

Sep 08, 11 5:31

bg_residuez_nl.txt

Page 1/4

```

1  %% Copyright (C) 2005 Julius O. Smith III
2  %%
3  %% This program is free software; you can redistribute it and/or modify it
4  %% under the terms of the GNU General Public License as published by
5  %% the Free Software Foundation; either version 2, or (at your option)
6  %% any later version.
7  %%
8  %% This program is distributed in the hope that it will be useful, but
9  %% WITHOUT ANY WARRANTY; without even the implied warranty of
10 %% MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. The GNU
11 %% General Public License has more details.
12 %%
13 %% You should have received a copy of the GNU General Public License
14 %% along with this program; see the file COPYING. If not, see
15 %% <http://www.gnu.org/licenses/>.
16
17 %% -*- texinfo -*-
18 %% @deftypefn {Function File} {@var{r}, @var{p}, @var{f}, @var{m}} = residuez (@var{B}, @var{A})
19 %% Compute the partial fraction expansion of filter @math{H(z) = B(z)/A(z)}.
20 %%
21 %% INPUTS:
22 %% @var{B} and @var{A} are vectors specifying the digital filter @math{H(z) = B(z)/A(z)}.
23 %% Say @code{help filter} for documentation of the @var{B} and @var{A}
24 %% filter coefficients.
25 %%
26 %% RETURNED:
27 %% @itemize
28 %% @item @var{r} = column vector containing the filter-pole residues@*
29 %% @item @var{p} = column vector containing the filter poles@*
30 %% @item @var{f} = row vector containing the FIR part, if any@*
31 %% @item @var{m} = column vector of pole multiplicities
32 %% @end itemize
33 %%
34 %% EXAMPLES:
35 %% @example
36 %% Say @code{test residuez verbose} to see a number of examples.
37 %% @end example
38 %%
39 %% For the theory of operation, see
40 %% @indicateurl{http://ccrma.stanford.edu/~jos/filters/residuez.html}
41 %%
42 %% @seealso{residue residuez}
43 %% @end deftypefn
44
45 %%%%%%%%%%%%%%%%% old_code %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
46 %function [r, p, f, m] = residuez(B, A, tol)
47 %%%%%%%%%%%%%%%%% new_code %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
48 %function [r, p, f, m] = bg_residuez(B, A, tol)
49 %%%%%%%%%%%%%%%%% end of new_code %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
50 % RESIDUEZ - return residues, poles, and FIR part of B(z)/A(z)
51 %
52 % Let nb = length(b), na = length(a), and N=na-1 = no. of poles.
53 % If nb<na, then f will be empty, and the returned filter is
54 %
55 % 
$$H(z) = \frac{r(1)}{[1-p(1)/z]^m(1)} + \dots + \frac{r(N)}{[1-p(N)/z]^m(N)} = R(z)$$

56 %
57 %
58 %
59 % If, on the other hand, nb >= na, the FIR part f will not be empty.
60 % Let M = nb-na+1 = order of f = length(f)-1. Then the returned filter is
61 %
62 % 
$$H(z) = f(1) + f(2)/z + f(3)/z^2 + \dots + f(M+1)/z^M + R(z)$$

63 %
64 % where R(z) is the parallel one-pole filter bank defined above.
65 % Note, in particular, that the impulse-response of the one-pole
66 % filter bank is in parallel with that of the the FIR part. This can
67 % be wasteful when matching the initial impulse response is important,
68 % since F(z) can already match the first N terms of the impulse
69 % response. To obtain a decomposition in which the impulse response of
70 % the IIR part R(z) starts after that of the FIR part F(z), use RESIDUED.
71 %
72 % J.O. Smith, 9/19/05
73
74 if nargin==3
75     warning("tolerance ignored");
76 end
77 NUM = B(:)'; DEN = A(:)';
78 % Matlab's residue does not return m (since it is implied by p):
79 %%%%%%%%%%%%%%%%% old_code %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
80 % [r,p,f,m]=residue(conj(fliplr(NUM)),conj(fliplr(DEN)));
81 %%%%%%%%%%%%%%%%% new_code %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
82 % global G_FID ;
83 % global G_DEBUG ;
84 % if G_DEBUG
85 %     fprintf(G_FID,'Converting a and b to A and B in bg_residuez give
s\n');

```

Sep 08, 11 5:31

bg_residuez_nl.txt

Page 2/4

```

86         bg_print('B',conj(fliplr(NUM)));
87         bg_print('A',conj(fliplr(DEN)));
88         fprintf(G_FID,'----- ');
89         fprintf(G_FID,'Into [R,P,F,M]=bg_residue(B,A,"MODE_Z");\n');
90     end
91     [r,p,f,m]=bg_residue(conj(fliplr(NUM)),conj(fliplr(DEN)),"MODE_Z");
92     if G_DEBUG
93         fprintf(G_FID,'----- ');
94         fprintf(G_FID,'Out of [R,P,F,M]=bg_residue(B,A,"MODE_Z");\n');
95         fprintf(G_FID,'Outputs from bg_residue are:\n');
96         bg_print('R',r);
97         bg_print('P',p);
98         bg_print('F',f);
99         bg_print('M',m);
100    end
101    %%%%%%%%%%% end of new_code %%%%%%%%%%%
102    p = 1 ./ p;
103    r = r .* ((-p) .^m);
104    if f, f = conj(fliplr(f)); end
105
106    %%%%%%%%%%% new_code %%%%%%%%%%%
107    if G_DEBUG
108        fprintf(G_FID,'Converting R,P,F and M to r,p,f and m ');
109        fprintf(G_FID,'in bg_residuez gives;\n');
110        bg_print('r',r);
111        bg_print('p',p);
112        bg_print('f',f);
113        bg_print('m',m);
114    end
115    endfunction
116    %%%%%%%%%%% end of new_code %%%%%%%%%%%
117    %!test
118    %! B=[1 -2 1]; A=[1 -1];
119    %%%%%%%%%%% old_code %%%%%%%%%%%
120    %! [r,p,f,m] = residuez(B,A);
121    %%%%%%%%%%% new_code %%%%%%%%%%%
122    %! [r,p,f,m] = bg_residuez(B,A);
123    %%%%%%%%%%% end of new_code %%%%%%%%%%%
124    %! [r,p,f,m] = residuez(B,A);
125    %! assert(r,0,100*eps);
126    %! assert(p,1,100*eps);
127    %! assert(f,[1 -1],100*eps);
128    %! assert(m,1,100*eps);
129
130    %!test
131    %! B=1; A=[1 -1j];
132    %%%%%%%%%%% old_code %%%%%%%%%%%
133    %! [r,p,f,m] = residuez(B,A);
134    %%%%%%%%%%% new_code %%%%%%%%%%%
135    %! [r,p,f,m] = bg_residuez(B,A);
136    %%%%%%%%%%% end of new_code %%%%%%%%%%%
137    %! assert(r,1,100*eps);
138    %! assert(p,1j,100*eps);
139    %! assert(f,[],100*eps);
140    %! assert(m,1,100*eps);
141
142    %!test
143    %! B=1; A=[1 -1 .25];
144    %%%%%%%%%%% old_code %%%%%%%%%%%
145    %! [r,p,f,m] = residuez(B,A);
146    %%%%%%%%%%% new_code %%%%%%%%%%%
147    %! [r,p,f,m] = bg_residuez(B,A);
148    %%%%%%%%%%% end of new_code %%%%%%%%%%%
149    %! [rs,is] = sort(r);
150    %! assert(rs,[0;1],1e-7);
151    %! assert(p(is),[0.5;0.5],1e-8);
152    %! assert(f,[],100*eps);
153    %! assert(m(is),[1;2],100*eps);
154
155    %!test
156    %! B=1; A=[1 -0.75 .125];
157    %%%%%%%%%%% old_code %%%%%%%%%%%
158    %! [r,p,f,m] = residuez(B,A);
159    %%%%%%%%%%% new_code %%%%%%%%%%%
160    %! [r,p,f,m] = bg_residuez(B,A);
161    %%%%%%%%%%% end of new_code %%%%%%%%%%%
162    %! [rs,is] = sort(r);
163    %! assert(rs,[-1;2],100*eps);
164    %! assert(p(is),[0.25;0.5],100*eps);
165    %! assert(f,[],100*eps);
166    %! assert(m(is),[1;1],100*eps);
167
168    %!test
169    %! B=[1,6,2]; A=[1,-2,1];
170    %%%%%%%%%%% old_code %%%%%%%%%%%
171    %! [r,p,f,m] = residuez(B,A);

```

Sep 08, 11 5:31

bg_residuez_nl.txt

Page 3/4

```

172 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
173 %! [r,p,f,m] = bg_residuez(B,A);
174 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
175 %! [rs,is] = sort(r);
176 %! assert(rs,[-10;9],1e-7);
177 %! assert(p(is),[1;1],1e-8);
178 %! assert(f,[2],100*eps);
179 %! assert(m(is),[1;2],100*eps);
180
181 %!test
182 %! B=[6,2]; A=[1,-2,1];
183 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
184 %! [r,p,f,m] = residuez(B,A);
185 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
186 %! [r,p,f,m] = bg_residuez(B,A);
187 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
188 %! [rs,is] = sort(r);
189 %! assert(rs,[-2;8],1e-7);
190 %! assert(p(is),[1;1],1e-8);
191 %! assert(f,[1],100*eps);
192 %! assert(m(is),[1;2],100*eps);
193
194 %!test
195 %! B=[1,6,6,2]; A=[1,-2,1];
196 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
197 %! [r,p,f,m] = residuez(B,A);
198 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
199 %! [r,p,f,m] = bg_residuez(B,A);
200 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
201 %! [rs,is] = sort(r);
202 %! assert(rs,[-24;15],2e-7);
203 %! assert(p(is),[1;1],1e-8);
204 %! assert(f,[10,2],100*eps);
205 %! assert(m(is),[1;2],100*eps);
206
207 %!test
208 %! B=[1,6,6,2]; A=[1,-(2+j),(1+2j),-j];
209 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
210 %! [r,p,f,m] = residuez(B,A);
211 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
212 %! [r,p,f,m] = bg_residuez(B,A);
213 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
214 %! [rs,is] = sort(r);
215 %! assert(rs,[-2+2.5j;7.5+7.5j;-4.5-12j],1E-6);
216 %! assert(p(is),[1j;1;1],1E-6);
217 %! assert(f,-2j,1E-6);
218 %! assert(m(is),[1;2;1],1E-6);
219
220 %!test
221 %! B=[1,0,1]; A=[1,0,0,0,-1];
222 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
223 %! [r,p,f,m] = residuez(B,A);
224 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
225 %! [r,p,f,m] = bg_residuez(B,A);
226 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
227 %! [as,is] = sort(angle(p));
228 %! rise = [ ...
229 %! 0.26180339887499 - 0.19021130325903i; ...
230 %! 0.03819660112501 + 0.11755705045849i; ...
231 %! 0.4; ...
232 %! 0.03819660112501 - 0.11755705045849i; ...
233 %! 0.26180339887499 + 0.19021130325903i;];
234 %! pise = [ ...
235 %! -0.80901699437495 - 0.58778525229247i; ...
236 %! 0.30901699437495 - 0.95105651629515i; ...
237 %! 1; ...
238 %! 0.30901699437495 + 0.95105651629515i; ...
239 %! -0.80901699437495 + 0.58778525229247i];
240 %! assert(r(is),rise,100*eps);
241 %! assert(p(is),pise,100*eps);
242 %! assert(f,[1],100*eps);
243 %! assert(m,[1;1;1;1],100*eps);
244 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
245 %!test
246 %! z1=+7.0372976777e+06;
247 %! p1=-3.1415926536e+09;
248 %! p2=-4.9964813512e+08;
249 %! r1=-(1+z1/p1)/(1-p1/p2)/p2/p1 ;
250 %! r2=-(1+z1/p2)/(1-p2/p1)/p2/p1 ;
251 %! r3=(1+(p2+p1)/p2/p1*z1)/p2/p1 ;
252 %! r4=z1/p2/p1 ;
253 %! fi= 1.05e+08 ;
254 %! T =1/fi ;
255 %! p1=exp(p1*T);
256 %! p2=exp(p2*T);
257 %! p3=1;

```

Sep 08, 11 5:31

bg_residuez_nl.txt

Page 4/4

```

258 %! p4=1;
259 %! p =[p1;p2;p3;p4];
260 %! r4=r4*T;
261 %! r3=r3-r4;
262 %! r =[r1;r2;r3;r4];
263 %! m=[      1      ,      1      ,      1      ,      1      ;
264 %!      -(2+p2)    ,  -(2+p1)    ,  -(1+p1+p2)    ,  -(p1+p2)    ;
265 %!      +(1+2*p2)  ,  +(1+2*p1)  ,  (p1*p2+p1+p2)  ,  +p1*p2    ;
266 %!      -p2       ,  -p1       ,  -p1*p2       ,      0       1;
267 %! a1=1.00;
268 %! a2=-(2+p1+p2);
269 %! a3=1+2*(p1+p2)+p1*p2;
270 %! a4=-(p1+p2+2*p1*p2);
271 %! a5=p2*p1;
272 %! a =[a1,a2,a3,a4,a5];
273 %! b=[m*r]';
274 %! k=[];
275 %! e=[1;1;1;2];
276 %! [rx,px,kx,ex] = bg_residuez(b,a);
277 %! assert ((abs (rx - r) < 1e-8
278 %!          && abs (px - p) < 1e-8
279 %!          && isempty (kx)
280 %!          && ex == e));
281 %%%%%%%%%%% end of new_code %%%%%%%%%%%

```