Warehouse Simulation Project 2

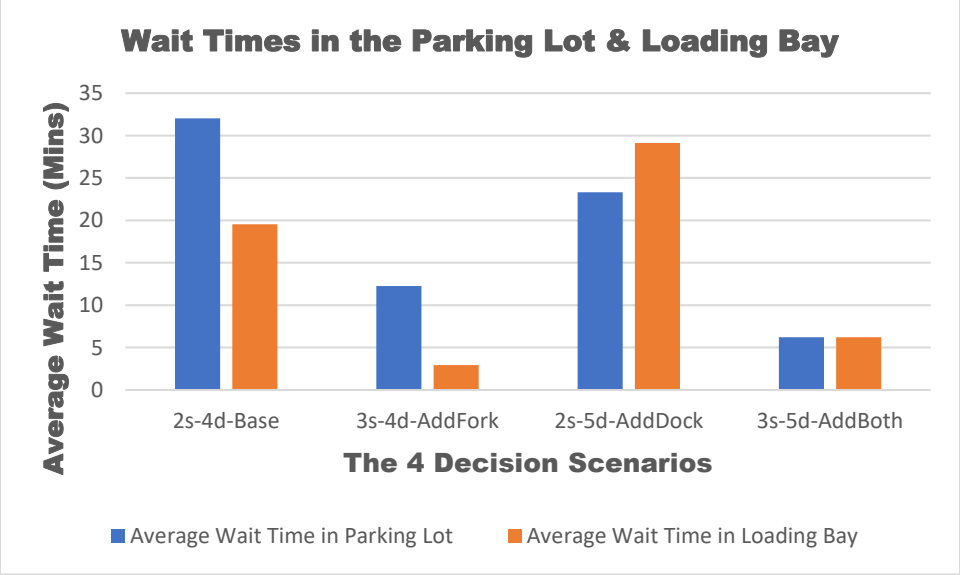
Vivek Gopalan

Purpose of the project:

The warehouse under consideration has a goal of improving service at its facility by pursuing two options: increasing the number of fork lift operators (servers) and/or increasing the number of warehouse docks (loading bays). Simulation is used as a modeling tool to predict service improvement levels for various combinations of these two options. In particular, the simulation model must be able to predict performance for 4 distinct scenarios: base scenario (2 fork lift operators and 4 loading bay docks, this scenario labeled as 2s-4d-Base); adding one fork lift operator only (this scenario labeled as 3s-4d-AddFork); adding one loading warehouse dock only (scenario labeled as 2s-5d-AddDock) and finally adding *both* a fork lift operator and a loading bay (scenario labeled as 3s-5d-AddBoth). The purpose of the project is to recommend one of these decisions to management based upon a scientific understanding of service levels as predicted by the simulation model. (Note: In the above notation, we can think of **s = server** and **d = warehouse dock**).

Executive summary:

Of all the metrics outlined in the next section, we emphasize in particular the wait time in the parking lot and loading bay as these are critical metrics from the customer viewpoint. The chart below compares these wait times for the 4 scenarios.



It is recommended that management purse the option of adding a single fork lift (i.e., select 3s-4d-AddFork) for the following reasons: for this option wait times in the parking lot goes down significantly from about 32 minutes to 12 minutes; moreover the option of adding a single loading bay is ineffective because it **simply transfers the wait time to the loading bay from the parking lot**; the option of adding both a fork lift and a loading bay is discarded because adding the dock is expensive and also adding the loading bay may lead to construction at the site which may add other

intangible delays. In addition the total wait (parking lot + dock) reduces by just 3 minutes for this expensive option.

Experimental design and analysis:

Overview:

The 4 scenarios analyzed are 2s-4d-Base, 3s-4d-AddFork, 2s-5d-AddDock, 3s-5d-AddBoth**. The performance metrics used to assess the 4 scenarios were (see SUMMARY TAB in attached Excel spreadsheet for details):** Average wait in the parking lot, average number of trucks in the parking lot, customers remaining at 4 p.m. in the parking lot, fork lift utilization, loading bay (dock) utilization, average wait time in the dock and the average number of customers waiting in the dock. In addition, the VBA program tracked the total system time, up until driver loading was complete. The simulation was run for a period of 540 minutes (9 hours) and 1600 replications were conducted (the rationale for choosing 1600 is explained shortly).

Performance analysis:

2s-4d-Base vs 3s-4d-AddFork:

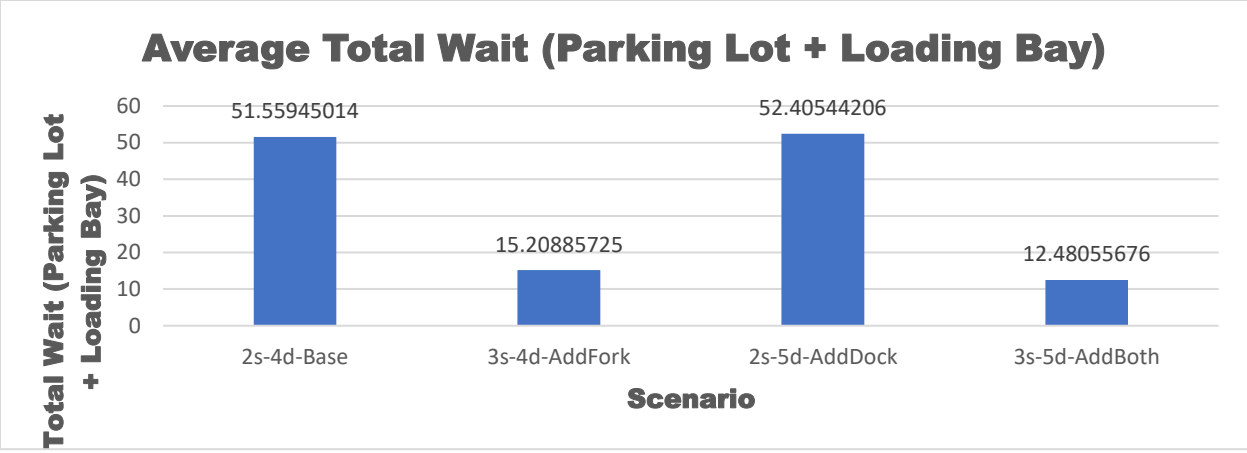
For 3s-4d-AddFork, the wait times in the parking lot go down significantly (from 32 minutes to 12) and the wait times in the loading bays also improves (from 19.53 to 2.94). Very importantly, the total system time reduces from 96.53 to 65.5. While the customer centric metrics improve, we do need to add a fork lift server and it is noteworthy that with 3 servers, the utilization is only about 2.1, so the 3rd added server may be idle quite a bit. The truck numbers remaining at 4 p.m. in the parking lot goes down just a bit from 4.3 to 1.13. **Despite these drawbacks, this is the configuration recommended to management, due to the improvement in customer-centric metrics.**

2s-4d-Base vs 2s-5d-AddLoad:

Here the comparison is not at all favorable. The reduction in parking lot waiting time is from 32 to 23 minutes. Moreover, the confidence interval for the dock utilization is [3.79, 3.86], so the 5th added loading bay is not really effective and remains unused. The average wait in the loading bay is now high almost 29 minutes. **Effectively, this is just transferring the waiting from the parking lot to the loading bay without increasing real throughput.** For this reason alone, we can reject configuration 2s-5d-AddLoad.

2s-4d-Base vs 3s-5d-AddBoth:

This configuration achieves a massive reduction in parking lot wait time which is now only 6.23 minutes. The total system time is also the best for this configuration, about 62.95 minutes. However this improvement comes at a massive cost of adding *both* a fork lift and a loading bay. The loading bay utilization is still low. **Very importantly, if we compare the total wait time in both the parking lot and loading bay combined, this option only provides a savings of about 3 minutes (see chart below).** For this reason, it is recommended that management only purse this option if a sufficient budget is available for expansion. Adding just a fork lift will get management to a total wait time of about 15 minutes.



Design specifications:

Number of replications:

The number of replications was determined by setting a relative error target of 5% for all performance metrics (i.e., half-width/mean < 0.05). For this purpose, first 100 replications were performed to obtain an approximate mean for all performance metrics. The mean was multiplied by 0.05 to get an estimated half-width. Then the sample size formula was used as indicated in **Appendix 2** of this document to obtain a required sample size for *each* performance metric. The metric "Number remaining in the parking lot" needed the maximum sample size (= 1539), followed by average number in the parking lot (= 1294). To make taking SQRT(N) easy, the **NREPS was set to 1600**. This was applied to all scenarios.

Confidence Intervals (reproduced from SUMMARY TAB in Excel spreadsheet):

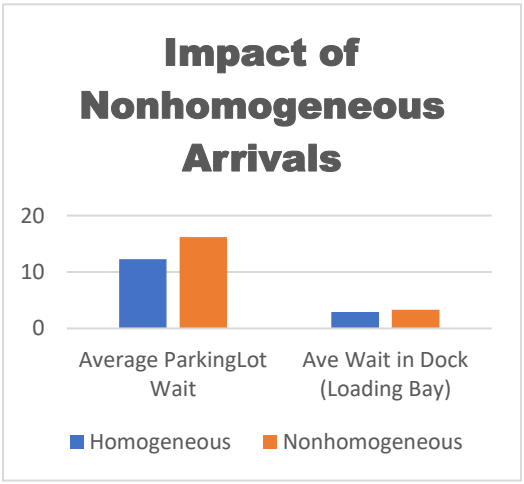
See Appendix 3 of this document for detailed confidence interval numbers and also the SUMMARY TAB in the submitted spreadsheet.

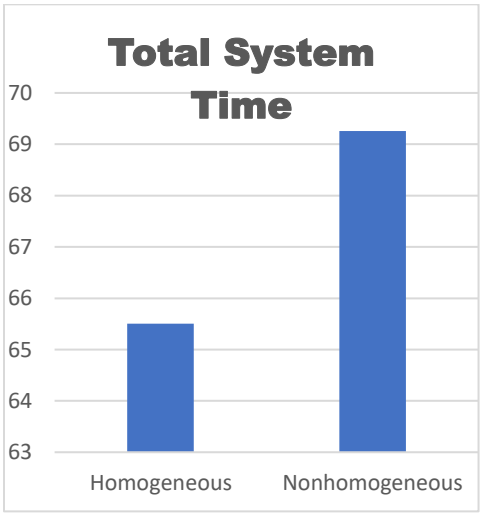
Input modeling (Extra credit):

- The input model for all 4 scenarios described followed a simple Poisson process with MeanTBA = 18 minutes as stipulated. This base Poisson input model was used to compare all 4 scenarios. A call was made to Expon(MeanTBA, stream = 1) for all inputs. For service processes, we used Erlang(m, MeanST, stream = 2).
- However, the extra credit assignment required consideration of changing the input model to incorporate a nonhomogeneous Poisson process. Two approaches were taken for this purpose: A) the Poisson thinning approach as described in the lecture power point. The VBA function GetThinningIAT was used (IAT = InterArrivalTime). This followed the methodology described in the lecture power point and summarized in **Appendix 4.** B) The second method used page 130 of the textbook and relies on simply flipping the arrival rate λ every hour. **Appendix 4** describes the VBA code parameters that need to be changed to implement the 2 methods. ***The 2 methods provided almost the same numbers, so Thinning is described here.***

Impact of nonhomogeneous Poisson processes on system performance:

Since we recommended choosing configuration 3s-4d-AddFork to management, we compare performance metrics only for 3s-4d-AddFork across the two types of inputs (i.e., homogeneous and NHPP) in the charts below. (Note carefully: In this comparison, for the homogeneous case, MeanTBA was set at 18 minutes as stated in the problem. If we consider taking the average MeanTBA *across* the 9 hours, this becomes too high, almost 38 minutes (probably due to the very high MeanTBA in the last hour) and this did not make sense for the comparison. Moreover, the *median* MeanTBA across the 9 hours is about 22 minutes, very close to the 18 minute value used in this comparison.)





Clearly, the added variability of nonhomogeneous arrivals detracts from some of the benefits of adding a fork lift driver to the system. However the numbers remaining in the parking lot only modestly increases for NHPP arrivals, but this may be due to the fact that the last hour has a very low arrival rate for trucks. **See the SUMMARY TAB in the submitted Excel sheet for details.**

Conclusion:

Simulation was used as a tool for comparing various system configurations, in particular determining whether it is better to add fork lift operators or loading bays to the warehouse. The problem statement for the project indicates that adding fork lift operators is less expensive and fortunately, this option also seems to lead to significant system performance improvement (as compared to adding loading bays). It is recommended to management that they implement option 3s-4d-AddFork. Nonhomogeneous arrivals also cause a problem and detract from system performance. Given that the simulation reveals this clearly, it may pay for management to implement an electronic "reservation system" whereby truck drivers *pre-select an arrival window* of 1 hour on a given day, well before they come in to the warehouse. By smoothing the load of arrivals, management can remove some of the impact of nonhomogeneous arrivals and improve system performance. Finally, the simulation model can be used in a number of other ways as well. For instance, we can study the impact of investing in warehouse layouts that reduce service times from the current mean of 40 minutes. What is the increased throughput of the warehouse when service times are reduced to say 35 minutes on average? Given that it is expensive to test many real-life changes by actually implementing them, simulation offers an ideal tool for performing these what-if analyses.

APPENDIX 1: VBA SIMULATION CODE

```
' Example illustrating use of VBASim for WAREHOUSE simulation
' Last update 12/5/2022

' See VBASim module for generic declarations
' See Class Modules for the supporting VBASim classes
Public Const Nservers = 3
Public Const DockSize = 4
Public Const PhasesFork = 4
Public Const PhasesLoad = 3
' PLEASE SEE TAB Determining_REPS_Worksheet and lines 100-108 in that tab for details of how 1600 was arrived at.
Public Const NREPS = 1600

' If ImplementNHPPArrivals = 0, NO NHPP implemented, if ImplementNHPPArrivals = 1, use a flat rate (page 130 of book)
' and if ImplementNHPPArrivals = 2, implement thinning (as outlined in lecture power point)

Public Const ImplementNHPPArrivals = 2

' Parameters we may want to change
Private MeanTBA As Double        ' mean time between arrivals
Private MeanSTFork As Double ' this var models average Fork operator time
Private MeanSTLoad As Double ' Models the driver's load time
Private MeanST As Double         ' mean service time
Private Phases As Integer        ' number of phases in service distribution
Private RunLength As Double      ' run length
Private WarmUp As Double         ' "warm-up
Private TotalDockTrucks As Integer ' Indicates a running count of total number of trucks = (being served + waiting for server) in the warehouse docks
```

```
Private arrChangeTime As Double
Public NHPP_RATES(1 To 9) As Double
Public ThinProbArray(1 To 9) As Double
Public ThinningProb As Double
Public MaxLambdaRate As Double
Public NHPP_EPOCH As Integer

Private NumCalls As Integer  ' Record TOTAL number of calls
' Global objects needed for simulation
' These will usually be queues and statistics

Dim Queue As New FIFOQueue          'customer = truck queue
Dim DockQueue As New FIFOQueue      'queue = trucks who are waiting in the dock area, docked but without a Fork Lift available
Dim Wait As New DTStat              'discrete-time statistics on customer waiting
Dim DockWait As New DTStat
Dim ServiceEndStats As New DTStat
Dim LoadEndStats As New DTStat

' Dim NumLostCalls As New DTStat      ' discrete-time statistics on number of calls lost
Dim Server As New Resource          'server resource
Dim WarehouseDock As New Resource

Private Sub Warehouse()
' Initialize
    Dim Reps As Integer
    Dim NextEvent As EventNotice
    Dim i As Integer
    Dim j As Integer
```

```
Call MyInit ' special initializations for this simulation

For Reps = 1 To NREPS
   NumCalls = 0
   TotalDockTrucks = 0
   arrChangeTime = 0
   Call VBASimInit 'initialize VBASim for each replication

   Call Schedule("Arrival", Expon(MeanTBA, 1))
   Call Schedule("EndSimulation", RunLength)
   Call Schedule("ClearIt", WarmUp)
   If ImplementNHPPArrivals >= 1 Then
      For j = 1 To 9
         arrChangeTime = WarmUp + (j - 1) * 60
         Call Schedule("UpdateMeanTBAorThinningProb", arrChangeTime)
      Next j
   End If

   Do
      Set NextEvent = Calendar.Remove
      Clock = NextEvent.EventTime
      Select Case NextEvent.EventType
      Case "Arrival"
         Call Arrival
         NumCalls = NumCalls + 1
      Case "UpdateMeanTBAorThinningProb"
         Call UpdateMeanTBAorThinningProb
      Case "EndOfService"
```

```
            Call EndOfService(NextEvent.WhichObject)
        Case "EndOfLoading"
            Call EndOfLoading(NextEvent.WhichObject)
        Case "ClearIt"
            Call ClearStats
            NumCalls = 0
        End Select
    Loop Until NextEvent.EventType = "EndSimulation"

' Write output report for each replication

    Call Report(Wait.Mean, "WarehouseSim", Reps + 1, 1)
    Call Report(Queue.Mean, "WarehouseSim", Reps + 1, 2)
    Call Report(Queue.NumQueue, "WarehouseSim", Reps + 1, 3)
    Call Report(Server.Mean, "WarehouseSim", Reps + 1, 4)
    Call Report(Nservers, "WarehouseSim", Reps + 1, 5)
    Call Report(DockQueue.Mean, "WarehouseSim", Reps + 1, 6)
    Call Report(DockWait.Mean, "WarehouseSim", Reps + 1, 7)
    Call Report(WarehouseDock.Mean, "WarehouseSim", Reps + 1, 8)
    Call Report(DockSize, "WarehouseSim", Reps + 1, 9)
    ' The metric below measures average TOTAL system time for trucks when loading is
completed.
    Call Report(LoadEndStats.Mean, "WarehouseSim", Reps + 1, 10)

  Next Reps

  End ' ends execution, closes files, etc.

End Sub
```

```
Private Sub Arrival()
    Dim IAT As Double
    Dim localMeanTBA As Double
' Arrival event
' Schedule next arrival
    If ImplementNHPPArrivals = 0 Then
        ' BASE CASE:  Arrivals are homogeneous
        Call Schedule("Arrival", Expon(MeanTBA, 1))
    ElseIf ImplementNHPPArrivals = 1 Then
        ' This implements the simple method in page 130 of the book.  No thinning but Lambda updated every hour.
        Call Schedule("Arrival", Expon(MeanTBA, 1))
    ElseIf ImplementNHPPArrivals = 2 Then
        ' This implements Thinning as discussed in the power point provided for NHPP. ImplementNHPPArrivals = 2 and ImplementNHPPArrivals = 1
        ' should provide answers that are close together...
        ' IAT = InterArrivalTime
        ThinningProb = ThinProbArray(NHPP_EPOCH)
        IAT = GetThinnedIAT(ThinningProb)
        ' MsgBox (IAT) ' This was used for debugging...Ignore.
        Call Schedule("Arrival", IAT)
        ' localMeanTBA = 1 / (MaxLambdaRate * ThinProbArray(NHPP_EPOCH))
        ' Call Schedule("Arrival", Expon(localMeanTBA, 1))
    End If

' Create and Process the newly arriving customer
    Dim Customer As New Entity
' If Docks can accommodate new arrival, place in dock else place in parking lot
' If fork lift operator is free schedule end of service as soon as the arrival happens, else place in Dock Queue waiting for fork lift
```

```
    If TotalDockTrucks = DockSize Then
        Queue.Add Customer
    ElseIf (Server.Busy = Nservers) And (TotalDockTrucks < DockSize) Then
        WarehouseDock.Seize (1)
        DockQueue.Add Customer
        TotalDockTrucks = TotalDockTrucks + 1
        Customer.DockWaitStart = Clock
        ' For such a customer note that the Parking Lot wait time = 0
        Wait.Record (Clock - Customer.CreateTime)
    ElseIf (Server.Busy < Nservers) And (TotalDockTrucks < DockSize) Then
        Server.Seize (1)
        WarehouseDock.Seize (1)
        Call SchedulePlus("EndOfService", Erlang(PhasesFork, MeanSTFork, 2), Customer)
        TotalDockTrucks = TotalDockTrucks + 1
        Customer.DockWaitStart = Clock
        Customer.DockWaitEnd = Clock
        DockWait.Record (Customer.DockWaitEnd - Customer.DockWaitStart)
        Wait.Record (Clock - Customer.CreateTime)
    End If

    Set Customer = Nothing
End Sub
Private Sub EndOfService(DepartingCustomer As Entity)
' End of service event
' Now that service is over, schedule the LOADING part
    DepartingCustomer.ServiceEnd = Clock
    ServiceEndStats.Record (Clock - DepartingCustomer.CreateTime)
    ' NOTE: TotalDockTrucks does not really change when a fork lift service finishes.
```

```
    Call SchedulePlus("EndOfLoading", Erlang(PhasesLoad, MeanSTLoad, 3), DepartingCustomer)
    Set DepartingCustomer = Nothing 'be sure to free up memory
' Check if there is another customer in the Dock Queue that needs the fork lift.  If yes schedule another EndOfService, else free server
' NOTE:  We can ignore the DockQueue in EndOfService IF Nservers = DockSize because end of service does not free up dock space per se
' In the special case above, DockQueue.NumQueue = 0 ALWAYS!!
    If DockQueue.NumQueue > 0 Then
        Set DepartingCustomer = DockQueue.Remove
        DepartingCustomer.DockWaitEnd = Clock
        DockWait.Record (DepartingCustomer.DockWaitEnd - DepartingCustomer.DockWaitStart)
        Call SchedulePlus("EndOfService", Erlang(PhasesFork, MeanSTFork, 2), DepartingCustomer)
        Set DepartingCustomer = Nothing 'be sure to free up memory
    Else
        Server.Free (1)
    End If

End Sub
Private Sub EndOfLoading(DepartingCustomer As Entity)
' End of service + load event.  Free up warehouse dock space
    DepartingCustomer.LoadEnd = Clock
    LoadEndStats.Record (Clock - DepartingCustomer.CreateTime)

    Set DepartingCustomer = Nothing    'be sure to free up memory
    TotalDockTrucks = TotalDockTrucks - 1
' Check to see if there is another customer waiting in the dock; if yes start service
' otherwise free the server
    If Queue.NumQueue > 0 Then
        Set DepartingCustomer = Queue.Remove
```

```
        Wait.Record (Clock - DepartingCustomer.CreateTime)
        If Server.Busy < Nservers And TotalDockTrucks < DockSize Then
            Server.Seize (1)
            DepartingCustomer.DockWaitStart = Clock
            DepartingCustomer.DockWaitEnd = Clock
            DockWait.Record (DepartingCustomer.DockWaitEnd - DepartingCustomer.DockWaitStart)
            Call SchedulePlus("EndOfService", Erlang(PhasesFork, MeanSTFork, 2),
DepartingCustomer)
            TotalDockTrucks = TotalDockTrucks + 1
        ElseIf Server.Busy = Nservers And TotalDockTrucks < DockSize Then
            DockQueue.Add DepartingCustomer
            TotalDockTrucks = TotalDockTrucks + 1
            DepartingCustomer.DockWaitStart = Clock
        ElseIf Server.Busy = Nservers And TotalDockTrucks = DockSize Then
            MsgBox ("THIS CONDITION SHOULD NEVER HAPPEN - FATAL ERROR?")
        End If
        Set DepartingCustomer = Nothing 'be sure to free up memory
    Else
        WarehouseDock.Free (1)
    End If
End Sub


Public Sub UpdateMeanTBAorThinningProb()
' Data for NHPP
' MeanTBA 16.51376147 9.32642487  11.39240506 22.22222222 33.33333333 20.2247191
23.37662338 60  150
    Dim jj As Integer
    Dim localIndex As Integer
    localIndex = 1
```

```
    For jj = 1 To 9
        If (Clock >= WarmUp + (jj - 1) * 60) And (Clock < WarmUp + (jj) * 60) Then
            localIndex = jj
            Exit For
        End If
    Next jj
    NHPP_EPOCH = localIndex
    If ImplementNHPPArrivals = 1 Then
        MeanTBA = NHPP(localIndex)
    ElseIf ImplementNHPPArrivals = 2 Then
        MeanTBA = 1 / MaxLambdaRate
        ThinningProb = ThinProbArray(localIndex)
        ' If 1 / (MaxLambdaRate * ThinningProb) <> NHPP(localIndex) Then
        '    MsgBox ("Thinning rate and NHPP array do not match up?")
        ' End If
    End If
End Sub

Public Function GetThinnedIAT(ThinningProb As Double) As Double
    Dim localIAT As Double
    Dim localFlag As Integer

    localIAT = 0
    localFlag = 0
    ' NOTE:  ThinningProb is already the ratio of current lambda/max-lambda, so can compare U to ThinningProb directly
    ' NOTE:  If this function is even called, MeanTBA = Min (MeanTBA) across all 9 hours
```

```
Do
        localIAT = localIAT + Expon(MeanTBA, 1)
        ' NOTE:  Seed for Uniform is 4 because seeds 1, 2 and 3 have been used for Arrivals, Fork Service and Loading respectively.
        If Uniform(0, 1, 4) <= ThinningProb Then
           localFlag = 1
           GetThinnedIAT = localIAT
           ' Exit Function
        End If
    Loop Until localFlag = 1
End Function


Private Sub MyInit()


' Initialize the simulation
    Call InitializeRNSeed
    Server.SetUnits (Nservers) ' set the number of servers to 1
    WarehouseDock.SetUnits (DockSize) ' set the number of warehouse docks to a constant called DockSize


    MeanTBA = 18
    MeanSTFork = 40
    MeanSTLoad = 12
    ' MsgBox (MeanST)
    Phases = 3
    ' RunLength = 50000
    RunLength = 540 '
    WarmUp = 5000
```

```
' Add queues, resources and statistics that need to be
' initialized between replications to the global collections

    TheDTStats.Add Wait
    TheDTStats.Add DockWait
    TheDTStats.Add ServiceEndStats
    TheDTStats.Add LoadEndStats

    ' TheDTStats.Add NumLostCalls
    TheQueues.Add Queue
    TheQueues.Add DockQueue
    TheResources.Add Server
    TheResources.Add WarehouseDock
    ' INITIALIZE NHPP array - may not be used if even rates are modeled
    ' MeanTBA 16.51376147 9.32642487  11.39240506 22.22222222 33.33333333 20.2247191
23.37662338 60  150
    ' ThinningProb    0.564766839 1   0.81865285  0.419689119 0.279792746 0.461139896
0.398963731 0.155440415 0.062176166
    If ImplementNHPPArrivals = 0 Then
        MeanTBA = 18
    ElseIf ImplementNHPPArrivals = 1 Then
        ' NOTE:  This is set at MeanTBA in hour 1 for this method
        MeanTBA = 16.51
    ElseIf ImplementNHPPArrivals = 2 Then
        ' NOTE:  This is set at MeanTBA in fastest arrival hour for this method
        MeanTBA = 9.326
    End If
    MaxLambdaRate = 1 / 9.32
```

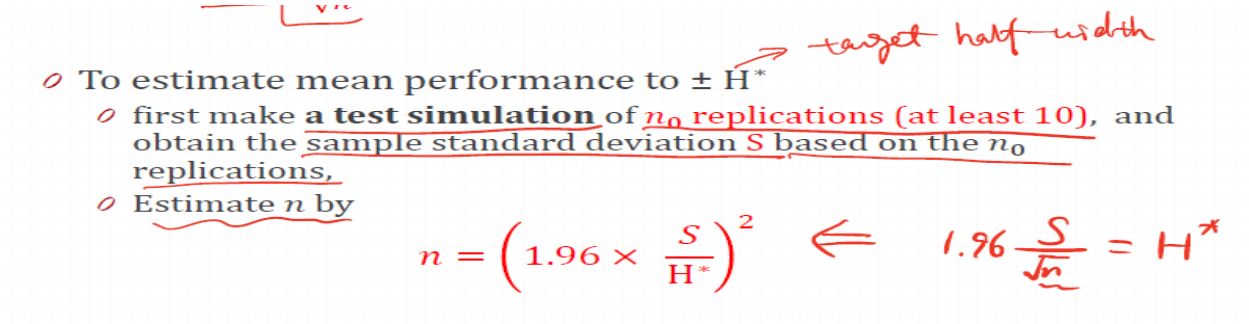
```
    NHPP_RATES(1) = 16.51
    NHPP_RATES(2) = 9.32
    NHPP_RATES(3) = 11.39
    NHPP_RATES(4) = 22.22
    NHPP_RATES(5) = 33.33
    NHPP_RATES(6) = 20.22
    NHPP_RATES(7) = 23.37
    NHPP_RATES(8) = 60
    NHPP_RATES(9) = 150
    NHPP_EPOCH = 1
    ' Populate thinning prob array
    ThinningProb = 1
    ThinProbArray(1) = 0.5648
    ThinProbArray(2) = 1
    ThinProbArray(3) = 0.819
    ThinProbArray(4) = 0.42
    ThinProbArray(5) = 0.28
    ThinProbArray(6) = 0.4611
    ThinProbArray(7) = 0.399
    ThinProbArray(8) = 0.155
    ThinProbArray(9) = 0.062
' Write headings for the output reports

    Call Report("Average ParkingLot Wait", "WarehouseSim", 1, 1)
    Call Report("Average Number in ParkingLot Queue", "WarehouseSim", 1, 2)
    Call Report("Number Remaining in ParkingLot", "WarehouseSim", 1, 3)
    Call Report("Server Utilization", "WarehouseSim", 1, 4)
    Call Report("No. Servers", "WarehouseSim", 1, 5)
```

```
    Call Report("Ave Num in Dock (Loading Bay) Que", "WarehouseSim", 1, 6)
    Call Report("Ave Wait in Dock (Loading Bay)", "WarehouseSim", 1, 7)
    Call Report("Dock (Loading Bay) Utilization", "WarehouseSim", 1, 8)
    Call Report("DockSize (Loading Bay Size)", "WarehouseSim", 1, 9)
    Call Report("Total System Time", "WarehouseSim", 1, 10)
End Sub
```

APPENDIX 2: DETERMINING NUMBER OF REPLICATIONS

The number of reps were determined using the methodology outlined in the lecture power point reproduced below:



In this case N0 = 100 and each of the metrics will imply a sample size based upon x-bar and the implied Half-width. **Taking the maximum n across all the simulation metrics, we get about 1600 replications as required to keep the ratio (half-width/x-bar) < 0.05.**

NOTES: The preliminary run was completed for the case nservers = 2, loading docks = 4, the base case.

PLEASE SEE THE EXCEL SPREADSHEET TAB **Determining_REPS_Worksheet** LINES 100-108 for details of the calculations.

Average ParkingLot Wait	Average Number in ParkingLot Queue	Number Remaining in ParkingLot
30.67775504	1.952504873	3.73
1.533887752	0.097625244	0.1865
25.31545001	1.791882873	3.733157463
1046.396521	1294.217053	**1539.242648**
		MAX OF SAMPLE SIZES ABOVE
		We will use NREPS = 1600

preliminary run to get x-bar), s = 2 servers and d = 4 dock size

Server Utilization	No. Servers	Ave Num in Dock (Loading Bay) Queue	Ave Wait in Dock (Loading Bay)	Dock (Loading Bay) Utilization
1.814510906	2	0.939496742	19.15403621	3.317770699
0.090725545	0.1	0.046974837	0.95770181	0.165888535
0.180931874	0	0.344613175	6.890971307	0.525824675
15.27858491	0	206.749911	198.8897533	38.59769084

DockSize (Loading Bay Size)	Total System Time	
4	95.88072217	<- X-bar
0.2	4.794036108	<- Target Half-wdith
0	28.60681058	<- Standard dev
0	136.7881329	<- Inferred sample size

APPENDIX 3: CONFIDENCE INTERVALS FOR ALL PERFORMANCE METRICS FOR ALL SCENARIOS

Scenario(2s-4d-BASE)	Average ParkingLot Wait	Average Number in ParkingLot Queue	Number Remaining in ParkingLot	Server Utilization	No. Servers	Ave Num in Dock (Loading Bay) Que	Ave Wait in Dock (Loading Bay)	Dock (Loading Bay) Utilization	DockSize (Loading Bay Size)	Total System Time
Mean	32.02620106	2.115267894	4.31375	1.805752355	2	0.941444659	19.53324908	3.304244836	4	96.5336842
Stdev	27.1605145	1.99235503	4.197478136	0.186325948	0	0.320949647	6.512050341	0.521116586	0	30.70686747
Half-wdith of CI	1.330865211	0.097625396	0.205676429	0.009129971	0	0.015726533	0.319090467	0.025534713	0	1.504636506
Lower Limit of CI	30.69533584	2.017642497	4.108073571	1.796622384	2	0.925718126	19.21415862	3.278710123	4	95.02904769
Upper Limit of CI	33.35706627	2.21289329	4.519426429	1.814882327	2	0.957171191	19.85233955	3.329779548	4	98.0383207
Scenario(3s-4d-Add-1-ForkLift)	Average ParkingLot Wait	Average Number in ParkingLot Queue	Number Remaining in ParkingLot	Server Utilization	No. Servers	Ave Num in Dock (Loading Bay) Que	Ave Wait in Dock (Loading Bay)	Dock (Loading Bay) Utilization	DockSize (Loading Bay Size)	Total System Time
Mean	**12.2684946**	0.79526495	1.13375	2.112479578	3	0.166930179	2.940362655	2.892991889	4	65.50532818
Stdev	13.66092916	1.000279894	2.037696264	0.355841106	0	0.098243246	1.550632602	0.518738809	0	16.07727393
Half-wdith of CI	0.669385529	0.049013715	0.099847117	0.017436214	0	0.004813919	0.075980998	0.025418202	0	0.787786422
Lower Limit of CI	11.59910907	0.746251235	1.033902883	2.095043364	3	0.16211626	2.864381657	2.867573687	4	64.71754176
Upper Limit of CI	12.93788013	0.844278665	1.233597117	2.129915792	3	0.171744098	3.016343652	2.918410091	4	66.29311461
Scenario(2s-5d-Add-1-LoadingBay)	Average ParkingLot Wait	Average Number in ParkingLot Queue	Number Remaining in ParkingLot	Server Utilization	No. Servers	Ave Num in Dock (Loading Bay) Que	Ave Wait in Dock (Loading Bay)	Dock (Loading Bay) Utilization	DockSize (Loading Bay Size)	Total System Time
Mean	23.29814731	1.604124345	3.560625	1.805875374	2	1.442783593	29.10729474	3.830845151	5	96.42531087
Stdev	23.90997442	1.793503584	3.941895	0.185210666	0	0.574117499	11.39615276	0.763818101	0	31.0929905
Half-wdith of CI	1.171588747	0.087881676	0.193152855	0.009075323	0	0.028131757	0.558411485	0.037427087	0	1.523556535
Lower Limit of CI	22.12655856	1.51624267	3.367472145	1.796800051	2	1.414651835	28.54888326	3.793418064	5	94.90175434
Upper Limit of CI	24.46973606	1.692006021	3.753777855	1.814950696	2	1.47091535	29.66570623	3.868272238	5	97.94886741
Scenario(3s-5d-Add-BOTH)	Average ParkingLot Wait	Average Number in ParkingLot Queue	Number Remaining in ParkingLot	Server Utilization	No. Servers	Ave Num in Dock (Loading Bay) Que	Ave Wait in Dock (Loading Bay)	Dock (Loading Bay) Utilization	DockSize (Loading Bay Size)	Total System Time
Mean	**6.237540757**	0.417592276	0.603125	2.13549356	3	0.367460008	6.243016002	3.130528656	5	62.9576753
Stdev	9.498819185	0.696243455	1.497754077	0.379182937	0	0.258903427	3.906316377	0.693886171	0	14.28601497
Half-wdith of CI	0.46544214	0.034115929	0.07338995	0.018579964	0	0.012686268	0.191409502	0.034000422	0	0.700014734
Lower Limit of CI	5.772098617	0.383476347	0.529735505	2.116913596	3	0.354773741	6.0516065	3.096528234	5	62.25766057
Upper Limit of CI	6.702982897	0.451708205	0.67651495	2.154073524	3	0.380146276	6.434425505	3.164529079	5	63.65769004
Scenario(3s-4d-Model-NHPP-PG130BOOK)	Average ParkingLot Wait	Average Number in ParkingLot Queue	Number Remaining in ParkingLot	Server Utilization	No. Servers	Ave Num in Dock (Loading Bay) Que	Ave Wait in Dock (Loading Bay)	Dock (Loading Bay) Utilization	DockSize (Loading Bay Size)	Total System Time
Mean	***16.58516991***	1.152145474	1.83125	2.245409747	3	0.196751077	3.271096602	3.093425361	4	69.71545989
Stdev	16.9610527	1.305740089	2.641189763	0.32959725	0	0.100498117	1.523683461	0.487648945	0	18.82659616
Half-wdith of CI	0.831091582	0.063981264	0.129418298	0.016150265	0	0.004924408	0.07466049	0.023894798	0	0.922503212
Lower Limit of CI	15.75407833	1.08816421	1.701831702	2.229259482	3	0.191826669	3.196436112	3.069530563	4	68.79295668
Upper Limit of CI	17.41626149	1.216126739	1.960668298	2.261560013	3	0.201675485	3.345757092	3.117320159	4	70.6379631
Scenario(3s-4d-Model-NHPP-ThinningPPT)	Average ParkingLot Wait	Average Number in ParkingLot Queue	Number Remaining in ParkingLot	Server Utilization	No. Servers	Ave Num in Dock (Loading Bay) Que	Ave Wait in Dock (Loading Bay)	Dock (Loading Bay) Utilization	DockSize (Loading Bay Size)	Total System Time
Mean	***16.16471097***	1.137869565	1.815	2.250103622	3	0.199949569	3.288131066	3.099652183	4	69.25870442
Stdev	15.85411713	1.248139206	2.701913713	0.337810157	0	0.102094149	1.540697935	0.499594516	0	17.79450623
Half-wdith of CI	0.77685174	0.061158821	0.132393772	0.016552698	0	0.005002613	0.075494199	0.024480131	0	0.871930805
Lower Limit of CI	15.38785923	1.076710743	1.682606228	2.233550924	3	0.194946956	3.212636867	3.075172052	4	68.38677361
Upper Limit of CI	16.94156271	1.199028386	1.947393772	2.266656319	3	0.204952182	3.363625265	3.124132314	4	70.13063522

APPENDIX 4: Two methods for generating NHPP arrivals.

In the VBA code, set parameter ImplementNHPPArrivals = 2 from the lecture power point method below.

Generation of an NSPP: Thinning

Thinning method for generating arrival times for an NSPP

1. Let $\lambda^* = \max_{0 \le t \le T} \lambda(t)$ be the maximum of the arrival rate function and set $t = 0$ and $i = 1$.
2. Generate $X \sim \text{Exp}(\lambda^*)$ and let $t = t + X$ (this is the arrival time of the stationary Poisson process). If $t > T$, stop; otherwise, continue.
3. Generate random number $U \sim \text{U}(0,1)$. If $U \le \lambda(t)/\lambda^*$, then accept this arrival by setting $\mathcal{T}_i = t$ and $i = i + 1$.
4. Go to Step 2.

APPENDIX 4 (continued):

In the VBA code set parameter ImplementNHPPArrivals = 1 for the method described in the book (page 130):

Next consider estimating the arrival rate function $\lambda(t)$ directly. A standard method is to assume that $\lambda(t)$ is piecewise constant over intervals of length $\delta > 0$, for δ small enough. Then $\widehat{\lambda}(t)$ is a piecewise constant rate function obtained as the average number of arrivals observed in nonoverlapping intervals of size δ.

To be specific, assume T/δ is integer. Then

$$\widehat{\lambda}(t) = \frac{1}{k\delta} \sum_{j=1}^{k} [C_j(\ell(t+\delta)) - C_j(\ell(t))] \tag{6.21}$$

where $\ell(t) = \lfloor t/\delta \rfloor \delta$ is the beginning of the interval in which time t falls. This is a rather complicated way to express a simple idea: To estimate the arrival rate between, say, times $i\delta < t \le (i+1)\delta$, compute the average number of arrivals that occurred during this interval across the k realizations, then divide by δ to make it a rate. The resulting $\widehat{\lambda}(t)$ can be incorporated into a thinning algorithm, or integrated and used with inversion.

To illustrate, consider again the small example where we observed an arrival process for $k = 2$ observation periods, each of $T = 5$ h. On the first observation period arrivals occurred at times $T_{11} = 1.2$ and $T_{21} = 4.1$ h, while on the second observation period we observed only 1 arrival at time $T_{12} = 2.4$ h. If we set $\delta = 2.5$ h, then there were in total two arrivals between $t = 0 \times \delta = 0$ and $t = 1 \times \delta = 2.5$, and only one arrival between times $t = 2.5$ and $t = 2\delta = 5$. Therefore, the estimated arrival rate is

$$\lambda(t) = \begin{cases} \frac{1}{k\delta} 2 = \frac{2}{5}, & 0 < t \le \delta = 2.5 \\ \frac{1}{k\delta} 1 = \frac{1}{5}, & 2.5 < t \le 2\delta = 5. \end{cases}$$

NOTE: Both Thinning and the PG130 method provided similar answers, so only Thinning is described in the results section.