Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ingeniería Eléctrica
Fondef IT16M10012

Documentación Libería MPC

Fondef IT16M10012 - MPC

23 de octubre de 2018

Pablo Diaz

---

1. Dependencias

Las dependencias externas de la librería son las siguientes:

- apscheduler
- numpy
- pandas
- cassandra

También se utiliza la librería sys para especificar paths de archivos.

2. Estructura Librería

El paquete completo de MPC se encuentra en una librería llamada *mpc_thickener*. Esta contiene a su vez las siguientes librerías:

2.1. Socket de Cassandra

El módulo se llama `rwCassandra.py`. Contiene el *socket* que escribe y lee de cassandra. La parte principal es la clase *MyCassandra* que cumples dos funciones fundamentalmente.

En primer lugar, actúa como wrapper del paquete de cassandra completo y está especialmente diseñado para su uso en aplicaciones del Fondef. La idea es que tiene métodos especialmente útiles para escribir y leer en las tablas guardadas en Cassandra. En particular, contiene métodos para escribir y leer de forma síncrona, especialmente útiles para operación en línea.

Al instanciar un objeto, el método `__init__` requiere:

- **cassandra_dict**: Diccionario con IP, keyspace y nombre de tablas de lectura y escritura.
- **sensor_dict**: Diccionario con dos keys - una "in" y otra "out" - que contienen a su vez un diccionario con los tags de los sensores de lectura y escritura respectivamente. Cada uno de estos diccionarios tiene como llave el *TAG* del sensor (como está guardado en Cassandra) y devuelve por key el nombre común y la unidad. Por ejemplo, Conc. Sólidos Entrada, %.
- **sample_times**: Arreglo que tiene los tiempos de muestreo de escritura y de lectura. En teoría pueden ser distintos.

Idealmente, cualquier otra librería que use este socket debiese instanciarlo como un atributo (self.cassandraSocket).

La segunda función tiene que ver con levantar excepciones - a raíz de excepciones levantadas por Cassandra - para así poder controlar el flujo del programa principal.

2.2. Controlador Predictivo

El módulo denominado `mlMPC` es la librería principal del paquete. Invoca a todos los otros módulos.

En este *release*, su única función es **tomar datos leídos, manipularlos aleatoriamente y escribirlos en la función de salida**. La idea es simplemente detectar el flujo de los datos a través del controlador y revisar que las funciones de lectura/escritura sean adecuadas y estables.

Al instanciar un objeto, el método `__init__` requiere:

- **chosen_obj_fun**: String que contiene palabras claves para decidir la función objetivo.
- **chosen_model**: String que selecciona de todos los modelos predictivos disponibles uno para ser usado por el MPC.
- **predictive_parameters**: Diccionario que contiene ciertos parámetros para el funcionamiento del MPC. Actualmente, se utiliza `back_samples`, que dice cuántas muestras hacia atrás debe considerar el MPC, y `tau_C` que especifica cada cuanto actúa el controlador.

El método principal del módulo es `execute_mpc`, que funciona en tres pasos fundamentales:

1. Lee los datos según el tiempo de muestreo y los preprocesa.
2. Resuelve el problema de control predictivo para el modelo de predicción especificado.
3. Escribe el resultado de la operación en el tag de salida respectivo.

En un futuro, se planea añadir funciones de update de los parámetros del controlador (límites de variables, pesos de la función objetivo, etc).

Además, posee una función para reconectar el socket de cassandra en caso de alguna desconexión.

3. Archivo principal

El `main` contempla principalmente la instanciación y especificación de todos los diccionarios y listas para el funcionamiento de los módulos.

El paquete apscheduler se encarga del funcionamiento del módulo. A través de timers va ejecutando cada `tau_C` el programa principal mediante el método execute descrito en la sección 2.2.

Por el momento, el thread principal del programa se mantiene vivo a través de un time.sleep. En un futuro, esto debiese corresponder o bien a la aplicación del MPC o a otro proceso que mantenga el main ejecutándose.

El release actual cuenta con protecciones frente a desconexiones de Cassandra principalmente frente a errores de desconexión del servidor (*NoHostAvailable*) y errores del tipo de datos en la lectura (*TypeError*). En un futuro, y a medida que vayan surgiendo complicaciones será necesario agregar más (por ejemplo, timeouts).