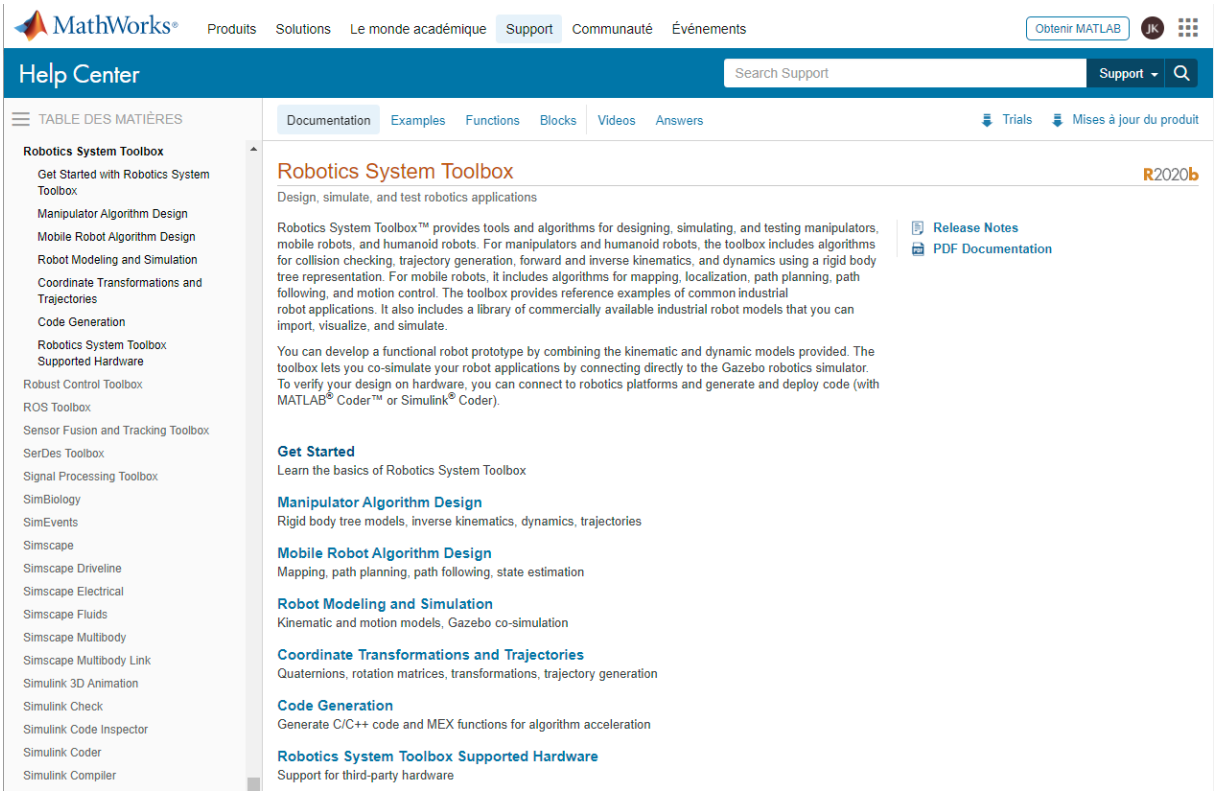
Toolbox ROS => 'rosbag' => SLAM App => 'Automated_occupancy_lidar_map' => IHM

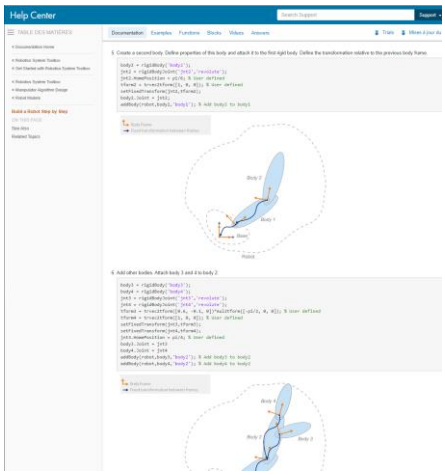
1) Créer un ROSbag depuis la ROStoolbox

Tutoriel : https://fr.mathworks.com/help/robotics/index.html?s_cid=doc_ftr



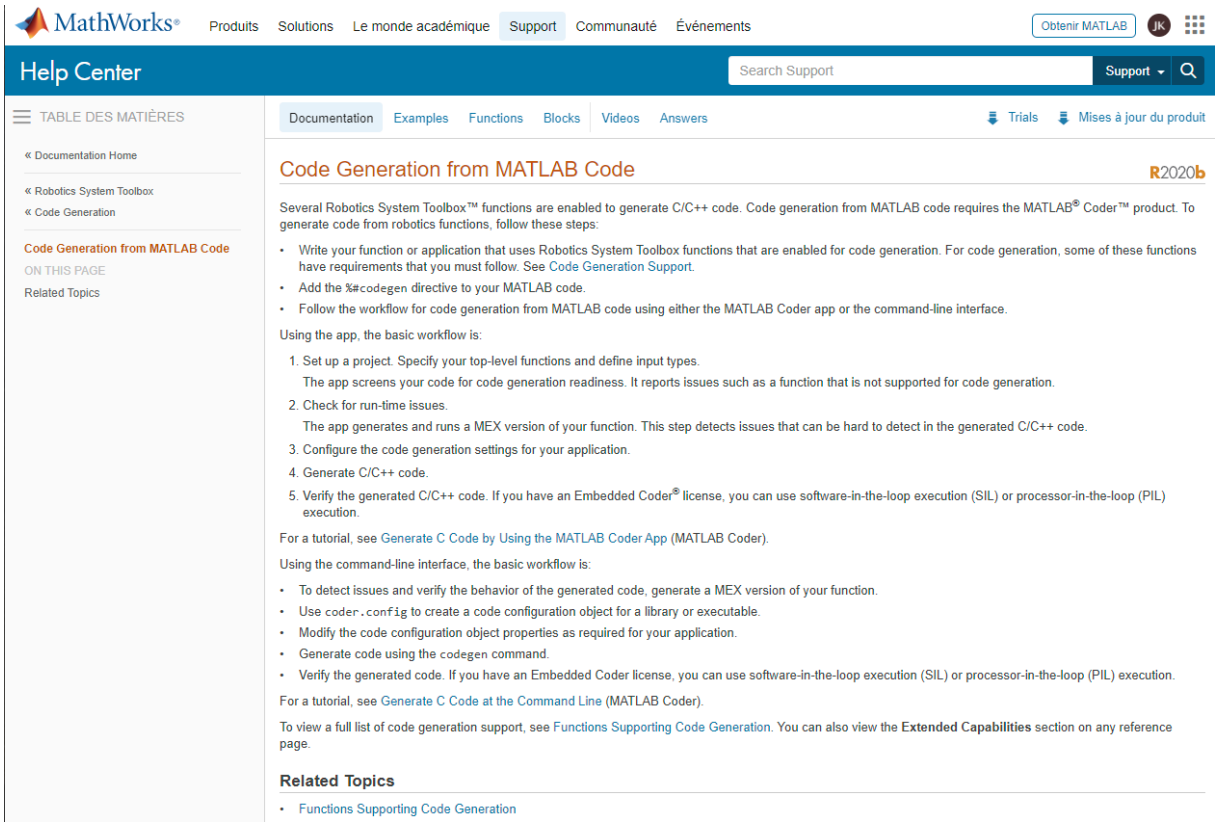
Points d'intérêt du tutoriel :

- Possibilité de créer des articulations et des membres :



https://fr.mathworks.com/help/robotics/ug/build-a-robot-step-by-step.html

- **Génération automatique de code « In the Loop » :**



MathWorks® Produits Solutions Le monde académique Support Communauté Événements Obtenir MATLAB

Help Center Search Support Support

TABLE DES MATIÈRES

« Documentation Home

« Robotics System Toolbox

« Code Generation

Code Generation from MATLAB Code

ON THIS PAGE

Related Topics

Documentation Examples Functions Blocks Videos Answers Trials Mises à jour du produit

Code Generation from MATLAB Code

R2020b

Several Robotics System Toolbox™ functions are enabled to generate C/C++ code. Code generation from MATLAB code requires the MATLAB® Coder™ product. To generate code from robotics functions, follow these steps:

- Write your function or application that uses Robotics System Toolbox functions that are enabled for code generation. For code generation, some of these functions have requirements that you must follow. See Code Generation Support.
- Add the `%#codegen` directive to your MATLAB code.
- Follow the workflow for code generation from MATLAB code using either the MATLAB Coder app or the command-line interface.

Using the app, the basic workflow is:

1. Set up a project. Specify your top-level functions and define input types.
   The app screens your code for code generation readiness. It reports issues such as a function that is not supported for code generation.
2. Check for run-time issues.
   The app generates and runs a MEX version of your function. This step detects issues that can be hard to detect in the generated C/C++ code.
3. Configure the code generation settings for your application.
4. Generate C/C++ code.
5. Verify the generated C/C++ code. If you have an Embedded Coder® license, you can use software-in-the-loop execution (SIL) or processor-in-the-loop (PIL) execution.

For a tutorial, see Generate C Code by Using the MATLAB Coder App (MATLAB Coder).

Using the command-line interface, the basic workflow is:

- To detect issues and verify the behavior of the generated code, generate a MEX version of your function.
- Use `coder.config` to create a code configuration object for a library or executable.
- Modify the code configuration object properties as required for your application.
- Generate code using the `codegen` command.
- Verify the generated code. If you have an Embedded Coder license, you can use software-in-the-loop execution (SIL) or processor-in-the-loop (PIL) execution.

For a tutorial, see Generate C Code at the Command Line (MATLAB Coder).

To view a full list of code generation support, see Functions Supporting Code Generation. You can also view the **Extended Capabilities** section on any reference page.

Related Topics

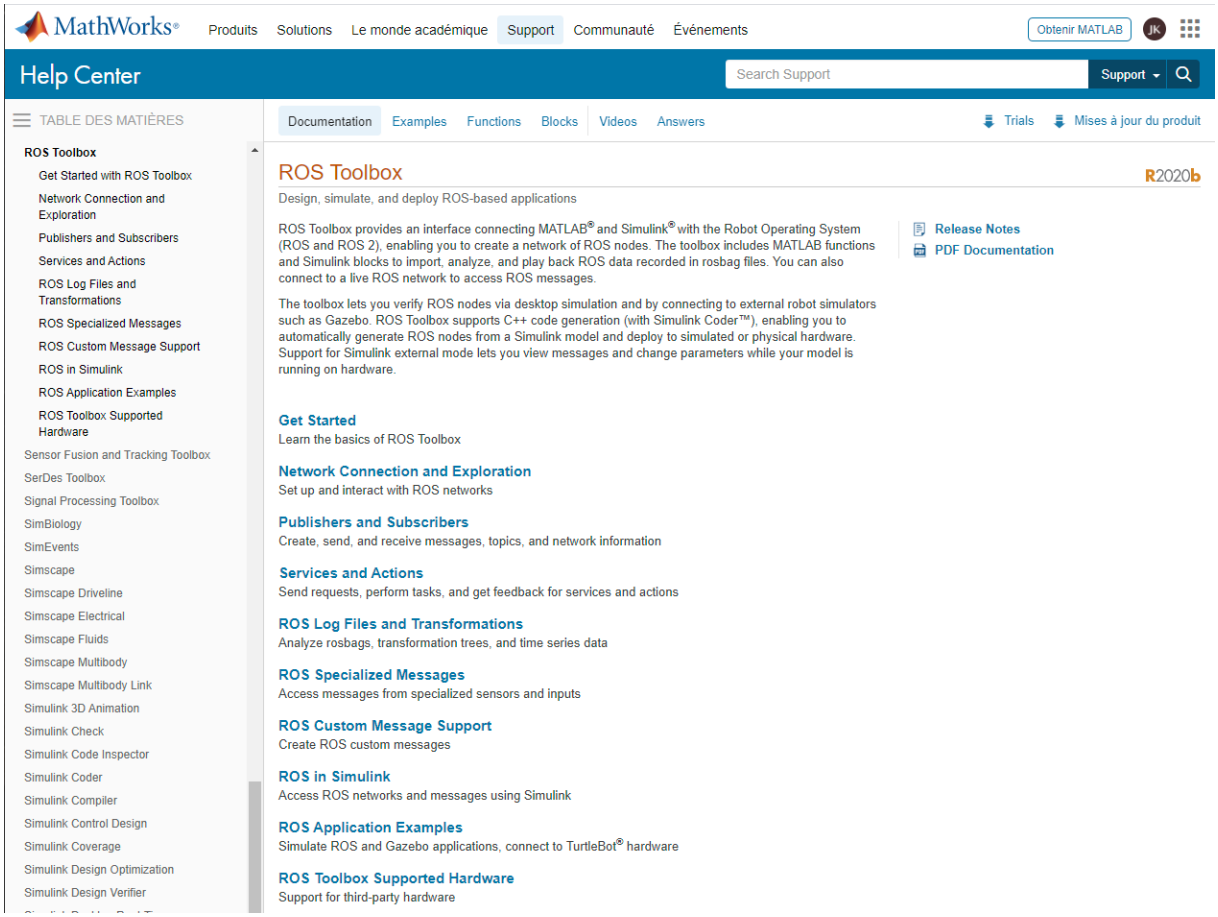
- Functions Supporting Code Generation

https://fr.mathworks.com/help/robotics/ug/code-generation-from-matlab-code.html

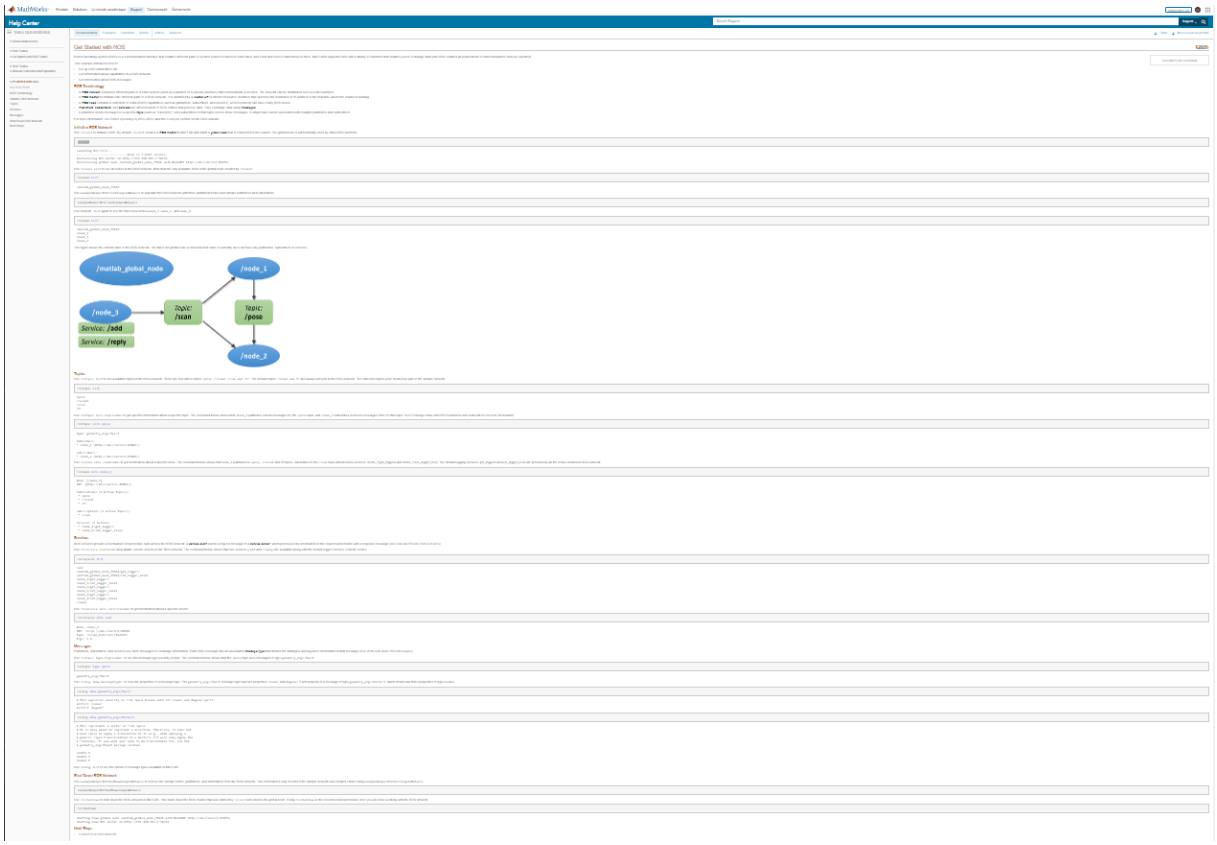
---

ROS TOOLBOX :

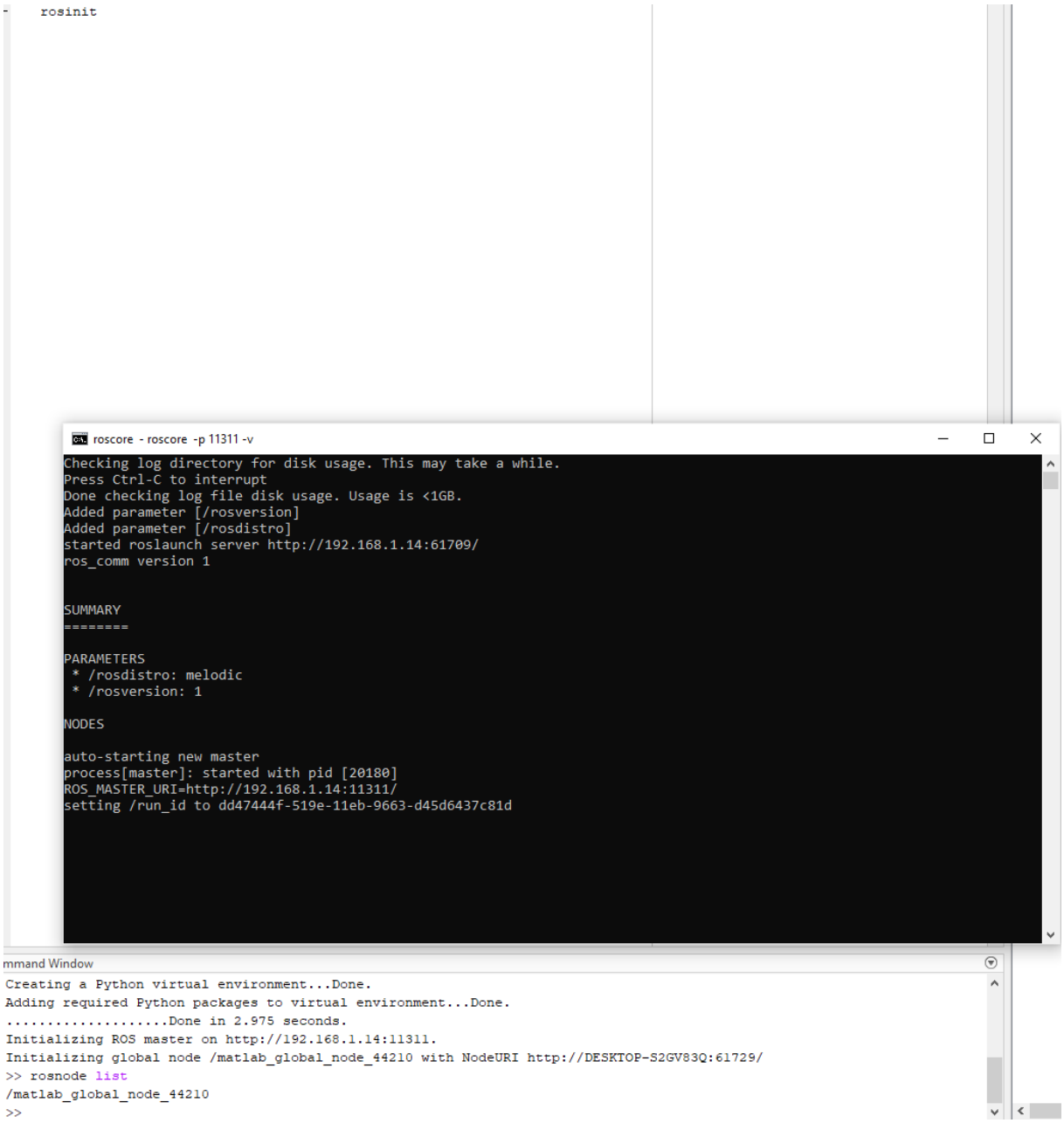
https://fr.mathworks.com/help/ros/index.html



Prise en main :



⇨ Nécessite l'installation de l'environnement Python Windows pour « rosinit »



Etude des ROSbags :

https://fr.mathworks.com/help/ros/ug/ros-log-files-rosbags.html

ROS Log Files (rosbags)

R2020b

Introduction

A rosbag or bag is a file format in ROS for storing ROS message data. These bags are often created by subscribing to one or more ROS topics, and storing the received message data in an efficient file structure. MATLAB® can read these rosbag files and help with filtering and extracting message data. The following sections detail the structure of rosbags in MATLAB® and the workflow for extracting data from them.

MATLAB rosbag Structure

When accessing rosbag log files, call `rosbag` and specify the file path to the object. MATLAB then creates a `BagSelection` object that contains an index of all the messages from the rosbag.

The `BagSelection` object has the following properties related to the rosbag:

- `FilePath`: a character vector of the absolute path to the rosbag file.
- `StartTime`: a scalar indicating the time the first message was recorded
- `EndTime`: a scalar indicating the time the last message was recorded
- `NumMessages`: a scalar indicating how many messages are contained in the file
- `AvailableTopics`: a list of what topic and message types were recorded in the bag. This is stored as table data that lists the number of messages, message type, and message definition for each topic. For more information on table data types, see Access Data in Tables. Here is an example output of this table:

```
ans =

                    NumMessages        MessageType          MessageDefinition
                    ___________    ___________________    ____________________

    /clock              12001      rosgraph_msgs/Clock    [1x185  char]
    /gazebo/link_states 11999      gazebo_msgs/LinkStates [1x1247 char]
    /odom               11998      nav_msgs/Odometry      [1x2918 char]
    /scan                 965      sensor_msgs/LaserScan  [1x2123 char]
```

- `MessageList`: a list of every message in the bag with rows sorted by time stamp of when the message was recorded. This list can be indexed and you can select a portion of the list this way. Calling `select` allows you to select subsets based on time stamp, topic or message type.

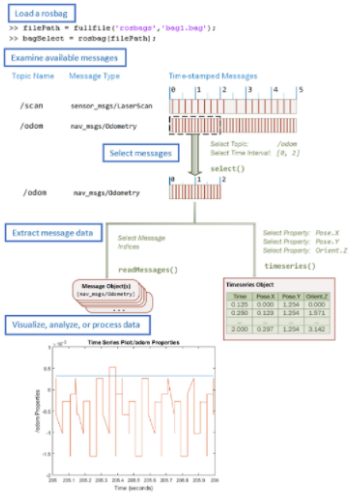
Also, note that the `BagSelection` object contains an index for all the messages. However, you must still use functions to extract the data. For extracting this information, see `readMessages` for getting messages based on indices as a cell array or see `timeseries` for reading the data of specified properties as a time series.

Workflow for rosbag Selection

When working with rosbags, there is a general procedure of how you should extract data.

- **Load a rosbag**: Call `rosbag` and the file path to load file and create `BagSelection`.
- **Examine available messages**: Examine `BagSelection` properties (`AvailableTopics`, `NumMessages`, `StartTime`, `EndTime`, and `MessageList`) to determine how to select a subset of messages for analysis.
- **Select messages**: Call `select` to create a selection of messages based on your desired properties.
- **Extract message data**: Call `readMessages` or `timeseries` to get message data as either a cell array or time series data structure.
- **Visualize, analyze or process data**: Use the extracted data for your specific application. You can plot data or develop algorithms to process data.

The following figure also shows the workflow.



Limitations

There are a few limitations in the rosbag support within MATLAB:

- MATLAB can only parse uncompressed rosbags. See the ROS Wiki for a tool to decompress a compressed rosbag.
- Only rosbags in the v2.0 format are supported. See the ROS Wiki for more information on different bag formats
- The file path to the rosbag must always be accessible. Because the message selection process does not retrieve any data, the file needs to be available for reading when the message data is accessed.

See Also

`BagSelection` | `readMessages` | `rosbag`