filterparam

構築: Doxygen 1.9.3

Chapter 1

クラス索引

1.1 クラス一覧

クラス・構造体・共用体・インターフェースの一覧です。

Chapter 2

ファイル索引

2.1 ファイル一覧

詳解が付けられているファイルの一覧です。

Chapter 3

クラス詳解

3.1 BandParam 構造体

公開メンバ関数

- **BandParam** (BandType input_type, double input_left, double input_right)
- BandType **type** () const
- double **left** () const
- double **right** () const
- double **width** () const
- std::string **sprint** ()

限定公開変数類

- BandType **band_type**
- double **left_side**
- double **right_side**

この構造体詳解は次のファイルから抽出されました:

- include/cascade_iir.hpp
- lib/iir/cascade_iir.cpp

3.2 filter::iir::FilterParam 構造体

公開メンバ関数

- **FilterParam** (unsigned int, unsigned int, BandParam, unsigned int, unsigned int, double)
- **FilterParam** (unsigned int, unsigned int, std::vector< BandParam >, unsigned int, unsigned int, double)
- unsigned int **pole_order** () const
- unsigned int **zero_order** () const
- unsigned int **opt_order** () const
- std::vector< BandParam > **fbands** () const
- unsigned int **partition_approx** () const

- unsigned int **partition_transition** () const
- double **gd** () const
- void **set_threshold_riple** (double input)
- std::vector< std::vector< std::complex< double > > > **freq_res** (const std::vector< double > &coef) const
- std::vector< std::vector< double > > **group_delay_res** (const std::vector< double > &coef) const
- double **judge_stability** (const std::vector< double > &coef) const
- double **evaluate** (const std::vector< double > &) const
- std::vector< double > **init_coef** (const double, const double, const double) const
- std::vector< double > **init_stable_coef** (const double, const double) const
- void **gprint_amp** (const std::vector< double > &, const std::string &, const double, const double) const
- void **gprint_mag** (const std::vector< double > &, const std::string &, const double, const double) const

静的公開メンバ関数

- static std::vector< FilterParam > **read_csv** (std::string &)
- template<typename... Args>
  static std::vector< BandParam > **gen_bands** (FilterType, Args...)
- static FilterType **analyze_type** (const std::string &)
- static std::vector< double > **analyze_edges** (const std::string &)
- static std::vector< std::complex< double > > **gen_csw** (const BandParam &, const unsigned int)
- static std::vector< std::complex< double > > **gen_csw2** (const BandParam &, const unsigned int)
- static std::vector< std::complex< double > > **gen_desire_res** (const BandParam &, const unsigned int, const double)

限定公開メンバ関数

- std::vector< std::vector< std::complex< double > > > **freq_res_se** (const std::vector< double > &) const
- std::vector< std::vector< std::complex< double > > > **freq_res_so** (const std::vector< double > &) const
- std::vector< std::vector< std::complex< double > > > **freq_res_no** (const std::vector< double > &) const
- std::vector< std::vector< std::complex< double > > > **freq_res_mo** (const std::vector< double > &) const
- std::vector< std::vector< double > > **group_delay_se** (const std::vector< double > &) const
- std::vector< std::vector< double > > **group_delay_so** (const std::vector< double > &) const
- std::vector< std::vector< double > > **group_delay_no** (const std::vector< double > &) const
- std::vector< std::vector< double > > **group_delay_mo** (const std::vector< double > &) const
- double **judge_stability_even** (const std::vector< double > &) const
- double **judge_stability_odd** (const std::vector< double > &) const

限定公開変数類

- unsigned int **n_order**
- unsigned int **m_order**
- std::vector< BandParam > **bands**
- unsigned int **nsplit_approx**
- unsigned int **nsplit_transition**
- double **group_delay**
- double **threshold_riple**
- std::vector< std::vector< std::complex< double > > > **csw**
- std::vector< std::vector< std::complex< double > > > **csw2**
- std::vector< std::vector< std::complex< double > > > **desire_res**
- std::function< std::vector< std::vector< std::complex< double > > >(const FilterParam *, const std::vector< double > &) > **freq_res_func**

- std::function< std::vector< std::vector< double > >(const FilterParam *, const std::vector< double > &) > **group_delay_func**
- std::function< double(const FilterParam *, const std::vector< double > &) > **stability_func**

この構造体詳解は次のファイルから抽出されました:

- include/cascade_iir.hpp
- lib/iir/cascade_iir.cpp

Chapter 4

ファイル詳解

4.1 include/cascade_iir.hpp ファイル

digital filter for cascade IIR filters

```
#include <cmath>
#include <complex>
#include <cstdio>
#include <cstdlib>
#include <ctime>
#include <fstream>
#include <functional>
#include <iostream>
#include <random>
#include <regex>
#include <sstream>
#include <string>
#include <vector>
```

クラス

- struct BandParam
- struct filter::iir::FilterParam

列挙型

- enum class FilterType { **LPF** , **HPF** , **BPF** , **BEF** , **Other** }
  フィルタの種類を示す列挙体
- enum class **BandType** { **Pass** , **Stop** , **Transition** }

関数

- template<typename... Args>
  std::string **format** (const std::string &, Args...)

4.1.1 詳解

digital filter for cascade IIR filters

著者

chatblanc

詳細な説明

4.2 cascade_iir.hpp

[詳解]

```
1
6
7 #ifndef FILTER_PARAM_HPP_
8 #define FILTER_PARAM_HPP_
9
10 #define _USE_MATH_DEFINES
11
12 #include <cmath>
13 #include <complex>
14 #include <cstdio>
15 #include <cstdlib>
16 #include <ctime>
17 #include <fstream>
18 #include <functional>
19 #include <iostream>
20 #include <random>
21 #include <regex>
22 #include <sstream>
23 #include <string>
24 #include <vector>
25
26 template< typename... Args >
27 std::string format( const std::string&, Args... );
28
29
33 enum class FilterType
34 {
35     LPF,
36     HPF,
37     BPF,
38     BEF,
39     Other
40 };
41
42 /* バンド(周波数帯域)の種別を示す列挙体
43  *   Pass : 通過域
44  *   Stop : 阻止域
45  *   Transition : 遷移域
46  */
47 enum class BandType
48 {
49     Pass,
50     Stop,
51     Transition
52 };
53
54 /* バンド(周波数帯域)の情報をまとめた構造体
55  *   type : 帯域の種類(通過・阻止・遷移)
56  *   left : 帯域の左端正規化周波数 [0:0.5)
57  *   right : 帯域の右端正規化周波数 (0:0.5]
58  *
59  *   left < rightであること。それ以外の場合、エラー終了。
60  */
61 struct BandParam
62 {
63 protected:
64
65     BandType band_type;
66     double left_side;
67     double right_side;
68
69 public:
```

```
70
71      BandParam( BandType input_type, double input_left, double input_right )
72          : band_type( input_type ), left_side( 0.0 ), right_side( 0.0 )
73      {
74          // バンドの条件に合わないとき、エラー終了
75          if ( input_left < 0.0 || input_left > input_right || input_right > 0.5 )
76          {
77              fprintf(
78                  stderr,
79                  "Error: [%s l.%d]Band edge is illegal(left :%6.3f, right "
80                  ":%6.3f)\n",
81                  __FILE__, __LINE__, input_left, input_right );
82              exit( EXIT_FAILURE );
83          }
84
85          left_side = input_left;
86          right_side = input_right;
87      }
88
89      BandType type() const { return band_type; }
90      double left() const { return left_side; }
91      double right() const { return right_side; }
92      double width() const { return right_side - left_side; }
93      std::string sprint();
94  };
95
96  namespace filter
97  {
98      namespace iir
99      {
100         struct FilterParam
101         {
102         protected:
103
104             // フィルタパラメータ
105
106             unsigned int n_order;
107             unsigned int m_order;
108             std::vector< BandParam > bands;
109             unsigned int nsplit_approx;
110             unsigned int nsplit_transition;
111             double group_delay;
112             double threshold_riple;
113
114             // 内部パラメータ
115
116             std::vector< std::vector< std::complex< double > > >
117                 csw;    // 複素正弦波e^-jωを周波数帯域別に格納
118             std::vector< std::vector< std::complex< double > > >
119                 csw2;    // 複素正弦波e^-j2ωを周波数帯域別に格納
120             std::vector< std::vector< std::complex< double > > >
121                 desire_res;    // 所望特性の周波数特性
122
123             std::function< std::vector< std::vector< std::complex< double > > >(
124                 const FilterParam*, const std::vector< double >& ) >
125                 freq_res_func;
126             std::function< std::vector< std::vector< double > >(
127                 const FilterParam*, const std::vector< double >& ) >
128                 group_delay_func;
129             std::function< double(
130                 const FilterParam*, const std::vector< double >& ) >
131                 stability_func;
132
133             // 内部メソッド
134
135             FilterParam()
136                 : n_order( 0 ), m_order( 0 ), nsplit_approx( 0 ),
137                   nsplit_transition( 0 ), group_delay( 0.0 ),
138                   threshold_riple( 1.0 )
139             {}
140
141             std::vector< std::vector< std::complex< double > > >
142             freq_res_se( const std::vector< double >& ) const;
143             std::vector< std::vector< std::complex< double > > >
144             freq_res_so( const std::vector< double >& ) const;
145             std::vector< std::vector< std::complex< double > > >
146             freq_res_no( const std::vector< double >& ) const;
147             std::vector< std::vector< std::complex< double > > >
148             freq_res_mo( const std::vector< double >& ) const;
149
150             std::vector< std::vector< double > >
151             group_delay_se( const std::vector< double >& ) const;
152             std::vector< std::vector< double > >
153             group_delay_so( const std::vector< double >& ) const;
154             std::vector< std::vector< double > >
155             group_delay_no( const std::vector< double >& ) const;
156             std::vector< std::vector< double > >
```

```
157             group_delay_mo( const std::vector< double >& ) const;
158
159             double judge_stability_even( const std::vector< double >& ) const;
160             double judge_stability_odd( const std::vector< double >& ) const;
161
162         public:
163
164             FilterParam(
165                 unsigned int,
166                 unsigned int,
167                 BandParam,
168                 unsigned int,
169                 unsigned int,
170                 double );
171             FilterParam(
172                 unsigned int,
173                 unsigned int,
174                 std::vector< BandParam >,
175                 unsigned int,
176                 unsigned int,
177                 double );
178
179             // get function
180
181             unsigned int pole_order() const { return m_order; }
182             unsigned int zero_order() const { return n_order; }
183             unsigned int opt_order() const { return 1 + n_order + m_order; }
184             std::vector< BandParam > fbands() const { return bands; }
185             unsigned int partition_approx() const { return nsplit_approx; }
186             unsigned int partition_transition() const
187             {
188                 return nsplit_transition;
189             }
190             double gd() const { return group_delay; }
191
192             // set function
193             /* # フィルタ構造体
194              *   振幅隆起として検知する閾値を変更する
195              *   デフォルト値は1.0
196              */
197             void set_threshold_riple( double input )
198             {
199                 threshold_riple = input;
200             }
201
202             // normal function
203             /* # フィルタ構造体
204              *   周波数特性計算関数
205              *   コンストラクタに与えられた周波数帯域に
206              *   応じて，縦続型IIRフィルタの周波数特性を計算する
207              *   また，係数列も次数によって適宜分割される
208              *
209              *   # 引数
210              *   vector<double> coef : 係数列(a0, a1, a2[0], a2[1],..., b1,
211              * b2[0], b2[1],...) #返り値 vector<vector<complex<double>>>
212              * response : 周波数帯域-周波数分割数の2重配列
213              */
214             std::vector< std::vector< std::complex< double > > >
215             freq_res( const std::vector< double >& coef ) const
216             {
217                 return this->freq_res_func( this, coef );
218             }
219
220             /* # フィルタ構造体
221              *   群遅延特性計算関数
222              *   コンストラクタに与えられた周波数帯域に
223              *   応じて，縦続型IIRフィルタの群遅延特性を計算する
224              *   また，係数列も次数によって適宜分割される
225              *
226              *   # 引数
227              *   vector<double> coef : 係数列(a0, a1, a2[0], a2[1],..., b1,
228              * b2[0], b2[1],...) #返り値 vector<vector<double>> response :
229              * 周波数帯域-周波数分割数の2重配列
230              */
231             std::vector< std::vector< double > >
232             group_delay_res( const std::vector< double >& coef ) const
233             {
234                 return this->group_delay_func( this, coef );
235             }
236
237             /* # フィルタ構造体
238              *   安定性判別関数
239              *
240              *   # 引数
241              *   vector<double> coef : 係数列(a0, a1, a2[0], a2[1],..., b1,
242              * b2[0], b2[1],...) #返り値 double response : 安定性のペナルティ
243              *                           0の場合に安定性を満たす
```

```
244              */
245             double judge_stability( const std::vector< double >& coef ) const
246             {
247                 return this->stability_func( this, coef );
248             }
249
250             double evaluate( const std::vector< double >& ) const;
251             std::vector< double >
252             init_coef( const double, const double, const double ) const;
253             std::vector< double >
254             init_stable_coef( const double, const double ) const;
255
256             void gprint_amp(
257                 const std::vector< double >&,
258                 const std::string&,
259                 const double,
260                 const double ) const;
261             void gprint_mag(
262                 const std::vector< double >&,
263                 const std::string&,
264                 const double,
265                 const double ) const;
266
267             // static function
268
269             static std::vector< FilterParam > read_csv( std::string& );
270
271             template< typename... Args >
272             static std::vector< BandParam > gen_bands( FilterType, Args... );
273             static FilterType analyze_type( const std::string& );
274             static std::vector< double > analyze_edges( const std::string& );
275             static std::vector< std::complex< double > >
276             gen_csw( const BandParam&, const unsigned int );
277             static std::vector< std::complex< double > >
278             gen_csw2( const BandParam&, const unsigned int );
279             static std::vector< std::complex< double > > gen_desire_res(
280                 const BandParam&, const unsigned int, const double );
281         };
282
283     }    // namespace iir
284 }    // namespace filter
285
286
287 //-------template function----------------------------------------
288 /* # String format function
289  *
290  */
291 template< typename... Args >
292 std::string format( const std::string& fmt, Args... args )
293 {
294     std::size_t len = static_cast< std::size_t >(
295         snprintf( nullptr, 0, fmt.c_str(), args... ) );
296     std::vector< char > buf( len + 1 );
297     snprintf( &buf[0], len + 1, fmt.c_str(), args... );
298     return std::string( &buf[0], &buf[0] + len );
299 }
300
301 /* # フィルタ構造体
302  *   フィルタのタイプと周波数帯域端により，
303  *   連続した周波数帯域の配列を生成する
304  *
305  * # 引数
306  * FilterType ftype : フィルタタイプ(L.P.F.やH.P.F.など)
307  * Args... edges : doubleの可変長引数を想定
308  *                     フィルタタイプに合わない帯域端，
309  *                     フィルタタイプが'Other'の場合には，
310  *                     異常終了する
311  * # 返り値
312  * vector<BandParam> bands : 連続した周波数帯域の配列
313  */
314 template< typename... Args >
315 std::vector< BandParam >
316 filter::iir::FilterParam::gen_bands( FilterType ftype, Args... edges )
317 {
318     std::vector< BandParam > bands;
319
320     switch ( ftype )
321     {
322         case FilterType::LPF:
323             {
324                 int nedge = sizeof...( edges );
325                 if ( nedge != 2 )
326                 {
327                     fprintf(
328                         stderr,
329                         "Error: [%s l.%d]It has not been implement yet.\n",
330                         __FILE__, __LINE__ );
```

```
331                     exit( EXIT_FAILURE );
332                 }
333
334                 double edge[] = { edges... };
335                 bands.emplace_back( BandParam( BandType::Pass, 0.0, edge[0] ) );
336                 bands.emplace_back(
337                     BandParam( BandType::Transition, edge[0], edge[1] ) );
338                 bands.emplace_back( BandParam( BandType::Stop, edge[1], 0.5 ) );
339                 break;
340             }
341         case FilterType::Other:
342             {
343                 fprintf(
344                     stderr,
345                     "Error: [%s l.%d]Other type filter can't use this "
346                     "function. "
347                     "Please, make `vector<BandParam>` your self.\n",
348                     __FILE__, __LINE__ );
349                 exit( EXIT_FAILURE );
350                 break;
351             }
352         case FilterType::HPF:
353         case FilterType::BPF:
354         case FilterType::BEF:
355         default:
356             {
357                 fprintf(
358                     stderr, "Error: [%s l.%d]It has not been implement yet.\n",
359                     __FILE__, __LINE__ );
360                 exit( EXIT_FAILURE );
361                 break;
362             }
363     }
364
365     return bands;
366 }
367
368 #endif /* FILTER_PARAM_HPP_ */
```

Index