How To Use Online Iteration (OLI)

BKT 14-Apr-01

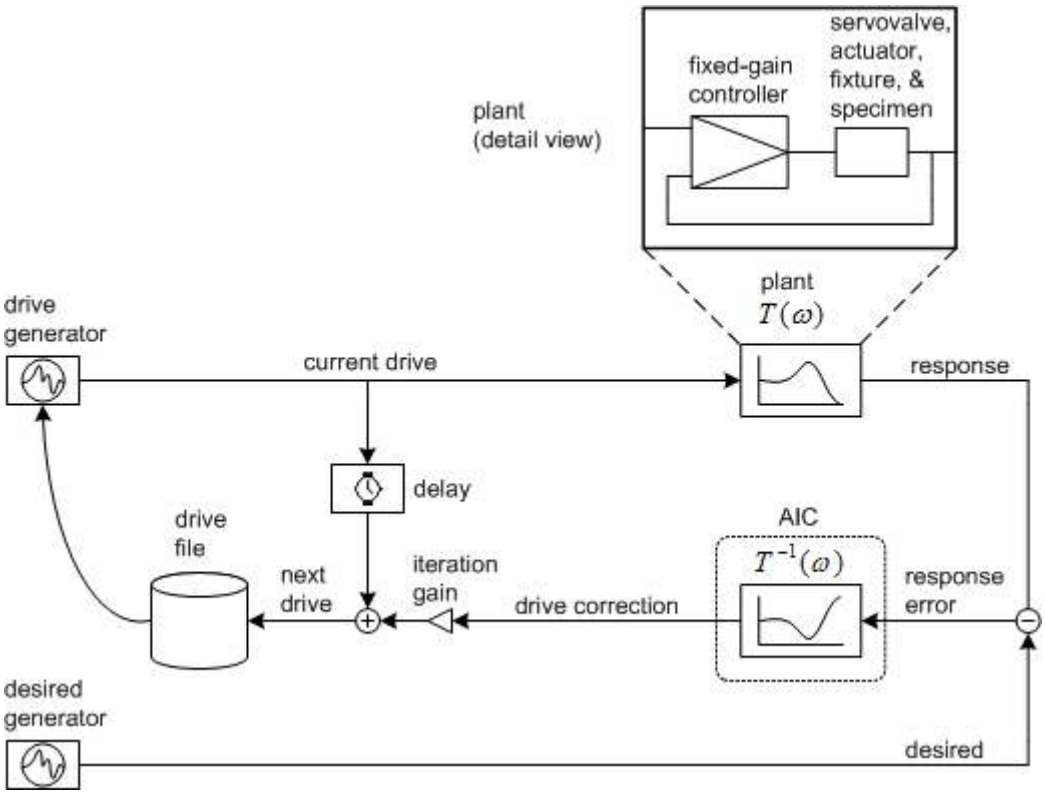
Online Iteration (OLI) is a control technique that repeatedly modifies the command input to a control system on an individual sample-by-sample basis until the control system response is almost a perfect replica of the original desired command.

OLI is optimized to work with non-sinusoidal command waveforms and significantly nonlinear systems. If the command waveform is a pure sine wave, Adaptive Harmonic Cancellation (AHC) works better and is easier to use. If the system does not have significant nonlinearities, use Adaptive Inverse Control (AIC) instead.

How OLI works

The focus of OLI is two time history files, the "desired" file and the "drive" file. The desired file is an unchanging file that contains the time history that you want to reproduce on the test system. The drive file is a time history file that is modified every time it is played out in such a way as to cause the system response to eventually match the desired time history. The process of repeatedly playing out the desired and drive files while updating the drive file is called "iteration".

OLI is shown schematically in the diagram below:

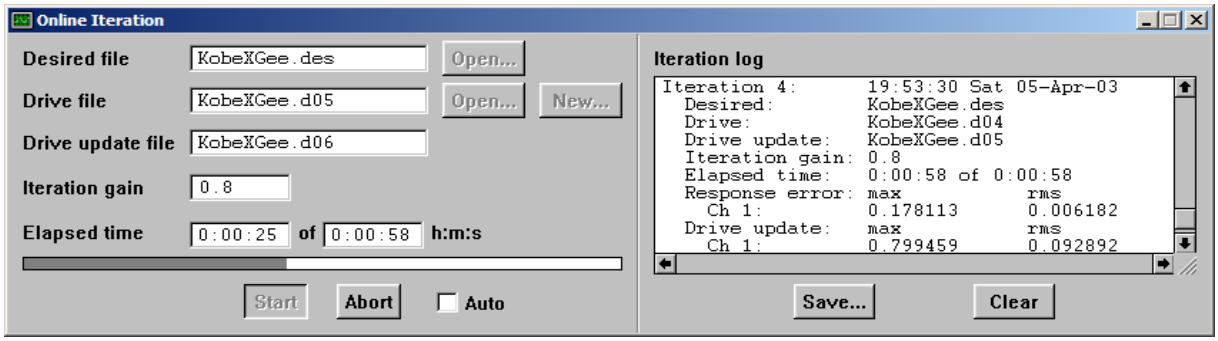


In the diagram above, the desired time history file and the drive time history file are played out simultaneously through their respective waveform generators. The drive is played directly into the *plant*, which is the term used hereafter to refer to the dynamics of the fixed-gain controller, actuator, fixture, and test specimen combination. Simultaneously, the desired is played out and compared to the plant response to develop a response error. A drive correction is computed by running the response error through a filter that approximates the inverse transfer function of the plant. This drive correction represents a best linear estimate of the amount of drive error that gave rise to the response error. A fraction of the drive correction specified by the "iteration gain" is added to the current drive waveform (delayed to account for plant and processing delays) to form the drive for the next iteration cycle and the result is stored in a disk file. The iteration gain, whose value ranges from zero to one, determines the rate at which the response converges to the desired and is chosen according to the amount of nonlinearity in the plant. For a very nonlinear plant, the iteration gain must be small and the number of iterations large for the iteration process to converge. For a predominantly linear plant, an iteration gain of nearly one is possible without divergence, in which case only two or three iterations are required.

Note in the diagram that OLI requires knowledge of the inverse transfer function of the plant in order to compute a drive correction. This inverse transfer function is provided by the Adaptive Inverse Controller (AIC), appropriately rewired to serve in this role, and is determined by a training process prior to running OLI; refer to the document "How To Use AIC" for details.

We now turn to specifics on using OLI. After a description of OLI's user interface, a step-by-step operating procedure will be presented.

OLI Panel Adjustments

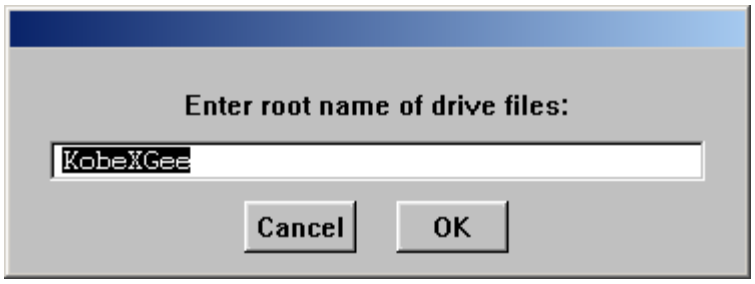


Desired File Open button

Select a desired time history file. You will be prompted with a file open dialog box.

Drive File New button

Create a new drive file sequence. You will be prompted to enter the root name of the drive file sequence with the dialog box shown below:



Drive File Open button

Open an existing drive file. You will be prompted with a file open dialog box.

Iteration gain edit box

Set the rate at which the iterations converge to a final solution. Acceptable values range from zero to one.

Elapsed time text displays and progress bar

Displays the elapsed time and the total length of an iteration textually. Elapsed time is also displayed graphically in the progress bar.

Start button

Start an iteration.

Abort button

Terminate the current iteration prematurely.

Auto checkbox 

Specify automatic iteration mode. When checked, after completion of an iteration, the next iteration is begun immediately without having to press the Start button. If unchecked during an iteration, motion will stop at the end of the current iteration. Do not use this feature unless you have run a few iterations manually and are confident that the iteration process is progressing smoothly.

Iteration Log text display

Displays the results of each iteration so that iteration progress can be monitored. A typical iteration log entry looks like this:

```
Iteration 4:       19:53:30 Sat 05-Apr-03
  Desired:         KobeXGee.des
  Drive:           KobeXGee.d04
  Drive update:    KobeXGee.d05
  Iteration gain:  0.8
  Elapsed time:    0:00:58 of 0:00:58
  Response error:  max            rms
    Ch 1:          0.178113       0.006182
  Drive update:    max            rms
    Ch 1:          0.799459       0.092892
```

Save button

Save the text in the Iteration Log to a text file.

Clear button

Clear the text in the Iteration Log.

OLI Operating Modes

OLI has three operating modes available from a popup menu on the Main Panel:

These modes are:

Disabled: OLI is off.

Frozen: OLI is on, but advancing to the next drive file at the end of the current iteration is inhibited. Use this mode if you want to OLI to monitor error statistics when playing out a final drive file.

Tracking: OLI is on, and advancing to the next drive file at the end of the current iteration is enabled. Use this mode to develop a final drive file.

When you turn on OLI to Frozen or Tracking, AIC is automatically turned on in Frozen mode for you. When you turn off OLI, AIC is automatically turned off.

You can change AIC to Tracking mode if you want AIC's inverse transfer function to update during iteration. In certain strongly nonlinear systems where the inverse transfer function varies significantly with signal level, this feature may help the iteration process to converge. However, because most systems converge nicely without this feature, and because the potential for AIC divergence in Tracking mode, we recommend that you leave AIC in Frozen mode.

Operation

Step 1: *Train the Adaptive Inverse Controller (AIC).* OLI uses AIC's inverse transfer functions, which must be identified during a training process. Consult the section on training tranfer functions in the document "How To Use AIC".

Step 2: *Prepare the desired time history file.* MTS's time history files have a proprietary binary format for which MTS provides two file translators. One translator translates a text file that you create using a text editor or a spreadsheet program. The text file must have the sample period on the first line, followed by a column for each data channel. For example:

```
0.00390625
0.0         0.0         0.0
-0.00021    0.00012     0.000867
-0.00033    0.000209    0.000957
-0.00036    0.000359    0.000987
-0.00042    0.000389    0.000389
-0.00036    0.000359    0.00012
(etc.)
```

After creating this text file, invoke the translator from the Main Panel File, Translate, Text to Binary menu item. You will be prompted for the name of the text file and a name for the binary file. Use the same name as the text file for the binary file but with a different extension. You can use any file extension you wish for desired time history files, but we recommend that you use ".des" for convention.

Another translator is provided for use within MathWorks' Matlab computation environment and translates Matlab arrays directly into binary time history files. The translator is a Matlab script "savebin.m" that you can run from Matlab's command window; type "help savebin" after the command prompt to see how to use it.

When preparing a desired file, there several restrictions that apply:

- The data in the file must have a sample period of 1/256 second.
- The number of channels in the file must be equal to the number of control channels.
- The time histories must begin and end at zero. For best results, you should also append an additional 1-2 seconds of zero samples at both ends.
- The signal content in the time histories must not exceed 64 Hz.

Step 3: *Turn on OLI in Tracking mode from the Main Panel.* AIC will be automatically turned on in Frozen mode if you have not already done so.

Step 4: *Enter the desired master span and hit the Run button on the Main Panel*. You can iterate at 100% amplitude, or you can iterate at a reduced level such as 10-20% to minimize damage to your specimen, then after completing the iteration process, run the test at 100% span. Mathematically, linear scaling is not a valid operation in nonlinear systems, but nonetheless this technique often works quite well.

Step 5: *Select the desired time history file.* Using the Desired File Open button, select the desired time history file.

Step 6: *Create the first drive time history file.* Using the Drive File New button, begin a new series of drive files by entering the root name of the drive file series and press OK. We recommend that you use the default root name, which is the same as the root name of the desired file. For example, if the desired file name is "myfile.des", the default root name is "myfile", and the sequence of drive files created during iteration will be "myfile.d00", "myfile.d01", "myfile.d02", etc. The first drive file in the series "myfile.d00" will be filled with zeros.

Alternatively, you can create your own initial drive file that contains a scaled down version of the desired rather than just zeros. This may reduce the number of iterations. In this case, use the Drive File Open button instead of the Drive File New button.

Step 7: *Enter an iteration gain.* The iteration gain determines how many iterations it takes to converge to the final solution. The ideal case is an iteration gain of 1.0, in which case convergence occurs in a single iteration. However, an iteration gain this high only works with a perfectly linear system; in a nonlinear system iterations would diverge. In general, the more nonlinear a system is, the lower the iteration gain must be for convergence, and therefore the greater the number of iterations that must be performed.

Only by experience will you know what iteration gain is appropriate for your testing situation. If you don't know what iteration gain to use, start with 0.5. Watch the trend of the response errors as iterations progress. If response errors monotonically decrease, it is probably okay to increase the iteration gain. If response errors oscillate up and down from iteration to iteration, you should decrease the iteration gain.

If you find that you are using an iteration gain of nearly one, your system is probably linear enough that you don't have to use OLI. If this is the case, just use AIC.

If you use too high of an iteration gain and cause your iterations to diverge, decrease the iteration gain and go back to a previous drive file using the Drive File Open button.

Lower iteration gain and tighter response error tolerance both increase the number of iterations that you need to perform. A formula for the number of iterations as a function of iteration gain to achieve a given percentage error is:

$$\#iterations = \frac{\ln(percent\_error/100)}{\ln(1 - iteration\_gain)}$$

Although this formula is valid only for perfectly linear systems, it is a useful approximation for nonlinear systems as well.

Step 8: *Start the iteration.* Hit the Start button on the OLI Panel. If this is Iteration 0 and you had OLI create the initial drive file for you, there will be no motion because the drive file is zero; otherwise motion will begin. We recommend that you watch the desired and response in the Digital Oscilloscope during the iteration to verify that motion is at acceptable levels. If you don't like what is happening, hit the Abort button on the OLI Panel to immediately terminate the iteration.

Step 9: *Examine the results of the iteration.* After each iteration, take time to look at the results of the iteration displayed in the Iteration Log. Verify that errors are decreasing. Iteration is complete when the errors have not significantly decreased for several iterations. Look carefully at max and RMS drive level figures; these tell you how large the motion will be on the next iteration. If these numbers seem unreasonably large, do not proceed to the next iteration or you may destroy your specimen and damage your machine.

If the errors are increasing from iteration to iteration, the iteration process is diverging. Divergence problems are almost always due to one of two reasons:

- The iteration gain is too high for the level of nonlinearity in the system. Lower the iteration gain, and go back a few iterations in the drive file sequence, especially if the amplitude of the response has built up to a level that exceeds the desired, because OLI has a much easier time scaling up to the desired level rather than scaling down to it. Use the Drive File Open button to call in a previous drive file.
- The inverse transfer function has been inadequately or improperly trained. Go back to AIC and retrain or refine its inverse transfer function.

If the errors are large but neither increasing nor decreasing on average, you may be at a physical velocity or force limit of your test system. There is nothing you can do in this case except accept the errors as they are, or decrease the amplitude of your desired waveform and start over.

Step 10: *Turn off OLI (when finished with the final iteration).* On the Main Panel, first hit the Stop button to zero the master span. Then disable OLI. AIC will be automatically disabled as well.

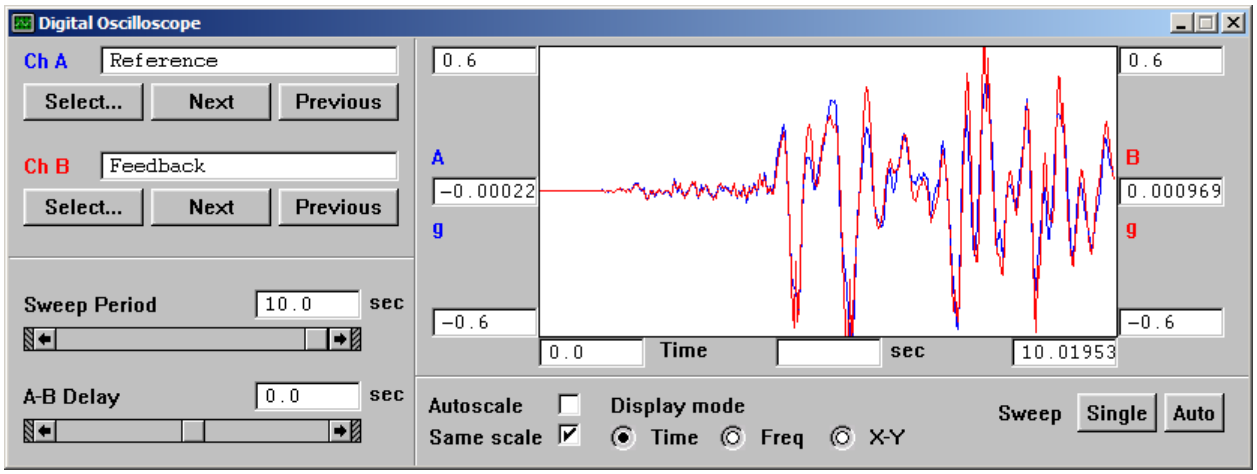
The end product of the iteration process is the final drive file; all other drive files can be discarded. You don't have to use the OLI Panel to play this file; the File Function Generator or Data Player can be used instead. However, if you want to monitor peak and RMS errors while playing the file, you can use the OLI Panel for file playout. In this case, turn on OLI in Frozen

mode so that OLI does not advance to the next drive file in the iteration sequence but rather continues to play the selected drive file.

Viewing the Results

You can use the Digital Oscilloscope to monitor desired and response during the iteration process. You can also capture the response to file using the Data Recorder for post-test error analysis.

Below is an example showing desired and response time histories without OLI:



After several iterations of OLI, desired and response are almost perfectly overlaid: