```
## Copyright (C) 1994-2011 John W. Eaton
## Copyright (C) 2007 Ben Abbott
##
## This file is part of Octave.
##
## Octave is free software; you can redistribute it and/or modify it
## under the terms of the GNU General Public License as published by
## the Free Software Foundation; either version 3 of the License, or (at
## your option) any later version.
##
## Octave is distributed in the hope that it will be useful, but
## WITHOUT ANY WARRANTY; without even the implied warranty of
## MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
## General Public License for more details.
##
## You should have received a copy of the GNU General Public License
## along with Octave; see the file COPYING.  If not, see
## <http://www.gnu.org/licenses/>.

## -*- texinfo -*-
## @deftypefn {Function File} {[@var{r}, @var{p}, @var{k}, @var{e}] =} residue (@var{b}, @var{a})
## Compute the partial fraction expansion for the quotient of the
## polynomials, @var{b} and @var{a}.
## @tex
## $$
## {B(s)\over A(s)} = \sum_{m=1}^M {r_m\over (s-p_m)^e_m}
##   + \sum_{i=1}^N k_i s^{N-i}.
## $$
## @end tex
## @ifnottex
##
## @example
## @group
##  B(s)    M       r(m)         N
##  ---- = SUM -------------  + SUM k(i)*s^(N-i)
##  A(s)   m=1 (s-p(m))^e(m)    i=1
## @end group
## @end example
##
## @end ifnottex
## @noindent
## where @math{M} is the number of poles (the length of the @var{r},
## @var{p}, and @var{e}), the @var{k} vector is a polynomial of order @math{N-1}
## representing the direct contribution, and the @var{e} vector specifies
## the multiplicity of the m-th residue's pole.
##
## For example,
##
## @example
## @group
## b = [1, 1, 1];
## a = [1, -5, 8, -4];
## [r, p, k, e] = residue (b, a);
##      @result{} r = [-2; 7; 3]
##      @result{} p = [2; 2; 1]
##      @result{} k = [](0x0)
##      @result{} e = [1; 2; 1]
## @end group
## @end example
##
## @noindent
## which represents the following partial fraction expansion
## @tex
## $$
## {s^2+s+1\over s^3-5s^2+8s-4} = {-2\over s-2} + {7\over (s-2)^2} + {3\over s-1}
## $$
## @end tex
## @ifnottex
##
## @example
## @group
##         s^2 + s + 1       -2        7        3
##    ------------------- = ----- + ------- + -----
##    s^3 - 5s^2 + 8s - 4   (s-2)   (s-2)^2   (s-1)
## @end group
## @end example
##
## @end ifnottex
##
## @deftypefnx {Function File} {[@var{b}, @var{a}] =} residue (@var{r}, @var{p}, @var{k})
## @deftypefnx {Function File} {[@var{b}, @var{a}] =} residue (@var{r}, @var{p}, @var{k}, @var{e})
## Compute the reconstituted quotient of polynomials,
## @var{b}(s)/@var{a}(s), from the partial fraction expansion;
## represented by the residues, poles, and a direct polynomial specified
## by @var{r}, @var{p} and @var{k}, and the pole multiplicity @var{e}.
##
```

```
## If the multiplicity, @var{e}, is not explicitly specified the multiplicity is
## determined by the function @code{mpoles}.
##
## For example:
##
## @example
## @group
## r = [-2; 7; 3];
## p = [2; 2; 1];
## k = [1, 0];
## [b, a] = residue (r, p, k);
##      @result{} b = [1, -5, 9, -3, 1]
##      @result{} a = [1, -5, 8, -4]
##
## where mpoles is used to determine e = [1; 2; 1]
##
## @end group
## @end example
##
## Alternatively the multiplicity may be defined explicitly, for example,
##
## @example
## @group
## r = [7; 3; -2];
## p = [2; 1; 2];
## k = [1, 0];
## e = [2; 1; 1];
## [b, a] = residue (r, p, k, e);
##      @result{} b = [1, -5, 9, -3, 1]
##      @result{} a = [1, -5, 8, -4]
## @end group
## @end example
##
## @noindent
## which represents the following partial fraction expansion
## @tex
## $$
## {-2\over s-2} + {7\over (s-2)^2} + {3\over s-1} + s = {s^4-5s^3+9s^2-3s+1\over s^3-5s^2+8s-4}
## $$
## @end tex
## @ifnottex
##
## @example
## @group
##     -2        7        3         s^4 - 5s^3 + 9s^2 - 3s + 1
##    ----- + ------- + ----- + s = --------------------------
##    (s-2)   (s-2)^2   (s-1)          s^3 - 5s^2 + 8s - 4
## @end group
## @end example
##
## @end ifnottex
## @seealso{poly, roots, conv, deconv, mpoles, polyval, polyderiv, polyint}
## @end deftypefn

## Author: Tony Richardson <arichard@stark.cc.oh.us>
## Author: Ben Abbott <bpabbott@mac.com>
## Created: June 1994
## Adapted-By: jwe

function [r, p, k, e] = residue (b, a, varargin)

  if (nargin < 2 || nargin > 4)
    print_usage ();
  endif

  toler = .001;

  if (nargin >= 3)
    if (nargin >= 4)
      e = varargin{2};
    else
      e = [];
    endif
    ## The inputs are the residue, pole, and direct part. Solve for the
    ## corresponding numerator and denominator polynomials
    [r, p] = rresidue (b, a, varargin{1}, toler, e);
    return
  endif

  ## Make sure both polynomials are in reduced form.

  a = polyreduce (a);
  b = polyreduce (b);

  b = b / a(1);
  a = a / a(1);
```

```
173
174     la = length (a);
175     lb = length (b);
176
177     ## Handle special cases here.
178
179     if (la == 0 || lb == 0)
180       k = r = p = e = [];
181       return;
182     elseif (la == 1)
183       k = b / a;
184       r = p = e = [];
185       return;
186     endif
187
188     ## Find the poles.
189
190     p = roots (a);
191     lp = length (p);
192
193     ## Sort poles so that multiplicity loop will work.
194
195     [e, indx] = mpoles (p, toler, 1);
196     p = p (indx);
197
198     ## For each group of pole multiplicity, set the value of each
199     ## pole to the average of the group. This reduces the error in
200     ## the resulting poles.
201
202     p_group = cumsum (e == 1);
203     for ng = 1:p_group(end)
204       m = find (p_group == ng);
205       p(m) = mean (p(m));
206     endfor
207
208     ## Find the direct term if there is one.
209
210     if (lb >= la)
211       ## Also return the reduced numerator.
212       [k, b] = deconv (b, a);
213       lb = length (b);
214     else
215       k = [];
216     endif
217
218     ## Determine if the poles are (effectively) zero.
219
220     small = max (abs (p));
221     if (isa (a, "single") || isa (b, "single"))
222       small = max ([small, 1]) * eps ("single") * 1e4 * (1 + numel (p))^2;
223     else
224       small = max ([small, 1]) * eps * 1e4 * (1 + numel (p))^2;
225     endif
226     p(abs (p) < small) = 0;
227
228     ## Determine if the poles are (effectively) real, or imaginary.
229
230     index = (abs (imag (p)) < small);
231     p(index) = real (p(index));
232     index = (abs (real (p)) < small);
233     p(index) = 1i * imag (p(index));
234
235     ## The remainder determines the residues.  The case of one pole
236     ## is trivial.
237
238     if (lp == 1)
239       r = polyval (b, p);
240       return;
241     endif
242
243     ## Determine the order of the denominator and remaining numerator.
244     ## With the direct term removed the potential order of the numerator
245     ## is one less than the order of the denominator.
246
247     aorder = numel (a) - 1;
248     border = aorder - 1;
249
250     ## Construct a system of equations relating the individual
251     ## contributions from each residue to the complete numerator.
252
253     A = zeros (border+1, border+1);
254     B = prepad (reshape (b, [numel(b), 1]), border+1, 0);
255     for ip = 1:numel(p)
256       ri = zeros (size (p));
257       ri(ip) = 1;
258       A(:,ip) = prepad (rresidue (ri, p, [], toler), border+1, 0).';
```

```
  endfor

  ## Solve for the residues.

  r = A \ B;

endfunction

function [pnum, pden, e] = rresidue (r, p, k, toler, e)

  ## Reconstitute the numerator and denominator polynomials from the
  ## residues, poles, and direct term.

  if (nargin < 2 || nargin > 5)
    print_usage ();
  endif

  if (nargin < 5)
    e = [];
  endif

  if (nargin < 4)
    toler = [];
  endif

  if (nargin < 3)
    k = [];
  endif

  if numel (e)
    indx = 1:numel(p);
  else
    [e, indx] = mpoles (p, toler, 0);
    p = p (indx);
    r = r (indx);
  endif

  indx = 1:numel(p);

  for n = indx
    pn = [1, -p(n)];
    if n == 1
      pden = pn;
    else
      pden = conv (pden, pn);
    endif
  endfor

  ## D is the order of the denominator
  ## K is the order of the direct polynomial
  ## N is the order of the resulting numerator
  ## pnum(1:(N+1)) is the numerator's polynomial
  ## pden(1:(D+1)) is the denominator's polynomial
  ## pm is the multible pole for the nth residue
  ## pn is the numerator contribution for the nth residue

  D = numel (pden) - 1;
  K = numel (k) - 1;
  N = K + D;
  pnum = zeros (1, N+1);
  for n = indx(abs (r) > 0)
    p1 = [1, -p(n)];
    for m = 1:e(n)
      if (m == 1)
        pm = p1;
      else
        pm = conv (pm, p1);
      endif
    endfor
    pn = deconv (pden, pm);
    pn = r(n) * pn;
    pnum = pnum + prepad (pn, N+1, 0, 2);
  endfor

  ## Add the direct term.

  if (numel (k))
    pnum = pnum + conv (pden, k);
  endif

  ## Check for leading zeros and trim the polynomial coefficients.
  if (isa (r, "single") || isa (p, "single") || isa (k, "single"))
    small = max ([max(abs(pden)), max(abs(pnum)), 1]) * eps ("single");
  else
    small = max ([max(abs(pden)), max(abs(pnum)), 1]) * eps;
  endif
```

```
345
346    pnum(abs (pnum) < small) = 0;
347    pden(abs (pden) < small) = 0;
348
349    pnum = polyreduce (pnum);
350    pden = polyreduce (pden);
351
352  endfunction
353
354  %!test
355  %! b = [1, 1, 1];
356  %! a = [1, -5, 8, -4];
357  %! [r, p, k, e] = residue (b, a);
358  %! assert (abs (r - [-2; 7; 3]) < 1e-12
359  %!   && abs (p - [2; 2; 1]) < 1e-12
360  %!   && isempty (k)
361  %!   && e == [1; 2; 1]);
362  %! k = [1 0];
363  %! b = conv (k, a) + prepad (b, numel (k) + numel (a) - 1, 0);
364  %! a = a;
365  %! [br, ar] = residue (r, p, k);
366  %! assert ((abs (br - b) < 1e-12
367  %!   && abs (ar - a) < 1e-12));
368  %! [br, ar] = residue (r, p, k, e);
369  %! assert ((abs (br - b) < 1e-12
370  %!   && abs (ar - a) < 1e-12));
371
372  %!test
373  %! b = [1, 0, 1];
374  %! a = [1, 0, 18, 0, 81];
375  %! [r, p, k, e] = residue (b, a);
376  %! r1 = [-5i; 12; +5i; 12]/54;
377  %! p1 = [+3i; +3i; -3i; -3i];
378  %! assert (abs (r - r1) < 1e-12 && abs (p - p1) < 1e-12
379  %!   && isempty (k)
380  %!   && e == [1; 2; 1; 2]);
381  %! [br, ar] = residue (r, p, k);
382  %! assert ((abs (br - b) < 1e-12
383  %!   && abs (ar - a) < 1e-12));
384
385  %!test
386  %! r = [7; 3; -2];
387  %! p = [2; 1; 2];
388  %! k = [1 0];
389  %! e = [2; 1; 1];
390  %! [b, a] = residue (r, p, k, e);
391  %! assert ((abs (b - [1, -5, 9, -3, 1]) < 1e-12
392  %!   && abs (a - [1, -5, 8, -4]) < 1e-12));
393  %! [rr, pr, kr, er] = residue (b, a);
394  %! [jnk, n] = mpoles(p);
395  %! assert ((abs (rr - r(n)) < 1e-12
396  %!   && abs (pr - p(n)) < 1e-12
397  %!   && abs (kr - k) < 1e-12
398  %!   && abs (er - e(n)) < 1e-12));
399
400  %!test
401  %! b = [1];
402  %! a = [1, 10, 25];
403  %! [r, p, k, e] = residue (b, a);
404  %! r1 = [0; 1];
405  %! p1 = [-5; -5];
406  %! assert (abs (r - r1) < 1e-12 && abs (p - p1) < 1e-12
407  %!   && isempty (k)
408  %!   && e == [1; 2]);
409  %! [br, ar] = residue (r, p, k);
410  %! assert ((abs (br - b) < 1e-12
411  %!   && abs (ar - a) < 1e-12));
```