数値解析ソフトウェア GNU Octave／MATLAB と MADOCA 制御システムとの連携

BUILDING MADOCA INTERFACE LIBRARIES FOR NUMERICAL ANALYSIS SOFTWARE GNU OCTAVE AND MATLAB

石塚規友紀 [A)], 増田剛正[#, B)]
Miyuki Ishizuka[A)], Takemasa Masuda[#, B)]
[A)] RIKEN SPring-8 Center
[B)] Japan Synchrotron Radiation Research Institute (JASRI)

Abstract

The control software framework named MADOCA (Message And Database Oriented Control Architecture) is adopted in the SPring-8 accelerators control system. In the MADOCA control system, widely-used numerical analysis software such as MATLAB and GNU Octave has been unavailable so far. Therefore, accelerator physicists and operators have to make numerical code for the dedicated analysis by ourselves, or to use numerical analysis software outside of the control system with dumped files from the logging database and then to operate the accelerators according to the results. In order to improve their convenience and to save their time and effort, we have newly built MADOCA interface libraries both in Oct-file format and in MEX format for GNU Octave to access MADOCA control system and database. Consequently, GNU Octave is available as a client for the MADOCA control system. We will report on the development of interface libraries mainly for GNU Octave to cooperate with MADOCA control system.

1. はじめに

SPring-8 加速器制御系では、MADOCA(Message And Database Oriented Control Architecture)と呼ばれる制御フレームワークを使用している[1][2]。MADOCA では機器を抽象化し、機器に対する制御コマンドは英語の構文になぞらえた S/V/O/C メッセージを送受信することによって行われる。また非常に強力なデータベース機能が特徴であり[3]、蓄積リングの COD 測定データをはじめとした大量のログデータがデータベースに蓄積され、運転や解析などに活用されている。

COD 補正など、大量に蓄積されたログデータを解析し、その結果をもとに機器に対する制御を行う場合、一般的には MATLAB[4]などの数値解析ソフトウェアをクライアントとして利用することが多い。しかしながら MADOCA 制御システムでは、これまでこうした数値解析ソフトウェアとの連携が全く取れていなかった。したがって、解析用の演算コードを C 言語等で自前に作成して制御用の GUI アプリケーションに埋め込むか、あるいは MADOCA のデータベースに蓄積されたデータをファイルにダンプして解析ソフトウェアにオフラインで取り込み、その解析結果をやはりファイル等を介して制御システムに反映させる必要があった。当然ながらこうした方法では、コーディング、およびそのテストに要する時間や手間、ファイル等を介してオフラインの数値解析ソフトウェアとやりとりを行う手間が掛かり、利用者にとっての利便性に著しく欠ける。

こうした状況を改善するため、数値解析ソフトウェアと MADOCA との連携を実現することとした。数値解析ソフトウェアとして、先ずは SPring-8 内でも利用者の多い MATLAB をターゲットとして整備することとした。そのための導入として、MATLAB との高い互換性をもつフリーの数値解析ソフトウェアである GNU Octave[5]を最初のターゲットとして MADOCA、より具体的には MADOCA II との連携を実現することとした。

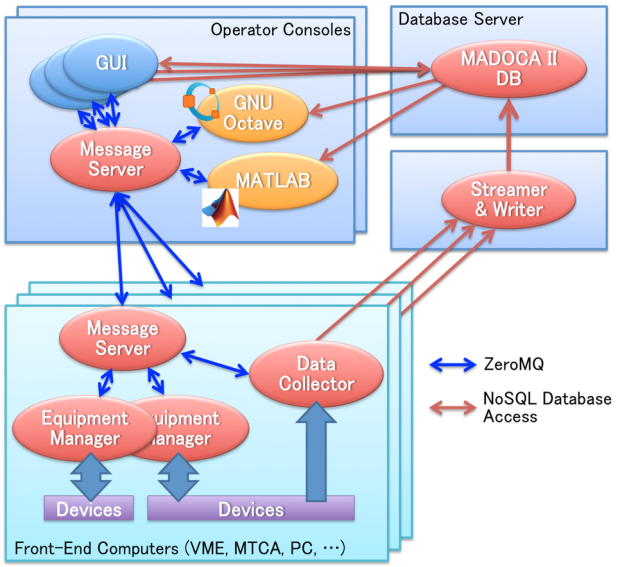

Figure 1: Schematic view of using GNU Octave and MATLAB as clients in the MADOCA control system.

2. 整備の指針

GNU Octave/MATLAB を MADOCA 制御システムのクライアントとして利用出来るようにする（Figure 1）ために、以下のような整備の指針を立てた。

- GNU Octave/MATLAB から直接機器の制御が行えること。具体的には、GNU Octave/MATLAB スクリプトから MADOCA 制御システムの Message Server(MS)に対して制御コマンドの送受信が行えること。
- GNU Octave/MATLAB のスクリプトから MADOCA

masuda@spring8.or.jp

データベースへアクセスして、データの読み出しが行えること。

- 上記 MS およびデータベースへのアクセスについては、MADOCA 制御システムで提供している API 関数と同等の API 関数を GNU Octave および MATLAB スクリプトに提供すること。

MADOCA で正式に提供している C 言語の API 関数を用いて MS とのやりとりを行う場合、MS との接続の確立(ms_open())→必要なオプションの設定(ms_option())→制御コマンドの送信(ms_client_send())→結果の受信(ms_recv_with_msg_id())→接続のクローズ(ms_close())という手順を踏む。必要に応じて制御コマンドの送信、受信の部分を繰り返すことになるわけだが、GUN Octave および MATLAB スクリプトにおいても同様の手順で MS とのやりとりが行えるよう API 関数を提供することとした。データベースアクセス関数についても同様に、GUI アプリケーション用に提供している C 言語用の API 関数と同様の API 関数を GUN Octave および MATLAB スクリプト用に提供することとした。

3. 実装

3.1 実装の方針

GNU Octave/MATLAB では、C++言語で書かれた外部関数を動的にロードしてスクリプト言語から利用することが出来る。通常の加速器の運転では、正式に MADOCA で提供している C 言語ベースの API 関数群をライブラリ化して GUI 等のアプリケーション内で呼び出して使用しているので、これら C 言語 API 関数群をそのまま動的にロードして使用する実装方法を採用することとした。動的にロードするためには、C++言語で記述された簡単な Wrapper 関数を用意する必要がある。その Wrapper 関数から MADOCA C 言語 API 関数を呼び出すよう実装することとした。

GNU Octave/MATLAB スクリプトから MADOCA API 関数を個別に呼び出せるようにするためには、C 言語 API 関数と 1 対 1 に対応する GNU Octave/MATLAB 用 API 関数を用意する必要がある。しかしながら、単純に C 言語 API 関数と1対1に対応する Wrapper 関数を用意する方法では、MS やデータベースのオープン時に取得した識別子を次の API 関数に引き渡すことが出来ず、メッセージの送受信やデータベースからのデータ取得を行うことが出来ない。そのため、これらの識別子を関数内部に保持出来るよう、実際に MADOCA の C 言語 API 関数を呼び出すための共通関数を用意して、GNU Octave/MATLAB 用の個別 API 関数は全てこの共通関数を呼び出すようにしてこの問題を解決することとした。実装のイメージを Figure 2 に示す。

実装にあたっては、最初のターゲットである GNU Octave で動作確認を行うために、先ず Octave ネイティブのファイルフォーマットである Oct file 形式での実装を行うこととした。具体的には、mkoctfile コマンドを用いて C++言語で記述された wrapper 関数を Oct file 形式にコンパイルすることで作成される。

Oct file 形式での実装が成功した場合、続いて GNU Octave および MATLAB の両方で動作する MEX 形式での実装を行うこととした。

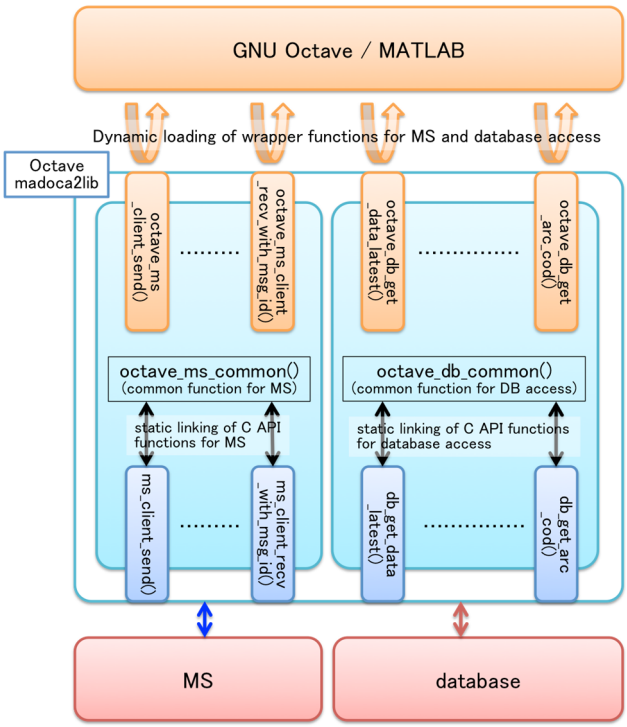

Figure 2: Schematic view of wrapper functions for GNU Octave to access MS and database by statically linking of MADOCA API functions in C.

3.2 Oct file 形式での C++ wrapper 関数の実装

C 言語と Oct file 形式の C++ wrapper 関数では引数や戻り値の構造が異なる。例えば C 言語ではポインター変数を介してデータの受け取りを行うことが出来るが、Oct file 形式ではそれができない。そのような場合、Oct file 形式ではベクタ型の関数の戻り値に含めることとした。

例えば、MS にメッセージを送信する C 言語 API 関数 ms_client_send()の仕様は Table 1 に示す通りで、送信するメッセージを msgusr->svoc にセットして関数をコールすると、メッセージ ID が msgusr->msg_id に戻される。戻り値には関数実行の結果が返される。

Table 1: Specification of the ms_client_send Function

ret = ms_client_send(*msgusr)	
argument :	Msgusr *msgusr
	typedef struct {
	….
	char svoc[256]; *// IN: Message Text*
	….
	int64_t msg_id; *// OUT: Message ID*
	….
	} Msgusr;
return :	int ret *// return ; 0 success, < 0 fail*

これに対応する Oct file 形式の API 関数 octave_ms_client_send()は Table 2 に示す通りの仕様とし

た。メッセージ ID は実行結果とともに関数の戻り値に返す仕様とした。

Table 2: Specification of the octave_ms_client_send Function

[ret, msg_id] = octave_ms_client_send(svoc)	
argument :	string svoc *// IN: Message Text*
return :	int ret *// result : 0 success, < 0 fail*
	int msg_id *// Message id*

データベースからデータを取得する C 言語 API 関数 db_get_data_latest()（Table 3）は、db_siglist_add()関数で予め登録した信号全ての最新データを取得する関数である。

Table 3: Specification of the db_get_data_latest Function

ret = db_get_data_latest (*out)	
argument :	M2dbreply *out
	typedef struct {
	….
	size_t num_signal; *// OUT: number of signals*
	M2dbdata *signals; *// OUT: pointer to start address of M2dbdata array*
	….
	} M2dbreply;
	typedef struct {
	char *name; *// OUT : signal name*
	size_t num_data; *// OUT : number of data*
	Logdata_t *data; *// OUT : pointer to start address of Logdata_t array*
	} M2dbdata;
	typedef struct {
	int err; *// OUT: error code*
	Timestamp_t time; *// OUT:time stamp*
	uint64_t tag; *// OUT: tag number*
	uint8_t type; *// OUT: data type*
	int idat; *// OUT: INT data*
	double fdat; *// OUT: DOUBLE data*
	….
	} Logdata_t;
	typdef struct {
	int64_t tv_sec; // second
	int64_t tv_nsec; // nanosecond
	} Timestamp_t;
return :	int ret *// return ; 0 success, < 0 fail*

Oct file 形式では、これに対応する関数を2つの関数に分離した。octave_db_siglist_add()関数で登録した全ての信号の最新データをデータベースから取得するために、まず octave_db_get_data_latest()（Table 4）を呼び出して読み出し対象の信号数を得たのち、octave_db_data_retrieve()関数（Table 5）を用いて、ある一つの信号を指定して最新データを読み出す。

Table 6 に今回整備を行った API 関数の一覧を示す。1列目に C 言語 API 関数、2列目にそれに対応する Oct File 形式の API 関数を示す。これらの API 関数が GNU Octave 上で全て問題なく動作することを確認した。

Table 4: Specification of the octave_db_get_data_latest Function

[ret, n] = octave_db_get_data_latest()	
return :	int ret *// result : 0 success, < 0 fail*
	int n *// number of signals*

Table 5: Specification of the octave_db_get_data_retrieve Function

[ret, num, err, tv_sec, tv_nsec, type, idat, fdat, tag] = octave_db_data_retrieve(n)	
argument :	int n *// IN: Specify the data to be read*
return :	int ret *// result : 0 success, < 0 fail*
	int num *// number of data*
	int32 err(num) *// error code (array)*
	uint64 tv_sec(num) *// time stamp second (array)*
	uint64 tv_nsec(num) *// time stamp nanosecond (array)*
	uint8 type(num) *// data type (array)*
	int32 idat(num) *// INT data (array)*
	double fdat(num) *// DOUBLE data (array)*
	uint64 tag(num) *// tag (array)*

3.3 MEX 形式での wrapper 関数の実装

Oct file 形式の API 関数と同様の実装方法で MEX 形式の API 関数を整備した（Table 6 3 列目）。具体的には、C++ wrapper 関数は共通ファイルとし、実装が異なる部分については mkoctfile コマンドでのコンパイル時のマクロ定義を変えることで対応した。

MEX 形式の API 関数に GNU Octave 上で問題なく動作することを確認した。

Table 6: A Main List of the Building MADOCA Interface API Functions for GNU Octave

API functions in C	API functions in Oct files	API functions in MEX	Purpose
ms_open()	octave_ms_open()	mex_ms_open()	Open MS
ms_option()	octave_ms_option()	mex_ms_option()	Set options to MS
ms_client_send()	octave_ms_send()	mex_ms_send()	Send a message
ms_client_recv_with_msg_id()	octave_ms_client_recv_with_msg_id()	mex_ms_client_recv_with_msg_id()	Receive a reply message by specifying message ID
ms_close()	octave_ms_close()	mex_ms_close()	Close MS
db_init_nosql()	octave_db_init_nosql()	mex_db_init_nosql()	Open No-SQL DB
db_end_nosql()	octave_db_end_nosql()	mex_db_end_nosql()	Close No-SQL DB
db_siglist_add()	octave_db_siglist_add()	mex_db_siglist_add()	Add new signals to siglist
db_siglist_reset()	octave_db_siglist_reset()	mex_db_siglist_reset()	Reset the siglist
db_get_data_by_timestamp()	octave_db_get_data_by_timestamp()	mex_db_get_data_by_timestamp()	Get data by timestamp.
db_get_data_latest()	octave_db_get_data_latest()	mex_db_get_data_latest()	Get the latest data
db_get_data_by_size()	octave_db_get_data_by_size()	mex_db_get_data_by_size()	Get data by number of data
----------	octave_db_data_retrieve()	mex_db_data_retrieve()	Retrieve data
db_get_arc_cod()	octave_db_get_arc_cod()	mex_db_get_arc_cod()	Get COD (0.1Hz)
db_get_nosql_cod()	octave_db_get_nosql_cod()	mex_db_get_nosql_cod()	Get COD (10Hz)
----------	octave_db_cod_data_retrieve()	mex_db_cod_data_retrieve()	Retrieve COD data
db_get_arc_cod_v()	octave_db_get_arc_cod_v()	mex_db_get_arc_cod_v()	Get COD with voltages (0.1Hz)
db_get_nosql_cod_v()	octave_db_get_nosql_cod_v()	mex_db_get_nosql_cod_v()	Get COD with voltages (10Hz)
----------	octave_db_cod_data_retrieve_v()	mex_db_cod_data_retrieve_v()	Retrieve COD data with voltages

4. 環境設定

4.1 GNU Octave のインストール

実際に SPring-8 加速器制御系のオペレータコンソール上で利用出来るよう、GNU Octave をインストールした。オペレータコンソールでは SUSE Enterprise Desktop [6] 11 SP3 を使用しているため、GNU Octave のソースコードをダウンロードして我々でビルドを行なった。GNU Octave のバージョンは 3.8.2 を選択した。

ビルドした GNU Octave はファイルサーバー上に置き、各々のオペレータコンソールは、それを/usr/local/octave として NFS マウントするようにした。

4.2 MADOCA インターフェースライブラリのインストール

C++ wrapper 関数を mkoctfile コマンドでコンパイルし、既存 C 言語 MADOCA インターフェース関数と静的にリンクした Oct file 形式および MEX 形式の API 関数群（Octave madoca2lib とする）をファイルサーバー上に置き、各々のオペレータコンソールはそれを /prj/opt/octave/madoca2lib として NFS マウントするようにした。

4.3 GNU Octave 起動スクリプトの設定

GNU Octave の起動時に、4.2 の Octave madoca2lib が必ずロードされるよう、GNU Octave 起動用スクリプト run-octave 内の LOADPATH 環境変数設定に /prj/opt/octave/madoca2lib を加えた。

4.4 Octave-Forge パッケージのインストールおよび設定

Octave-Forge[7]のサイトから、利用頻度の高そうなパッケージを選択してダウンロードしてインストールした。インストールしたパッケージは以下の通り。

- signal (version 1.3.2)
  Signal processing tools, including filtering, windowing and display functions.
- splines (version 1.3.2)
  Additional spline functions.
- Statistics (version 1.2.4)
  Additional statistics functions for Octave.
- optim (version 1.5.2)
  Non-linear optimization toolkit.

これらのパッケージをインストールする際、control(version 3.0.0)、io(version 2.4.7)、struct(1.0.14)の3パッケージが依存関係上必要であったため一緒にイン

ストールした。インストールしたパッケージがシステム内でグローバルに使用され、かつ Octave 起動時に自動的にロードされるようにするためには、pkg コマンドでパッケージをインストールする際に–global および–auto オプションを追加すれば良い。

5. まとめ

SPring-8 加速器制御系において、これまで連携が取れていなかった数値解析ソフトウェアのうち GNU Octave と MADOCA との連携を実現した。具体的には MADOCA 制御システムの MS との制御メッセージの送受信、MADOCA データベースからのデータの読み出しが行えるようになった。これで、オペレータコンソール上で動作する他の GUI アプリケーション同様、GNU Octave を MADOCA 制御システムのクライアントとして直接利用できるようになった。従来、GUI アプリケーション内において自力で解析コードを作成したり、ファイル等を経由してオフラインで数値解析ソフトウェアを使用したりする手間が省け、利用者の利便性が向上した。GNU Octave のスクリプト言語を用いて、コンパイルをすることなく、その場で即座に制御プログラム、解析プログラムが組めるようになり、加速器のマシンスタディなどに非常に有効である。

今後、加速器制御系に MATLAB をインストールし、MEX 形式の madoca2lib の動作確認を行い、それらが利用できるように整備を進める予定である。

謝辞

本ソフトウェアの製作ならびに本稿のまとめにあたっては、エム・アイ・システムズ株式会社池田孝利氏に数多くのご助言を頂きました。ここに感謝の意を表します。

参考文献

[1] R. Tanaka *et al*., “The first operation of control system at the SPring-8 storage ring”, Proceedings of ICALEPCS1997, Beijing, China, 1997, p.1.
[2] T. Matsumoto *et al*., “Next-Generation MADOCA SPring-8 Control Framework”, Proceedings of ICALEPCS2013, San Francisco, California, USA, 2013, p.944.
[3] M. Kago *et al*., “Development of a Scalable and Flexible Data Logging System Using NoSQL Databases”, Proceedings of ICALEPCS2013, San Francisco, California, USA, 2013, p.532.
[4] https://jp.mathworks.com/products/matlab.html
[5] https://www.gnu.org/software/octave
[6] https://www.suse.com/ja-jp/products/desktop
[7] https://octave.sourceforge.io/index.php