# Artefact Removal Program (ARP) functionality

Contents

[Artefact Removal Program (ARP) functionality 1](#_Toc390254874)

[BEFORE YOU START YOUR STUDY – READ ME FIRST 2](#_Toc390254875)

[Artefact Removal System Program (ARP) Functionality 2](#_Toc390254876)

[Setting up your Matlab environment for ARP 4](#_Toc390254877)

[Setting up your data directories 4](#_Toc390254878)

[Defining Your Parameters 5](#_Toc390254879)

[Channel Definition Files 5](#_Toc390254880)

[Making A Custom .mat Channel File (this section may no longer be needed) 6](#_Toc390254881)

[Creating a Generic Eye Artifacts file 7](#_Toc390254882)

[Re-referencing Neuroscan Data prior to running ARP 8](#_Toc390254883)

[NICER LAB EEGLAB implementation INSTALL PROCEDURES 9](#_Toc390254884)

[Batch Processing – Running ARP 11](#_Toc390254885)

[batch.m – the master controller 11](#_Toc390254886)

[batch1.m – statistics and initial “quick” ICA 11](#_Toc390254887)

[batch2.m – epoch based on paired events (if relevant), automatically reject bad trials, and merge 12](#_Toc390254888)

[batch3.m – run the full ICA 13](#_Toc390254889)

[batch4.m - ICA selection and removal 13](#_Toc390254890)

[batch5.m – Artefact REMOVAL and merging of sets 17](#_Toc390254891)

[ICA Analysis techniques 19](#_Toc390254892)

[AVERAGING Your Data (can only be run after completing batch5.m) 20](#_Toc390254893)

[TBD 21](#_Toc390254894)

[Resolving "Out of Memory" Errors in Matlab 22](#_Toc390254895)

[General Suggestions for Reclaiming Memory 22](#_Toc390254896)

[Setting the Process Limit 22](#_Toc390254897)

[Disabling Java VM on Startup 23](#_Toc390254898)

[Increasing System Swap Space 23](#_Toc390254899)

[Using the 3GB Switch on Windows Systems 24](#_Toc390254900)

[Freeing Up System Resources on Windows Systems 24](#_Toc390254901)

# BEFORE YOU START YOUR STUDY – READ ME FIRST

There are some VERY important first steps you must undertake, BEFORE you start recording participants, to simplify your life for processing data afterwards.

1. **Ensure ALL FILE NAMES have NO SPACES when you are recording**. You can use underscores, but do NOT put spaces in the file names. This is especially important for Brain Vision files, as simply changing the filename after the fact is not sufficient. The data and marker file names are embedded in the .vhdr and .vmrk files, so if you do not record without spaces, you will have to manually edit EVERY .vhdr and .vmrk file to fix this, or the ARP system will throw errors.
2. **Ensure your BASELINE parameters are DOCUMENTED and known (how many seconds before and after stimulus are to be part of an epoch)**, because these will be crucial to properly epoching data and running the ICA analysis.
3. **For BRAIN VISION,** **ensure that all channels are recorded in the format**:

Ch1=FP1,, 1.0,µV

(most important is that **Ch1=*Name***, where “***Name***” is a STANDARD BESA format channel name. Do NOT deviate from this format)

1. **Save all images produced and/or take screen shots of them - especially the COMPONENT plots and the COMPONENT HEADMAPS**

# Artefact Removal System Program (ARP) Functionality

ARP is used to preprocess your EEG data in preparation for averaging. It uses Independent Component Analysis (ICA) to do the cleaning (muscle contraction, movement, and noise) and Principle component Analysis (PCA) for Ocular Correction – (removing eye blinks and saccades)

ARP runs in 6 basic program steps – these can be run all at once or separately, if needed, provided a prior step has been completed at some point earlier.

1. Batch\_Initialize.m – the parameters for your project/study
2. Batch.m – sets basic parameters
3. Batch1.m – initial preprocessing
4. Batch2.m - epoch based on paired events (if relevant), automatically reject bad trials, and merge
5. Batch3.m – takes a very long time to run, even using BINICA – finds potential components for removal
6. Batch4.m – displays Headmaps from ICA for you to review and select for removal – do NOT INTERRUPT –must run to completion

* You see about half the number of channels in components , ie 35 components for 70 channels
* One plot is components, the other original EEG data

1. Batch5.m

Depending on the EEG system used, there are variations in the way you must set up your data and parameter definitions (See Defining Your Parameters).

**Files produced by the EEG systems in the NICER LAB**

The two primary systems used in the lab are the Brain Vision (BV) and Neuroscan (NS).

**Neuroscan (NS) Files:**

* NS collects **69 Channels** of data (the 70th is the ground). Reference is usually Channel 69.
* Header (channel definitions), EEG data, and marker data (triggers and responses) are all included in **\*.cnt** – continuous EEG files.
* NS data must be preprocessed by the Neuroscan software, *on the data collection system*, to put reference back in before Artefact Removal processing can take place. (See: *Preprocessing Neuroscan Data*).

**BrainVision (BV) Files:**

* BV collects **32 channels** of data. If data is recorded with “Average Reference” then no re-referencing can be done and the reference cannot be added back in. If at all possible DO NOT record with the “AVERAGE REFERENCE” setting in BrainVision. – i.e. do not select a reference as Brain Vision will record as average referenced.
* Header data (channel definitions) are included in **\*.vhdr** files.

Header data MAY need to be edited if the channel names are not in the correct format (earlier stimulus response programs put the wrong channel names in and other programs have trouble recognizing the names if standard electrode names are not used).

* EEG data is in **\*.eeg** files.
* Marker data (triggers and responses) are in **\*.vmrk.** The marker data file contains the triggers and recording information of the EEG file (e.g., the timing for each trigger presented).
* **HEOG data is collected in channel 32, but has amplitude problems – for now do not use it as we are unsure of the amplification level and Brain Vision’s information seems to be inaccurate.**

# Setting up your Matlab environment for ARP

You must setup your environment and run your data in the following steps:

1. Optional: Install Matlab and then EEGLab (optional if Matlab/EEGlab is already installed – check with the lab manager), and make the appropriate modifications to the paths. (See *NICER LAB EEGLAB implementation INSTALL PROCEDURES*).
2. Set up your data directories in the prescribed format (see below).
3. Copy the template parameter file (Batch\_InitializeNS.m or Batch\_InitializeBV32.m or Batch\_InitializeBV31.m) for the system you are using to a file name specific to your study. Edit and modify the parameter file to suit your study parameters, and then save to Batch\_Initialize.m at the root of the directory where the ARP code is installed.
4. Launch Matlab.
5. Add EEGLab to the MATLAB path:

Under the HOME tab In Matlab, browse to the folder where you have EEGLab installed, and right mouse-click on the folder icon. Choose “add to path with subfolders” and add EEGLab to your path. This varies with the version of Matlab. Alternatively, you can select File/Set Path/, browse to your folder and select “Add with subfolders” in the drop down selection window. If you do not “Add with subfolders”, EEGlab may not be able to access other programs in its own directory (e.g., functions, etc.).

1. Add your data directories to the Matlab path in the same way.
2. Add the ARP program files to the Matlab path in the same way.
3. Run Batch.m.

NOTE: This procedure may change once the Linux server has been set up.

## Setting up your data directories

To use the NICER Lab artefact removal system program, you must ensure your data is in the correct directory structure:

1. Use a consistent naming convention for each Participant, Session, Block, and Task
   1. A task is a functional unit of testing such as Stroop colour (SC), stroop word (SW), go-nogo (GNG).
   2. A block is a series of one or more tasks (i.e. you might run a combination of tasks or just one).
   3. A session contains one or more blocks (i.e. you may run the participant through the same block 2 or 3 times, or you may have different blocks which are run consecutively).
   4. In general, it is good practice to ensure that individual task data is written to separate files to make averaging easier.

i.e. Participant is C120, two blocks are run, and each block contains a Stroop Colour and a Stroop Word task, so the resulting files would be:

* C120\_SC1.cnt for Stroop Colour, Block 1
* C120\_SW1.cnt for Stroop Word, Block 1
* C120\_SC2.cnt for Stroop Colour Block 2
* C120\_SW2.cnt for Stroop Word, Block 2
* NOTE: It is a good idea to use underscores between each variable (e.g., C120SC1.cnt could be C120\_SC1.cnt). It is also a good idea to begin your filename with a letter rather than a number.

1. Create a main folder that contains all your data (i.e. C:\StudyData), and within it, a separate folder for EACH PARTICIPANT where the folder name is the base participant filename, i.e. files C120\_gng1.\*, C120\_gng2.\*, C120\_RESTING.\* and C120\_PR.\* must all be in a folder named “C120”. Remember: **NO SPACES in the directory or file names – use underscores instead.**
2. The program will create a “Results” directory under your main data, which will have subfolders by participant number, containing the results files.

NOTE: If you have MANY trials per participant, you may have to break the data up further, as there are memory limitations in Matlab. See “**Resolving “Out of Memory” Errors in Matlab**”.

## Defining Your Parameters

ARP requires certain input parameters to be defined, which are specific to a particular study. Two sample template parameter files are available to review, **Batch\_Initialize\_NS.m**, and **Batch\_Initialize\_BV.m**, for a Neuroscan and BrainVision study respectively.

Each template sets a series of parameters in name and value pairs. Within the template are comments that describe each of the parameters and restrictions on setting the parameters. Copy the appropriate template to **Batch\_Initialize.m** in your working directory, and edit to ensure parameters are set according to your study requirements.

## Channel Definition Files

Designate the correct channel definition file:

**For Neuroscan**, after re-inputting the reference, use: **69Ch\_NS.mat.**

* 70 channel.elp includes ground and reference and the reference is used to make the average reference.

**For BrainVision**, use: **32Ch\_BV.mat** if you recorded “average referenced” or **33Ch\_BV.mat** if you recorded “reference free” (It is highly recommended to record in this mode because it allows you the freedom to choose your reference later. Recording with a selected reference means that the EEG already has the reference subtracted from each channel and, if there is an issue with the reference, it cannot be corrected. You lose that data because there is no way to remove the average reference and select a different one).

Check the .ELP file version of the above to verify that the channel order listed in the .elp file is the SAME ORDER THAT YOU RECORDED IN. If not, you must MAKE A CUSTOM .mat CHANNEL FILE.

Biosemi\*.mat: Binary Matlab file of electrode locations .

69Ch\_NS.mat: Binary Matlab file of electrode locations, including reference.

68Ch\_NS.mat: Binary Matlab file of electrode locations, NO reference.

32Ch\_BV.mat: Binary Matlab file of electrode locations, NO reference.

33Ch\_BV.mat: Binary Matlab file of electrode locations, including reference.

## Making A Custom .mat Channel File (this section no longer needed)

**This section is no longer needed, but the documentation is preserved in case someone needs to do this in future.**

The ARP system uses a binary version of the channel locations, in Matlab **.mat** format. There are already **.mat** files for Neuroscan 69channel and 70channel, as well as Brain Vision 32channel with average reference. If your study uses a different setup, you will need to create your own binary channel location file using the following steps:

1. Make a COPY of the .elp file that is nearest to the configuration you used.
2. In NOTEPAD, edit the .elp file to place the channels in the recorded order.
3. Save the .elp file under a new file name:  **ie. *custom*.elp.**
4. Launch EEGLAB from the Matlab interface.
5. File 🡪IMPORT Data - load a SINGLE data file from your study.
6. Edit 🡪 channel locations.
7. Locate the “read locations” button within the dialog. Select this and browse to the location of your ***custom*.elp** file.
8. Once loaded, verify all the channels and then choose the option to “SAVE .CED”.
9. Run “**make\_chan\_mat.m**” and input the body of the filename for your ***custom*.ced** file (don’t include the .ced extension!)
10. The program will produce a ***custom*.mat** file that you must then reference in your Batch\_Initialize.m file.

**\*.elp:** text file: channel definition files in BESA format.

**\*.ced:** text file: Produced by EEGLAB as a result of converting an .elp file - EEGLAB converts incoming channel definition files to a standardized .ced format.

# Creating a Generic Eye Artifacts file

The generic eye artifacts file is used for automated processing and removal of eye artifacts. This file contains a single vector of floating point numbers (one component per channel), **mean\_blink\_winv**. The data represents topoplot of a typical blink component, obtained by averaging blink components across several participants. A 69 channel system should have 69 components (68 recording channels plus the REF channel – in the Neuroscan system, the REF is a separate channel located directly behind Cz; Cz is a recording channel) and a 32 channel system should have 32 components, except that in the case of BrainVision we are removing HEOG, so typically, BrainVision data will only have 31 components.

Topoplot command: Plots a topographic map of a scalp data field in a 2-D circular view (looking down at the top of the head) using interpolation on a fine cartesian grid. Can also show specified channel location(s), or return an interpolated value at an arbitrary scalp location (see 'noplot'). By default, channel locations below head center (arc\_length 0.5) are shown in a 'skirt' outside the cartoon head (see 'plotrad' and 'headrad' options below). Nose is at top of plot; left is left; right is right. Using option 'plotgrid', the plot may be one or more rectangular grids.

The best way to make this is:

1. In EEGLAB, process 5 or 6 participants. For each participant:
   1. **File->Import-> ….** - Load a clean run.
   2. **Edit->Channel Locations** - Make sure channels are properly set.
   3. **Tools-> Extract Epochs** - Epoch the data.
   4. **Tools-> Run ICA** - Choose binica – run a quick ICA.
   5. Determine which component best represents the eye blink component (usually component 1 or 2), and write it down:
      1. **Plot->Component Activations (scroll).**
      2. **Tools-> Reject Data using ICA -> Reject components by map.**
   6. **File -> Save current dataset as…** - Save the SET FILE for each participant, with the body of the file name as “S#” – ie. S1, S2, S3, etc… (this saves both a .set and a .fdt file for each dataset)
2. Edit **makegen.m** code to store the blink component numbers in the **bcomps** variable. For example, if blink components are 1, 1, 2, 1, 1 for sets S1, S2, S3, S4 and S5 respectively,

**bcomps = [ 1 1 2 1 1];**

1. Edit the file paths in **makegen.m** to point to the directory containing the **.set** files. Save and run **makegen.m.**
2. Rename and move the resulting **.mat** file to your working code directory.

# Re-referencing Neuroscan Data prior to running ARP

1. On the Neuroscan computer, copy the LED\_REF.TEMPLATE file to LED\_REF.TCL in YOUR OWN DIRECTORY.
2. Run Neuroscan by clicking on the desktop icon ().
3. Click on the “Edit” button ().
4. Load the LED\_REF.TCL file from your directory by clicking on the “load file” button ().
5. Click on the EDIT TCL button to edit ().
6. There are a series of three commands per file you want to process. You must replace each of the file names/paths with the path and file names of your data files.
   * The first line contains the path/name of your raw data file.
   * The second line should contains a **MODIFIED name** that will be your output file (**NEVER OVERWRITE YOUR RAW DATA**) – this adds the reference channel at the end of the input file and outputs it to the new name specified.
   * The last line closes your input file.

**OPENFILE {C:\Patricia\LED\SN2508\SN2508S2.cnt}**

**REFER N Y N { ALL } {C:\Patricia\LED\SN2508\XSN2508S2.cnt}**

**CLOSEFILE "SN2508S2.cnt"**

**NOTE: This can also be done using the Scan/Edit program but takes much longer if you have multiple files to process.**

1. Save your changes.
2. Run the TCL file by clicking on the “RUN” button.
3. Copy your processed files to a USB stick and upload them to the computer you will be using to run ARP.

Note: If you need to rename the "REF" to "CZ", insert the following two lines before **CLOSEFILE** for every file specified:

**RENAME “REF” “CZ”**

**UPDATE CHANGES**

# NICER LAB EEGLAB implementation INSTALL PROCEDURES

1. INSTALL MATLAB
2. Download and install (at your C:\ drive root) the EEG\_Lab implementation from <http://sccn.ucsd.edu/eeglab/install.html>

You should have a directory something like: **C:\eeglab13\_1\_1b**

1. Unzip the artifactremoval.zip file and place the artifact\_removal\_code in a directory path that has NO SPACES IN IT – this is vital. **All directory names MUST have NO SPACES, use underscores or capitals between words instead**
2. The BINICA implementation is a “c” program that encapsulates the ICA functionality. When compiled into an executable it speeds up the ICA processing by 6-10x times. The distribution comes with a LINUX implementation. To run EEGLab on Windows, you must modify CORE EEGLab definition file to have location of ICA.EXE:

LAUNCH MATLAB.

Open and edit: **C:\{your eeglab directory}\functions\sigprocfunc\icadefs.m**

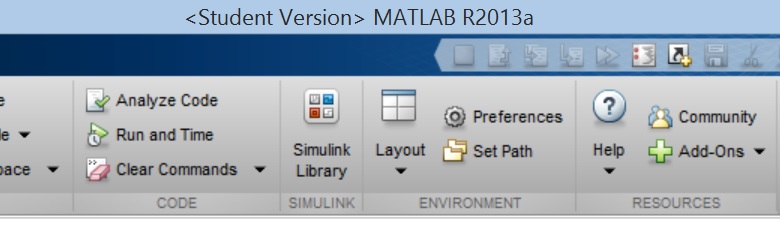
Comment out the original ica\_linux line and replace it with the location of your ICA.EXE file, ie.:

**%ICABINARY = fullfile(eeglab\_p, 'functions', 'resources', 'ica\_linux');**

**ICABINARY = 'C:\Artifact\_removal\_code\BinICA\ICA.EXE';**

1. Tell Matlab about the paths to your program and data files.

Click on the “Set Path” icon in the Matlab “Home” tab (this differs for different versions of Matlab but is usually located under the File tab or on the Home tab in newer versions).



Click on the “Add with Subfolders” button and navigate to your artifact\_removal\_code directory. Select it and click SAVE on the main Paths dialog.

Click on the “Add with Subfolders” button and navigate to your data files directory. Select the top level directory for the tree, and SAVE it in the path.

(You may need to do this add path step every time you start Matlab. This is usually because your permissions will not allow Matlab to write to the Programs folder and save the new path there. You can try saving the path in your data directory if you find this is happening.)

1. Test your implementation by running sample files through it. You should see no errors.

(sample test instructions?)

Once you have installed MATLAB, it will create a startup folder where you can put startup files. You can find the default startup folder for your install by typing:

>>userpath

This should tell you the location of your startup folder. If you don’t want to set paths manually every time, you can create a “startup.m” file that contains the paths in this folder using the following command for every directory that must be seen by the ARP implementation”.

Create a new “script” in file: New->Script.

Add lines for every path you need to set. Remember that you must also specify all sub-directories required. i.e.:

**addpath C:\NICERLAB\LNF;**

**addpath C:\NICERLAB\LNF\C120;**

Select “save as” startup.m in your startup folder.

# Batch Processing – Running ARP

## batch.m – the master controller

The batch.m file contains all the key instructions for running batches 1 through 5 and attempts to run them all in sequence.

After any given batch is complete, the Matlab console will display a message indicating the batch is complete. The console also displays lots of informational messages from EEGLab functions that are running. Each batch\*.m file saves an output file that is used by the next batch in the sequence. If you encounter a problem in a particular batch, you can re-run starting only at that batch by editing batch.m and commenting out the preceding batches. You can also launch EEGLab and analyze the .set files if you need to do further analysis with EEGlab.

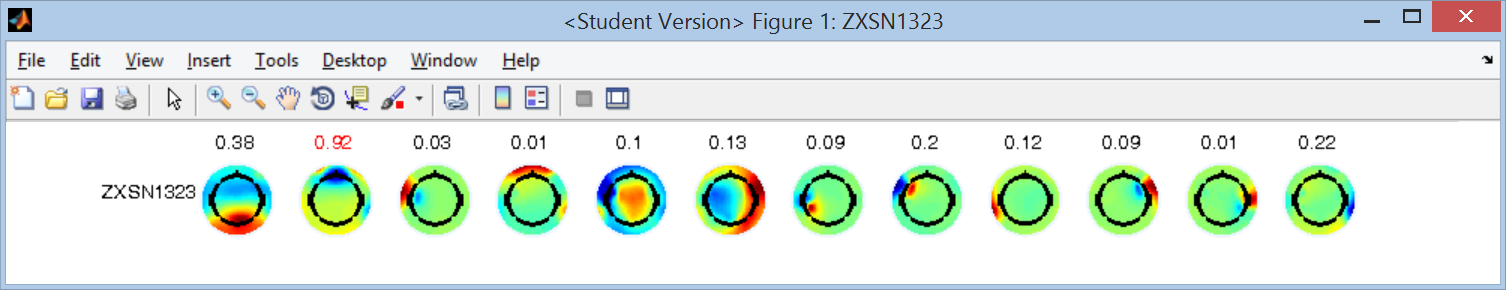
Before running batch.m it is important you have copied and edited the template initialization file and have named it Batch\_Initialize.m in the working directory of your ARP install.

Once you start batch.m, all keyboard input and console output (not images), will be logged to **logfile.txt** in your RESULTS directory. You should preserve this file for future reference, in case you run into an issue or need to go back and review what you did.

## batch1.m – statistics and initial “quick” ICA

This file runs through all the participant data, producing statistics and performing an initial “quick” ICA on the data. On a per-participant basis, all statistics, including the ICA stats, are saved to **batch1\_summary.mat** in the specified “results/(participantid)” directory.

**batch1\_summary.m** produces output plots for the components, highlighting in red any that are over the percentage specified in **variance\_portion\_quickICA** (components marked in red will be automatically rejected)**.**



Click on File-> Save As… in the plot window and save the plot in your RESULTS directory.

(Data is epoched according to the events specified in **ICA\_events** prior to ICA analysis).

Ideally, the quick ICA should run on at least 8 minutes of data (480 seconds), but sometimes your runs are not that long. The program will halt with a warning if the **chunk\_length** you specified in Batch\_Initialize.m exceeds the duration of the data in the run.

Event frequencies for each participant block file (.eeg or .cnt) are displayed to the screen. Review them to make sure they look consistent and acceptable.

## batch2.m – epoch based on paired events (if relevant), automatically reject bad trials, and merge

In **batch2.m**, the **batch1\_summary.mat** file is read in and the quick ICA stats are used to calculate the global field power. The Global Field Power (GFP) is the standard deviation of the potentials at all electrodes of an average-reference map at each point in time. GFP is used as a measure of signal-to-noise ratio.

The participant data is also read in again, and this time epoched around paired events if required (such as only selecting events with a “good” response). In the case of go-nogo where you want to save all events, you can specify **save\_unpaired\_event = 1** in your Batch\_Initialize.m file.

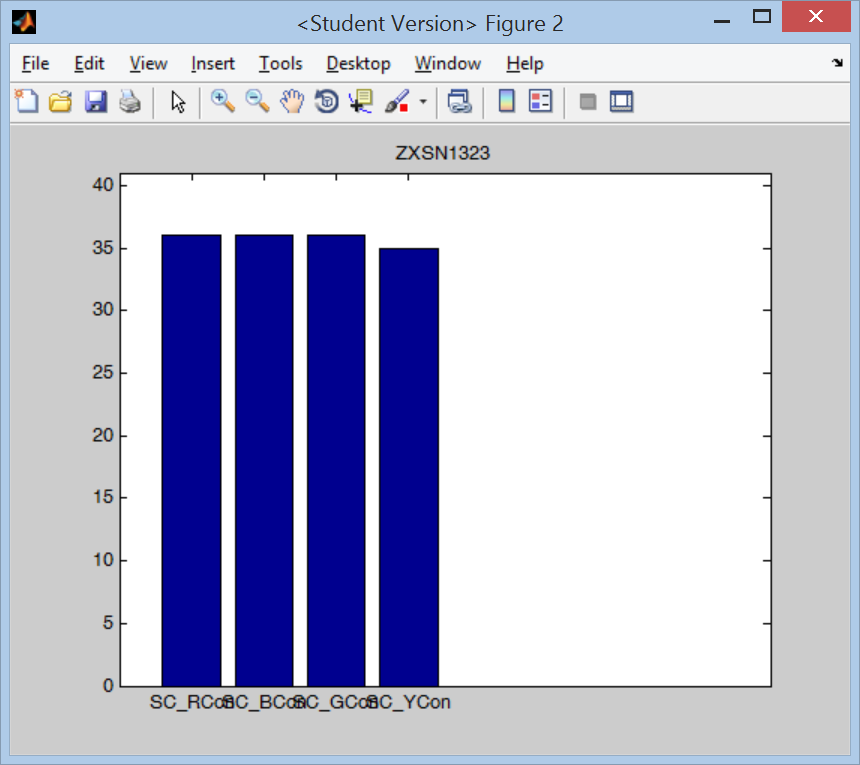
All trials are analyzed and components are once again extracted. The component global field power (standard deviation) is calculated by using only components below the 30th percentile. The gfp threshold is calculated as the 30th percentile ICA + 5\*gfp. Components are then analyzed and full trials (epochs) will be rejected if most of the components exceed threshold.

The console window will report how many trials were dropped due to exceeding gfp\_30 thresholds.

A histogram of the number of stimulus events per condition specified will be produced. If you have lots of conditions, this can become very crowded.

(We have been unable to get it to space the histogram bARP better – any help would be appreciated).

Once again, click on file->Save As… and save this histogram for future reference.



A **batch2\_summary.mat** file and **(participant)\_merged.set** file will be written to each participant subdirectory under your main RESULTS directory. The epoched conditions that are in each participant’s merged.set files are dependent on the **conditions** and matching **stims** settings in Batch\_Initialize.m.

## batch3.m – run the full ICA

For each participant, this function reads in the **(participant)\_merged.set** file and calculates a full ICA on the data, producing an **ica.mat** file for each participant.

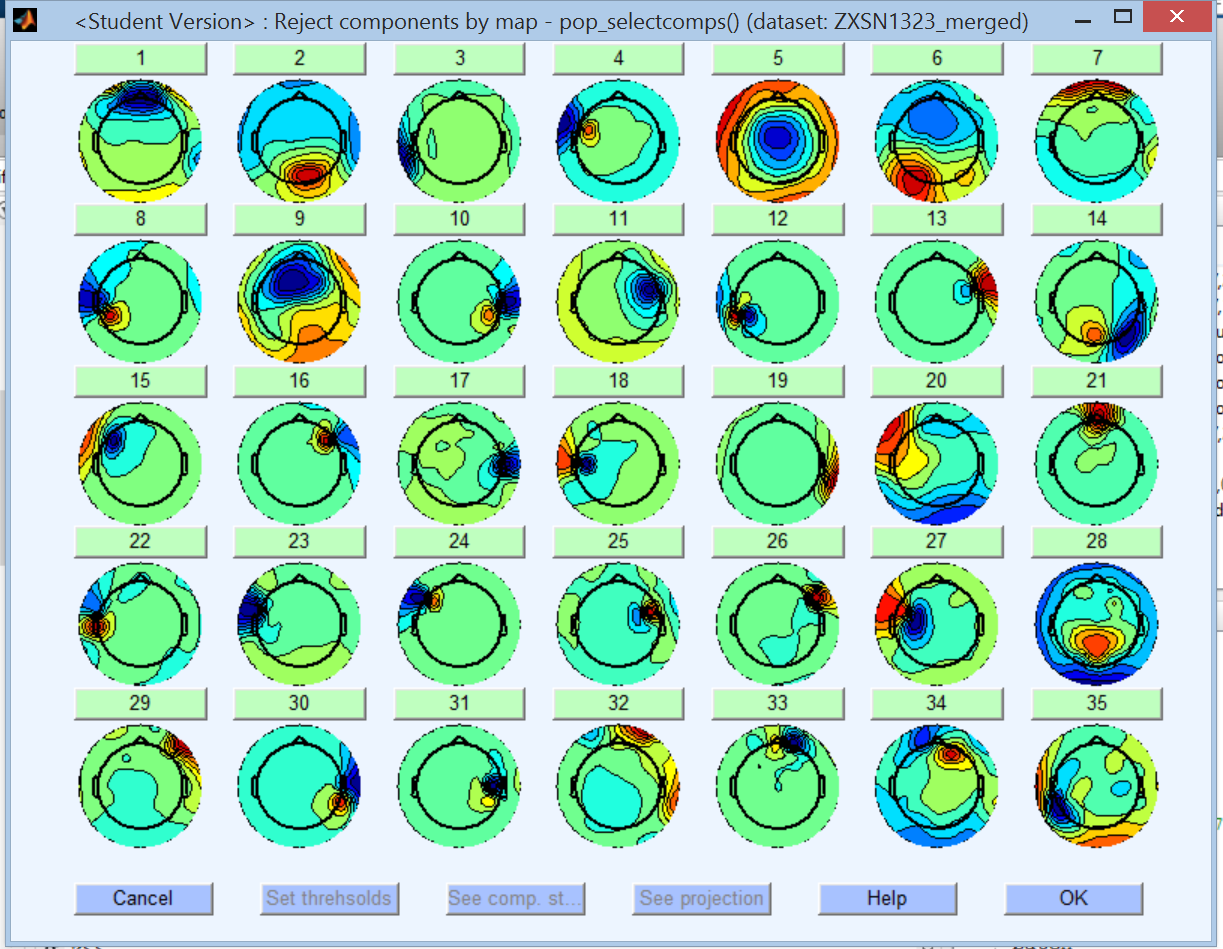
This routine uses BINICA (the faster executable version of runica), but is still VERY SLOW. This will be the slowest part of your processing and can take hours if you have lots of participants.

## batch4.m - ICA selection and removal

batch4.m displays the ICA components for a selected participant, allows you to review and select components to be rejected, and saves the information to text files for later use.

On running batch4.m, the console window will display a list of all available participants (as per your definition in the BatchInitialize.m file). You will be prompted to enter a participant identifier in the console window. At this point the program will produce a head-mapped COMPONENT plot.

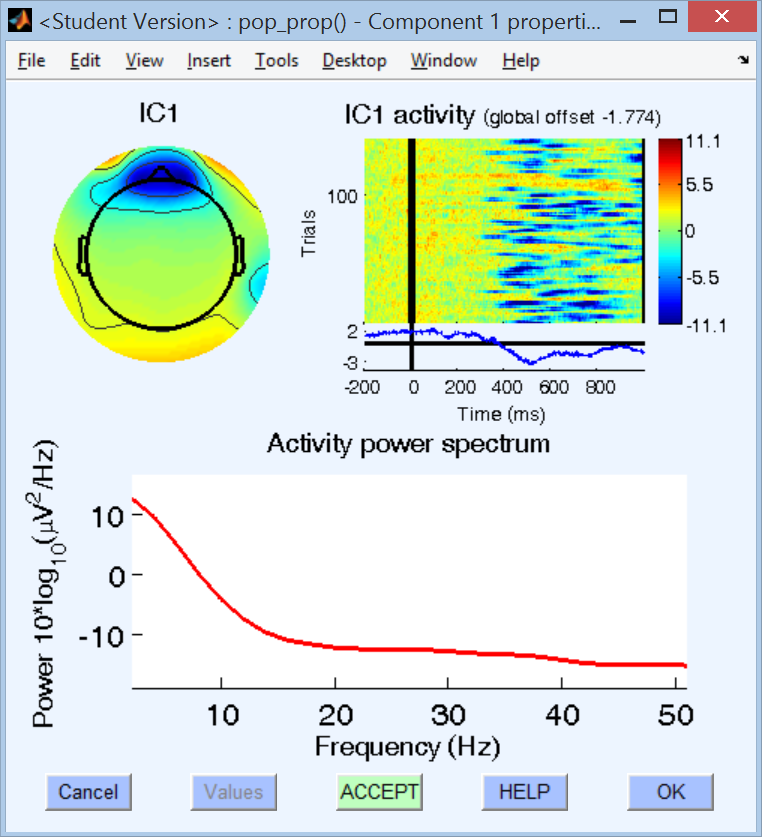
You must identify and reject bad components from this plot:



***It is absolutely critical that you do not click on this window or minimize it while it is loading or you will have to restart Batch4 again.***  (Matlab is buggy and will halt plotting if the window is at all disturbed).

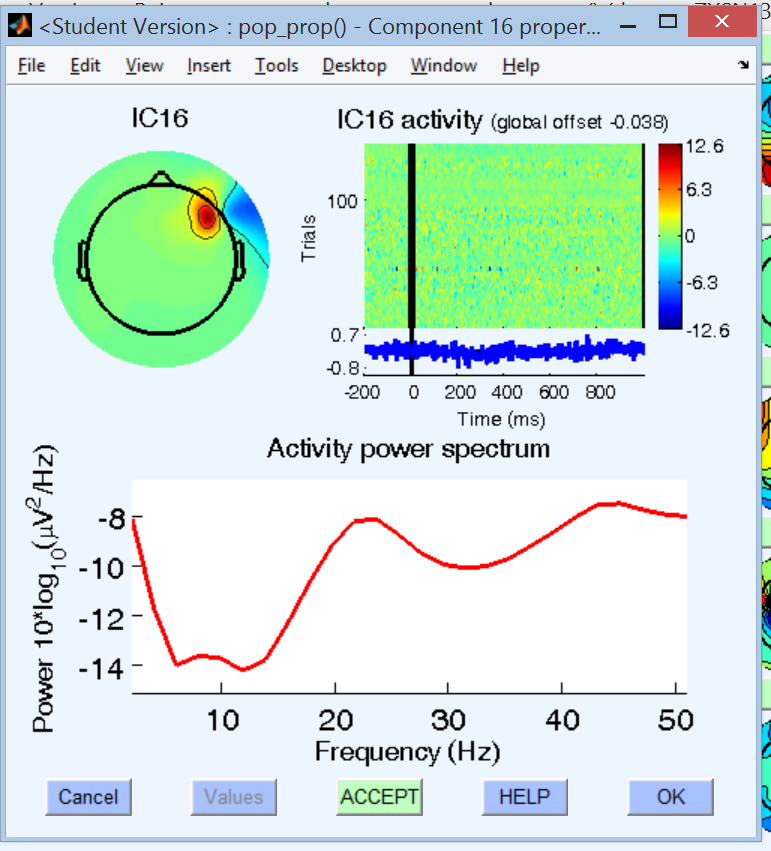
For artefact rejection, generally, anything with activity inside the head is not rejected. Look at your component plots/trace and your EEG trace to determine if the plot selected is artefact and should be removed.

Click on the numbered button ABOVE a suspected component and analyze whether or not it is an artefact and must be removed:



Even though this component looks like it might be squinting or an eye blink, it shows power at the back of the head and a decided negative offset starting at 400. It may be imagery (i.e. visualizing), where the power is not actually showing up on the head map directly. (it may have deflection – too much activity in the frontal lobes). Click on OK to leave this component alone.

The curve on the power spectrum indicates legitimatelegitmate signal.



This component looks completely noisy – no power dips during the epoch. This is a classic indicator of noise, and should be removed.

Click on ACCEPT to reject this component

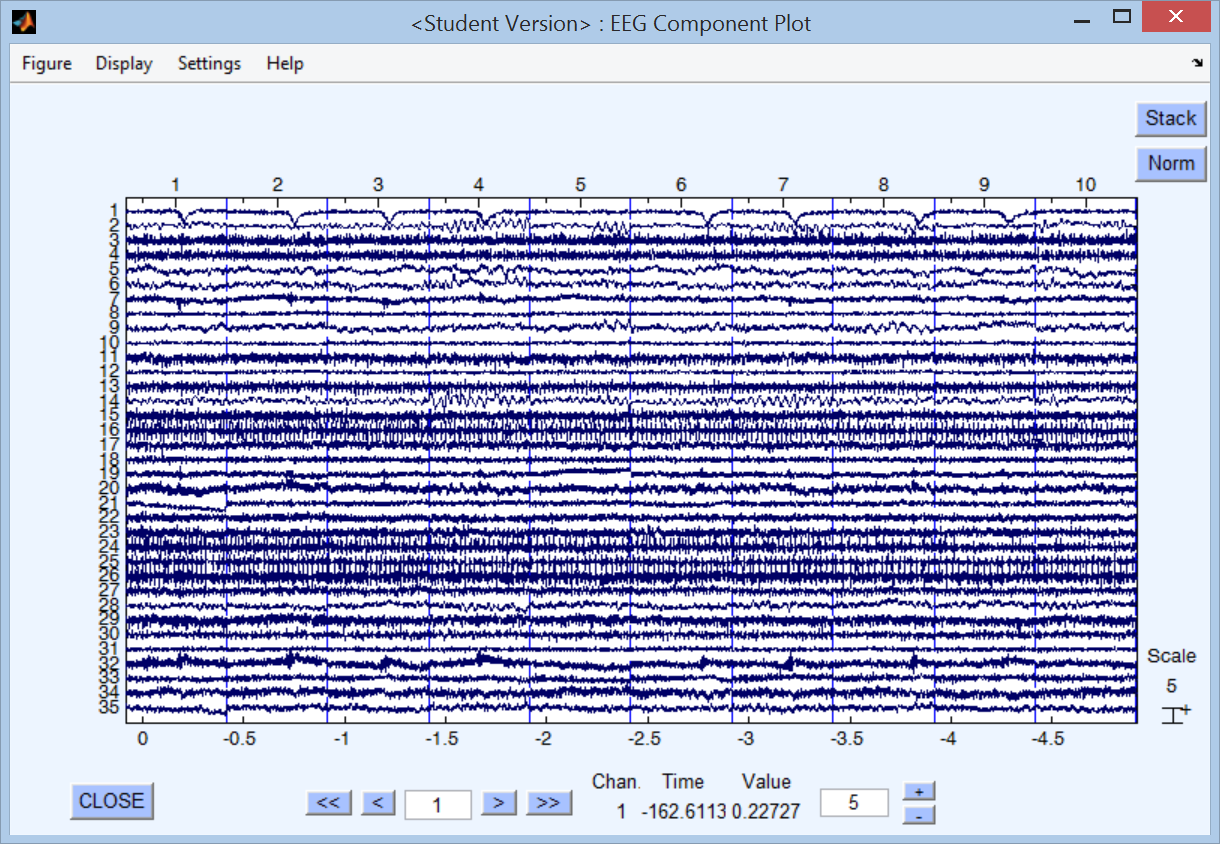
Power spectrum does not show nice curve

When you have selected all your components, write them down – and mark which ones are EYEBLINK vs SACCADEs.

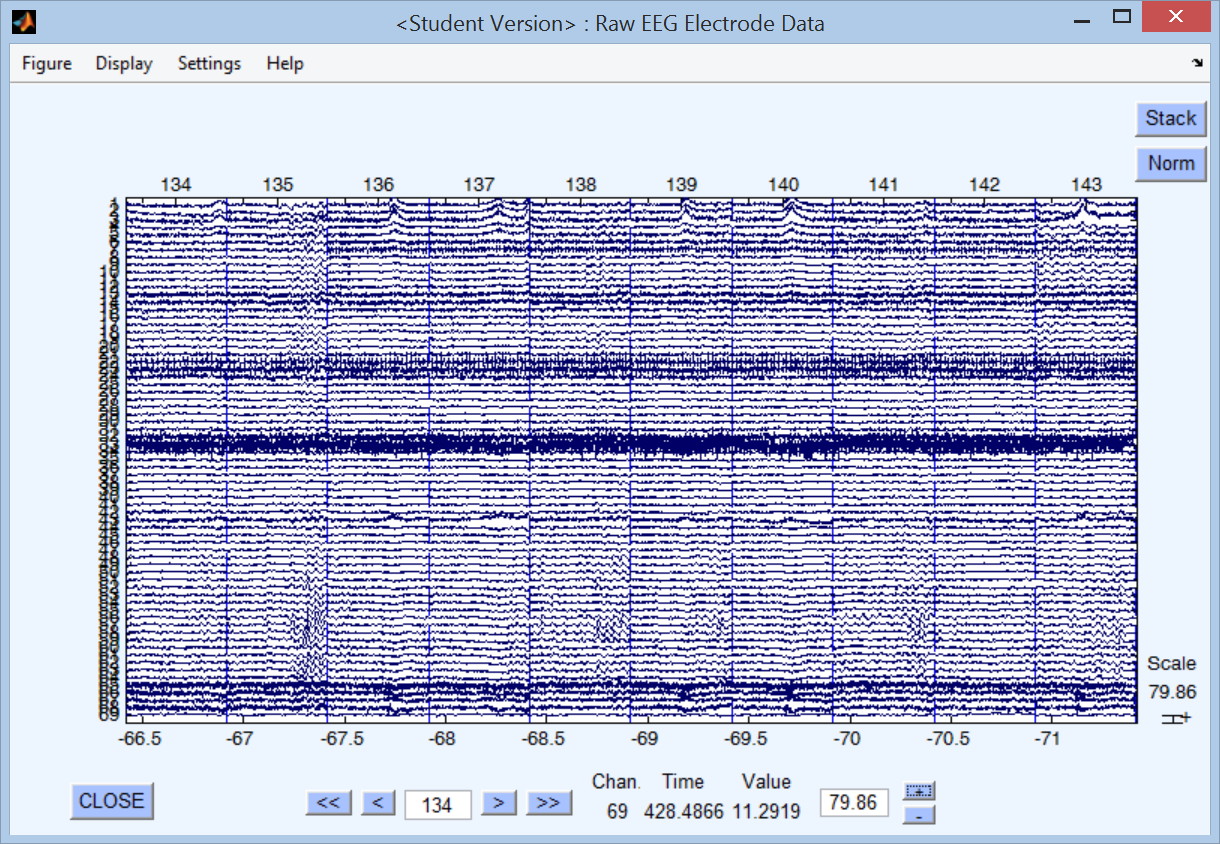
**NOTE: DO NOT click the “OK” BUTTON on the main “Reject Components by Map” window – you will get an error. When you are done, close this window by clicking on the “x” in the upper right corner.**

(More details on how to do ICA analysis and select components for removal are in the ***ICA Analysis*** sectionof this document).

The program will also display a COMPONENT PLOT which is useful to review when looking at individual components:



Additionally, you will get a plot of the raw EEG Electrode data:



From either of these two plots, you can (click on and) highlight and reject parts of trials. To apply rejections to the continuous raw data or component data from these plots, Click on the menu item, **Figure->Accept and Close.** It is not recommended to reject from the continuous raw data. Use the ICA analysis and component plots to determine the best artefacts to remove. When you highlight and reject parts of trials as described, you are also losing EEG. Please remember that when you are deciding on artefacts to remove.

When all data editing and component selection are complete, go back to the Matlab console window, and at the prompt type **return** and hit enter.

You will see a prompt:

**Enter additional components (by number) that**

**should be removed from the entire dataset, ensuring**

**that there are spaces between each number and the set**

**is enclosed in square brackets i.e. "[1 5 9]":**

Anything that you selected via the ACCEPT button will automatically be used to do the ICA correction. Only enter values here if you have any ADDITIONAL components to remove. (This is expert level, so in general, press ENTER at this prompt) until you are clear about what every aspect of this program does and how it affects your EEG data.

You will then see the following prompt:

**Enter components that should be committed to the EYEBLINK library**

**enclosed in square brackets, i.e. "[1 5 9]":**

Enter the components you rejected that were related to EYEBLINKS – these are generally components with power across the front of the head (very little if any inside the head), generally in one colour. Eye blinks are saved in **ica\_blinkcomps.txt** in the participants’ results directory.

You will then see the following prompt:

**Enter components that should be committed to the SACCADE library**

**enclosed in square brackets, i.e. "[1 5 9]":**

Enter the components you rejected that were related to SACCADES – these generally appear as red on one side of the head and blue on the other side. Saccades are saved in **ica\_saccadecomps.txt** in the participant’s results directory.

These will be saved in **librejcomps.txt** at the top level of your data directory (above the individual participant directories), and the specified components will be removed for ALL participants processed going forward (in Batch5.m).

*While we currently don’t use the Eye Blink and Saccade components separately, there is an expectation in the future that we can combine this with the ICA data from each participant to build a better ICA removal algorithm for Eye components.*

All the rejected components for this participant will be written out to **rejcomps.txt** in your RESULTS directory.

***NOTE: Because this is a lengthy process, batch4.m does not loop through all participants. You must run batch4.m individually for EVERY PARTICIPANT you want to process because you are selecting the components for that participant alone. The easiest way to do this is from the console command line: >batch4***

## batch5.m – Artefact REMOVAL and merging of sets

batch5.m processes every participant in your data directory. It takes the rejected components that you identified in batch4, plus the ICA files and merged .set files from batch3. It removes trials that contain excessive amounts of bad components, and then epochs data, producing .set files for each condition.

For each participant:

* Loads the **merged.set** files
* Loads the **rejcomps.txt** and **librejcomps.txt** files
* Loads the **ica.mat** file
* Applies the ICA data to the EEG merged set data to determine the 40th percentile for global field power
* Cleans each trial by subtracting rejected components plus components that are above 40p+6\*stddev threshold
* Splits data into condition-specific merged subsets per participant, based on variables **conditions** and **allowable\_values** from the **Batch\_Initialize.m** file, and saves individual ***condition*.set** files.
* For each condition specified, displays the number of matching trials (epochs), and the number rejected from the total in the merged set. (i.e., not relevant for this condition). Then it remaps the events to the selected epochs. Don’t be concerned with the “removing unreferenced events” – it’s looking at the entire merged set, so any events (stimulus or response) that are not part of the current condition are “rejected” for that condition. Example output for one condition:

**Condition: SC\_RCon has 36 matching trials**

**107/143 trials rejected**

**Removing 107 trial(s)...**

**Pop\_select: removing 159 unreferenced events**

**Saving dataset...**

# ICA Analysis techniques

This section needs to be completed. (with input from Patricia and more example screenshots)

* One colour in front: blink.
* One colour one eye, another colour on the other eye: saccade – anywhere on front of head
* Write down each component and what type for later inclusion in library
* If noise, there will be no signal inside head
* In the signal plot right below the IC Activity for a given component, if there are NO peaks or troughs, it’s likely just noise. If it is just noise, the IC plot itself should be random colours, with no distinct vertical banding in the spectral plot.
* Check for over-processing of data….

# AVERAGING Your Data (can only be run after completing batch5.m)

Once ICA has been performed, data can be averaged. Merged .SET files are used as input to the final averaging.

The **create\_avrs.m** file includes **Batch\_Initialize.m** from the local directory. Batch\_Initialize.m contains parameters which also define averaging behavior (based on **merged\_conditions** and **merged\_names** values). The .avr file produced is in ASCII format and can be imported into BESA for further processing.

You will need to edit **create\_avrs.m** to change key parameters in the output header:

% You must modify the fprintf line to contain the correct values:

**fprintf(fid, '%s %d\n', ['Npts= 2400 TSB= -200.0 DI= 0.5 SB= 1.000 SC= 200 Nchan=',nchans]);**

**% Npts - based on size of epoch and frequency (i.e. 2400 for 2.4 seconds at 1Hz)**

**% TSB - start of baseline (prior to stimulus) in ms (i.e. -200)**

**% DI – ms before stimulus?**

**% SB – baseline in seconds?**

**% SC - ?**

**% Nchan = number of channels used**

For each participant in the data directory, averages will be created for every sub-condition within every set of merged\_conditions, as well as a grand average for the overall merge condition. For example

**merged\_conditions = {{'SC\_RCon','SC\_BCon','SC\_GCon', 'SC\_YCon'}};**

**merged\_names = {'SCCong'};**

will produce .avr files for each of the **merged\_conditions** and an overall grand average of all conditions , saved in **SCCong.avr**.

If you have more than one group that you want to merge, you can include them by adding the new set of condition names encapsulated in curly braces as follows:

**merged\_conditions = {{'SC\_RCon','SC\_BCon','SC\_GCon', 'SC\_YCon'} {'SW\_RCon','SW\_BCon','SW\_GCon', 'SW\_YCon'} };**

**merged\_names = {'SCCong', ‘SWCong’};**

**Note that these parameters must be set and used when running batch4**

# TBD

* **Fix bugs so warnings do not appear**

Warning that appears when starting to process data:

Warning CNTOPEN: month and day were mixed up 2012-20-9-14-46-25

readlocs() warning: Fewer columns in the input than expected.

See >> help readlocs

* **FREQUENCY analysis**
* **GUI to allow processing steps to be completed without having to manually modify batch.m**

# Resolving "Out of Memory" Errors in Matlab

[General Suggestions for Reclaiming Memory](#_General_Suggestions_for)

[Setting the Process Limit](#_Setting_the_Process)

[Disabling Java VM on Startup](#_Disabling_Java_VM)

[Increasing System Swap Space](#_Increasing_System_Swap)

[Using the 3GB Switch on Windows Systems](#_Windows_Systems)

[Freeing Up System Resources on Windows Systems](#_Linux_Systems)

### General Suggestions for Reclaiming Memory

The MATLAB® software generates an Out of Memory message whenever it requests a segment of memory from the operating system that is larger than what is currently available. When you see the Out of Memory message, use any of the techniques discussed under [Strategies for Efficient Use of Memory](http://www.mathworks.com/help/matlab/matlab_prog/strategies-for-efficient-use-of-memory.html) to help optimize the available memory. If the Out of Memory message still appears, you can try any of the following:

* Compress data to reduce memory fragmentation.
* If possible, break large matrices into several smaller matrices so that less memory is used at any one time.
* If possible, reduce the size of your data.
* Make sure that there are no external constraints on the memory accessible to MATLAB. (On UNIX® systems, use the limit command to check).
* Increase the size of the swap file. We recommend that you configure your system with twice as much swap space as you have RAM. See [Increasing System Swap Space](http://www.mathworks.com/help/matlab/matlab_prog/resolving-out-of-memory-errors.html#brh72ex-54), below.
* Add more memory to the system.

### Setting the Process Limit

The platforms and operating systems that MATLAB supports have different memory characteristics and limitations. In particular, the *process limit* is the maximum amount of virtual memory a single process (or application) can address. On 32-bit systems, this is the most important factor limiting data set size. The process limit must be large enough for MATLAB to store all of the data it is to process, any MATLAB program files, the MATLAB executable itself, and additional state information.

Where possible, choose an operating system that maximizes this number, that is, a 64-bit operating system. The following is a list of MATLAB supported operating systems and their process limits.

| **Operating System** | **Process Limit** |
| --- | --- |
| 32-bit Microsoft® Windows® XP, Windows Vista™, Windows 7 and higher | 2 GB |
| 32-bit Windows XP with 3 GB boot.ini switch or 32-bit Windows Vista or Windows 7 and higher with increaseuserva set (see later) | 3 GB |
| 32-bit Linux® (Linux is a registered trademark of Linus Torvalds) | ~3 GB |
| 64-bit Windows or Linux running 32-bit MATLAB | ≤ 4 GB |
| 64-bit Windows, Apple Macintosh OS X, or Linux running 64-bit MATLAB | 8 TB |

To verify the current process limit of MATLAB on Windows systems, use the [memory](http://www.mathworks.com/help/matlab/ref/memory.html) function.

Maximum possible array: 583 MB (6.111e+008 bytes) \*

Memory available for all arrays: 1515 MB (1.588e+009 bytes) \*\*

Memory used by MATLAB: 386 MB (4.050e+008 bytes)

Physical Memory (RAM): 2014 MB (2.112e+009 bytes)

\* Limited by contiguous virtual address space available.

\*\* Limited by virtual address space available.

When called with one output variable, the memory function returns or displays the following values. See the function reference for [memory](http://www.mathworks.com/help/matlab/ref/memory.html) to find out how to use it with more than one output.

| **memory Return Value** | **Description** |
| --- | --- |
| MaxPossibleArrayBytes | Size of the largest single array MATLAB can currently create |
| MemAvailableAllArrays | Total size of the virtual address space available for data |
| MemUsedMATLAB | Total amount of memory used by the MATLAB process |

View the value against the Total entry in the Virtual Memory section. It is shown as 2 GB in the table, which is the default on Windows XP systems. On UNIX systems, see the ulimit command to view and set user limits including virtual memory.

### Disabling Java VM on Startup

On UNIX systems, you can increase the workspace size by approximately 400 MB if you start MATLAB without the Java® JVM™. To do this, use the command line option -nojvm to start MATLAB. This also increases the size of the largest contiguous block (and therefore the largest matrix) by about the same.

Using -nojvm comes with a penalty in that you will lose many features that rely on the Java software, including the entire development environment.

Starting MATLAB with the -nodesktop option does not save any substantial amount of memory.

Shutting down other applications and services (e.g., using msconfig on Windows systems) can help if total system memory is the limiting factor, but usually process limit (on 32-bit machines) is the main limiting factor.

### Increasing System Swap Space

The total memory available to applications on your computer is comprised of physical memory (RAM), plus a *page file*, or *swap file*, on disk. The swap file can be very large (e.g., 16 TB on 32-bit Windows, 512 TB on 64-bit Windows). The operating system allocates the virtual memory of each process to physical memory or to the swap file, depending on the needs of the system and other processes.

Most systems allow you to control the size of your swap file. The steps involved depend on the system you are running on.

|  |
| --- |
| **Note:**   There is no interface for directly controlling the swap space on Macintosh OS X systems. |

#### Windows Systems

Use the Windows Control Panel to change the size of the virtual memory paging file on your system. For more information, refer to the Windows help.

#### Linux Systems

You can change your swap space by using the mkswap and swapon commands. For more information on the above commands, type man followed by the command name at the Linux prompt.

### Using the 3GB Switch on Windows Systems

Microsoft Windows XP systems can allocate 3 GB (instead of the default 2 GB) to processes, if you set an appropriate switch in the boot.ini file of the system. MathWorks® recommends that you only do this with Windows XP SP2 systems or later. This gives an extra 1 GB of virtual memory to MATLAB, not contiguous with the rest of the memory. This enables you to store more data, but not larger arrays, as these are limited by contiguous space. This is mostly beneficial if you have enough RAM (e.g., 3 or 4 GB) to use it.

After setting the switch, confirm the new value of the virtual memory after restarting your computer and using the[memory](http://www.mathworks.com/help/matlab/ref/memory.html) function.

[userview systemview] = memory;

systemview.VirtualAddressSpace

ans =

Available: 1.6727e+009 % Virtual memory available to MATLAB.

Total: 2.1474e+009 % Total virtual memory

For more documentation on this option, use the following URL:

<http://support.microsoft.com/kb/291988>

Similarly, on machines running Microsoft Windows Vista and Windows 7 and higher, you can achieve the same effect by using the command:

BCDEdit /set increaseuserva 3072

For more information on this option, go to the following website:

[http://msdn.microsoft.com](http://msdn.microsoft.com/)

### Freeing Up System Resources on Windows Systems

There are no functions implemented to manipulate the way MATLAB handles Microsoft Windows system resources. Windows systems use these resources to track fonts, windows, and screen objects. Resources can be depleted by using multiple figure windows, multiple fonts, or several UI controls. One way to free up system resources is to close all inactive windows. Windows system icons still use resources.