# Analyzing Bagged Data in Octave/Matlab

Wyatt Newman

February, 2015

This document describes how to log data in ROS and interpret the data in Matlab or Octave.

**Bagging data:**

We have seen that ROS provides a convenient logging tool, "rosbag". When conducting experiments (physical or simulated), it is often convenient to log sensor data for post processing. These files can quickly get very large when camera data is included, so it is prudent to specify which topics are to get logged.

Start with a robot running—e.g. cwruBot or Jinx. Start up whatever nodes are desired (e.g., velocity profiler, steering algorithm, etc.) and begin logging. You can start/stop the logger at any time without disrupting the robot control.

When performing logging, first navigate to a directory where you desire the logs to reside. From a terminal at this directory, start recording.

A specific example uses cwruBot and example_robot_commander/example_cwruBot_commander. The example_cwruBot_commander augments the cmd_vel message to include a header. The message is: geometry_msgs::TwistStamped twist_cmd2, and the topic is: cmd_vel_stamped. In each iteration of velocity commands, the header's time stamp is filled in with the line:

```
  twist_cmd2.header.stamp = ros::Time::now();
```

This looks up the current time and uses it to fill in the "stamp" field of the twist_cmd2 message. This allows the logger to associate the recorded velocity command with a uniform system clock, and thus this data can be aligned with other logged topics that also contain a header field with time stamps.

To record the topics cmd_vel_stamped and odom, enter:

```
  rosbag record cmd_vel_stamped odom
```

This will start recording messages transmitted to these topics. The output file will, by default, be named with the current date and time. Optionally, you can specify an output file name, e.g. "exp":

```
      rosbag record -O exp odom cmd_vel_stamped
```

in which case the bag file will be contained in a new file called "exp.bag".

You can also list more topics to be bagged, if desired.

When you have enough data of interest, stop the logging with a control-C in the terminal from which you started rosbag.

The bag file may be examined with:

```
        rosbag info exp.bag
```

**Converting to Matab format:**

For post-processing logged data, it is convenient to convert the bag file into a format compatible with Matlab/Octave. There is a tool for this (written by Devin Schwab and Neal Aungst of CWRU). It is located in …/catkin/src/cwru_base_hydro/rosbag_to_matlab. The README file contained in this package includes details of operation.

To use this converter, enter:

```
rosrun rosbag_to_matlab bag_to_mat.py <path to bag file> <output directory>
```

`<path to bag file>` is the path to the bag file you wish to convert

`<output directory>` is the path to the directory where the .mat files will be written.

For example, from the same terminal used for bagging (and still in the same directory), enter:

```
rosrun rosbag_to_matlab bag_to_mat.py exp.bag .
```

This will convert the file "exp.bag", located in the current directory, and will create output files called "exp_odom.mat" and "exp_cmd_vel_stamped.mat" in the same directory (since "." is Unix shorthand for "current directory"). The file name prepends the original file name and post-pends the name of the topics logged (odom and cmd_vel_stamped). The output file will be in a format compatible with Matlab or Octave.

**Interpreting Data in Matlab/Octave:**
From a Matlab or Octave command window, navigate to the directory where your data resides. For the current example, the converted files ar called "exp_odom.mat" and "exp_cmd_vel_stamped.mat". In the command line, enter:

```
  load  exp_odom.mat
```

Then type "who", and you will see that two new variables exist: "data" and "header"
Enter: header
and you will see a long list of elements stored from odom. A snippet of this follows:

```
header =


header.seq
header.stamp.secs
header.stamp.nsecs
header.frame_id
child_frame_id
pose.pose.position.x
pose.pose.position.y
pose.pose.position.z
pose.pose.orientation.x
pose.pose.orientation.y
pose.pose.orientation.z
pose.pose.orientation.w
```

Entering: size(header) reveals that the header describes 90 variables stored in the bag file.

An inconvenience is that one must assign the correct index number to each of the data items logged, in order to plot or analyze specific sensor values.

An example Octave file is located in:
cwru-ros-pkg-hydro/Octave/odom_plotter.m

An excerpt from this program is:

```
load exp_odom.mat
odom_x = data(:,6);
odom_y = data(:,7);
odom_qz = data(:,11);
odom_qw = data(:,12);
odom_heading = 2.0*atan2(odom_qz,odom_qw);%cheap conversion from quaternion to heading
vel = data(:,49);
omega = data(:,54);
```

In the above, data components of interest are extracted from the full bag file.

The header from the odom message contains a ROS time stamp, recorded in separate fields of seconds and nanoseconds. These are combined with the lines:

```
odom_secs=data(:,2);
odom_nsecs=data(:,3);
odom_time= odom_secs+odom_nsecs*0.000000001;
```

The ROS time, however, starts at an arbitrary (typically large) value. To make the time vector more intuitive, one can subtract the first value from all values of the vector, so time starts at zero:

```
odom_start_time = odom_time(1)
odom_time = odom_time-odom_start_time;
```

Next, the stamped velocity command is loaded, and values of interest are extracted:

```
cmd_vel = data(:,5);
cmd_omega = data(:,10);
```

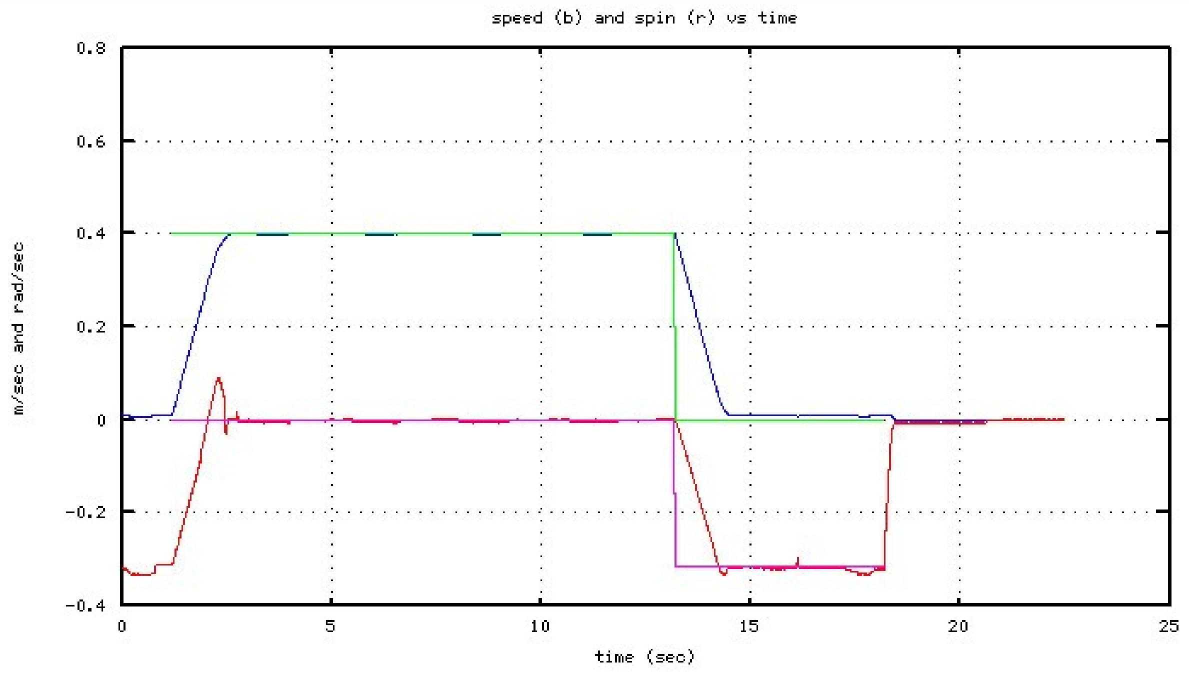
These values are also associated with time stamps.

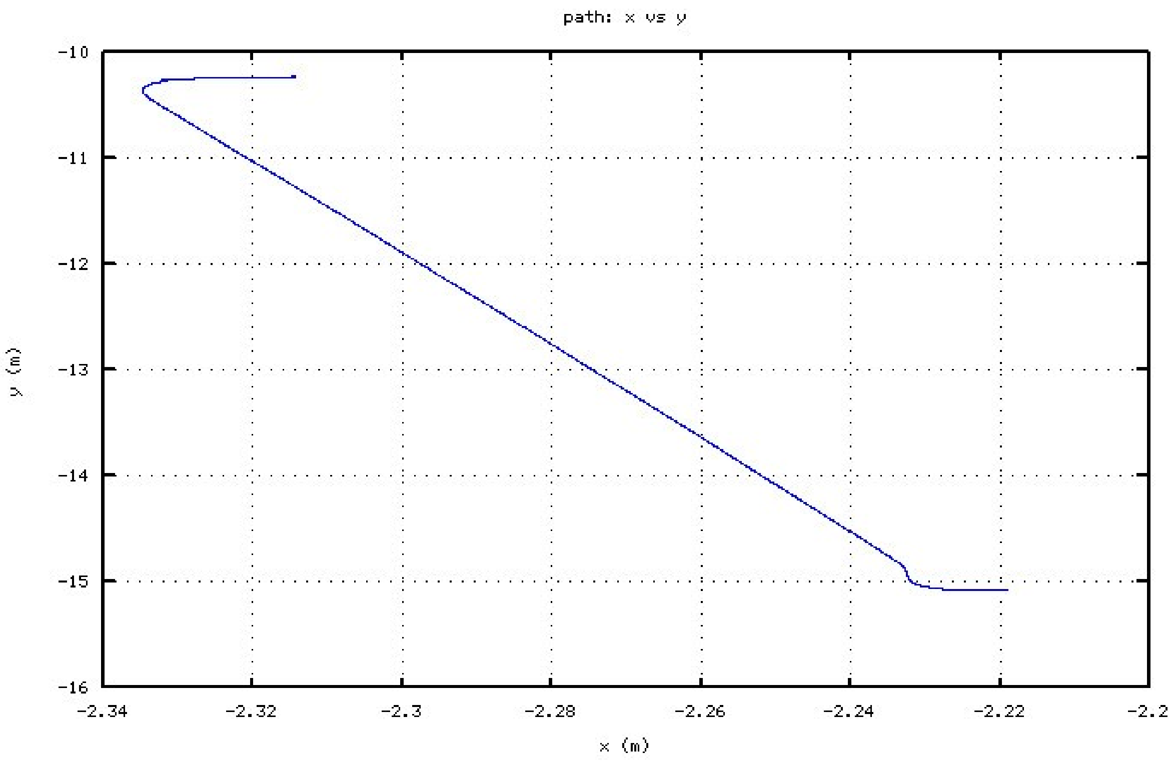
```
cmd_vel_secs=data(:,2);
cmd_vel_nsecs=data(:,3);
cmd_vel_time= cmd_vel_secs+cmd_vel_nsecs*0.000000001;
```

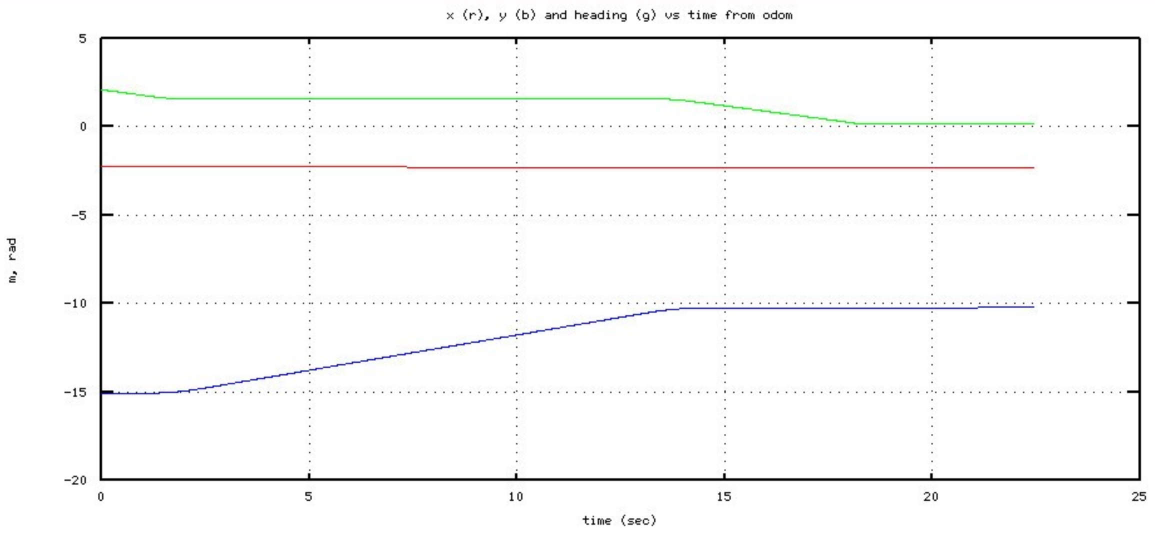
These time stamps use the same ROS time, and thus they are aligned. However, to keep them aligned, the second time vector should use the same offset as the first time vector:

```
cmd_vel_time = cmd_vel_time-odom_start_time; %offset relative to odom time, so time is aligned
```

Running this Octave code on example data from cwruBot yields useful plots, examples of which are below:


speed (b) and spin (r) vs time
m/sec and rad/sec
0.8
0.6
0.4
0.2
0
-0.2
-0.4
0
5
10
15
20
25
time (sec)


path: x vs y
y (m)
-10
-11
-12
-13
-14
-15
-16
-2.34
-2.32
-2.3
-2.28
-2.26
-2.24
-2.22
-2.2
x (m)



These plots show a history of the path of the robot (vs. time or x vs. y), as well as the response of speed and spin (according to odom) to input commands.