

A Proposal for Community (Quasi-) Monte Carlo Software

Fred J. Hickernell

Department of Applied Mathematics

Center for Interdisciplinary Scientific Computation

Illinois Institute of Technology

`hickernell@iit.edu` `mypages.iit.edu/~hickernell`

Thanks to the workshop organizers, the GAIL team

NSF-DMS-1522687 and NSF-DMS-1638521 (SAMSI)

Follow-Up to Discussions, June 28, 2018





Questions that I Ask or Hear Asked

- Where can I find **quality**, **free** quasi-Monte Carlo (qMC) software?



Questions that I Ask or Hear Asked

- Where can I find **quality, free** quasi-Monte Carlo (qMC) software?
- Where can I find **use cases** that illustrate how to employ qMC methods?



Questions that I Ask or Hear Asked

- Where can I find **quality**, **free** quasi-Monte Carlo (qMC) software?
- Where can I find **use cases** that illustrate how to employ qMC methods?
- How can I try that qMC **method** developed by X's group for my problem?



Questions that I Ask or Hear Asked

- Where can I find **quality**, **free** quasi-Monte Carlo (qMC) software?
- Where can I find **use cases** that illustrate how to employ qMC methods?
- How can I try that qMC **method** developed by X's group for my problem?
- How can I try my qMC method on the **example** shown by Y's group?



Questions that I Ask or Hear Asked

- Where can I find **quality**, **free** quasi-Monte Carlo (qMC) software?
- Where can I find **use cases** that illustrate how to employ qMC methods?
- How can I try that qMC **method** developed by X's group for my problem?
- How can I try my qMC method on the **example** shown by Y's group?
- How can my student get results without writing code from **scratch**?



Questions that I Ask or Hear Asked

- Where can I find **quality**, **free** quasi-Monte Carlo (qMC) software?
- Where can I find **use cases** that illustrate how to employ qMC methods?
- How can I try that qMC **method** developed by X's group for my problem?
- How can I try my qMC method on the **example** shown by Y's group?
- How can my student get results without writing code from **scratch**?
- How can the code that I use benefit from **recent** developments?



- Where can I find **quality**, **free** quasi-Monte Carlo (qMC) software?
- Where can I find **use cases** that illustrate how to employ qMC methods?
- How can I try that qMC **method** developed by X's group for my problem?
- How can I try my qMC method on the **example** shown by Y's group?
- How can my student get results without writing code from **scratch**?
- How can the code that I use benefit from **recent** developments?
- How can my work receive wider **recognition**?



Questions that I Ask or Hear Asked

- Where can I find **quality**, **free** quasi-Monte Carlo (qMC) software?
- Where can I find **use cases** that illustrate how to employ qMC methods?
- How can I try that qMC **method** developed by X's group for my problem?
- How can I try my qMC method on the **example** shown by Y's group?
- How can my student get results without writing code from **scratch**?
- How can the code that I use benefit from **recent** developments?
- How can my work receive wider **recognition**?

My initial attempts in this direction over the past 5 years have produced GAIL¹

¹Choi, S.-C. T. *et al.* *GAIL: Guaranteed Automatic Integration Library (Versions 1.0–2.2)*. **MATLAB** software. 2013–2017. http://gailgithub.github.io/GAIL_Dev/.



Can We Have qMC Community Software that Grows Up to Be Like ...

Chebfun Computing with Chebyhsev polynomials, chebfun.org

Clawpack Solution of conservation laws, clawpack.org

deal.II Finite-elements, <http://dealii.org>

Mission: To provide well-documented tools to build finite element codes for a broad variety of PDEs, from laptops to supercomputers.

Vision: To create an open, inclusive, participatory community providing users and developers with a state-of-the-art, comprehensive software library that constitutes the go-to solution for all finite element problems.

FEniCS Finite-elements, fenicsproject.org

Gromacs Molecular dynamics, gromacs.org

Stan Markov Chain Monte Carlo, mc-stan.org

Trilinos Multiphysics computations, trilinos.org

- Developed and supported by multiple research groups
- Used beyond the research groups that develop it
- A recognized standard in its field



What Is Available Now

John Burkhardt Variety of qMC Software in C++, Fortran, MATLAB, and Python,
`people.sc.fsu.edu/~jburkardt/`

Mike Giles Multi-Level Monte Carlo Software in C++, MATLAB, Python, and R,
`people.maths.ox.ac.uk/gilesm/mlmc/`

Fred Hickernell Guaranteed Automatic Integration Library (GAIL) in MATLAB,
`gailgithub.github.io/GAIL_Dev/`

Stephen Joe & Frances Kuo Sobol' generators in C++, Generating vectors for lattices,
`web.maths.unsw.edu.au/~fkuo/`

Pierre L'Ecuyer Random number generators, Stochastic Simulation, Lattice Builder in C/C++ and Java,
`simul.iro.umontreal.ca`

Dirk Nuyens Magic Point Shop, QMC4PDE, etc. in MATLAB, Python, and C++,
`people.cs.kuleuven.be/~dirk.nuyens/`

Art Owen Various code, `statweb.stanford.edu/~owen/code/`

Christoph Schwab Partial differential equations with random coefficients

MATLAB Sobol' and Halton sequences

Python Sobol' and Halton sequences

R `randtoolbox` Sobol', lattice, and Halton sequences



Decisions to Make If We Want to Succeed

Key Elements

- Sequences—IID, Sobol', lattice, Halton, sparse grid, . . . , including randomization; fixed and extensible sample size and dimension; constructions using optimization



Decisions to Make If We Want to Succeed

Key Elements

- Sequences—IID, Sobol', lattice, Halton, sparse grid, . . . , including randomization; fixed and extensible sample size and dimension; constructions using optimization
- Sequence generators—with inputs d , coordinate indices, and index range



Decisions to Make If We Want to Succeed

Key Elements

- Sequences—IID, Sobol', lattice, Halton, sparse grid, . . . , including randomization; fixed and extensible sample size and dimension; constructions using optimization
- Sequence generators—with inputs d , coordinate indices, and index range
- Discrepancy measures—various kernels and domains



Decisions to Make If We Want to Succeed

Key Elements

- Sequences—IID, Sobol', lattice, Halton, sparse grid, . . . , including randomization; fixed and extensible sample size and dimension; constructions using optimization
- Sequence generators—with inputs d , coordinate indices, and index range
- Discrepancy measures—various kernels and domains
- Variable transformations to accommodate non-uniform distributions and domains other than the unit cube



Decisions to Make If We Want to Succeed

Key Elements

- Sequences—IID, Sobol', lattice, Halton, sparse grid, . . . , including randomization; fixed and extensible sample size and dimension; constructions using optimization
- Sequence generators—with inputs d , coordinate indices, and index range
- Discrepancy measures—various kernels and domains
- Variable transformations to accommodate non-uniform distributions and domains other than the unit cube
- Integrands, but some will come from external library, e.g., PDE solvers



Decisions to Make If We Want to Succeed

Key Elements

- Sequences—IID, Sobol', lattice, Halton, sparse grid, . . . , including randomization; fixed and extensible sample size and dimension; constructions using optimization
- Sequence generators—with inputs d , coordinate indices, and index range
- Discrepancy measures—various kernels and domains
- Variable transformations to accommodate non-uniform distributions and domains other than the unit cube
- Integrands, but some will come from external library, e.g., PDE solvers
- Integrators, including multilevel and multivariate decomposition methods



Decisions to Make If We Want to Succeed

Key Elements

- Sequences—IID, Sobol', lattice, Halton, sparse grid, . . . , including randomization; fixed and extensible sample size and dimension; constructions using optimization
- Sequence generators—with inputs d , coordinate indices, and index range
- Discrepancy measures—various kernels and domains
- Variable transformations to accommodate non-uniform distributions and domains other than the unit cube
- Integrands, but some will come from external library, e.g., PDE solvers
- Integrators, including multilevel and multivariate decomposition methods
- Stopping criteria



Decisions to Make If We Want to Succeed

Key Elements

- Sequences—IID, Sobol', lattice, Halton, sparse grid, . . . , including randomization; fixed and extensible sample size and dimension; constructions using optimization
- Sequence generators—with inputs d , coordinate indices, and index range
- Discrepancy measures—various kernels and domains
- Variable transformations to accommodate non-uniform distributions and domains other than the unit cube
- Integrands, but some will come from external library, e.g., PDE solvers
- Integrators, including multilevel and multivariate decomposition methods
- Stopping criteria
- Compelling use cases



Decisions to Make If We Want to Succeed

Key Elements

- Sequences—IID, Sobol', lattice, Halton, sparse grid, . . . , including randomization; fixed and extensible sample size and dimension; constructions using optimization
- Sequence generators—with inputs d , coordinate indices, and index range
- Discrepancy measures—various kernels and domains
- Variable transformations to accommodate non-uniform distributions and domains other than the unit cube
- Integrands, but some will come from external library, e.g., PDE solvers
- Integrators, including multilevel and multivariate decomposition methods
- Stopping criteria
- Compelling use cases
- Packages that display output in tables or plots



Decisions to Make If We Want to Succeed

Languages and Architectures

- Which one(s)? Python? C++?



Decisions to Make If We Want to Succeed

Languages and Architectures

- Which one(s)? Python? C++?
- How do we balance performance, developer time, and portability?



Decisions to Make If We Want to Succeed

Languages and Architectures

- Which one(s)? Python? C++?
- How do we balance performance, developer time, and portability?
- How will users connect the software with other software packages and environments?



Decisions to Make If We Want to Succeed

Languages and Architectures

- Which one(s)? Python? C++?
- How do we balance performance, developer time, and portability?
- How will users connect the software with other software packages and environments?
- How will parallel computing be supported?



Decisions to Make If We Want to Succeed

Good Development Practices

- Start small, with good skeleton



Decisions to Make If We Want to Succeed

Good Development Practices

- Start small, with good skeleton
- Version control on Git or equivalent



Decisions to Make If We Want to Succeed

Good Development Practices

- Start small, with good skeleton
- Version control on Git or equivalent
- Ownership of routines, updates require owners' approval



Decisions to Make If We Want to Succeed

Good Development Practices

- Start small, with good skeleton
- Version control on Git or equivalent
- Ownership of routines, updates require owners' approval
- Comprehensive tests run regularly



Decisions to Make If We Want to Succeed

Good Development Practices

- Start small, with good skeleton
- Version control on Git or equivalent
- Ownership of routines, updates require owners' approval
- Comprehensive tests run regularly
- Reasonable license



Decisions to Make If We Want to Succeed

Good Development Practices

- Start small, with good skeleton
- Version control on Git or equivalent
- Ownership of routines, updates require owners' approval
- Comprehensive tests run regularly
- Reasonable license
- Marketing on websites and at conferences



Bigger than One Research Group

- We have experience over the past 5 years developing GAIL²
- Our group has learned some of the discipline required to develop good software
- Our group does not have the capacity to tackle this whole project, and neither does your group
- A good software library should attract developers
- Let's leave a legacy to our community that goes beyond theorems and algorithms

²Choi, S.-C. T. *et al.* *GAIL: Guaranteed Automatic Integration Library (Versions 1.0–2.2)*. MATLAB software. 2013–2017. http://gailgithub.github.io/GAIL_Dev/.



We Must Weigh ...

Costs

Less time to prove theorems

Learning a new language

Compromise with other research groups

Time spent writing documentation and tests

Benefits

More impact for our theorems

Wider access and better performance for our code

More capable code than can be produced by one research group

Fewer bugs for those who use the code

Attract more qMC developers

Attract more qMC users

Happier funding agencies



Current Discussions

- If you are interested, whether you are a potential developer or user, let's talk



Current Discussions

- If you are interested, whether you are a potential developer or user, let's talk
- Determine what we can agree on as initial answers to the big questions



Current Discussions

- If you are interested, whether you are a potential developer or user, let's talk
- Determine what we can agree on as initial answers to the big questions
- Right now we are trying to focus on



Current Discussions

- If you are interested, whether you are a potential developer or user, let's talk
- Determine what we can agree on as initial answers to the big questions
- Right now we are trying to focus on
 - An initial language



Current Discussions

- If you are interested, whether you are a potential developer or user, let's talk
- Determine what we can agree on as initial answers to the big questions
- Right now we are trying to focus on
 - An initial language
 - An initial version control platform, and a model for collaboration



Current Discussions

- If you are interested, whether you are a potential developer or user, let's talk
- Determine what we can agree on as initial answers to the big questions
- Right now we are trying to focus on
 - An initial language
 - An initial version control platform, and a model for collaboration
 - A few initial routines



Current Discussions

- If you are interested, whether you are a potential developer or user, let's talk
- Determine what we can agree on as initial answers to the big questions
- Right now we are trying to focus on
 - An initial language
 - An initial version control platform, and a model for collaboration
 - A few initial routines
 - An initial use case



Current Discussions

- If you are interested, whether you are a potential developer or user, let's talk
- Determine what we can agree on as initial answers to the big questions
- Right now we are trying to focus on
 - An initial language
 - An initial version control platform, and a model for collaboration
 - A few initial routines
 - An initial use case
- We welcome you to join us. Email me to join our Google group



Elements of the Community Software—Discrete Distributions

```
classdef (Abstract) discreteDistribution
```

```
%Specifies and generates the components of  $a_n \sum_{i=1}^n w_i \delta_{x_i}(\cdot)$ 
```

```
properties (Abstract)
```

```
    distribData %information required to generate the distribution
```

```
    state %state of the generator
```

```
    nStreams
```

```
end
```

```
properties
```

```
    domain = [0 0; 1 1]; %domain of the discrete distribution,  $\mathcal{X}$ 
```

```
    domainType = 'box' %domain of the discrete distribution,  $\mathcal{X}$ 
```

```
    dimension = 2 %dimension of the domain,  $d$ 
```

```
    trueDistribution = 'uniform' %name of the distribution that the discrete ...  
        distribution attempts to emulate
```

```
end
```



Elements of the Community Software—Discrete Distributions

```
classdef (Abstract) discreteDistribution
methods (Abstract)
    genDistrib(obj, nStart, nEnd, n, coordIndex)
        % nStart = starting value of  $i$ 
        % nEnd = ending value of  $i$ 
        % n = value of  $n$  used to determine  $a_n$ 
        % coordIndex = which coordinates in sequence are needed
    end
end
```



Elements of the Community Software—Discrete IID Uniform

```
classdef IIDDistribution < discreteDistribution

%Specifies and generates the components of  $\frac{1}{n} \sum_{i=1}^n \delta_{\mathbf{x}_i}(\cdot)$ 
%where the  $\mathbf{x}_i$  are IID uniform on  $[0,1]^d$  or IID standard Gaussian
properties
    distribData %stream data
    state = [] %not used
    nStreams = 1
end

methods
    function obj = initStreams(obj,nStreams)
        obj.nStreams = nStreams;
        obj.distribData.stream = ...
            RandStream.create('mrg32k3a','NumStreams',nStreams,'CellOutput',true);
    end
end
```



Elements of the Community Software—Discrete IID Uniform

```

classdef IIDDistribution < discreteDistribution
    function [x, w, a] = genDistrib(obj, nStart, nEnd, n, coordIndex, ...
        streamIndex)
        if nargin < 6
            streamIndex = 1;
        end
        nPts = nEnd - nStart + 1; %how many points to be generated
        if strcmp(obj.trueDistribution, 'uniform') %generate uniform points
            x = ...
                rand(obj.distribData.stream{streamIndex}, nPts, numel(coordIndex)); ...
                %nodes
        else %standard normal points
            x = ...
                randn(obj.distribData.stream{streamIndex}, nPts, numel(coordIndex)); ..
                %nodes
        end
        w = 1;
        a = 1/n;
    end
end

```



Elements of the Community Software—Functions

```

classdef (Abstract) fun
% Specify and generate values  $f(\mathbf{x})$  for  $\mathbf{x} \in \mathcal{X}$ 
properties
    domain = [0 0; 1 1] %domain of the function,  $\mathcal{X}$ 
    domainType = 'box' %e.g., 'box', 'ball'
    dimension = 2 %dimension of the domain,  $d$ 
    distribType = 'uniform' %e.g., 'uniform', 'Gaussian'
    nominalValue = 0 %a nominal number,  $c$ , such that  $(c, \dots, c) \in \mathcal{X}$ 
end

methods (Abstract)
    y = f(obj, xu, coordIndex)
    % xu = nodes,  $\mathbf{x}_{u,i} = i^{\text{th}}$  row of an  $n \times |u|$  matrix
    % coordIndex = set of those coordinates in sequence needed,  $u$ 
    % y =  $n \times p$  matrix with values  $f(\mathbf{x}_{u,i}, \mathbf{c})$  where if  $\mathbf{x}'_i = (x_{i,u}, \mathbf{c})_j$ , then  $x'_{ij} = x_{ij}$  for
        %  $j \in u$ , and  $x'_{ij} = c$  otherwise
end

end

```



Elements of the Community Software—Keister's Function

```
classdef KeisterFun < fun
% Specify and generate values  $f(\mathbf{x})$  for  $\mathbf{x} \in \mathcal{X}$ 
methods
    function y = f(obj, x, coordIndex)
        %if the nominalValue = 0, this is efficient
        normx2 = sum(x.*x,2);
        if (numel(coordIndex)  $\neq$  obj.dimension) && (obj.nominalValue  $\neq$  0)
            normx2 = normx2 + (obj.nominalValue.^2) * (obj.dimension - ...
                numel(coordIndex));
        end
        y = exp(-normx2) .* cos(sqrt(normx2));
    end
end
end
end
```



Elements of the Community Software—StoppingCriteria

```
classdef (Abstract) stoppingCriterion
% Decide when to stop a
properties
    absTol = 1e-2 %absolute tolerance, d
    relTol = 0 %absolute tolerance, d
    nInit = 1024 %initial sample size
    nMax = 1e8 %maximum number of samples allowed
end
properties (Abstract)
    discDistAllowed %which discrete distributions are supported
    decompTypeAllowed %which decomposition types are supported
end
methods (Abstract)
    stopYet(obj, distribObj)
        % distribObj = data or summary of data computed already
    end
end
```



Elements of the Community Software—CLTStopping

```
classdef CLTStopping < stoppingCriterion
% Stopping criterion based on the Central Limit Theorem
properties
    discDistAllowed = "IIDDistribution" %which discrete distributions are ...
        supported
    decompTypeAllowed = ["single"; "multi"] %which decomposition types are ...
        supported
    inflate = 1.2 %inflation factor
    alpha = 0.01;
end

properties (Dependent)
    quantile
end
```




Elements of the Community Software—CLTStopping

```
classdef CLTStopping < stoppingCriterion
methods
    function [obj, dataObj, distribObj] = ...
        stopYet(obj, dataObj, funObj, distribObj)
        if ~numel(dataObj)
            dataObj = meanVarData;
        end
        switch dataObj.stage
```



Elements of the Community Software—CLTStopping

```
classdef CLTStopping < stoppingCriterion
    switch dataObj.stage
        case 'begin' %initialize
            dataObj.timeStart = tic;
            if ~any(strcmp(obj.discDistAllowed,class(distribObj)))
                error('Stoppoing criterion not compatible with sampling ...
                    distribution')
            end
            nf = numel(funObj); %number of functions whose integrals add up ...
                to the solution
            distribObj = initStreams(distribObj,nf); %need an IID stream ...
                for each function
            dataObj.prevN = zeros(1,nf); %initialize data object
            dataObj.nextN = repmat(obj.nInit,1,nf);
            dataObj.muhat = Inf(1,nf);
            dataObj.sighat = Inf(1,nf);
            dataObj.nSigma = obj.nInit; %use initial samples to estimate ...
                standard deviation
            dataObj.costF = zeros(1,nf);
            dataObj.stage = 'sigma'; %compute standard deviation next
```



Elements of the Community Software—CLTStopping

```
classdef CLTStopping < stoppingCriterion
    case 'sigma'
        dataObj.preVN = dataObj.nextN; %update place in the sequence
        tempA = sqrt(dataObj.costF); %use cost of function values to ...
            decide how to allocate
        tempB = sum(tempA .* dataObj.sighat); %samples for computation ...
            of the mean
        nM = ceil((tempB*(obj.quantile*obj.inflate ...
            /max(obj.absTol,dataObj.solution*obj.relTol))^2) ...
            * (dataObj.sighat./sqrt(dataObj.costF)));
        dataObj.nMu = min(max(dataObj.nextN,nM),obj.nMax - dataObj.preVN);
        dataObj.nextN = dataObj.nMu + dataObj.preVN;
        dataObj.stage = 'mu'; %compute sample mean next
```



Elements of the Community Software—CLTStopping

```
classdef CLTStopping < stoppingCriterion
    case 'mu'
        dataObj.solution = sum(dataObj.muhat);
        dataObj.nSamplesUsed = dataObj.nextN;
        errBar = (obj.quantile * obj.inflate) * ...
            sqrt(sum(dataObj.sighat.^2/dataObj.nMu));
        dataObj.errorBound = dataObj.solution + errBar*[-1 1];
        dataObj.stage = 'done'; %finished with computation
    end
    dataObj.timeUsed = toc(dataObj.timeStart);
end

function value = get.quantile(obj)
    value = -norminv(obj.alpha/2);
end

end
end
```



Elements of the Community Software—Integration

```
function [solution, dataObj] = integrate(funObj, distribObj, stopCritObj)
%Specify and generate values  $f(\mathbf{x})$  for  $\mathbf{x} \in \mathcal{X}$ 
% funObj = an object from class fun
% distribObj = an object from class discrete_distribution
% stopcritObj = an object from class stopping_criterion

%Initialize the accumData object and other crucial objects
[stopCritObj, dataObj, distribObj] = stopYet(stopCritObj, [], funObj, ...
    distribObj);
while ~strcmp(dataObj.stage, 'done') %the dataObj.stage property tells us ...
    where we are in the process
    dataObj = updateData(dataObj, distribObj, funObj); %compute additional data
    [stopCritObj, dataObj] = stopYet(stopCritObj, dataObj, funObj); %update ...
        the status of the computation
end
solution = dataObj.solution; %assign outputs
dataObj.timeUsed = toc(dataObj.timeStart);
```



Elements of the Community Software—Integration Example

```
%% How to integrate a function using our community QMC framework
% An example with Keister's function integrated with respect to the uniform
% distribution over the unit cube
stopObj = CLTStopping %stopping criterion for IID sampling using the ...
    Central Limit Theorem
distribObj = IIDDistribution; %IID sampling with uniform distribution
[sol, out] = integrate(KeisterFun, distribObj, stopObj)
```

```
>> IntegrationExample
sol = 0.4310
out =
timeUsed: 0.0014
nSamplesUsed: 7540
errorBound: [0.4210 0.4410]
```



Elements of the Community Software—Integration Example

```
stopObj.absTol = 1e-3 %decrease tolerance  
[sol, out] = integrate(KeisterFun, distribObj, stopObj)
```

```
sol = 0.4253  
out =  
timeUsed: 0.0333  
nSamplesUsed: 652546  
errorBound: [0.4243 0.4263]
```



Elements of the Community Software—Integration Example

```
stopObj.absTol = 0; %impossible tolerance
stopObj.nMax = 1e6; %calculation limited by sample budget
[sol, out] = integrate(KeisterFun, distribObj, stopObj)
```

```
sol = 0.4252
out =
timeUsed: 0.0392
nSamplesUsed: 1000000
errorBound: [0.4244 0.4260]
```




Elements of the Community Software—Asian Call, Low D

```
%A multilevel example of Asian option pricing
distribObj.trueDistribution = 'normal'; %Change to normal distribution
stopObj.absTol = 0.01; %increase tolerance
stopObj.nMax = 1e8; %pushing the sample budget back up
OptionObj = AsianCallFun(4) %4 time steps
[sol, out] = integrate(OptionObj, distribObj, stopObj)
```

```
OptionObj =
dimension: 4
sol = 6.1740
out =
timeUsed: 1.0517
nSamplesUsed: 5680546
errorBound: [6.1640 6.1840]
```



Elements of the Community Software—Asian Call, High D

```
OptionObj = AsianCallFun(64) %single level, 64 time steps  
[sol, out] = integrate(OptionObj, distribObj, stopObj)
```

```
OptionObj =  
AsianCallFun with properties:  
dimension: 64  
sol =6.2036  
out =  
meanVarData with properties:  
timeUsed: 25.1910  
nSamplesUsed: 5610402  
errorBound: [6.1936 6.2136]
```



Elements of the Community Software—Asian Call, Multi-Level

```
OptionObj = AsianCallFun([4 4 4]) %multilevel, 64 time steps, faster  
[sol, out] = integrate(OptionObj, distribObj, stopObj)
```

```
OptionObj =  
1×3 AsianCallFun array with properties:  
sol = 6.2052  
out =  
timeUsed: 2.2171  
nSamplesUsed: [8080862 446720 85907]  
errorBound: [6.1968 6.2135]
```

Thank you

Slides available on SpeakerDeck at
speakerdeck.com/fjhickernell/qmc-software-presentation-to-gail





Choi, S.-C. T. *et al.* *GAIL: Guaranteed Automatic Integration Library (Versions 1.0–2.2)*. MATLAB software. 2013–2017. http://gailgithub.github.io/GAIL_Dev/.