



# 模糊控制算法——C语言版本

此工程为C语言版本的模糊控制算法，我将在这里提供模糊控制算法的核心及相关的示例项目。

- 模糊控制算法——C语言版本
  - 1. 工程框架
    - 1.1 数据结构
      - 1.1.1 模板式单向链表
        - 谓词和事件回调函数指针
        - 默认谓词和回调函数
        - 链表操作API
        - 链表节点操作API
        - 参数获取API
      - 1.1.2 模糊运算
    - 1.2 算法核心框架
      - 1.2.1 模糊控制器接口基类
      - 1.2.2 模糊输入
        - 模糊输入部件对自由链表使用规则
      - 1.2.3 模糊推理
      - 1.2.4 模糊输出
      - 1.2.5 模糊控制器
    - 1.3 算法外围框架
      - 1.3.1 输入数据模糊化
      - 1.3.2 输出数据清晰化

## 1. 工程框架

### 1.1 数据结构

#### 1.1.1 模板式单向链表

模板式单向链表能够存放各种类型的数据，在实际结构中使用 `void* data` 指针进行管理。

```
struct list
{
    void *data;
    struct list *next;
};
```

由于 `list.data` 是 `void*` 类型，没办法在指针库中对数据进行更多的操作，因此需要使用时传入谓词或事件回调函数等对数据进行操作。

库中链表的操作函数如下：

谓词和事件回调函数指针

名称	参数	返回值	描述
<code>list_pred</code>	<code>list_node,void*</code>	<code>bool</code>	谓词，通常用于确定是否是指定节点，
<code>list_node_data_construct_cb</code>	<code>void*</code>	<code>bool</code>	节点参数构造函数
<code>list_node_data_deconstruct_cb</code>	<code>void*</code>	<code>bool</code>	节点参数析构函数
<code>list_event_cb</code>	<code>list_node,void*</code>	<code>bool</code>	事件回调函数，可用于操作节点参数

默认谓词和回调函数

名称	参数	返回值	描述
<code>list_pred_true</code>	<code>list_node,void*</code>	<code>bool</code>	默认真谓词
<code>list_pred_false</code>	<code>list_node,void*</code>	<code>bool</code>	默认假谓词

链表操作API

名称	参数	返回值	描述
<code>list_create</code>	-	<code>list_head</code>	创建链表句柄（头节点）
<code>list_clear</code>	<code>list_head,list_node_data_deconstruct_cb</code>	<code>bool</code>	清空链表中的节点
<code>list_delete</code>	<code>list_head,list_node_data_deconstruct_cb</code>	<code>bool</code>	删除链表（包括链表中的节点和
<code>list_length</code>	<code>list_head</code>	<code>list_index</code>	获取链表长度（链表中的节点数

链表节点操作API

名称	参数	返回值	
list_node_is_on_list	list_head,list_node	bool	判断
list_find_prev_node	list_head,list_node	list_node	查找
list_find_next_node	list_head,list_node	list_node	查找
list_push	list_head,void*,size_t	bool	向链
list_pop	list_head	bool	将链
list_find_if	list_head,void*,list_pred	list_node	在链
list_push_if	list_head,void*,size_t,list_pred,list_node_data_construct_cb	bool	向链
list_pop_if	list_head,void*,list_pred,list_node_data_deconstruct_cb	bool	将链
list_remove_if	list_head,void*,list_pred,list_node_data_deconstruct_cb	bool	移除
list_swap_if	list_head,void*,void*,list_pred,list_pred	bool	交换
list_trav	list_head,void*,list_event_cb	bool	遍历

参数获取API

名称	参数	返回值	描述
list_get_first_node	list_head	list_node	获取链表中的第一个节点， 如果链表是
list_get_last_node	list_head	list_node	获取链表中的最后一个节点， 如果链表
list_get_node_data	list_head,list_index	void*	获取指定索引处节点的数据
list_get_node_data_if	list_head,void*,list_pred	void*	获取满足谓词描述的节点的数据

1.1.2 模糊运算

模糊运算是进行模糊推理的基础， 在这里， 仅讨论对模糊矩阵的运算。

为了实现模糊矩阵运算， 首先得有一个表示模糊矩阵的数据类型：

```
/**
 * @brief Fuzzy matrix
 *
 * @memberof mat matrix
 * @memberof row matrix rows
 * @memberof col matrix columns
 */
struct fuzzy_matrix
{
    fuzzy_number** mat;
    fuzzy_size row;
    fuzzy_size col;
};
```

该数据类型使用二维指针对模糊矩阵进行管理，因此涉及到了动态内存分配，这是很危险的，所以我们需要将危险放在模块中，尽量避免使用者造成内存泄漏，我创建了以下几个函数，用于管理模糊矩阵。

名称	参数	返回值	
fuzzy_matrix_init	struct fuzzy_matrix*	bool	初始化模糊矩阵
fuzzy_matrix_create	struct fuzzy_matrix*,fuzzy_size,fuzzy_size	bool	创建指定行数和列数的模糊矩阵
fuzzy_matrix_reshape	struct fuzzy_matrix*,fuzzy_size,fuzzy_size	bool	使用 <code>realloc</code> 重新分配内存
fuzzy_matrix_reshape_s	struct fuzzy_matrix*,fuzzy_size,fuzzy_size	bool	更加安全的 <code>realloc</code>
fuzzy_matrix_clear	struct fuzzy_matrix*	bool	将矩阵中的每个元素置为0
fuzzy_matrix_pay_tribute	struct fuzzy_matrix*,struct fuzzy_matrix*	bool	纳贡函数，皇朝向藩属国进贡
fuzzy_matrix_rob	struct fuzzy_matrix*,struct fuzzy_matrix*	bool	抢夺函数，国与国之间抢夺
fuzzy_matrix_delete	struct fuzzy_matrix*	bool	销毁创建的模糊矩阵
fuzzy_matrix_copy	struct fuzzy_matrix*,struct fuzzy_matrix*	bool	将源模糊矩阵复制到目标矩阵
fuzzy_matrix_copy_just_elem	struct fuzzy_matrix*,struct fuzzy_matrix*,fuzzy_size,fuzzy_size	bool	仅赋值对应位置的元素

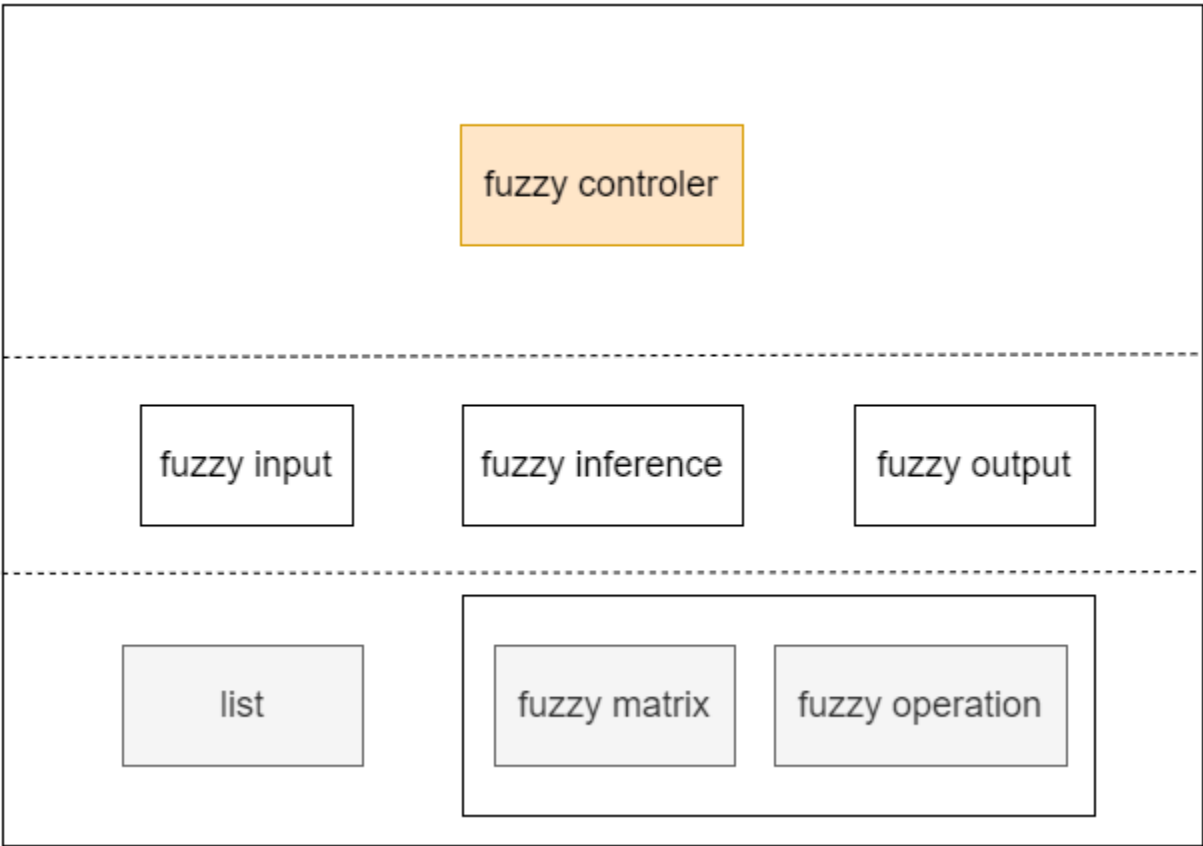
名称	参数	返回值	
fuzzy_matrix_horzcat	struct fuzzy_matrix*,struct fuzzy_matrix*,struct fuzzy_matrix*	bool	对两个矩阵进行水平连接，返回的矩阵的维数为 [n, m1+m2] dst
fuzzy_matrix_vertcat	struct fuzzy_matrix*,struct fuzzy_matrix*,struct fuzzy_matrix*	bool	对两个矩阵进行垂直连接，返回的矩阵的维数为 [n1+n2, m] dst
fuzzy_matrix_repmat	struct fuzzy_matrix*,struct fuzzy_matrix*,fuzzy_size,fuzzy_size	bool	将矩阵堆叠，返回的矩阵的维数为 [n, m, size] dst
fuzzy_matrix_trav	struct fuzzy_matrix*,void*,fuzzy_opera_event_cb	bool	遍历矩阵，并执行回调函数 dst
fuzzy_matrix_print	struct fuzzy_matrix*,const char*	-	打印出矩阵中的元素

除了这些对模糊矩阵内存的操作以外，还需要对模糊矩阵进行运算。

名称	参数	返回值	
fuzzy_opera_trans	struct fuzzy_matrix*,struct fuzzy_matrix*	bool	模糊矩阵转置，如果以自己为模板或者没有模板，则返回的矩阵维数为 [m, n] matT
fuzzy_opera_dir_pro	struct fuzzy_matrix*,struct fuzzy_matrix*,struct fuzzy_matrix*	bool	求模糊矩阵的直积。相比于 fuzzy_opera_dir_pro_s，该函数在直积失败时，返回的矩阵维数仍为 [n, m]
fuzzy_opera_dir_pro_s	struct fuzzy_matrix*,struct fuzzy_matrix*,struct fuzzy_matrix*	bool	求模糊矩阵的直积。注意，如果直积失败，返回的矩阵维数为 [0, 0]
fuzzy_opera	struct fuzzy_matrix*,struct fuzzy_matrix*,struct fuzzy_matrix*	bool	实现矩阵的模糊运算，如果模糊运算失败，返回的矩阵维数为 [0, 0]

# 1.2 算法核心框架

算法核心框架图如下：



核心需要使用链表和一些基本的模糊矩阵及其运算，在它们的基础之上，搭建起模糊控制器的输入输出和推理，之后整合三者成为模糊控制器。

## 1.2.1 模糊控制器接口基类

通过分析模糊控制器的三大部件（输入部件、推理部件、输出部件），发现其有一定的相同点，即都需要一个名称、一个类别、一个模糊矩阵、一个未限制使用方式的链表，因此我创建了模糊控制器的接口基类，其形式如下：

```
/**
 * @brief Fuzzy controller interface components
 *
 * @memberof Obj object
 * @memberof data fuzzy Data
 * @memberof l list
 */
struct fc_interface
{
    struct fc_obj obj;

    struct fuzzy_matrix data;
    list_head l;
};
```

其中，使用到了最终基类 `struct fc_obj`，其形式如下：

```
struct fc_obj
{
    const char* name;

    fc_obj_type type;
};
```

在接口基类中提供了一些API用于注册、注销、打印数据等。

名称	参数	返回值	
fc_interface_register	struct fc_interface* const,const char*,const fc_obj_type	bool	注册对象，赋予对象名称、类别，初始化模糊矩阵
fc_interface_unregister	struct fc_interface* const	bool	注销对象

名称	参数	返回值	
fc_interface_print_data	const struct fc_interface* const	bool	将对象中的模糊矩阵打印出来

### 1.2.2 模糊输入

模糊输入部件继承了接口基类，并添加了一系列函数指针，方便实现函数式操作对象，其形式如下：

```
/**
 * @brief Fuzzy controller input components
 * @details The linked list in the interface is used to store the membership
 *          function and its labels(struct membership_fn_label)
 *
 * @memberof interface object
 * @memberof register_dev register
 * @memberof unregister_dev unregister
 * @memberof add_membership_fn add membership function
 * @memberof clear_membership_fn clear membership function
 * @memberof fuzzing fuzzing data
 * @memberof print print fuzzied data
 */
struct fc_input
{
    struct fc_interface interface;

    bool (*register_dev)(struct fc_input* const in, const char* name);
    bool (*unregister_dev)(struct fc_input* const in);
    bool (*add_membership_fn)(const struct fc_input* const in, const fc_membership_
    bool (*clear_membership_fn)(const struct fc_input* const in);
    bool (*fuzzing)(struct fc_input* const in, const accurate_number* const value,
    bool (*print)(const struct fc_input* const in);
};
```

输入部件需要实现添加隶属函数标签对、清空隶属函数标签对、模糊化数据等功能，另外还需要实现获取输入部件名称、标签序列、模糊矩阵等功能。



与计算有关的API :

名称	参数	返回值	
fc_input_register	struct fc_input* const,const char*	bool	注册输入部件。
fc_input_unregister	struct fc_input* const	bool	注销输入部件
fc_input_add_membership_fn	const struct fc_input* const,const fc_membership_fn,const char*	bool	添加隶属函数
fc_input_clear_membership_fn	const struct fc_input* const	bool	清空隶属函数
fc_input_fuzzing	struct fc_input* const,const accurate_number* const,const fuzzy_size	bool	将精确值转换
fc_input_print_membership_vector	const struct fc_input* const	bool	打印隶属度向量
fc_input_print_membership_vector_with_label	const struct fc_input* const	bool	打印带有标签的

参数获取API :

名称	参数	返回值	
fc_input_get_name	const struct fc_input* const	const char*	获取输入部件的名称， 可以当作输入参数的类别。
fc_input_get_label	const struct fc_input* const,const list_head	bool	获取输入部件中的标签信息。如果传入的链表非空。

名称	参数	返回值	
fc_input_get_fuzzy_data	const struct fc_input* const,struct fuzzy_matrix*	bool	获取输入部件中的模糊矩阵

**模糊输入部件对自由链表使用规则**

模糊输入部件用自由链表存储隶属度函数标签对，其形式如下：

```
/**
 * @brief Membership Function and Label Pairs
 *
 * @memberof fn membership fn
 * @memberof label fn's label
 */
struct membership_fn_label
{
    fc_membership_fn fn;
    const char* label;
};
```

**1.2.3 模糊推理**

**1.2.4 模糊输出**

**1.2.5 模糊控制器**

**1.3 算法外围框架**

算法外围框架图如下：

算法外围框架.png

**1.3.1 输入数据模糊化**

**1.3.2 输出数据清晰化**