Autopoietic Bloom

For *Amatria*

Composer's Guide

Table of Contents

Improvisational Schemata

Autopoietic Bloom analyzes breakpoint improvisational schemata to determine the large-scale musical and formal design for each improvisation. These schemata are written in the JSON format. The following is a list of each parameter and a description including default settings and proper syntax. Consult the Glossary at the end of this document for a more thorough explanation of specific terminology.

Root Level Parameters

`duration`

The total duration of the improvisation in minutes (float). Default is 3.0.

`breakpoints`

An array of **key-value pairs**, each one representing a point in a breakpoint function that *Autopoietic Bloom* will follow to shape the improvised piece. The **key** for each pair is a float between 0.0 and 1.0 that specifies the timestamp of each breakpoint as a percentage of the total duration, while the associated **value** for each breakpoint contains nested objects which together describe all the data for every parameter that will be parsed by *Autopoietic Bloom* at that point in the piece.

Breakpoint Parameters

`tempo`

The tempo in beats-per-minute at the specified breakpoint (float or int). Automatically interpolates linearly between breakpoints.

`donatoni`

A key, the value for which contains all the data for every parameter that will be parsed by *Autopoietic Bloom's* improvisation engine, *Donatoni*.

`syrinx`

A key, the value for which contains all the data for every parameter that will be parsed by *Autopoietic Bloom's* multichannel modular synthesizer, *Syrinx*.

`quincy`

A key, the value for which contains all the data for every parameter that will be parsed by *Autopoietic Bloom's* dynamic multichannel mixer, *Quincy*.

Donatoni Parameters

`voice001, voice002, voice003, voice004`

Key-value pairs that denote each of *Donatoni's* voices. *Donatoni* currently supports up to four voices improvising independently. The value for each key is an object that contains the data for every parameter of the specified voice that *Donatoni* will parse (at the specified breakpoint).

Donatoni Voice Parameters

`active`

An integer value (1 or 0) that specifies whether a voice will be used in the current improvisation. Any voice whose `active` value is 0 at the first breakpoint ("0.0"), will not receive a clock signal and thus will not be used. The `active` value is also used to gate the clock signal to each voice's corresponding *Syrinx* synth, conserving processing power.

N.B.: the `active` parameter is **only read at the first breakpoint** ("0.0"). The value of `active` at any other breakpoint **will not be read**. Thus, if you set the `active` value of a voice to 0 at the beginning of a piece, you will not be able to "turn it on" later in the piece. Conversely, if a voice's `active` value is set to 1, changing it to 0 later in the piece will have no effect. To turn voices on and off over the course of a piece, use the next parameter, `playStop`.

`playStop`

An integer value (1 or 0) that denotes whether a specified voice is currently playing. Play = 1, Stop = 0.

`MIDIChannel`

An integer value (1 to 16) that denotes which MIDI channel will be used to transmit the note information out from *Donatoni* to *Syrinx*. for the specified voice. By default, each `MIDIChannel` is the same as the specified voice number (e.g., voice 1 uses channel 1, voice 2 uses channel 2, etc).

`pitches`

A Lisp-like linked list of integers or floats in **bach** format that specifies the pitches in the **ur-gesture** upon which *Autopoietic Bloom* will improvise. All pitches, notated in midi-cents (i.e. a midi note value of 60 would be 6000 bach midi-cents), **must** be surrounded by two sets of square brackets [[]]. To specify a chord, midi-cent values must be ensconced in an additional set of square brackets []. A rest is denoted either by the value `[null]` (in square brackets) or by `nil` (without square brackets). For more on Lisp-like linked lists and **bach** syntax and notation, consult **bach's** documentation.

The default value for `pitches` is `"[[6000]]"` which specifies a single note at middle C.

Warning: The number of chords (including single-note "chords" and rests (i.e., empty chords)) must match the number of durations specified in the `rhythm` parameter.

Examples:

An ascending C-major arpeggio beginning at middle C would be written as

```
"[[6000 6400 6700]]"
```

A C-major chord (all three voices playing simultaneously) would be written as

```
"[[[6000 6400 6700]]]"
```

An ascending C-major arpeggio with each note separated by a rest and ending on a full chord would be written as

```
"[[6000 [null] 6400 [null] 6700 nil [6000 6400
6700 7200]]]"
```

`rhythm`

A Lisp-like linked list in **bach** format of rationals that specifies the rhythm in the **ur-gesture** upon which *Autopoietic Bloom* will improvise. All values, notated as fractions of a whole note (i.e. a quarter note would be `"1/4"`, an eighth note triplet would be `"1/12"`, etc.), **must** be surrounded by two sets of square brackets [[]]. Note that while **bach** uses negative rationals to denote rests, in *Autopoietic Bloom*, durations are always positive and rests are specified in the `pitches` parameter.

The default value is `"[[1/2]]"`, which denotes a single half note.

Warning: The number of durations must match the number of chords (including single-note "chords" and rests (i.e., empty chords)) specified in the `pitches` parameter.

Examples:

A series of 4 eighth notes would be written as

```
"[[1/8 1/8 1/8 1/8]]"
```

A series of 4 eighth notes rests would be written as

```
"[[1/8 1/8 1/8 1/8]]"
```

A sixteenth note quintuplet truncated after its fourth note followed by a quarter note would be written as

```
"[[1/20 1/20 1/20 1/20 1/4]]"
```

`phraseLength`

An integer specifying the number of times a gesture will be transformed before re-calculating a new set of transformations. The longer the phrase length, the less responsive the phrases will be to changes in **improvisational flux** (Φ), as transformation values using the current Φ will not be calculated until the beginning of the next phrase.

Specifying the phrase length is especially useful when providing a list of manual transformation values for **Autopoietic Bloom** to use for any given transformation; `phraseLength` should be the lowest common multiple of the length of the list(s) of manual transformation values. If no manual values are being used, `phraseLength` can remain at its default value of 5.

`baseRests`

A list of rationals (enclosed in a single set of square brackets []) or the integer 0, specifying the duration of rests to be appended to the pitch content of the gesture after each iteration. Default value is 0.

`maxRests`

An integer value denoting the maximum number of additional rests to be randomly added to `baseRests` at the end of each gesture. The length of these additional rests is specified by `restAtom`. Default value is 0.

`restAtom`

A rational denoting the duration of the additional rests that will be randomly added to `baseRests` at the end of each gesture if `maxRests` is greater than 0. Default value is 1/16 (i.e., sixteenth note rests).

`transform`

An object containing data that describes the various transformations that will be applied to the **ur-gesture** in a particular voice at a particular breakpoint.

Donatoni Gesture Transformation Parameters

`transposition`

An object containing data that describes how the pitches in the **ur-gesture** will be transposed in a particular voice at a particular breakpoint.

`onOff`

An integer value (1 or 0) that specifies whether transposition will be applied to the pitches of the **ur-gesture** in a particular voice at a particular breakpoint.

`manualValues`

A list of integers, positive or negative, denoting the explicit transposition values or maximum/minimum transposition values to be applied to the **ur-gesture** in a particular voice at a particular breakpoint.

Donatoni will cycle through this list of values continuously, stepping to the next value in the list each time it generates a new gesture.

The default value is 16. If `"null"`, `manualValues` will revert to default.

N.B.: If `flux` is set to 0, `manualValues` denotes the explicit transposition values to be applied with each transformation of the **ur-gesture**. If `flux` is greater than 0, `manualValues` denotes the upper bound (or lower bound if negative) of the transposition values to be applied with each transformation of the **ur-gesture**, with the exact transposition values being randomized within that range.

`modeOfTransp`

A list of three Lisp-like linked lists that contain integers that encode the mode of transposition and registral range within which *Donatoni* will transpose the **ur-gesture** using the following syntax:

[pitch classes in mode (0-11)][base pitch (0-11)][octave range (0-9)(1-10)]

Numbers 0 through 11 in the first two lists correspond to the notes C-B in the chromatic scale.

All modes of transposition are circular, i.e., if a transposition value were to exceed the number of pitches in a mode, it will wrap around to the bottom of the mode. E.g., let's say your mode of transposition were: [0 2 4 7 9][1][5 6], the five black notes between C#4 on the piano (6100 midi-cents) and A#4 on the piano (7000 midi-cents) and your ur-gesture were a single note, C#4 on the piano (6100). If your transposition value were 4, the resultant note would be A#4 on the piano (7000). If your transposition value were 5, it would wrap back around and the resultant note would once again be 6100.

The default value is the fully chromatic set from 0 midi-cents (C-2 off the bottom of the piano) to 10700 midi-cents (B7 on the piano) inclusive. `modeOftransp` also takes the message `"reset"` to revert to its default values.

N.B.: When selecting the octave range, note that octave 5-6 corresponds to the notes beginning at middle C (6000 midi-cents) to the B above that (7100 midi-cents). This differs from the convention that C4 is middle C. In **bach**, middle C is C5. Throughout this document, midi-cents will be provided to avoid confusion or the two conventions will be disambiguated thus: "C4 on the piano" refers to middle C (6000 midi-cents); "**bach** C4" refers to the C below middle C (4800 midi-cents).

Warning: if the **ur-gesture** contains notes that are not in the `modeOfTransp`, unexpected errors may arise.

Examples:

The default fully chromatic set across all nine octaves would be written as

```
"[0 1 2 3 4 5 6 7 8 9 10 11][0][0 10]"
```

The mode of C-major across all nine octaves would be written as

```
"[0 2 4 5 7 9 11][0][0 10]"
```

The mode of F-major across all nine octaves would be written as

```
"[0 2 4 5 7 9 11][5][0 10]"
```

The mode of C-mixolydian across all nine octaves would be written as

```
"[0 2 4 5 7 9 10][0][0 10]"
```

The mode of F-major, beginning on C (scale degree 5) and ending on B♭ (scale degree 4), across all nine octaves would be written as

```
"[0 2 4 5 7 9 10][0][0 10]"
```

`flux`

Improvisational Flux (**Φ**) of the transposition function. A float between 0.0 and 1.0 that affects 1) how often a new transposition value will be generated and 2) how large that value will be. See Glossary for more on **Improvisational Flux**.

If set to 0, no randomization will occur and transposition function will simply step through the list of transposition values specified in `manualValues`. If greater than 0, transposition values will be bounded by the value(s) specified in `manualValues`. If `manualValues` is `"null"`, the maximum transposition value will be 16.

Interpolates linearly between breakpoints.

`polarity`

An integer value (0 or 1) that specifies whether transposition values will be generated in a monopolar or bipolar manner when `flux` is greater than 0.

If flux is greater than 0,

If 0, transposition is monopolar and positive and negative values are obeyed as specified in `manualValues`. If no `manualValues` are specified, the default positive value of 16 is used.

If 1, transposition is bipolar and the sign of new transposition values will be randomized.

If flux is 0, polarity is monopolar.

Default value is 0 (monopolar).

`inversion`

An object containing data that describes how the pitches in the **ur-gesture** will be inverted in a particular voice at a particular breakpoint.

`onOff`

An integer value (1 or 0) that specifies whether inversion will be applied to the pitches of the **ur-gesture** in a particular voice at a particular breakpoint.

`manualValues`

A list of midi-cents (integers or floats), denoting the pitch around which to invert the gesture if `flux` is 0.

Donatoni will cycle through this list of values continuously, stepping to the next value in the list each time it generates a new gesture.

If 0 or `"null"`, no inversion will occur. If `flux` is greater than 0, *Donatoni* will disregard `manualValues`.

`flux`

Improvisational Flux (Φ) of the inversion function. A float between 0.0 and 1.0 that affects 1) how often a new inversion value will be generated and 2) how far from the pitch centroid of the **ur-gesture** that inversion value will be. See Glossary for more on **Improvisational Flux**.

If `flux` is close to 0, inversion will occur around the pitch centroid of the **ur-gesture**. As `flux` approaches 1, pitches will invert around a randomly selected pitch within a pitch space the bounds of which approach the highest and lowest notes in the **ur-gesture**. Thus, assuming a `flux` of 1, the highest pitch around which the inversion function will invert a gesture is the highest pitch in its **ur-gesture**, and the lowest pitch around which the inversion function will invert a gesture is the lowest pitch in its **ur-gesture**.

If set to 0, no randomization will occur and the inversion function will simply step through the list of inversion values specified in `manualValues`.

Interpolates linearly between breakpoints.

`pRetro`

An object containing data that describes how the pitches in the **ur-gesture** will be retrogressed in a particular voice at a particular breakpoint.

`onOff`

An integer value (1 or 0) that specifies whether retrogression will be applied to the pitches of the **ur-gesture** in a particular voice at a particular breakpoint.

`manualValues`

A list of integers (1 or 0) denoting whether the pitches are arranged normally (0) or in retrograde (1).

Donatoni will cycle through this list of values continuously, stepping to the next value in the list each time it generates a new gesture.

If 0 or `"null"`, no retrogression will occur. If `flux` is greater than 0, *Donatoni* will disregard `manualValues`.

`flux`

Improvisational Flux (Φ) of the pitch retrogression function. A float between 0.0 and 1.0 that affects how often the pitch retrogression function will be applied to the **ur-gesture**. See Glossary for more on **Improvisational Flux**.

If set to 0, no randomization will occur and the rhythm retrogression function will simply step through the list of retrogression values specified in `manualValues`.

Interpolates linearly between breakpoints.

`pRot`

An object containing data that describes how the pitches in the **ur-gesture** will be rotated in a particular voice at a particular breakpoint.

`onOff`

An integer value (1 or 0) that specifies whether rotation will be applied to the pitches of the **ur-gesture** in a particular voice at a particular breakpoint.

`manualValues`

A list of integers, positive or negative, denoting the rotation values to be applied to the **ur-gesture** in a particular voice at a particular breakpoint.

Donatoni will cycle through this list of values continuously, stepping to the next value in the list each time it generates a new gesture.

If 0 or `"null"`, no rotation will occur. If `flux` is greater than 0, *Donatoni* will disregard `manualValues`.

`polarity`

An integer value (0, 1, or 2) that specifies the polarity of the rotation values generated by *Donatoni* if `flux` is greater than 0.

If `flux` is greater than 0,
If 0, *Donatoni* will generate only positive rotation values.
If 1, *Donatoni* will generate both positive and negative values.
If 2, *Donatoni* will generate only negative rotation values.

If `flux` is 0, *Donatoni* will disregard `polarity` and respect the sign of rotation values specified in `manualValues`.

The default value is 1.

`flux`

Improvisational Flux (Φ) of the pitch rotation function. A float between 0.0 and 1.0 that affects 1) how often a new rotation value will be generated and 2) how large a rotation value—relative to the number of notes in the **ur-gesture**—will be. See Glossary for more on **Improvisational Flux**.

A `flux` near 0 will infrequently rotate the pitches by relatively few steps, while a `flux` of 1 will rotate the pitches constantly at great intervals with respect to the number of pitches in the **ur-gesture**.

If set to 0, no randomization will occur and the pitch rotation function will simply step through the list of rotation values specified in `manualValues`.

Interpolates linearly between breakpoints.

`pScramb`

An object containing data that describes how the pitches in the **ur-gesture** will be scrambled in a particular voice at a particular breakpoint.

`onOff`

An integer value (1 or 0) that specifies whether scrambling will be applied to the pitches of the **ur-gesture** in a particular voice at a particular breakpoint.

`manualValues`

A list of integers (0, 1, or 2) denoting whether the gesture is unscrambled (0), remains scrambled (1), or is re-scrambled (2).

Donatoni will cycle through this list of values continuously, stepping to the next value in the list each time it generates a new gesture.

If 0 or `"null"`, no pitch scrambling will occur. If `flux` is greater than 0, *Donatoni* will disregard `manualValues`.

`mode`

An integer value (0 or 1) that specifies whether *Donatoni* will exclude the unscrambled version of the gesture among its randomized pitch scrambling values, if `flux` is greater than 0.

If `flux` is greater than 0,

> If 0, *Donatoni* will include unscrambled versions of the gesture in its randomized output (it will generate scrambling values of 0, 1, or 2).
> If 1, *Donatoni* will exclude unscrambled versions of the gesture in its randomized output (it will generate scrambling values of 1 or 2).

If `flux` is 0, *Donatoni* will disregard `mode` and respect the scrambling values specified in `manualValues`.

The default value is 0.

`flux`

Improvisational Flux (Φ) of the pitch scrambling function. A float between 0.0 and 1.0 that affects how often the pitches from the ur-gesture will be scrambled. See Glossary for more on **Improvisational Flux**.

A `flux` near 0 will infrequently scramble the pitches, while a `flux` of 1 will scramble the pitches constantly.

If set to 0, no randomization will occur and the pitch scrambling function will simply step through the list of scrambling values specified in `manualValues`.

Interpolates linearly between breakpoints.

`augDim`

An object containing data that describes how the rhythms in the **ur-gesture** will be augmented or diminished in a particular voice at a particular breakpoint.

`onOff`

An integer value (1 or 0) that specifies whether augmentation/diminution will be applied to the rhythms of the **ur-gesture** in a particular voice at a particular breakpoint.

`manualValues`

A list of rationals (each enclosed in square brackets; e.g., `"[1][1/2][3/4]"`) denoting the coefficients by which the durations of the ur-gesture will be multiplied.

Donatoni will cycle through this list of values continuously, stepping to the next value in the list each time it generates a new gesture.

If 0 or `"null"`, no augmentation/diminution will occur and `manualValues` will default to `[1]`. If `flux` is greater than 0, *Donatoni* will disregard `manualValues`.

`flux`

Improvisational Flux (Φ) of the augmentation/diminution function. A float between 0.0 and 1.0 that affects 1) how often the durations from the ur-gesture will be augmented or diminished and 2) how much augmentation/diminution is applied with each transformation. See Glossary for more on **Improvisational Flux**.

A `flux` near 0 will slightly and infrequently augment or diminish the rhythms, while a `flux` of 1 will greatly and constantly augment or diminish the rhythms.

The maximum factor by which durations are multiplied or divided is 6.

If set to 0, no randomization will occur and the augmentation/diminution function will simply step through the list of augmentation/diminution values specified in `manualValues`.

Interpolates linearly between breakpoints.

`rRetro`

An object containing data that describes how the rhythms in the **ur-gesture** will be retrogressed in a particular voice at a particular breakpoint.

`onOff`

An integer value (1 or 0) that specifies whether retrogression will be applied to the rhythms of the **ur-gesture** in a particular voice at a particular breakpoint.

`manualValues`

A list of integers (1 or 0) denoting whether the rhythms are arranged normally (0) or in retrograde (1).

Donatoni will cycle through this list of values continuously, stepping to the next value in the list each time it generates a new gesture.

If 0 or `"null"`, no retrogression will occur. If `flux` is greater than 0, *Donatoni* will disregard `manualValues`.

`flux`

Improvisational Flux (**Φ**) of the rhythm retrogression function. A float between 0.0 and 1.0 that affects how often the rhythm retrogression function will be applied to the **ur-gesture**. See Glossary for more on **Improvisational Flux**.

If set to 0, no randomization will occur and the rhythm retrogression function will simply step through the list of retrogression values specified in `manualValues`.

Interpolates linearly between breakpoints.

`rRot`

An object containing data that describes how the rhythms in the **ur-gesture** will be rotated in a particular voice at a particular breakpoint.

`onOff`

An integer value (1 or 0) that specifies whether rotation will be applied to the rhythms of the **ur-gesture** in a particular voice at a particular breakpoint.

`manualValues`

A list of integers, positive or negative, denoting the rotation values to be applied to the **ur-gesture** in a particular voice at a particular breakpoint.

Donatoni will cycle through this list of values continuously, stepping to the next value in the list each time it generates a new gesture.

If 0 or `"null"`, no rotation will occur. If `flux` is greater than 0, *Donatoni* will disregard `manualValues`.

`polarity`

An integer value (0, 1, or 2) that specifies the polarity of the rotation values generated by *Donatoni* if `flux` is greater than 0.

If `flux` is greater than 0,
If 0, *Donatoni* will generate only positive rotation values.
If 1, *Donatoni* will generate both positive and negative values.
If 2, *Donatoni* will generate only negative rotation values.

If `flux` is 0, *Donatoni* will disregard `polarity` and respect the sign of rotation values specified in `manualValues`.

The default value is 1.

`flux`

Improvisational Flux (**Φ**) of the rhythm rotation function. A float between 0.0 and 1.0 that affects 1) how often a new rotation value will be generated and 2) how large a rotation value—relative to the number of notes in the **ur-gesture**—will be. See Glossary for more on **Improvisational Flux**.

A `flux` near 0 will infrequently rotate the rhythms by relatively few steps, while a `flux` of 1 will rotate the rhythms constantly at great intervals with respect to the number of pitches in the **ur-gesture**.

If set to 0, no randomization will occur and the rhythm rotation function will simply step through the list of rotation values specified in `manualValues`.

Interpolates linearly between breakpoints.

`prRot`

An object containing data that describes how the rhythms and pitches (coupled together) in the **ur-gesture** will be rotated in a particular voice at a particular breakpoint.

`onOff`

An integer value (1 or 0) that specifies whether rotation will be applied to the rhythm and pitch of the **ur-gesture** in a particular voice at a particular breakpoint.

`manualValues`

A list of integers, positive or negative, denoting the rotation values to be applied to the **ur-gesture** in a particular voice at a particular breakpoint.

Donatoni will cycle through this list of values continuously, stepping to the next value in the list each time it generates a new gesture.

If 0 or `"null"`, no rotation will occur. If `flux` is greater than 0, *Donatoni* will disregard `manualValues`.

`polarity`

An integer value (0, 1, or 2) that specifies the polarity of the rotation values generated by *Donatoni* if `flux` is greater than 0.

If `flux` is greater than 0,
If 0, *Donatoni* will generate only positive rotation values.
If 1, *Donatoni* will generate both positive and negative values.
If 2, *Donatoni* will generate only negative rotation values.

If `flux` is 0, *Donatoni* will disregard `polarity` and respect the sign of rotation values specified in `manualValues`.

The default value is 1.

`flux`

Improvisational Flux (**Φ**) of the pitch and rhythm rotation function. A float between 0.0 and 1.0 that affects 1) how often a new rotation value will be generated and 2) how large a rotation value—relative to the number of notes in the **ur-gesture**—will be. See Glossary for more on **Improvisational Flux**.

A `flux` near 0 will infrequently rotate the rhythm and pitch by relatively few steps, while a `flux` of 1 will rotate the rhythm and pitch constantly at great intervals with respect to the number of pitches in the **ur-gesture**.

If set to 0, no randomization will occur and the pitches and rhythm rotation function will simply step through the list of rotation values specified in `manualValues`.

Interpolates linearly between breakpoints.

`rScramb`

An object containing data that describes how the rhythms in the **ur-gesture** will be scrambled in a particular voice at a particular breakpoint.

`onOff`

An integer value (1 or 0) that specifies whether scrambling will be applied to the rhythms of the **ur-gesture** in a particular voice at a particular breakpoint.

`manualValues`

A list of integers (0, 1, or 2) denoting whether the gesture is unscrambled (0), remains scrambled (1), or is re-scrambled (2).

Donatoni will cycle through this list of values continuously, stepping to the next value in the list each time it generates a new gesture.

If 0 or `"null"`, no rhythm scrambling will occur. If `flux` is greater than 0, *Donatoni* will disregard `manualValues`.

`mode`

An integer value (0 or 1) that specifies whether *Donatoni* will exclude the unscrambled version of the gesture among its randomized rhythm scrambling values, if `flux` is greater than 0.

If `flux` is greater than 0,

If 0, *Donatoni* will include unscrambled versions of the gesture in its randomized output (it will generate scrambling values of 0, 1, or 2).
If 1, *Donatoni* will exclude unscrambled versions of the gesture in its randomized output (it will generate scrambling values of 1 or 2).

If `flux` is 0, *Donatoni* will disregard `mode` and respect the scrambling values specified in `manualValues`.

The default value is 0.

`flux`

Improvisational Flux (**Φ**) of the rhythm scrambling function. A float between 0.0 and 1.0 that affects how often the rhythm from the ur-gesture will be scrambled. See Glossary for more on **Improvisational Flux**.

A `flux` near 0 will infrequently scramble the rhythm, while a `flux` of 1 will scramble the rhythm constantly.

If set to 0, no randomization will occur and the rhythm scrambling function will simply step through the list of scrambling values specified in `manualValues`.

Interpolates linearly between breakpoints.

`window`

An object containing data that describes how the **ur-gesture** will be windowed in a particular voice at a particular breakpoint.

`onOff`

An integer value (1 or 0) that specifies whether windowing will be applied to the notes of the **ur-gesture** in a particular voice at a particular breakpoint.

`manualValues`

A list of integer pairs, each pair enclosed in square brackets []. The first number of each pair denotes the ordinal note from the beginning of the **ur-gesture** after which the windowed gesture will begin. The second number of each pair denotes the ordinal note from the end of the **ur-gesture** before which the windowed gesture will end.

Donatoni will cycle through this list of values continuously, stepping to the next value in the list each time it generates a new gesture.

If 0 or `"null"`, no retrogression will occur. If `flux` is greater than 0, *Donatoni* will disregard `manualValues`.

Warning: If the windowing values exceed the number of notes in the gesture, unexpected errors may arise. Bear in mind that the function first truncates the beginning of the melody, then truncates the end of the melody. Therefore, care must be taken that the second number in each pair doesn't exceed the difference of the total number of notes in the **ur-gesture** minus the first number of the windowing value (i.e., the total number of notes in the gesture that are left after the front of it has been truncated).

Examples:

A windowing function with the windowing values [1 1] applied to a five-note melody would truncate the first and last notes from the melody, resulting in a three-note melody consisting of the middle three notes of the original melody.

`flux`

Improvisational Flux (**Φ**) of the windowing function. A float between 0.0 and 1.0 that affects how 1) often a new window will be applied to the **ur-gesture**, and 2) how much variation there is from one window to the next. See Glossary for more on **Improvisational Flux**.

A `flux` close to zero will result in infrequent and light windowing applied to the **ur-gesture**, while a `flux` of 1 will result in constantly changing window sized ranging from very large to very small.

If set to 0, no randomization will occur and the windowing function will simply step through the list of windowing values specified in `manualValues`.

Interpolates linearly between breakpoints.

Syrinx Parameters

`voice001, voice002, voice003, voice004`

Key-value pairs that denote each of *Syrinx's* synths. *Syrinx* currently supports up to four synthesizers/samplers. The value for each key is an object that contains the data for every parameter of the specified synth that *Syrinx* will parse (at the specified breakpoint).

Syrinx Voice Parameters

`progNum`

An integer value corresponding to the program number that a synth of a particular voice will use at a particular timestamp. Each program is essentially an instrument preset. There are up to 128 possible programs.

`volume`

A float value between 0.0 and 1.0 that adjusts the master volume fader for a particular synth.

Interpolates linearly between breakpoints.

`ctrl001, ctrl002, ctrl003, ctrl004`

Float values between 0.0 and 1.0 that correspond to the first four MIDI controllers for each voice.

A value of `"null"` in any of the control values at the first breakpoint ("0.0") of the improvisation scheme will disable that control signal, allowing the program's default settings for that control value to take priority. This behavior is similar to `active` in *Donatoni* above.

N.B.: If `"null"` is applied to a control value the first breakpoint ("0.0"), none of the other breakpoints will be read. Conversely, `"null"` can only disable a control signal if it's used at the first breakpoint ("0.0"). If it is used at subsequent breakpoints, *Interpol* will interpolate over it as though the breakpoint didn't exist. To avoid unexpected errors, only use `"null"` if you know you will not be adjusting a certain MIDI controller away from its defaults for the duration of the piece. When in doubt, use explicit float values.

Quincy Parameters

`qClock`

An object containing data for configuring and controlling *Quincy's* internal clock that drives its transformations independently of *Donatoni* but with respect to the global `tempo`.

`onOff`

An integer value (1 or 0) that specifies whether the `qClock` is turned on or off. Default value is 0.

`beatMult`

An integer value that is multiplied by the global `tempo` to determine the pulse rate of `qClock`. Default value is 1.

`beatDiv`

An integer value that by which four times the global `tempo` is divided to determine the pulse rate of `qClock`. Default value is 4.

Examples:

With an arbitrary global `tempo`, and assuming a quarter note beat, to make the `qClock` pulse every quarter note the `beatMult` would be set to 1 and the `beatDiv` set to 1.

With an arbitrary global `tempo`, and assuming a quarter note beat, to make the `qClock` pulse every half note the `beatMult` would be set to 1 and the `beatDiv` set to 2.

With an arbitrary global `tempo`, and assuming a quarter note beat, to make the `qClock` pulse every two whole notes the `beatMult` would be set to 2 and the `beatDiv` set to 1.

With an arbitrary global `tempo`, and assuming a quarter note beat, to make the `qClock` pulse every half note septuplet the `beatMult` would be set to 2 and the `beatDiv` set to 7.

`qConfig`

An Lisp-like linked list describing how each of *Syrinx's* voices map to *Qunicy's* 10-channel output at the particular timestamp. The list is 10 elements long, each element can consist of an integer—corresponding to the voice number of a *Syrinx* synth—or a list of integers—each integer corresponding to the voice number of a *Syrinx* synth.

A value of 0, `[null]`, or `nil` denotes a muted channel.

The default configuration is arbitrarily `"[1 1 1 2 3 4 3 4 3 4]"`.

Example:

To send voices 1 and 2 through channels 1, 3, and 4; voice 3 to channels 5, 6, and 8; and voice 4 to channels 8 and 9, the appropriate `qConfig` would be:

`"[[1 2][null][1 2][1 2] 3 3 0 [3 4] 4 nil]"`.

N.B.: in this example, 0, `[null]`, and `nil` are used together to demonstrate their interchangeability. In a real-world application one may choose any of these three options to represent a muted channel, but consistency is highly recommended.

`ramp`

The time in milliseconds to crossfade between configurations with upper and lower bounds of 10000ms and 10ms respectively. The default, and recommended minimum, is 40ms.

`transform`

An object containing data that describes the various transformations that will be applied to the `qConfig` at a particular breakpoint.

Quincy Transformation Parameters

`qRot`

An object containing data that describes how *Quincy* rotates the input signals from *Syrinx* rotate through its various output channels at a particular breakpoint.

`onOff`

An integer value (1 or 0) that specifies whether rotation will be applied to the `qConfig` in a particular voice at a particular breakpoint.

`affectedChannels`

An unbracketed list of integers denoting the channels that will be affected by the channel rotation. Affected channels need not be contiguous or adjacent.

Also takes the message `"all"` to select all 10 channels and `"null"` to select none. Default value is `"all"`.

Example:

With a `qConfig` of `"[1 2 1 2 3 0 0 4 0 0]"`, to rotate the signals from synths 3 and 4 around channels 5-10, the appropriate `affectedChannels` value would be:

`"5 6 7 8 9 10"`

`manualValues`

A list of integers, positive or negative, denoting the rotation values to be applied to the `qConfig` at a particular breakpoint.
Quincy will cycle through this list of values continuously, stepping to the next value in the list each time it generates a new gesture.

If 0 or `"null"`, no rotation will occur. If `flux` is greater than 0, *Quincy* will disregard `manualValues`.

`polarity`

An integer value (0, 1, or 2) that specifies the polarity of the rotation values generated by *Quincy* if `flux` is greater than 0.

If `flux` is greater than 0,
If 0, *Quincy* will generate only positive rotation values.
If 1, *Quincy* will generate both positive and negative values.
If 2, *Quincy* will generate only negative rotation values.

If `flux` is 0, *Quincy* will disregard `polarity` and respect the sign of rotation values specified in `manualValues`.

The default value is 1.

`flux`

Improvisational Flux (**Φ**) of the channel rotation function. A float between 0.0 and 1.0 that affects 1) how often a new rotation value will be generated and 2) how large a rotation value—relative to the number of notes in the **ur-gesture**—will be. See Glossary for more on **Improvisational Flux**.

A `flux` near 0 will infrequently rotate the signals by relatively few steps, while a `flux` of 1 will rotate the signals constantly at great intervals

If set to 0, no randomization will occur and the channel rotation function will simply step through the list of rotation values specified in `manualValues`.

Interpolates linearly between breakpoints.

`qScramb`

An object containing data that describes how the channels in the `qConfig` will be scrambled at a particular breakpoint.

`onOff`

An integer value (1 or 0) that specifies whether scrambling will be applied to the channel mapping of the `qConfig` at a particular breakpoint.

`affectedChannels`

An unbracketed list of integers denoting the channels that will be affected by the channel scrambling. Affected channels need not be contiguous or adjacent.

Also takes the message `"all"` to select all 10 channels and `"null"` to select none. Default value is `"all"`.

Example:

With a `qConfig` of `"[1 2 1 2 3 0 0 4 0 0]"`, to scramble the signals in channels 5-10, the appropriate `affectedChannels` value would be:

`"5 6 7 8 9 10"`

`manualValues`

A list of integers (0, 1, or 2) denoting whether the gesture is unscrambled (0), remains scrambled (1), or is re-scrambled (2).

Quincy will cycle through this list of values continuously, stepping to the next value in the list each time it generates a new gesture.

If 0 or `"null"`, no channel scrambling will occur. If `flux` is greater than 0, *Quincy* will disregard `manualValues`.

`mode`

An integer value (0 or 1) that specifies whether *Quincy* will exclude the unscrambled version of the gesture among its randomized rhythm scrambling values, if `flux` is greater than 0.

If `flux` is greater than 0,

If 0, *Quincy* will include unscrambled versions of the gesture in its randomized output (it will generate scrambling values of 0, 1, or 2).
If 1, *Quincy* will exclude unscrambled versions of the gesture in its randomized output (it will generate scrambling values of 1 or 2).

If `flux` is 0, *Quincy* will disregard `mode` and respect the scrambling values specified in `manualValues`.

The default value is 0.

Glossary

Amatria:

The sentient sculpture for whom *Autopoietic Bloom* was written. ***Amatria*** was developed and created by the Living Architecture Systems Group in Waterloo, Ontario, let by the architect Philip Beesley.

Ambison:

Ambison is Autopoietic Bloom's ambisonic spatializer. It simulates a virtual soundspace that is similar to what one would actually hear if one were standing directly in front of *Amatria*. In addition to the native 10-channel configuration, ***Ambison*** currently supports three ambisonic configurations: binaural (headphones), stereo (loudspeakers), and 8-channel surround.

bach:

A set of patches and externals for Max upon which the *Donatoni* improvisation engine is buttressed. **bach** (intentionally lower-case to distinguish it from its 18th-Century namesake) uses Lisp-like linked lists as abstract representations of musical hierarchies, representations which can easily be transformed in various ways before converting being converted into musical sound. See **bach** documentation for more information.

Donatoni:

Autopoietic Bloom's core improvisation engine that converts abstract representations of pitches into real-time improvised MIDI signals. ***Donatoni*** makes heavy use of the **bach** library for Max. It gets its namesake from the Italian Modernist composer Franco Donatoni, whose non-repeating musical gestures, known in Italian as *figure*, inspired the engine's iterative approach.

Improvisational Flux (Φ):

A float between 0.0 and 1.0 which corresponds to the frequency and amplitude of changes in a given improvised parameter. The higher the **improvisational flux**, the more often a parameter will change and when it does change, it will change with greater magnitude.

Improvisational Scheme:

A structured data file in the JSON format that allows *Autopoietic Bloom* to determine the large-scale musical and formal design for a particular improvisation.

Interpol:

Autopoietic Bloom's interpolation engine. ***Interpol*** is responsible for parsing information from the **improvisational schemata,** interpolating over that data, and then transmitting it to *Donatoni*, *Syrinx*, and *Quincy* in real-time.

Key-Value Pair:

The basic syntactical element of JSON structured data format is the **JSON object**, which contains zero, one, or more **key-value pairs**. The object is surrounded by curly braces { } and every key-value pair is separated by a comma (,).

A key-value pair consists of a **key** and a **value**, separated by a colon (:). The **key** is a string which identifies the key-value pair. The value can be any of the following data types: string, number, float, array, object, Boolean, empty.

Because a value can be an object or an array of objects, one can easily create nested hierarchical structures such as those used in *Autopoietic Bloom's* **improvisational schemata**.

Note that all strings must be surrounded by quotation marks (“”).

Lisp-like linked list:

Both a delightful tongue twister and a convenient way of structuring hierarchical data. Lisp-like linked lists are essentially nested lists created using square brackets [] or parentheses (). Because the lists can be nested, they provide an extremely intuitive way of representing musical information which is also extremely hierarchical. The **bach** library for Max makes heavy use of **Lisp-like linked lists**, and as such, they play a large roll in *Autopoietic Bloom's* real-time processing of musical data.

qConfig:

A **Lisp-like linked list** succinctly representing of how *Quincy* maps signals from *Syrinx* to its 10 output channels.

Quincy:

Autopoietic Bloom's multichannel dynamic mixer. ***Quincy*** manages the routing of four signals from *Syrinx* to ten channels of sound, transforming those mappings in real-time. ***Quincy's*** name is an homage to Quincy Jones, the legendary record producer who worked with artists such as Frank Sinatra and Michael Jackson, and who studied under Olivier Messiaen in his youth.

Syrinx:

Syrinx is *Autopoietic Bloom's* multichannel modular synthesizer. ***Syrinx*** is buttressed upon *Absynth*, the semi-modular synth plug-in developed by Native Instruments. Using a plug-in in this way allows ***Syrinx*** to load and transform presets, meaning an extremely large diversity of sounds can be produced and modulated with relatively few commands. ***Syrinx*** is named for the eponymous organ present in some birds that allows them to sing more than one pitch at the same time.

Ur-gesture:

The prototypical musical gesture that will be improvised upon by *Autopoietic Bloom*. This gesture is iteratively transformed in different ways by *Donatoni* to produce musical variation and development. It can be thought of as a parent gesture or a background gesture to what is actually heard.