Specification: bnf algorithm (beta-Normal feedback)

1. Overview:

The purpose of this algorithm is to control neutral beam injection (NBI) to target a given value of β_N, β_T, or W_{MHD}. A proportional/integral (PI) gain scheme is used to adjust the injected power to achieve a desired value of these target quantities. The NB modulation is then adjusted to achieve the desired value of injected power.

This should (hopefully) improve NSTX operations in two regards:

- Allow reliable operation at β_N (or W_{MHD} or β_T) levels just below instability thresholds.
- Accommodate transient changes in confinement with minimal changes in stored energy content (for fixed B_T, I_P, plasma volume).

Note that not all sections of the algorithm described here will be implemented in the first version. However, by specifying all possible inputs in this first iteration, upgrades to the code are allowed without naming new algorithms.

2. Algorithm:

2.1: Proportional/Integral Gain Schemes

In all cases, there is an error defined as the difference between the requested and actual quantity, a proportional gain constant of order unity, a constant C which connects the proportional gain the quantity to be controlled, and an integral gain parameter to help in feeding back to zero. The constant C is evaluated using typical values, i.e. I_P=900 kA, B_T=0.45 T, a=0.6 m, V=12m^3. A further upgrade to the algorithm could allow for some of these three fixed parameters to be derived from the PCS requests in other control categories.

2.1.1: For β_N feedback

The definition of β_N is:

$$\beta_T = \frac{2\mu_0 W_{MHD}}{VB_T^2}, \quad \beta_N = 100\frac{\beta_T a B_T}{I}$$

where I_P is in MA. Combining these yields β_N in terms of W_{MHD}, and vice-versa:

$$\beta_N = 100\frac{a}{I_P}\frac{2\mu_0 W_{MHD}}{VB_T} \Rightarrow W_{MHD} = \beta_N \frac{I_P V B_T}{200\mu_0 a}$$

The relationship between W_{MHD} and injected power is given by:

$$W_{MHD} = \frac{P_{inj}}{\tau},$$

where τ is here taken as a constant independent of heating power. Adjustment of this parameter to account for power degradation of confinement could be a future upgrade. Combining these yields the change in injection power for a given change in β_N.

$$\Delta P_{inj} = \tau \frac{I_P V B_T}{200 \mu_0 a} \Delta \beta_N$$

The feedback algorithm will be simple proportional/integral gain, with the following equation:

$$\Delta P_{inj} = G_{\beta_N} \cdot \overline{C}_{\beta_N} \cdot e + I_{\beta_N} \cdot \overline{C}_{\beta_N} \cdot \int e dt$$

$$e = \beta_{N,RTEFIT} - \beta_{N,reqeust}$$

$$\overline{C}_{\beta_N} = \tau \frac{I_P V B_T}{200 \mu_0 a}$$

G is a gain coefficient of order unity, and $\overline{C}$ is evaluated using typical values, i.e. I_P=900 kA, B_T=0.45 T, a=0.6 m, V=12m^3. A further upgrade to the algorithm could allow for some of these parameters to be derived from the PCS requests in other control categories.

2.1.2: For W_{MHD} feedback *(input waveforms specified, but code likely delayed)*

The relationship between W_{MHD} and injected power is given by:

$$W_{MHD} = \frac{P_{inj}}{\tau},$$

where τ is here taken as a constant independent of heating power. Combining these yields the change in injection power for a given change in β_N.

$$\Delta P_{inj} = \tau \Delta W_{MHD}$$

The feedback algorithm will be simple proportional/integral gain, with the following equation:

$$\Delta P_{inj} = G_{W_{MHD}} \cdot \overline{C}_{W_{MHD}} \cdot e + I_{W_{MHD}} \cdot \overline{C}_{W_{MHD}} \cdot \int e dt$$

$$e = W_{MHD,RTEFIT} - W_{MHD,reqeust}$$

$$\overline{C}_{W_{MHD}} = \tau$$

2.1.3: For β_T feedback *(input waveforms specified, but code likely delayed)*

$$\beta_T = \frac{2\mu_0 W_{MHD}}{V B_T^2} \Rightarrow \Delta W_{MHD} = \frac{V B_T^2}{2\mu_0} \Delta \beta_T$$

The feedback algorithm will be simple proportional/integral gain, with the following equation:

$$\Delta P_{inj} = G_{\beta_T} \cdot \overline{C}_{\beta_T} \cdot e + G_{\beta_T} \cdot \overline{C}_{\beta_T} \cdot \int e dt$$

$$e = \beta_{T,RTEFIT} - \beta_{T,reqeust}$$

$$\overline{C}_{\beta_T} = \tau \frac{VB_T^2}{2\mu_0}$$

2.1.4: For β_P feedback *(input waveforms specified, but code likely delayed)*

2.1.5: For pre-programmed injected power specifications.

The methods in sections 2.1.1-2.1.4 specify a method to define the requested input power. It should also be possible to specify the input power directly, through a floating step waveform. The algorithm described in 2.2 will then adjust the modulation rate to achieve that power. However, for this to achieve the correct power, the values in the "Source Power" Subcategory (#9) must be entered correctly.

2.2: Adjustment of Injected Power

The PI gain scheme produces a requested increase or decrease in the heating power, based upon deviation from the target. It is then necessary to adjust the NB modulation to achieve the requested power. The following requirements hold for the algorithm that adjusts to beams:

- There should be the ability to pre-program one, two, or three sources. This would allow all three sources to be preprogrammed for the initial phase, or for modulation on A to be prevented, so that MSE data is collected without interruption. This programming is specified in the *"Preprogrammed"* subcategory.
- A priority order should be specified. This is specified as a parameter data block.
- There should be minimum on and off times, with separate values for each of the sources.[1] These are specified in the *"On Off Times"* subcategory.
- There should be a threshold |error|, beneath which no notches are issued or removed. This is specified in the "Minimum Error" waveform for each category.

The final binary (on or off) request for a given source is derived from either the β_N feedback determination or the preprogrammed input. Which of these determines the request is determined by the "Src. X. Feedback On/Off" waveforms, where X={A, B, or C}.

[1] Separate minimum times for each source can be useful for MSE, where a 10msec on-time for Source A may not be enough for the voltage to settle.

The feedback algorithm should determine the beam on/off requests using the following scheme. The sources are ranked in order or important A, B, C. Hence, C will always be the last on in a beta ramp-up, and the first off in a beta ramp down. This order is required because the sources themselves are tuned to certain line voltages, which deviate when the source come on.

The duty cycles for the sources are determined as follows in idl code, where SourcePowers is a three element array containing the approximate powers of the three sources (the detailed implementation will have to change somewhat in PCS code):

```
if NewPower gt total(SourcePowers) then begin
        SourceDutyCycles=[1,1,1]
        return,SourceDutyCycles
endif

if NewPower gt (SourcePowers[0]+SourcePowers[1]) then begin
        SourceDutyCycles=[1,1,(NewPower(SourcePowers[0]+SourcePowers[1]))/(SourcePowers[2])]
        return,SourceDutyCycles
endif

if NewPower gt (SourcePowers[0]) then begin
        SourceDutyCycles=[1,(NewPower-(SourcePowers[0]))/(SourcePowers[1]),0]
        return,SourceDutyCycles
endif

if NewPower lt (SourcePowers[0]) And NewPower gt 0 then begin
        SourceDutyCycles=[(NewPower)/(SourcePowers[0]),0,0]
        return,SourceDutyCycles
endif
```

Having determined the duty cycle, the decision to modulate the source (or not) is determined by the following method. SourceOnTimes and SourceOffTimes contain the times when the source was last turned on or off.

```
IF Duty Cycle is 1, THEN turn the source on
IF Duty Cycle is 0, THEN turn the source off

IF 1>DutyCycle>0 then
        T=max([ MinOffTime/(1-SourceDutyCycles),MinOnTime/(SourceDutyCycles) ])

                IF SourceStatus eq 1 then begin
                        IF time-SourceOnTimes gt SourceDutyCycles*T then begin
                                SourceStatus=0
                                SourceOffTimes=time
                         ENDIF
                ENDIF

                IF SourceStatus[i] eq 0 then begin
                        IF time-SourceOffTimes gt (1-SourceDutyCycles)*T then begin
                                SourceStatus=1
```

```
                    SourceOnTimes=time
            ENDIF
        ENDIF

ENDIF
```

2.3: Filtering of Target Waveforms

It is likely that the β and W_{MHD} targets from RT-EFIT will have some noise, and may completely fail for some time-slices. In order to counteract this likely problem, the following two additional

- For each target, there is a maximum allowed change hard-coded in the algorithm. If the change from one time-slice to the next exceeds this value, then the previous value is used. Typical values might be 0.5 for β_N feedback, 3 for β_T feedback , and 30kJ for W_{MHD} feedback.
- In order to minimize the number of unnecessary notches, it is desirable to lowpass filter the targets before using them to compute error. The filter will be the same lowpass filter utilized to filter the SPA requests. The value of this filter time constant is specified in "Target Filter Time Constant" in the "Main" subcategory.

3. Control Waveforms:

In "Main" Subcategory (0):

Name	**type**	**Units**	**Min**	**Max**
Target Filter Time Constant	Floating Step	sec	0	.1
Feedback On/Off	Integer Step	none	0	1
Feedback Control	Integer Step	none	0	5

The values of Feedback Control are as follows:

- 1: β_N Feedback
- 2: β_T Feedback
- 3: β_P Feedback
- 4: W_{MHD} Feedback
- 5: P_{NBI} control through modulation

In "BetaN" Subcategory (1):

Name	**type**	**Units**	**Min**	**Max**
BetaN Proportional Gain	Floating Step	none	0	500
BetaN Integral Gain	Floating Step	1/sec	0	500

BetaN Target	Floating Step	none	0	10
BetaN Minimum Error	Floating Step	none	0	5

In "BetaT" Subcategory (2):

Name	**type**	**Units**	**Min**	**Max**
BetaT Proportional Gain	Floating Step	none	0	500
BetaT Integral Gain	Floating Step	1/sec	0	500
BetaT Target	Floating Step	%	0	1
BetaT Minimum Error	Floating Step	none	0	1

In "W_{MHD}" Subcategory (5):

Name	**type**	**Units**	**Min**	**Max**
Wmhd Proportional Gain	Floating Step	none	0	500
Wmhd Integral Gain	Floating Step	1/sec	0	500
Wmhd Target	Floating Step	MJ	0	2
Wmhd Minimum Error	Floating Step	MJ	0	1

In "BetaP" Subcategory (6):

Name	**type**	**Units**	**Min**	**Max**
BetaP Proportional Gain	Floating Step	none	0	500
BetaP Integral Gain	Floating Step	1/sec	0	500
BetaP Target	Floating Step	none	0	5
BetaP Minimum Error	Floating Step	none	0	5

In "P_{inj}" Subcategory (7):

Name	**type**	**Units**	**Min**	**Max**
P_{inj} Target	Floating Step	MW	0	15

In "Preprogrammed" Subcategory (8):

Name	**type**	**Units**	**Min**	**Max**
Src. A Preprogram	Integer Step	None	0	1
Src. A. Feedback On/Off	Integer Step	None	0	1
Src. B Preprogram	Integer Step	None	0	1
Src. B. Feedback On/Off	Integer Step	None	0	1
Src. C Preprogram	Integer Step	None	0	1
Src. C. Feedback On/Off	Integer Step	None	0	1

In "Source Power" Subcategory (9):

These numbers should be input by the PCS operator to be consistent with the request made to the beam operator. Small errors will not matter much when doing feedback, but best to be accurate.

Name	**type**	**Units**	**Min**	**Max**
Src. A Power	Floating Step	MW	0	3
Src. B Power	Floating Step	MW	0	3
Src. C Power	Floating Step	MW	0	3

In "On Off Times" Subcategory (10):
These numbers are used in determining whether a given source can be turned on or off during the adjustment of the injected power.

Name	**type**	**Units**	**Min**	**Max**
Src. A Min On Time	Floating Step	sec	0.01	1
Src. A Min Off Time	Floating Step	sec	0.01	1
Src. B Min On Time	Floating Step	sec	0.01	1
Src. B Min Off Time	Floating Step	sec	0.01	1
Src. C Min On Time	Floating Step	sec	0.01	1
Src. C Min Off Time	Floating Step	sec	0.01	1

In "Enable Window" Subcategory (11):
These waveforms are transferred to the outputs, to notify the NB hardware that PCS considers itself to be in charge. Also, the actual source requests will be zero at any time that these are zero.

Name	**type**	**Units**	**Min**	**Max**
Source A Control Enabled	Integer Step	none	0	1
Source B Control Enabled	Integer Step	none	0	1
Source C Control Enabled	Integer Step	none	0	1

Source Order
Finally, there is a parameter datablock which specifies the prioritization of the sources in terms of source order. This is a three element array, with the first element corresponding to the "highest priority" source, the second element to the second highest priority, and so on. In this case, a "high priority" source will be modulated before a "low priority" source. Sad another way, the lowest priority source will be the last one modulated. A typical priority array would thus typically look like [3,2,1]=[C,B,A].

4. Archived Quantities:

All the above waveforms should be archived. In addition, the following internal calculations should be archived:

- The three feedback source waveforms.[2]
- The error vector.

5. Outputs to FPDB-I/O:

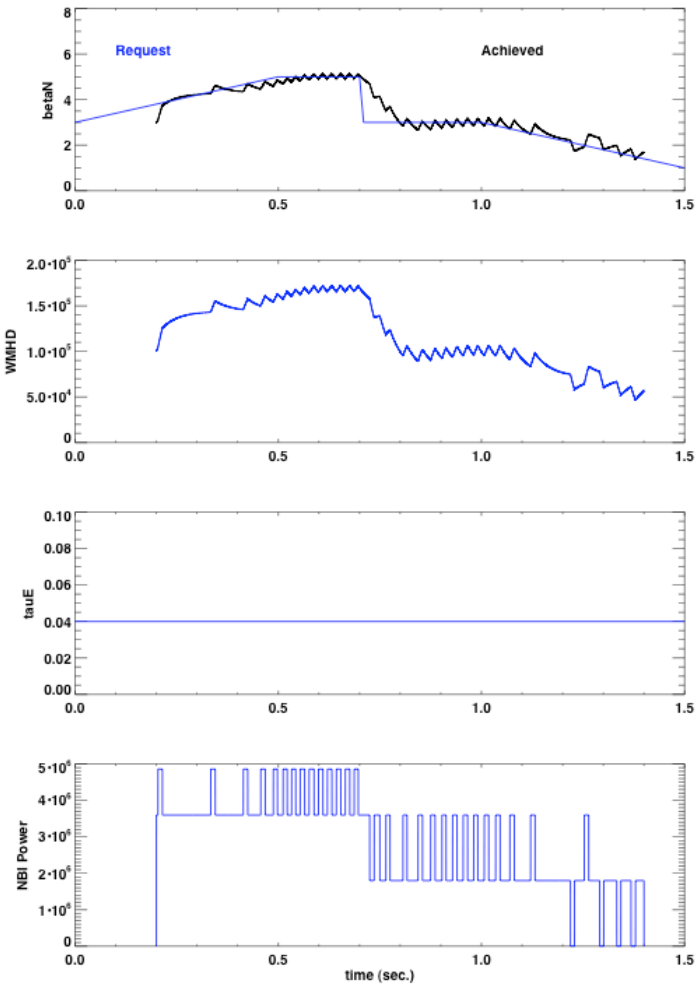
The following six binary quantities are generated by this algorithm, sent out through FPDP-I/O, and used by the NBI control hardware.

Name	Min	Max
A Control Enabled	0	1
B Control Enabled	0	1
C Control Enabled	0	1
A On/Off	0	1
B On/Off	0	1
C On/Off	0	1

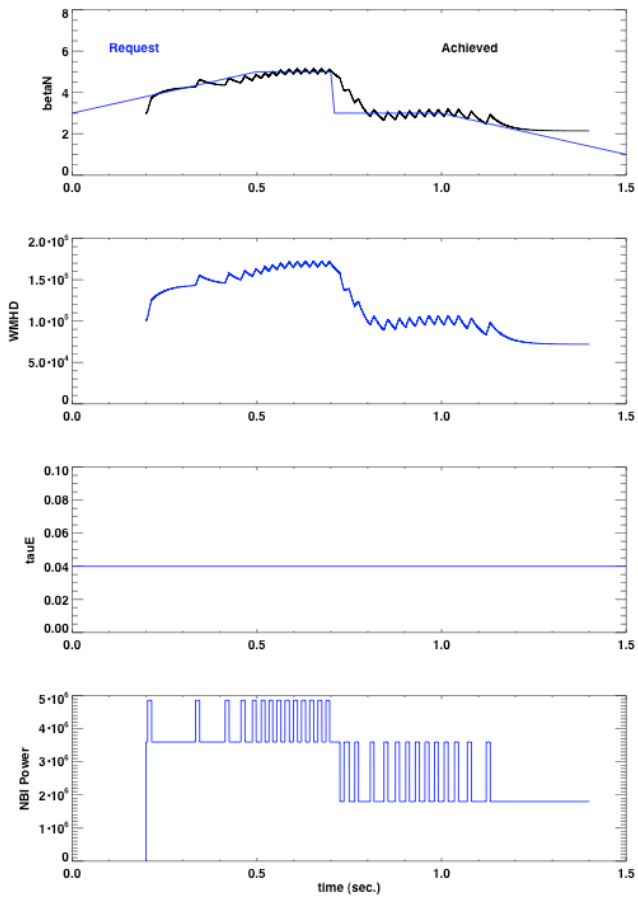
[2] Although the feedback is always calculated, it may differ from the final request if the "Src. X. Feedback On/Off" value is set to 0.

6. Examples of Performance:

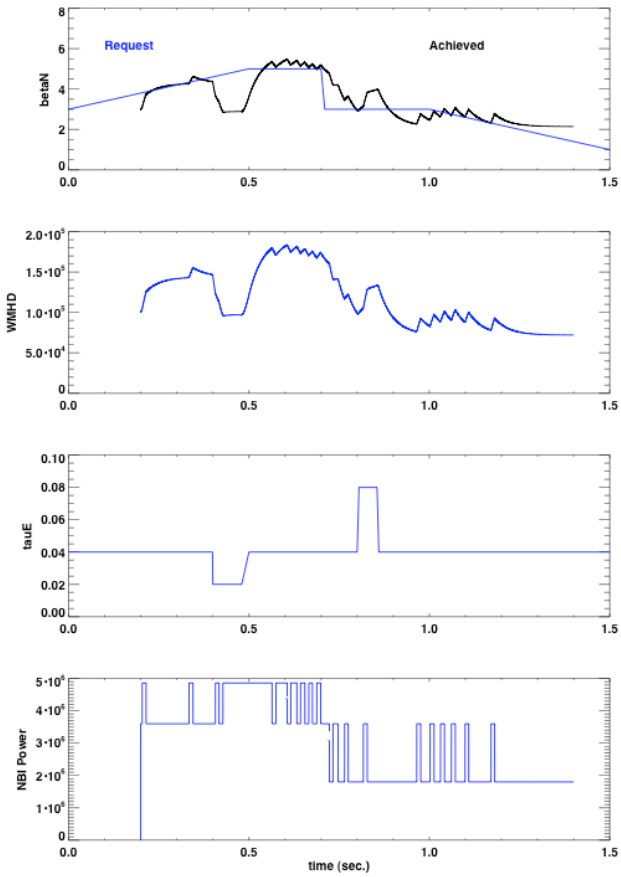
A simple simulation of this algorithm was coded in idl, and used to simulate β_N feedback. Note that the error threshold is set to zero in all of these examples. The gain parameter is set to 20.



Example 1: In this example, Source A is allowed to modulate 10 On/10 Off. The target β_N value (blue in top trace) is largely achieved. There are no excursions in the energy confinement.



Example 2: This is the same as Example 1, but source A is not allowed to modulate. Thus, the late β_N ramp-down cannot be achieved.



Example 3: This is the same as Example 1, but with transient in the energy confinement time. The algorithm tries to compensate, but cannot do so perfectly. Note that source A is not allowed to modulate in this case.