Lưu hành nội bộ

Luật viết code

MATLAB

Version 1.1

8-2014

# Các thay đổi

|  |  |  |  |
| --- | --- | --- | --- |
| Phiên bản | Ngày | Tác giả | Mô tả |
| 1.0 | 18.08.2014 | Dũng | Bắt đầu |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# Mục lục

[Các thay đổi 2](#_Toc396080140)

[Mục lục 3](#_Toc396080141)

[1. Một số luật chung 4](#_Toc396080142)

[2. Cách đặt tên 6](#_Toc396080143)

[2.1. Biến 6](#_Toc396080144)

[2.2. Hằng số, biến toàn cục, tên cấu trúc 6](#_Toc396080145)

[2.3. Hàm 7](#_Toc396080146)

[3. Cách thức viết code 9](#_Toc396080147)

[3.1. Tối ưu về tính toán 9](#_Toc396080148)

[3.2. Tối ưu về test, debug 9](#_Toc396080149)

[3.3. Tối ưu cho việc sử dụng kết quả 9](#_Toc396080150)

[3.4. Một số vấn đề khác 9](#_Toc396080151)

[4. Cấu trúc thư mục và file của chương trình xử lý tín hiệu cho viễn thông 11](#_Toc396080152)

[4.1. Cấu trúc thư mục 11](#_Toc396080153)

[4.2. Cấu trúc file TOP 11](#_Toc396080154)

[4.3. Các file trong thư mục kết quả (Results) 11](#_Toc396080155)

[5. File và cách tổ chức 12](#_Toc396080156)

# Một số luật chung

*Lý do cần tuân theo luật viết code:*

Trong môi trường cộng tác, ít khi phải tự viết code từ đầu mà phải tiếp nhận những đoạn code có sẵn từ người đi trước. Nếu cách viết code được chuẩn hóa thì người mới tham gia sẽ dễ tiếp thu. Một người có thể viết rất nhiều code cho nhiều dự án nên sẽ quên dần các đoạn code đã viết. Nếu cách viết rõ ràng, có cấu trúc thì khi sử dụng lại code, người lập trình sẽ dễ dàng nhớ lại hơn.

Viết code cần có cấu trúc rõ ràng, có đầy đủ chú thích để người đọc dễ dàng hiểu được ý tưởng.

*Martin Fowler: “Any fool can write code that a computer can understand. Good programmers write code that humans can understand.”*

*Kreitzberg and Shneiderman: “Programming can be fun, so can cryptography; however they should not be combined.”*

Có nhiều cách viết code cho một thuật toán, cố gắng sử dụng cách viết dễ hiểu nhất kể cả khi cách viết đó làm tăng thời gian xử lý một chút.

Đặt tên biến có cấu trúc và kiểu biến sẽ làm người lập trình dễ đọc và sửa lỗi.

Các tên cần có ý nghĩa và có thể tự giải thich ý nghĩa. Lúc này có thể không cần chú thích nữa. Chấp nhận đặt tên dài.

Viết code theo phong cách hướng đối tượng

Sử dụng tiếng Anh vì tiếng Việt dễ gây nhầm lẫn khi viết không dấu

Tránh viết tắt từ tiếng Anh: calculateSnr thay vì calcSnr

Lý do cần viết tài liệu

To be useful documentation should include a readable description of what the code is supposed to

do (Requirements), how it works (Design), which functions it depends on and how it is used by

other code (Interfaces), and how it is tested. For extra credit, the documentation can include a

discussion of alternative solutions and suggestions for extensions or maintenance.

Consider writing the documentation first.

Some programmers believe that the best approach is “Code first and answer questions later.”

Through experience most of us learn that developing a design and then implementing it leads to a

much more satisfactory result.

Development projects are almost never completed on schedule. If documentation and testing are

left for last, they will get cut short. Writing the documentation first assures that it gets done and

will probably reduce development time.

Changes.

The professional way to manage and document code changes is to use a source control tool. For

very simple projects, adding change history comments to the function files is certainly better than

nothing.

# Cách đặt tên

## Biến

Sử dụng cả chữ hoa chữ thường theo phương pháp CamelBack. Chữ đầu tiên viết thường.

VD: linearity, credibleThreat, qualityOfLife

Các chữ viết tắt cũng chỉ viết hoa chữ cái đầu. VD: testSNRValues 🡪 testSnrValues

Các biến cục bộ có thể viết ngắn để giữ cho thuật toán dễ đọc hơn.

Thông thường thuật toán matlab sẽ tuần theo một công thức trong bài báo nên ngoài cách dặt tên biến rõ ràng có thể đăt tên biến đúng như trong công thức. Vì vậy các biến chỉ số ( i, j, k, m, x,y,z) thường được để sẵn cho mục đích này. VD: không dùng I,j cho các biến chạy trong vòng for nếu như nó không xuất hiện trong công thức.

Tiếp đầu ngữ (prefix): Dùng để phân loại các biến một cách nhanh chóng. Chỉ dung với biến có tên dài

|  |  |  |
| --- | --- | --- |
| Tiếp đầu ngữ | Loại | Ví dụ |
| n | Số lượng | nFiles: số các file |
| m | Số lượng hàng trong các biến mảng | mRows: |
| i,j,k | Chỉ số trong mảng | iFile: chỉ số của file (không có số nhiều)  có thể viết fileNo |
| st | Biến có cấu trúc | stTree |
| f | Biến kiểu số thực/phức | fEqualizerOutput |
| b | Biến kiểu Boolean | bNoiseEnabled |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

Chú ý: cần có số nhiều cho các biến chỉ số lượng ( thêm ‘s’)

for iFile = 1:nFiles

:

end

Số phức: sử dụng 1i, 1j thay vì i,j.

Tránh đặt tên có tính phủ định: dùng isFound thay vì isNotFound

## Hằng số, biến toàn cục, tên cấu trúc

Hằng số và biến toàn cục, tên cấu trúc (struct) sẽ được viết hoa toàn bộ và dùng \_ để thay dấu cách

VD: MAX\_ITERATIONS, COLOR\_RED ,

Tên của hằng số có thể được chia thành các phần trong đó phần tiền tố (prefix) có giá trị giống nhau: VD:

COLOR\_RED, COLOR\_GREEN, COLOR\_BLUE

## Hàm

Mỗi hàm được ghi vào 1 file (trừ một số hàm cục bộ)

Tên hàm viết thường và cùng tên với file

getname(.), computetotalwidth(.)

There are two other function name conventions commonly used. Some people prefer to use

underscores in function names to enhance readability. Others use the naming convention

proposed here for variables.

Hàm có 1 biến đầu ra thì có thể đặt tên theo biến đầu ra đó

VD: mean() thay vì calculateMean()

Hàm không có đầu ra hoặc chỉ trả về handle thì cần có động từ giải thích xem hàm đó làm gì.

VD: plot(.)

Hàm bắt đầu bằng get/set thì thường để dành cho việc truy cập biến trong đối tượng.

Một số từ nên dùng: compute, find, initialize

Hay dùng cho hàm trả về biến Boolean: is,has,can, should

VD: iscomplete(.), hasLicense(.); canEvaluate(.); shouldSort(.);

Complement names should be used for complement operations.

Giảm độ phức tạp bằng cách sử dụng các từ đối xứng nhau:

get/set, add/remove, create/destroy, start/stop, insert/delete, increment/decrement, old/new, begin/end, first/last, up/down, min/max, next/previous, old/new, open/close, show/hide, suspend/resume, etc.

Thêm tiếp vĩ ngữ (suffix) là đơn vị để có thể làm rõ nghĩa hơn cho hàm

VD: incidentAngleRadians(), computeSnrDb()

Phần mào đầu của hàm (header) có cấu trúc sau:

Dòng 1: khai báo hàm, VD: function [X,Y] = abs\_error(A,B)

Dòng 2: (comment) : Tên người viết và thời gian

Dòng 3: Tên project

Dòng 4: Mục đích

Danh sách biến vào ra, mô tả và đơn vị.

Danh sách các hàm được gọi, nều không có thì ghi none.

Tránh sử dụng biến toàn cục (global)

Sử dụng biến cấu trúc để tránh gán các biến dài dòng

Hàm con: đặt ở cùng 1 file với hàm chính nếu chỉ có hàm chính sử dụng

Script test hàm: Mỗi hàm cần có 1 đoạn script test. (VD: TODO)

# Cách thức viết code

## Tối ưu về tính toán

Matlab tối ưu tính toán cho mảng theo chiều ngang, 1xN, nên đối với nhưng dữ liệu lớn (trên 1000 điểm) thì cần viết code theo kiểu này.

## Tối ưu về test, debug

Trong khi làm thí nghiệm các mô hình với matlab việc thay đổi thuật toán luôn luôn xảy ra. Vì vậy, khi thực hiện các khối xử lý, không đưa các khối đó vào hàm mà chỉ đưa khối đó vào một file và gọi. Như vậy, khi thay đổi nội dung khối đó không nhất thiết phải khởi tạo lại cả hệthống. Chỉ có những khối xử lý nào đã được test kỹ và ít thay đổi thì mới đưa vào hàm.

Ví dụ: file đầu tiên siso\_top chứa các khối: addAwgn và receiver thì 2 khối này sẽ được lưu trong 2 file .m cùng tên và không khai báo hàm. Các biến trong 2 file này cũng sẽ là các biến toàn cục và truy cập được khi ở trong siso\_top. Cách thức viết này tốt cho việc debug, sửa lỗi nhưng việc dùng các biến cũng cần cẩn thận hơn.

Các file (module) có cùng chức năng sẽ có các biến đầu vào , đầu ra, các biến quan trọng giống nhau để dễ so sánh.

Ví dụ: có nhiều phương pháp equalizer, các hàm này sẽ sử dụng chung đầu vào equalizerInput, tham số bộ lọc w, biến đầu ra w, sinr, isi…

## Tối ưu cho việc sử dụng kết quả

Để tránh mất dữ liệu đã tính toán khi mất điện và để dễ sử dụng, các kết quả tính toán quan trọng, tham số hệ thống… được lưu trong biến toàn cục sys (không viết hoa) và các thông tin này được lưu định kỳ sau mỗi lần thay đổi tham số. Dữ liệu trung gian do máy tính tạo ngẫu nhiên sẽ không được lưu.

Tên file này phụ thuộc vào hệ thống đang được thử nghiệm. Một số tham số được thể hiện ra tên file để về sau dễ so sánh.

## Một số vấn đề khác

* Loops

Khởi tạo ngay trước khi dùng (tăng tốc và tránh nhầm lẫn)

Chỉ dùng break, continue nếu cần thiết.

Thêm chú thích vào end của các vòng lặp, if, switch để dễ đọc hơn

* ĐIều kiện

Tránh dùng điều kiện phức tạp vì không dễ gỡ lỗi, sử dụng 1 biến trung gian

VD: if (value>=lowerLimit)&(value<=upperLimit)&~ismember(value,…valueArray)

Cần thay bằng

isValid = (value >= lowerLimit) & (value <= upperLimit);

isNew = ~ismember(value, valueArray);

if (isValid & isNew)

switch cần thêm otherwise để tránh lỗi.

Biến switch hay dùng bằng biến string vì dễ hiểu hơn biến int.

* Các số

Tránh các con số không rõ ràng. Nếu các con số được dung đi dung lại, cần khai báo hằng số

* Trình bày

Nội dung 1 dòng code cần được viết trong 80 ký tự. Nếu code dài hơn thì dung dấu … để xuống dòng tại những vị trí hợp lý: sau dấu phẩy, trước toán tử,

Dấu cách: nên sử dụng thêm để cách ly các toán tử

VD: a = b + c; sẽ tốt hơn a=b+c;

# Cấu trúc thư mục và file của chương trình xử lý tín hiệu cho viễn thông

## Cấu trúc thư mục

Thư mục gốc bao gồm:

File config.m : chứa các thông tin config cơ bản hệ thống

File \_top.m: chức các vòng lặp, các lời gọi đến các file khác

Thư mục channel: chứa các hàm, dữ liệu liên quan đến kênh

Thư mục transmitter: chứa các hàm xử lý tín hiệu tại phía phát

Thư mục receiver: chứa các hàm xử lý tín hiệu tại phía thu

Thư mục results : chứa kết quả thô (do copy bằng tay)

Thư mục common : chứa các file chứa hàm thông dụng

## Cấu trúc file TOP

TODO

## Các file trong thư mục kết quả (Results)

TODO

# Tài liệu tham khảo

[1].  **C++ Coding Standards**: 101 Rules, Guidelines, and Best Practices. read.pudn.com/downloads106/ebook/438804/CPP101.**pdf**

**[2]** **MATLAB Programming Style Guide** www.usna.edu/Users/oceano/barrett/SO414/**Style**\_**Guide**

[1