Computer Science and Application 计算机科学与应用, 2021, 11(11), 2652-2661
Published Online November 2021 in Hans. http://www.hanspub.org/journal/csa
https://doi.org/10.12677/csa.2021.1111269

Hans汉斯

C程序调用MATLAB算法的方法与实现

李靖宇，刘　阳，马晓波

湖南省可孚芯驰医疗科技有限公司，湖南 长沙

收稿日期：2021年10月5日；录用日期：2021年11月4日；发布日期：2021年11月11日

摘　要

MATLAB是一款广泛应用于科学计算领域的数学计算软件，非常适合信号处理、算法研究、数据仿真等工作，但是因其代码必须运行在MATLAB环境下，而实际应用中常常需要脱离MATLAB环境，用其它软件来调用MATLAB代码，所以带来了混合编程的需要。本文针对实际项目需求，讨论了用C程序调用MATLAB代码的几种方法，经过实践，最终选择了MATLAB代码转C动态链接库作为本地调用方法，以及借助MATLAB Production Server及其C客户端作为远程调用方法，该结果已经被用于产品研发中，达到了预期的效果。

关键词

MATLAB，MATLAB Production Server，混合编程，C语言

Method and Realization of Calling MATLAB Algorithm by C Program

Jingyu Li, Yang Liu, Xiaobo Ma

Hunan Kefuxinchi Medical Technology Limited Company, Changsha Hunan

Received: Oct. 5th, 2021; accepted: Nov. 4th, 2021; published: Nov. 11th, 2021

Abstract

MATLAB is a mathematical calculation software widely used in the field of scientific computing, which is very suitable for signal processing, algorithm study, data simulation and other works. However, its code must run in MATLAB environment, but in practical application, it often needs to be divorced from MATLAB environment and use other software to call MATLAB code, which brings the need of mixed programming. According to the actual project requirements, this article discusses several methods of calling MATLAB code with C program. After practice, we finally choose

文章引用: 李靖宇, 刘阳, 马晓波. C程序调用MATLAB算法的方法与实现[J]. 计算机科学与应用, 2021, 11(11): 2652-2661.
DOI: 10.12677/csa.2021.1111269

MATLAB code to C dynamic link library as the local call method, and use MATLAB Production Server with its C client as the remote call method. The results have been used in product development and achieved the expected effect.

Keywords

MATLAB, MATLAB Production Server, Mix Programming, C

1. 引言

MATLAB 是一款功能强大的数学计算软件，它在图像处理、数据分析、控制系统、深度学习和信号处理等领域应用颇广[1]。MATLAB 的基本数据单位是矩阵，这一特性使得它在解决许多数学和工程问题时要比其他语言更为简捷[2]，因此，本公司在开发自动心电分析软件包时，选择了 MATLAB 作为研发心电分析算法的工具，并编写了 autoECG 软件包，用美国麻省理工大学心电数据库的不同数据对该软件进行检验，部分输出结果如图 1 所示(共输出 215 条诊断结果及诊断时间)。

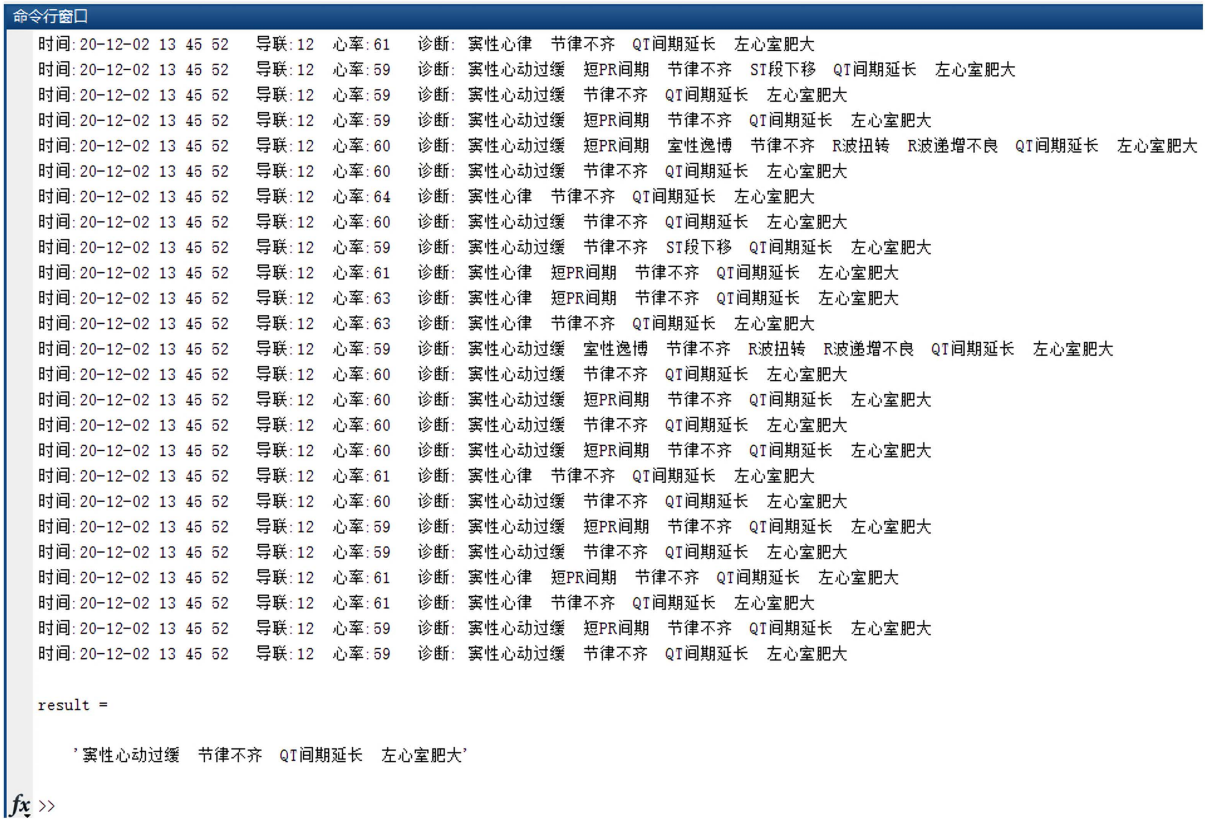

Figure 1. Some results of running autoECG software package under MATLAB
图 1. MATLAB 下 autoECG 软件包运行部分结果

由于用 MATLAB 编写的代码不能脱离 MATLAB 环境独立运行，无法直接用于产品软件的发布，同时，考虑到本项目中其它软件均采用 C 语言进行开发，所以需要用 C 程序调用 MATLAB，由此产生了混合编程的需求[3] [4]。

MATLAB 与 C 混合编程方法主要有以下几种：MATLAB 代码转 C 源代码[5]、MATLAB 代码转 C 动态链接库[1] [2] [3] [4]、MATLAB 代码部署到 MATLAB Production Server (以下简称 MPS)并用 C 客户端远程调用、C 程序调用 MATLAB 引擎[6]。其中，C 程序调用 MATLAB 引擎方法由于其用户环境必须安装有 MATLAB，并不适合软件的发布，因而在前期调研时便舍弃了该方法。本文基于 autoECG 软件包对其余三种混合编程方法进行了实践，分析比较了它们各自的优缺点，讨论了它们在不同应用场景下的适用性[7]。

本文涉及到的软件如下：NetBeans IDE 12.0，MATLAB R2020a，GCC 8.3.1，OpenSSL 1.1.1c。运行环境为：CentOS 8.1.1911 (Core 版本，x86_64 架构)，Ubuntu 18.04 (x86_64 架构)。

2. MATLAB 转 C 源代码方法

利用 MATLAB Coder 工具将 MATLAB 代码自动转换为 C/C++源代码。此法经过笔者的实践证明实用性较差，原因是有些 MATLAB 系统函数不能直接转换成 C 源代码，如图 2 所示。

Errors

	Function	Line	Description
⚠	ECG	12	fullfile is not supported for code generation
⚠	ECG	14	fgetl is not supported for code generation
⚠	ECG	15	sscanf is not supported for code generation
⚠	ECG	20	fgetl is not supported for code generation
⚠	ECG	21	sscanf is not supported for code generation

Figure 2. Matlab coder cannot convert some functions
图 2. MATLAB Coder 无法转换部分函数

在实际应用时，MATLAB 代码中常常会包含大量的 MATLAB 系统函数，难以避免地会使用到无法自动转换成 C 源代码的 MATLAB 函数。由于该局限性，笔者在实际应用没有采用此法。

3. MATLAB 转 C 动态链接库方法

3.1. 方法流程

利用 MATLAB 自带的 Library Compiler 工具将 MATLAB 代码编译成 C 动态链接库，然后在 C 代码中调用生成的动态链接库。该方法生成的动态链接库依赖于 MCR (MATLAB Compiler Runtime，简称 MATLAB Runtime)，因此用户环境必须安装 MCR。MATLAB 转 C 动态链接库方法流程如图 3 所示：首先准备好我们的 MATLAB 函数(图中以 ECGmainfcn.为例)，并配置 MATLAB Library Compiler 将 GCC 设置为 C 编译器，然后用 MATLAB Library Compiler 将 MATLAB 函数编译成 C 动态链接库和头文件。用户在安装了对应版本的 MATLAB Runtime 的环境下即可在程序中调用上面得到的动态链接库，进行混合编程[8]。

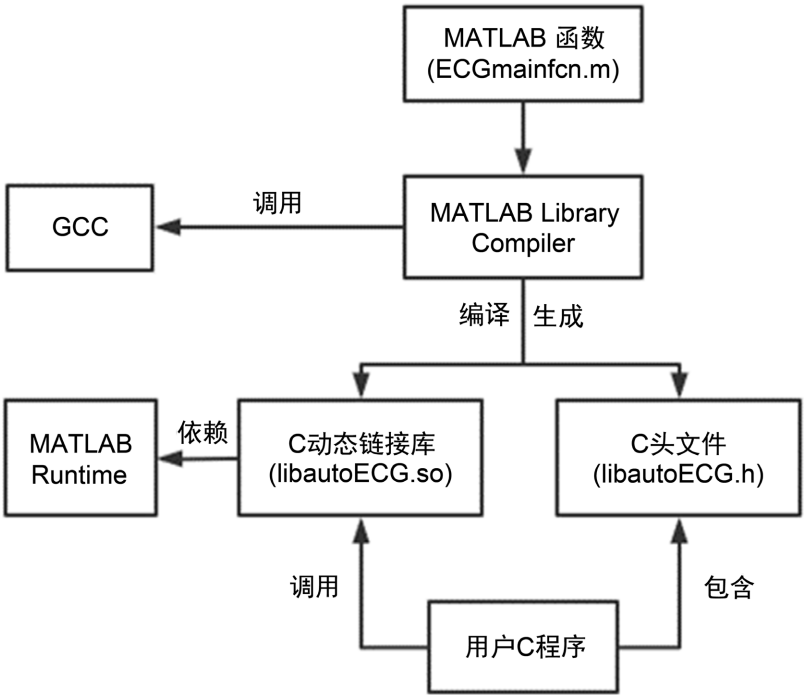

Figure 3. Method and flow of MATLAB to C dynamic link library
图 3. MATLAB 转 C 动态链接库方法流程

3.2. 实现过程

在 CentOS8 系统上，此法实现过程可分为以下三步：

1) 将 MATLAB 函数编译成动态链接库

首先需要指定 mcc 使用的编译器。在 MATLAB 命令行使用 mbuild-setup 命令来选择 gcc 作为 C 编译器。然后通过如下 mcc 命令将 autoECG 软件包编译为动态链接库，库名为 libautoECG.so，输出文件存放在同级目录 project 下：

mcc -W lib:libautoECG -T link:libautoECG.m -d ./project/

2) 配置用户环境

首先配置 MCR 环境。下载 MCR 安装包并安装，然后将 MCR 安装目录下的 externs 目录内全部头文件拷贝到/usr/local/include 目录下。接着根据 1)中输出目录 project 下的 readme 文件提示设置环境变量 XAPPLRESDIR 和 LD_LIBRARY_PATH。注意：笔者的 XAPPLRESDIR 设置与 readme 文档所述不同：

XAPPLRESDIR=/usr/share/X11/app-defaults

并且笔者额外设置了一个环境变量 LIBRARY_PATH，以保证 C 代码能够编译成功：

LIBRARY_PATH=${LD_LIBRARY_PATH}

接下来修改/etc/ld.so.conf 文件，在末尾加一行：/usr/local/lib，并将输出目录 project 下的 autoECG.so 文件拷贝到/usr/local/lib 目录下。最后执行 ldconfig 命令，环境配置完成。

3) 在 C 代码中调用动态链接库

代码流程为：初始化 MATLAB 运行环境和函数、准备 MATLAB 函数入参、调用 MATLAB 函数、处理返回数据、终止 MATLAB 运行环境、释放资源、退出程序。

笔者采用 NetBeans IDE 作为开发环境，通过 ssh 连接至 Linux 主机来进行远程开发。在开始编码前需要将输出目录 project/中的头文件 autoECG.h 拷贝至项目头文件目录中，以在代码中对其进行引用。

在编译代码前需要添加链接库选项：

-lautoECG -lmwmclmcrrt

图 4 是 C 程序在 Linux 系统下调用 autoECG 软件包运行的结果(省略部分诊断结果输出)，程序输出和返回值与图 1 完全一致。为了便于比较不同方法下的运行性能，程序末尾还打印了初始化 MATLAB 运行环境、MATLAB 函数计算、终止 MATLAB 运行环境的耗时。

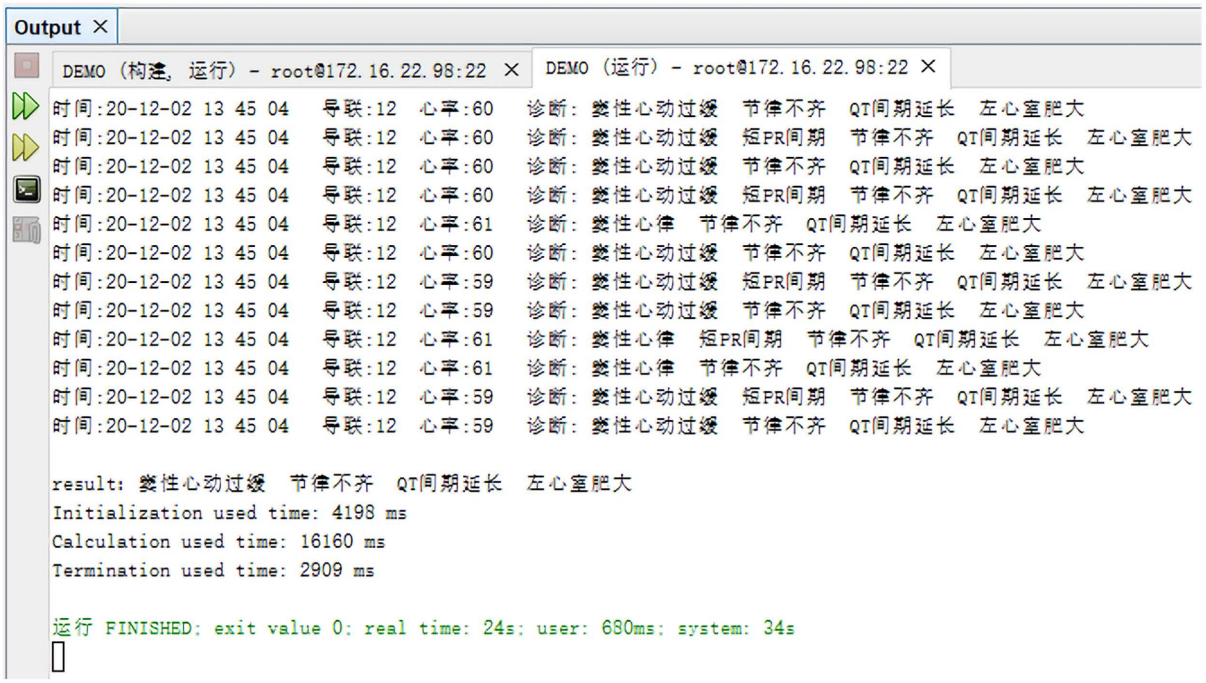

Figure 4. Program running results
图 4. 程序运行结果

3.3. 优缺点分析

此法的缺点在于用户环境依赖 MCR，安装 MCR 会占用一定大小的磁盘空间，间接增加了程序包的大小。同时，由图 4 运行结果可知，程序初始化过程耗时 4 秒左右，所需的时间成本较大。尽管初始化耗时较长，但在实际应用中可以通过控制程序初始化的次数来降低其时间成本。

此法的优点在于以简单有效的方式整合了 MATLAB 和 C 两种语言，充分发挥了两者的优势，应用空间十分广阔。而且将 MATLAB 函数打包编译成 C 动态链接库，使得修改算法实现只需替换动态链接库而无需修改 C 程序代码，程序开发非常灵活[9]。

4. MPS 结合 C 客户端方法

MPS 是 MATLAB 发布的一款适用于企业生产环境的服务器。任何自定义的 MATLAB 函数都可以部署于 MPS 之上，用户可以通过调用多种语言的接口(以 http/https 请求的形式)，或通过调用 RESTful API，来访问部署于 MPS 上的 MATLAB 函数。下面描述配置部署 MPS 以及通过 C 客户端来调用 MPS 上的函数的方法、实现及关键技术。

4.1. 服务器环境搭建

服务器上需要安装 MPS 和 MCR。注意，MCR 的版本一定要与 MATLAB 的版本保持一致，且服务

器搭载的 Linux 系统必须是符合 unix 系统标准的发行版，如笔者使用的 CentOS8。安装好 MPS 之后，添加${MPS 安装目录}/script 路径到环境变量 PATH 的末尾。

4.2. 管理和配置 MPS 实例

MPS 实例进程是以守护进程的形式运行于 Linux 系统上的服务器进程，它依赖于 MCR 和 MPS。一个 MPS 实例可以看成一台服务器，它能够与多个客户端相连，并行处理客户端的分析请求。一个 Linux 系统上可以同时存在多个 MPS 实例，它们在逻辑上相互独立，且可以使用不同版本的 MCR。每个 MPS 实例都会占用独立的端口，用于和客户端进行通讯。图 5 为 MPS 的部署架构示例图。

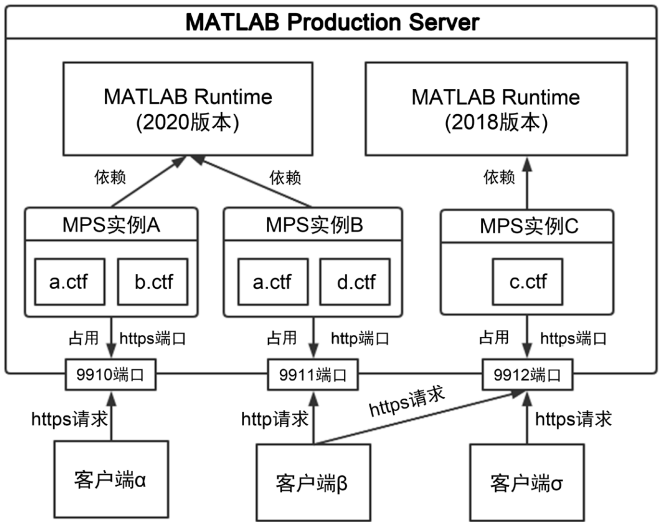

Figure 5. MPs deployment architecture example
图 5. MPS 部署架构示例

在创建 MPS 实例之前，我们需要通过 mps-setup 命令指定它所使用的 MCR 的路径，当系统上存在多个 MCR 或 MCR 没有按照默认路径安装时，这一步尤为关键。接下来，使用 mps-new 命令创建一个 MPS 实例。

由于 MPS 默认使用 http 协议，而这是一种不安全的应用层协议，如非特殊需求我们都应该关闭 MPS 实例的 http 端口，并启用 https 端口来与客户端进行更为安全的连接。具体方法是：通过修改 MPS 实例的配置文件(路径为${MPS 实例目录路径}/config/main_config)来启用 https 端口：

```
--https 9920#启用 https 端口
#启用客户端证书验证
--ssl-verify-peer-mode verify-peer-require-peer-cert
#服务器私钥路径
--x509-private-key ./x509/server.key
#服务器私钥密码文件路径
--x509-passphrase ./x509/pwd
#服务器证书链文件路径
--x509-cert-chain ./x509/cert_chain.pem
#服务器根 CA 证书路径
--x509-ca-file-store /etc/pki/CA/cacert.pem
```

服务器采用自签名的 CA 证书即可。使用 openssl 生成 CA 私钥、CA 证书、服务器私钥和客户端私钥，再用 CA 证书和私钥生成服务器证书和客户端证书。

配置完成之后，使用 mps-start 命令启动 MPS 实例，并用 mps-status 命令确认 MPS 实例是否成功启动。

4.3. 客户端环境配置

客户端程序运行于 Ubuntu (非 unix 标准)操作系统上。首先将服务器上${MPS 安装目录路径}/client/c 目录(以下简称 c 目录)拷贝至客户端操作系统上。接着添加${c 目录路径}/glnxa64/lib 路径到环境变量 LD_LIBRARY_PATH 和 LIBRARY_PATH 的末尾。

4.4. 部署 MATLAB 函数并调用

下面把 autoECG 转换成可部署文件并部署到 MPS 实例，再用 C 客户端调用 MPS 实例上部署的 MATLAB 函数。

首先在 MATLAB 命令行通过如下命令，将 autoECG 软件包转换为可部署文件(.ctf 后缀)，输出文件存放在同级目录 project 中：

mcc-W CTF:autoECG -U autoECG.m -d ./project/

然后将输出目录 project/中的 autoECG.ctf 文件拷贝至 MPS 实例目录下的 auto_deploy 目录中，MPS 实例会自动检测该目录下的.ctf 文件并将其部署。检查日志文件${MPS 实例目录路径}/log/main.log 来确认 MATLAB 函数已成功部署。

autoECG.ctf 部署成功后，编写客户端代码来连接 MPS 实例并调用该函数。将${c 目录路径}/include/mps/目录下的头文件拷贝至项目头文件目录。再将 4.2 节中生成的客户端证书和私钥以及 CA 证书拷贝至项目目录下，并将它们的路径设置到 mpsClientConfig 结构体中，如图 6 为笔者编写的设置函数。

```
int mps_https_conf_and_ini(mpsClientRuntime** mpsruntime,
        mpsClientConfig** mpsconfig, mpsClientContext** mpscontext)
{
    mpsStatus status;

    *mpsruntime = mpsInitializeEx(MPS_CLIENT_1_1);
    status = (*mpsruntime)->createConfig(mpsconfig);
    if (status == MPS_FAILURE)
    {
        printf("creatConfig failed\n");
        return -1;
    }

    (*mpsruntime)->setCAFile(*mpsconfig, CA_FILE_PATH);
    (*mpsruntime)->setRevocationListFile(*mpsconfig, "");
    (*mpsruntime)->setVerifyHost(*mpsconfig, true);
    (*mpsruntime)->setVerifyPeer(*mpsconfig, true);

    (*mpsruntime)->setClientCertFile(*mpsconfig, CLIENT_CRT_FILE_PATH);
    (*mpsruntime)->setPrivateKeyFile(*mpsconfig, PK_FILE_PATH);
    (*mpsruntime)->setPrivateKeyPasswd(*mpsconfig, PK_FILE_PWD_PATH);

    status = (*mpsruntime)->createContext(mpscontext, *mpsconfig);
    if (status == MPS_FAILURE)
    {
        printf("creatContext failed\n");
        return -2;
    }

    return 0;
}
```

Figure 6. Https setting function
图 6. Https 设置函数

接下来的代码流程与 3.2 节中步骤 3 的流程类似：准备 MATLAB 函数入参、连接 MPS 调用 MATLAB 函数、处理返回数据、释放资源、退出程序。笔者编写的代码实现了连续对 MPS 实例请求 autoECG 算法，并将每一次请求耗时和得到的返回值 result 打印输出。程序执行结果如图 7 所示，返回值 result 与图 1 相同，程序调用 autoECG 成功。

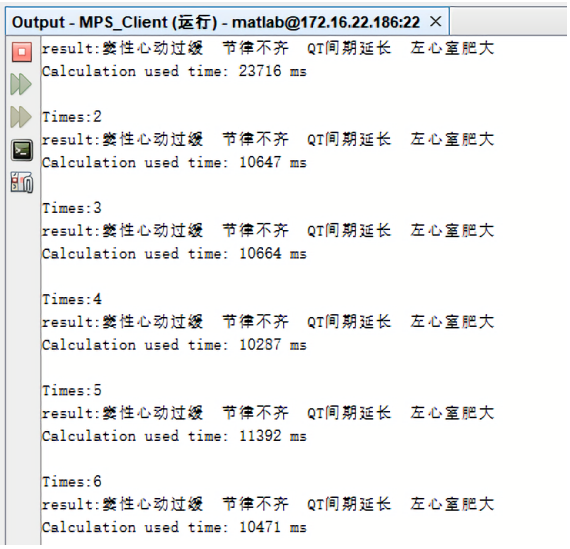

Figure 7. Client program execution results
图 7. 客户端程序执行结果

4.5. 优缺点分析

此法的缺点在于函数调用以网络请求的方式实现，对于单机软件或有网络条件限制的场景不适用。同时，由图 7 运行结果可看出，在 MPS 实例启动后的第一次请求耗时为正常情况下请求耗时的两倍多，这一现象会导致服务器初始化的时间成本增加。

此法的优点在于将 MATLAB 函数以服务的形式单独部署于服务器上，将算法实现与用户程序完全解耦；同时 MPS 的设计已经较为完善，它具有自动化部署、同时支持多个 MCR 版本、支持数据缓存、支持多处理器、支持低延迟并发请求、提供多种语言客户端及 RESTful API 等等强大的功能，且具有良好的可扩展性和安全机制，适用于企业级系统。采用此法可以减少大量开发人员的工作，避免“重复造轮子”。

5. MPS 结合 C 客户端方法与 MATLAB 转 C 动态链接库方法的比较

下面从空间成本、时间成本和限制条件三个方面对两种方法进行比较，以分析它们各自的适用场景。为表示方便，以下用方法一指代 MPS 结合 C 客户端方法，方法二指代 MATLAB 转 C 动态链接库方法。

5.1. 空间成本

对于方法一，需要从服务器端和客户端两方面来分析：服务器端需要安装 MCR、MPS，客户端则无需安装任何 MATLAB 环境。方法二要求用户环境安装有 MCR，这会占用较大的磁盘空间(MCR 大小为几百 Mb 到几 Gb 不等，取决于 MCR 版本等因素)。所以，针对用户程序来说，方法一的空间成本要比方法二小很多。

5.2. 时间成本

对于方法一，同样从服务器端和客户端两方面来分析：服务器端创建、配置、启动 MPS 实例以及初始化 MATLAB 函数需要一定的时间，客户端请求调用函数则无额外的时间成本。方法二中用户程序需要耗费一定时间来初始化和终止 MATLAB 运行环境和函数。同时，从计算速度的角度来说，方法一由服务器来进行计算，因此计算速度很大程度上取决于服务器配置和网络环境，而方法二则由用户程序来进行计算，计算速度主要取决于用户环境。对于用户计算机的计算能力较差的情况，方法一可以提供比方法二更稳定快速的算法调用。所以，方法一的时间成本比方法二要小很多。

5.3. 限制条件

方法一的限制条件是服务器与客户端之间必须网络畅通，如发生服务器宕机、客户端不能联网等情况，程序就无法调用算法。方法二的限制条件是用户环境依赖于 MCR，如果用户的计算机难以满足 MCR 的运行条件，程序就无法正常工作。

5.4. 适用场景

由以上分析可知，在大部分应用场景下，尤其是在企业级系统、科研系统等大型系统中，方法一要比方法二更适用；而在开发单机应用程序或网络条件受限等场景下，方法二比方法一更适用，如表 1 所示：

Table 1. Comparison of two hybrid programming methods
表 1. 两种混合编程方法的比较

方法	用户环境是否依赖 MCR	网络环境要求	时间成本	空间成本	适用场景
MPS 与 C 客户端结合	否	要求用户与服务器之间网络畅通	低	低	企业及系统、科研系统等大部分应用场景
MATLAB 转 C 动态链接库	是	无	高	高	单机应用程序、嵌入式系统等

6. 总结

MATLAB 作为世界上最强大的科学计算软件之一，广泛应用于科学研究和工程设计等领域[2]。本文主要研究了两种通过 C 程序调用 MATLAB 算法的方法，其中 MATLAB 转 C 语言动态链接库方法更适用于单机应用程序的开发，MPS 结合 C 客户端方法则更适用于大小型系统及应用程序的开发。两种方法都已被应用于本公司承担的“湖南创新型省份建设专项重点领域研发计划动态多参融合家庭智能诊断高端设备关键技术研究及产业化”项目的研发中。实践表明，通过将 MATLAB 与 C 语言有机结合，能够充分发挥两者的优势，大大扩展 MATLAB 的应用空间，提高程序设计和开发效率，具有很高的应用价值。

基金项目

“湖南创新型省份建设专项重点领域研发计划——动态多参融合家庭智能诊断高端设备关键技术研究及产业化”项目。

参考文献

[1] 李中科, 赵慧娟, 苏晓萍. 基于 OpenCV 的 C/C++和 Matlab 计算机视觉混合编程[J]. 计算机时代, 2018(7): 69-72.

[2] 亓雪冬, 李霞. MATLAB 与 C#混合编程方法研究[J]. 微型电脑应用, 2020, 36(4): 85-87.
[3] 胡其荣. 图形处理中 C 语言的综合应用策略[J]. 计算机产品与流通, 2020(11): 44.
[4] 王成杰, 汪晓斌, 廖定安. Matlab 与 VC++混合编程综述[J]. 电脑与信息技术, 2012, 20(5): 57-58+62.
[5] 周世钦, 王波涛. MATLAB 程序转 C 代码的方法研究[J]. 价值工程, 2018, 37(2): 182-185.
[6] 陈正伟. 使用 MATLAB 引擎库函数实现.NET 下调用 MATLAB [J]. 中国新通信, 2008(13): 14-19.
[7] 廖灿灿, 张树群, 雷兆宜. Matlab Coder 生成 C 代码的研究与应用[J]. 计算机与现代化, 2013(3): 175-178+183.
[8] 徐治. Visual C++调用 MATLAB 函数库的混合编程技术[J]. 软件, 2015, 36(2): 55-58.
[9] 潘大夫, 汪渤, 周志强. Matlab 与 C/C++混合编程技术研究[J]. 计算机工程与设计, 2009, 30(2): 465-468+521.