ghkfast() — GHK multivariate normal simulator using pregenerated points

Description

Please see [M-5] ghk(). The routines documented here do the same thing, but ghkfast() can be faster at the expense of using more memory. First, code $S = \text{ghkfast\_init}(\ldots)$ and then use \text{ghkfast}(S, \ldots) to obtain the simulated values. There is a time savings because the simulation points are generated once in \text{ghkfast\_init}(), whereas for \text{ghk()} the points are generated on each call to \text{ghk}(). Also, \text{ghkfast}() can generate simulated probabilities from the generalized Halton sequence; see [M-5] halton().

\text{ghkfast\_init}(n, \text{npts}, \text{dim}, \text{method}) computes the simulation points to be used by \text{ghkfast}(). Inputs $n$, \text{npts}, and \text{dim} are the number of observations, the number of repetitions for the simulation, and the maximum dimension of the multivariate normal (MVN) distribution, respectively. Input \text{method} specifies the type of points to generate and can be one of "halton", "hammersley", "random", or "ghalton".

\text{ghkfast}(S, X, V) returns an $n \times 1$ real vector containing the simulated values of the MVN distribution with $\text{dim} \times \text{dim}$ variance–covariance matrix $V$ at the points stored in the rows of the $n \times \text{dim}$ matrix $X$.

\text{ghkfast}(S, X, V, \text{dfdx}, \text{dfdv}) does the same thing as \text{ghkfast}(S, X, V) but also returns the first-order derivatives of the simulated probability with respect to the rows of $X$ in $\text{dfdx}$ and the simulated probability derivatives with respect to $\text{vech}(V)$ in $\text{dfdv}$. See $\text{vech}()$ in [M-5] vec() for details of the half-vectorized operator.

The \text{ghk\_query\_n}(S), \text{ghk\_query\_npts}(S), \text{ghk\_query\_dim}(S), and \text{ghk\_query\_method}(S) functions extract the number of observations, number of simulation points, maximum dimension, and method of point-set generation that is specified in the construction of the transmorphic object $S$. Use \text{ghk\_query\_rseed}(S) to retrieve the uniform random-variate seed used to generate the "random" or "ghalton" point sets. The \text{ghkfast\_query\_pointset\_i}(S, \text{i}) function will retrieve the $i$th point set used to simulate the MVN probability for the $i$th observation.

The \text{ghkfast\_i}(S, X, V, \text{i}, \ldots) function computes the probability and derivatives for the $i$th observation, $i = 1, \ldots, n$. 
Syntax

\[ S = \text{ghkfast}_\text{init}(\text{real scalar } n, \text{npts}, \text{dim}, \text{string scalar method}) \]

- \( S \) is a transmorphic object
- \( \text{ghkfast}_\text{init}_\text{pivot}(S[,\text{real scalar pivot}]) \)
- \( \text{ghkfast}_\text{init}_\text{antithetics}(S[,\text{real scalar anti}]) \)
- \( \text{real scalar } \text{ghkfast}_\text{query}_\text{n}(S) \)
- \( \text{real scalar } \text{ghkfast}_\text{query}_\text{npts}(S) \)
- \( \text{real scalar } \text{ghkfast}_\text{query}_\text{dim}(S) \)
- \( \text{string scalar } \text{ghkfast}_\text{query}_\text{method}(S) \)
- \( \text{string scalar } \text{ghkfast}_\text{query}_\text{rseed}(S) \)
- \( \text{real matrix } \text{ghkfast}_\text{query}_\text{pointset}_\text{i}(S, i) \)
- \( \text{real colvector } \text{ghkfast}(S, \text{real matrix } X, V) \)
- \( \text{real colvector } \text{ghkfast}(S, \text{real matrix } X, V, \text{dfdx}, \text{dfdv}) \)
- \( \text{real scalar } \text{ghkfast}_\text{i}(S, \text{real matrix } X, V, i) \)
- \( \text{real scalar } \text{ghkfast}_\text{i}(S, \text{real matrix } X, V, i, \text{dfdx}, \text{dfdv}) \)

where \( S \), if it is declared, should be declared as a transmorphic object

and where \text{method} specified in \text{ghkfast}_\text{init}() is

<table>
<thead>
<tr>
<th>\text{method}</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;halton&quot;</td>
<td>Halton sequences</td>
</tr>
<tr>
<td>&quot;hammersley&quot;</td>
<td>Hammersley’s variation of the Halton set</td>
</tr>
<tr>
<td>&quot;random&quot;</td>
<td>pseudorandom uniforms</td>
</tr>
<tr>
<td>&quot;ghalton&quot;</td>
<td>generalized Halton sequences</td>
</tr>
</tbody>
</table>

Remarks and examples

For problems where repetitive calls to the GHK algorithm are required, \text{ghkfast}() might be a preferred alternative to \text{ghk}(). Generating the points once at the outset of a program produces a speed increase. For problems with many observations or many simulation points per observation, \text{ghkfast}() will be faster than \text{ghk}() at the cost of requiring more memory.

If \text{ghkfast}() is used within a likelihood evaluator for \text{ml} or \text{optimize}(), you will need to store the transmorphic object \( S \) as an \text{external} global and reuse the object with each likelihood evaluation. Alternatively, the initialization function for \text{optimize}(), \text{optimize_init_argument}(), can be used.
Prior to calling `ghkfast()`, call `ghkfast_init_npivot(S, 1)` to turn off the integration interval pivoting that takes place in `ghkfast()`. By default, `ghkfast()` pivots the wider intervals of integration (and associated rows/columns of the covariance matrix) to the interior of the multivariate integration to improve quadrature accuracy. This option may be useful when `ghkfast()` is used in a likelihood evaluator for [R] `ml` or [M-5] `optimize()` and few simulation points are used for each observation. Here the pivoting may cause discontinuities when computing numerical second-order derivatives using finite differencing (for the Newton–Raphson technique), resulting in a non–positive-definite Hessian.

Also the sequences "halton", "hammersley", and "random", `ghkfast()` will use the generalized Halton sequence, "ghalton". Generalized Halton sequences have the same uniform coverage (low discrepancy) as the Halton sequences with the addition of a pseudorandom uniform component. Therefore, "ghalton" sequences are like "random" sequences in that you should set the random-number seed before using them if you wish to replicate the same point set; see [M-5] `runiform()`.

**Conformability**

All initialization functions have $1 \times 1$ inputs and have $1 \times 1$ or `void` outputs, and all query functions have the transmorphic input and $1 \times 1$ outputs except

`ghkfast_init(n, npts, dim, method)`:  
**input:**  
- $n$: 1 x 1  
- $npts$: 1 x 1  
- $dim$: 1 x 1  
- $method$: 1 x 1  
**output:**  
- result: transmorphic

`ghkfast_query_pointset_i(S, i)`:  
**input:**  
- $S$: transmorphic  
- $i$: 1 x 1  
**output:**  
- result: npts x dim

`ghkfast(S, X, V)`:  
**input:**  
- $S$: transmorphic  
- $X$: n x dim  
- $V$: dim x dim (symmetric, positive definite)  
**output:**  
- result: n x 1

`ghkfast(S, X, V, dfdx, dfdv)`:  
**input:**  
- $S$: transmorphic  
- $X$: n x dim  
- $V$: dim x dim (symmetric, positive definite)  
**output:**  
- result: n x 1  
- $dfdx$: n x dim  
- $dfdv$: n x dim(dim + 1)/2
ghkfast_i(S, X, V, i, dfdx, dfdv):

input:
S: transmorphic
X: \( n \times \text{dim} \) or \( 1 \times \text{dim} \)
V: \( \text{dim} \times \text{dim} \) (symmetric, positive definite)
i: \( 1 \times 1 \) \( (1 \leq i \leq n) \)

output:
result: \( n \times 1 \)
dfdx: \( 1 \times \text{dim} \)
dfdv: \( 1 \times \text{dim}(\text{dim} + 1)/2 \)

Diagnostics

ghkfast_init(n, npts, dim, method) aborts with error if the dimension, \( \text{dim} \), is greater than 20.

ghkfast(S, X, V, ...) and ghkfast_i(S, X, V, i, ...) require that \( V \) be symmetric and positive definite. If \( V \) is not positive definite, then the returned vector (scalar) is filled with missings.

Also see

[M-5] ghk() — Geweke–Hajivassiliou–Keane (GHK) multivariate normal simulator
[M-5] halton() — Generate a Halton or Hammersley set
[M-4] Statistical — Statistical functions