I/O Request Prioritization – An Analytical Analysis and Quantification

Introduction

A disk drive is considered as a delivery mechanism for persistent data storage that utilizes magnetic recording techniques. The design objectives are to provide adequate capacity, reliability, and performance at a minimal cost factor. A single disk drive represents a 3-dimensional space of recorded information. The surface of a disk platter provides 2 dimensions, whereas the stack of disk platters is considered as the third dimension. Adding additional disk platters increases the total capacity in the disk subsystem. This approach though increases the cost factor and causes some difficulties in regards to increasing the disk's *areal density* [1],[2],[7]. Increased vibrations in the spindle, susceptibility to external vibration, and internal disturbance resulting from turbulent airflow are all artifacts of increasing the number of platters in a given space. For decades, drive capacity has been increased by reducing the space between data tracks, measured in tracks per inch (TPI), and increasing the linear density of the bits along a track, measured in bits per inch (BPI). The product of these 2 terms is known as *areal density*, and is measured today in Gbits per square inch. A contemporary disk drive may have 30+ Gbits per square inch. The fact of the matter is that the challenge of designing head, media, and signal-processing systems to achieve higher areal density dominates the development cycle of any disk drive.

In the realm of the older I/O programming model, to locate specific data, the disk drive's logic requires the cylinder, the head, and the sector information. The cylinder specifies the track on which the data resides. Based on the layering technique used, the tracks underneath each other form a cylinder. The head information identifies the specific read-write head, and therefore the exact platter. At this point, the search is narrowed down to a single track on a single platter. Ultimately, the sector value represents the specific sector on the track, and the search is completed. Contemporary disk subsystems do not communicate in terms of cylinders, heads and sectors. Instead, modern disk drives map a unique block number over each cylinder/head/sector construct. Operating systems address the disk drives by utilizing these block numbers (logical block addressing). The culprit is that it is not guaranteed though that the physical mapping is actually sequential [4]. But the statement can be made that there is a rather high probability that a logical block *n* is physically adjacent to a logical block *n+1*. The existence of the discussed sequential layout is paramount to the host system while scheduling or reordering I/O requests. Contemporary disk drives utilize a technology called Zone Bit Recording (sometimes referred to as Zoned Constant Angular Velocity) to increase capacity. Incorporating the technology, cylinders are grouped into zones based on their distance from the center of the disk. Each zone is assigned a number of sectors per track [6]. The outer zones contain more sectors per track compared to the inner zones that are located closer to the spindle. With ZBR disks, the actual data transfer rate varies depending on the physical sector location. Given the fact that a disk drive spins at a constant rate, the outer zones that contain more sectors will transfer data at a higher rate than the inner zones that contain fewer sectors [5].

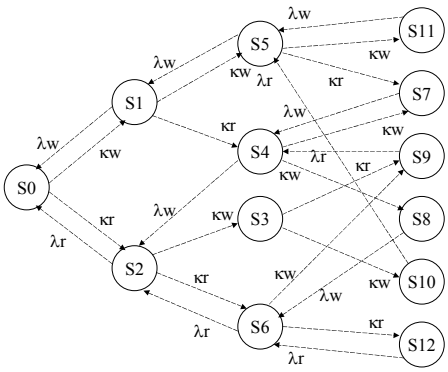
Further, present-day requirements for systemized, network-based storage of data encompass a range of users and applications that would have staggered the imagination 2 decades ago [5]. The meteoric rise of Internet data access became possible as a result of improved network capabilities, and then commenced to drive frenetic development of newer and more effective advances in those capabilities themselves. Nowadays, it is difficult to architect a scalable IO solution in which RAID systems are not deployed. The two-decade-old principles of large data storage and management have grown and evolved into new forms, yet the basic advantages of dependability, availability, and protection remain key factors in RAID technology's enduring value [4]. The basic function of a RAID structure is to provide a plurality of disk units for the purpose of massive data storage. The intention of the RAID technology is to take advantage of the large amount of space by storing important data in more secure (redundant) locations, and yet access the data as easily and quickly as on a *single disk drive* based system. With the increased popularity and availability of RAID technology came an explosion in available solutions, each suitable to different workload profiles. In addition, the evolution of the RAID

Dominique A. Heger (dom@fortuitous.com) – Fortuitous Technologies, Austin, TX

technology produced hugely improved storage capabilities. Essential to the efficient optimization of RAID technology is the prevention of a *bottleneck* scenario. One basic approach taken to mitigate the issue lies in distributing the burden of transfer among all the disks in the array, thus reducing the workload borne by any single disk drive. Nevertheless, the argument made in this study is that any RAID solution is still strongly impacted by the capabilities of the single disk drives that constitute the IO subsystem. Therefore, understanding the single disk drive capabilities is still paramount in a RAID environment. The (well-documented) performance delta between SATA and FC disks solutions underlines this claim [9]. Further, rather complex storage management requirements are significant, as the focus is on allowing efficient access to an entire data pool, by presenting applications with a single access point that is irrespective of the actual RAID array size.

1.0 IO Prioritizing Model

Figure 1: State Transition Diagram ($N = 3$)



The next few paragraphs introduce the proposed stochastic Markov model that incorporates the finite buffer constrains of an actual disk drive, and allows quantifying the I/O performance behavior based on workload scenarios that simulate concurrent *read()* and *write()* requests. The assumption made in this study is that *read()* requests have a higher priority than *write()* requests, a claim that is supported by the Linux 2.6 anticipatory (AS) and deadline I/O schedulers, respectively. The issue surrounding the various disk scheduling polices has been an area of extended discussions [3],[5]. The disk characteristics and the distribution of the disk I/O requests determine the performance of any scheduling policy. The mean service time for small random I/O requests equals to the summation of the seek time, disk rotational latency, transfer time, controller latency, and the queuing latency, respectively. It has to be pointed out that the queuing latency is not a fixed entity, as the queuing behavior is impacted by the set of IO operations that await service. Based on a geometrical probability argument, the average-seek distance for random I/O operations equals to *1/3* of the number of available disk cylinders ($d \approx C/3$). It has to be pointed out though that as the seek time characteristic *t(d)* ($1 \leq d \leq C-1$) is nonlinear, the mean seek time can not be expressed as being equal to *t(d)* [8]. The (conceptional) I/O model introduced in this study focuses on quantifying the single disk queue blocking probability, as well as the response time behavior. The study assumes a finite disk queue of size *N*. The single disk drive is modeled as a 2-priority M/M/1/N system. Arrival rates into the system are labeled as λw (write) and λr (read), respectively, whereas the service rates are expressed as μw (write) and μr (read). The *read()* requests have a higher (non-preemptive) priority than the *write()* requests. The state of the system is represented as a vector *iov = [w,r,sio]*, where *w* represents the number of write requests, *r* the number of read requests, and *sio* the state of the system (an integer value in a set *siov [0,1,2]*). The *sio* indicator represents either an empty system (*0*), a system servicing a *read()* request (*1*), or a system processing a *write()* request (*2*). To simplify the mathematical abstraction of the proposed approach, this study assumes that the queue size $N = 3$. The model though can easily be extended to simulate much larger queue sizes.

Table 1: Workload Behavior ($N = 3$)

State	Workload (see Vector *iov*)
S0	0 read, 0 write, 0 = system empty
S1	0 read, 1 write, 2 = processing write request
S2	1 read, 0 write, 1 = processing read request
S3	1 read, 1 write, 1 = processing read request
S4	1 read, 1 write, 2 = processing write request
S5	0 read, 2 write, 2 = processing write request
S6	2 read, 0 write, 1 = processing read request
S7	1 read, 2 write, 2 = processing write request
S8	2 read, 1 write, 2 = processing write request
S9	2 read, 1 write, 1 = processing read request
S10	1 read, 2 write, 1 = processing read request
S11	0 read, 3 write, 2 = processing write request
S12	3 read, 0 write, 1 = processing read request

Based on the State Transition Diagram in Figure 1 and the workload behavior expressed in vector *iov* (see Table 1), the following global

balance equations can be defined (in this scenario, *p0* correlates to *S0*, *p1* to *S1*, etc.).

$$p0 := \frac{p1 \cdot \mu w + p2 \cdot \mu r}{\lambda r + \lambda w} \quad p1 := \frac{p0 \cdot \lambda w + p5 \cdot \mu w}{\lambda r + \lambda w + \mu w}$$

$$p2 := \frac{p0 \cdot \lambda r + p6 \cdot \mu r + p4 \cdot \mu w}{\lambda r + \lambda w + \mu r} \quad p3 := \frac{p2 \cdot \lambda w}{\lambda r + \lambda w + \mu r}$$

$$p4 := \frac{p1 \cdot \lambda r + p7 \cdot \mu w + p9 \cdot \mu r}{\lambda r + \lambda w + \mu w} \quad p7 := \frac{p4 \cdot \lambda w + p5 \cdot \lambda r}{\mu w}$$

$$p5 := \frac{p1 \cdot \lambda w + p10 \cdot \mu r + p11 \cdot \mu w}{\lambda r + \lambda w + \mu w} \quad p8 := \frac{p4 \cdot \lambda r}{\mu w} \qquad (1)$$

$$p6 := \frac{p2 \cdot \lambda r + p8 \cdot \mu w + p12 \cdot \mu r}{\lambda r + \lambda w + \mu r} \quad p9 := \frac{p3 \cdot \lambda r + p6 \cdot \lambda w}{\mu r}$$

$$p10 := \frac{p3 \cdot \lambda w}{\mu r} \quad p11 := \frac{p5 \cdot \lambda w}{\mu w} \quad p12 := \frac{p6 \cdot \lambda r}{\mu r}$$

The system as presented in Figure 1 is considered to be conservative, and therefore the summation of all the state probabilities equals to 1. This implies that the 13 system equations are not independent. Considering the equations for the states 0 through 11 as the independent equations of the system allows determining the state probabilities (see Appendix A for a detailed discussion on how to resolve the system). The goal of the analysis is now to mathematically abstract the mean number of write requests (*Nw*), the mean number of read requests (*Nr*), the mean throughput of the system (*Xs*), and the mean response time of the requests (*Rr*). Further, the study quantifies the queue size (*N*) based on the blocking probability (*Pq*) [5],[10],[11].

$$N_r = \sum_r r \cdot p(w, r, sio)$$

$$N_r = p2 + p3 + p4 + p7 + p10 + 2 \cdot (p6 + p8 + p9) + 3 \cdot p12 \qquad (2)$$

$$N_w = \sum_w w \cdot p(w, r, sio)$$

$$N_w = p1 + p3 + p4 + p8 + p9 + 2 \cdot (p5 + p7 + p10) + 3 \cdot p11 \qquad (3)$$

$$X_s = \mu w \cdot \sum_{sio=1} p(w, r, sio) + \mu r \cdot \sum_{sio=2} p(w, r, sio)$$

$$X_s = \mu w \cdot (p2 + p3 + p6 + p9 + p10 + p12) + \mu r \cdot (p1 + p4 + p5 + p7 + p8 + p11) \qquad (4)$$

Based on Little's Law [10], the mean aggregate response time can be expressed as *Rr = (Nr + Nw) / Xs* (Equation 5). A new I/O request will have to be blocked if *r + w = 3*. Therefore, the queue size based blocking probability can be expressed as:

$$P_q = \sum_{r+w=3} p(w, r, sio)$$

$$P_q = p7 + p8 + p9 + p11 + p12 \qquad (6)$$

2.0 Simulation & Results

To illustrate the application of the (conceptional) model as presented in this study, the following experiments were conducted. In a 1st phase, the model was calibrated with a read arrival rate of 1 ($\lambda r = 1$), a read service rate of 20 ($\mu r = 20$), and a write service rate of 15 ($\mu w = 15$). In this 1st scenario, the arrival rate of the write requests (λw) was scaled from 3 to 10. In a 2nd experiment, the arrival rate of the write requests was set to 1 ($\lambda w = 1$), while the arrival rate of the read requests (λr) was scaled from 3 to 10. The service rate of the read (μr) and write requests (μw) equaled to 20 and 15, respectively. The last 2 experiments were conducted with a read arrival rate of 1 ($\lambda r = 1$), and a write arrival rate of 2 ($\lambda w = 2$). This scenario basically implies that $N = 3$. In a first experiment, the write service rate was set to 15 ($\mu w = 15$), whereas the read service rate (μr) was scaled from 10 to 20. In a second experiment, the read service rate was set to 20 ($\mu r = 20$), whereas the write service rate (μw) was scaled from 10 to 20. The goal if these experiments was not to quantify effective performance, but to discuss the relative performance behavior among the *read()* and the *write()* requests, focusing on the priority scheme outlined in this study.

Figure 2: Blocking Probability (Scaled Arrival Rate)

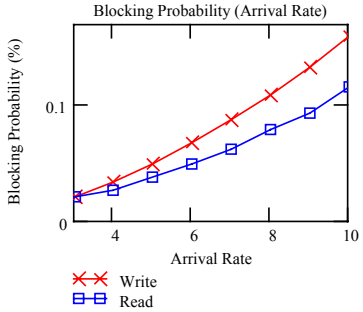
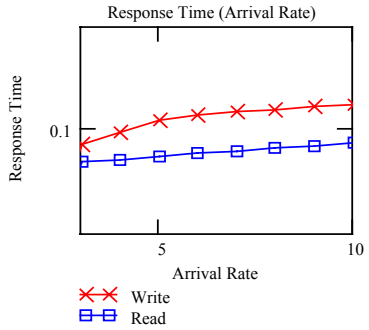


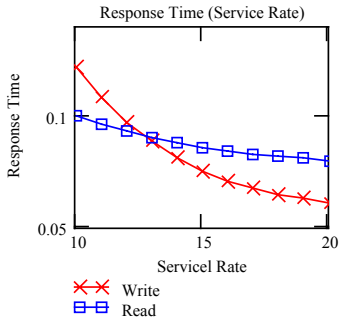
Figure 2 and 3 outline the relative performance behavior of the *read()* verses the *write()* requests while scaling the I/O request arrival rate. The delta in the processing rate, as well as the fact that in this study the *read()* request are prioritized over the *write()* requests is visible in the blocking probability graph.

Figure 3: Response Time (Scaled Arrival Rate)



Further, the trajectories for the response time reveal that while scaling the *write()* request arrival rate, the performance is impacted by the lower service rate for processing the additional *write()* operations. In the same scenario, the additional *read()* requests are processed much more efficiently. The relative throughput behavior disclosed in Figure 4 underlines that the model was calibrated with a lower fixed service rate for the *write()* requests ($\mu w = 15$), as compared to the *read()* operations (($\mu r = 20$). The analysis revolved around 2 *write()* and 1 *read()* operation, respectively. While scaling the service rate for the I/O operations, the read-scaling scenario resulted in an overall smaller throughput gain (actual performance delta) than the write scaling option.

Figure 4: Response Time (Scaled Service Rate)



At a service rate of 10, the read scenario benefits from a higher ($\mu w = 15$) fixed write service rate employed for the 2 *write()* operations. At the same service rate level, the write scenario is impacted by having to process 2 I/O operations at a service rate of 10 ($\mu w = 10$). At a service rate of 20, the picture is reversed, as the benchmark simulates 3 I/O operations at a service rate of 20 for the *write()* scenario, whereas the *read()* scenario results in operating on 2 *write()* operations that both encounter a service rate of 15 ($\mu w = 15$).

References

1. *Anderson, D., "A Drive Perspective", IEEE MMST 03, San Diego, CA, 2003*

2. *Anderson, D., "From Storage", Storage Journal, Vol. 1, No.4, Seagate Technology, 2003*

3. *Barve, R., Shriver, E., Gibbons, P., Hillyer, B., Matias, Y., Vitter, J., "Modeling and Optimizing I/O Throughput of Multiple Disks on a Bus", Bell Labs, 2000*

4. *Ganger, G., Patt, Y., "Using System-Level Models to Evaluate I/O Subsystem Designs", IEEE Computer Society, 1998*

5. *Hennessy, J., Patterson, D., "Computer Architecture, a Quantitative Approach", Third Edition, Morgan Kaufmann, 2003*

6. *Worthington, B., Ganger, G., "Scheduling Algorithms for Modern Disk Drives", ACM Sigmet-Rics Conference, Nashville, TN, May 1994*

7. *McCarthy, S., Leis, M., Byan, S., "Large Disk Blocks – Or Not", USENIX 02, Santa Cruz, 2002*

8. *Heger, D., "Modeling and Predicting Load-Dependent I/O Performance in a ZBR Environment", CMG Journal, Issue 111, Summer Edition, 2003*

9. *Friesenborg, S., "IBM TotalStorage FastT600 Storage Server and AIX Operating System Disk Performance Measurements", IBM Storage Systems Group, 2000*

10. *Jain, R., "The Art of Computer System Performance Analysis", John Wiley, 1991*

11. *Gross, D., Harris, C., "Queuing Theory, Second Edition, Wiley, 1998*

1. Appendix A: Linear Systems

A system of N first-order linear homogeneous differential equations with constant coefficient can be expressed in matrix form as $P(t) = MP(t)$ (Equation 1), where $P(t)$ represents a column vector consisting of the N functions ($P_1(t), P_2(t), \ldots, P_N(t)$), and M reflects the $N \times N$ coefficient matrix. Assuming the determinant of M is non-zero (which implies that the N equations are independent), and assuming that the initial values for $P(0)$ are known, the complete solution of this system of equations can be expressed as:

$$P(t) = e^{Mt} \cdot P(0) \quad (2)$$

Equation 2 can be utilized to determine the values of each function $Pj(t)$ (for any given time t). In some circumstances, it is useful though to generate explicit expressions for the individual functions $Pj(t)$. To accomplish this, first let $f(r) = |M-\rho I|$ be the characteristic polynomial of the system, with the roots ($\rho_1 1, \rho_2, \ldots, \rho_N$). To account for duplicate roots, let n denote the number of distinct roots, and let mj denote the multiplicity of the j-th distinct root. Therefore, $m_1 + m_2 + \ldots + m_n = N$. Hence, each individual component of $P(t)$ reflects the form:

$$P_i(t) = \sum_{j=1}^{n} \left[\left(\sum_{k=1}^{m_j} C_{i,j,k} \cdot t^{k-1} \right) \cdot e^{\rho j t} \right] \quad (3)$$

The objective is to determine the coefficients $C_{i,j,k}$ explicitly in terms of M and $P(0)$. The starting point is to note that if $P^{(s)}(t) = d^s P(t)/dt^s$ denotes the s-th derivative of P with respect to t, Equation (1) immediately implies that:

$$P^{(s)}(t) = MP^{(s-1)}(t) = M^s P(t) \quad (4)$$

$$\begin{bmatrix} P_1(t) \\ P_2(t) \\ P_3(t) \\ P_4(t) \end{bmatrix} = \begin{pmatrix} C_{11} & C_{12} & C_{13} & C_{14} \\ C_{21} & C_{22} & C_{23} & C_{24} \\ C_{31} & C_{32} & C_{33} & C_{34} \\ C_{41} & C_{42} & C_{43} & C_{44} \end{pmatrix} = \begin{bmatrix} e^{\rho_1 t} \\ e^{\rho_2 t} \\ te^{\rho_1 t} \\ \left(t^2 \cdot e\right)^{\rho_1 t} \end{bmatrix} \quad (5)$$

To express Equations 3 in matrix form, it is feasible to place the C coefficients into a $N \times N$ matrix C, and to place the factors of the form $t^k e^{\rho t}$ into a column vector (E). To illustrate, assuming a system with a degree $N = 4$ with $n = 2$ distinct roots, where 1 root has a multiplicity $m = 3$, the general solution can be expressed as in Equation 5. Therefore, it is feasible to outline that $P(t) = CE(t)$, and hence $P^s(t) = CE^s(t)$ (Equation 6). By letting $[V_1, V_2, \ldots, V_N]$ denote the square matrix with columns ($V_1, V_2, \ldots, V_N$), it is feasible to combine Equations 4 and 5 to obtain the C matrix as expressed in Equation 7:

$$C\left[\left(E^0\right), E^1, E^2, \ldots, E^{N-1}\right] = \left[(P), MP, M^{2P}, \ldots, M^{N-1}P\right]$$

$$C = \frac{\left[(P), MP, M^{2P}, \ldots, M^{N-1}P\right]}{\left[\left(E^0\right), E^1, E^2, \ldots, E^{N-1}\right]} \quad (7)$$

Equation 7 omits the argument (t) from the symbols for $P(t)$ and $E(t)$, respectively. The understanding here is that the actual evaluation occurs at the same t. Given the initial vector $P(0)$, it is convenient to process the evaluations at $t = 0$. The ramification is that each element of $E(0)$ is either 1 (1st appearance of a distinct root) or 0 (subsequent appearances), and that the remaining columns $E^{(j)}$ can be formulated based on a simple rule. To illustrate, if the i-th element of $E(0)$ represents the form $e^{\rho t}$, the corresponding row of the square matrix composed of the E columns can be expressed as $[1, \rho, \rho^2, \ldots, \rho^{N-1}]$. In the scenario where there are repeated roots, some of the elements of $E(0)$ will represent the form $t^k e^{\rho t}$, a form where the corresponding row begins with k zeros, and contains the remaining terms as indicated by:

$$\left[0, \ldots, 0, \frac{k!}{0!}, \frac{(k+1!)}{1!} \cdot \rho, \ldots, \frac{(N-1)!}{(N-k-1)!} \cdot \rho^{N-k-1}\right] \quad (8)$$

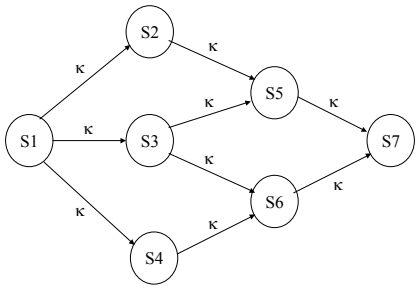
In Equation 8, if $k = 0$, the already discussed $[1, \rho, \rho^2, \ldots, \rho^{N-1}]$ form applies. Assuming that ε denotes the square matrix whose columns represent the $E^{(j)}(0)$ vectors, and utilizing the orthogonal unit row vectors $R_1 = [1,0,0,..], R_2 = [0,1,0,..], R_3 = [0,0,1,..]$ etc. results in re-formulating Equation 7 as depicted in Equation 9, which allows for expressing the complete solution as depicted in Equation 10.

$$C = \left(\sum_{j=1}^{N} M^{j-1} \cdot P(0) \cdot R_j \right) \cdot \varepsilon^{-1} \quad (9)$$

$$P(t) = \left(\sum_{j=1}^{N} M^{j-1} \cdot P(0) \cdot R_j \right) \cdot \varepsilon^{-1} \cdot E(t) \quad (10)$$

To further illustrate the application of the discussed solution, this study considers a simple monotonic Markov model as shown in Figure 1. In the presented Markov model, all the *probability* is initially in state 1.

Figure 1. Monotonic Markov Model



The presented system in Figure 1 is considered to be conservative, and therefore the summation of all the state probabilities always equals to 1. This implies that the 7 system equations are not independent. Considering the equations for the states 1 through 6 as the independent equations of the system, it is feasible to formulate that $P'(t) = MP(t)$ where

$$M = \begin{pmatrix} -3\lambda & 0 & 0 & 0 & 0 & 0 \\ \lambda & -\lambda & 0 & 0 & 0 & 0 \\ \lambda & 0 & -2\lambda & 0 & 0 & 0 \\ \lambda & 0 & 0 & -\lambda & 0 & 0 \\ 0 & \lambda & \lambda & 0 & -\lambda & 0 \\ 0 & 0 & \lambda & \lambda & 0 & -\lambda \end{pmatrix} \quad P = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \quad (11)$$

The 6 *eigenvalues* of the system, as determined by solving for the roots of the characteristic polynomial that is give by the determinant $|M-\rho I|$ equal to $\rho_1 = -3\lambda$, $\rho_2 = -2\lambda$, ρ_1, and $\rho_3 \ldots \rho_6 = \lambda$, respectively.

$$\varepsilon = \begin{bmatrix} 1 & -3\lambda & (-3\lambda)^2 & (-3\lambda)^3 & (-3\lambda)^4 & (-3\lambda)^5 \\ 1 & -2\lambda & (-2\lambda)^2 & (-2\lambda)^3 & (-2\lambda)^4 & (-2\lambda)^5 \\ 1 & -\lambda & (-\lambda)^2 & (-\lambda)^3 & (-\lambda)^4 & (-\lambda)^5 \\ 0 & 1 & 2(-\lambda) & 3(-\lambda)^2 & 4(-\lambda)^3 & 5(-\lambda)^4 \\ 0 & 0 & 2 & 6(-\lambda) & 12(-\lambda)^2 & 20(-\lambda)^3 \\ 0 & 0 & 0 & 6 & 24(-\lambda) & 60(-\lambda)^2 \end{bmatrix} \quad (12)$$

Therefore, the system discloses 3 distinct roots, with 1 that represents a multiplicity of 4. Hence, the matrix comprised of the $E^{(j)}(0)$ column vectors can be expressed as in Equation 12. Substituting Equation 12 into Equation 10 results in defining $P(t)$ as expressed in Equation 13.

$$P(t) = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ \frac{-1}{2} & 0 & \frac{1}{2} & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 & 0 \\ \frac{-1}{2} & 0 & \frac{1}{2} & 0 & 0 & 0 \\ \frac{3}{4} & -1 & \frac{1}{4} & \frac{\lambda}{2} & 0 & 0 \\ \frac{3}{4} & -1 & \frac{1}{4} & \frac{\lambda}{2} & 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} e^{-3\lambda t} \\ e^{-2\lambda t} \\ e^{-\lambda t} \\ t e^{-\lambda t} \\ t^2 e^{-\lambda t} \\ t^3 e^{-\lambda t} \end{pmatrix} \quad (13)$$

The individual time-dependent probability functions can now be defined. The time-dependent probability for state 7 can be expressed as the complement of the probability for the other 6 states.

$$P_1(t) = e^{-3\lambda t} \quad (14)$$

$$P_2(t) = P_4(t) = \frac{-1}{2} \cdot e^{-3\lambda t} + \frac{1}{2} \cdot e^{-\lambda t} \quad (15)$$

$$P_3(t) = (-e)^{-3\lambda t} + e^{-2\lambda t} \quad (16)$$

$$P_5(t) = P_6(t) =$$

$$\frac{3}{4} \cdot e^{-3\lambda t} - e^{-2\lambda t} + \frac{1}{4} \cdot e^{-\lambda t} + \frac{1}{2} \cdot \lambda t \cdot e^{-\lambda t} \quad (17)$$

$$P_7(t) = 1 - \frac{1}{2} \cdot e^{-3\lambda t} + e^{-2\lambda t} - \frac{3}{2} \cdot e^{-\lambda t} - \lambda t \cdot e^{-\lambda t} \quad (18)$$

References

1. *Datta, B., "Partial Eigenvalue Assignment in Linear Systems. Existence, Uniqueness and Numerical Solution", Department of Math, University of N. Illinois, 2000*

2. *Datta, B., "Numerical Linear Algebra and Applications", Brook-Cole Publishing, Pacific Grove, California, 1998*

3. *Korilis, Y., "Theory & Fundamentals", Math, and Science, University of Pennsylvania, 2002*

4. *Standish, R., "Linear Systems", ac3, 2003*