Guaranteed Automatic Integration Library (GAIL)—Before & After

Sou-Cheng (Terrya) Choi
Univ of Chicago/Argonne National Lab
Illinois Institute of Technology

Meshfree Methods Seminar
Illinois Institute of Technology
Aug 6, 2013
The GAIL Team

CEO: Fred Hickernell
Engineering Manager/Mislead: C
Release Lead: Lan Jiang

Documentation Lead: Yuhan Ding
Test Lead: Yizhi Zhang
Engineer: Rohan Sathe
Engineer: Xincheng Sheng
About **GAIL**—words from the CEO, May 6

- **Before GAIL**: Automatic numerical integration algorithms have inherent flaws in their error estimation based on balls of integrands.

- **GAIL** overcomes the flaws by considering cones of integrands. This allows us to construct upper bounds on costs of our integration routines with rigorous guarantees of accuracy and develop algorithms that provide the value of the integral with an error of no more than the user-defined tolerance.

- **Mission** (possible): To create a well-documented and well-tested library of univariate & multivariate integration routines that have rigorous guarantees
• **GAIL version 1**: By the end of the summer we hope to have our automatic routines for function recovery, univariate integration, and Monte Carlo estimation of mean on the GAIL site in good form, meaning that these routines need to be
  
  – well-documented
  
  – well-tested
  
  – optimized for speed
  
  – accompanied by examples
  
  – in a repository where they can be modified and re-tested as needed

• Later we will improve and add to these routines.
## Getting Things Done . . .

### DONE

<table>
<thead>
<tr>
<th>Owner</th>
<th>Description</th>
<th>Due Date</th>
<th>Complete</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALL</td>
<td>Second GAIL Team Meet @ 2p</td>
<td>May 21, 2013</td>
<td>✓</td>
</tr>
<tr>
<td>Fred</td>
<td>Push existing Matlab scripts to GAIL Dev website</td>
<td>May 20, 2013</td>
<td>✓</td>
</tr>
<tr>
<td>Lan</td>
<td>Push existing Matlab scripts for meanMC to GAIL Dev website</td>
<td>May 20, 2013</td>
<td>✓</td>
</tr>
<tr>
<td>Lan</td>
<td>Document on how to set up a development repository (rename files, etc)</td>
<td>May 20, 2013</td>
<td>✓</td>
</tr>
<tr>
<td>Lan</td>
<td>Remove May14th.m in GAIL_Dev. Rename “UniTests” to “UniTests”.</td>
<td>May 20, 2013</td>
<td>✓</td>
</tr>
<tr>
<td>Yuhai</td>
<td>Push existing Matlab scripts to GAIL Dev website</td>
<td>May 20, 2013</td>
<td>✓</td>
</tr>
<tr>
<td>Yuhai</td>
<td>Push existing Matlab scripts to GAIL Dev website</td>
<td>May 20, 2013</td>
<td>✓</td>
</tr>
<tr>
<td>Lan</td>
<td>Create top-two level directories in GAIL dev</td>
<td>May 17, 2013</td>
<td>✓</td>
</tr>
<tr>
<td>Lan</td>
<td>Create GAIL development repository</td>
<td>May 17, 2013</td>
<td>✓</td>
</tr>
<tr>
<td>ALL</td>
<td>First GAIL Team meet</td>
<td>May 13, 2013</td>
<td>✓</td>
</tr>
</tbody>
</table>

**Showing 125 items**

### TODO

<table>
<thead>
<tr>
<th>Owner</th>
<th>Description</th>
<th>Due Date</th>
<th>Complete</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALL</td>
<td>GAIL Release 1 Due</td>
<td>September 1, 2013</td>
<td></td>
</tr>
<tr>
<td>Yizhi</td>
<td>Replace tau as an input by n0 as an input in integral_g (after we finish the paper)</td>
<td>August 20, 2013</td>
<td></td>
</tr>
<tr>
<td>Yuhai</td>
<td>Replace tau as an input by n0 as an input in funapx_g (after we finish the paper)</td>
<td>August 20, 2013</td>
<td></td>
</tr>
<tr>
<td>Yizhi</td>
<td>Schedule your comprehensive exam</td>
<td>August 14, 2013</td>
<td></td>
</tr>
<tr>
<td>ALL</td>
<td>Prepare Aug 8th’s talk on Mushfree Seminar</td>
<td>August 8, 2013</td>
<td></td>
</tr>
<tr>
<td>ALL (Lan, Yizhi)</td>
<td>Revise help so that the simplest ways to call the function come first</td>
<td>August 8, 2013</td>
<td></td>
</tr>
<tr>
<td>Lan</td>
<td>Fix failed “doctest meanMC_g”</td>
<td>August 8, 2013</td>
<td></td>
</tr>
<tr>
<td>Lan</td>
<td>Fix bug in meanMCparam.m [Undefined 'n0/MinDefault' in line 45]</td>
<td>August 8, 2013</td>
<td></td>
</tr>
<tr>
<td>Lan</td>
<td>Create unit tests for cubMC_g</td>
<td>August 8, 2013</td>
<td></td>
</tr>
<tr>
<td>Lan</td>
<td>polish meanMC_g time budget to sample part to make it more robust</td>
<td>August 8, 2013</td>
<td></td>
</tr>
<tr>
<td>Lan</td>
<td>Failed unit tests for meanMC: results = run(ut_meanMC_g)</td>
<td>August 8, 2013</td>
<td></td>
</tr>
<tr>
<td>Lan</td>
<td>Complete help_meanMC_g.m and help_cubMC.m</td>
<td>August 8, 2013</td>
<td></td>
</tr>
<tr>
<td>Rohan</td>
<td>Check main approximation code</td>
<td>August 8, 2013</td>
<td></td>
</tr>
<tr>
<td>Xincheng</td>
<td>Look through the meanMC_g.m code</td>
<td>August 8, 2013</td>
<td></td>
</tr>
</tbody>
</table>
A “disintegrating” integral

Spiky \( f \) with \( I = \int_{0}^{1} f(x)dx \approx 0.3694. \)

\text{quad}(f, [0,1], 1e-14) = 0 \text{ giving error } = I.

Strategy: ↑ number of points to ↓ error

Q: How many number of points (or function evaluations), \( n \) in a quadrature rule do we need to guarantee that a given error tolerance, \( \epsilon \) is met?

A: Hickernell et al. 2012, Clancy et al. 2013, GAIL
## GAIL version 1 at a glance

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>Key developers</th>
<th>GAIL functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Function recovery</td>
<td>Yuhan Ding</td>
<td>funappx_g</td>
</tr>
<tr>
<td>Univariate integration</td>
<td>Yizhi Zhang</td>
<td>integral_g</td>
</tr>
<tr>
<td>Monte Carlo mean estimation</td>
<td>Lan Jiang</td>
<td>meanMC_g</td>
</tr>
</tbody>
</table>

Reminder: Thur (Aug 8) talk on GAIL by Yuhan, Yizhi, and Lan.
Guaranteed function approximation/recovery

(Clancy et al. 2013, Theorem 9)

Based on Balls. Not Adaptive

\[
\begin{align*}
  x & \mapsto f(x) \\
  \epsilon & = \text{tolerance} \\
  \|f''\|_{\infty} & \leq \sigma
  \end{align*}
\]

Linear spline to approximate \( f \)

\[
\text{cost} = \left\lceil \sqrt{\frac{\sigma}{8\epsilon}} \right\rceil + 1
\]

\[
A(f) := \text{approximation} \\
\|f - A(f)\|_{\infty} \leq \epsilon
\]

Based on Cones. Adaptive

\[
\begin{align*}
  x & \mapsto f(x) \\
  \epsilon & = \text{tolerance} \\
  \|f''\|_{\infty} & \leq \tau \|f' - f(1) + f(0)\|_{\infty}
  \end{align*}
\]

Linear spline to approximate \( f \)

\[
\text{cost} \lesssim \sqrt{\frac{\tau \|f''\|_{\infty}}{4\epsilon}}
\]

\[
A(f) := \text{approximation} \\
\|f - A(f)\|_{\infty} \leq \epsilon
\]

guaranteed
Guaranteed univariate integration

(Clancy et al. 2013, Theorem 7)

Based on Balls. Not Adaptive

$x \mapsto f(x)$

$\epsilon = \text{tolerance}$

$\text{Var}(f') \leq \sigma$

Trapezoidal rule

to approximate $\int_0^1 f(x)dx$

cost $= \lceil \sqrt{\frac{\sigma}{8\epsilon}} \rceil + 1$

$A(f) := \text{approximation}$

$| \int_0^1 f(x)dx - A(f) | \leq \epsilon$

guaranteed

Based on Cones. Adaptive

$x \mapsto f(x)$

$\epsilon = \text{tolerance}$

$\text{Var}(f') \leq \tau \| f' - f(1) + f(0) \|_1$

Trapezoidal rule

to approximate $\int_0^1 f(x)dx$

cost $\lesssim \sqrt{\frac{\tau \| f'' \|_1}{4\epsilon}}$

$A(f) := \text{approximation}$

$| \int_0^1 f(x)dx - A(f) | \leq \epsilon$

guaranteed
Guaranteed Monte Carlo mean estimation

(Hickernell et al. 2012, Theorem 5)

Based on Balls. Not Adaptive

\[
\begin{array}{lll}
\text{random variable } Y = f(X) & \text{MC sample mean} & A(Y) := \text{approximation} \\
\epsilon = \text{tolerance} & \text{to approximate } \mathbb{E}(Y) & \mathbb{P}(|\mathbb{E}(Y) - A(Y)| \leq \epsilon) \geq 1 - \alpha \\
\alpha = \text{uncertainty} = 5\% & \text{cost} = \left( \frac{1.96 \sigma_{\max}}{\epsilon} \right)^2 & \text{guaranteed} \\
\|f - \mathbb{E}(Y)\|_2^2 \leq \sigma_{\max}^2 & A(Y) := \text{approximation} \\
\end{array}
\]

Based on Cones. Adaptive

\[
\begin{array}{lll}
\text{random variable } Y = f(X) & \text{MC sample mean} & A(Y) := \text{approximation} \\
\epsilon = \text{tolerance} & \text{to approximate } \mathbb{E}(Y) & \mathbb{P}(|\mathbb{E}(Y) - A(Y)| \leq \epsilon) \geq 1 - \alpha \\
\alpha = \text{uncertainty} & \text{cost} = n\sigma & \text{guaranteed} \\
\|f - \mathbb{E}(Y)\|_4 \leq \tilde{\kappa}_{\max} \|f - \mathbb{E}(Y)\|_2 \\
\text{cost} = n\sigma + n(\epsilon, \sigma_{\max}, \alpha, \tilde{\kappa}_{\max})^3 & A(Y) := \text{approximation} \\
\end{array}
\]

\( \text{NOTE: } \tilde{\kappa}_{\max} = \text{upper bound on kurtosis} + 3, \text{ measure of "peakedness" } \)
Guaranteed framework

Solution operator $S : \mathcal{F} \to \mathcal{G}$

$(\mathcal{F}, \cdot \lvert \mathcal{F})$ stronger than $(\mathcal{F}, \cdot \lvert \tilde{\mathcal{F}})$, $(\mathcal{G}, \cdot \lvert \mathcal{G})$

Construct cone $C_\tau = \{ f \in \mathcal{F} : \lVert f \rVert_\mathcal{F} \leq \tau \lVert f \rVert_{\tilde{\mathcal{F}}} \}$

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>$\mathcal{F}$</th>
<th>$\mathcal{G}$</th>
<th>$\lVert f \rVert_\mathcal{F}$</th>
<th>$\lVert f \rVert_{\tilde{\mathcal{F}}}$</th>
<th>$\lVert g \rVert_\mathcal{G}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Function recovery</td>
<td>$\mathcal{W}^{2,\infty}[0, 1]$</td>
<td>$\mathcal{L}_\infty$</td>
<td>$\lVert f'' \rVert_\infty$</td>
<td>$\lVert f' + f(1) - f(0) \rVert_\infty$</td>
<td>$\lVert g \rVert_\infty$</td>
</tr>
<tr>
<td>Univariate integration</td>
<td>$\mathcal{V}^1$</td>
<td>$\mathcal{R}$</td>
<td>$\text{Var}(f')$</td>
<td>$\lVert f' + f(1) - f(0) \rVert_1$</td>
<td>$\lVert g \rVert_1$</td>
</tr>
<tr>
<td>MC mean estimation</td>
<td>$\mathcal{L}_4$</td>
<td>$\mathcal{R}$</td>
<td>$\lVert f - \mathbb{E}(Y) \rVert_4$</td>
<td>$\lVert f - \mathbb{E}(Y) \rVert_2$</td>
<td>$\lVert g \rVert_1$</td>
</tr>
</tbody>
</table>
The advantage of the Cone for function recovery

- Pick seminorm $\| \cdot \|_{W^{1,\infty}}$, which is weaker than $\| \cdot \|_{W^{2,\infty}}$. Also assume that $f$ is not too spiky and lies in a cone, i.e., $af$ also in the cone $\forall a > 0$:

$$2\| f' - f(1) + f(0) \|_\infty \leq \| f'' \|_\infty \leq \tau \| f' - f(1) + f(0) \|_\infty$$

Linear $f$ vanishes weaker cone

- Use data-driven $V_n(f) \equiv \| (A(f))' - f(1) + f(0) \|_\infty$ (real $A(f)$ is linear spline) to reliably bound $\| f' - f(1) + f(0) \|_\infty$ and $\| f'' \|_\infty$:

$$0 \leq \| f' - f(1) + f(0) \|_\infty - V_n(f) \leq \frac{\| f'' \|_\infty}{2(n-1)^2} \leq \frac{\tau \| f' - f(1) + f(0) \|_\infty}{2(n-1)^2}$$

$$\hat{V}_n(f) \leq \| f'' \|_\infty \leq \tau \| f' - f(1) + f(0) \|_\infty \leq \frac{2\tau V_n(f)}{2 - \frac{1}{n} - \frac{\tau}{n}} \leq \frac{\tau \| f'' \|_\infty}{2 - \frac{1}{n} - \frac{\tau}{n}}$$

- Cost $n$ needed to meet tolerance $\epsilon$ satisfies the data-driven criterion:

$$\frac{\tau V_n(f)}{4(n-1)(2n-2-\tau)} \leq \epsilon$$

- $\max \left( \left\lceil \frac{\tau+1}{2} \right\rceil, \left\lceil \sqrt{\frac{\| f'' \|_\infty}{8\epsilon}} \right\rceil \right) + 1 \leq n := \text{cost (#function values) of algorithm } A(f) \text{ for } f \text{ in cone with unknown } \| f'' \|_\infty \leq \sqrt{\frac{\tau \| f'' \|_\infty}{4\epsilon}} + \tau + 4$
Reproducible research (RR) pioneers and champions

Jon Claerbout
SEP (Stanford Exploration Project, 1973–present)

David Donoho
What is *reproducible research (RR)*?

- Claerbout, “The markings [ER], [CR], and [NR] are promises by the author(s) about the reproducibility of each figure result.” (url: j.mp/VM7Xq4)
  - ER: Easily reproducible: “programs, parameters, and makefiles . . . data”
  - CR: Conditionally reproducible: “processing requires 20 minutes or more, or commercial packages”
  - NR: Not reproducible: drawings
  - M: Movie “in a figure”; ER, CR, or NR

- Donoho paraphrasing Claerbout, “an article about computational result is *advertising*, not scholarship. The actual *scholarship* is the full *software environment, code* and *data*, that produced the result.” (Buckheit & Donoho 1995)
Figure 1: The dipping planes synthetic model. Although a trivial flattening test case, the boundaries of the divergence of the dip are not periodic.
Directory Structure of GAIL (Release 1)

- **Algorithms/** – Automatic routines for univariate integration, Monte Carlo estimation of the mean, and function recovery

- **Datasets/** – Input data for scripts in **Papers/** and other directories

- **Documentation/** – System-wide HTML documentation

- **Papers/** – Demos and scripts for reproducing figures in published articles

- **UnitTests/** – For unit test and doctest scripts

- **Utilities/** – General tools for developers and users

- **Workouts/** – Scripts giving BeamLab features a workout

- **ThirdParty/** – Matlab tools/packages with open-source licenses, but not produced by the GAIL team
Online Survey: Reproducibility in Computational Science and Engineering (CSE)

- 13 questions on opinions and experiences concerning the reproducibility of computational results.
- Results collected on August 1.

Direct Emails
- Call went out on July 5 to ~ 500 addresses.
- Resulted in ~ 80 answers.

InterPore Newsletter
- Newsletter was sent out on July 6 to ~ 1000 addresses.
- Resulted in 2 answers.

SIAM Activity Group on CSE Mailing List
- Call went out on July 10 to ~ 2000 addresses.
- Resulted in ~ 300 answers.

Survey Results I (n = 385)

- I understand what the reproducibility of computational results means:
  - No
  - Yes
  - The estimation of the effort to reproduce does not influence the estimation of the importance of reproducibility.

- I consider the reproducibility of computational results to be:
  - Not important
  - Very important
  - The effort estimated for oneself influences the effort estimated for others, and the effort for the others is considered to be higher.

- The importance ... is sufficiently reflected by today’s journal policies:
  - I don’t know
  - Yes
  - The estimated effort to reproduce does not influence the number of employed strategy items.

- I already had problems with reproducing computational results of my own/other's results:
  - Yes
  - No
  - The amount of work related to coding influences the estimation of the importance, but not the number of employed strategy items.

- I don’t know
  - No
  - The effort it would take me/other's to reproduce my computational results:
    - I don’t know
    - Very little
    - Very high
  - Compared to the average, I think that I invest more effort in ensuring reproducibility:
    - Less
    - More
  - The ratio of working hours I devote to coding (including thinking and talking about coding):
    - 0 – 20%
    - 21 – 40%
    - 41 – 60%
    - 61 – 80%
    - 81 – 100%
  - The effort it would take me to reproduce my computational results from three years ago:
    - Very high
    - Very little

Survey Results II (n = 385)

- My current education/position:
  - Other
  - BSc/MSc student
  - PhD student
  - Postdoc
  - Professor

- My age in years:
  - ≤ 20
  - 21 – 30
  - 31 – 40
  - 41 – 50
  - 51 – 60
  - > 60

- The effort it would take me/other's to reproduce my computational results:
  - I don’t know
  - Yes

Survey Results: A Slightly Deeper Look

- The estimation of the effort to reproduce does not influence the estimation of the importance of reproducibility.
- The effort estimated for oneself influences the effort estimated for others, and the effort for the others is considered to be higher.
- The estimated effort to reproduce does not influence the number of employed strategy items.
- The amount of work related to coding influences the estimation of the importance, but not the number of employed strategy items.
- Age does not have an influence on the quantitative results, apart from the time devoted to coding.
How reliable or limiting is RR?

- As much as the underlying theory, data, code, and software
- Platform and version dependent
- Less over time due to new software versions
- Commercial software expensive—Cf. quality free software
- Loss of data or code due to lack of frequent backup
- Big data inputs hard to clean, outputs manually verified, corruptable by sharing or transferring processes
- Big code binary, poor design or documentation, slow to run
- Lack testing
- No bug-fix patches or slow new releases
- Lack communication, community support & feedback
- Hamper creativity and/or productivity?
Reliable RR via Stauncheon Scientific Software (SSS)

SSS heavyweights

Richard Stallman
GNU, Free software movement

Ian Foster (v.)
Father of the Grid, Globus Online, Galaxy, SWIFT, CIM-EARTH
Scientific research vs. Software engineering

Sciences:
- Wavelets
- Signal processing
- Image processing
- Biostatistics
- PDEs
- Economics
- Physics
- Chemistry
- Mathematics
- Data structure
- Algorithms
- Monte Carlo simulation
- Numerical integration

Software:
- Test-driven development
- Object oriented design
- API and GUI
- Software reuse
- Logging, error handling
- Paired programming
- Nightly build
- Client-server functionality tests
- Load testing
- Documentation
- Continuous release
- Research project websites
- Licenses (BSD), copyleft
GAIL team working with a remote, central repository

Lan initialized GAIL-dev repository

Everybody cloned the repository

Yizhi worked on his local repository

Yuhan worked on her local repository

Yizhi published his code

Yuhan could not publish her code

Yuhan pulled Yizhi’s commits

Yuhan resolved merge conflicts

Yuhan published her code

Image credit: https://www.atlassian.com/git/workflows
Matlab/Octave software development/analysis tools

- Matlab 2013 Unit Testing Framework
- Matlab Central (url: j.mp/anwdaP)
- Matlab mex to interface with C/Fortran functions
- Matlab reports:
GAIL’s Application Programming Interfaces (APIs)

Our key algorithms have three API patterns:

\[
\begin{align*}
[x, \text{out}_{-}\text{param}] &= \text{algo}_g(f, \text{in}_{-}\text{param}); \\
[x, \text{out}_{-}\text{param}] &= \text{algo}_g(f, \text{inputVal1}, \text{inputVal2}, \text{inputVal3}); \\
[x, \text{out}_{-}\text{param}] &= \text{algo}_g(f, 'input2', \text{inputVal2}, 'input3', \text{inputVal3});
\end{align*}
\]

\(f\): compulsory, function handle  
\(\text{in}_{-}\text{param}, \text{out}_{-}\text{param}\): optional, structures  
\(\text{input1,}\ldots\): optional, numeric

Automatically correct out-of-range input values
GAIL home directory & help

```
>> help GAIL_Matlab
GAIL_MATLAB

Files
GAILstart - Initialize all the GAIL global parameters
GAIL_Install - Install GAIL
GAIL_Reinstall - Reinstall GAIL
GAIL_Uninstall - Uninstall GAIL

Folders
Algorithms - GAIL algorithms
Datasets - Dataset related with GAIL
Documentation - Document related with GAIL
OutputFiles - Output result
Papers - Papers and slides related with GAIL
ThirdParty - Open-source tools used but not produced by GAIL team
UnitTests - Unit tests of GAIL algorithms
Utilities - Tools for the GAIL package
Workouts - Workouts of GAIL algorithms
```
FUNAPPX_G One dimensional guaranteed function recovery on the interval

\[ f_{appx} = \text{FUNAPPX}_G(f) \text{ recovers function } f \text{ on the interval } [0,1] \text{ by a piecewise linear interpolant } f_{appx} \text{ to within a guaranteed absolute error of } 1e^{-6}. \text{ Default cone constant is 10 and default cost budget is } 1e7. \text{ Input } f \text{ a function handle. The function } Y=f(X) \text{ should accept a vector argument } X \text{ and return a vector result } Y, \text{ the integrand evaluated at each element of } X. \]

\[ f_{appx} = \text{FUNAPPX}_G(f,\text{tol},\tau,\text{Nmax}) \text{ for given function } f \text{ and the ordered input parameters with the guaranteed absolute error } \text{tol}, \text{ cone condition } \tau, \text{ and cost budget } \text{Nmax}. \]

\[ f_{appx} = \text{FUNAPPX}_G(f,\text{'tol'},\tau,\text{'Nmax'},\text{Nmax}) \text{ recovers function } f \text{ with the guaranteed absolute error } \text{tol}, \text{ cone constant } \tau, \text{ and cost budget } \text{Nmax}. \text{ All three filed-value pairs are optional and can be supplied in different order.} \]

\[ f_{appx} = \text{FUNAPPX}_G(f,\text{in_param}) \text{ recovers function } f \text{ with the guaranteed absolute error } \text{in_param.tol}, \text{ cone constant } \text{in_param.tau}, \text{ and cost budget } \text{in_param.Nmax}. \text{ If a field is not specified, the default value is used.} \]

\[ \text{in_param.tol} --- \text{ guaranteed absolute error, default value is } 1e^{-6}. \]

(More content, omitted)
funappx_g

One dimensional guaranteed function recovery on the interval \([0,1]\).

Contents
- Syntax
- Description
- Examples
- See Also
- Reference

Syntax

\[
fa = \text{funappx}_g(t)
\]

\[
fa = \text{funappx}_g(t,\text{tol},\text{tau},\text{Nmax})
\]

\[
fa = \text{funappx}_g(t,\text{tol},\text{tol}',\text{tau}',\text{tau'},\text{'Nmax'},\text{Nmax})
\]

\[
fa = \text{funappx}_g(t,\text{in}_\text{param})
\]

\[
\{fa,\text{out}_\text{param}\} = \text{funappx}_g(t,...)
\]

Description

\[
fa = \text{funappx}_g(f)\] recovers function \(f\) on the interval \([0,1]\) by a piecewise linear interpolant \(fappx\) to within a guaranteed absolute error of 1e-6. Default cone constant is 10 and default cost budget is 1e6. Input \(f\) a function handle. The function \(Y=f(X)\) should accept a vector argument \(X\) and return a vector result \(Y\), the integrand evaluated at each element of \(X\).

\[
fa = \text{funappx}_g(f,\text{tol},\text{tau},\text{Nmax})\] for given function \(f\) and the ordered input parameters with the guaranteed absolute error \(\text{tol}\), cone condition \(\text{tau}\), and cost budget \(\text{Nmax}\).

\[
fa = \text{funappx}_g(f,\text{tol},\text{tol}',\text{tau}',\text{tau'},\text{'Nmax'},\text{Nmax})\] recovers function \(f\) with the guaranteed absolute error \(\text{tol}\), cone constant \(\text{tau}\), and cost budget \(\text{Nmax}\). All three fixed-value pairs are optional and can be supplied in different order.

Credits: Yuhan
GAIL doctest for documentation examples

Examples

Example 1

```matlab
format short; f = @(x) x.^2; [fappx, out_param] = funappx_g(f)

% Approximate function x^2 with default input parameter to make the error
% less than 1e-6.

fappx = @(x)interp1([x1, y1], x, 'linear')

out_param =
    tol: 1.0000e-06
    tau: 10
    Nmax: 1000000
    warning: 1
tauchange: 0
evaluation: 2.0000e-07
spoints: 2003
```

```bash
g >> doctest funappx_g
TAP version 13
1..6
ok 1 - "format short; f = @(x) x.^2; [fappx, out_param] = funappx_g(f)"
ok 2 - "clear in_param; format short; in_param.tol = 10^-8;"
ok 3 - "in_param.tau = 15; in_param.Nmax = 10^6;"
ok 4 - "[fappx, out_param] = funappx_g(@(x) x.^2, in_param)"
ok 5 - "format short; f = @(x) x.^2;"
ok 6 - "[fappx, out_param] = funappx_g(f,'tau',15,'Nmax',1e6,'tol',1e-8)"
```

Example 2

```matlab
format short; [fappx, out_param] = funappx_g(@(x) x.^2, 'tau', 15, 'Nmax', 1e6, 'tol', 1e-8)

% Approximate function x^2 with tolerance 1e-6, cost budget 100000
% and cone condition 15

fappx = @(x)interp1([x1, y1], x, 'linear')

out_param =
    f: @(x) x.^2
    Nmax: 1000000
    tau: 15
    tol: 1.0000e-08
    warning: 1
tauchange: 0
evaluation: 1.3333e+03
spoints: 25633
```
Matlab unit tests

Every time the code is changed, unit tests are run.
Every time a bug is found, unit tests are expanded.

>> results = run(ut_funappx_g)
Running ut_funappx_g
....
Done ut_funappx_g

results =

1x4 TestResult array with properties:

Name    Passed    Failed    Incomplete    Duration

Totals:
  4 Passed, 0 Failed, 0 Incomplete.
0.10896 seconds testing time.

classdef ut_funappx_g < matlab.unittest.TestCase

    methods(Test)

    function funappx_g0fx(testCase)
        f = @(x) x;
        in_param.tol = 10^(-8);
        in_param.tau = 15;
        in_param.Nmax = 10^6;
        [appxf, result] = funappx_g(f,in_param);
        x = sqrt(2)-1;
        actualerr = abs(appxf(x)-f(x));
        testCase.verifyLessThanOrEqual...
          (actualerr,in_param.tol);
        testCase.verifyLessThanOrEqual...
          (result.npoint,in_param.Nmax);
    end

(More test cases, omitted)
Measuring and Improving Performance

- Matlab Profiler
- Matlab Mex
- Parallelization with SWIFT
GAIL Central, http://code.google.com/p/gail/

About Guaranteed Automatic Integration Library (GAIL)

GAIL is a suite of algorithms for integration problems in one, many, and infinite dimensions, and whose answers are guaranteed to be correct.

GAIL is created, developed, and maintained by Fred Hickernell (Illinois Institute of Technology), Sou-Cheng Choi (University of Chicago, Argonne National Laboratory, and IIT), and their collaborators including Yuhan Ding (IIT), Lan Jiang (IIT), and Yizhi Zhang (IIT).

The first release of GAIL is scheduled for fall 2013.

Getting Started


Sneak Preview

- Aug 8: Seminar by Yuhan, Yizhi, and Lan
- Sep 1: Release of GAIL version 1
- Sep 2: Development of GAIL version 2 commences
- Fall: Elective “Reliable Mathematical Software”
- Sep 2014: Release of GAIL version 2
- Sep 2015: Release of GAIL version 3
Conclusions

1. RR is a philosophy that also serves as a practical guide for producing quality scholarly publications in computational sciences

2. SSS involves a number of software engineering principles and tools that enable experiments and actions. Open source plays a key role.

3. RR is made more reliable by SSS development

4. Numerical quadrature with a guarantee of accuracy and certainty is new and worthy

5. GAIL is to demonstrate the theorems and make an impact

6. Eventually and ideally, to be introduced into first-class first-course NA/SC courses

7. Handling and producing data/code/software taken to the extreme are challenging research and practical problems

8. HPC, mobile apps, industrial employment
“God does not care about our mathematical difficulties. He integrates empirically.” —Albert Einstein

Thank you!