NAME

gjqrc − Gauss-Jacobi logarithmic Quadrature Recursion Coefficients

SYNOPSIS

Fortran (77, 90, 95, HPF):

f77 *[flags] file(s) . . . -L/usr/local/lib -lgjl*

SUBROUTINE gjqrc (a, b, s, t, alpha, beta, nquad, ierr)
DOUBLE PRECISION a(0:MAXPTS), alpha, b(0:MAXPTS), beta
DOUBLE PRECISION s(0:MAXPTS), t(0:MAXPTS)
INTEGER ierr, nquad

C (K&R, 89, 99), C++ (98):

cc *[flags] -I/usr/local/include file(s) . . . -L/usr/local/lib -lgjl*

Use

#include <gjl.h>

to get this prototype:

void gjqrc(fortran_double_precision a_[],
fortran_double_precision b_[],
fortran_double_precision s_[],
fortran_double_precision t_[],
const fortran_double_precision * alpha_,
const fortran_double_precision * beta_,
const fortran_integer * nquad_,
fortran_integer * ierr_);

NB: The definition of C/C++ data types **fortran_** *xxx*, and the mapping of Fortran external names to C/C++ external names, is handled by the C/C++ header file. That way, the same function or subroutine name can be used in C, C++, and Fortran code, independent of compiler conventions for mangling of external names in these programming languages.

DESCRIPTION

Compute the recursion coefficients and zeroth and first moments of the monic polynomials corresponding to the positive weight function

w(x,\alpha,\beta) = (1−x)^\alpha (1+x)^\beta (−ln((1+x)/2))

with recursion relation (n = 0, 1, 2, . . .)

P_{n+1}^{\alpha,\beta}(x) =
(x − B_n^{\alpha,\beta}) * P_n^{\alpha,\beta}(x)
− A_n^{\alpha,\beta} * P_{n−1}^{\alpha,\beta}(x)

and initial conditions

P_{−1}^{\alpha,\beta}(x) = 0
P_{0}^{\alpha,\beta}(x) = 1

Except in the weight function, the superscripts indicate dependence on \alpha, NOT exponentiation.

The required moments are:

T_n^{\alpha,\beta} = \int_0^infty w(x,\alpha,\beta) * (P_n^{\alpha,\beta}(x))^2 dx
S_n^{\alpha,\beta} = \int_0^infty w(x,\alpha,\beta) * (P_n^{\alpha,\beta}(x))^2 x dx

From these moments, the recursion coefficents are computed as:

A_n^{\alpha,\beta} = T_n^{\alpha,\beta} / T_{n−1}^{\alpha,\beta}

B_n^{\alpha,\beta} = S_n^{\alpha,\beta} / T_n^{\alpha,\beta}

On entry:

alpha Power of (1−x) in the integrand (**alpha** > −1).

beta Power of (1+x) in the integrand (**beta** > -1).

nquad Number of quadrature points to compute. It must be less than the limit MAXPTS defined in the header file, *maxpts.inc*. The default value chosen there should be large enough for any realistic application.

On return:

a(0. .**nquad**) Recursion coefficients: **a**(n) = A_n^{\alpha,\beta}.

b(0. .**nquad**) Recursion coefficients: **b**(n) = B_n^{\alpha,\beta}.

s(0. .**nquad**) First moments: **s**(n) = S_n^{\alpha,\beta}.

t(0. .**nquad**) Zeroth moments: **t**(n) = T_n^{\alpha,\beta}.

ierr Error indicator:
= 0 (success),
1 (eigensolution could not be obtained),
2 (destructive overflow),
3 (**nquad** out of range),
4 (**alpha** out of range).

AUTHORS

The algorithms and code are described in detail in the paper

Fast Gaussian Quadrature for Two Classes of Logarithmic Weight Functions

in ACM Transactions on Mathematical Software, Volume ??, Number ??, Pages ????--???? and ????--????, 20xx, by

Nelson H. F. Beebe
Center for Scientific Computing
University of Utah
Department of Mathematics, 110 LCB
155 S 1400 E RM 233
Salt Lake City, UT 84112-0090
Tel: +1 801 581 5254
FAX: +1 801 581 4148
Email: `beebe@math.utah.edu`, `beebe@acm.org`, `beebe@computer.org`
WWW URL: `http://www.math.utah.edu/~beebe`

and

James S. Ball
University of Utah
Department of Physics
Salt Lake City, UT 84112-0830
USA
Tel: +1 801 581 8397
FAX: +1 801 581 6256
Email: `ball@physics.utah.edu`
WWW URL: `http://www.physics.utah.edu/people/faculty/ball.html`