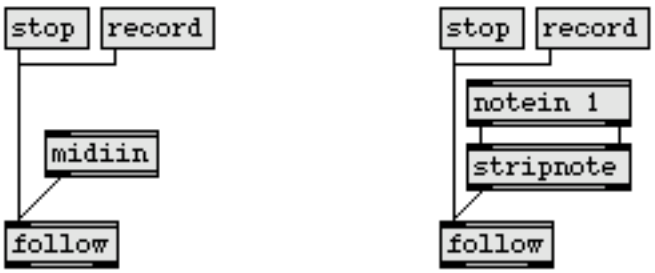
In general, whenever your patch is capable of stopping **seq** while notes are being recorded or played back, there is the potential for vital note-off messages to be lost. This is especially true if your patch sends stop, record, or play messages by some automatically generated means. Bear this potential danger in mind when constructing your patch, and include an object such as **flush**, **midiflush**, **poly**, or **makenote**—whichever is appropriate—to provide missing note-offs. Examples are shown in Tutorial 13.

- Record a sequence (or use the bourrée excerpt), and play the sequence with a start message. Try changing the transposition with the **hslider** while the sequence is playing.

follow

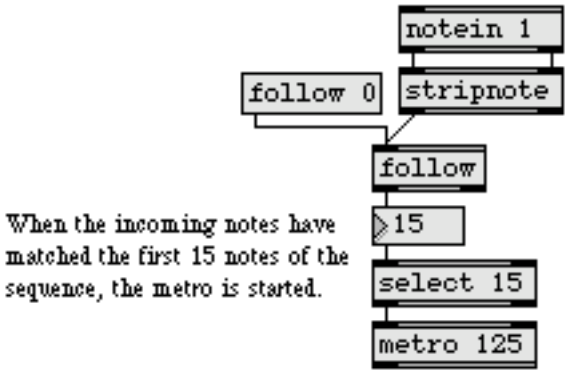
The **follow** object is very similar to **seq** in its ability to record MIDI data. But whereas **seq** only records MIDI messages, **follow** can also record a sequence of single numbers that are not in the form of complete MIDI messages (such as the pitches from MIDI note-ons).



follow can record MIDI messages, or single numbers (e.g., just note-on pitches)

A sequence can be stored in **follow** by recording MIDI data, by recording a series of single numbers, by reading in a file with a read message, or by typing in a file name argument. Once it has a stored sequence, **follow** can use that sequence as a musical score, and follow along while a performer plays the music. Each time the performer plays a pitch that matches the next note-on in the stored sequence, **follow** sends the pitch out its right outlet and sends the index number of that note's position in the sequence (1, 2, 3, etc.) out its left outlet.

The particular utility of this score-following feature is that the index numbers can be used to trigger other notes, or any other process such as, say, turning on a **metro** when the 15th note is matched.



How follow Follows

When **follow** receives the message follow with a number argument, it begins to look for incoming pitches which match the notes in the score, starting at the index specified in the argument. For example, follow 10 causes the object to look for incoming pitches that match the 10th note in the score. When the matching pitch is received, **follow** sends that pitch out its right outlet, and sends the index out its left outlet.

The **follow** object even allows for wrong notes, so if the performer plays a couple of spurious notes, or skips a note or two, **follow** will still be able to keep track of the performer's progress through the score.

One can also step through the score with repeated next messages. After a follow message has been received, the message next triggers the pitch at the specified index and increments the pointer to the next index.

An Attentive Accompanist

When we use the index numbers from the left outlet of **follow** as addresses of a **table**, or addresses of some other array object like **funbuff**, the index numbers can trigger other values. In this way, we can create an accompanist who "knows the score" and follows along with the performer. Each time the performer plays a note of the score, the accompanist has a specific reaction—play a simultaneous note or notes, play some independent melody, rest, whatever—and seems to follow along with the performer.

Tutorial 35

We've made such an accompanist in Patch 2. The accompanist plays the left hand part of the Bach E-minor bourrée while you play the right hand part. The **follow** object has loaded the sequence file *bourrée.sc* to use as the score. Each time a note of the score is played, an index number is sent out that triggers some sort of reaction.

- Click on the follow 0 message to start the score-follower at the beginning of the score. Play the right hand part of the bourrée excerpt and Patch 2 will play the left hand part along with you.

- If you've forgotten how the melody goes, read the *bourrée.sc* sequence into the **seq** object in Patch 1 and listen to it.

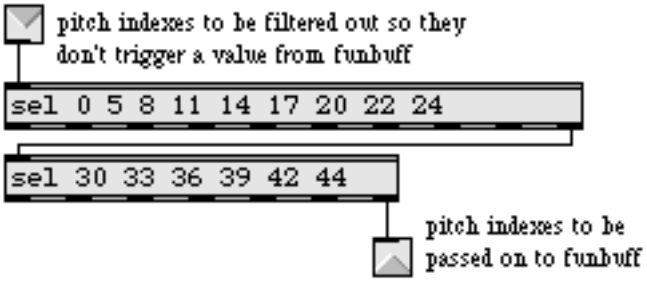
- Click on follow 0 again, and play the melody with an occasional wrong note or skipped note. If you don't mess up too much, **follow** manages to account for your mistakes and continues following the score.

- Try the melody again, with ritards at the end of the phrases. The extra notes that the accompanist plays match your tempo.

Analysis of Patch 2

Sometimes we want the left hand to play a note along with the right hand, other times we want the left hand to do nothing new (when the right hand is playing the second of a pair of eighth notes and the left hand is just holding a quarter note), and occasionally we want the left hand to play a note in between notes played by the right hand. How do we accomplish each reaction?

The index numbers are first sent to a subpatch called **patcher** silencer. This subpatch simply filters out the index numbers which we don't want to trigger a note of the accompaniment. The **sel** objects select those index numbers and pass the rest on.



Contents of the patcher silencer subpatch

Notice that **sel** objects can be linked together to select more than 10 numbers, since the numbers that are not matched by the first **sel** object are passed out the rightmost outlet to the second **sel** object.

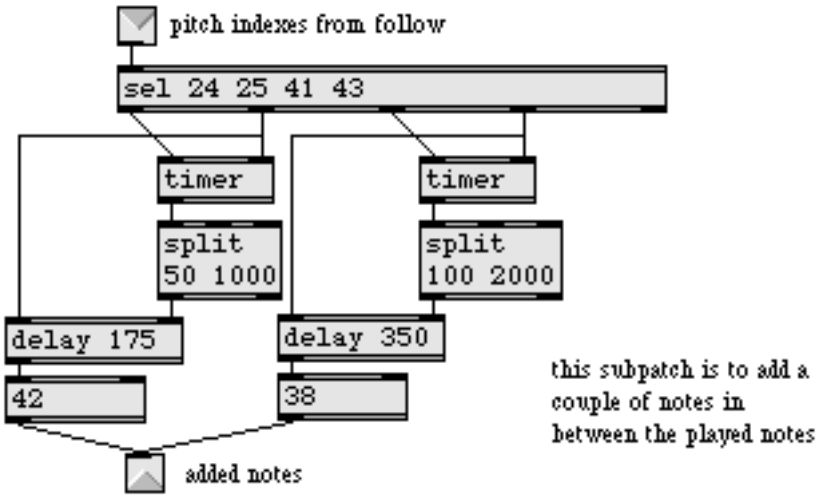
The remaining index numbers are sent as addresses to **funbuff**, which sends out an appropriate accompanying pitch value. To make **funbuff** respond properly, we simply made a list of *addresses* and *values* and saved the list as a **funbuff** file named *bourrée.fb*.

- If you want to see the contents of **funbuff**, choose **Open As Text…** from the File menu and open the file named *bourrée.fb*.

We could have also stored the accompaniment pitches in a **table**—or in a **coll** object, which will be explained in Tutorial 37.

So far we have made the accompanist play some notes that are simultaneous with the melody notes, and we've made the accompanist rest on melody notes that are unaccompanied, but how about when the accompanist has to play notes on its own, in between melody notes? This occurs twice in the score, once at the end of each phrase.

To help the accompanist play notes on its own, the **patcher** addnotes object measures the tempo of the performance and plays notes with a delay time based on its perception of the performer's tempo.



Contents of the patcher addnotes subpatch

For example, the subpatch measures the amount of time between notes 24 and 25 of the melody (the speed of an eighth note), then delays for that amount of time before triggering the pitch 42. Likewise, the time between the 41st and 43rd melody notes (the

speed of a quarter note) is used as a delay time before sending out the pitch 38. This is a simple (but fairly effective) method of analyzing the performer's tempo and playing notes in that tempo.

It's always a good idea in programming (and elsewhere, for that matter) to prepare for the unexpected. What happens if the performer accidentally misses one of these notes that we need for analyzing the tempo and triggering added accompaniment notes?

If the performer misses the first note of a pair, for example, the second note will trigger a ridiculously large value from the **timer** and the accompaniment note will get delayed far too long. To protect against this eventuality, we have used **split** objects to limit the time values that can be sent to **delay** within certain (only *moderately* ridiculous) extremes. If the value from **timer** exceeds these limits, the **delay** object will use the delay time in its argument. If the performer misses the second note of a pair but continues on, the added note will never get played, but by then the performer will have passed that point anyway, and **follow** will keep up with the performer.

The pitches from **patcher** addnotes and from **funbuff** are sent to **makenote** where they are paired with the velocity of the right hand melody notes, so the accompanist is sensitive to the performer's dynamics, as well. Rather than use an algorithm or a lookup table to provide durations for the accompaniment notes, we just picked a duration that seems to work both as an eighth note duration and as a stylistically staccato quarter note.

Summary

A single track of raw MIDI data can be recorded and played back (at any speed) with the **seq** object. The MIDI data is received from **midiin** and is transmitted by **midiout**. You can also parse the data from **seq** using **midiparse**, and process the numbers with other Max objects before transmitting them.

A recorded sequence can be saved as a separate file by sending a write message to **seq**. If you check the *Save as Text* option in the Save As dialog box, you can open and edit the file later with **Open As Text…**. A MIDI file can be read into **seq** by sending a read message, or by typing in the file name as an argument.

The **follow** object allows you to record or read in a sequence, then use that sequence as a musical score to follow along with a live performance. As the pitches received in the inlet are matched with notes in the score, the index number for each note is sent out, and can be used to trigger other notes or processes.