}{\alpha}\left[3a x_0^3 + bx_0 y_0 + cx_0 y_0 \right] \right) t^{2\alpha} + \ldots .$$

3. Neural network algorithm

3.1 Mathematical modeling of the problem

To simulate the conformable fractional helium burning network represented by Equation (8), we use the neural network architecture shown in Figure 1.

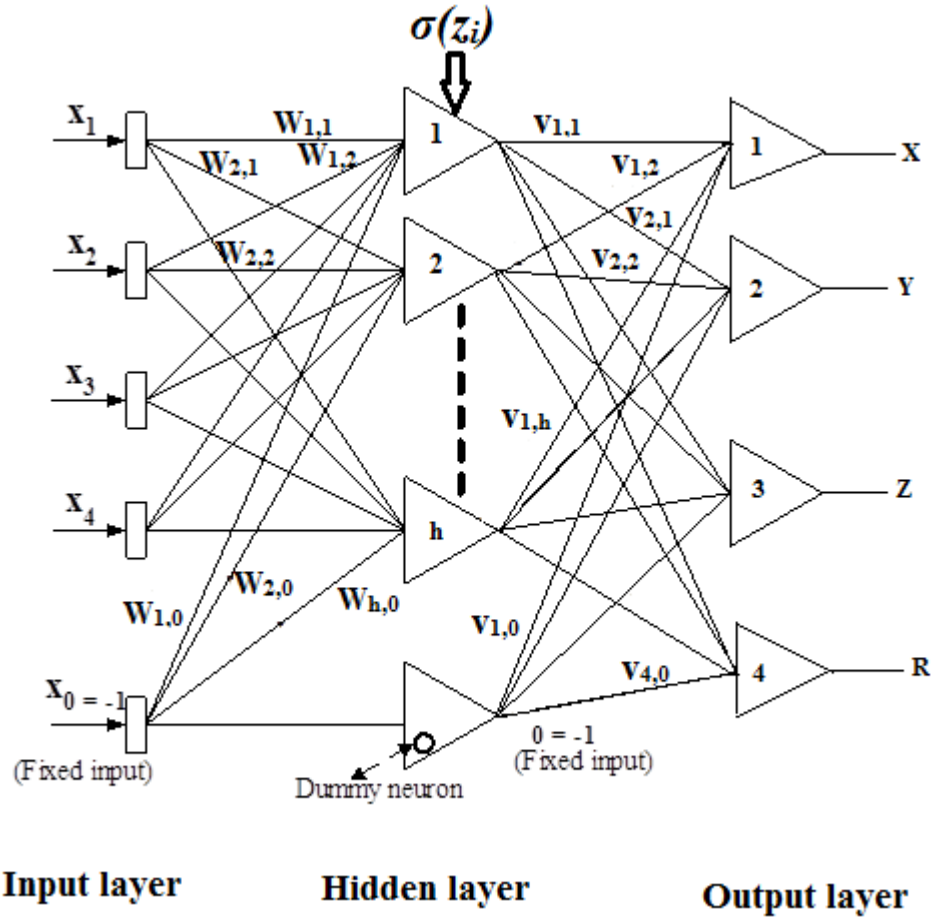


Fig. 1. ANN architecture proposed to simulate conformable fractional helium burning network

Considering the initial conditions $X_0 = x_0; Y_0 = y_0;\ Z_0 = z_0;\ R_0 = r_0$, the neural network could be obtained following the next steps [25]:

The form of the neural approximate solution of Equation (8) will have two terms, the first represents the initial values and the second represents the feed-forward neural network, where *x* is the input vector and *p* is the corresponding vector of adjustable weight parameters. Then the output of the neural network $N_I(x_I, p)$ is written as

$$X_t(x,p) = A_1(x,) + f_1(x, N_1(x,p)),$$
$$Y_t(y,p) = A_2(y) + f_2(y, N_2(y,p)),$$
$$Z_t(z,p) = A_3(z) + f_3(z, N_3(z,p)),$$
$$R_t(r,p) = A_4(r) + f_4(r, N_4(r,p)),$$

(16)

Then neural network output $N_\ell(x_\ell, p)$ is given by

$$N_\ell(x_\ell, p) = \sum_{j=1}^{H} v_{ij}\sigma_i(z_j), \qquad \ell = 1,2,3,4 \tag{17}$$

where $z_j = \sum_{i=1}^{n} w_{ij}x_j + \beta_i$ and w_{ij} is the weight from the input unit *j* to the hidden unit *i*, v_i is the weight from the hidden unit *i* to the output, β_i represents the bias of the *i*th hidden unit and $\sigma(z)$ is the sigmoid activation function that has the form $\sigma(x) = \frac{1}{1+e^{-x}}, \sigma(y) = \frac{1}{1+e^{-y}},$ $\sigma(z) = \frac{1}{1+e^{-z}},$ and $\sigma(r) = \frac{1}{1+e^{-r}},$

by differentiating the networks output *N* with respect to the vector x_j, we get:

$$D^{\alpha}_{x_j} N(x,p) = D^{\alpha}_{x_j}\left(\sum_{i=1}^{H} v_i\sigma\left(z_i = \sum_{i=1}^{n} w_{ij}x_j + \beta_i\right)\right) = \sum_{i=1}^{h} v_i w_{ij}\sigma^{(\alpha)}, \qquad \sigma^{(\alpha)} = D^{\alpha}_{x}\sigma(x), \tag{18}$$

Differentiate Equation (17) *n* times gives

$$D^{\overbrace{\alpha \ldots \alpha}^{n \text{ times}}}_{x_j} N(x,p) = \sum_{i=1}^{n} v_i P_i \sigma_i^{(n\alpha)}, \qquad , P_i = \prod_{k=1}^{n} w_{ik}^{\alpha_k}, \quad \sigma_i = \sigma(z_i), \tag{19}$$

As a result, the solution of the helium burning network is given as

$$X_t(x,p) = x_0 + x N_1(x,p),$$
$$Y_t(x,p) = y_0 + y N_2(y,p),$$
$$Z_t(x,p) = z_0 + z N_3(z,p),$$
$$R_t(x,p) = r_0 + r N_4(r,p), \tag{20}$$

which fulfills the initial conditions as:

$$X_t(0,p) = x_0 + 0.N_1(0,p) = x_0, \quad Y_t(0,p) = y_0 + 0.N_2(0,p) = y_0,$$
$$Z_t(0,p) = z_0 + 0.N_3(0,p) = z_0, \quad R_t(0,p) = r_0 + 0.N_4(0,p) = r_0,$$

(21)

and

$$\begin{aligned} D_x^\alpha X_t(x,p) &= x^{1-\alpha} N_1(x,p) + x D_x^\alpha N_1(x,p), \\ D_x^\alpha Y_t(y,p) &= y^{1-\alpha} N_2(y,p) + y D_x^\alpha N_2(y,p), \\ D_x^\alpha Z_t(z,p) &= z^{1-\alpha} N_3(z,p) + z D_z^\alpha N_3(z,p), \\ D_r^\alpha R_t(r,p) &= r^{1-\alpha} N_4(r,p) + r D_r^\alpha N_4(r,p), \end{aligned} \quad , \tag{22}$$

3.2 Gradient Computations and Parameter Updating

Using Equation (20) to update the network parameters and computing the gradient, the error quantity needed to be minimized is given by

$$\begin{aligned} E(x) = \sum_t \Big\{ D_t^\alpha X_t(x_t,p) + 3aX_t^3(x_t,p) + bX_t(x_t,p)Y_t(y_t,p) + cX_t(x_t,p)Z_t(z_t,p) \Big\}^2 \\ + \sum_t \Big\{ D_t^\alpha Y_t(x_t,p) - aX_t^3(x_t,p) + bX_t(x_t,p)Y_t(y_t,p) \Big\}^2 + \sum_t \Big\{ D_t^\alpha R_t(r_t,p) - cX_t(x_t,p)Z_t(z_t,p) \Big\}^2 \\ + \sum_t \Big\{ D_t^\alpha Z_t(z_t,p) - bX_t(x_t,p)Y_t(y_t,p) + cX_t(x_t,p)Z_t(z_t,p) \Big\}^2, \end{aligned} \tag{23}$$

where

$$\begin{aligned} D_x^\alpha X_t(x,p) &= x^{1-\alpha} N_1(x,p) + x D_x^\alpha N_1(x,p), \\ D_x^\alpha Y_t(y,p) &= y^{1-\alpha} N_2(y,p) + y D_x^\alpha N_2(y,p), \\ D_x^\alpha Z_t(z,p) &= z^{1-\alpha} N_3(z,p) + z D_z^\alpha N_3(z,p), \\ D_r^\alpha R_t(r,p) &= r^{1-\alpha} N_4(r,p) + r D_r^\alpha N_4(r,p), \end{aligned} \tag{24}$$

where $D_x^\alpha N(x,p)$ is given by Equation (18). So, the problem is converted to an unconstrained optimization problem.

To update the network parameters, we train the neural network for the optimized parameter values. After the training process, we obtained the network parameters and computed the following

$$X_i(x,p) = x_0 + x\,N_1(x,p),$$
$$Y_i(x,p) = y_0 + y\,N_2(y,p),$$
$$Z_i(x,p) = z_0 + z\,N_3(z,p),$$
$$R_i(x,p) = r_0 + r\,N_4(r,p).$$

Now, N with one hidden layer is analogous to the conformable fractional derivative. By replacing the hidden unit transfer function with the n[th] order fractional derivative, the fractional N gradient differentiating with respect to v_i, β_i and w_{ij} could be written as

$$D^{\alpha}_{v_i} N = P_i\ \sigma_i^{(n\alpha)}$$
$$D^{\alpha}_{\beta_i} N = v_i P_i\ \sigma_i^{((n+1)\alpha)}$$
$$D^{\alpha}_{w_{ij}} N = x_i v_i P_i\ \sigma_i^{((n+1)\alpha)} + v_i \alpha_j w_{ij}^{1-\alpha_j} \left(\prod_{k=1, k\neq j} w_{ik}^{\alpha_k} \right) \sigma_i^{(n\alpha)} . \quad (25)$$

The network parameters updating rule can be given as,

$$v_i(x+1) = v_i(x) + a\,D^{\alpha}_{v_i} N , \quad (26)$$

$$\beta_i(x+1) = \beta_i(x) + b\,D^{\alpha}_{\beta_i} N , \quad (27)$$

$$w_{ij}(x+1) = w_{ij}(x) + c\,D^{\alpha}_{w_{ij}} N , \quad (28)$$

where a, b, c are learning rates, $i = 1, 2, \ldots, n,$ and $j = 1, 2, \ldots, h$.

In the stellar helium burning model based on ANN, the neuron is the fundamental processing unit that can process a local memory and carry out localised information. At each neuron, the net input (z) is calculated by supplementing the received weights to obtain an aggregate weight of those inputs and add it to a bias (β). The net input (z) is then passed by a nonlinear activation function, which results in the neuron output u_j (as is seen in Figure 1) [26].

3.3 Training of BP Algorithm

The back-propagation (BP) training algorithm is a gradient algorithm aimed to minimize the average square error between the desired output and the actual output of a feed-forward network.

It requires continuously differentiable non-linearity. Fig 2 displays a flow chart of a back-propagation off-line learning algorithm [27]:

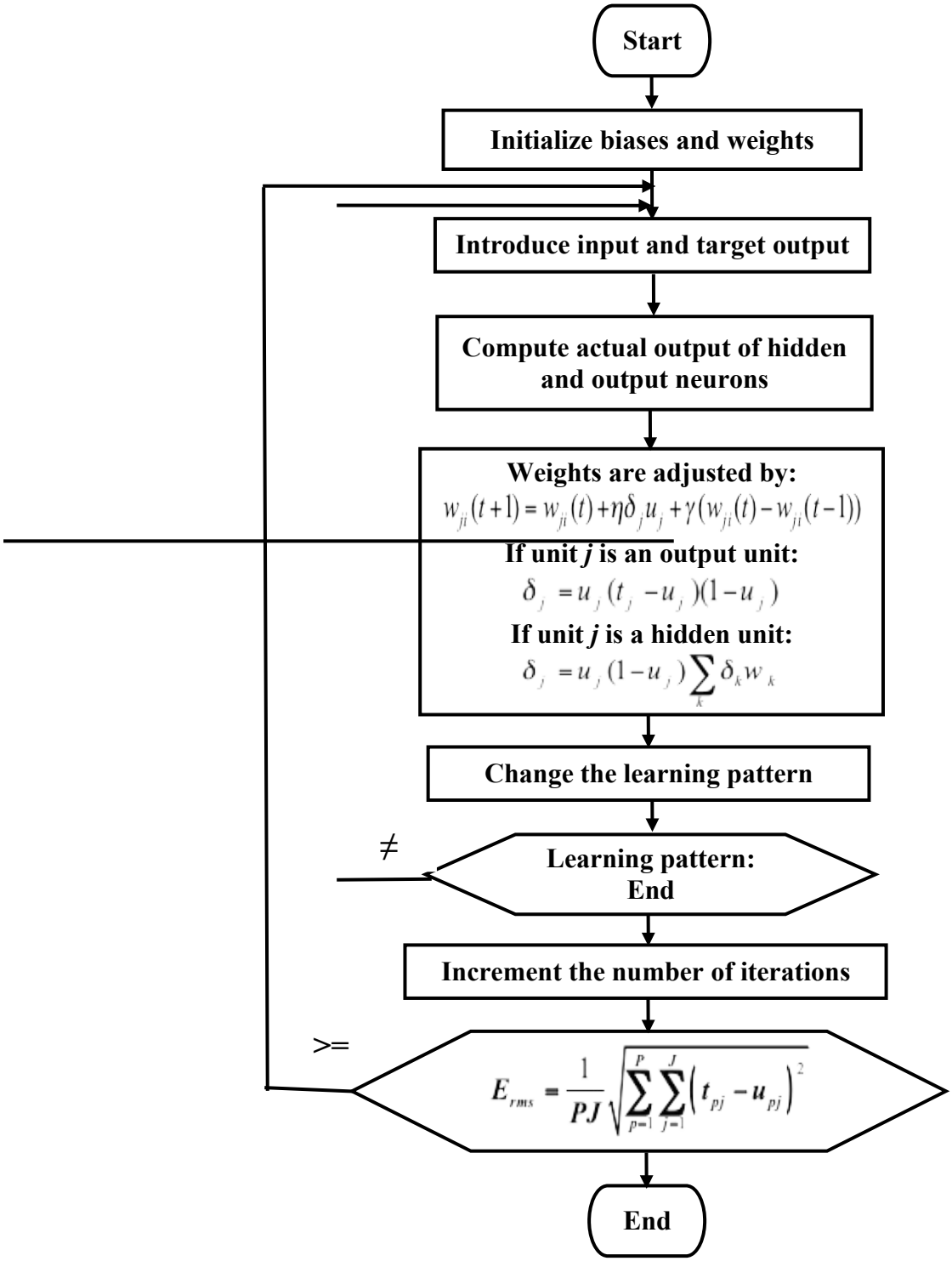


Fig 2. Flowchart of an off-line back-propagation training algorithm

The algorithm is a recursive algorithm that starts at the output units and working back to the first hidden layer. A comparison of the output X_j, Y_j, Z_j, R_j at the output layer with the desired outputs $tx,\ ty,\ tz,\ tr$ are performed using an error function which has the following form:

$$\delta_j = X_j(tx_j - X_j)(1 - X_j) + Y_j(ty_j - Y_j)(1 - Y_j) + Z_j(tz_j - Z_j)(1 - Z_j) + R_j(tr_j - R_j)(1 - R_j) \quad (29)$$

For the hidden layer, the error function takes the form:

$$\delta_j = \left\{ X_j(1 - X_j) + Y_j(1 - Y_j) + Z_j(1 - Z_j) + R_j(1 - R_j) \right\} \sum_k \delta_k w_k \quad . \quad (30)$$

where δ_j is the error term of the output layer, and w_k is the weight between the output and hidden layers. The update of the weight of each connection is implemented by replicating the error in a backward direction from the output layer to the input layer as follows:

$$w_{ji}(t+1) = w_{ji}(t) + \eta \delta_j u_j + \gamma (w_{ji}(t) - w_{ji}(t-1)) \quad (31)$$

The value of learning rate η is chosen such that it is neither too large leading to overshooting nor very small leading to a slow convergence rate. The value of the momentum term found in the last part in Equation (31) which is affixed with a constant γ (momentum) is used to accelerate the error convergence of the back-propagation learning algorithm and also to assist in pushing the changes of the energy function over local increases and boosting the weights in the direction of the overall downhill [28]. This term adds a fraction of the most recent weight values to the current weight values. Both η and γ terms are set at the start of the training phase and determine the network speed and stability [26], [30].

The process is repeated for each input pattern until the output error of the network is decreased to a pre-specified threshold value. The final weight values are frozen and utilized to get the precise product abundances during the test session. The quality and success of training of ANN are assessed by calculating the error for the whole batch of training patterns using the normalized RMS error that is defined as:

$$E_{rms} = \frac{1}{PJ} \sqrt{\sum_{p=1}^{P} \sum_{j=1}^{J} \left\{ \left(tx_{pj} - X_{pj} \right)^2 + \left(ty_{pj} - Y_{pj} \right)^2 + \left(tz_{pj} - Z_{pj} \right)^2 + \left(tr_{pj} - R_{pj} \right)^2 \right\}} \quad ,$$

(32)

where J is the number of output units, P is the number of training patterns, tx_{pj}, ty_{pj}, tz_{pj} and tr_{pj} are the desired outputs at unit j, whereas X_{pj}, Y_{pj}, Z_{pj}, and R_{pj} are the actual outputs at the same unit j. A zero error denotes that all the output patterns computed by the stellar helium burning model match the expected values perfectly and that the stellar helium burning model is fully trained. Similarly, internal unit thresholds are adjusted by supposing they are connection weights on links from the input with an auxiliary constant-value. The previous algorithm has been programmed using C++ programming language running on Windows 7 of a CORE i7 PC.

4. Results and discussions

4.1 Data Preparation

Based on the recurrence relations (Equation (15)), we computed one pure helium gas model; X_0=1, Y_0=0, Z_0=0, R_0=0 and three rich helium gas models; X_0=0.95, Y_0=0.05, Z_0=0, R_0=0; X_0=0.9, Y_0=0.1, Z_0=0, R_0=0; and X_0=0.85, Y_0=0.15, Z_0=0, R_0=0. The fractional parameter covers the range $\alpha = 0.5 - 1$ with a step of 0.05. The calculations are performed for a time $T = 2100$ s. Consequently, we have a total sum of 44 fractional helium burning models.

Figure (3) plots the two product abundances from gas models calculated at $\alpha = 0.95$, where the solid lines are for the pure helium model with initial abundance X_0=1, Y_0=0, Z_0=0, R_0=0; and the dashed lines are for the rich helium model with initial abundances X_0=0.95, Y_0=0.05, Z_0=0, R_0=0. The effects of changing the composition of the gas are remarkable, especially for the carbon C^{12}.

In figure (4), we illustrated the effects of changing the fractional parameters on the product abundances calculated for a gas model with initial abundance X_0=0.85, Y_0=0.15, Z_0=0, R_0=0. It is clear that the effects of the change of the fractional parameter on the behavior of the product abundances are small. This result is similar to the results obtained by [5] for the models computed in the sense of modified Riemann-Liouville fractional derivative.

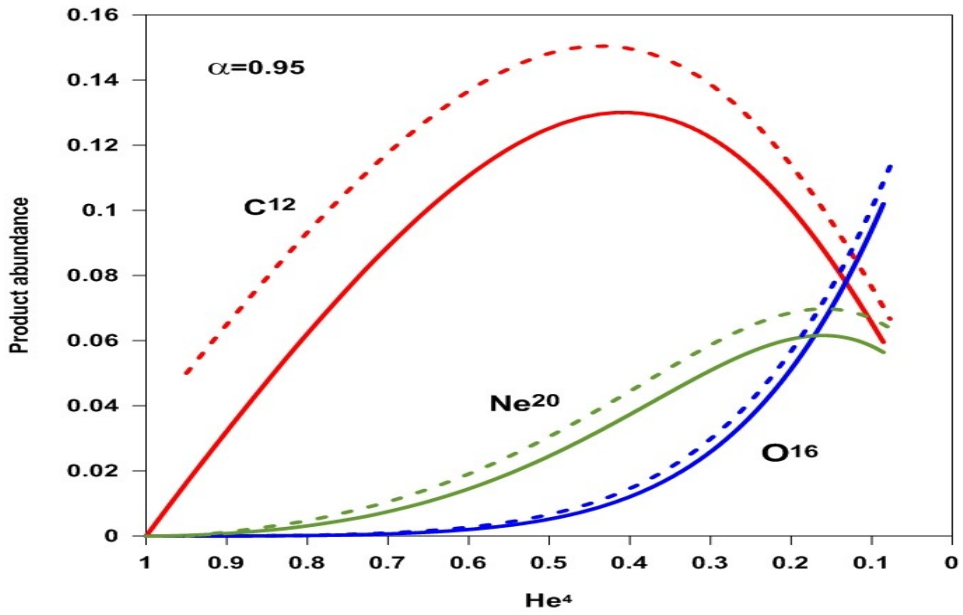


Figure 3. The product abundance (C12, O16 and Ne20) computed analytically at $\alpha = 0.95$ for the two different abundances. The solid lines represent the helium network with X0=1, Y0=0, Z0=0, R0=0, and the dashed lines represent the helium network X0=0.95, Y0=0.05, Z0=0, R0=0.

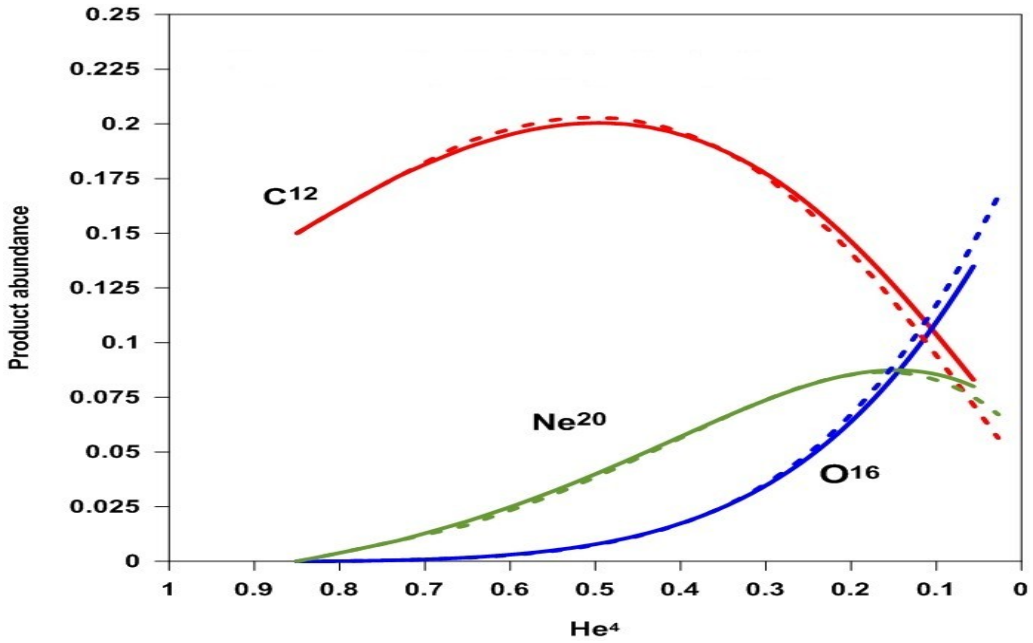


Figure 4. The product abundance (C12, O16, and Ne20) computed analytically at X0=0.85, Y0=0.15, Z0=0, R0=0 and for two different fractional parameters. The solid lines represent the helium network with $\alpha = 0.9$ and the dashed lines represent the helium network $\alpha = 0.5$.

4.2 ANN training

The training of NN used to simulate the helium burning network is implemented by using part of the data calculated in the previous subsection. The data used for training of the ANN are as shown in the second column of Table (1).

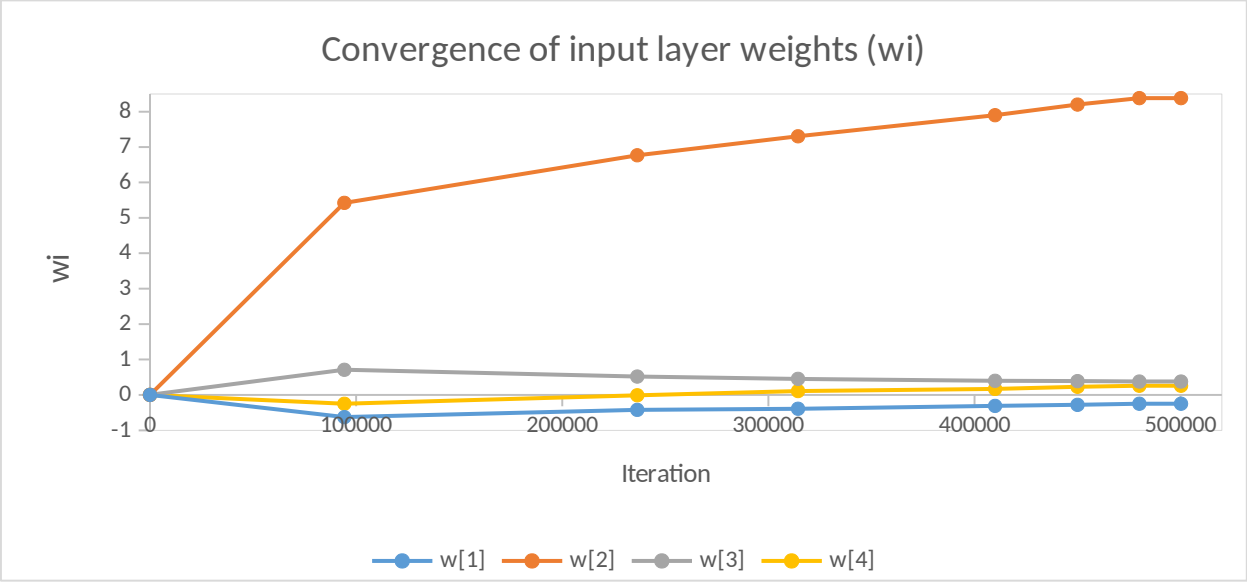
Table 1: Training and testing data for the helium burning network

	Training phase	Testing phase
α	0.5, 0.6, 0.7, 0.8, 0.9, 1	0.55, 0.65, 0.75, 0.85, 0.95
Time	0 - 2100 sec (Δt= 3)	0 - 2100 sec (Δt= 3)
Initial abundances of the HB	X_0=0.85, Y_0=0.15, Z_0=0, R_0=0	X_0=0.85, Y_0=0.15, Z_0=0, R_0=0
	X_0=0.90, Y_0=0.1, Z_0=0, R_0=0	X_0=0.90, Y_0=0.1, Z_0=0, R_0=0
	X_0=0.95, Y_0=0.05, Z_0=0, R_0=0	X_0=0.95, Y_0=0.05, Z_0=0, R_0=0
	X_0=1, Y_0=0, Z_0=0, R_0=0	X_0=1, Y_0=0, Z_0=0, R_0=0

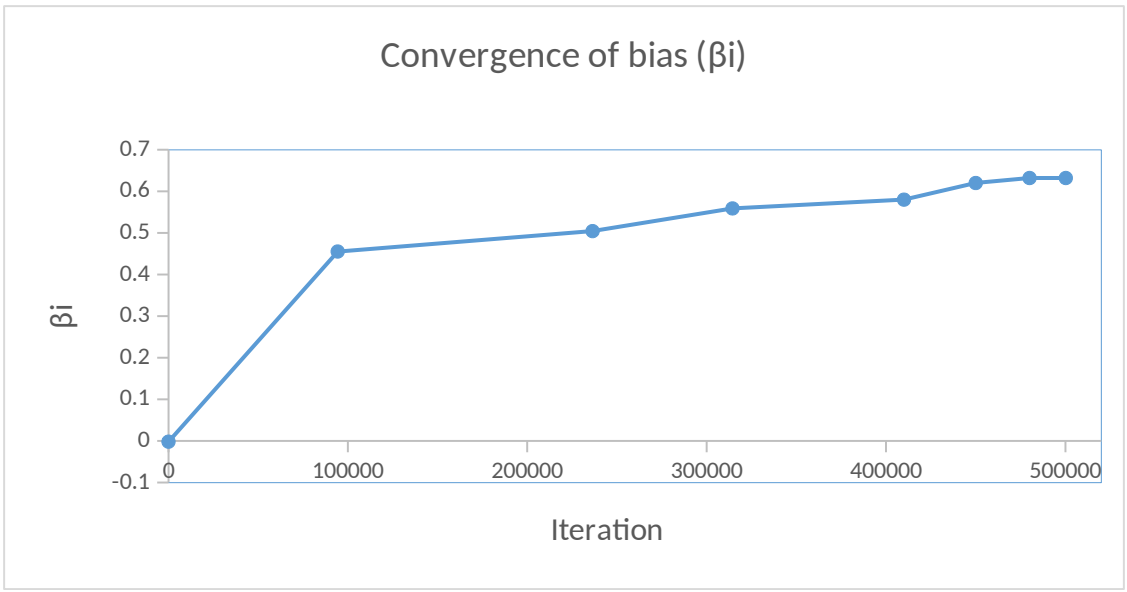
The neural network (NN) architecture used in this paper for the helium burning network has three layers as shown in Figure 1. These layers are the input layer, hidden layer, and output layer. Different configurations of hidden neurons of 10, 20, and 40 have been tested, where we

concluded that 20 neurons in a single hidden layer are giving the best model of the network to simulate the helium burning network. This number of neurons in the hidden layer was found to give the minimum value of RMS error of 0.000005 in an almost similar number of training iterations. As a result, the configuration of the NN we used was 4-20-4, where the input layer has four inputs which are the fractional parameter α, the time t (t takes values from 3 to 2100 in steps of 3 seconds), two of the initial abundances which are the helium (X_0), and carbon (Y_0). We excluded the other two initial abundances (Z_0 and R_0) because their values are always zero as indicated in Table 1. The output layer has 4 nodes which are the time dependent product abundances for helium (X), carbon (Y), oxygen (Z), and neon (R).

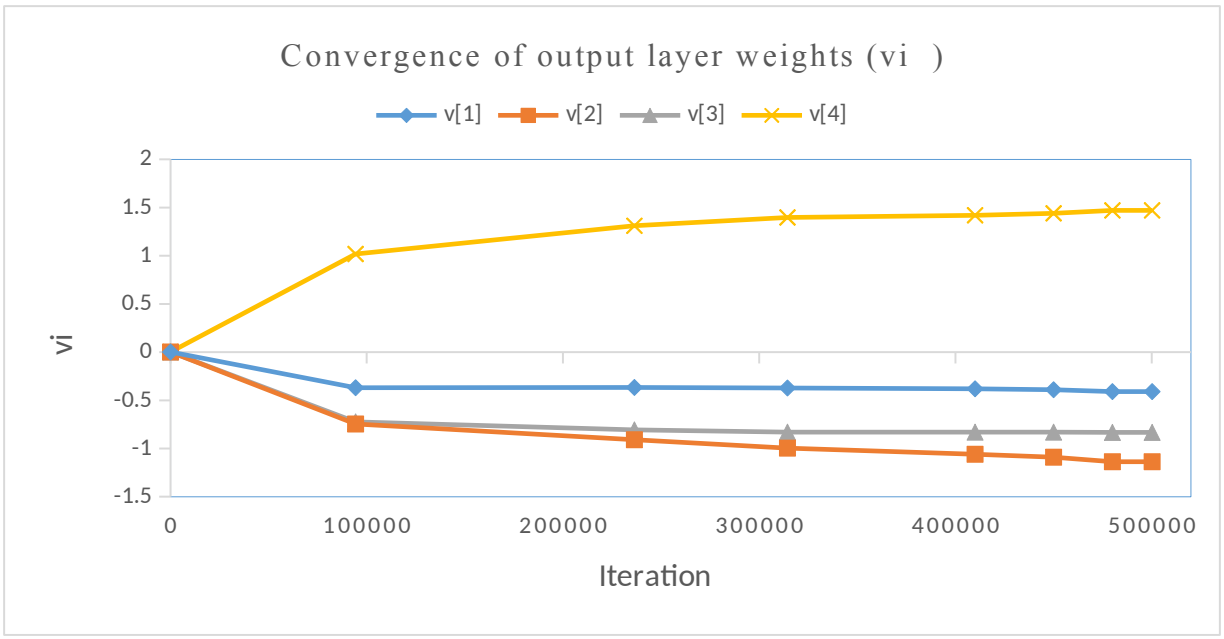
During the training of the NN, we used a value for the learning rate ($\eta = 0.035$) and for the momentum ($^{\gamma} = 0.5$). Those values for η and $^{\gamma}$ were proved to quicken the convergence of the back-propagation training algorithm without exceeding the solution. For the demonstration of the convergence and stability of the values computed for weight parameters of network layers, the behaviors of the convergence of the input layer weights, bias, and output layer weights (w_i , β_i, and v_i) for the helium burning network are as displayed in Fig. 5. As these figures show, the weight values are initialized to random values and after somewhat considerable iterations they converged to stable values.



(a) The convergence of input layer weights (w_i)



(a) The convergence of bias (β_i)

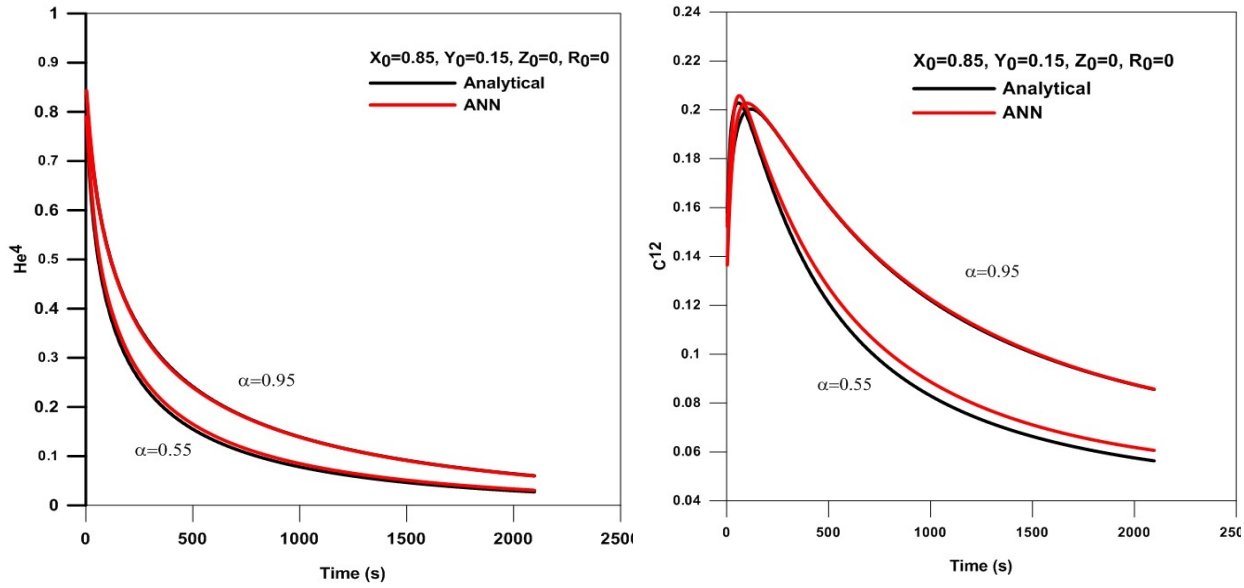


(c) Convergence of output layer weights (v_i)

Figure 5: Convergence of the weights of input, bias, and output layers for the training of the NN used to simulate helium burning network

4.3 Comparison between the NN model and analytical model

After the end of the training phase of NN, we used the final frozen weight values in the test phase to predict the time dependent product abundances for helium (X), carbon (Y), oxygen (Z), and neon (R). In this test phase, we used values for a fractional parameter α not being used in the training phase to predict the helium burning network model. These values are shown in the third column of Table 1. The results of the predicted values show very good agreement with the analytical values for different helium modes. A comparison between the predicted NN model values and analytical model for two values of the fractional parameters (α =0.55 and α=0.95) along with different helium modes shown in Table 1 are displayed in the range of figures from Fig. 6 till Fig. 9 for one pure helium gas model; X0=1, Y0=0, Z0=0, R0=0 and three rich helium gas models; X0=0.95, Y0=0.05, Z0=0, R0=0; X0=0.9, Y0=0.1, Z0=0, R0=0; and X0=0.85, Y0=0.15, Z0=0, R0=0. In all of these figures, it is clear the very good agreement between both of NN model and analytical model, which elect the NN to be considered as a powerful tool to solve and model nuclear burning networks and could be applied to the other nuclear stellar burning networks.



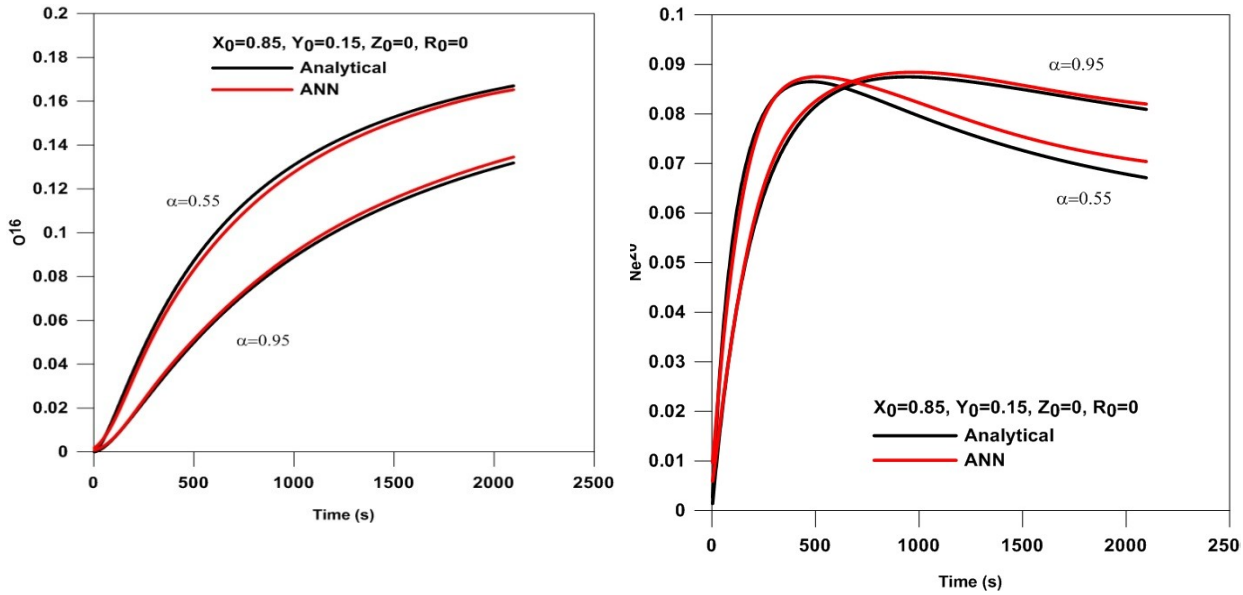
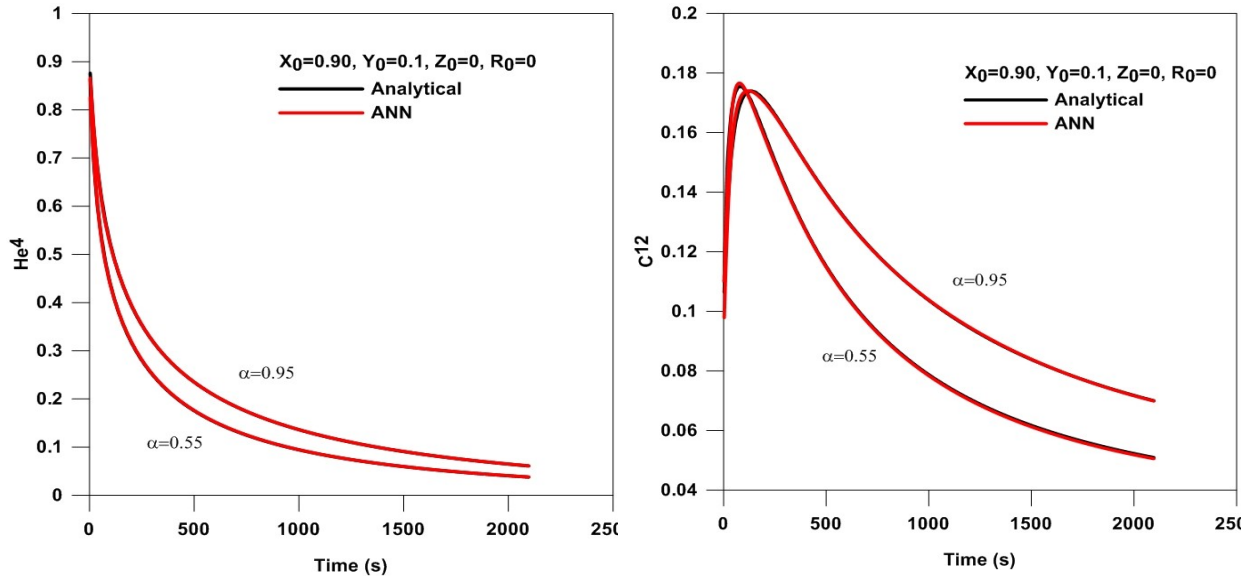


Figure 6: The distribution of the product abundance with time for the rich helium burning network, Xo=0.85, Yo=0.15, Zo=0, Ro=0.



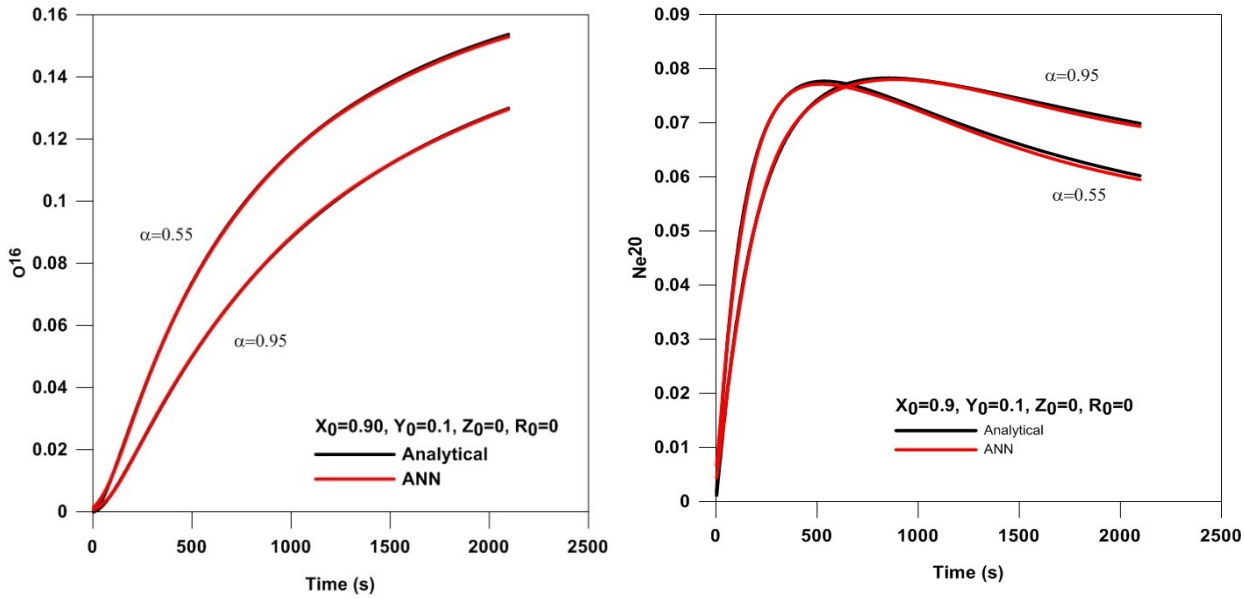
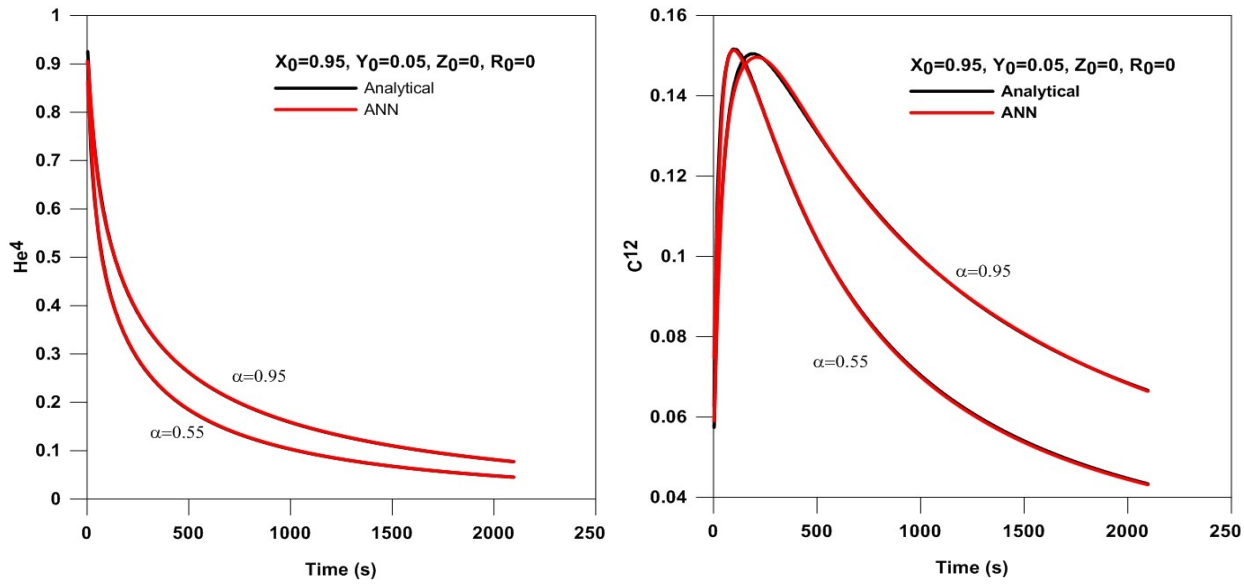


Figure 7: The distribution of the product abundance with time for the rich helium burning network, Xo=0.9, Yo=0.1, Zo=0, Ro=0.



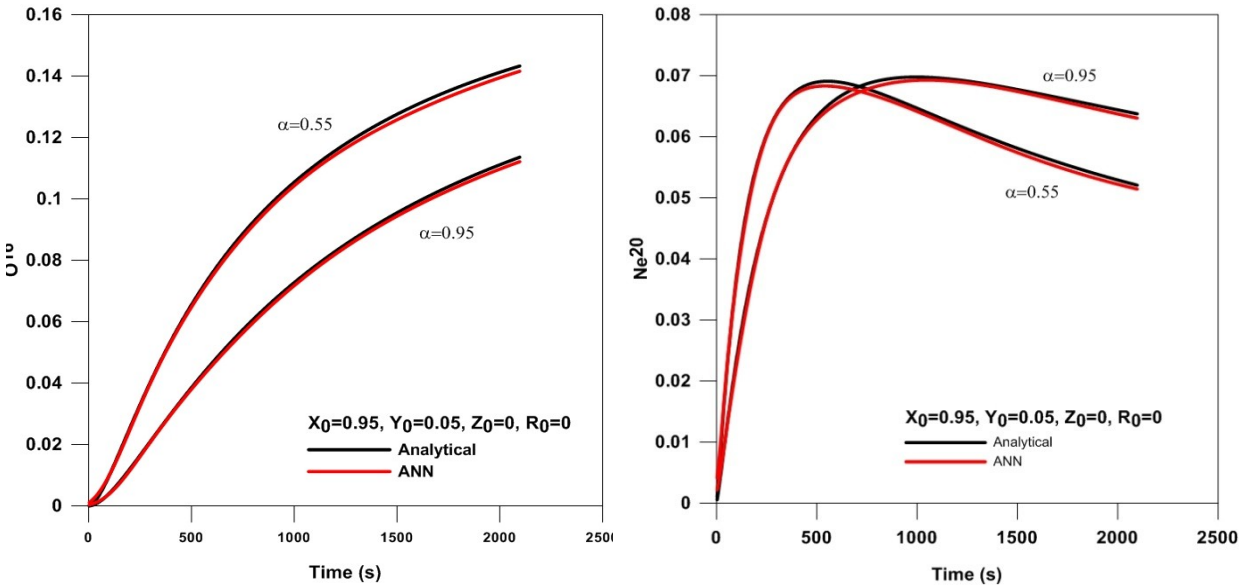
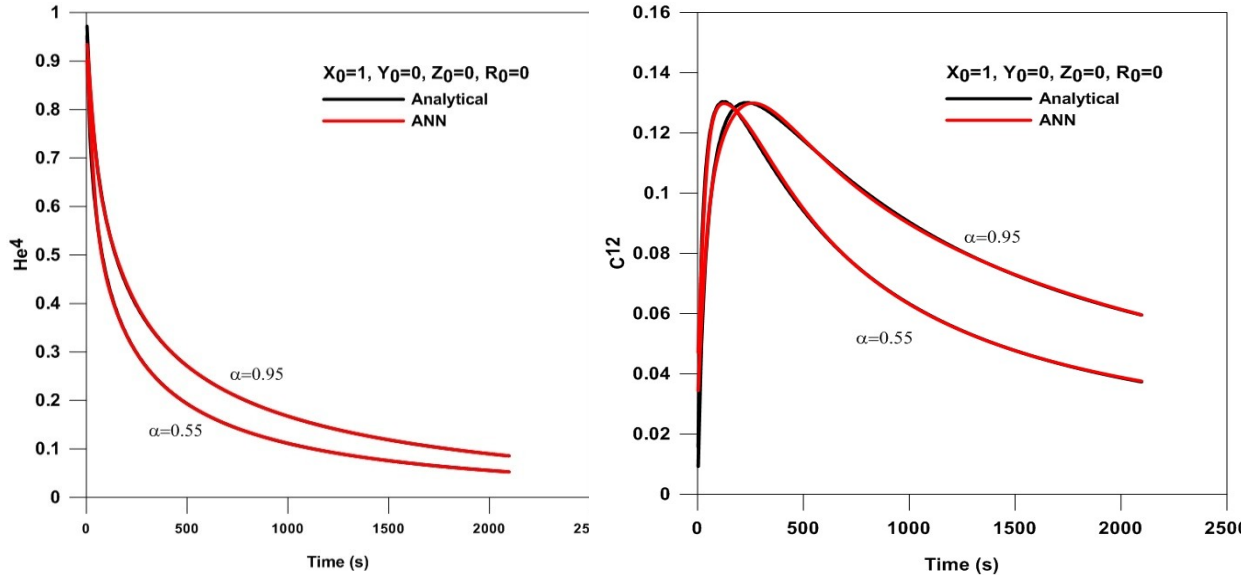


Figure 8: The distribution of the product abundance with time for the rich helium burning network, Xo=0.95, Yo=0.05, Zo=0, Ro=0.



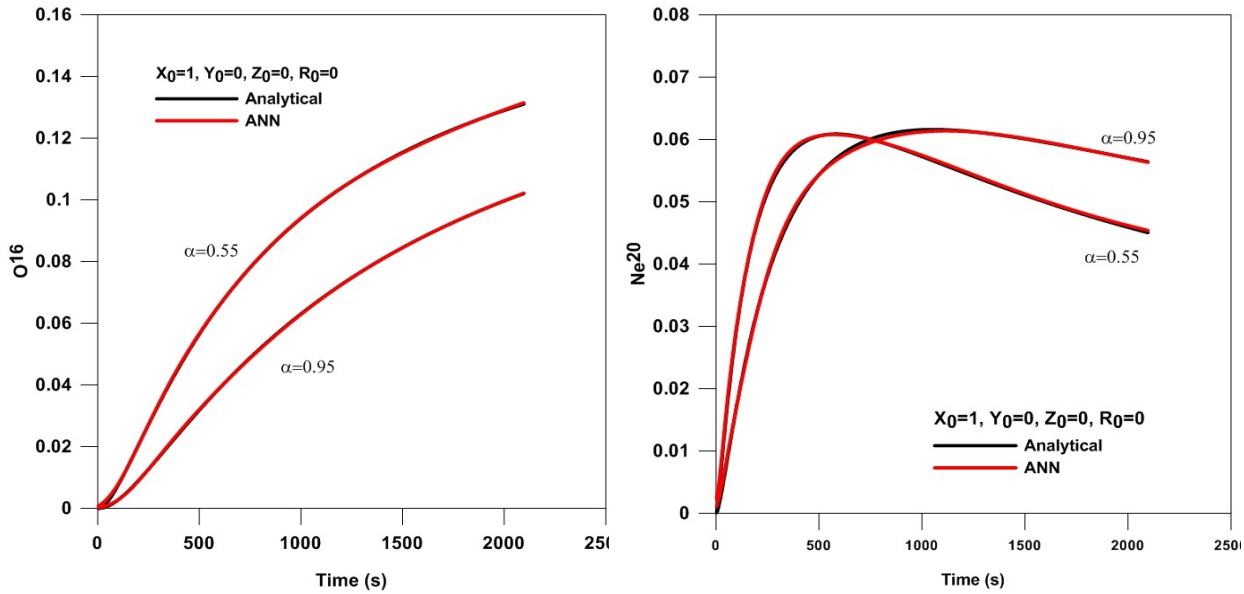


Figure 9: The distribution of the product abundance with time for the pure helium burning network, Xo=1, Yo=0, Zo=0, Ro=0.

5. Conclusion

In the current research, we introduced an analytical solution to the conformable fractional helium burning network via a series expansion method where we obtained the product abundances of the syntheses elements as a function of time. The calculations are performed for the four different initial abundances; (X_0=1, Y_0=0, Z_0=0, R_0=0), (X_0=0.95, Y_0=0.05, Z_0=0, R_0=0); (X_0=0.9, Y_0=0.1, Z_0=0, R_0=0); and (X_0=0.85, Y_0=0.15, Z_0=0, R_0=0). The results of the analytical solution revealed that the conformable models have the same behaviors as the fractional models computed using the modified Riemann-Liouville fractional derivative. Second, we used the NN in its feed forward type to simulate the system of the differential equations of the HB. To do that, we performed the mathematical modeling of a NN to simulate the conformable helium burning

network. We trained the NN using back propagation delta rule algorithm and used training data for models with the fractional parameter range $\alpha = 0.5 - 1$ with step $\Delta\alpha = 0.1$. We predicted the fractional models for the range $\alpha = 0.55 - 0.95$ with step $\Delta\alpha = 0.1$. The comparison with the analytical solutions gives a very good agreement for most cases, a small difference obtained for the model with fractional parameters $\alpha = 0.55$. The results obtained in this research proves that modeling of nuclear burning networks using NN give very good results and validates the NN to be an accurate, robust, and trustworthy method to solve and model similar networks and could be applied to other nuclear stellar burning networks comprised of conformable fractional differential equations.

Acknowledgments: The authors would like acknowledged the Academy of Scientific Research & Technology (ASRT), Egypt Grant no. 6413 under the project Science Up.

References

[1] R.A. El-Nabulsi, Appl. Math. Comput. 218, 2837 (2011).

[2] S.S. Bayin, J.P. Krisch, Astrophys. Space Sci. 359, 58 (2015)

[3] E.A.-B. Abdel-Salam, M.I. Nouh, Astrophysics 59, 398 (2016)

[4] M. I. Nouh and E.A.-B. Abdel-Salam, EPJP, 133, 149.

[5] Nouh, M.I., 2019. New Astronomy 66, 40

[6] E.A.-B. Abdel-Salam, M.I. Nouh, 2020, New Astronomy, 76, 101322

[7] Yousif , E., Adam, A., Hassaballa, A. and Nouh, M. I., (2021), New Astronomy, 84, 101511

[8]M. W. M. G. Dissanayake and N. Phan-Thien. Neural-network-based approximations for solving partial differential equations, Communications in Numerical Methods in Engineering,10(3):195–201, March 1994.

[9]I.E. Lagaris, A. Likas, and D.I. Fotiadis. Artificial neural networks for solving ordinary and partial differential equations, IEEE Transactions on Neural Networks, 9(5):987–1000, September 1998.

[10]I.E. Lagaris, A.C. Likas, and D.G. Papageorgiou. Neural-network methods for boundary valueproblems with irregular boundaries. IEEE Transactions on Neural Networks, 11(5):1041–1049,September 2000. Conference Name: IEEE Transactions on Neural Networks.

[11]Kevin Stanley McFall and James Robert Mahan. Artificial Neural Network Method for Solution of Boundary Value Problems With Exact Satisfaction of Arbitrary Boundary Conditions, IEEE Transactions on Neural Networks, 20(8):1221–1233, August 2009. Conference Name: IEEE Transactions on Neural Networks.

[12]Modjtaba Baymani, Asghar Kerayechian, and Sohrab Effati. Artificial Neural Networks Approach for Solving Stokes Problem, Applied Mathematics, 01(04):288–292, 2010.

[13] Lu Lu, Xuhui Meng, Zhiping Mao, and George E. Karniadakis. DeepXDE: A deep learning library for solving differential equations. arXiv:1907.04502 [physics, stat], February 2020.arXiv: 1907.04502.

[14]Alexander Koryagin, Roman Khudorozkov, and Sergey Tsimfer. PyDEns: a Python Framework for Solving Differential Equations with Neural Networks.arXiv:1909.11544 [cs,stat], September 2019. arXiv: 1909.11544.

[15]Feiyu Chen, David Sondak, Pavlos Protopapas, Marios Mattheakis, Shuheng Liu, Devansh Agarwal, and Marco Di Giovanni. Neuro Diff Eq: A Python package for solving differential equations with neural networks, Journal of Open Source Software, 5(46):1931, February 2020.

[16] Nouh, M.I., Azzam, Y.A. & Abdel-Salam, E. A-B., 2020, Neural Comput & Applic,

https://doi.org/10.1007/s00521-020-05277-9

[17] Azzam, Y. A.; Abdel-Salam, E. A. -B.; Nouh, M. I., 2020, arXiv:2010.12768.

[18] Clayton, D.D., 1983. Principles of Stellar Evolution and Nucleosynthesis. University of Chicago Press, Chicago.

[19] Duorah, H.L., Kushwaha, R.S., 1963. Helium-Burning reaction products and the rate of energy generation. Astrophys. J. 137, 566

[20] Hix, W.R., Thielemann, F.-K., 1999. Computational methods for nucleosynthesis and nuclear energy generation. J. Comput. Appl. Math. 109, 321

[21] Nouh, M. I., Sharaf, M. A. and Saad, A. S., 2003, AN, 324, 432.

[22] Haubold, H.J., Mathai, A.M., 2000. The fractional kinetic equation and thermonuclear functions. Astrophys. Space Sci. 273, 53–63

[23] Saxena, R.K., Mathai, A.M., Haubold, H.J., 2002. On fractional kinetic equations,. Astrophys. Space Sci. 282, 281–287.

[24] Chaurasia, V., Pandey, S., 2010. Research in Astron. Astrophys. 10, 22

[25] Yadav N, Yadav A, Kumar M (2015) An introduction to neural network methods for differential equations, springer briefs in applied science and technology. Springer, Berlin

[26] Elminir Hamdy K, Azzam Yosry A, Younes Farag I (2007) Prediction of hourly and daily diffuse fraction using neural network, as compared to linear regression models. Energy 32:1513–1523

[27] Fukuda T, Hasegawa Y, Sekiyama K, Aoyama Tadayoshi (2012), Multi-Locomotion Robotic Systems: New Concepts of Bio-inspired Robotics. Springer, Berlin

[28] Cornelia Denz, 1998, Optical Neural Networks, Springer.

[29] Toshio Fukuda, Yasuhisa Hasegawa, Kosuke Sekiyama, Tadayoshi Aoyama, 2012, Multi-Locomotion Robotic Systems: New Concepts of Bio-inspired Robotics, Springer.

[30] Basheer IA, Hajmeer M (2000) Artificial neural networks: fundamentals, computing, design, and application. J Microbiol Methods 43:3–31