

Manipulating tonals

You must set up your environment before these examples will work.

If you are using:

- **the *silbido* detector, read the ReadMeFirst file.**
- ***silbido*'s file format, read the SilbidoFileFmt file.**

Saving and loading sets of tonals

The dtTonalsLoad and dtTonalsSave functions can be used to load and save lists of tonal time x frequency contours.

tonal_set = dtTonalsLoad(filename) will load a set of tonals from the specified filename. An optional true/false flag controls whether or not a user interface dialog is presented. When true, filename may be the empty matrix [], or contain a name that will be used as the default value.

dtTonalsSave(filename, tonal_set) saves a set of tonals to the specified file. Like dtTonalsLoad, an optional true/false flag can be used to request a user interface dialog.

Using detected tonals

Sets of tonals are instances of Java collections. As such, one can use methods associated with the collection interface. Suppose we had a set of tonals called tonal_set. The following are examples of methods that could be used:

- tonal_set.size() – Returns the number of tonals in the set.
- tonal_set.get(n) – Return the n^{th} tonal. Java enumerates arrays and collections starting at 0, so n must be in the range $0 \leq n < \text{tonal_set.size}()$.
- tonal_set.add(t) – Add a tonal t to the set.
- tonal_set.iterator() – Returns a Java iterator, an object that can be used to loop over the tonal set:

```
% Assume that tonals contains a tonal set
% We will loop to find the minimum and maximum
% frequency
minfreq = Inf;
maxfreq = -Inf;
it = tonals.iterator(); % Create an iterator
while it.hasNext() % any more?
    ton = it.next(); % get next tonal
    f = ton.get_freq(); % get frequency list
    % update min/max frequencies
    minfreq = min(minfreq, min(f));
```

```

maxfreq = max(maxfreq, max(f));

% We could plot the tonal with:
%     plot(ton.get_time(), ton.get_freq());
end

```

Each tonal has a number of methods associated with it. A complete list can be seen in the source code for Java class `tonal` in the `tonals` package. Some of the more useful ones are:

- `get_time()` – Returns array of time offsets from the start of the detection file in s.
- `get_freq()` – Returns the frequencies associated with each time.
- `get_duration()` – Returns length of detection in s.
- `overlapping_tonals(tonal_set)` – Returns a new set containing tonals in `tonal_set` that overlap in time with this one.
- `toString(firstN, lastN)` - When tonals are displayed in Matlab, by default the first two time x frequency nodes and the last one are displayed. To see more of the tonal, the `toString` method can be used specifying how many nodes should be displayed at the head and tail of the list. Specifying -1 for the `firstN` argument will display all nodes.

Constructing tonals and tonal sets

When creating tonal objects, it is important to first tell Matlab that the `tonals` package will be used via the `import` command:

```
import tonals.*; % Import Java's tonals package
```

Once this has been done, tonals can be created by using the `tonal` constructor, providing a pair of vectors specifying times and frequencies:

```
new_tonal = tonal(time, frequency);
```

Be sure to avoid using the variable name `tonal`, or you will not be able to create new tonal objects until it is cleared.

Tonal sets can be created as follows, this example creates a set whose order is dependent upon the insertion order:

```

tonals = java.util.LinkedList(); % Empty linked list created
tonals.add(new_tonal); % Adds tonal to the list
another_tonal = tonal(time, frequency);
tonals.add(another_tonal); % Adds another tonal to the list

```