Introduction

The codes presented follow a few examples in *Time Series: Theory and Methods* by Peter J. Brockwell, Richard A. Davis[1]. Examples 3.3.1 and 3.3.2 deal with finding the auto-covariance function (ACF) $\gamma(t)$ for an $ARMA[2,1]$ process. Three different methods for finding $\gamma(t)$ are coded. Any of these methods can then be used as a pre-requisite to employing the innovations algorithm to predict $\hat{X}_{n+1}$, as is done in example 5.3.4 which deals with an $ARMA[2,3]$ process. Specific references to page numbers and examples are made throughout the code.

Here I'll clarify a few things which I found unclear on a first reading and then focus on describing the methods of converting the results given by Brockwell and Davis into a computatoinal algorithm for finding $\gamma(t)$. The innovations algorithm will not be discussed, but it has been coded for and the results based the three different methods of finding $\gamma(t)$ are presented at the end.

A brief note about the projection theorem is in order as Brockwell and Davis make extensive use of it. At first glance it involves more esoteric notation than other texts seem to use, but it is worth the effort to familiarize oneself with it as it is a powerful approach in parameter estimation. Again referring the reader to the text that is available online, I will only clarify some things which I myself found difficult to grasp on first reading.

The Projection Theorem

Given observed values $x_t, x_{t-1}, x_{t-2}, ...$ a widely used method to predicting x_{n+1} is weighted least square linear regression. To take two the simple case of two observed values x_1, x_2 we seek to fit a line $w = \beta_1 x_1 - \beta_2 x_2$ which will [hopefully] allow us to predict a future estimate x_3. To be as general as possible let x_1, x_2 be vectors of length k.

Defining the inner product operator as,

$$\langle x, w \rangle = \sum_{i=1}^{k} x_i w_i$$

where k is the length of x and w. The norm of vector x is defined as,

$$\|x\| = \sqrt{\langle x, w \rangle}$$

We seek to minimize the quantity,

$$\begin{aligned} S^2 = \|w - \beta_1 x_1 - \beta_2 x_2\|^2 &= \|w - \sum_{i=1}^{2} \beta_i x_i\|^2 \\ &= \langle w - \sum_{i=1}^{2} \beta_i x_i \ , \ w - \sum_{i=1}^{2} \beta_i x_i \rangle \\ &= \sum_{j=1}^{k} \left[w_j - \sum_{i=1}^{2} \beta_i x_{i,j} \right]^2 \end{aligned}$$

Where β_2, β_2 are constants. We would minimize with respect to the β coefficients as follows,

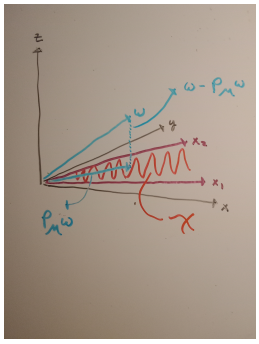
$$\begin{aligned}
\frac{\partial}{\partial \beta_s} S^2 &= \frac{\partial}{\partial \beta_s} \sum_{j=1}^{k} \left[w_j - \sum_{i=1}^{2} \beta_i x_{i,j} \right]^2 \\
&= \sum_{j=1}^{k} x_{s,j} \left[y_j - \sum_{i=1}^{2} \beta_i x_{i,j} \right] \\
&= \langle x_s, w \rangle - \sum_{i=1}^{2} \beta_i \langle x_s, x_i \rangle \\
&= \langle w - \sum_{i=1}^{2} \beta_i x_i \ , \ x_s \rangle = 0 \qquad\qquad s \in \{1, 2\}
\end{aligned}$$

Calling $\hat{w} = \sum_{i=1}^{2} \beta_i x_i$ the projection of w onto the basis x_1, x_2 we have the projection theorem:

$$\langle w - \hat{w} \ , \ x \rangle = 0 \qquad\qquad x \in \{x_1, x_2\}$$

In terms of the projection operator $\mathcal{P}$ we define the subspace $\mathcal{M}$ as the set of all possible vectors which can be formed by taking linear combinations of x_1 and x_2, which is to say all vectors which lie in the plane $\alpha_i x_1 + \alpha_j x_2$ where α_i, α_j are real constants. We used α instead of β here only to emphasize that $\mathcal{M}$ encompasses the entire plane, not just the final projected vector, which we now denote as $\mathcal{P}_{\mathcal{M}} w = \hat{w} = \beta_1 x_1 + \beta_2 x_2$. We then have,

$$\langle w - \mathcal{P}_{\mathcal{M}} w \ , \ x \rangle = 0 \qquad\qquad x \in \mathcal{M}$$



Geometrically speaking, what we've done is minimize the mean-squared distance between vector w and the plane formed by x_1 and x_2 . Stated otherwise, given a hypothesized vector w and two observed vectors x_1, x_2, the best we can do for an estimate of w is to find a vector which lies *somewhere* in the plane $\mathcal{X} = \{\alpha_i x_1 + \alpha_j x_2 \ ; \ \alpha_i, \alpha_j \in \mathcal{R}\}$. More specifically, the vector closest to w in this plane will be the one defined as the *orthogonal* projection of w onto this plane, which is the vector we have found ($\mathcal{P}_{\mathcal{M}} w = \beta_1 x_1 + \beta_2 x_2$), as is evident by the fact that the inner product (or dot product) resulted in a value of zero. We then have that $w - \mathcal{P}_{\mathcal{M}} w$ must be orthogonal to all possible vectors which lie in $\mathcal{X}$. The situation is depicted in the figure to the right.

Brockwell and Davis would denote the plane $\mathcal{X}$ as $\overline{sp}\{x_1, x_3\}$ where $\overline{sp}$ stands for 'closed span'. As previously mentioned, the notation is a bit esoteric, but familiarizing oneself with it has its uses. There are for example, a number of useful properties of the projection operator which can be derived in a very succinct fashion using the symbology $\mathcal{P}_{\overline{sp}\{x_1, x_3, \ldots, x_n\}}$.

ACF

The $ARMA[p, q]$ process is modeled as,

$$\phi(\mathbf{B}) x_t = \theta(\mathbf{B}) z_t \tag{1}$$

Where $\mathbf{B}$ is the backshift operator, $\phi(\mathbf{B}) = 1 - \phi_1 \mathbf{B} - \phi_2 \mathbf{B}^2 - ... - \phi_p \mathbf{B}^p$, $\theta(\mathbf{B}) = 1 + \theta_1 \mathbf{B} + \theta_2 \mathbf{B}^2 + ... + \theta_p \mathbf{B}^q$, x_t are zero-mean i.i.d random variables with variance σ^2, and z_t is a white noise process with zero mean, variance σ^2, and autocovariance values which obey the following relation,

$$\gamma(h) = \begin{cases} \sigma^2, & \text{if } h = 0 \\ 0, & \text{if } h \neq 0 \end{cases}$$

We can invert the process using,

$$\psi(\mathbf{B}) = \sum_{i=0}^{\infty} \psi_i \mathbf{B}^i = \frac{\theta(\mathbf{B})}{\phi(\mathbf{B})}$$

Rewriting the above as $\psi(\mathbf{B})\phi(\mathbf{B}) = \theta(\mathbf{B})$, then comparing coefficients of $\mathbf{B}$ we get the following relation for the coefficients of ψ_i

$$\psi_i \ = \begin{cases} \theta_j + \sum\limits_{0<k\leq j} \phi_k \psi_{j-k}, & \text{if } 0 \leq j < max(p, q+1) \\ \sum\limits_{0<k\leq p} \phi_k \psi_{j-k}, & \text{if } j \geq max(p, q+1) \end{cases} \tag{2}$$

Which is equation 3.3.3 and 3.3.4 in [1]. Equation 2 is the first thing to code for (see $solve\_\psi\_ARMA(\theta, \phi)$ in $ACF\_methods.py$ in the $funcs\_$ folder).

In order to proceed with the first and second methods of finding the ACF consider the following expression

$$h_t = \sum_{i=1}^{k} \sum_{j=0}^{r_i - 1} \beta_{ij} t^j \zeta_i^{-t} \qquad\qquad t \geq max(p, q+1) - p \tag{3}$$

where β_{ij} are constants, ζ_i is one of k unique roots of the polynomial,

$$\phi(\mathbf{B}) = 1 - \phi_1 \mathbf{B} - \phi_2 \mathbf{B}^2 - ... - \phi_p \mathbf{B} \ = \ 0 \tag{4}$$

and r_i is the multiplicity of the root ζ_i. Equation 3 is the general solution of $\phi(\mathbf{B})h_t = 0$ where $h_t = h(t)$. To see this note that by the fundamental theorem of algebra every univariate polynomial can be factored into monic terms. In our case we have

$$\left(1 - \phi_1 \mathbf{B} - \phi_2 \mathbf{B}^2 - ... - \phi_p \mathbf{B}\right) h_t \ = \ (\zeta_1 - \mathbf{B})^{r_1} (\zeta_2 - \mathbf{B})^{r_2} ... (\zeta_k - \mathbf{B})^{r_k} h_t \ = \ 0$$

dividing by $\zeta_1^{r_1} \zeta_2^{r_2} ... \zeta_k^{r_k}$ we get

$$\left((1 - \zeta_1^{-1}\mathbf{B})^{r_1} (1 - \zeta_2^{-1}\mathbf{B})^{r_2} ... (1 - \zeta_k^{-1}\mathbf{B})^{r_k}\right) h_t = 0$$

It is then relatively easy to see that equation 3 is the general solution of $\phi(\mathbf{B})h_t = 0$. Take the simple case of an $AR[2]$ process,

$$\left(1 - \mathbf{B} + \frac{1}{4}\mathbf{B}^2\right) h_t = \left(1 - \frac{1}{2}\mathbf{B}\right)^2 h_t = 0 \tag{5}$$

In this case we have one unique root ($k = 1$) with a multiplicity of two, so $r_1 = 2$. The value of this root is $\zeta_1 = 2$. If we take $\sigma^2 = 1$ we get for a solution,

$$h_t = (\beta_{10} + t\beta_{10}) 2^{-t} \tag{6}$$

Using the property of the backshift (or lag) operator $\mathbf{B}f(t) = f(t-1)$ it is easily verified that this is a solution to equation 5.

1 Computational methods for finding $\gamma(t)$ for $ARMA[2,1]$ process

Now for the computational method for determining the solution we need to specify under what conditions equation 1 reduces to 4. There are three methods for finding $\gamma(t)$ presented on pgs. 91-97 of [1], but only the first two make use of equation 3. Here I'll simply cite the results then show how the code functions so as to solve these equations computationally. We will begin with the second method for finding $\gamma(t)$ as its relation to equation 3 is the most direct.

In order to exemplify the computational method, we continue using the following ARMA[2,1] process,

$$x_t - x_{t-1} + \frac{1}{4}x_{t-2} = z_t + z_{t-1}$$

second method

The second method for finding $\gamma(t)$ begins with the equations (see equations 3.3.8 and 3.3.9 in [1]),

$$\gamma(t) - \phi_1\gamma(t-1) - ... - \phi_p\gamma(t-p) \;= \begin{cases} \sigma^2 \sum\limits_{t\le j\le q} \theta_j\psi_{j-t}, & \text{if } 0 \le t < max(p, q+1) \\ 0, & \text{if } t \ge max(p, q+1) \end{cases} \tag{7}$$

The first of the above equations are the intial conditions that we use to determine the constants β_{ij} in the general solution (equation 3). As previously noted the ψ coefficients are the first thing we need to find the values of, and this is accomplished with the function $solve\_\psi\_ARMA(\theta, \phi)$ contained in $ACF\_methods.py$ in the $funcs\_$ folder.

Equation 6 has two constants we need to solve for. Noting the time constraint in equation 3; $t \ge max(p, q+1) - p$, in comparison to the initial conditions in equation 7 we see there is an overlap of p time values which we can use in order to solve for the p different coefficients β_{ij}. In general we have p constants so we need p initial conditions.

Replacing h_t with $\gamma_t = \gamma(t)$, we can use the general solution for solving equation 7. Doing so and also utilizing the property of the ACF that $\gamma(-t) = \gamma(t)$ we get for the left hand side of 7 we get,

$$\begin{aligned} &\gamma(t) - \phi_1\gamma(t-1) - ... - \phi_p\gamma(t-p) \\ &\quad= \sum_{i=1}^{k}\sum_{j=0}^{r_i-1} \beta_{ij} t^j \zeta_i^{-t} - \phi_1 \sum_{i=1}^{k}\sum_{j=0}^{r_i-1} \beta_{ij} |t-1|^j \zeta_i^{-|t-1|} - ... - \phi_p \sum_{i=1}^{k}\sum_{j=0}^{r_i-1} \beta_{ij} |t-p|^j \zeta_i^{-|t-p|} \\ &\quad= \sum_{i=1}^{k}\sum_{j=0}^{r_i-1} \left[t^j \zeta_i^{-t} - \sum_{l=1}^{p} \phi_l |t-l|^j \zeta_i^{-|t-l|} \right] \beta_{ij} \;= 0 \qquad t \;\ge\; max(p, q+1) - p \end{aligned} \tag{8}$$

We can begin to see how this equation can be coded for the general case of an $ARMA[p,q]$ process; for every i, j step we allow the matrix element $M_{i,j}$ to be the value in the square brackets. We then need to construct an array of initial conditions (calll it v_0) according to the first (upper) value on the right side of equation 7. We then solve the matrix equation $M\beta = v_0$ where $\beta = [\beta_{10}, ..., \beta_{1\,r_0-1}, ..., \beta_{k0}, ..., \beta_{k\,r_k-1}]$

The polynomial we are concerned with is $\phi(\mathbf{B}) = 1 - \mathbf{B} + \frac{1}{4}\mathbf{B}^2$ which has only one root and this implies $k = 1$. The value of the root is $\zeta_1 = 2$ with multiplicity 2. When this polynomial acts on $\gamma(t)$ we get $\gamma(t) - \phi_1\gamma(t-1) - \phi_2\gamma(t-2)$. The general solution to $\gamma(t)$ is,

$$\gamma(t) = (\beta_{10} + t\beta_{11})2^{-t} \qquad\qquad t \ge max(p, q+1) - p = p - p = 0 \tag{9}$$

We need to take a superposition of the above solution in accordance with 8. We have,

$$\gamma(t) - \phi_1\gamma(t-1) - \phi_p\gamma(t-2)$$
$$= \sum_{j=0}^{1}\left[t^j 2^{-t} - \sum_{l=1}^{2}\phi_l|t-l|^j 2^{-|t-l|}\right]\beta_{1j} \;= \begin{cases}\sigma^2 \sum\limits_{t\le j\le 1}\theta_j\psi_{j-t}, & \text{if } 0\le t<2\\ 0, & \text{if } t\ge 2\end{cases}$$

Putting this in matrix form,

$$\left[(2^{-t} - 2^{-|t-1|} + \tfrac{1}{4}2^{-|t-2|})\;,\quad (t2^{-t} - |t-1|2^{-|t-1|} + \tfrac{1}{4}|t-2|2^{-|t-2|})\right]\begin{bmatrix}\beta_{10}\\ \beta_{11}\end{bmatrix} = \begin{cases}\sigma^2 \sum\limits_{t\le j\le 1}\theta_j\psi_{j-t}, & \text{if } 0\le t<2\\ 0, & \text{if } t\ge 2\end{cases}$$

Using $\theta_0 = \psi_0 = 1, \psi_1 = \theta_0 + \psi_0\phi_1 = \theta_1 + \phi_1 = 2$ we can ascertain the initial conditions for $t < p = 2$. Doing so for $t = 0, 1$ and stacking the results as rows of a matrix we have,

$$\begin{bmatrix}(1 - 2^{-1} + \frac{1}{4}2^{-2})\;, & (0 - 2^{-1} + \frac{1}{4}2*2^{-2})\\ (2^{-1} - 1 + \frac{1}{4}2^{-1})\;, & (2^{-1} - 0 + \frac{1}{4}2^{-1})\end{bmatrix}\begin{bmatrix}\beta_{10}\\ \beta_{11}\end{bmatrix} = \begin{bmatrix}\theta_0\psi_0 + \psi_1\\ \theta_0\end{bmatrix} = \begin{bmatrix}3\\ 1\end{bmatrix}$$

which reduces to,

$$\begin{bmatrix}3\;, & -2\\ -3\;, & 5\end{bmatrix}\begin{bmatrix}\beta_{10}\\ \beta_{11}\end{bmatrix} = \begin{bmatrix}16\\ 8\end{bmatrix}$$

We then make use of the function $Isolve$ contained in $Complex\_solve.py$ within the $funcs\_$ folder to solve for the coefficients β_{10}, β_{11} with the result $\beta_{10} = 8,\ \beta_{11} = 32/3 \approx 10.667$. These results are in agreement with those presented in [1].

The general solution now takes the form,

$$\gamma_t = (10.667 + 8t)2^{-t}$$

With the above general solution we find,

$$\gamma(0) = 10.667 \approx 32/3,$$
$$\gamma(1) = 9.3335 \approx 28/3,$$
$$\gamma(2) = 6.6668 \approx 20/3$$

which are in agreement with the results presented on pg. 97 by Davis and Brockwell.

First Method:

The general solution is again utilized in the first method for finding $\gamma(t)$, albeit it is applied to the ψ_i coefficients. Noting the second relation in equation 2 can be put in the form

$$\phi(\mathbf{B})\psi_j = \psi_j - \psi_{j-1} + \frac{1}{4}\psi_{t-2} = 0 \qquad\qquad j \ge 2$$

This gives a form suitable for the applicatoin of the general solution. The initial conditions are the first (upper) values on the right side of equation 2. As in the second method we solve these coefficients with use of the function $solve\_\psi\_ARMA(\theta, \phi)$ contained in $ACF\_methods.py$ in the $funcs\_$ folder.

Here the general solution is of $\psi(t)$ is of the same form as equation 6, but we don't need to add solutions and this has the effect of simplifying things appreciably. Furthermore our initial conditions are simply $[\psi_0, \psi_1] = [1, 2]$. We find,

$$\begin{bmatrix} 2^{-0} & 0 \\ 2^{-1} & 2^{-1} \end{bmatrix} \begin{bmatrix} \beta_{10} \\ \beta_{11} \end{bmatrix} = \begin{bmatrix} \psi_0 \\ \psi_1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

with the results $\beta_{10} = 1, \beta_{11} = 3$. The general solution for ψ_t is then,

$$\psi_t = (1 + 3t)2^{-t} \tag{10}$$

In contrast to the second method which found directly a general solution to γ_t, we now must caluclate γ_t from the general solution of ψ_t. This is accomplished via the function $first\_meth\_autocovariance$ in $ACF\_methods.py$ contained in the $funcs\_$ folder. This function is essentially equation 3.3.1 in [1] which is,

$$\gamma_t = \sigma^2 \sum_{j=0}^{\infty} \psi_t \psi_{j+|t|} \tag{11}$$

It is an infinite sum so hopefully it converges! If not we must have a problem relating to something called unit roots – a topic we'll not get into here. The results are,

$$\begin{aligned} \gamma(0) &= 10.667 \approx 32/3, \\ \gamma(1) &= 9.3334 \approx 28/3, \\ \gamma(2) &= 6.6668 \approx 20/3 \end{aligned}$$

in excellent agreement with the results found on pg. 97 of [1].

Third Method:

The third method utilizes equation 7, albeit it does not utilize the general solution dealt with thus far. Instead we make use of the relation $\gamma(-t) = \gamma(t)$ to write,

$$\begin{aligned} \gamma(0) - \phi_1\gamma(1) - \phi_2\gamma(2) &= 3, \\ \gamma(1) - \phi_1\gamma(0) - \phi_2\gamma(1) &= 1, \\ \gamma(2) - \phi_1\gamma(1) - \phi_2\gamma(0) &= 0, \end{aligned} \tag{12}$$

where the same initial condition values as in the second method have been used, or as far as the coding goes, we again use $solve\_\psi\_ARMA(\theta, \phi)$ to find the ψ_j coefficients then construct a column vector of initial conditions according to the first (upper) value on the right side of equation 7. We have three equations with three unknows, so it is at first glance a simple matter of solving them. But in order to find a computational method which is suitably general to be applied to other problems requires a little more thinking.

Combining like terms in equation 12 gives,

$$M = \begin{bmatrix} 1 & -\phi_1 & -\phi_2 \\ -\phi_1 & (1 - \phi_2) & 0 \\ -\phi_2 & -\phi_1 & 1 \end{bmatrix} \begin{bmatrix} \gamma(0) \\ \gamma(1) \\ \gamma(2) \end{bmatrix} = \begin{bmatrix} 3 \\ 1 \\ 0 \end{bmatrix}$$

We find,

$$\gamma(0) = 10.667 \approx 32/3,$$
$$\gamma(1) = 9.3333 \approx 28/3,$$
$$\gamma(2) = 6.6667 \approx 20/3$$

Which are again in agreement with the results of [1].

For a general method of constructing the appropriate matrix we can start with

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

then enumerate through $i \in 1, ..., p$ initial time values and, for every i value we then enumerate through $j \in 2, ..., p$ time values and subtract a value of ϕ_j from the matrix element $M_{i,|i-j|}$. This is a sufficiently generalized method to be applied to other $ARMA[p, q]$ processes. We will now test all three methods on the $ARMA[2, 3]$ process.

$ARMA[2, 3]$ process: Finding $\gamma(t)$ and Using the Innovations Algorithm to predict $\hat{x}_{n+1}$

Everything presented thus far was in a sense a pre-requisite to using the Innovations algorithm to find $\hat{x}_{n+1}$ – the projection of x_{n+1} onto the basis $\{x_1, x_2, ..., x_n\}$. I will neither derive or present the results of the Innovations algorithm as I feel if one can understand the results up to this point then an understanding of the innovations algorithm as well as the Durbin-Livinson Algorithm are a relatively trivial extension. These derivations can be found in chapter five of [1]. The results of the innovations algorithm are however presented, and the reader may compare them to those presented on page 172 of [1].

Here we will test the algorithms derived thus far on example 5.3.4 of Brockwell and Davis's book. If the ACF (or $\gamma(t)$) values are close to their results then we have reason to suspect that we have a reliable method of computing the ACF values of a variety of ARMA models.

The $ARMA[2, 3]$ process in question is

$$x_t - x_{t-1} + 0.24x_{t-2} = z_t + 0.4z_{t-1} + 0.2z_{t-2} + 0.1z_{t-3}$$

The polynomial $\phi(\mathbf{B}) = 1 - \mathbf{B} + 0.24\mathbf{B}^2$ has two roots and this implies $k = 2$. The value of the roots are $\zeta_1 = 2.5, \zeta_1 \approx 1.66667$ each with multiplicity 1 ($r_1 = r_2 = 1$). When this polynomial acts on $\gamma(t)$ we get $\gamma(t) - \gamma(t-1) - 0.024\gamma(t-2)$. The general solution of $\gamma(t)$ is,

$$\gamma(t) = \beta_{10} 2.5^{-t} + \beta_{11} 1.667^{-1} \qquad t \geq max(p, q+1) - p = 2 \tag{13}$$

The θ coefficient values are identified as $\theta_0, \theta_1, \theta_2, \theta_3 = 1, 0.4, 0.2, 0.1$. The results for $\gamma(0), \gamma(1), \gamma(2)$ presented in [1] are found using the second method. These values are found to be,

$$\gamma(0) = 7.17133$$
$$\gamma(1) = 6.44139$$
$$\gamma(2) = 5.06027$$

We now present the computational results of all three methods contained in *ACF_methods.py* contained in the *funcs_* folder. These results are to be compared to the above values.

$\gamma(t)$ values using the first method:

Using $\phi_0, \phi_1, \phi_2 = 1, 1, -0.24$ and $\theta_0, \theta_1, \theta_2, \theta_3 = 1, 0.4, 0.2, 0.1$ in the function $first_method$ contained in $ACF_methods.py$ we find,

$$\gamma(0) = 7.1711$$
$$\gamma(1) = 6.4411$$
$$\gamma(2) = 5.0599$$

These results are satisfactorily close to our target values.

$\gamma(t)$ values using the second method:

For the second method if we use the time values $t = 2, 3$ to calculate the initial conditions. Inserting the same coefficients as in the first method into the function *second_method* we get,

$$\gamma(0) = -694451.389$$
$$\gamma(1) = -416587.5159$$
$$\gamma(2) = -249902.529$$

Which are waaay off! But if we use the time values $t = 0, 1$ to calculate the initial conditions we get,

$$\gamma(0) = 7.419$$
$$\gamma(1) = 6.6412$$
$$\gamma(2) = 4.8602$$

These results are still off by as much as 0.2. Being as it was the second method which Brockwell and Davis used to calculate their results, this suggests something is off with the coding of the second method. According to the time constraints in equation 7 the initial time values should be taken as $t = 2, 3$. Obviously there are some bugs to work out here, so we do not consider the coding of the second method to be reliable for the general $ARMA[p, q]$ process as it stands at the time of posting this paper. It did however produce satisfactory results for the $ARMA[2, 1]$ process which is suggestive we are on the right track.

$\gamma(t)$ values using the third method:

For the third method we get,

$$\gamma(0) = 7.1713$$
$$\gamma(1) = 6.4414$$
$$\gamma(2) = 5.0603$$

These results are satisfactorily close to our target.

2 Innovation Algorithm Results:

The results for the innovations algorithm using the first method to calculate $\gamma(t)$ values are as follows,

n	x_{n+1}	r_n	θ_{n1}	θ_{n2}	θ_{n3}	$\hat{x}_{n+1}$
0.0	1.704	7.1711	0.0000	0.0000	0.0000	0.000
1.0	0.527	1.3856	0.8982	0.0000	0.0000	1.531
2.0	1.041	1.0057	1.3685	0.7056	0.0000	−0.171
3.0	0.942	1.0019	0.4008	0.1806	0.0139	1.243
4.0	0.555	1.0016	0.3998	0.2020	0.0722	0.744
5.0	−1.002	1.0005	0.3992	0.1996	0.0994	0.314
6.0	−0.585	1.0000	0.4000	0.1997	0.0998	−1.729
7.0	0.010	1.0000	0.4000	0.2000	0.0998	−0.169
8.0	−0.638	1.0000	0.4000	0.2000	0.0999	0.319
9.0	0.525	1.0000	0.4000	0.2000	0.1000	−0.873
10.0	0.000	1.0000	0.4000	0.2000	0.1000	1.064

These values are all very close (¡0.005 difference in most cases) to those presented on page 172 of [1]. The only value we really care about is is $\hat{x}_{10}$ which is the projection of x_{10} onto the basis $x_1, x_2, ..., x_9$. Prior to rounding $\hat{x}_{10}$ was calculated to be $\hat{x}_{10} = 1.063792$. This value compares favorably to $\hat{x}_{10} = 1.0638$ given by Brockwell and Davis.

Using the third method we get $\hat{x}_{10} = 1.06513$.

Using the second method with the [wrong?] initial time values $t = 0, 1$ we get $\hat{x}_{10} = 1.081$ which isn't too far off, but if we don't know *why* the coding of the second method is not giving satisfactory results under the correct initial time values then we should think twice about trusting this estimate just because it happened to be close.

References

[1] Peter J. Brockwell, Richard A. Davis *Time Series: Theory and Methods*. Springer Verlag, New York Inc., 1987