Introduction

Having the machine capability to provide reserves and actually delivering reserves causes various forms of efficiency losses. The Generation and Reserves Dispatch (GARD) Model's primary purpose is to calculate the cost of the various forms of efficiency loss associated with ensuring sufficient machine capability is ready and capable of responding to and delivering the BPA Balancing Authority Area's (BAA) requirements for regulating reserves, load following reserves, and imbalance reserves. These costs are generally referred to as variable costs.

GARD was created specifically for use in the WP-10 rate case. The goal was to create a model that could capture the efficiency impacts yet function within the confines of the available rate case models. The variable costs associated with providing a quantity of reserves are assessed in GARD using inputs from the HYDSIM model, actual system data, and a pre-processing spreadsheet. The purpose of the GARD model is to calculate the variable costs incurred as a result of operating the Federal Columbia River Power System (FCRPS) with the necessary reserves to maintain reliability and deploying those reserves to maintain load-resource balance within the BPA BAA. Load-resource balance is maintained by automatically increasing or decreasing generation in response to instantaneous changes in demand and/or power production. The need to be ready and capable of automatically increasing generation is referred to as an incremental (inc) reserve. Likewise, the need to be ready and capable of automatically decreasing generation is referred to as a decremental (dec) reserve.

General Description

The GARD model is a MS Excel 2003 based model. All inputs and outputs are based in Excel spreadsheets. The core of the model is written in Visual Basic for Applications (VBA). The GARD model analyzes variable costs in two general categories. The first category is the "stand ready" costs; those costs associated with making a project capable of providing reserves. The next general cost category is the "deployment" costs; those costs incurred when the system uses its reserve capability to actually deliver in response to a reserve need. The deployment costs are calculated using the same inputs as the stand ready costs combined with a BAA station control error signal. The station control error signal is used to simulate the real-time movements of generation on a one minute basis to calculate the cost of delivering reserves.

The GARD model specifically outputs the following costs associated with standing ready:

1. energy shift associated with providing dec reserves,
2. energy shift associated with providing non-spinning reserves,
3. energy shift associated with providing spinning reserves,
4. efficiency changes associated with providing dec reserves,
5. efficiency changes associated with providing non-spinning reserves,
6. efficiency changes associated with providing spinning reserves,
7. unit cycling costs associated with providing dec reserves,
8. unit cycling costs associated with providing non-spinning reserves,
9. unit cycling costs associated with providing spinning reserves,
10. spill costs associated with providing non-spinning reserves, and
11. spill costs associated with providing spinning reserves.

GARD also provides the following costs associated with deploying reserves:

1. response losses associated with deploying incs,
2. response losses associated with deploying decs,
3. cycling losses associated with deploying incs,
4. cycling losses associated with deploying decs, and
5. spill associated with decs.

For each cost category, GARD produces monthly results for HLH and LLH by water year denominated in MWh losses (note: positive losses are gains).

In considering the variable costs, the GARD model seeks to efficiently commit and dispatch the units at projects armed for Automatic Generation Control (AGC) response, generally referred to in this document as controller projects, such that each project's generation request is met while at the same time meeting the reserve obligation and responding to a simulated reserve need. In the process of making projects capable of responding and then actually providing response, the efficiency of the generators changes. Measuring the net efficiency change associated with providing reserves is the primary concern of the GARD model.

Disclaimer This document and associated models are not final versions and are subject to change for development of the WP-12 Initial Proposal. Additionally, any data contained within the models or documentation are illustrative only and are not indicative of Initial Proposal development.

Sections

I. Calculating Stand Ready Costs 3
II. Calculating Deployment Costs 3
III. Generating Plant Curves 4
IV. Pre-Launch Check 4
V. Updating Unit Performance Functions 5
VI. Updating Unit Limits 6
VII. INPUT (GENERATION) 9
VIII. INPUT (PREDEC) 9
IX. Station Control Error File 9
X. OUTPUT 9
XI. OUTPUT() 10
XII. OUTPUT(SETUP) 10
XIII. Newton-Raphson Functions 12
XIV. Optimal Unit Dispatch 12
XV. The Reserve Obligation 15
XVI. Efficiency Loss Detail 15
XVII. Cycling Loss Detail 15
XVIII. Unit Commitment Filter 16
XIX. Processing Time Comments 16
XX. General Comments Regarding Source Code 18
XXI. Some Variable Definitions 19

Figures

Figure 1 – CONTROL 3
Figure 2 – CONTROL 3
Figure 3 – INPUT (POWERHOUSE) 5
Figure 4 – INPUT (CALCULATED) 5
Figure 5 – Example Unit Performance Curve and Function Parameters 6
Figure 6 – Example Unit Performance Curve and Limits 8
Figure 7 – Example Unit 1% Limits 8
Figure 8 – Example Project HK Curve 11
Figure 9 – Example Spinning Reserve Availability 11
Figure 10 – Minimum Turbine Flow 13
Figure 11 – Maximum Plant Efficiency 13
Figure 12 – Newton-Raphson Example 14
Figure 13 – Dynamic Filter Function 17
Figure 14 – Example of Excessive Filtering 17
Figure 15 – Quickly Identifying Subroutine Calls 19
Figure 16 – Nested If Statement Identification 19
Figure 17 – Optimized Turbine Flow Boundaries 20

Equations

Equation 1 – Objective Function 12
Equation 2 – Newton-Raphson Method 14
Equation 3 – Quadratic Equation 15
Equation 4 – Change in Project Efficiency 15
Equation 5 – Synchronization Loss 16
Equation 6 – Average Water Consumption Over Ramp 16
Equation 7 – Evaluate Anti-Derivative 16
Equation 8 – Ramping Loss 16

Process Flows

General Modeling Process Flow 20
Command() Process Flow 21
CommitSub() Process Flow 24
CycleSubDEP() Process Flow 26
CycleSubSR() Process Flow 27
DeployDispatch() Process Flow 28
Deployment() Process Flow 30
DispatchSub() Process Flow 31
NewtonRaphsonK() Process Flow 35
ShiftSub() Process Flow 36
Signals() Process Flow 38
StandReady() Process Flow 42

Disclaimer This document and associated models are not final versions and are subject to change for development of the WP-12 Initial Proposal. Additionally, any data contained within the models or documentation are illustrative only and are not indicative of Initial Proposal development.

I. Calculating Stand Ready Costs

Stand ready costs may be calculated without calculating the deployment costs (see Section II). However, calculating deployment costs requires the running of the stand ready cost calculation.

1 – Select the CONTROL tab.

2 – Activate the STAND READY button.

Note If the PLANT CURVE button is activated, the STAND READY button cannot be activated. De-activate the PLANT CURVE button if needed.

3 – If needed, deactivate the DEPLOY button. Deactivating the DEPLOY button will automatically deactivate the IMPORT ERROR SIGNAL and WRITE DEPLOY DETAIL buttons if they are currently activated.

4 – Under "Fiscal Year," enter the fiscal year of the study.

5 – Under "Water Year," enter the starting water year after "Beg" and the ending water year after "End."

Note 1 >= Beg <= 70
1 >= End <= 70
Beg <= End

6 – Under "Month," enter the starting month after "Beg" and the ending month after "End."

Note 1 >= Beg <= 12
1 >= End <= 12
Beg <= End

7 – Perform pre-launch check per Section IV.

8 – Select the LAUNCH button.

9 – When the calculations are complete, a dialog box will appear indicating the starting and ending times of the run.

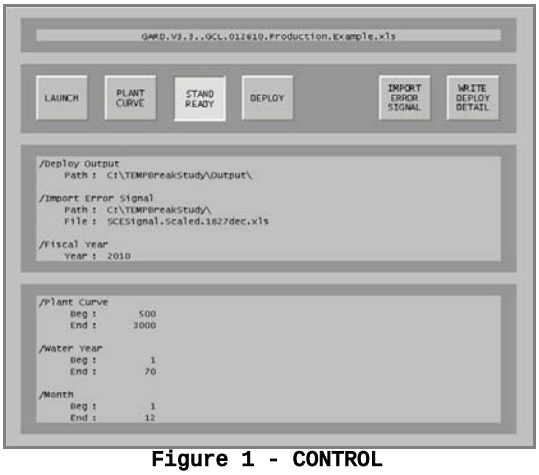


Figure 1 - CONTROL

II. Calculating Deployment Costs

Calculating deployment costs requires the running of the stand ready cost calculation. Stand ready costs are calculated and the resultant unit commitment and dispatch for the project are used as starting points when deploying reserves.

1 – Perform steps 1 through 6 in section I, Calculating Stand Ready Costs.

2 – Activate the DEPLOY button.

3 – If the user has a defined error signal, activate IMPORT ERROR SIGNAL. See Section IX for details regarding the preparation of the error signal file.

Note if the IMPORT ERROR SIGNAL button is not activated, a default signal is generated. The default signal is simply a random, normally distributed error signal.

4 – If it is desired to store the minute-to-minute detail from the deployment simulation, activate the WRITE DEPLOY DETAIL button.

5 – If the WRITE DEPLOY DETAIL button is activated, go to "Deploy Output" and specify the path where the output should be stored.

Note Writing out the deployment details increases an already long run time. Unless necessary, it is advisable to only write out specific periods of interest.

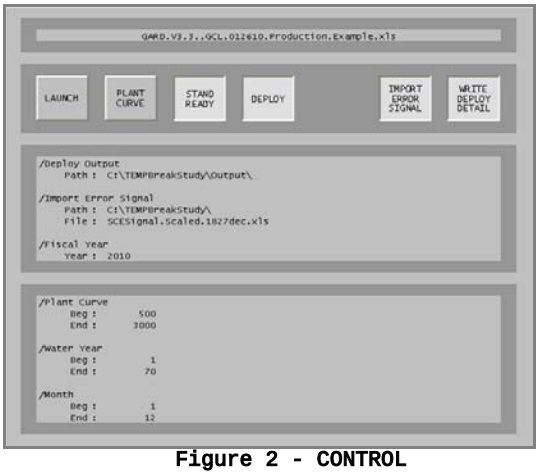


Figure 2 - CONTROL

6 – If the IMPORT ERROR SIGNAL button is activated, go to "Import Error Signal" and specify the path where the error signal file is stored as well as the name of the error signal file.

7 – Perform pre-launch check per Section IV.

Disclaimer This document and associated models are not final versions and are subject to change for development of the WP-12 Initial Proposal. Additionally, any data contained within the models or documentation are illustrative only and are not indicative of Initial Proposal development.

8 – Select the LAUNCH button.

9 – When the calculations are complete, a dialog box will appear indicating the starting and ending times of the run.

Note Calculating deployment costs for each of the 12 months in all 70 water years is best done as an overnight run due to the long processing time.

III. Generating Plant Curves

The GARD model may be used to generate plant efficiency curves. Generating the plant curves is not necessary for calculating either the stand ready or deployment costs. A PLANT CURVE run will provide the user with the necessary information to create a plant efficiency curve as well as detailed unit commitment, dispatch, and reserve information.

1 – Select the CONTROL tab.

2 – Activate the PLANT CURVE button.

Note If the STAND READY button is activated, it must be de-activated before PLANT CURVE may be selected.

3 – Under "Plant Curve," enter the starting generation after "Beg" and the ending generation after "End."

Note Beg >= Plant Minimum
End > Beg

End is automatically limited to the maximum powerhouse generation given the unit availabilities.

4 – Perform pre-launch check per Section IV.

5 – Select the LAUNCH button.

Note When running in PLANT CURVE mode, GARD uses the responses from the first water year for calculating the project's reserve obligation.

IV. Pre-Launch Check

Given the GARD model's long processing times, it is important to minimize the chance of having an input error disrupt the calculations. The following check list should help avoid such mishaps and the associated lost productivity.

1 – Select the CONTROL tab.

2 – Press the F9 key on the keyboard. This ensures any calculated values, such as the title on the control panel, are updated.

3 – Double check the button activation given the type of study being performed.

4 – Under "Deploy Output," double check the Path.

5 – Under "Import Error Signal," double check the Path

Note Be sure that the last character for a Path is a backslash (\). For example:
Correct: C:\Output\
Incorrect: C:\Output

6 – Under "Import Error Signal," double check that File contains the correct signal filename.

7 – Under "Fiscal Year," ensure the correct Year is entered.

8 – Under "Plant Curve," ensure that Beg is >= Plant Minimum.

9 – Under "Plant Curve," ensure that End > Beg.

10 – Under "Water Year," ensure that Beg <= End and that 1 >= Beg <= 70.

11 – Under "Water Year," ensure that End >= Beg and that 1 >= End <= 70.

12 – Under "Month," ensure that Beg <= End and that 1 >= Beg <= 12.

13 – Under "Month," ensure that End >= Beg and that 1 >= End <= 12.

14 – Select the INPUT (POWERHOUSE) tab.

15 – Under "POWERHOUSE DATA," check for correct UNMin values.

Note The correctness of the UNMin values is very important. Incorrect, i.e. too low, UNMin values can lead to a failure to solve in the DispatchSub().

16 – Ensure that UNMax> UNMin < UNPk.

17 – Ensure that UNMax >= UNPk.

18 – Ensure that the UNTagRef values for each unit family are correct.

19 – Ensure that the UNMMin values are with the bounds of the unit performance equations. For further detail see Section VI.

20 – Under "UNIT AVAILABILITY," double check entries for correctness.

Disclaimer This document and associated models are not final versions and are subject to change for development of the WP-12 Initial Proposal. Additionally, any data contained within the models or documentation are illustrative only and are not indicative of Initial Proposal development.

/COMMENT

/GRAND COULEE

/POWERHOUSE DATA

	/FAM1	/FAM2	/FAM3	/FAM4	/FAM5	/FAM6	/FAM7
/UNMin	55	30	325	325	0	0	0
/UNMax	115	115	701	796	0	0	0
/UNPk	95	93	684	686	0	0	0
/UN#	9	9	3	3	0	0	0
/UNTagRef	0	1	2	3	0	0	0
/UNMMin	1	1	1	1	0	0	0

/UNIT AVAILABILITY

	/OCT	/NOV	/DEC	/JAN	/FEB	/MAR	/APR	/MAY	/JUN	/JUL	/AUG	/SEP
/FAM1	6	6	7	8	8	7	8	7	8	7	7	6
/FAM2	7	7	7	8	8	8	7	8	7	7	8	6
/FAM3	3	3	2	2	2	2	2	2	3	3	3	3
/FAM4	2	2	3	3	3	3	3	3	2	2	2	2
/FAM5	0	0	0	0	0	0	0	0	0	0	0	0
/FAM6	0	0	0	0	0	0	0	0	0	0	0	0
/FAM7	0	0	0	0	0	0	0	0	0	0	0	0

/RESERVE OBLIGATION TOTALS

	/OCT	/NOV	/DEC	/JAN	/FEB	/MAR	/APR	/MAY	/JUN	/JUL	/AUG	/SEP
/REGINC	122	122	125	125	125	125	125	125	125	125	132	132
/FOLINC	386	386	400	400	400	404	404	404	404	404	433	433
/IMBINC	764	764	811	811	811	822	822	822	822	822	917	917
/REGDEC	125	125	128	128	128	129	129	129	129	129	135	135
/FOLDEC	400	400	413	413	413	416	416	416	416	416	441	441
/IMBDEC	1141	1141	1214	1214	1214	1230	1230	1230	1230	1230	1375	1375
/CRO	428	428	428	428	428	428	428	428	428	428	428	428

/RESPONSE PERCENTAGE

	/OCT	/NOV	/DEC	/JAN	/FEB	/MAR	/APR	/MAY	/JUN	/JUL	/AUG	/SEP
/RESP	70	70	70	70	75	75	75	75	75	75	75	75

Figure 3 - INPUT (POWERHOUSE)

21 – Under "RESERVE OBLIGATION TOTALS," double check entries for correctness.

Note All reserve entries are entered as positive values.

22 – Under "RESPONSE PERCENTAGE," double check entries for correctness.

Note Do not enter the response percentages as decimal values. For example, a 50 percent response is entered as:
Correct: 50
Incorrect: 0.50

23 – Select the INPUT (CALCULATED) tab.

24 – All values contained in this tab are calculated. Ensure that none of the values are error values.

25 – Take note of the values under "FLAG." All values should read, "OK." If a "WARNING" value appears, this indicates that the maximum powerhouse capacity is less than the maximum generation request.

26 – Select the INPUT (GENERATION) tab.

27 – Ensure the generation table is populated.

/TIME	/OCT	/NOV	/DEC	/JAN	/FEB	/MAR	/APR	/MAY	/JUN	/JUL	/AUG	/SEP
BOM	10/1/2009	11/1/2009	12/1/2009	1/1/2010	2/1/2010	3/1/2010	4/1/2010	5/1/2010	6/1/2010	7/1/2010	8/1/2010	9/1/2010
EOM	10/31/2009	11/30/2009	12/31/2009	1/31/2010	2/28/2010	3/31/2010	4/30/2010	5/31/2010	6/30/2010	7/31/2010	8/31/2010	9/30/2010
WKDAY	22	21	23	21	20	23	22	21	22	22	22	22
SAT	5	4	4	5	4	4	4	5	4	5	4	4
SUN	4	5	4	5	4	4	4	5	4	4	5	4
HLH	496	480	496	496	448	496	480	496	480	496	496	480
LLH	248	240	248	248	224	248	240	248	240	248	248	240
DAYS	31	30	31	31	28	31	30	31	30	31	31	30
/MAX PH	5190	5190	5400	5630	5630	5515	5515	5515	5420	5305	5420	5075
/MAX LD	3471	3920	4858	4680	4744	3683	4021	4667	4947	5054	5048	4177
/FLAG	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK

Figure 4 - INPUT (CALCULATED)

28 – Select the INPUT (PREDEC) tab.

29 – Ensure the predec table is populated.

V. Updating Unit Performance Functions

Periodically, it may be necessary to update the unit performance functions. The unit performance functions relate a unit's loading in MW to its fuel consumption in KCFS. The process involves estimating function parameters and hardcoding the parameters into the GARD model's source code. Function parameters are estimated for each unit family at a given

Disclaimer This document and associated models are not final versions and are subject to change for development of the WP-12 Initial Proposal. Additionally, any data contained within the models or documentation are illustrative only and are not indicative of Initial Proposal development.

hydroelectric project. Several operating limits are derived from the unit performance functions. Those limits include the economic minimum generation, the peak efficient generation and the maximum generation. Additional constraints may exist, such as cavitation related limits. The following procedure assumes that the function parameters for the relevant unit families have been calculated.

Note GARD expects the functions to relate KCFS of turbine flow to MW/100 of unit loading.

1 – Go to the VBA editor (Alt + F11 from Excel).

2 – Go to the "SubData" module. Double-clicking on the module from the Project window will open up the "SubData" module.

3 – Find ReadInput(). Update the function parameters for each unit family. Function parameters are specified as b1 through b9 where b9 is the constant regardless of polynomial order.

4 – Find the array bx(). This array is a two dimensional array. The first dimension contains the parameter estimates of the unit performance curve. The unit performance functions are specified as follows:

f(x) = b1 x + b2 x^2 +b3 x^3 + . . .+ b7 x^7 + b9.

The second dimension references the unit family number. The first unit family reference is 0, the second 1, etc.

Example A unit performance function for the first unit family, array element 0, fitting a third order polynomial would be input as follows:

bx(1, 0) = 4.44218431952663
bx(2, 0) = -1.75611282051282
bx(3, 0) = 1.1047
bx(4, 0) = 0#
bx(5, 0) = 0#
bx(6, 0) = 0#
bx(7, 0) = 0#
bx(8, 0) = 0#
bx(9, 0) = 0.303032577251807

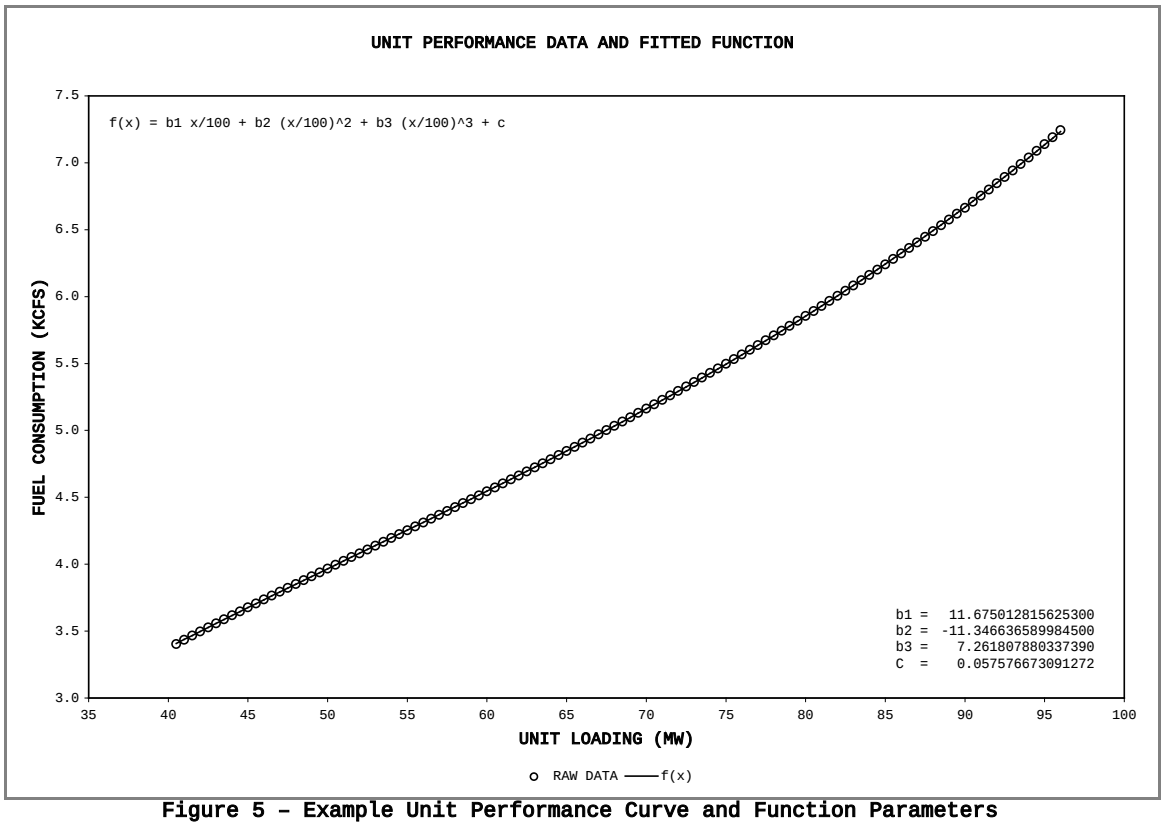


Figure 5 – Example Unit Performance Curve and Function Parameters

VI. Updating Unit Limits

Anytime unit performance functions are updated, the unit limits must also be updated. Most unit limits are calculated off of the unit performance function. The exception is the unit minimum generation. The unit minimum may have a minimum generation higher than would be calculated as an economic minimum. Unit level minimums should be confirmed with the project operators. Insight into unit level minimums may also be had from studying historical unit level dispatch.

Note In order to expedite the calculation of the unit limits, a locally created spreadsheet should be used. Copying the "FunctionFamilyKCFS," "FunctionFamilyKCFSperMW," and "NewtonRaphonFamily" modules from the GARD model and into the local spreadsheet will help facilitate the necessary calculations. The following procedure assumes a local spreadsheet with the relevant functionality has been built.

Disclaimer This document and associated models are not final versions and are subject to change for development of the WP-12 Initial Proposal. Additionally, any data contained within the models or documentation are illustrative only and are not indicative of Initial Proposal development.

1 – Calculate the economic minimum generation for each unit family. This is done by finding the generation associated with the minimum marginal fuel cost for each unit family. The minimum marginal fuel cost is the unit loading where the first derivative of the marginal cost function equals zero.

The function NewtonRaphsonKprime() may be used to directly calculate economic minimum generation. See Section XIII, for more information on using the NewtonRaphon…() functions.

Alternatively, the function Fofx2PrimeK(x, Tag As Integer) may be used calculate the change in marginal cost as a function of generation. The generation resulting in an Fofx2PrimeK(x, Tag As Integer) value of zero is the economic minimum generation.

Note Fofx2PrimeK(x, Tag As Integer) is the second derivative of the unit performance function. The first derivative of the unit performance function, FofxPrimeK(x, Tag As Integer), yields the marginal fuel cost. Thus, the second derivative of the unit performance function is the first derivative of the marginal cost function.

Compare the calculated economic minimum to the unit minimum. Select the higher of the two values as UNMin.

2 – In the GARD model, select the INPUT (POWERHOUSE) tab. Enter the calculated unit minimum values for each unit family in the UNMin row under POWERHOUSE DATA.

3 – Under POWERHOUSE DATA, row UNMax, enter the capacity value for each unit family. The capacity value is equal to the maximum generation value observed in the raw unit performance data.

4 – Using the locally created spreadsheet, calculate the peak efficient generation values for each unit family. As with calculating the economic minimum generation, this calculation may be performed multiple ways.

The function NewtonRaphsonKperMW() may be used to directly calculate the peak efficient generation. See Section XIII, for more information on using the NewtonRaphon…() functions.

Note All FofXKperMW…(x, Tag As Integer) functions use the same parameter estimates as calculated in Section V. Ensure the parameter estimates are indeed the same in the code before proceeding.

Alternatively, the function FofXKperMWPrime(x, Tag As Integer) may be used calculate the change in the marginal KCFS/MW as a function of generation. The generation resulting in FofXKperMWPrime(x, Tag As Integer) value of zero is the peak efficient generation.

5 – In the GARD model, select the INPUT (POWERHOUSE) tab. Enter the calculated peak efficient generation values for each unit family in the UNPK row under POWERHOUSE DATA.

6 – Under POWERHOUSE DATA, row UNMMin, enter the lowest observed generation value for each unit family. The lowest observed generation value is equal to the lowest generation value observed in the raw unit performance data.

Note Certain projects at times of the year are required to operate within 1% of peak efficiency. Implementing this constraint requires an alternative calculation for the UNMin and UNMax values. These values must be calculated and input into the POWERHOUSE DATA tab constrained by the 1% peak efficiency requirement. This constraint is a feature of LCOL and LSN projects in the spring and summer. Generation capability beyond the upper 1% cannot be counted toward available spinning capability.

The following procedure outline the additional steps required when considering 1% peak efficiency limits.

1 – In a locally created spreadsheet, calculate the MW/KCFS for associated with the peak efficient generation using the FofXKperMW(x, Tag As Integer) function.

2 – Multiply the value obtained in step 1 by 1.01.

3 – Calculate the upper 1% limit. Using the NewtonRaphsonKperMWTarget() function, set the Target value equal to the value obtained in step 2. Set the x value to a value close to the unit's maximum generation.

4 – In tab POWERHOUSE DATA, row UNMMax, enter the value obtained in step 3.

5 – Calculate the lower 1% limit. Using the NewtonRaphsonKperMWTarget() function, set the Target value equal to the value obtained in step 2. Set the x value to a value close to the unit's economic minimum generation.

6 - In tab POWERHOUSE DATA, row UNMMin, enter the value obtained in step 5

Note During the normal course of running and using GARD, unit limits do not need updating or tuning. Unit limits only need updating when unit performance curves are updated and/or related unit level constraints change.

Disclaimer This document and associated models are not final versions and are subject to change for development of the WP-12 Initial Proposal. Additionally, any data contained within the models or documentation are illustrative only and are not indicative of Initial Proposal development.

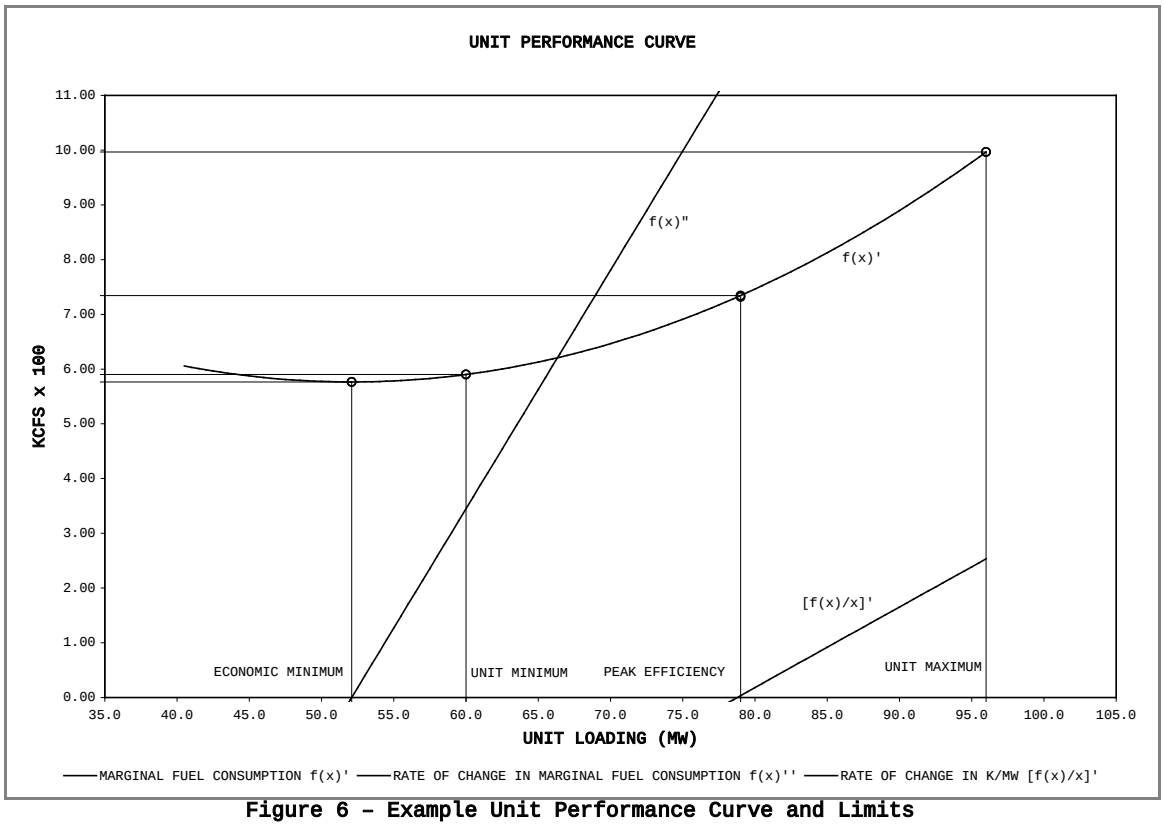


Figure 6 – Example Unit Performance Curve and Limits

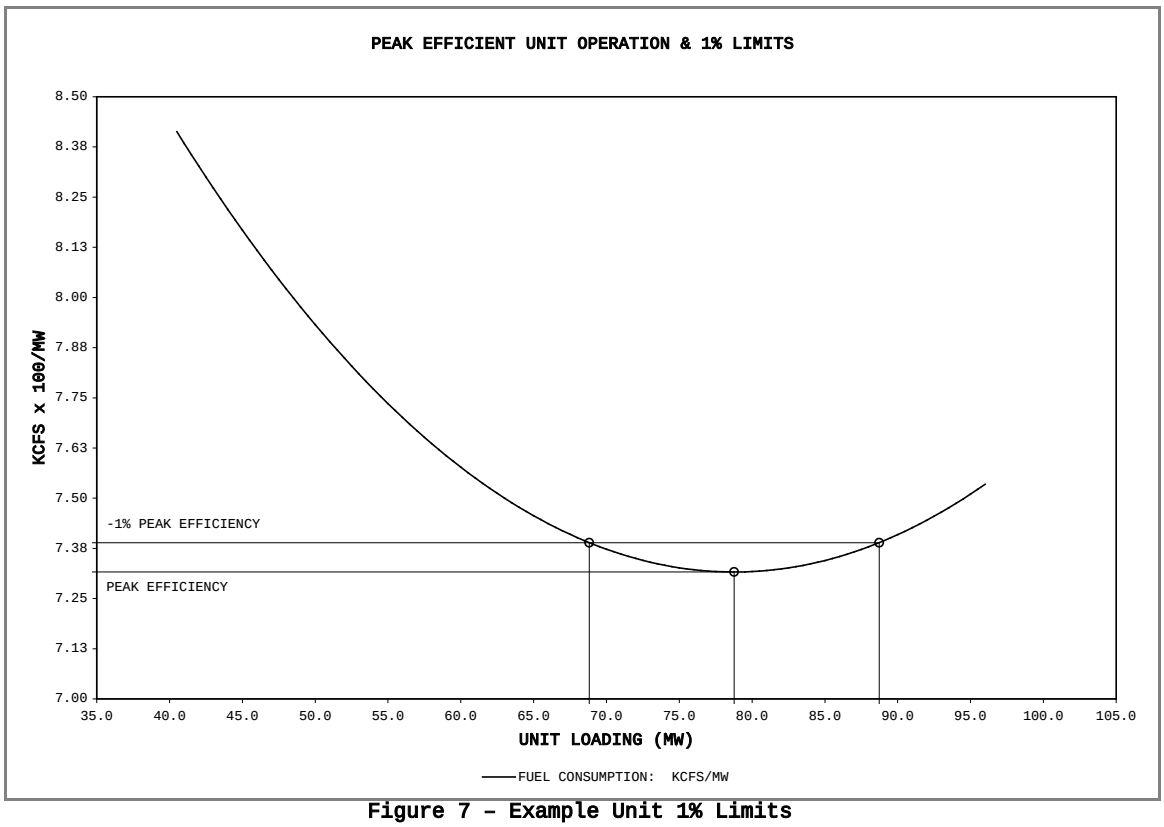


Figure 7 – Example Unit 1% Limits

.

Disclaimer This document and associated models are not final versions and are subject to change for development of the WP-12 Initial Proposal. Additionally, any data contained within the models or documentation are illustrative only and are not indicative of Initial Proposal development.

VII. INPUT (GENERATION)

The INPUT (GENERATION) tab contains a table of generation requests for the project to meet. The table is 12 months by 140 rows; where each row represents the HLH and LLH generation request for up to 70 water years. The first row contains the HLH generation for the first water year. The second row contains the LLH generation for the first water year. This sequence repeats for remaining rows of data. The generation table should contain generation values calculated without carrying reserves. Any change in generation due to the need to carry reserves is calculated by GARD and recorded as a cost of carrying the reserve.

The generation values originate from an external data source or direct user input.

Note For a full 70 year model run, the generation values are typically rooted in a 70 year HYDSIM run.

VIII. INPUT (PREDEC)

The INPUT (PREDEC) tab contains a table of dec capability available to the project in the base case. The table is 12 months by 140 rows; where each row represents the HLH and LLH generation request for up to 70 water years. The first row contains the HLH capability for the first water year. The second row contains the LLH capability for the first water year. This sequence repeats for remaining rows of data. The table should contain dec values calculated without carrying reserves. Any additional dec capability needed to carry reserves is calculated by GARD and recorded as a cost of carrying the reserve.

Note HLH value for the PREDEC table may be zero. PREDEC tables influence the need to shift energy into the LLH and graveyard periods in order to increase available nighttime dec carrying capability. Because GARD will never reverse load factor the project, HLH PREDEC values are not necessary.

IX. Station Control Error File

The station control error (SCE) file contains inc and dec signals for each minute of each month being studied. The data contained in the SCE file should be provided at the system level, i.e. the data represent the total error on within the BAA in need of balancing. As the deployment of reserves is being modeled, the SCE is allocated to a given project based on the project's response setting; where the response setting is an allocation of the total SCE to a given project denominated as a percentage (see Section IV, step 22).

1 – Open a blank Excel spreadsheet.

2 – Populate cells in the range A1:L44640 with desired signal data. Each column represents one month beginning with October. Each row represents one minute within the given month.

3 – Save the signal file to desired directory and close the file.

4 – Enter the signal file name and directory into the CONTOL tab per Section IV.

Note The error signal file is what the projects will control to. Ensure that positive values are associated with the need to increase generation and negative values are associated with the need to decrease generation.

X. OUTPUT

The OUTPUT tab contains tables showing the details of the project's unit commitment and dispatch. This tab is populated when the stand ready impacts are calculated per Section I as well as when the deployment impacts are calculated per Section II.

There are four data groups contained in the tab. The first are the data associated with the base case. These data are the starting point from which all impacts are calculated. The next data group contains project information associated with carrying dec reserves. The third data group contains project information associated with carrying both dec and non-spinning reserves. The fourth and final data group contains project information associated with carrying dec, non-spinning and spinning reserves. The tab contains all the necessary data to calculate the stand ready impacts of carrying reserves.

a – Rows designate the month, water year and diurnal period. For example, the row label "/OCT 1929 H" indicate that the results are for the month of October, water year 1929, and the HLH time period.

b – Column "/FAM1 C" through "/FAM7 C" indicate the unit commitment by unit family. Although only 4 slots are currently used, up to seven unit families may be modeled.

c – Column "/TOTGEN" shows the total generation in MW.

d – Column "/FAM1 D" through "/FAM7 D" indicate the dispatch in MW by unit family. Although only 4 slots are currently used, up to seven unit families may be modeled. The sum of "/FAM1 D" through "/FAM7 D" equals column "/TOTGEN."

e – Column "/TARGET" indicates the marginal cost is KCFS x 100. In an ideal dispatch, all unit families are running to the same marginal cost.

f – Column "/KCFS" indicates the total turbine flow for all unit families currently committed and dispatched.

g – Column "/Spin" indicates the available spinning reserve for the current unit commitment and dispatch.

Disclaimer This document and associated models are not final versions and are subject to change for development of the WP-12 Initial Proposal. Additionally, any data contained within the models or documentation are illustrative only and are not indicative of Initial Proposal development.

h – Column "/NONSPIN" indicates the available non-spinning reserve given the current unit commitment, dispatch, and unit availability.

i – Cells A1:T1692 contain the output data for the "/BASE" case. These data are the basis from which all impacts are measured.

j – Cells V1:AO1692 contain the output data for the "+/DEC" case. These data contain any changes in unit commitment, dispatch, total generation and turbine flow due to carrying the dec reserve.

k – Cells AQ1:BJ1692 contain the output data for the "+/NONSPIN" case. These data contain any changes in unit commitment, dispatch, total generation and turbine flow due to carrying the dec and non-spinning reserve.

l – Cells BL1:CE1692 contain the output data for the "+/SPIN" case. These data contain any changes in unit commitment, dispatch, total generation and turbine flow due to carrying the dec, non-spinning, and spinning reserve.

XI. OUTPUT()

The OUTPUT() tabs contain the detailed results denominated in MWh of gains and losses due to carrying and deploying reserves. The tables show the gains and losses for each month, water year and diurnal time period. From the data contained in these tables, the impact of carrying and deploying reserves may be calculated by month, diurnal period, inc provision and dec provision.

The following list shows the grouping of tables by stand ready and deployment impacts:

Stand Ready

1. OUTPUT (ESHIFT DEC)
2. OUTPUT (ESHIFT NS)
3. OUTPUT (ESHIFT S)
4. OUTPUT (EFF LOSS DEC)
5. OUTPUT (EFF LOSS NS)
6. OUTPUT (EFF LOSS S)
7. OUTPUT (CYC LOSS DEC)
8. OUTPUT (CYC LOSS NS)
9. OUTPUT (CYC LOSS S)
10. OUTPUT (SPL LOSS NS)
11. OUTPUT (SPL LOSS S)

Note All stand ready impacts (ESHIFT, EFF LOSS, CYC LOSS, and SPL LOSS) are reported on a cumulative basis. For example, OUTPUT(ESHIFT S) records the cumulative effect of standing ready to provide dec, non-spinning, and spinning reserves. OUTPUT(ESHIFT NS) records the cumulative effect of standing ready to provide dec and non-spinning reserves. If one desires to isolate the impact of standing ready to provide spinning, the results contained in OUTPUT(ESHIFT NS) need to be subtracted from OUTPUT(ESHIFT S).

Energy shift associated with spinning: OUTPUT(ESHIFT S) – OUTPUT(ESHIFT NS)
Energy shift associated with non- spinning: OUTPUT(ESHIFTN S) – OUTPUT(ESHIFT DEC)
Energy shift associated with dec: OUTPUT(ESHIFTN DEC)

Deployment

1. OUTPUT (DEP RESP INC)
2. OUTPUT (DEP RESP DEC)
3. OUTPUT (DEP CYC INC)
4. OUTPUT (DEP CYC DEC)
5. OUTPUT (DEP SPL DEC)

Note Deployment impacts, unlike stand ready impacts, are not recorded on a cumulative basis. The impacts are recorded on a stand alone basis.

XII. OUTPUT (SETUP)

The OUTPUT(SETUP) tab contains the same type and format of output data as described in Section IX, except the project generation and associated unit commitment and dispatch are calculated over a user specified range of generation in 1 MW increments. From this data, the user can calculate the efficiency curve for the project as well as other comparative information, such changes in unit commitment and available spinning reserve. OUTPUT(SETUP) is populated per the procedure in Section III. The results are shown for a base case without a reserve obligation as well as a case with the input spinning reserve obligation. Comparing the HK (total generation / total turbine flow) values between the base and reserve case reveals, for a given range of generation and unit availability, where the project realizes efficiency losses for carrying a given level of spinning reserve.

Figure 8 shows a plot of generation versus project HK for a base case and a case where a minimum level of spinning reserve is being requested. For this particular example, it is observable that efficiency losses associated with standing ready to carry a minimum reserve are realized in plant loading ranging from approximately 200 to 500 MW. Loadings beyond 500 MW have sufficient spinning capability as a natural byproduct of an efficient unit commitment and dispatch.

Disclaimer This document and associated models are not final versions and are subject to change for development of the WP-12 Initial Proposal. Additionally, any data contained within the models or documentation are illustrative only and are not indicative of Initial Proposal development.

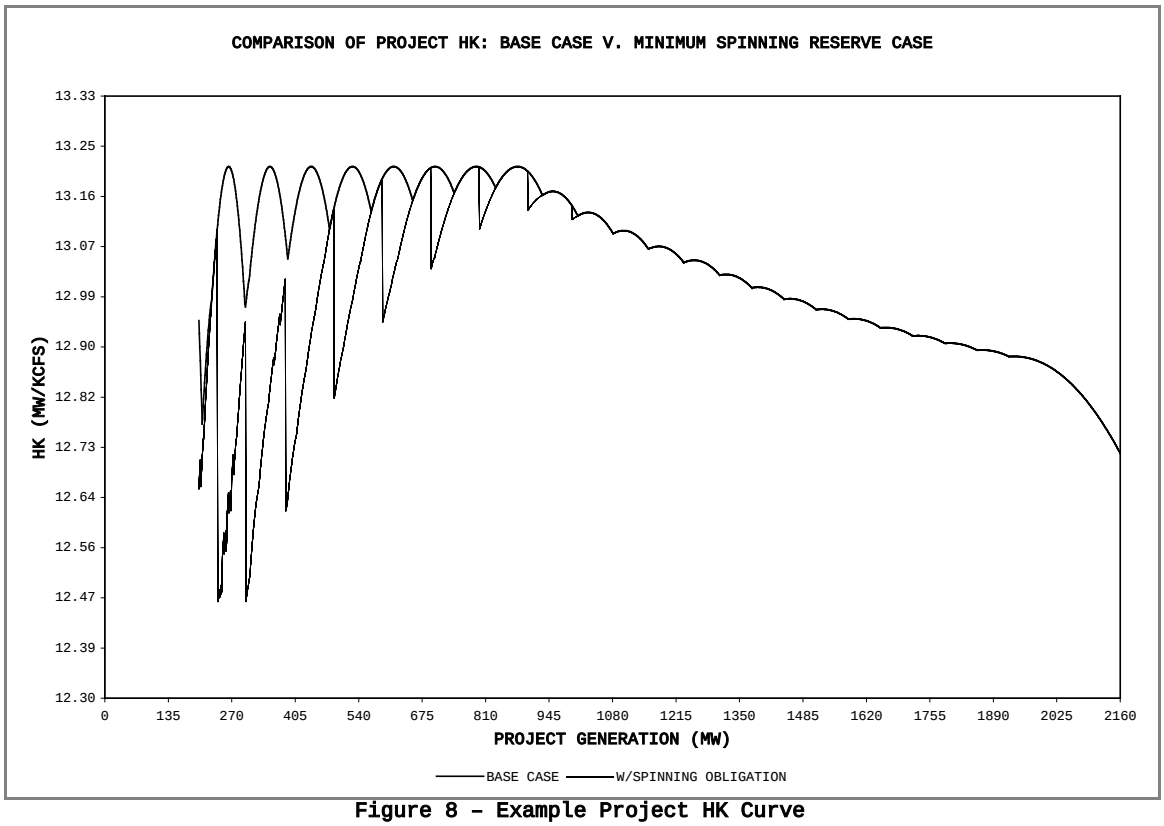


Figure 8 – Example Project HK Curve

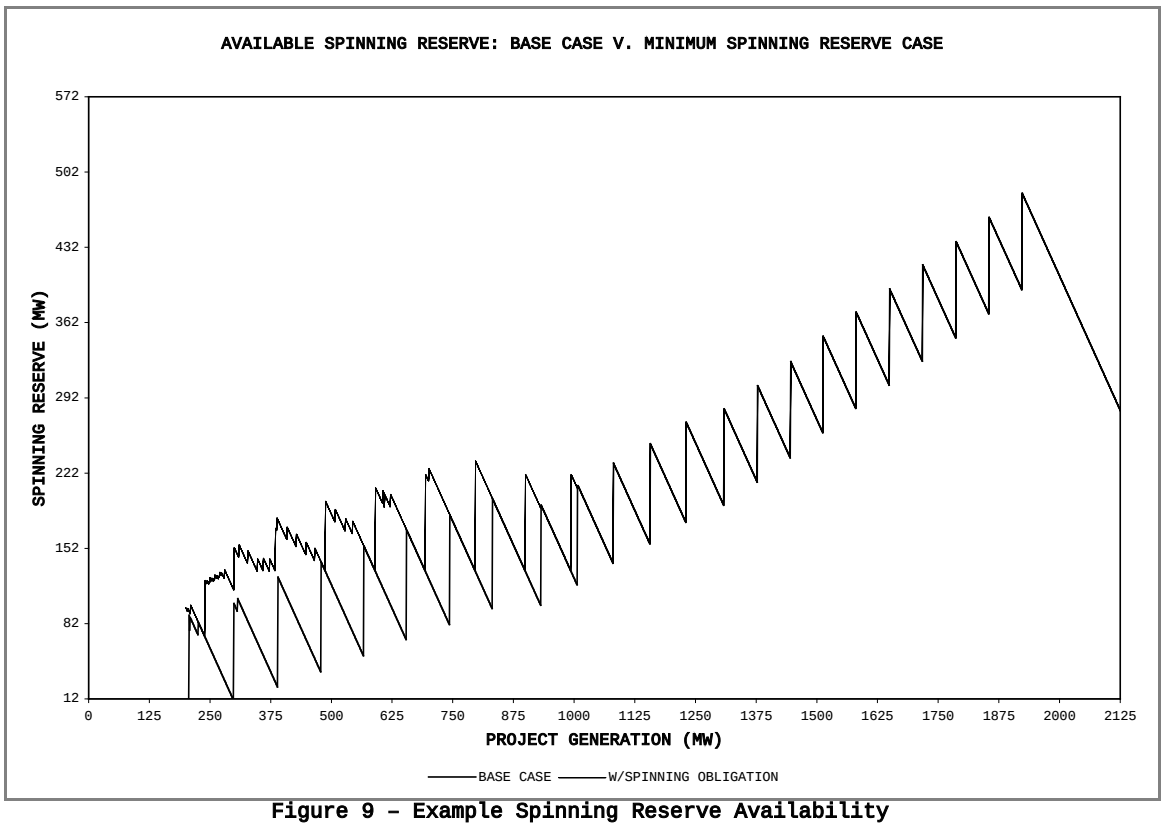


Figure 9 – Example Spinning Reserve Availability

Disclaimer This document and associated models are not final versions and are subject to change for development of the WP-12 Initial Proposal. Additionally, any data contained within the models or documentation are illustrative only and are not indicative of Initial Proposal development.

For the same example scenario, Figure 9 shows the project generation versus the available spinning reserve. The gap between the base case and the minimum reserve case correspond to the efficiency differences notes in Figure 8. The data contained in OUTPUT(SETUP) are not explicitly used in any of the study calculation for stand ready and deployment impacts, but are useful for aiding in explaining observed phenomena.

XIII. Newton-Raphson Functions

The families of Newton-Raphson functions all operate identically. The differences between functions are simply the hardcoded references to the input functions and their derivative functions. The Newton-Raphson function is at the core of the GARD model's unit dispatch algorithm and is key to finding dispatch solutions to non-quadratic marginal cost functions. In order to minimize the fuel cost for a given level of generation, the all committed units are dispatched such that their marginal costs are equal.

All of the Newton-Raphson functions are of identical syntax. The syntax of the function is as follows:

NewtonRaphson…(x, Target, Tag, Prec, MaxIt, Default)

where;

x:	the initial x value guess for f(x) where the calculation begins.
Target:	the target value y for f(x).
Tag:	an integer value indicating the unit family.
Prec:	the desired precision.
MaxIt:	the maximum number of iterations.
Default:	user specified default value should calculation fail.

The following are the various versions of the Newton-Raphson function used either in GARD directly or available for use in a locally created spreadsheet:

Newton-Raphson Function Definitions

- **NewtonRaphsonK()**
  Returns the generation associated with a target marginal cost.
- **NewtonRaphsonKperMW()**
  Returns the peak efficient generation. Target is set to 0 (when f(x)' = 0, f(x) is at its minimum).
- **NewtonRaphsonKperMWTarget()**
  Returns the generation associated with a targeted KCFS/MW value.

Note The Newton-Raphson functions are designed as array formulas. If used in a spreadsheet as an array formula, the function returns the generation for the given target, the number of iterations required to converge, and an integer value of 0 if the calculation was successful or a value of 1 if the maximum number of iterations was reached.

XIV. Optimal Unit Dispatch

The GARD model optimizes the unit dispatch by loading each online unit such that the marginal cost of each unit is identical and the requested generation and reserve is met. Dispatching units at equal marginal costs results in the model meeting the objective of minimizing total turbine outflow per unit of fuel (water in KCFS). This objective function changes if the generation request and the reserve requirement become mutually exclusive. Under this circumstance, the objective function becomes maximize reserves given the generation request.

$$\text{minimize} \sum_{i=1}^{n} C_i f(x_i)$$

Equation 1 - Objective Function

where

n is the number of unit families at a project,

C_i is the number of committed units in the ith family,

$f(x_i)$ is the per unit turbine flow for a unit in the ith family given a per unit dispatch x_i.

For a given unit commitment, GARD dispatches the units such that the generation request is met while setting the marginal cost of the online units equal. Synchronizing the marginal cost of the online units results in minimizing the turbine flow per unit of generation at the entire project. A simple two unit example is shown in Figure 10. In Figure 10, the goal is to minimize the water consumed in the process of generating 170 MW with two units committed. The two x-axis show the generation for each of the respective units. The sums of the values along each point on the primary and secondary x-axis total the required 170 MW of generation. The primary y-axis indicates the marginal outflow of each turbine for a given unit loading. The secondary y-axis show the total water consumed in generating 170 MW as a function of the combined loading of unit 1 and unit 2.

Disclaimer This document and associated models are not final versions and are subject to change for development of the WP-12 Initial Proposal. Additionally, any data contained within the models or documentation are illustrative only and are not indicative of Initial Proposal development.

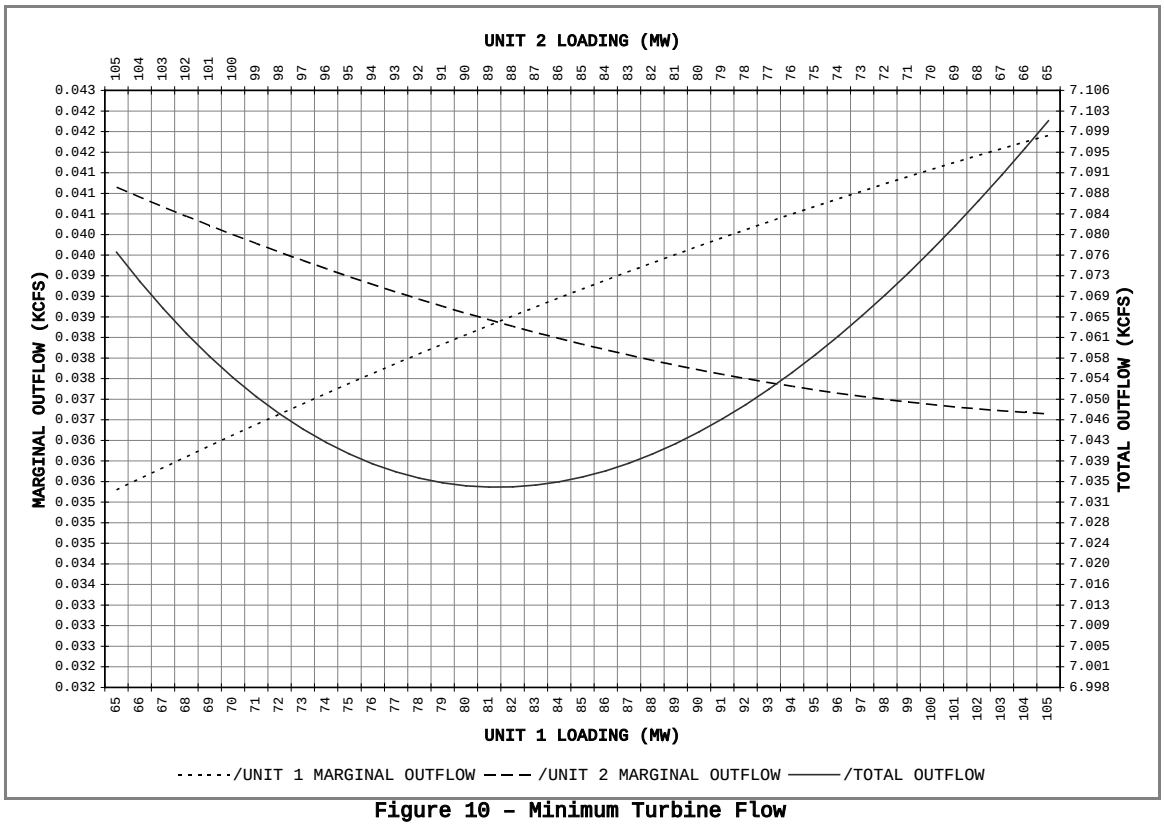


Figure 10 – Minimum Turbine Flow

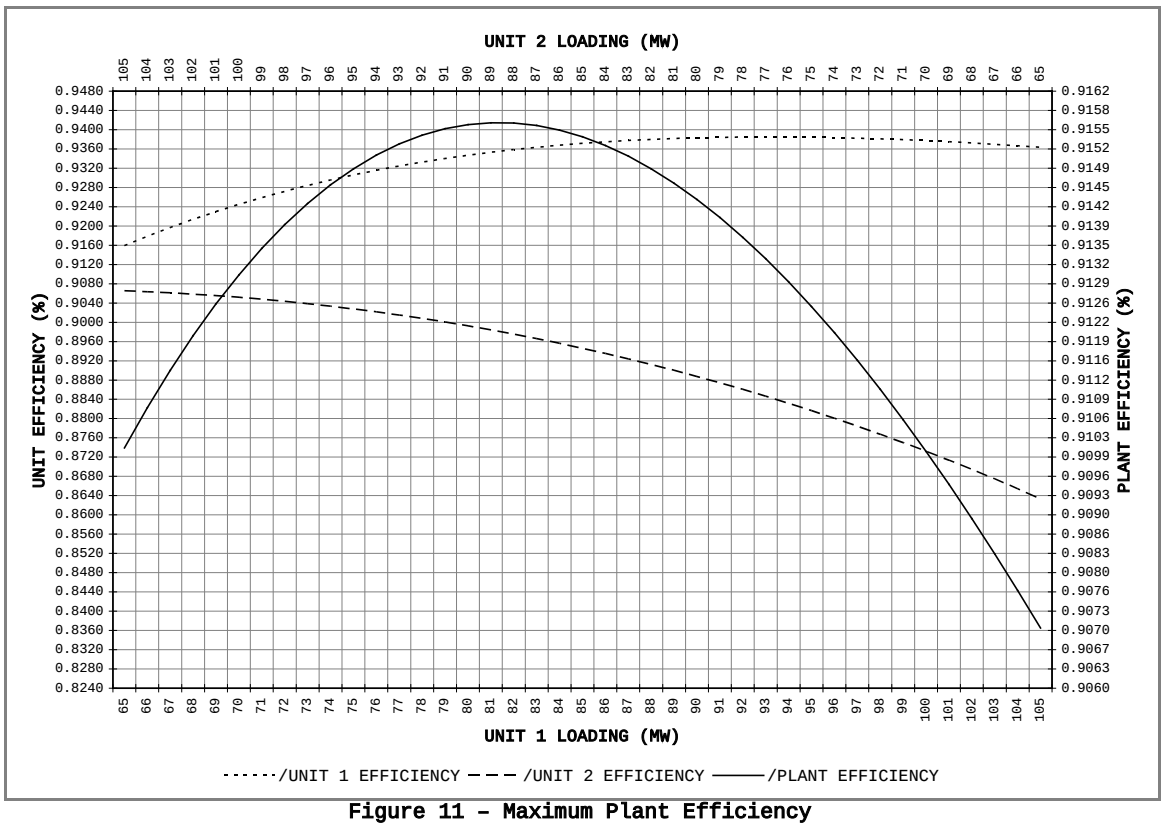


Figure 11 – Maximum Plant Efficiency

Disclaimer This document and associated models are not final versions and are subject to change for development of the WP-12 Initial Proposal. Additionally, any data contained within the models or documentation are illustrative only and are not indicative of Initial Proposal development.

Observing Figure 10 shows that total project outflow is minimized where the marginal cost curves of units 1 and 2 intersect. When turbine flow per unit of generation is minimized, the efficiency of the plant is maximized. However, it does not necessarily hold true that the individual units are being dispatched at their maximum efficiency.

Figure 11 is another representation of the results shown in Figure 10. Rather than showing the results in terms of total project outflow and marginal cost, the same results are shown comparing the individual unit efficiencies against the total plant efficiency.

The optimal dispatch is solved using either the Newton-Raphson method or, if the marginal cost function is quadratic, the quadratic equation. The Newton-Raphson method is an iterative process relying on the fact that the unit performance functions are differentiable functions of x, where x is the unit loading for a given unit family. See Section V for a description of the unit performance functions.

$$x_{i+1} = x_i + (f(x_i)' - mc) / f(x_i)''$$

Equation 2 - Newton-Raphson Method

where

x_i is the unit loading for a given unit family for the ith iteration,

$f(x_i)'$ is the marginal cost function of a given unit family ($f(x_i)$ = total cost),

$f(x_i)''$ is the first derivative of the marginal cost function of a given unit family,

mc is the target marginal cost.

Successive passes are made through the above equation until the value $f(x_i)'$ equals the value of mc +/- a tolerance.

Figure 12 shows graphically the Newton-Raphson concept. Given a marginal outflow (fuel cost) target, the associated generation needs to be calculated. Starting with a guess, 100 MW in the case of Figure 9, the first derivative of the marginal cost function is calculated. The derivative is used to estimate the change in generation required to achieve the desired marginal outflow. The new generation value becomes the next best estimate. The derivative is again used to calculate the change in generation needed to reach the targeted marginal outflow. This process is repeated until the calculated marginal outflow is sufficiently close to the desired marginal outflow.

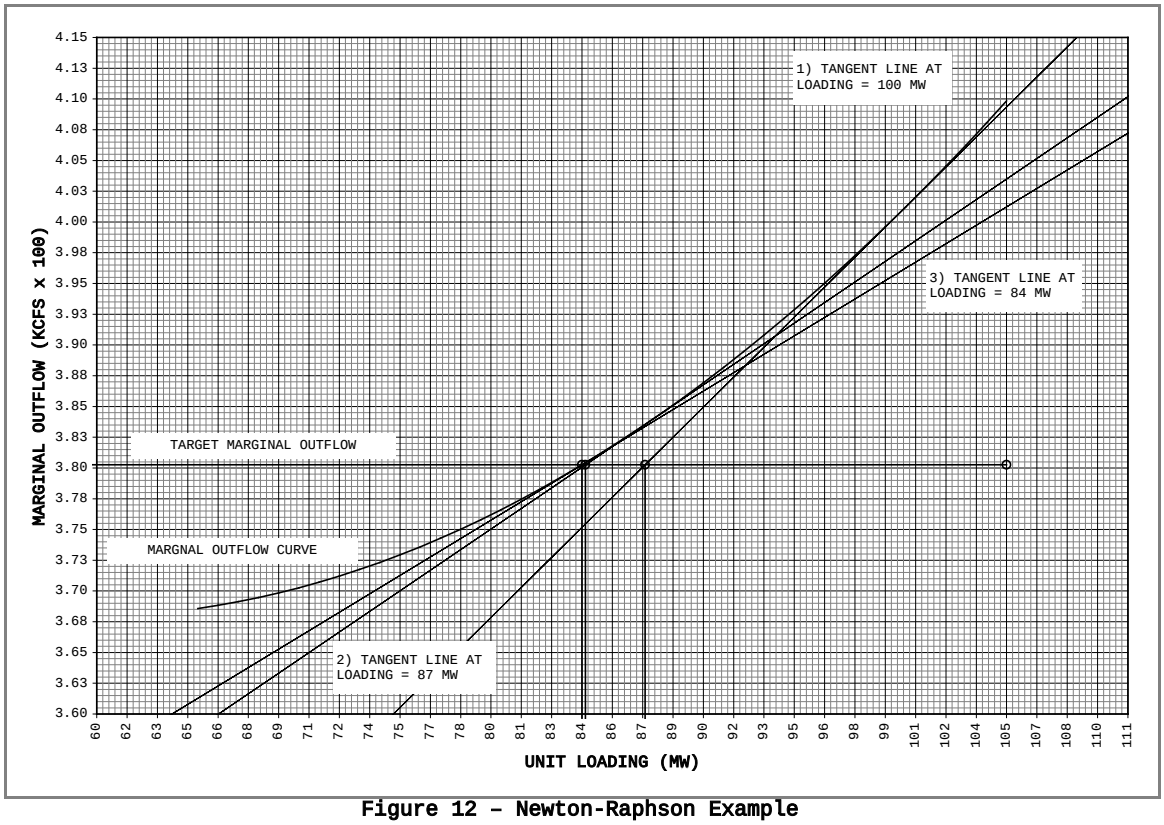


Figure 12 – Newton-Raphson Example

Disclaimer This document and associated models are not final versions and are subject to change for development of the WP-12 Initial Proposal. Additionally, any data contained within the models or documentation are illustrative only and are not indicative of Initial Proposal development.

If f(x)' is quadratic, GARD will solve for the optimal dispatch using the quadratic equation. GARD toggles between the Newton-Raphson method and the quadratic equation in order to save on run time.

$$x = b +/- \sqrt{b^2 - 4a(c - mc)}/2ac$$

Equation 3 - Quadratic Equation

where

x is the per unit loading for a given unit family and,

mc is the target marginal cost.

Note The Newton-Raphson routine is used to optimize dispatch for a given unit commitment and generation request when unit performance functions cannot be solved by the quadratic equation. Regardless of method used, Newton-Raphson or quadratic equation, the objective function remains the same: minimize total project turbine flow.

XV. The Reserve Obligation

The determination of the quantities of spinning versus the quantities of non-spinning is derived from the NERC requirements as well as system operator judgment. NERC requires that at least 50 percent of the BAA contingency reserve obligation is capable of being met with spinning capability responsive to AGC. NERC requires that 100 percent of the BAA regulating reserve must be carried on units with spinning capability responsive to AGC.

In contrast, the reserve categories of following reserve and imbalance reserve do not have NERC-defined criteria. Lacking NERC criteria, it is assumed that at least 50 percent of the inc following reserve must be carried as a spinning obligation and up to 50 percent as a non-spinning obligation. For imbalance reserve, up to 100 percent of the inc obligation may be met with non-spinning capability.

The rationale for carrying at least 50 percent of the inc following requirement as spinning is to provide sufficient response over the first five minutes of movement while simultaneously providing enough time to synchronize non-spinning units and ramp the units through their rough zones. Synchronization generally takes about three minutes, with the unit fully ramped in over the next seven minutes. Should additional reserves be required to cover a growing imbalance, additional units are synchronized and ramped as the following reserve is consumed and the imbalance reserve is deployed with non-spinning capability. By definition, all dec reserves (the dec portion of the regulating reserve, following reserve and imbalance reserve) are spinning, because units must be generating in order to deploy dec reserves.

Note The reserve obligation values are always input into the model as a positive value. Sign convention is handled in the model's code.

XVI. Efficiency Loss Detail

All forms of efficiency loss, whether associated with stand ready or deployment, are calculated as impacts to the project's HK. Although called efficiency loss through this document and code, they are more truly efficiency changes. On average, carrying and deploying reserves results in a net decrease in efficiency. As a result, the term efficiency loss is generally used to describe the efficiency changes.

$$dn = (HK_b - HK_a)\ Q_t\ h$$

Equation 4 - Change in Project Efficiency

where

HK_a is the starting production coefficient in MW/KCFS,

HK_b is the ending production coefficient in MW/KCFS,

Q_t is the total project turbine outflow, and

h is the duration in hours (or fraction of hours in the case of minutes).

XVII. Cycling Loss Detail

Cycling losses are realized when units are synchronized and ramped. Synchronization loss is the water lost while the units are brought up to grid frequency. Ramping losses occur as the units are loaded up to their requested generation or taken off-line from its current generation. In the process of bringing a unit up to its desired loading, the unit moves through inefficient zones resulting in excess consumption of water.

The water lost during synchronization is calculated as 10% of full gate flow for 3 minutes.

Disclaimer This document and associated models are not final versions and are subject to change for development of the WP-12 Initial Proposal. Additionally, any data contained within the models or documentation are illustrative only and are not indicative of Initial Proposal development.

$$s_{MWH} = 0.10\ f(x)|x_{max}\ b/f(x)|b\ h$$

Equation 5 - Synchronization Loss

where

f(x) is the unit performance function (see Section V),

b is the desired unit loading.

h is the duration of synchronization in hours (typically 3/60).

Ramping typically takes 7 minutes. Over 7 minutes, it is assumed that the unit linearly progresses from its minimum loading up to the desired dispatch. The ramp loss begins with an evaluation of the water consumed during the 7 minute ramp.

$$r = 1/(b-a)\int_a^b f(x)dx$$

Equation 6 - Average Water Consumption Over Ramp

where

$$\int_a^b f(x)dx = G(b) - G(a)$$

Equation 7 - Evaluate Anti-Derivative

where

G() is the anti derivative of the unit performance function and,

a is the starting unit loading at the beginning of the ramp,

The MWH of lost generation during the ramp is calculated as:

$$r_{MWH} = ((b+a) / 2r - b / f(x)|b)\ r\ h$$

Equation 8 - Ramping Loss

where

h is the duration of the ramp in hours (typically 7/60).

XVIII. Unit Commitment Filter

The run times associated with the optimal unit commitment and dispatch algorithms may be improved in some cases with a filter to quickly eliminate unit commitment combinations that, although feasible, would produce results so far from optimal that they are not even worth considering. Using available spinning reserves as a proxy for plant efficiency, where excessive spinning reserves are associated with a sub-optimal unit commitment and dispatch, a filter may be activated that automatically discards unit commitments resulting in excessive spinning capability. This filter seems especially effective when performing analysis for GCL, where there exists many unit commitment combinations for a given generation request along with a wide range of resultant spinning reserve. To date, implementing a similar filter at other projects can result in increased run time because the additional computation time required to implant a dynamic filer is greater than the time saved in avoided unit commitment and dispatch calculations.

Note During the normal course of running and using GARD, this filter does not need tuning. Only when unit performance curves are changed and/or unit availabilities change should the need arise to revisit the filter occur.

Disclaimer This document and associated models are not final versions and are subject to change for development of the WP-12 Initial Proposal. Additionally, any data contained within the models or documentation are illustrative only and are not indicative of Initial Proposal development.

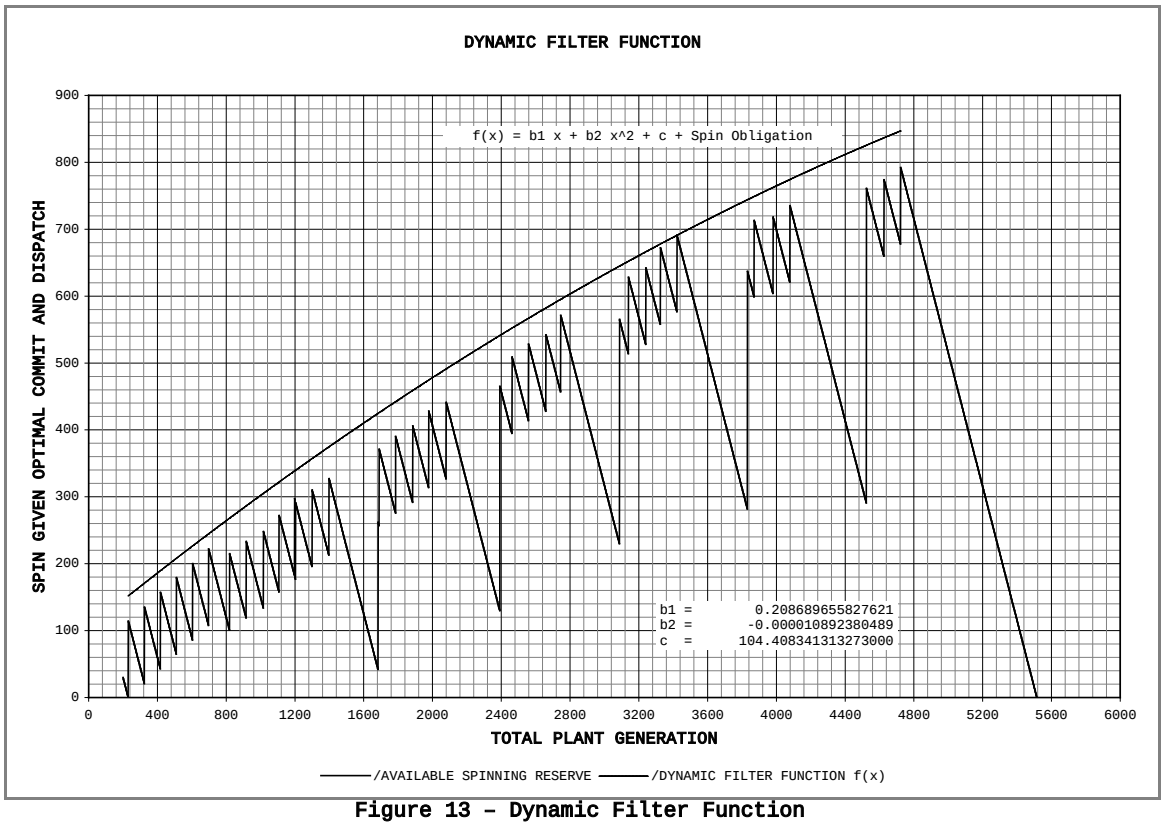


Figure 13 – Dynamic Filter Function

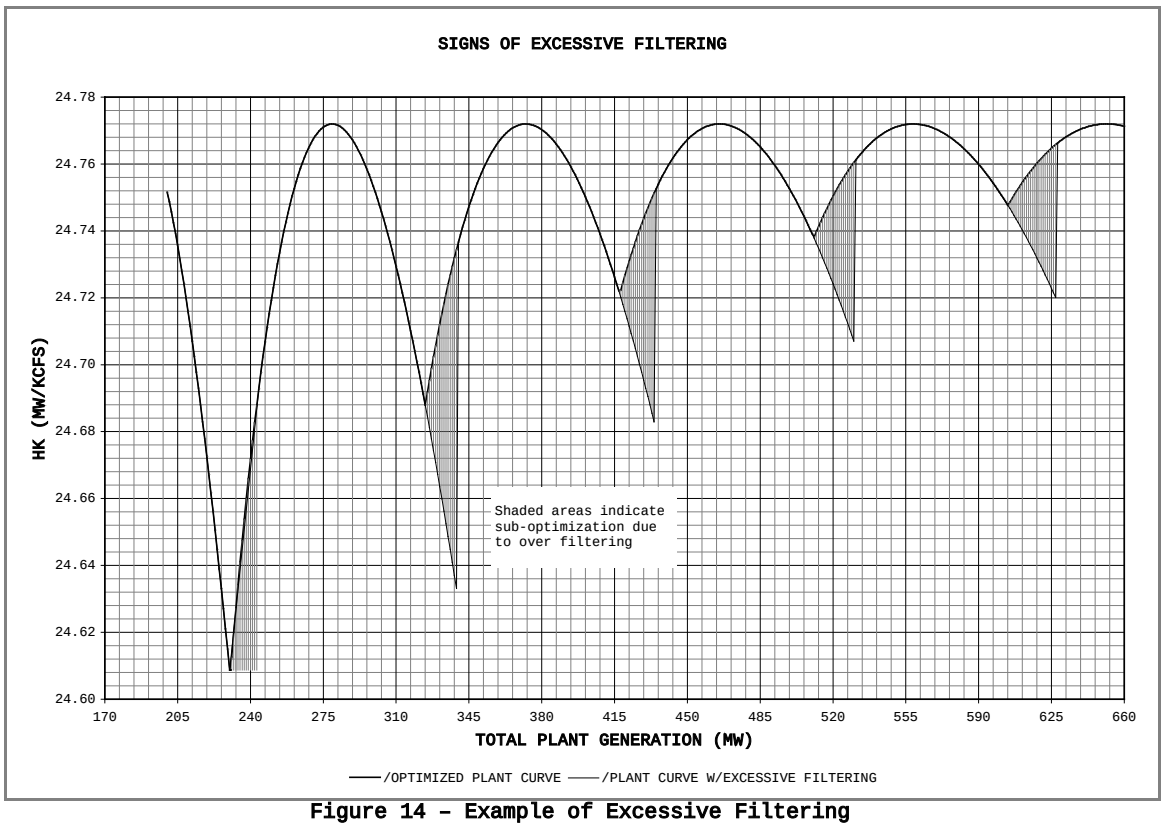


Figure 14 – Example of Excessive Filtering

Disclaimer This document and associated models are not final versions and are subject to change for development of the WP-12 Initial Proposal. Additionally, any data contained within the models or documentation are illustrative only and are not indicative of Initial Proposal development.

The dynamic filter may be created using the GARD calculated plant curve data, as generated per Section III. The following procedure assumes that the relevant plant curve data has been created.

1 – Using the plant curve data, create a plot of plant generation versus available spinning reserve. This plot should be based on the data set where the project is not being required to meet a reserve obligation.

The goal is to create a relationship between ideal unit commitment and dispatch and the spinning capability that is a byproduct of being optimized.

2 – Fit a function such that the curve generated just skims over the highest available spinning reserve while conforming to the general shape of the plant generation versus spinning reserve plot. See Figure 13.

3 – With GARD open, go to the VBA editor (Alt + F11 from Excel).

4 – Go to the "FunctionsMisc" module. Double-clicking on the module from the Project window will open up the "FunctionsMisc" module.

5 – Find Filter(). Update the function parameters and any boundaries per the curve fit in step 2.

6 – Go to the "Command" module. Double-clicking on the module from the Project window will open up the "Command" module.

7 – Find the variable "ComFilter." If it is desired that GARD implement the filter, ensure "ComFilter" is set as follows:

ComFilter = Filter(TargetGen, TargetSpin).

If the filter is not to be used set "ComFilter" equal to a large value. For example:

ComFilter = 2000.

8 – Using the currently generated plant curve data, create a plot of project generation versus project HK.

9 – With the filter in place, re-run the plant curve per Section III.

10 – Using the newly generated plant curve data, create a plot of project generation versus project HK. Overlay this plot with that generated in step 8.

Note If the filter is working properly, the two plots will be identical. If the filtering is excessive, the two plots will differ, with the filtered plot unable to achieve the same efficiency as the plot based on the unfiltered run. See Figure 14 for an example.

If the filtering is excessive, increase the value of the constant, c, input in step 5. Repeat step 9 and 10 until the plots are identical.

11 – With the filter working satisfactorily, go to the "SubDeploy" module. Double-clicking on the module from the Project window will open up the "SubDeploy" module.

12 – Find DeployDispatch().

13 – Find the variable "ComFilter." If it is desired that GARD implement the filter, ensure "ComFilter" is set as follows:

ComFilter = Filter(TargetGen, TargetSpin).

If the filter is not to be used set "ComFilter" equal to a large value. For example:

ComFilter = 2000.

Note The current Filter() function is set up to take one, general filter meant to be applicable across all unit availabilities. Given this general nature, care must be taken to not over filter under different unit availability scenarios. It is therefore recommended that the filter be created using data from the period with the highest unit availability.

Modifying the Filter() function to have a different filter by month is a simple matter of creating a filter function for each month, housing the function parameters in a two dimensional array, and passing on the month variable to the Filter() function. This implementation would structurally mimic the unit performance curve structure. See Section V for detail on the unit performance structure.

XIX. Processing Time Comments

Running GARD is CPU time intensive. Performing a full 12 month, 70 year stand ready and deployment run is expected to take approximately 70-150 minutes. This run time expectation is based on a machine with a 2.26 GHz processor and 1.00 GB of RAM. The run time is sensitive to the characteristics of the SCE signal. For SCE signal with high frequency and magnitude errors, the expected run time will increase. Likewise, assigning a higher response to a given project will result in an increased run time, all else being equal.

Note Running GARD on a machine with multiple processors will allow running multiple applications of GARD with little to no degradation in run time. When running multiple applications simultaneously, it is recommended that the read and write directories for the SCE signal and output files be unique to each application. Simultaneously accessing the same directory from different applications has proven problematic.

XX. General Comments Regarding Source Code

The VBA source code for the GARD model is constructed with a structure that should ease editing and trouble shooting. The GARD model is controlled by the Command() subroutine housed in the "CommandModule" module. The Command() routine sequences the calling of relevant subroutines for specific aspects of the model run. For example, specific routines

Disclaimer This document and associated models are not final versions and are subject to change for development of the WP-12 Initial Proposal. Additionally, any data contained within the models or documentation are illustrative only and are not indicative of Initial Proposal development.

exist for committing units, dispatching units, deploying reserves, etc. To aid in understanding and future editing, the VBA code is organized with the following format:

Subroutine Calls
Calls to subroutines are identified in the code using the fore slash character (/). The subroutine call is typically surrounded by fore slash characters to help quickly identify the routine. For example, in the following excerpt, the subroutine calls to "DispatchSub" and "BestCommitDispatch" is surrounded by fore slash characters:

```
                If AvailSpin < ComFilter _
                        And AvailSpin >= SpinMinFilter Then
                    'Iterate through all available unit families, sending a single row
                    'of unit commit data to the DispatchSub at a time.
                    For x = 1 To FamNumOut
                        Send(x) = CommitOutput(i - 1, x - 1)
                    Next x

                    'Given the unit commitment, optimize the dispatch.
                    'the DispatchSub routine dispatches units such that
                    'the marginal fuel cost of each unit is matched, thus
                    'minimizing total fuel consumption.
                    '////////////////////////////////////////////////
                    DispatchSub Send, DispatchOut               '///
                    '////////////////////////////////////////////

                    'Find the optimal unit commitment and dispatch for
                    'the given generation and reserve request.
                    '////////////////////////////////////////////////
                    BestCommitDispatch Send, DispatchOut, _
                        FirstPass, BestCommit, BestDispatch    '///
                    '////////////////////////////////////////////
                End If
```

Figure 15 – Quickly Identifying Subroutine Calls

Nested If Statements
Some section of code have several nested If statements containing For Next and nested For Next statements. To assisting editing and understanding, the components each If statement are assigned an alphabet character. For example, the first in a series of nested If statements will have and "a" in a comment following the "if. . .then," the "else," and "End If," statements. The following excerpt illustrates this structure:

```
        If x Mod 2 = 0 Then                                                    'a
            'Set up tables for tracking energy shift.
            MinGenTable(x, y) = GenTableIn(x, y) - PreDec(x, y)
            GYGenTable(x, y) = GRV(GenTableIn(x, y), GenTableIn(x - 1, y))
            If MinGenTable(x, y) > GYGenTable(x, y) _
                Then GYGenTable(x, y) = MinGenTable(x, y)                      'b
            SLLHGenTable(x, y) = _
                (GenTableIn(x, y) * 8 - GYGenTable(x, y) * 4) / 4
            GYAvailDec(x, y) = GYGenTable(x, y) - MinGenTable(x, y)
            SLLHAvailDec(x, y) = SLLHGenTable(x, y) - MinGenTable(x, y)

            'Calculate any energy shift into LLH due to dec.
            If TotDec(y) > GYAvailDec(x, y) Then                               'c
                GYSourced = TotDec(y) - GYAvailDec(x, y)
            Else                                                               'c
                GYSourced = 0
            End If                                                             'c
            'Calculate any energy shift that may be needed to
            'provide sufficient SLLH dec capability.
            If TotDec(y) > SLLHAvailDec(x, y) Then                             'd
                SLLHSourced = TotDec(y) - SLLHAvailDec(x, y)
            Else                                                               'd
                SLLHSourced = 0
            End If                                                             'd
```

Figure 16 – Nested If Statement Identification

XXI. Some Variable Definitions

Many of the variable used in the GARD VBA code are fairly self evident or contain explanatory comments in the VBA code. However, some variables, in terms of their definition and use and/or their structure, warrant some additional discussion. The intent of this section is to clarify some of the more cryptic variables to ease understanding and future editing.

ResultsBig(140, 20, 12, 4)
ResultsBig() is a public array variable containing all of the data written to the OUTPUT tab. All of the data contained in the ResultsBig array are used to calculate the stand ready impacts. Additionally, the output contained in ResultsBig(…, …,…,

Disclaimer This document and associated models are not final versions and are subject to change for development of the WP-12 Initial Proposal. Additionally, any data contained within the models or documentation are illustrative only and are not indicative of Initial Proposal development.

4) is the starting point from which the deployment impacts are calculated. That is, with the FCRPS now fully standing ready with reserve capability, the system is now ready to respond to reserve needs. The array elements are defined as:

- Element 1: Indicates the water year and diurnal period for up to 70 water years. For example, ResultsBig(1, …, …, …) contains results for the HLH period of the first water year. ResultsBig(2, …, …, …) contains output for the LLH period of the first water year, etc.

- Element 2: Indicates the type of output being reported. Registers 1 – 7 contain unit commitment output for up to seven unit families. Register 8 contains the total plant generation. Registers 9-15 contain the dispatch by unit family. Register 16 contains the targeted marginal cost. Register 17 contains the total turbine flow. Register 18 contains the available spinning reserve. Register 19 contains the available non-spinning reserve. Register 20 contains spill for tail water management.

- Element 3: Indicates the month. Month 1 being associated with October and Month 12 being associated with September.

- Element 4: Indicates the scenario associated with the output data. Scenario 1 is the base case. Scenario 2 includes dec reserves. Scenario 3 includes dec and non-spinning reserves. Scenario 4 includes dec, non-spinning and spinning reserves.

DeployBig(44640, 21)
DeployBig() is a public array similar to the ResultsBig() array, except the DeployBig() array contains deployment impacts for each minute of one specific month of one specific water year. The results are not kept due to memory constraints. Rather, for each month and water year, the DeployBig() array is re-initialized. If the data contained in DeployBig() is in need of viewing, activate the WRITE DEPLOY DETAIL button when calculating deployment costs (see Section II).

- Element 1: Indicates the minute of the month and water year for which results are being reported.

- Element 2: Indicates the type of output being reported. Element 2 in the DeployBig() array is structured identically to element 2 in the ResultsBig() array (see Element 2 of the ResultsBig() array for more detail).

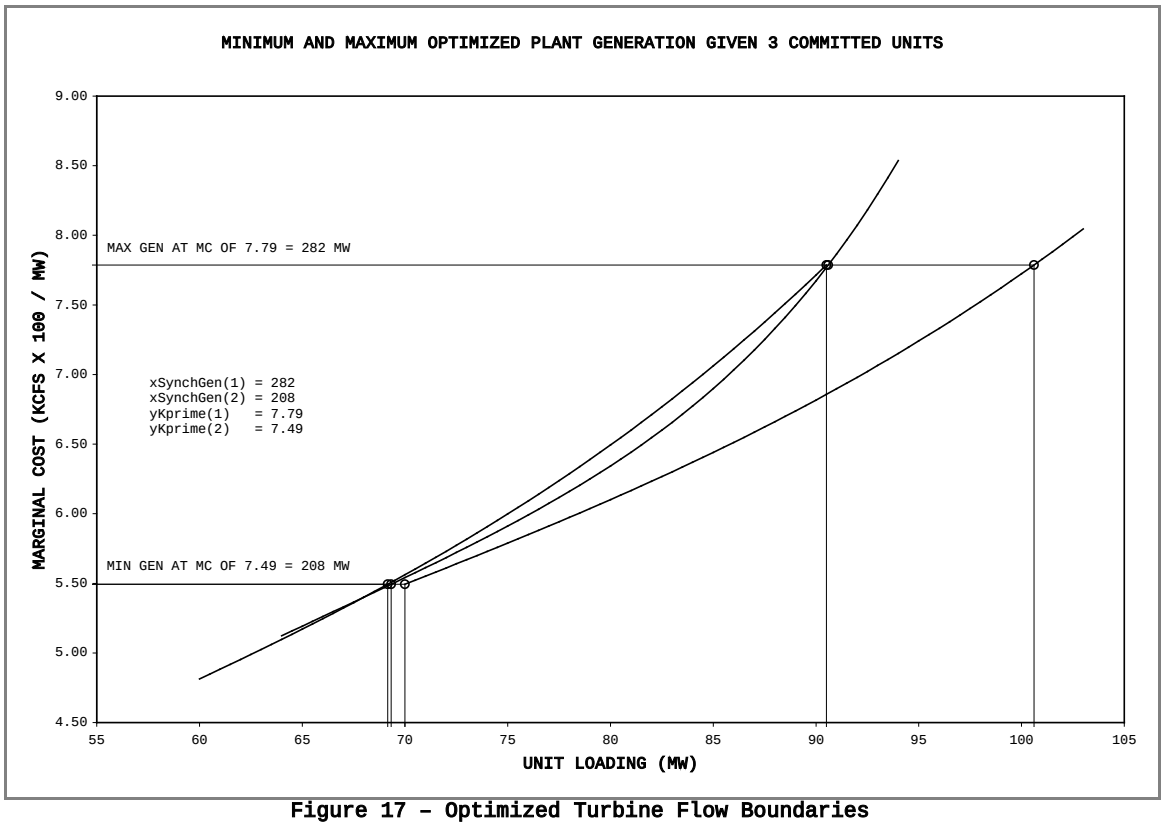


Figure 17 – Optimized Turbine Flow Boundaries

xSynchGen(2)
xSynchGen() is an array in the DispatchSub() containing the maximum and minimum generation achievable where the marginal cost of each committed unit is identical. This defines the boundary of optimal unit dispatch.

- Element 1: The maximum, marginal cost synchronized plant generation.

Disclaimer This document and associated models are not final versions and are subject to change for development of the WP-12 Initial Proposal. Additionally, any data contained within the models or documentation are illustrative only and are not indicative of Initial Proposal development.

- Element 2: The minimum, marginal cost synchronized plant generation.

yKprime(2)
yKprime() is an array in the DispatchSub() containing the marginal cost associated with the maximum and minimum synchronized generation. This defines the boundary of optimal unit dispatch.

- Element 1: The marginal cost associated with the maximum, marginal cost synchronized plant generation.
- Element 2: The marginal cost associated with the minimum, marginal cost synchronized plant generation.

Note The definition and purpose of arrays xSynchGen() and yKprime() may be better understood by observing Figure 17. Figure 17 show the marginal cost curves for three committed units. Total, plant turbine flow per MW generated is minimized when the marginal cost of each committed unit is set equal. Given the different characteristics that may exist between units, the minimum and maximum generation achievable where the marginal costs are equal may differ from the sum of the individual unit minimum and maximum loadings. xSynchgen() and yKprime() contain the upper and lower boundaries where total optimal turbine flow per MW is achievable.

Send, HSend, LSend
The array variable Send is used in the Command() and DeployDispatch() sub routines while the array variables HSend and LSend appear in the Signals() subroutine. Each of these variables contain a specific unit commitment. HSend and LSend are merely used to hold a unit commitment contained in Send depending on whether the period in question is HLH (HSend) or LLH (LSend).

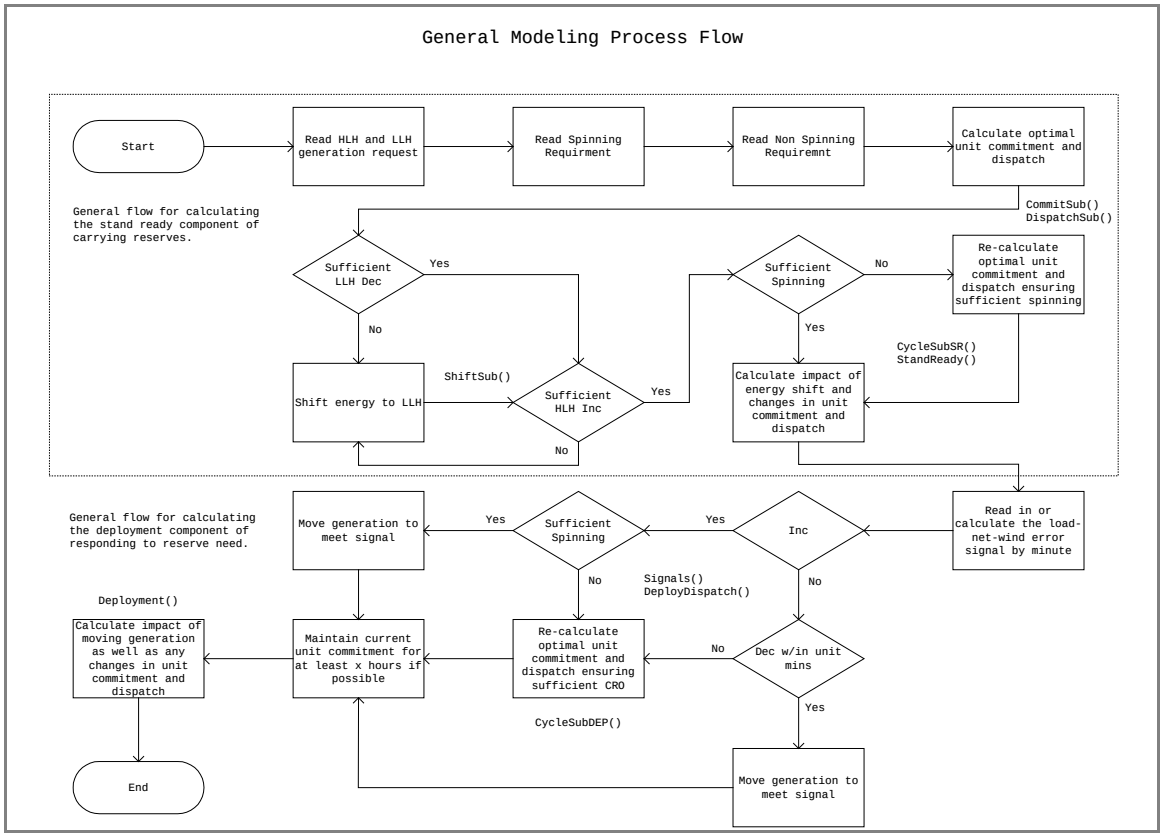
MAXGen1, HMAXGen1, LMAXGen1
MAXGen1 is defined as the maximum generation for the current dispatch. H and LMAXGen1 contain the maximum generation for the current unit dispatch by diurnal period. These variables appear in the Signals() and DeployDispatch() subroutines.

MINGen1, HMINGen1, LMINGen1
MINGen1 is defined as the minimum generation for the current dispatch. H and LMINGen1 contain the minimum generation for the current unit dispatch by diurnal period.

Disclaimer This document and associated models are not final versions and are subject to change for development of the WP-12 Initial Proposal. Additionally, any data contained within the models or documentation are illustrative only and are not indicative of Initial Proposal development.

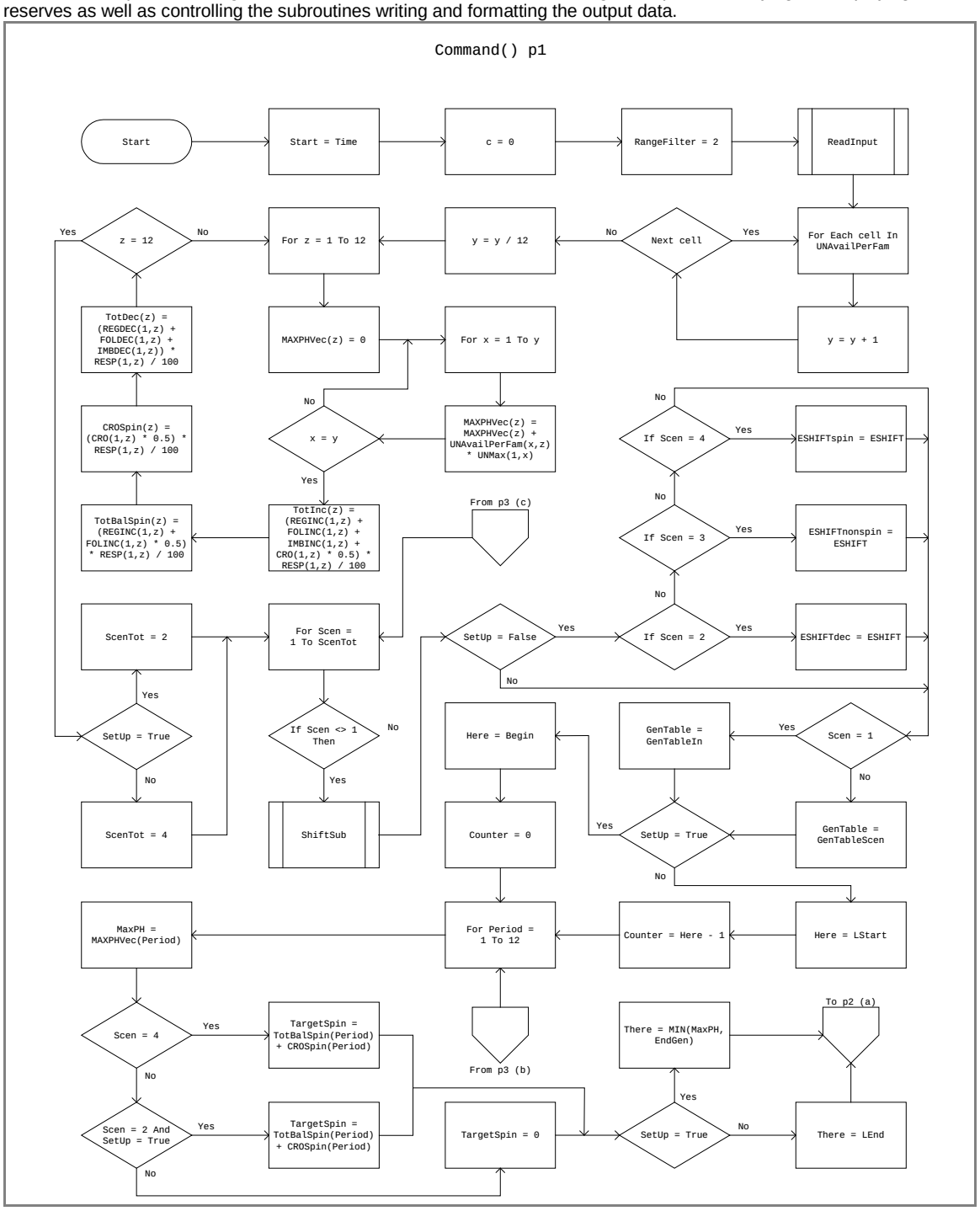
General Modeling Process Flow

The General Modeling Process Flow diagram is a general overview of the GARD model showing the general sequencing of calculations and subroutine calls.

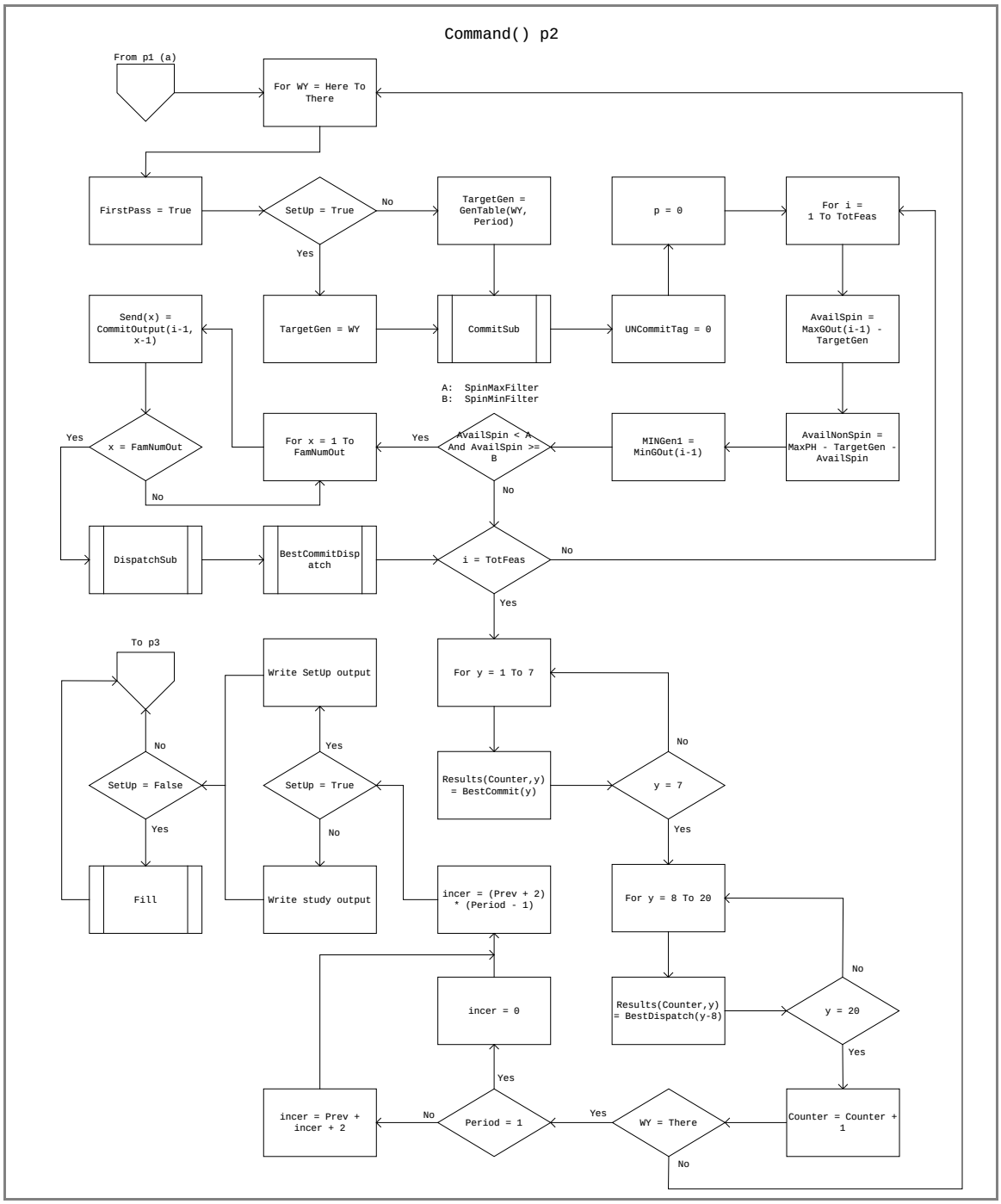


Command() Process Flow

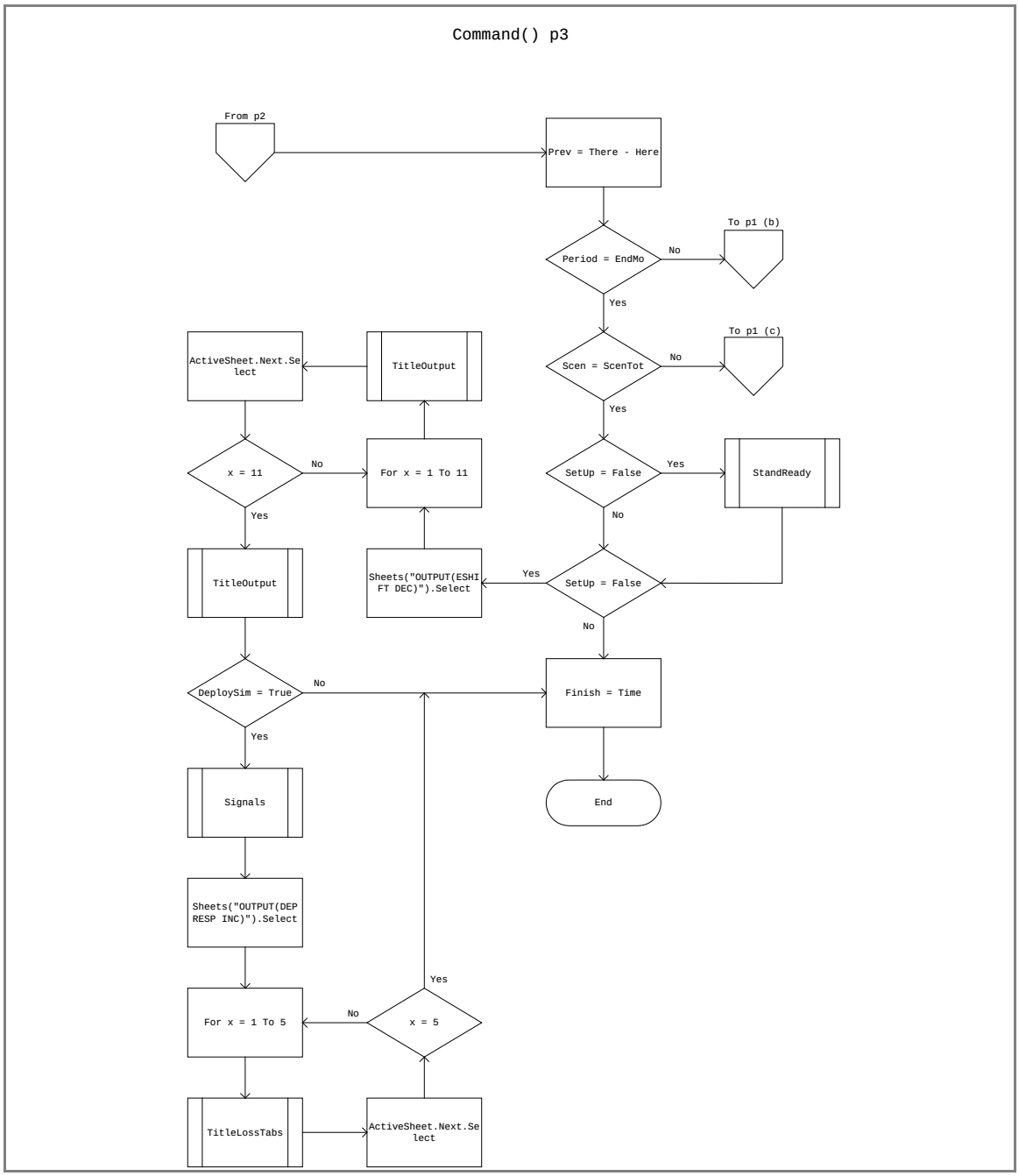
The Command() subroutine is the primary controlling code. In a general sense, the Command() subroutine merely controls the sequential calling of all the other subroutines used in calculating the impacts of carrying and deploying reserves as well as controlling the subroutines writing and formatting the output data.



Disclaimer This document and associated models are not final versions and are subject to change for development of the WP-12 Initial Proposal. Additionally, any data contained within the models or documentation are illustrative only and are not indicative of Initial Proposal development.



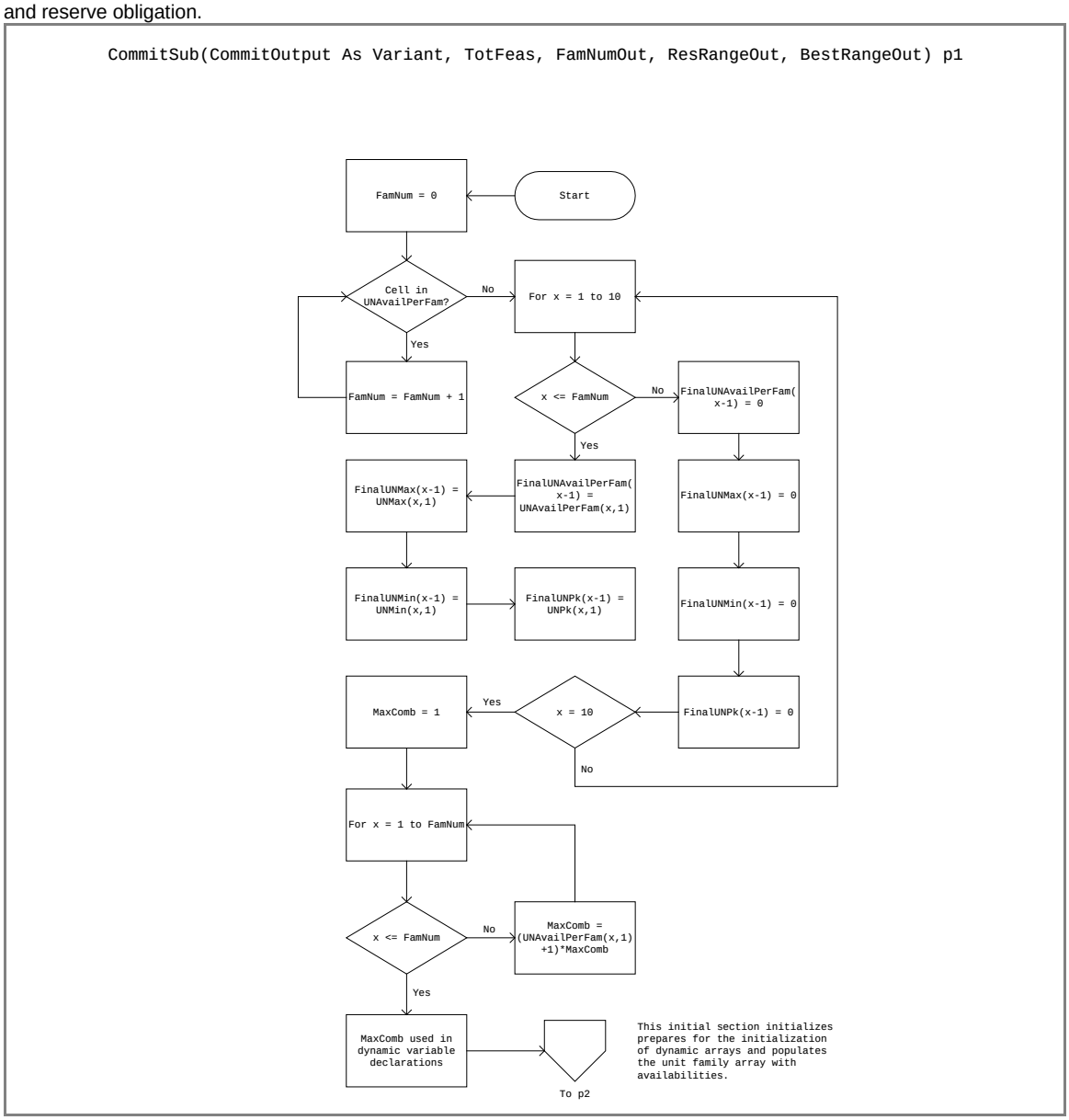
Disclaimer This document and associated models are not final versions and are subject to change for development of the WP-12 Initial Proposal. Additionally, any data contained within the models or documentation are illustrative only and are not indicative of Initial Proposal development.



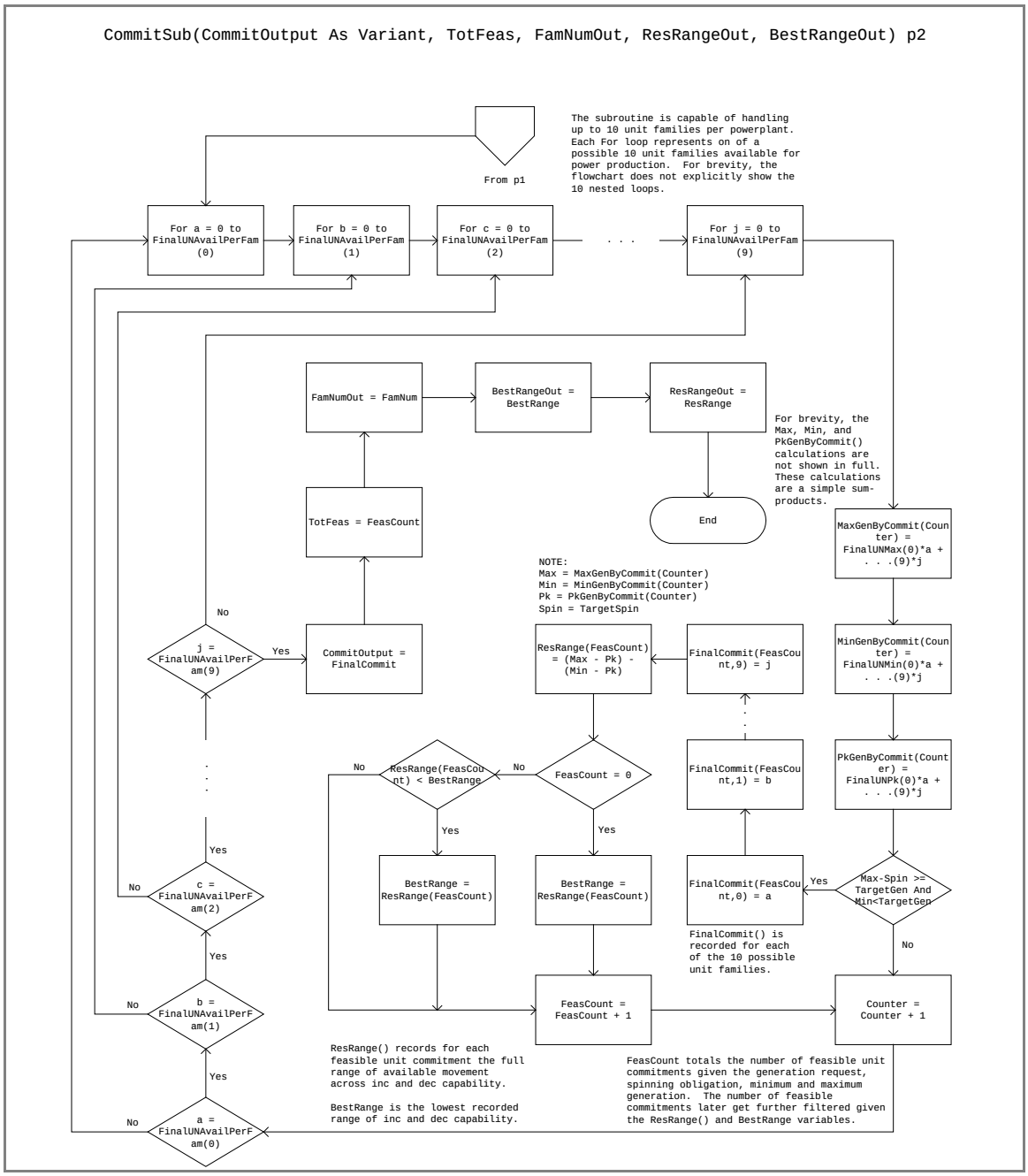
Disclaimer This document and associated models are not final versions and are subject to change for development of the WP-12 Initial Proposal. Additionally, any data contained within the models or documentation are illustrative only and are not indicative of Initial Proposal development.

CommitSub() Process Flow

The CommitSub() subroutine calculates the feasible unit commitment combinations meeting both the generation request and reserve obligation.



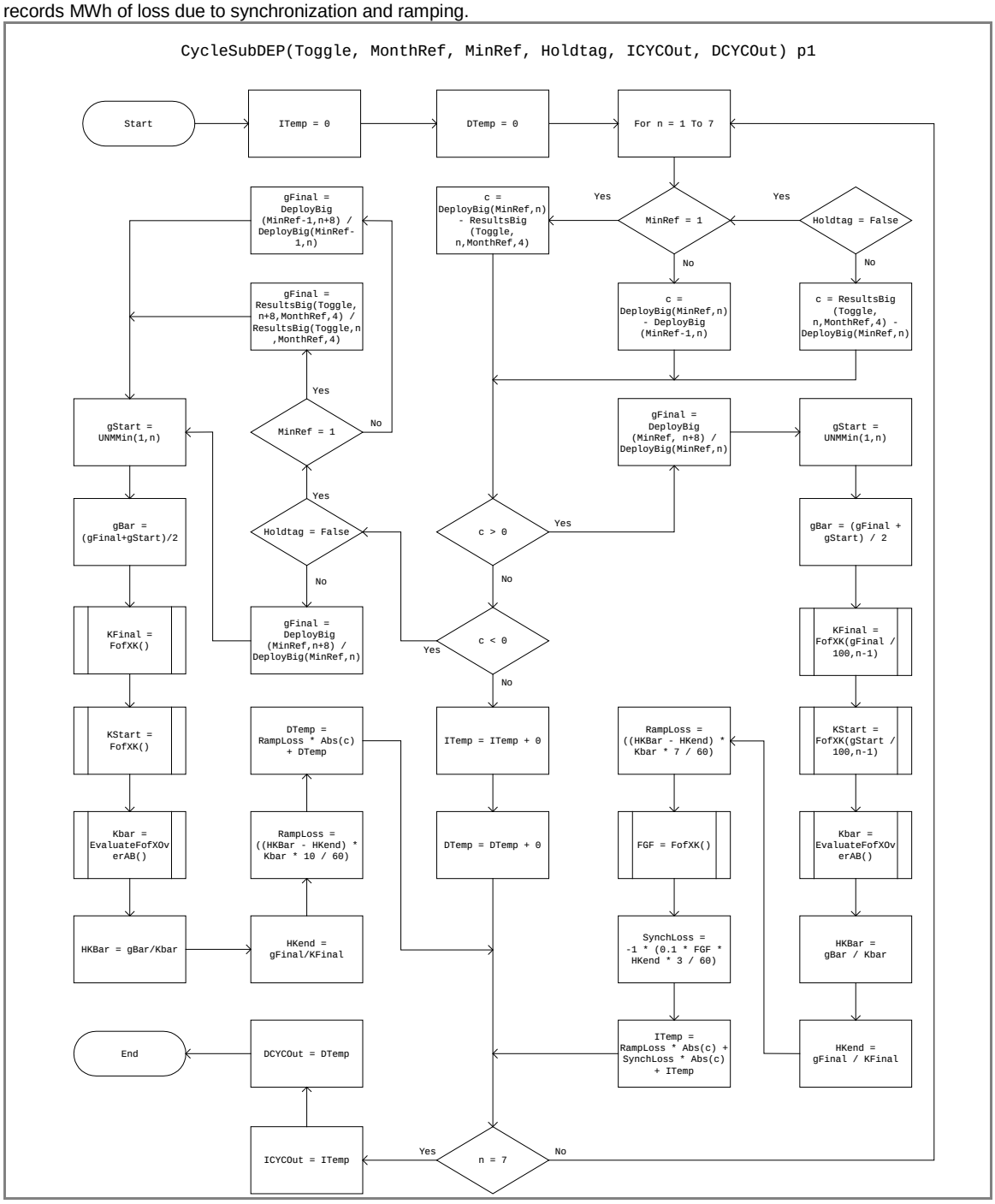
Disclaimer This document and associated models are not final versions and are subject to change for development of the WP-12 Initial Proposal. Additionally, any data contained within the models or documentation are illustrative only and are not indicative of Initial Proposal development.



Disclaimer This document and associated models are not final versions and are subject to change for development of the WP-12 Initial Proposal. Additionally, any data contained within the models or documentation are illustrative only and are not indicative of Initial Proposal development.

CycleSubDEP() Process Flow

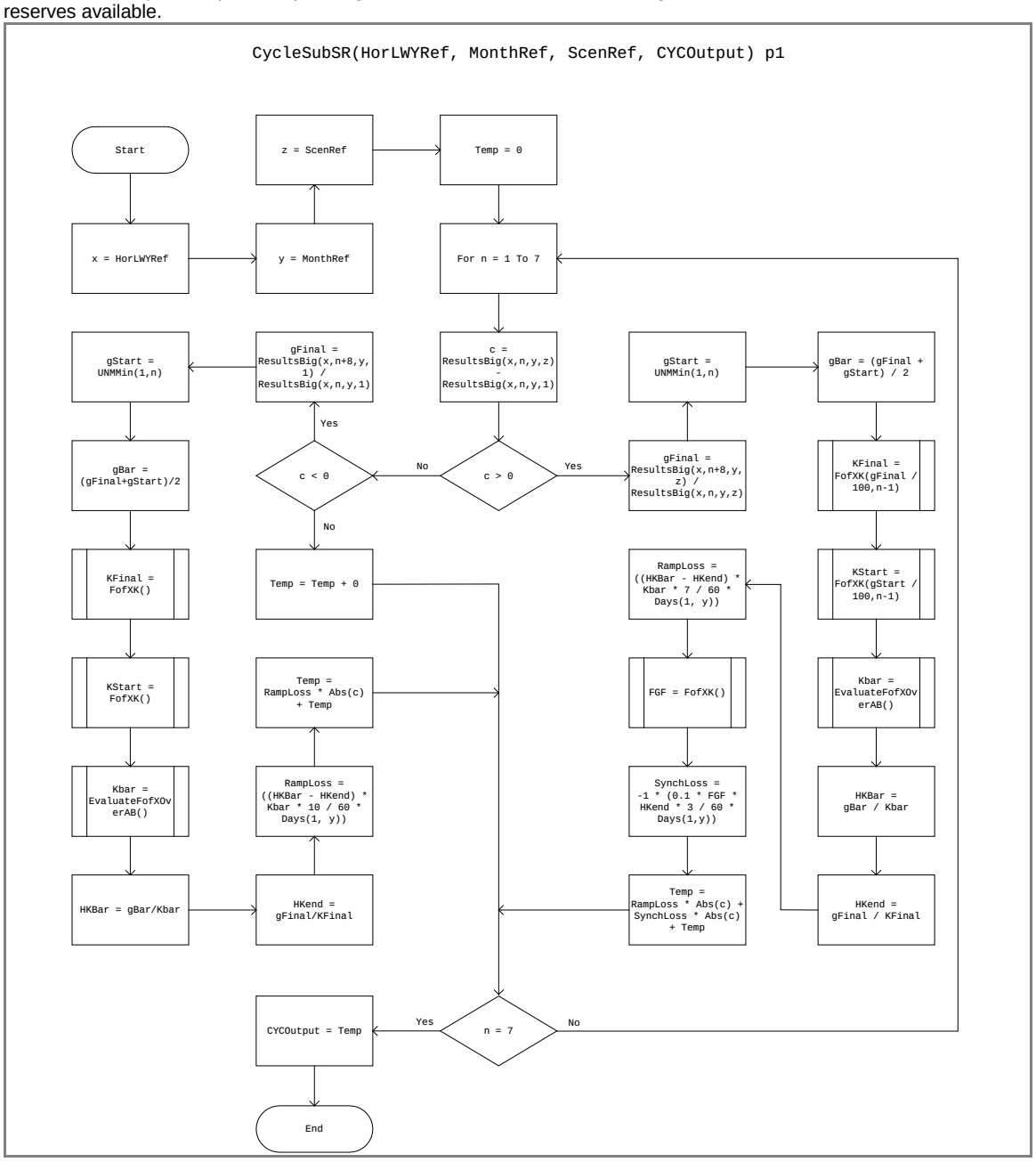
The CycleSubDEP() subroutine calculates the impact of unit cycling events during the course of reserve deployment and records MWh of loss due to synchronization and ramping.



Disclaimer This document and associated models are not final versions and are subject to change for development of the WP-12 Initial Proposal. Additionally, any data contained within the models or documentation are illustrative only and are not indicative of Initial Proposal development.

CycleSubSR() Process Flow

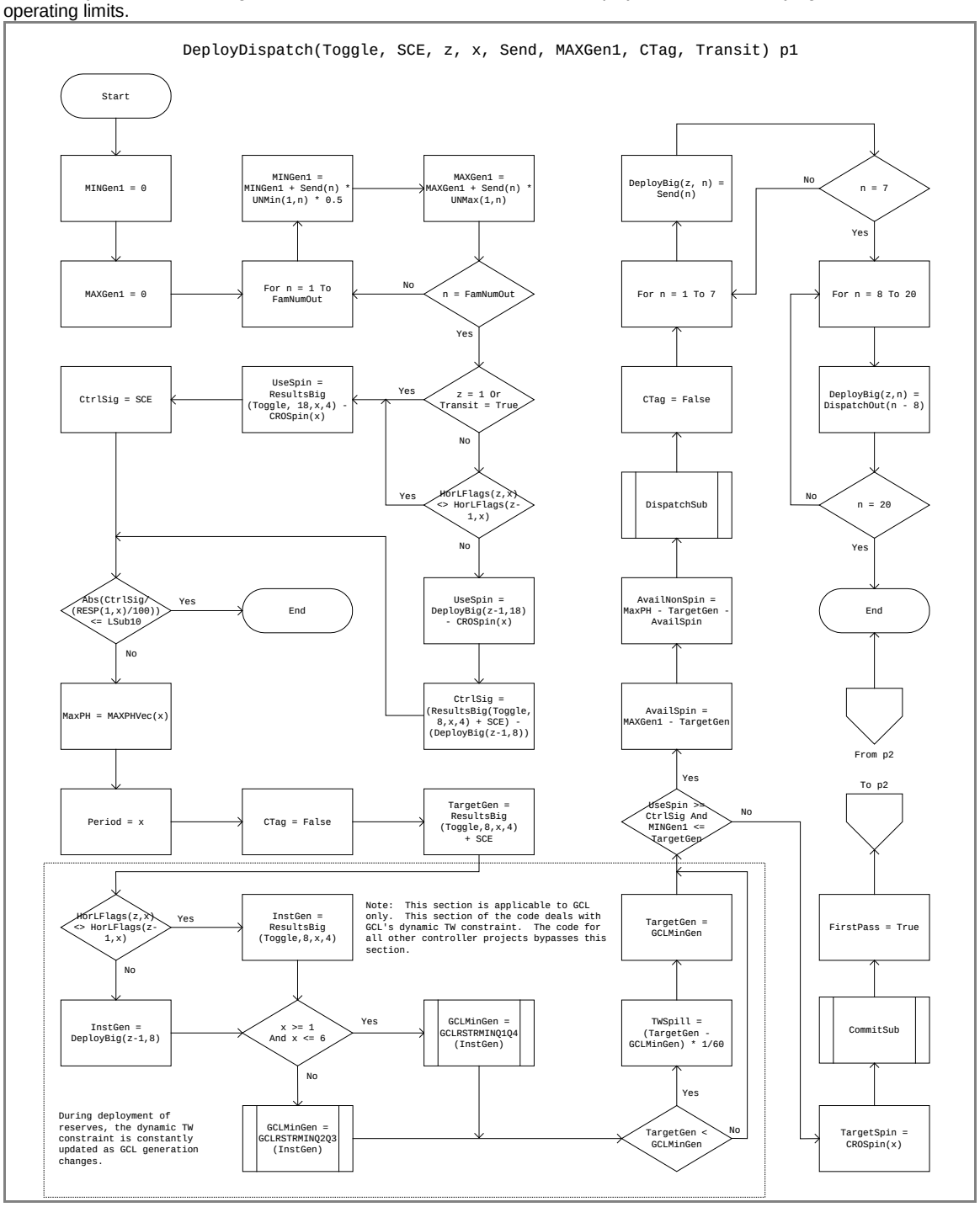
The CycleSubSR() subroutine calculates the impact of unit cycling required to be standing ready with sufficient reserves. These are the impacts of potentially having some amount of incremental change in unit commitment in order to have reserves available.



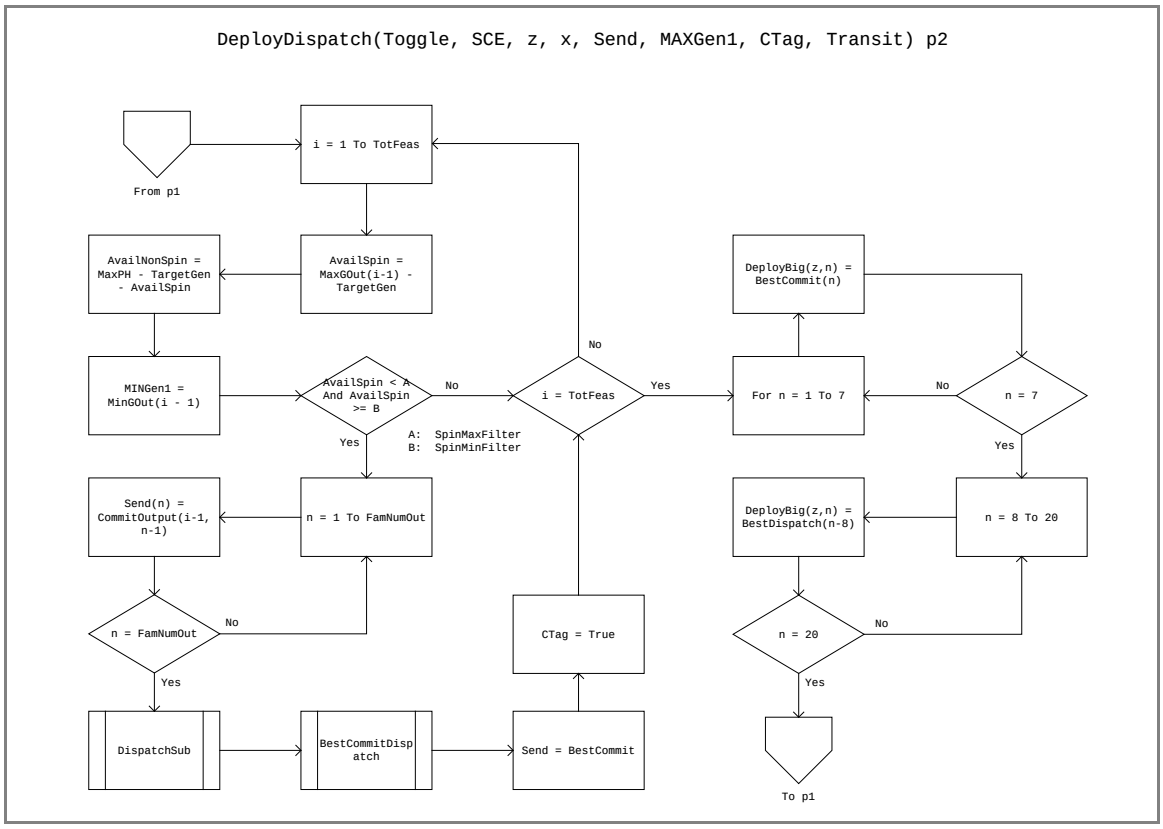
Disclaimer This document and associated models are not final versions and are subject to change for development of the WP-12 Initial Proposal. Additionally, any data contained within the models or documentation are illustrative only and are not indicative of Initial Proposal development.

DeployDispatch() Process Flow

The DeployDispatch() subroutine controls the projects unit dispatch during the course of reserve deployment. Additionally, this routine changes unit commitment as needed in order to deploy reserves while staying within unit operating limits.



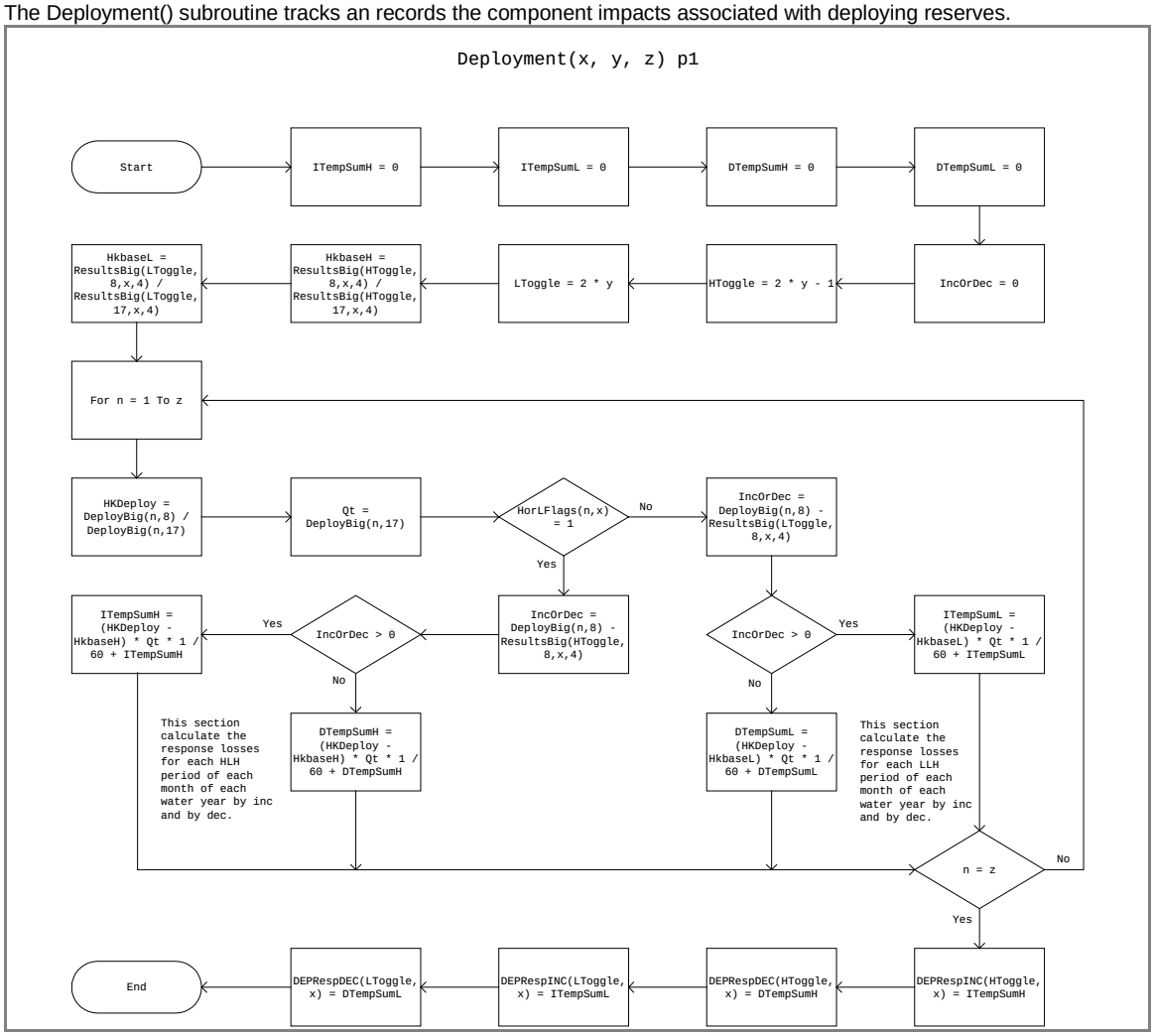
Disclaimer This document and associated models are not final versions and are subject to change for development of the WP-12 Initial Proposal. Additionally, any data contained within the models or documentation are illustrative only and are not indicative of Initial Proposal development.



Disclaimer This document and associated models are not final versions and are subject to change for development of the WP-12 Initial Proposal. Additionally, any data contained within the models or documentation are illustrative only and are not indicative of Initial Proposal development.

Deployment() Process Flow

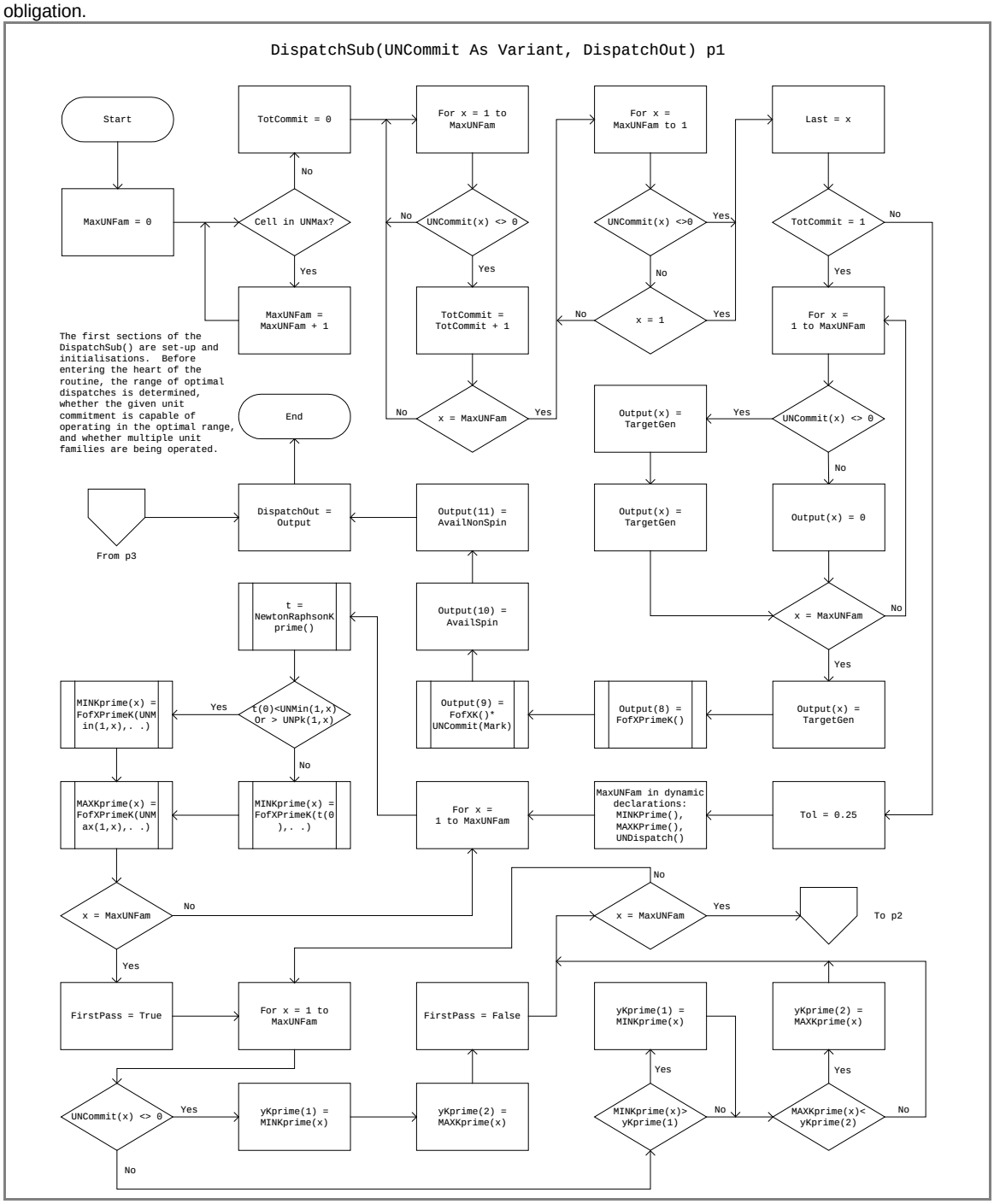
The Deployment() subroutine tracks an records the component impacts associated with deploying reserves.



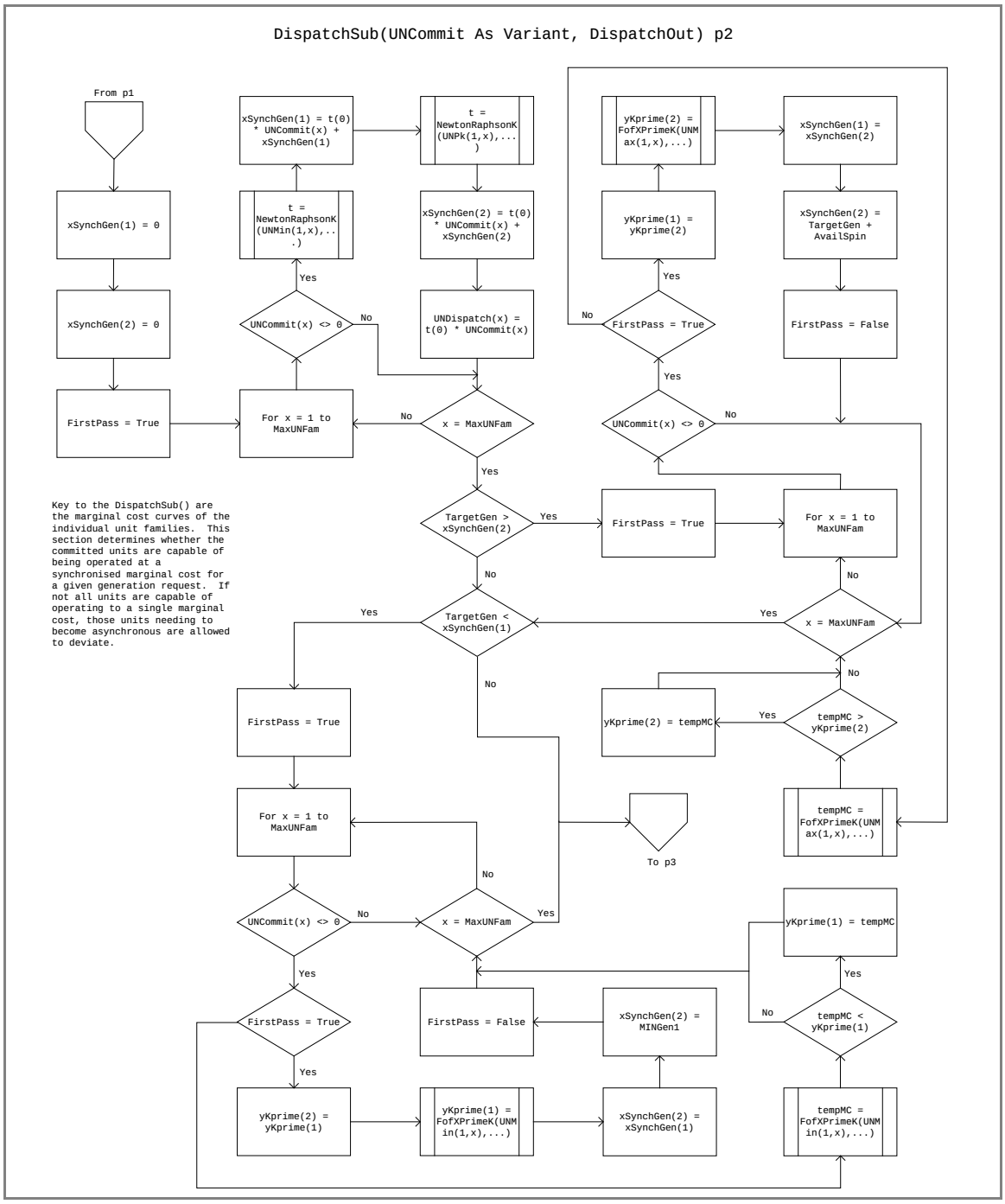
Disclaimer This document and associated models are not final versions and are subject to change for development of the WP-12 Initial Proposal. Additionally, any data contained within the models or documentation are illustrative only and are not indicative of Initial Proposal development.

DispatchSub() Process Flow

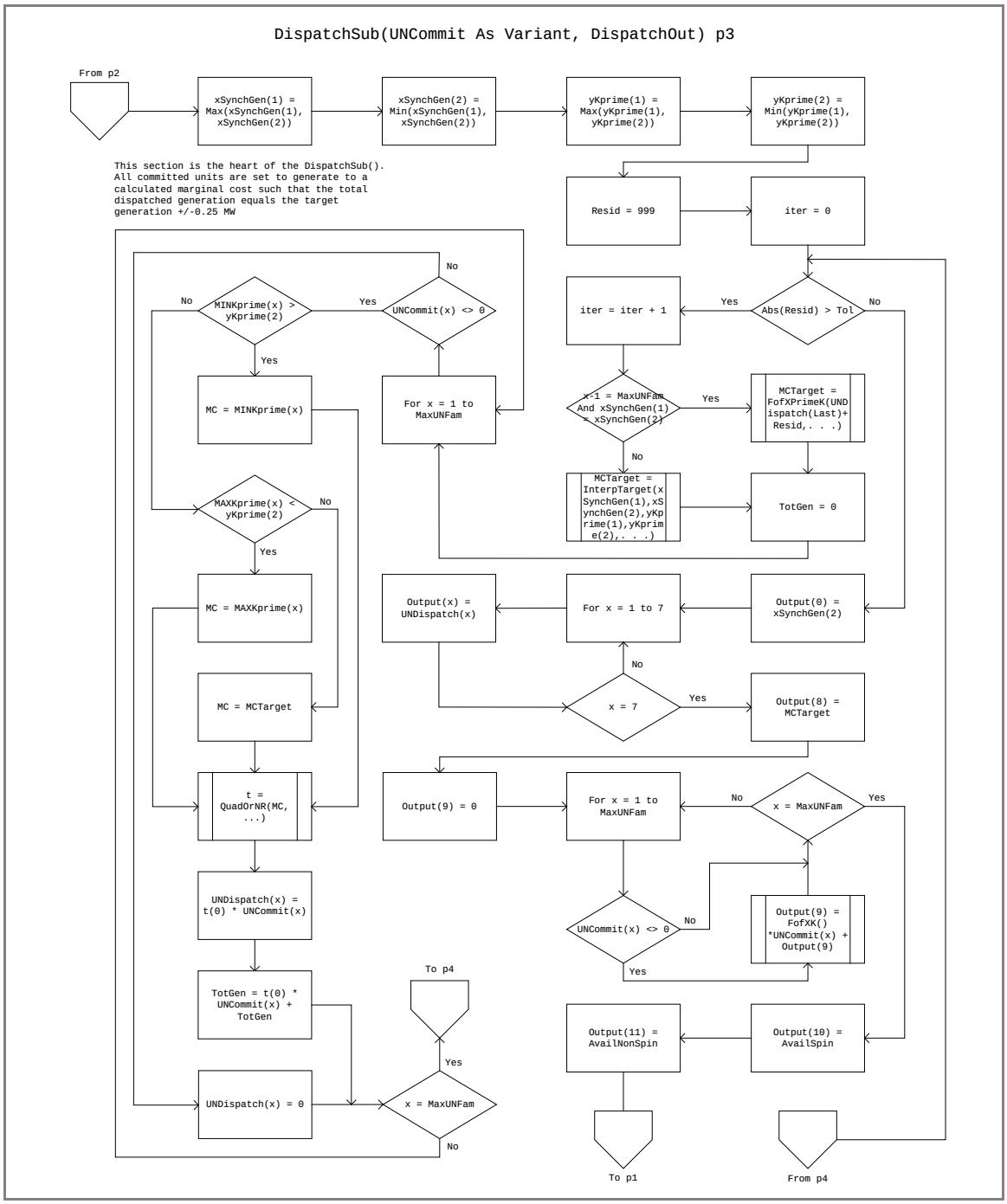
The DispatchSub() subroutine optimizes the unit loading given a unit commitment, generation request and reserve obligation.



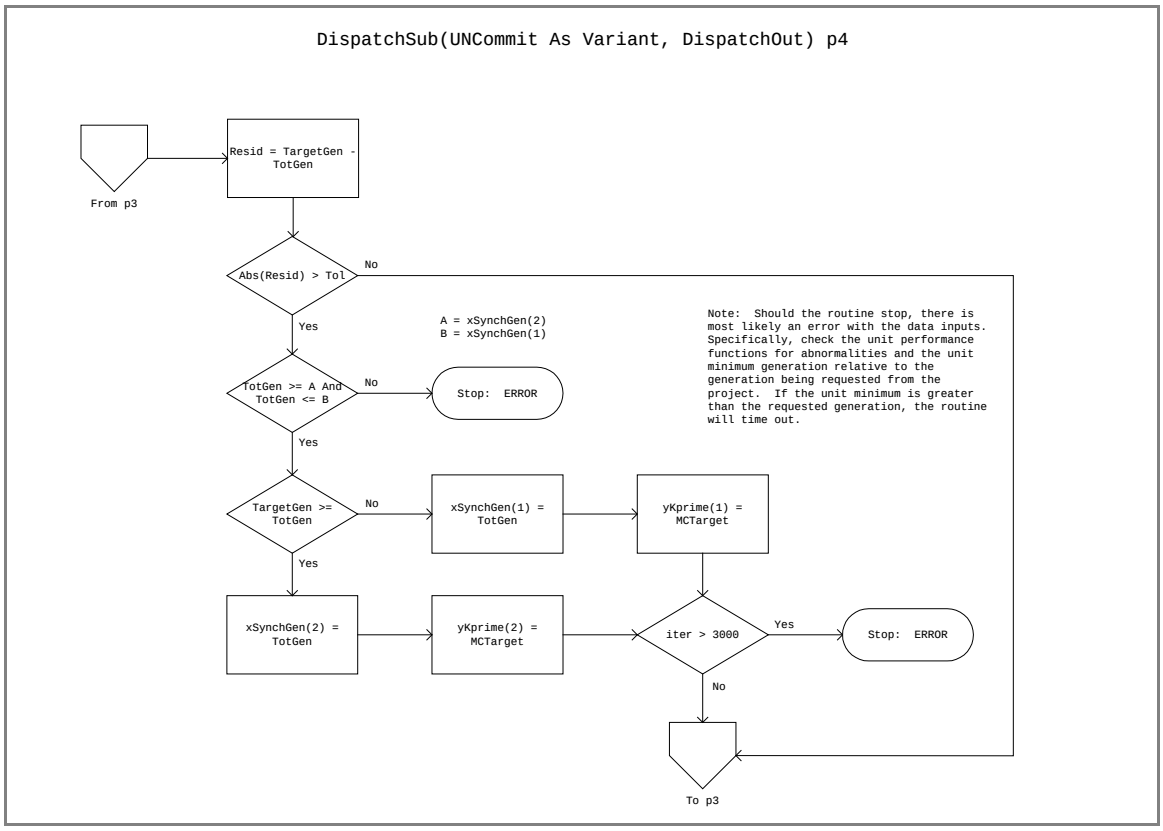
Disclaimer This document and associated models are not final versions and are subject to change for development of the WP-12 Initial Proposal. Additionally, any data contained within the models or documentation are illustrative only and are not indicative of Initial Proposal development.



Disclaimer This document and associated models are not final versions and are subject to change for development of the WP-12 Initial Proposal. Additionally, any data contained within the models or documentation are illustrative only and are not indicative of Initial Proposal development.



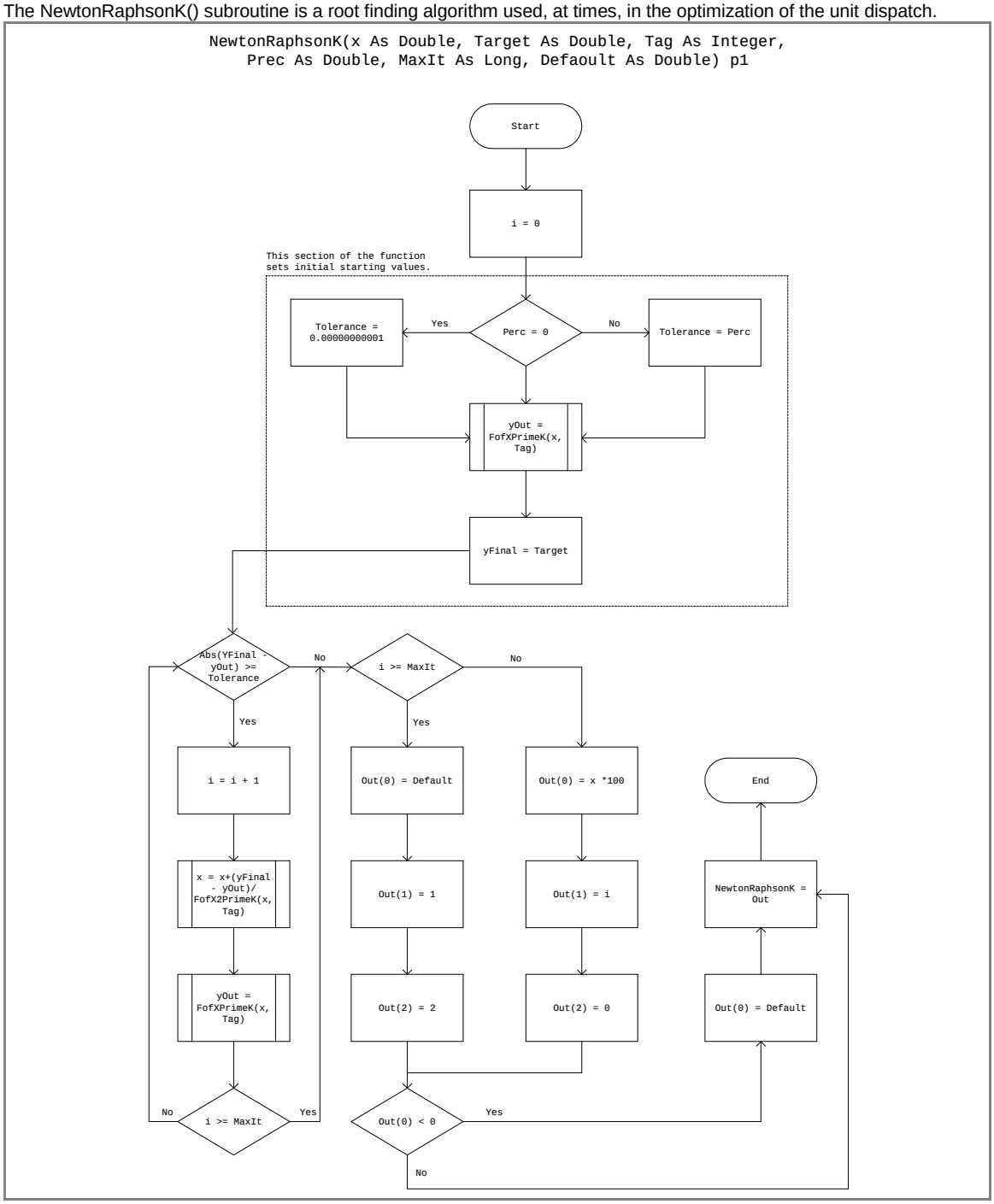
Disclaimer This document and associated models are not final versions and are subject to change for development of the WP-12 Initial Proposal. Additionally, any data contained within the models or documentation are illustrative only and are not indicative of Initial Proposal development.



Disclaimer This document and associated models are not final versions and are subject to change for development of the WP-12 Initial Proposal. Additionally, any data contained within the models or documentation are illustrative only and are not indicative of Initial Proposal development.

NewtonRaphsonK() Process Flow

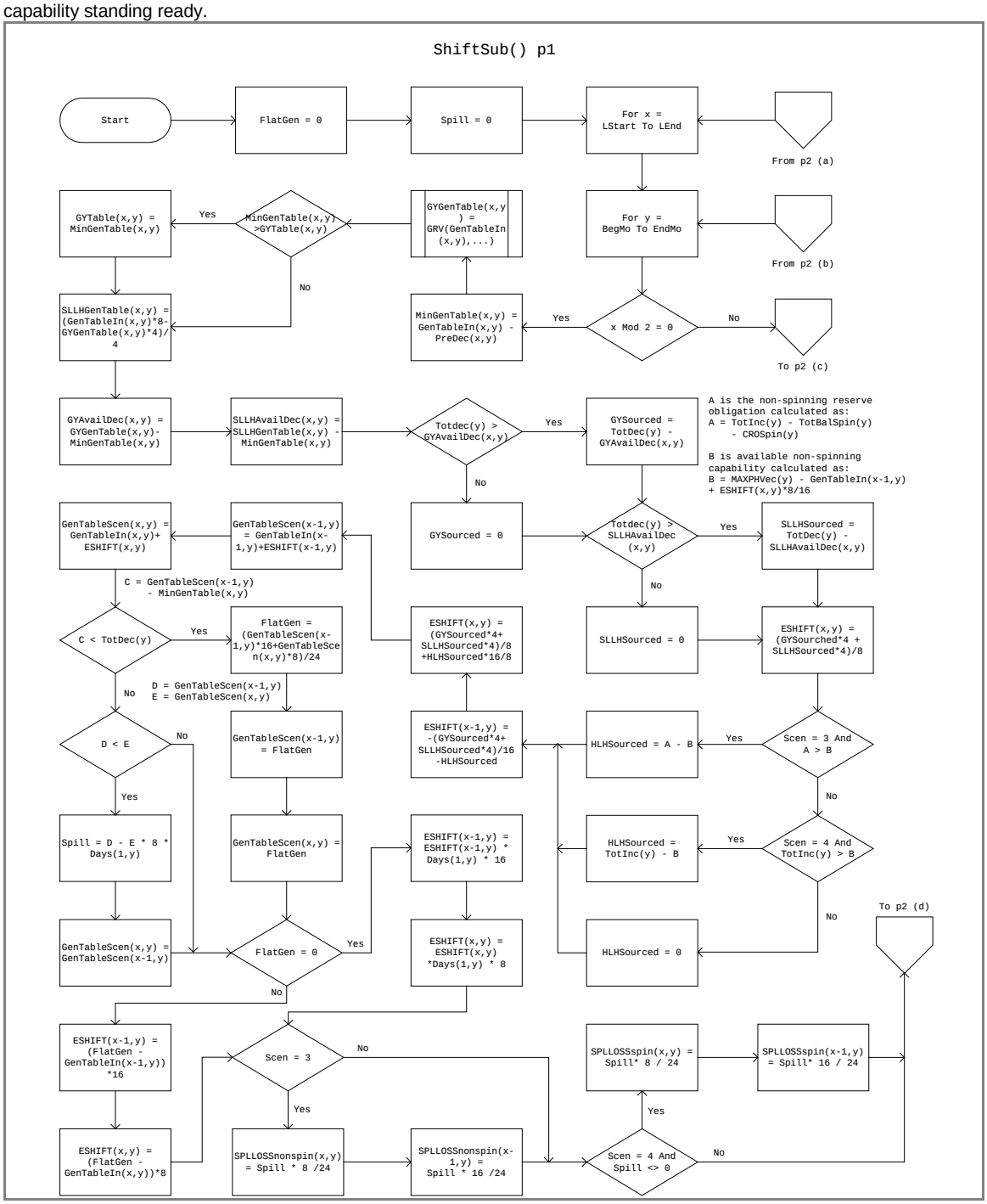
The NewtonRaphsonK() subroutine is a root finding algorithm used, at times, in the optimization of the unit dispatch.



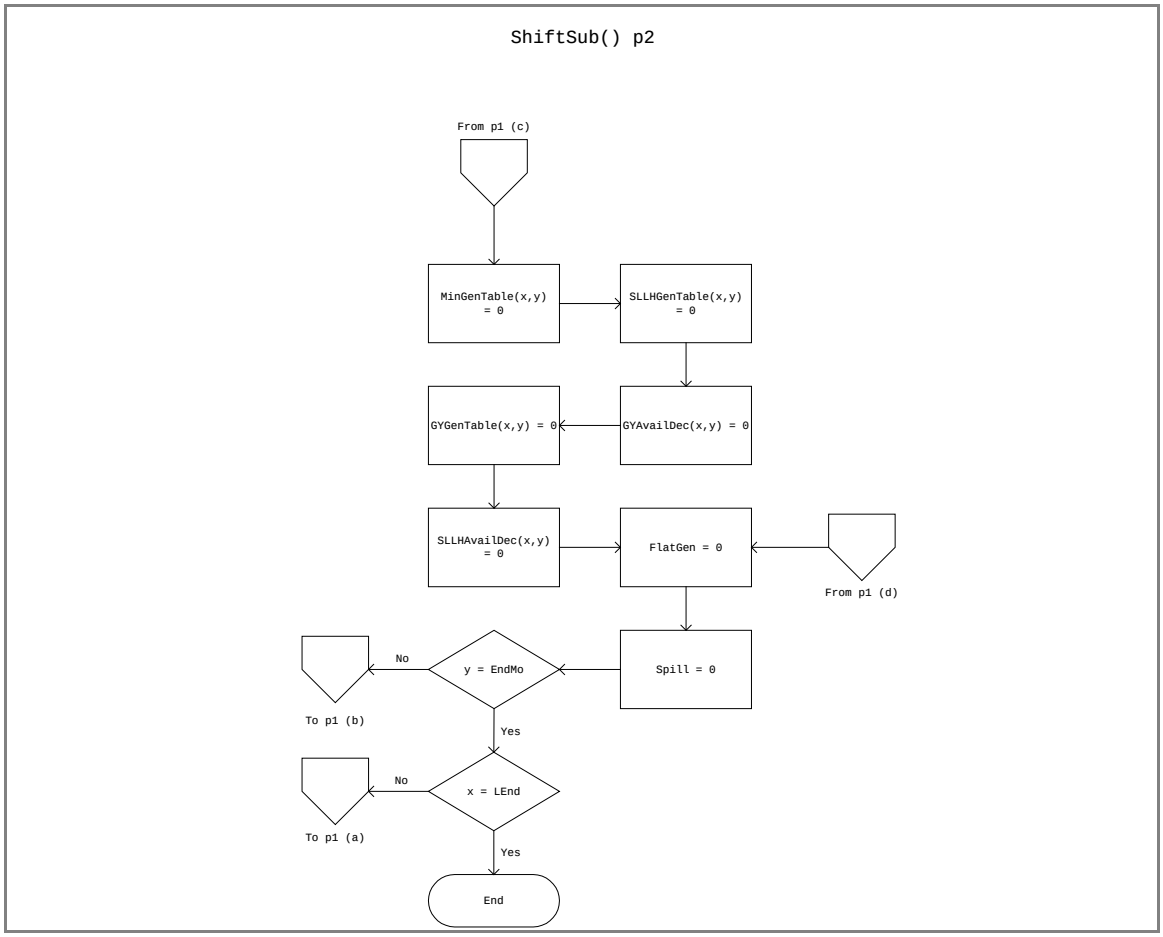
Disclaimer This document and associated models are not final versions and are subject to change for development of the WP-12 Initial Proposal. Additionally, any data contained within the models or documentation are illustrative only and are not indicative of Initial Proposal development.

ShiftSub() Process Flow

The ShiftSub() subroutine calculates the amount of energy shaped into the LLH period in order to have sufficient reserve capability standing ready.



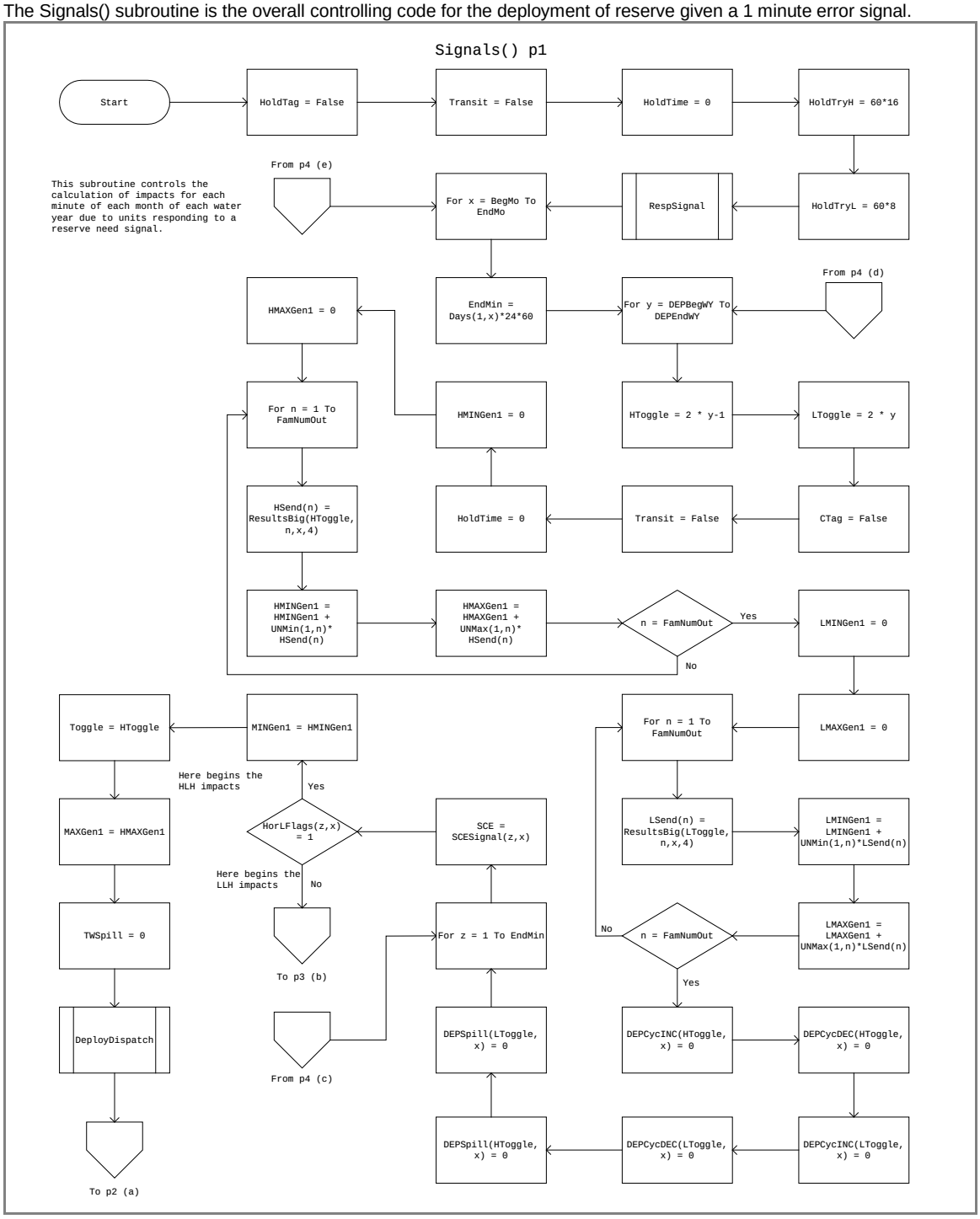
Disclaimer This document and associated models are not final versions and are subject to change for development of the WP-12 Initial Proposal. Additionally, any data contained within the models or documentation are illustrative only and are not indicative of Initial Proposal development.



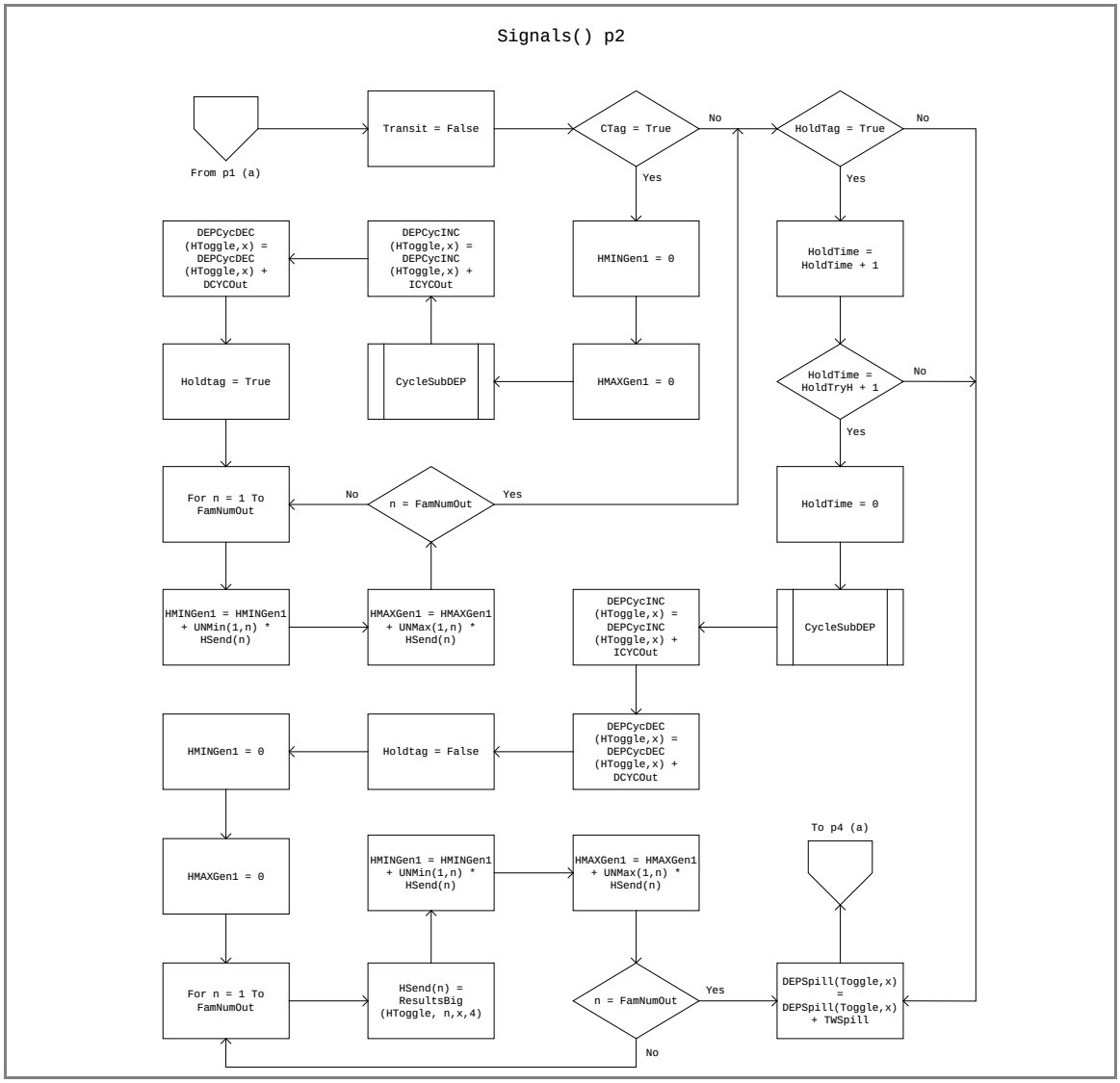
Disclaimer This document and associated models are not final versions and are subject to change for development of the WP-12 Initial Proposal. Additionally, any data contained within the models or documentation are illustrative only and are not indicative of Initial Proposal development.

Signals() Process Flow

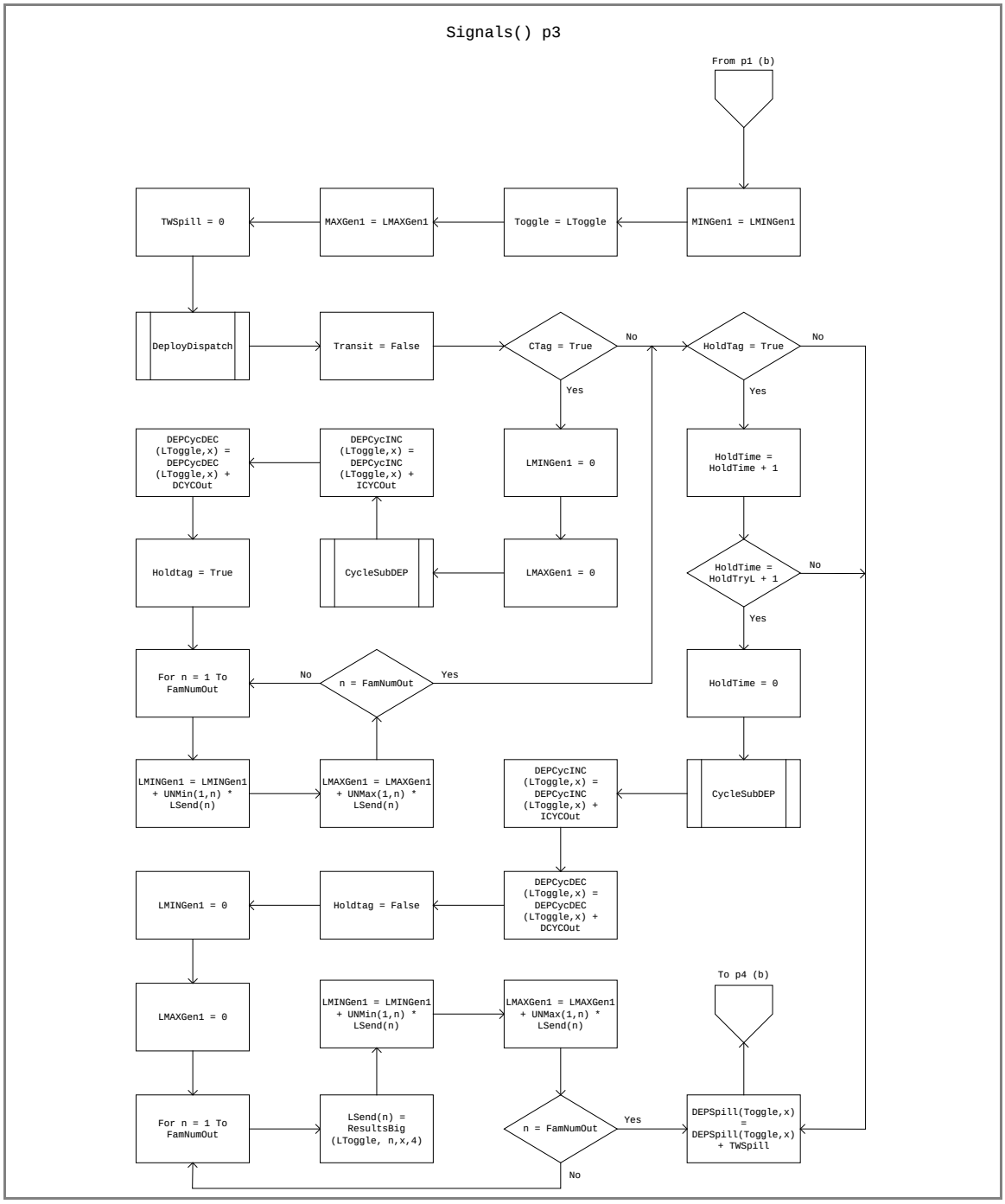
The Signals() subroutine is the overall controlling code for the deployment of reserve given a 1 minute error signal.



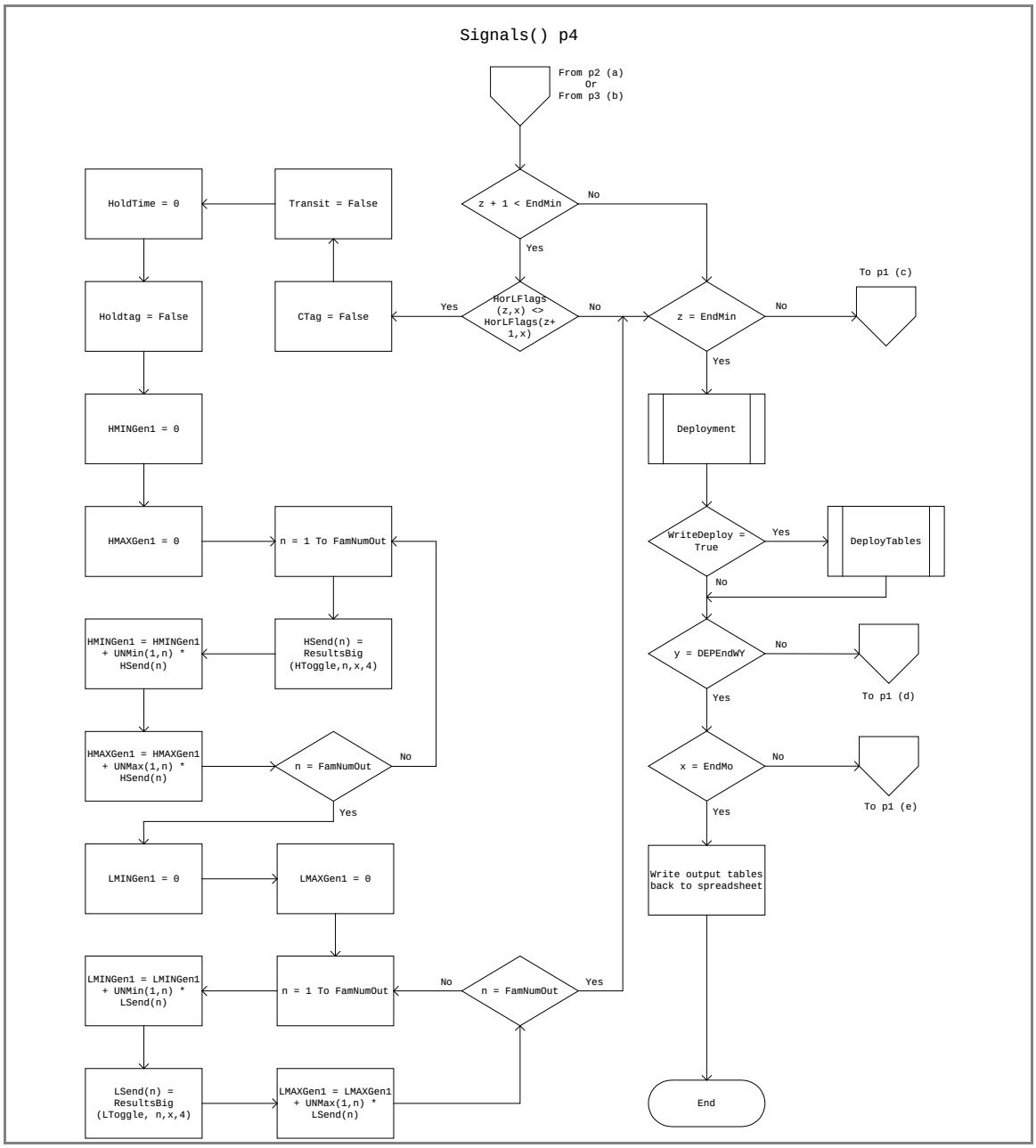
Disclaimer This document and associated models are not final versions and are subject to change for development of the WP-12 Initial Proposal. Additionally, any data contained within the models or documentation are illustrative only and are not indicative of Initial Proposal development.



Disclaimer This document and associated models are not final versions and are subject to change for development of the WP-12 Initial Proposal. Additionally, any data contained within the models or documentation are illustrative only and are not indicative of Initial Proposal development.



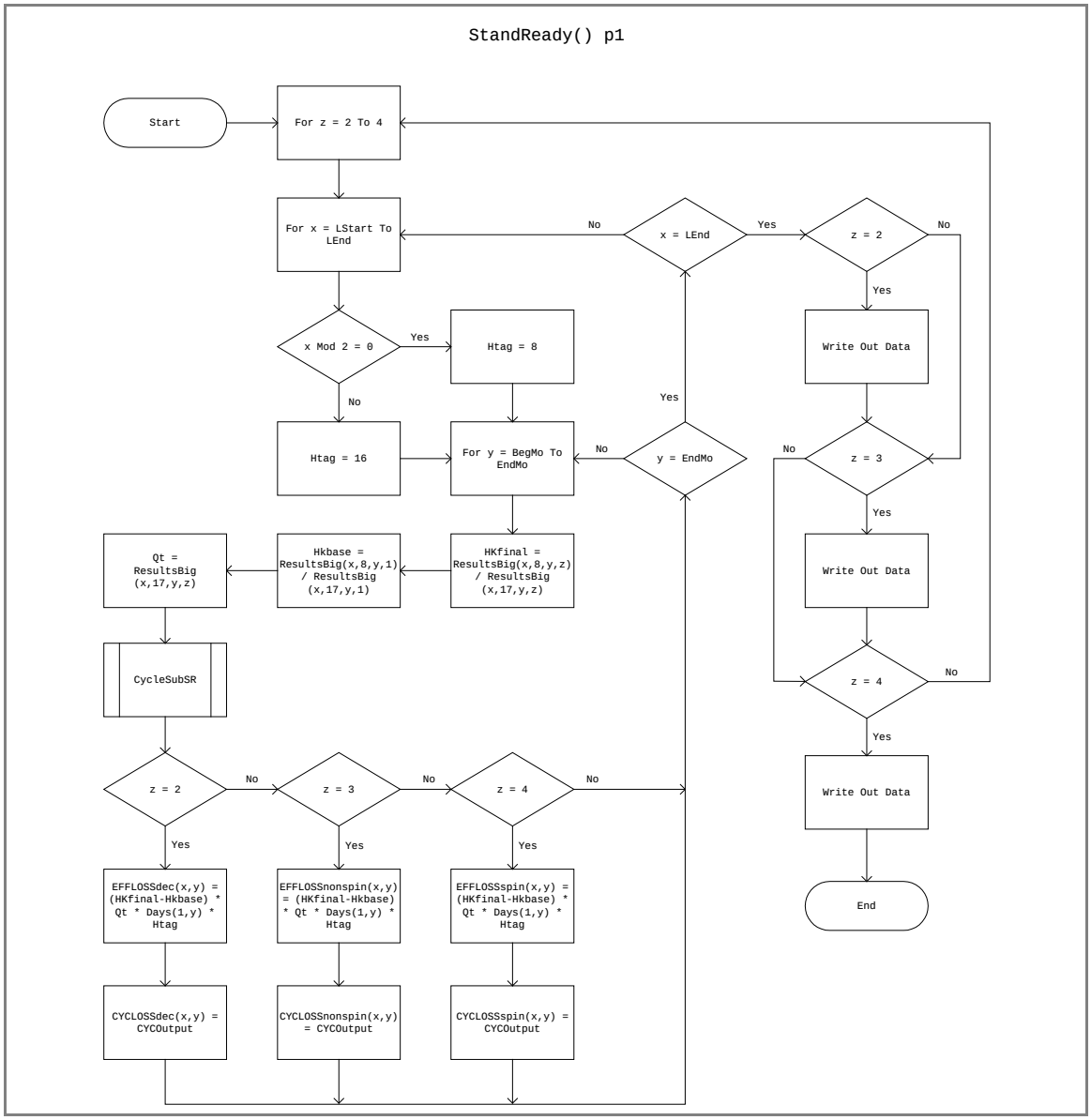
Disclaimer This document and associated models are not final versions and are subject to change for development of the WP-12 Initial Proposal. Additionally, any data contained within the models or documentation are illustrative only and are not indicative of Initial Proposal development.



Disclaimer This document and associated models are not final versions and are subject to change for development of the WP-12 Initial Proposal. Additionally, any data contained within the models or documentation are illustrative only and are not indicative of Initial Proposal development.

StandReady() Process Flow

The StandReady() subroutine tracks an records the component impacts associated with making reserve capability available.



Disclaimer This document and associated models are not final versions and are subject to change for development of the WP-12 Initial Proposal. Additionally, any data contained within the models or documentation are illustrative only and are not indicative of Initial Proposal development.