Eight Functions for Verification of "bg_residue" and "bg_residuez"

By Bernie Grung (9/8/2011)

The directory "coar39" contains eight functions for verifying "bg_residue" and "bg_residuez" which are corrected versions of "residue" and "residuez". We will use Mathcad equations in most of this note since these equations are easier to read in a Microsoft Word document. However, the Octave results are identical to those obtained using Mathcad, Matlab or C++.

A. Verification Data Using Function "bg_Sec_A"

For the phase-locked loop (PLL) circuit, the open-loop transfer function is

$$h(s) := \frac{s + z_1}{(s - p_1)(s - p_2)s^2} \qquad \text{A-1}$$

where z_1 is a zero and p_1 and p_2 are poles. If we define the numerator as **num(s)=s+z$_1$** and the denominator as **den(s)=(s-p$_1$)(s-p$_2$)s^2**, then **num(s)=h(s)*den(s).** The transfer function **h(s)** can also be rewritten as

$$h(s) := \frac{r_1}{s - p_1} + \frac{r_2}{s - p_2} + \frac{r_3}{s} + \frac{r_4}{s^2} \qquad \text{A-2}$$

where r_1, r_2, r_3, and r_4 are residues. Multiplying both sides of this equation by **den(s)** gives

$$num(s) := r_1 \cdot (s - p_2) \cdot s^2 + r_2 \cdot (s - p_1) \cdot s^2 + r_3 \cdot (s - p_1) \cdot (s - p_2) \cdot s + r_4 (s - p_1)(s - p_2) \qquad \text{A-3}$$

After expansion of terms, this becomes

$$\begin{aligned} num(s) := {} & \left(r_1 \cdot s^3 - r_1 \cdot p_2 \cdot s^2\right) \ldots \\ & + \left(r_2 \cdot s^3 - r_2 p_1 s^2\right) \ldots \\ & + \left[r_3 s^3 - \left[r_3 (p_1 + p_2)\right] s^2 + r_3 p_1 p_2 s\right] \ldots \\ & + \left[r_4 s^2 - r_4 (p_1 + p_2) s + r_4 p_1 p_2\right] \end{aligned} \qquad \text{A-4}$$

We can define a row vector **b** as containing the polynomial coefficients of **num(s)** so that **b=[0,0,1,z$_1$]** and a second row vector **a** as containing the polynomial coefficients of **den(s)** so that **a=[1,-(p$_1$+p$_2$),p$_1$p$_2$,0,0]**. In addition, we can define the column vector **r** as **[r$_1$,r$_2$,r$_3$,r$_4$]^T** where **T** is denotes the transpose. With these definitions and A-4, we can define a matrix **m**

$$m := \begin{bmatrix} 1 & 1 & 1 & 0 \\ -p_2 & -p_1 & -(p_1 + p_2) & 1 \\ 0 & 0 & p_1 p_2 & -(p_1 + p_2) \\ 0 & 0 & 0 & p_1 p_2 \end{bmatrix} \qquad \text{A-5}$$

such that

$$b := (m \cdot r)^T \qquad \text{A-6}$$

Note that **b** is defined as a row vector and **m*r** is a column vector so that the transpose **T** must be taken. Solving A-6 for **r** gives

$$\mathbf{r} := \mathbf{m}^{-1} \cdot \mathbf{b}^{T} \qquad \text{A-7}$$

Figure A-1 summarizes these equations and presents numerical values for a typical continuous-time phase-locked loop. Specifically, the zero $\mathbf{z_1}$ is at 7.04 MHz, the two poles $\mathbf{p_1}$ and $\mathbf{p_2}$ are at 3.14 GHz and 0.5 GHz, respectively.

$$\mathbf{b} := \begin{pmatrix} 0 & 0 & 1 & z_1 \end{pmatrix} = \begin{pmatrix} 0.0000000000 \times 10^{0} & 0.0000000000 \times 10^{0} & 1.0000000000 \times 10^{0} & 7.0372976777 \times 10^{6} \end{pmatrix}$$

$$\mathbf{a} := \begin{bmatrix} 1 & -(p_1 + p_2) & p_1 p_2 & 0 & 0 \end{bmatrix}$$

$$\mathbf{a} = \begin{pmatrix} 1.0000000000 \times 10^{0} & 3.6412407887 \times 10^{9} & 1.5696909107 \times 10^{18} & 0.0000000000 \times 10^{0} & 0.0000000000 \times 10^{0} \end{pmatrix}$$

$$\mathbf{p} := \begin{pmatrix} p_1 \\ p_2 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} -3.1415926536 \times 10^{9} \\ -4.9964813512 \times 10^{8} \\ 0.0000000000 \times 10^{0} \\ 0.0000000000 \times 10^{0} \end{pmatrix} \qquad \mathbf{m} := \begin{bmatrix} 1 & 1 & 1 & 0 \\ -p_2 & -p_1 & -(p_1 + p_2) & 1 \\ 0 & 0 & p_1 p_2 & -(p_1 + p_2) \\ 0 & 0 & 0 & p_1 p_2 \end{bmatrix}$$

$$\mathbf{m} = \begin{pmatrix} 1.0000000000 \times 10^{0} & 1.0000000000 \times 10^{0} & 1.0000000000 \times 10^{0} & 0.0000000000 \times 10^{0} \\ 4.9964813512 \times 10^{8} & 3.1415926536 \times 10^{9} & 3.6412407887 \times 10^{9} & 1.0000000000 \times 10^{0} \\ 0.0000000000 \times 10^{0} & 0.0000000000 \times 10^{0} & 1.5696909107 \times 10^{18} & 3.6412407887 \times 10^{9} \\ 0.0000000000 \times 10^{0} & 0.0000000000 \times 10^{0} & 0.0000000000 \times 10^{0} & 1.5696909107 \times 10^{18} \end{pmatrix}$$

$$\mathbf{r} := \mathbf{m}^{-1} \cdot \mathbf{b}^{T} = \begin{pmatrix} 1.2021329617 \times 10^{-19} \\ -7.4688154330 \times 10^{-19} \\ 6.2666824713 \times 10^{-19} \\ 4.4832378335 \times 10^{-12} \end{pmatrix}$$

Figure A-1: Continous-Time Equations and Corresponding Numerical Values for a Typical Phase-Locked Loop Circuit.

B. **Verification Data Using Function "bg_Sec_B"**

Using the impulse invariance method, A-2 (a continuous-time equation in **s**) can be transformed to the following discrete-time equation in **z**

$$\mathbf{h(z)} := \frac{\mathbf{r_1}}{1 - \mathbf{p_1} z^{-1}} + \frac{\mathbf{r_2}}{1 - \mathbf{p_2} z^{-1}} + \frac{\mathbf{r_3}}{1 - z^{-1}} + \frac{\mathbf{r_4}}{\left(1 - z^{-1}\right)^2} \qquad \text{B-1}$$

where the new variables **p** and **r** are defined below. Each discrete-time pole $\mathbf{p_i}$ is converted from the continuous time pole $\mathbf{A\_p_i}$ by using $\mathbf{p_i = exp(T*A\_p_i)}$ where **T** is the sampling period. Note that in this section "**A_**" is prefixed to all continuous-time variables calculated above in Section A. Thus,

$$\mathbf{p} := \begin{pmatrix} \exp(\mathbf{T} \cdot \mathbf{A\_p_1}) \\ \exp(\mathbf{T} \cdot \mathbf{A\_p_2}) \\ 1 \\ 1 \end{pmatrix} \qquad \text{B-2}$$

The first two residues $\mathbf{r_1}$ and $\mathbf{r_2}$ are calculated as $\mathbf{r_1 = A\_r_1}$ and $\mathbf{r_2 = A\_r_2}$. To find the remaining two residues, let's start with

$$\mathbf{h(s)} := \frac{\mathbf{r_3}}{\mathbf{s}} + \frac{\mathbf{r_4}}{\mathbf{s}^2} \qquad \text{B-3}$$

Using the impulse invariance method, this becomes

$$\mathbf{h(z)} := \frac{\mathbf{r_3}}{1 - z^{-1}} + \frac{\mathbf{T} \cdot \mathbf{r_4} \cdot z^{-1}}{\left(1 - z^{-1}\right)^2} \qquad \text{B-4}$$

where **T** is the sampling period. This can be rewritten as

$$\mathbf{h(z)} := \frac{\mathbf{r_3} - \mathbf{T} \cdot \mathbf{r_4}}{1 - z^{-1}} + \frac{\mathbf{T} \cdot \mathbf{r_4}}{\left(1 - z^{-1}\right)^2} \qquad \text{B-5}$$

so it follows that the discrete-time **r** is given by

$$\mathbf{r} := \begin{pmatrix} \mathbf{A\_r_1} \\ \mathbf{A\_r_2} \\ \mathbf{A\_r_3} - \mathbf{A\_r_4} \cdot \mathbf{T} \\ \mathbf{A\_r_4} \cdot \mathbf{T} \end{pmatrix} \qquad \text{B-6}$$

where we have again use a prefix to indicate values from Section A. The denominator **den(z)** of B-1 is given by

$$\mathbf{den(z)} := \left(1 - \mathbf{p_1} z^{-1}\right)\left(1 - \mathbf{p_2} z^{-1}\right)\left(1 - z^{-1}\right)^2 \qquad \text{B-7}$$

Expanding the factors gives

$$\begin{aligned} \mathbf{den(z)} := 1 - \left(2 + \mathbf{p_1} + \mathbf{p_2}\right) z^{-1} + \left[1 + \mathbf{p_1}\mathbf{p_2} + 2\left(\mathbf{p_1} + \mathbf{p_2}\right)\right] z^{-2} \ldots \\ + -\left(2\,\mathbf{p_1}\mathbf{p_2} + \mathbf{p_1} + \mathbf{p_2}\right) z^{-3} + \mathbf{p_1}\mathbf{p_2} z^{-4} \end{aligned} \qquad \text{B-8}$$

Defining the row vector **a** as the polynomial coefficients of **den(z)**, we find

$$\mathbf{a} := \begin{bmatrix} 1 & -\left(2 + \mathbf{p_1} + \mathbf{p_2}\right) & \left[1 + \mathbf{p_1}\mathbf{p_2} + 2\left(\mathbf{p_1} + \mathbf{p_2}\right)\right] & -\left(2\,\mathbf{p_1}\mathbf{p_2} + \mathbf{p_1} + \mathbf{p_2}\right) & \mathbf{p_1}\mathbf{p_2} \end{bmatrix} \qquad \text{B-9}$$

The numerator **num(z)** of B-1 is

$$\begin{aligned}\mathbf{num}(z) := {} & r_1\left(1 - p_2 z^{-1}\right)\left(1 - z^{-1}\right)^2 \ \ldots \\ & + r_2\cdot\left(1 - p_1 z^{-1}\right)\left(1 - z^{-1}\right)^2 \ \ldots \\ & + r_3\cdot\left(1 - p_1 z^{-1}\right)\left(1 - p_2 z^{-1}\right)\left(1 - z^{-1}\right) \ .. \\ & + r_4\cdot\left(1 - p_1 z^{-1}\right)\left(1 - p_2 z^{-1}\right)\end{aligned} \qquad \text{B-10}$$

since **num(z)=den(z)*h(z)**. Expanding the various factors gives

$$\begin{aligned}\mathbf{num}(z) := {} & r_1 - r_1(2 + p_2) z^{-1} + r_1\cdot(1 + 2p_2)\cdot z^{-2} - r_1 p_2 z^{-3} \ \ldots \\ & + r_2 - r_2\cdot(2 + p_1)\cdot z^{-1} + r_2\cdot(1 + 2p_2)\cdot z^{-2} - r_2\cdot p_1\cdot z^{-3} \ \ldots \\ & + r_3 - r_3\cdot(p_1 + p_2 + 1)\cdot z^{-1} + r_3\cdot(p_1 + p_2 + p_1 p_2)\cdot z^{-2} - r_3\cdot p_1 p_2\cdot z^{-3} \ .. \\ & + r_4 - r_4\cdot(p_1 + p_2)\cdot z^{-1} + r_4\cdot p_1 p_2\cdot z^{-2}\end{aligned} \qquad \text{B-11}$$

We can define a row vector **b** as containing the polynomial coefficients of **num(z)** and a column vector **r** such that

$$\mathbf{b} := (\mathbf{m}\cdot\mathbf{r})^{\mathrm{T}} \qquad \text{B-12}$$

where **m** is defined as

$$\mathbf{m} := \begin{bmatrix} 1 & 1 & 1 & 1 \\ -(2 + p_2) & -(2 + p_1) & -(p_1 + p_2 + 1) & -(p_1 + p_2) \\ (1 + 2\cdot p_2) & (1 + 2\cdot p_1) & (p_1 + p_2 + p_1 p_2) & p_1 p_2 \\ -p_2 & -p_1 & -p_1 p_2 & 0 \end{bmatrix} \qquad \text{B-13}$$

These equations are summarized in Figure B-1, along with numerical values for a typical phase-locked loop.

$$\mathbf{p} := \begin{pmatrix} \exp(T\cdot A\_p_1) \\ \exp(T\cdot A\_p_2) \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 1.0137701213 \times 10^{-13} \\ 8.5780070775 \times 10^{-3} \\ 1.0000000000 \times 10^{0} \\ 1.0000000000 \times 10^{0} \end{pmatrix} \qquad \mathbf{r} := \begin{pmatrix} A\_r_1 \\ A\_r_2 \\ A\_r_3 - A\_r_4\cdot T \\ A\_r_4\cdot T \end{pmatrix} = \begin{pmatrix} 1.2021329617 \times 10^{-19} \\ -7.4688154330 \times 10^{-19} \\ 5.8397074396 \times 10^{-19} \\ 4.2697503176 \times 10^{-20} \end{pmatrix}$$

$$\mathbf{a} := \begin{bmatrix} 1 & -(2 + p_1 + p_2) & [1 + p_1 p_2 + 2(p_1 + p_2)] & -(2 p_1 p_2 + p_1 + p_2) & p_1 p_2 \end{bmatrix}$$

$$\mathbf{a} = \begin{pmatrix} 1.0000000000 \times 10^{0} & -2.0085780071 \times 10^{0} & 1.0171560142 \times 10^{0} & -8.5780070776 \times 10^{-3} & 8.6961272755 \times 10^{-16} \end{pmatrix}$$

$$\mathbf{m} := \begin{bmatrix} 1 & 1 & 1 & 1 \\ -(2 + p_2) & -(2 + p_1) & -(p_1 + p_2 + 1) & -(p_1 + p_2) \\ (1 + 2\cdot p_2) & (1 + 2\cdot p_1) & (p_1 + p_2 + p_1 p_2) & p_1 p_2 \\ -p_2 & -p_1 & -p_1 p_2 & 0 \end{bmatrix}$$

$$\mathbf{m} = \begin{pmatrix} 1.0000000000 \times 10^{0} & 1.0000000000 \times 10^{0} & 1.0000000000 \times 10^{0} & 1.0000000000 \times 10^{0} \\ -2.0085780071 \times 10^{0} & -2.0000000000 \times 10^{0} & -1.0085780071 \times 10^{0} & -8.5780070776 \times 10^{-3} \\ 1.0171560142 \times 10^{0} & 1.0000000000 \times 10^{0} & 8.5780070776 \times 10^{-3} & 8.6961272755 \times 10^{-16} \\ -8.5780070775 \times 10^{-3} & -1.0137701213 \times 10^{-13} & -8.6961272755 \times 10^{-16} & 0.0000000000 \times 10^{0} \end{pmatrix}$$

$$\mathbf{b} := (\mathbf{m}\cdot\mathbf{r})^{\mathrm{T}} = \begin{pmatrix} 0.0000000000 \times 10^{0} & 6.6295899514 \times 10^{-19} & -6.1959656095 \times 10^{-19} & -1.0311905052 \times 10^{-21} \end{pmatrix}$$

Figure B-1 Discrete-Time Equations and Corresponding Numerical Values for a Typical Phase-Locked Loop Circuit.

C. Verification Data Using Function "bg_Sec_C"

The previous problem can be solved using a different method, one that is suggested by the original Octave function "residuez". First, we rewrite B-1 as

$$H(S) := \frac{R_1}{S - P_1} + \frac{R_2}{S - P_2} + \frac{R_3}{S - 1} + \frac{R_4}{(S - 1)^2} \qquad \text{C-1}$$

where **S** is defined as z^{-1} and **H(S)** is defined as $h(z^{-1})$. Note that we are using capital letters here to indicate that the above equation is not an actual **s**-transform equation but a **z**-transform equation using new notation. To derive **R** and **P**, let's start with the first term in B-1 which is $r_1/(1-p_1*z^{-1})$. Dividing numerator and denominator by $-p_1$ and replacing z^{-1} by **S** so that this term becomes $[r_1/(-p_1)]/[S-(1/p_1)]$. Comparing with the first term of C-1, we find that $P_1=1/p_1$ and $R_1=r_1/(-p_1)$. Similar results are derived for the next two terms. For the final term, we find that $P_4=1/p_4$ and $R_4=r_4/(-p_4)^2$. So far, we have been using **p** and **r** to have the same values as those calculated in Section B above. In the following, we will prepend "**B_**" to express explicitly this relationship. Thus, **R** and **P** are given by the next two equations:

$$P := \begin{pmatrix} \frac{1}{B\_p_1} & \frac{1}{B\_p_2} & \frac{1}{B\_p_3} & \frac{1}{B\_p_4} \end{pmatrix}^T \qquad \text{C-2}$$

and

$$R := \begin{bmatrix} \frac{B\_r_1}{(-p_1)^1} & \frac{B\_r_2}{(-B\_p_2)^1} & \frac{B\_r_3}{(-B\_p_3)^1} & \frac{B\_r_4}{(-B\_p_4)^2} \end{bmatrix}^T \qquad \text{C-3}$$

The denominator **DEN(S)** of C-1 is given by

$$DEN(S) := (S - P_1)\cdot(S - P_2)\cdot(S - 1)^2 \qquad \text{C-4}$$

Expanding the terms gives

$$\begin{aligned} DEN(S) := {} & S^4 - (P_1 + P_2 + 2)\cdot S^3 \ldots \\ & + \left[P_1\cdot P_2 + 2(P_1 + P_2) + 1\right]\cdot S^2 - (P_1 + P_2 + 2P_1P_2)\cdot S + P_1P_2 \end{aligned} \qquad \text{C-5}$$

We can define a row vector **A** that contains the coefficients of **DEN(S)** as

$$A := \begin{bmatrix} 1 & -(P_1 + P_2 + 2) & \left[P_1\cdot P_2 + 2(P_1 + P_2) + 1\right] & -(P_1 + P_2 + 2P_1P_2) & P_1P_2 \end{bmatrix} \qquad \text{C-6}$$

The numerator **NUM(S)** of C-1 is given by

$$\begin{aligned} NUM(S) := {} & R_1(S - P_2)(S - 1)^2 \ldots \\ & + R_2(S - P_1)(S - 1)^2 \ldots \\ & + R_3(S - P_1)(S - P_2)(S - 1) \ldots \\ & + R_4(S - P_1)(S - P_2) \end{aligned} \qquad \text{C-7}$$

since **NUM(S)=DEN(S)*H(S).** Expanding terms gives

$$\begin{aligned} NUM(S) := {} & R_1S^3 - R_1(2 + P_2)S^2 + R_1(1 + 2P_2)S - R_1P_2 \ldots \\ & + R_2S^3 - R_2(2 + P_1)S^2 + R_2(1 + 2P_1)S - R_2P_1 \ldots \\ & + R_3S^3 - R_3(P_1 + P_2 + 1)S^2 + R_3(P_1P_2 + P_1 + P_2)S - R_3P_1P_2 \ldots \\ & + R_4S^2 - R_4(P_1 + P_2)\cdot S^1 + R_4P_1P_2 \end{aligned} \qquad \text{C-8}$$

We can define a row vector **B** as containing the polynomial coefficients of **NUM(S)** so that C-8 becomes

$$B := \frac{(M\cdot R)^T}{P_1P_2} \qquad \text{C-9}$$

where

$$\mathbf{M} := \begin{bmatrix} 1 & 1 & 1 & 0 \\ -(2+P_2) & -(2+P_1) & -(P_1+P_2+1) & 1 \\ (1+2P_2) & (1+2\cdot P_1) & (P_1+P_2+P_1P_2) & -(P_1+P_2) \\ -P_2 & -P_1 & -P_1P_2 & P_1P_2 \end{bmatrix} \qquad \text{C-10}$$

The equations in this section are summarized in Figure C-1 along with the corresponding values for a typical example of a phase-locked loop circuit.

$$\mathbf{P} := \begin{pmatrix} \frac{1}{B\_p_1} \\ \frac{1}{B\_p_2} \\ \frac{1}{B\_p_3} \\ \frac{1}{B\_p_4} \end{pmatrix} = \begin{pmatrix} 9.8641691937 \times 10^{12} \\ 1.1657719456 \times 10^{2} \\ 1.0000000000 \times 10^{0} \\ 1.0000000000 \times 10^{0} \end{pmatrix} \qquad \mathbf{R} := \begin{bmatrix} \frac{B\_r_1}{(-p_1)^1} \\ \frac{B\_r_2}{(-B\_p_2)^1} \\ \frac{B\_r_3}{(-B\_p_3)^1} \\ \frac{B\_r_4}{(-B\_p_4)^2} \end{bmatrix} = \begin{pmatrix} -1.1858042927 \times 10^{-6} \\ 8.7069354985 \times 10^{-17} \\ -6.2666824713 \times 10^{-19} \\ 4.2697503176 \times 10^{-20} \end{pmatrix}$$

$$\mathbf{A} := \begin{bmatrix} 1 & -(P_1+P_2+2) & [P_1\cdot P_2+2(P_1+P_2)+1] & -(P_1+P_2+2P_1P_2) & P_1P_2 \end{bmatrix}$$

$$\mathbf{A} = \begin{pmatrix} 1.0000000000 \times 10^{0} & -9.8641691938 \times 10^{12} & 1.1696655096 \times 10^{15} & -2.3097385117 \times 10^{15} & 1.1499371712 \times 10^{15} \end{pmatrix}$$

$$\mathbf{M} := \begin{bmatrix} 1 & 1 & 1 & 0 \\ -(2+P_2) & -(2+P_1) & -(P_1+P_2+1) & 1 \\ (1+2P_2) & (1+2\cdot P_1) & (P_1+P_2+P_1P_2) & -(P_1+P_2) \\ -P_2 & -P_1 & -P_1P_2 & P_1P_2 \end{bmatrix}$$

$$\mathbf{M} = \begin{pmatrix} 1.0000000000 \times 10^{0} & 1.0000000000 \times 10^{0} & 1.0000000000 \times 10^{0} & 0.0000000000 \times 10^{0} \\ -1.1857719456 \times 10^{2} & -9.8641691937 \times 10^{12} & -9.8641691938 \times 10^{12} & 1.0000000000 \times 10^{0} \\ 2.3415438912 \times 10^{2} & 1.9728338387 \times 10^{13} & 1.1598013404 \times 10^{15} & -9.8641691938 \times 10^{12} \\ -1.1657719456 \times 10^{2} & -9.8641691937 \times 10^{12} & -1.1499371712 \times 10^{15} & 1.1499371712 \times 10^{15} \end{pmatrix}$$

$$\mathbf{B} := \frac{(\mathbf{M}\cdot\mathbf{R})^{\mathrm{T}}}{P_1P_2} = \begin{pmatrix} -1.0311905052 \times 10^{-21} & -6.1959656095 \times 10^{-19} & 6.6295899514 \times 10^{-19} & 0.0000000000 \times 10^{0} \end{pmatrix}$$

Figure C-1 Equations and Numerical Results for a Typical Phase-Locked Loop.

D. **“bg_residuez” Verification Using Function “bg_Sec_D”**

So far, we have not employed any high-level function similar to “residue” and “residuez” in Mathcad and Octave. Now, we will use new high level functions, specifically, “bg_residue” and “bg_residuez”. The prefix “bg_” is used here to indicate that they are corrected versions of the original Octave functions.

As indicated in Figure D-1, we start with the inputs **a** and **b** which are calculated in Section B. Then, we calculate **r, p, k** and **e** using the command “[**r,p,k,e**] =bg_residuez (**b,a**)” where “bg_residue(**b,a**)” is called from this function. The numerical results are identical to those calculated above in Section B and are summarized in Figure D-1.

```
Octave program "coar39.m" using "bg_Sec_D.m":
------------------------------------------------------
Discrete-time calculation of r,p,k,and m from a and b,
using [r,p,k,m] = bg_residuez(b,a) The output file
is "test/Sec_D.log" and identical results are obtained
using Matlab and C++.
------------------------------------------------------
Inputs a and b are from Section B:
b=[-7.2222372915e-35,+6.6295899514e-19,-6.1959656095e-19,-1.0311905052e-21];
a=[+1.0000000000e+00,-2.0085780071e+00,+1.0171560142e+00,-8.5780070776e-03,+8.6961272755e-16];
----------------------------------- Into [r,p,k,e]=bg_residuez(b,a);
----------------------------------- Out of [r,p,k,e]=bg_residuez(b,a);
Final outputs are r, p, k and e:
r=\
[+1.2021329617e-19;
 -7.4688154330e-19;
 +5.8397074396e-19;
 +4.2697503177e-20];
p=\
[+1.0137701213e-13;
 +8.5780070775e-03;
 +1.0000000000e+00;
 +1.0000000000e+00];
k=\
[];
e=\
[+1.0000000000e+00;
 +1.0000000000e+00;
 +1.0000000000e+00;
 +2.0000000000e+00];
```

Figure D-1 Numerical Results Using “bg_Sec_D”

The function “bg_residue” employs a new method for calculating matrix **A,** a critical function. In “residue”, elements of **A** are calculated by first using the convolution function “conv” to form the polynomial **pden**. Then, the deconvolution function “deconv” is employed to provide the various columns of matrix **A**.

In function “bg_residue”, a new function “cal_**M**” finds matrix **A** by a faster and more efficient method. Before giving the details of the new function, we will, outline the general method using Mathcad code assuming that **p=[p_1,p_2,p_3,p_4]** and **e=[1,1,1,2]** so that p_3=p_4. The calculations are divided into three parts.

In Part 1, as given in Figure D-2, the initial values for **C_2(z)**, **C_3(z)**, and **C_4(z)** are calculated in sequence. Since **e_2=1, $C_2(z)=(1-p_1z^{-1})$**. Since **e_3=1, $C_3(z)=(1-p_1z^{-1})(1-p_2z^{-1})$**. Since **$e_4$=2, $C_4(z)=C_3(z)=(1-p_1z^{-1})(1-p_2z^{-1})$**, which is the final value for **C_4(z)**.

$$x(z) := \left(1 - p_1 \cdot z^{-1}\right) \qquad x(z) := x(z) \cdot \left(1 - p_2 \cdot z^{-1}\right) \qquad x(z) := x(z) \cdot \left(1 - p_3 \cdot z^{-1}\right)$$

$$C_2(z) := \begin{cases} x(z) & \text{if } e_2 = 1 \\ 1 & \text{otherwise} \end{cases} \qquad C_3(z) := \begin{cases} x(z) & \text{if } e_3 = 1 \\ C_2(z) & \text{otherwise} \end{cases} \qquad C_4(z) := \begin{cases} x(z) & \text{if } e_4 = 1 \\ C_3(z) & \text{otherwise} \end{cases}$$

Figure D-2 Part 1 Mathcad Calculations

In Part 2, the final values for vectors **$C_3(z)$**, **$C_2(z)$**, and **$C_1(z)$** are calculated in this sequence as shown in Figure D-3. **$C_3(z)=(1-p_1z^{-1})(1-p_2z^{-1})(1-p_4z^{-1})$**. **$C_2(z)=(1-p_1z^{-1})(1-p_3z^{-1})(1-p_4z^{-1})$**. **$C_1(z)=(1-p_2z^{-1})(1-p_3z^{-1})(1-p_4z^{-1})$** since **$e_1=1$.**

$$x(z) := \left(1 - p_4 \cdot z^{-1}\right) \qquad x(z) := x(z) \cdot \left(1 - p_3 \cdot z^{-1}\right) \qquad x(z) := x(z) \cdot \left(1 - p_2 \cdot z^{-1}\right)$$

$$C_3(z) := C_3(z) \cdot x(z) \qquad C_2(z) := C_2(z) \cdot x(z) \qquad C_1(z) := \begin{cases} x(z) & \text{if } e_1 = 1 \\ \dfrac{x(z)}{\left(1 - p_4 \cdot z^{-1}\right)} & \text{otherwise} \end{cases}$$

Figure D-3 Part 2 Mathcad Calculations

In Part 3, the matrix **M** is found using **$C_1(z)$**, **$C_2(z)$**, **$C_3(z)$** and **$C_4(z)$**. More details of this calculation are given in Figure D-4 using Octave.

```
function  M = cal_M(p,e)
    N = length(p); %% Define N as the number of poles
    C = cell(N,1); %% Define cell C to contain N vectors
    %%%%%%%%%%%%%%%%%%% PART 1: Calculation of the initial values for vectors C(2) through C(N)
    x=[1 -p(1)];
    if (e(2)==1)
        C{2}=x;
    else
        C{2}=[1];
    endif
    for i=3:N
        x=conv( x,[1 -p(i-1)]);
        if (e(i)==1)
            C{i}=x;
        else
            C{i}=C{i-(e(i)-1)};
        endif
    endfor
    %%%%%%%%%%%%%%%%%%%% PART 2: Calculation of the final values for vectors C(N) through C(1)
    x=[1 -p(N)];
    if (N>2)
        C{N-1}=conv(C{N-1},x);
    else
        C{N-1}=x;
    end
    for i=N-2:-1:2;
        x=conv(x,[1 -p(i+1)]);
        C{i}=conv(C{i},x);
    endfor
    if (N>2)
        x=conv(x,[1 -p(2)]);
        if (e(1)>1)
            x=deconv(x,[1 -p(N)]);
        end
        C{1}=x;
    end
    %%%%%%%%%%%%%%%%%%%%%%%%%% PART 3: Calculation of matrix M using vectors C(1) through C(N)
    M = zeros(N,N);
    for j=1:N
        x=C{j};
        len_x=length(x);
        offset=N-len_x;
        for i=1:offset
            M(i,j)=0;
        endfor
        for i=1:N-offset
            M(i+offset,j)=x(i);
        endfor
    endfor
endfunction
```

Figure D-4 The New Function "cal_M".

E. **“bg_residue” Verification Using Function “bg_Sec_E”**

Here, we use the function “bg_residue” to calculate the same results as given in Section C. We start with the **R, P, K** and **E** values given in Section C. Also, we need to employ a new input parameter “**MODE_Z**” to modify “bg_residue” for z transform analysis. Executing “**[B,A]=**bg_residue(**R,P,K,E,”MODE_Z”)”** gives the results listed in Figure E-1.

```
Octave program "coar39.m" using "bg_Sec_E.m":
-----------------------------------------------------
Discrete-time calculation of A and B from R ,P ,K and E
using [B,A] = bg_residue(R,P,K,E,"MODE_Z"). The output
file is "test/Sec_E.log" and identical results are
obtained using Matlab and C++.
-----------------------------------------------------
Inputs R, P, K and E are from Section C:
R=\
[-1.1858042927e-06;
 +8.7069354985e-17;
 -5.8397074396e-19;
 +4.2697503176e-20];
P=\
[+9.8641691937e+12;
 +1.1657719456e+02;
 +1.0000000000e+00;
 +1.0000000000e+00];
K=\
[];
E=\
[+1.0000000000e+00;
 +1.0000000000e+00;
 +1.0000000000e+00;
 +2.0000000000e+00];
----------------------------------- Into [B,A]=bg_residue(R,P,K,E,"MODE_Z");
---------------------------------- Out of [B,A]=bg_residue(R,P,K,E,"MODE_Z");
Final outputs are A and B:
A=[+1.0000000000e+00,-9.8641691938e+12,+1.1696655096e+15,-2.3097385117e+15,+1.1499371712e+15];
B=[-1.0311905052e-21,-6.1959656095e-19,+6.6295899514e-19,+0.0000000000e+00];
```

Figure E-1 Numerical Results Using “bg_Sec_E”.

F. “bg_residue” Verification Using Function “bg_Sec_F”

Here, we use the function “bg_residue” to calculate the same results as given in Section B. We start with the **r, p, k** and **e** values given in Section B. Also, we need to employ a new input parameter “**MODE_Z2**” to modify “bg_residue” for **z** transform analysis. Executing “**[b,a]=**bg_residue(**r,p,k,e,”MODE_Z2”)”** gives the results listed in Figure F-1.

```
Octave program "coar39.m" using "bg_Sec_F.m":
------------------------------------------------------------
Discrete-time calculation using:
       [b,a] = bg_residue(r,p,k,e,"MODE_Z2");
The output file is "test/Sec_F.log" and identical results
are obtained using Matlab and C++.
------------------------------------------------------------
Inputs r, p, k and e are from Section B:
r=\
[+1.2021329617e-19;
 -7.4688154330e-19;
 +5.8397074396e-19;
 +4.2697503176e-20];
p=\
[+1.0137701213e-13;
 +8.5780070775e-03;
 +1.0000000000e+00;
 +1.0000000000e+00];
k=\
[];
e=\
[+1.0000000000e+00;
 +1.0000000000e+00;
 +1.0000000000e+00;
 +2.0000000000e+00];
---------------------------- Into   [b,a]=bg_residue (r,p,k,e,"MODE_Z2");
---------------------------- Out of [b,a]=bg_residue (r,p,k,e,"MODE_Z2");
Final outputs are a and b:
b=[-4.2697503176e-20,+7.0602275781e-19,-6.1996282043e-19,-1.0311905052e-21];
a=[+1.0000000000e+00,-2.0085780071e+00,+1.0171560142e+00,-8.5780070776e-03,+8.6961272755e-16];
```

Figure F-1 Numerical Results Using “bg_Sec_F”

G. “bg_residue” Test Verification Using “bg_Sec_G”

We have added a new test function at the end of function “bg_residue”, as listed in Figure G-1. This function is based on the data given in Section A for the continuous time phase-locked loop.

```
%!test
%! z1=+7.0372976777e+06;
%! p1=-3.1415926536e+09;
%! p2=-4.9964813512e+08;
%! r1=-(1+z1/p1)/(1-p1/p2)/p2/p1  ;
%! r2=-(1+z1/p2)/(1-p2/p1)/p2/p1  ;
%! r3=(1+(p2+p1)/p2/p1*z1)/p2/p1  ;
%! r4=z1/p2/p1                    ;
%! r=[r1;r2;r3;r4];
%! p=[p1;p2;0;0];
%! k=[];
%! e=[1;1;1;2];
%! b=[1,z1];
%! a=[1,-(p1+p2),p1*p2,0,0];
%! [br, ar] = bg_residue (r, p, k, e);
%! assert ((abs (br - b) < 1e-8
%!        && abs (ar - a) < 1e-8));
```

Figure G-1 New Test Function for “bg_residue”

If we run “test residue” and “test bg_residue” in Octave, we get the results given in Figure G-2. We see that “residue” passes 4 of 4 tests and that “bg_residue” passes 5 of 5 tests where the above test is included only in “bg_residue”. If we add this test to “residue” the test will fail.

```
Starting test functions in "bg_residue.m".
Using "bg_Sec_G.m" to run "test    residue":  PASSES 4 out of 4 tests
Using "bg_Sec_G.m" to run "test bg_residue":  PASSES 5 out of 5 tests
Completed test functions in "bg_residue.m".
Starting details of a new test function.
```

Figure G-2 Test Results for “residue” and “bg_residue”

H. “bg_residuez” Test Verification Using “bg_Sec_H

We have added a new test function at the end of function “bg_residuez”, as listed in Figure H-1. This function is based on the data given in Section B for the discrete time phase-locked loop.

```
%!test
%! z1=+7.0372976777e+06;
%! p1=-3.1415926536e+09;
%! p2=-4.9964813512e+08;
%! r1=-(1+z1/p1)/(1-p1/p2)/p2/p1  ;
%! r2=-(1+z1/p2)/(1-p2/p1)/p2/p1  ;
%! r3=(1+(p2+p1)/p2/p1*z1)/p2/p1  ;
%! r4=z1/p2/p1                    ;
%! fi= 1.05e+08     ;
%! T =1/fi          ;
%! p1=exp(p1*T);
%! p2=exp(p2*T);
%! p3=1;
%! p4=1;
%! p =[p1;p2;p3;p4];
%! r4=r4*T;
%! r3=r3-r4;
%! r =[r1;r2;r3;r4];
%! m=[       1      ,     1      ,     1          ,     1          ;
%!        -(2+p2)   ,  -(2+p1)   , -(1+p1+p2)     ,  -(p1+p2)     ;
%!       +(1+2*p2) , +(1+2*p1) , (p1*p2+p1+p2) ,   +p1*p2      ;
%!          -p2    ,    -p1     , -p1*p2         ,     0         ];
%! a1=1.00;
%! a2=-(2+p1+p2);
%! a3=1+2*(p1+p2)+p1*p2;
%! a4=-(p1+p2+2*p1*p2);
%! a5=p2*p1;
%! a =[a1,a2,a3,a4,a5];
%! b=[m*r]';
%! k=[];
%! e=[1;1;1;2];
%! [rx,px,kx,ex] = bg_residuez(b,a);
%! assert ((abs (rx - r) < 1e-8
%!       && abs (px - p) < 1e-8
%!       && isempty (kx)
%!       && ex == e));
```

Figure H-1 New Test Function for “bg_residuez”

If we run “test residuez” and “test bg_residuez” in Octave, we get the results given in Figure H-2. We see that the “residuez” passes 9 of 9 tests and that “bg_residuez” passes 10 of 10 tests where the above test is included only in “bg_residuez” . If we add this test to “residuez” the test will fail.

```
Starting test functions in "bg_residuez.m".
Using "bg_Sec_H.m" to run "test    residuez": PASSES 9 out of 9 tests
Using "bg_Sec_H.m" to run "test bg_residuez": PASSES 10 out of 10 tests
Completed test functions in "bg_Sec_H.m".
Starting details of a new test function.
Completing details of a new test function.
```

Figure H-2 Test Results for “residuez” and “bg_residuez”