Efficient Importance Sampling

David N. DeJong
University of Pittsburgh

Spring 2008
Our goal is to calculate integrals of the form

$$G(Y) = \int_{\Theta} \phi(\theta; Y) \, d\theta.$$ 

Special case (e.g., posterior moment):

$$G(Y) = \int_{\Theta} \phi(\theta; Y) \, p(\theta|Y) \, d\theta.$$ 

Scenario: analytical solutions to these integrals are unavailable. We will remedy this problem using numerical approximation methods.
Overview, cont.

In the context of achieving likelihood evaluation and filtering in state space representations, recall that we face the challenge of constructing the filtering density

$$f(s_t | Y_t) = \frac{f(y_t, s_t | Y_{t-1})}{f(y_t | Y_{t-1})} = \frac{f(y_t | s_t, Y_{t-1}) f(s_t | Y_{t-1})}{f(y_t | Y_{t-1})},$$

where

$$f(s_t | Y_{t-1}) = \int f(s_t | s_{t-1}, Y_{t-1}) f(s_{t-1} | Y_{t-1}) \, ds_{t-1},$$

and

$$f(y_t | Y_{t-1}) = \int f(y_t | s_t, Y_{t-1}) f(s_t | Y_{t-1}) \, ds_t.$$
Overview, cont.

To gain intuition, suppose the integral we face is of the form

$$G(Y) = \int_{\Theta} \phi(\theta; Y) p(\theta|Y) \, d\theta,$$

and it is possible to obtain pseudo-random drawings $\theta_i$ from $p(\theta|Y)$. Then by the law of large numbers,

$$\overline{G(Y)}_N = \frac{1}{N} \sum_{i=1}^{N} \phi(\theta_i; Y)$$

converges in probability to $G(Y)$. We refer to $\overline{G(Y)}_N$ as the **Monte Carlo** estimate of $G(Y)$. As we shall see, from the standpoint of **numerical efficiency**, this represents a **best-case scenario**.
The MC estimate of the standard deviation of $G(Y)$ is given by

$$
\bar{\sigma}_N (G(Y)) = \left[ \left( \frac{1}{N} \sum_{i}^{N} \phi (\theta_i; Y)^2 \right) - \overline{G(Y)}_N^2 \right]^{1/2}.
$$

The numerical standard error associated with $\overline{G(Y)}_N$ is given by

$$
s.e. \left( \overline{G(Y)}_N \right) = \frac{\bar{\sigma}_N (G(Y))}{\sqrt{N}}.
$$

Thus for $N = 10,000$, $s.e. \left( \overline{G(Y)}_N \right)$ is 1% of the size of $\bar{\sigma}_N (G(Y))$. 
Importance Sampling (Geweke, 1989 Econometrica)

If \( p(\theta|Y) \) is unavailable as a sampler, one remedy is to augment the targeted integrand with an importance sampling distribution \( g(\theta|a) \):

\[
G(Y) = \int_{\Theta} \frac{\phi(\theta; Y)}{g(\theta|a)} g(\theta|a) d\theta
= \int_{\Theta} \frac{\phi(\theta; Y) p(\theta|Y)}{g(\theta|a)} g(\theta|a) d\theta.
\]

Key requirements:

- Support of \( g(\theta|a) \) must span that of \( \phi(\theta|Y) \)
- \( E[G(Y)] \) must exist and be finite.
- \( g(\theta|a) \) must be implementable as a sampler
Importance Sampling, cont.

- **MC estimate of** $G(Y)$:
  \[
  \overline{G(Y)}_N = \frac{1}{N} \sum_{i=1}^{N} \omega_i ; \quad \omega_i = \frac{\varphi(\theta_i|Y)}{g(\theta_i|a)}
  \]

- **MC estimate of the standard deviation of** $\omega$ w.r.t. $g(\theta|a)$:
  \[
  \overline{\sigma}_N (\omega(\theta,Y)) = \left[ \left( \frac{1}{N} \sum \omega_i^2 \right) - \overline{G(Y)}_N^2 \right]^{1/2}
  \]

- **Numerical standard error associated with** $\overline{G(\theta,Y)}_N$:
  \[
  s.e. \left( \overline{G(Y)}_N \right)_l = \frac{\overline{\sigma}_N (\omega(\theta,Y))}{\sqrt{N}}
  \]

- **Note:** Variability in $\omega$ translates into increased n.s.e. (numerical inefficiency)
Importance Sampling, cont.

For the special case in which the integrand factorizes as

\[ \varphi (\theta_i | Y) = \phi (\theta; Y) p (\theta | Y), \]

\[
G (Y) = \frac{1}{N} \sum_{i=1}^{N} \varphi (\theta_i; Y) w_i, \quad w_i = \frac{p (\theta_i | Y)}{g (\theta_i | a)}
\]

\[
\overline{\sigma}_N (G(Y)) = \left[ \left( \frac{1}{N} \sum_{i}^{N} \phi (\theta_i; Y)^2 w_i \right) - \overline{G(Y)}^2 \right]^{1/2}.
\]
Importance Sampling, cont.

Continuing with the special case, if we lack an integrating constant for either \( p(\theta|Y) \) or \( g(\theta|a) \), we can work instead with

\[
\overline{G(Y)}_N = \left( \frac{1}{\sum_{i=1}^{N} w_i} \right) \sum_{i=1}^{N} \phi(\theta_i; Y) w_i
\]

\[
\overline{\sigma}_N(G(Y)) = \left[ \left( \frac{1}{\sum_{i=1}^{N} w_i} \right) \sum_{i=1}^{N} \phi(\theta_i; Y)^2 w_i \right]^{1/2} - \overline{G(Y)}_N^2
\]

The impact of ignoring integrating constants is eliminated by the inclusion of the accumulated weights in the denominators of these expressions.
Regardless of whether $p(\theta|Y)$ and $g(\theta|a)$ are proper p.d.f.s, notice that when $p(\theta|Y)$ itself can be used as a sampling density (i.e., $p(\theta|Y) = g(\theta|a)$), then $w_i = 1 \ \forall \ i$, and we revert to the best-case scenario.

In general, the accuracy of our approximation will fall short of the best-case scenario. To judge the degree of the shortfall, various metrics are available.
Importance Sampling, cont.

Two metrics for judging numerical accuracy:

1. \( \frac{w_{\text{max}}}{\sum w_i^2} \) (good practical benchmark: 1%)
2. Relative numerical efficiency (RNE)
Importance Sampling, cont.

Motivation for RNE. Issue: how close are we to the best-case scenario?

- Under the best-case scenario, recall that n.s.e. is given by
  \[
  \text{s.e. } \left( \frac{G(Y)}{N} \right) = \frac{\bar{\sigma}_N(G(Y))}{\sqrt{N}}.
  \]

- Actual n.s.e. is given by
  \[
  \text{s.e. } \left( \frac{G(Y)}{N} \right)_I = \frac{\bar{\sigma}_N(\omega(\theta, Y))}{\sqrt{N}},
  \]

where recall

\[
\bar{\sigma}_N(\omega(\theta, Y)) = \left[ \left( \frac{1}{N} \sum \omega^2_i \right) - \frac{G(Y)^2}{N} \right]^{1/2}, \quad \omega_i = \frac{\varphi(\theta_i|Y)}{g(\theta_i|a)}
\]
The idea behind RNE is to compare the actual n.s.e. to an estimate of the optimal (best-case) n.s.e.:

\[
RNE = \frac{(\text{ideal n.s.e.})^2}{(\text{actual n.s.e.})^2} = \frac{\left(\frac{\sigma_N(G(Y))}{\sqrt{N}}\right)^2}{\left(s.e. \left(\overline{G(Y)}_N\right)_I\right)^2}.
\]
Importance Sampling, cont.

Rearranging yields

$$s.e. \left( \frac{G \left( Y \right) \_N}{N} \right)_I = \frac{\sigma \_N \left( G \left( Y \right) \right)}{\sqrt{N \cdot RNE}}.$$ 

Note: relative to the best-case scenario, $\sqrt{N \cdot RNE}$ replaces $\sqrt{N}$ in the denominator. Thus the further is RNE from 1, the more draws are required to achieve a given level of accuracy relative to the best-case scenario.
Example

Suppose $p(\theta|Y) \sim N(\mu, \Sigma)$,

$$
\mu = \begin{bmatrix}
1 \\
2 \\
3 \\
4 \\
5
\end{bmatrix}, \quad corr(\Sigma) = \begin{bmatrix}
1 & 0.6 & 0 & 0 & 0 \\
0.6 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & -0.8 \\
0 & 0 & 0 & -0.8 & 1
\end{bmatrix},
$$

$$
\sqrt{\text{diag}(\Sigma)^{'}} = \begin{bmatrix}
2 \\
0.2 \\
5 \\
1 \\
0.1
\end{bmatrix}
$$

Statistics of interest:

$$
E(\text{sumc}(\mu)), \quad E(\text{prodc}(\mu))
$$
Example, cont.

Using \texttt{rndseed 123456789}, and \( N = 10,000 \), MC estimates \((\hat{\mu}, \sqrt{\text{diag}(\Sigma)})\), n.s.e.(\(\hat{\mu}\)):

\[
\begin{array}{ccc}
1.0077 & 1.9664 & 0.0197 \\
2.0031 & 0.2007 & 0.0020 \\
2.9157 & 4.9899 & 0.0499 \\
4.0041 & 0.9793 & 0.0098 \\
4.9993 & 0.0986 & 0.0010
\end{array}
\]
Example, cont.

MC estimate of $corr(\Sigma)$:

\[
\begin{array}{cccccc}
1.0000 & 0.6060 & 0.0044 & -0.0139 & 0.0107 \\
0.6060 & 1.0000 & 0.0049 & -0.0048 & 0.0016 \\
0.0044 & 0.0049 & 1.0000 & -0.0013 & -0.0110 \\
-0.0139 & -0.0048 & -0.0013 & 1.0000 & -0.7969 \\
0.0107 & 0.0016 & -0.0110 & -0.7969 & 1.0000 \\
\end{array}
\]
Example, cont.

MC estimates of statistics (mean, std. dev., n.s.e.(mean)):

\[
\begin{align*}
E(\text{sumc}(\mu)) & \quad 14.9299 \quad 5.4883 \quad 0.0549 \\
E(\text{prodc}(\mu)) & \quad 130.7018 \quad 533.3792 \quad 5.3338
\end{align*}
\]
Example, cont.

Exercise: replicate

Hint for exercise.
To obtain draws from $N(\mu, \Sigma)$ distribution:

- `swish = chol(sig)'`; swish is lower-diagonal Cholesky decomposition of sig (i.e., $\text{sig} = \text{swish} \times \text{swish}'$).
- `draw = mu + swish*rndn(n,1);`
Example, cont.

Suppose instead we seek to obtain estimates using an Importance Sampling density

\[ g(\theta | a) \sim N(\mu_I, \Sigma_I), \]

with

\[
\begin{align*}
\mu_I &= \mu + 1.5 \cdot \sqrt{\text{diag}(\Sigma)}, \\
\Sigma_I &= \text{diagrv}(\Sigma, \sqrt{\text{diag}(\Sigma)})
\end{align*}
\]
Example, cont.

Using `rndseed 123456789`, and \( N = 10,000 \), MC versus IS estimates \((\hat{\mu}, \sqrt{\text{diag}(\Sigma)})\), n.s.e.(\(\hat{\mu}\)):

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0077</td>
<td>1.9664</td>
<td>0.0197</td>
</tr>
<tr>
<td>2.0031</td>
<td>0.2007</td>
<td>0.0020</td>
</tr>
<tr>
<td>2.9157</td>
<td>4.9899</td>
<td>0.0499</td>
</tr>
<tr>
<td>4.0041</td>
<td>0.9793</td>
<td>0.0098</td>
</tr>
<tr>
<td>4.9993</td>
<td>0.0986</td>
<td>0.0010</td>
</tr>
<tr>
<td>1.1646</td>
<td>1.5813</td>
<td>0.4140</td>
</tr>
<tr>
<td>1.9884</td>
<td>0.1721</td>
<td>0.0572</td>
</tr>
<tr>
<td>3.1161</td>
<td>3.5692</td>
<td>0.6526</td>
</tr>
<tr>
<td>4.2545</td>
<td>0.8731</td>
<td>0.2691</td>
</tr>
<tr>
<td>4.9947</td>
<td>0.0803</td>
<td>0.0173</td>
</tr>
</tbody>
</table>
Example, cont.

MC versus IS estimate of \( \text{corr}(\Sigma) \):

\[
\begin{array}{cccccc}
1.0000 & 0.6060 & 0.0044 & -0.0139 & 0.0107 \\
0.6060 & 1.0000 & 0.0049 & -0.0048 & 0.0016 \\
0.0044 & 0.0049 & 1.0000 & -0.0013 & -0.0110 \\
-0.0139 & -0.0048 & -0.0013 & 1.0000 & -0.7969 \\
0.0107 & 0.0016 & -0.0110 & -0.7969 & 1.0000 \\
\end{array}
\]

\[
\begin{array}{cccccc}
1.0000 & 0.6712 & 0.0623 & -0.2506 & -0.2307 \\
0.6712 & 1.0000 & 0.2031 & -0.2387 & 0.0065 \\
0.0623 & 0.2031 & 1.0000 & -0.1987 & 0.2975 \\
-0.2506 & -0.2387 & -0.1987 & 1.0000 & -0.6440 \\
-0.2307 & 0.0065 & 0.2975 & -0.6440 & 1.0000 \\
\end{array}
\]
Example, cont.

MC versus estimates of statistics (mean, std. dev., n.s.e.(mean)):

<table>
<thead>
<tr>
<th></th>
<th>E($\text{sumc}(\mu)$)</th>
<th>E($\text{prodc}(\mu)$)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>14.9299</td>
<td>5.4883</td>
</tr>
<tr>
<td></td>
<td>130.7018</td>
<td>533.3792</td>
</tr>
<tr>
<td></td>
<td>15.5184</td>
<td>3.9212</td>
</tr>
<tr>
<td></td>
<td>168.4666</td>
<td>471.0707</td>
</tr>
<tr>
<td></td>
<td>0.0549</td>
<td>5.3338</td>
</tr>
<tr>
<td></td>
<td>0.8116</td>
<td>64.8759</td>
</tr>
</tbody>
</table>
Accuracy Diagnostics
Summary statistics on weights:

<table>
<thead>
<tr>
<th>avg</th>
<th>stddev</th>
<th>min</th>
<th>max</th>
<th>maxsq/totsq</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0120</td>
<td>36.3643</td>
<td>0.0000</td>
<td>3341.8561</td>
<td>0.8440</td>
</tr>
</tbody>
</table>

RNEs and 1/RNEs:

<table>
<thead>
<tr>
<th>0.0023</th>
<th>428.4143</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0053</td>
<td>189.6681</td>
</tr>
</tbody>
</table>
Example, cont.

Plot of weights (all, top 1,000, top 100):
Example, cont.

Exercise: Replicate
Improving Efficiency

From a programming perspective, two simple approaches to improving efficiency are:

- **Increase N** (RNEs indicate good rules of thumb for necessary increases). This brute-force method is often computationally prohibitive.
- **Sequential updating.** (Can still be expensive, but less brutish.)
Improving Efficiency, cont.

Sequential updating:

- Begin with an initial parameterization \( a_0 \) for \( g(\theta|a) \) (e.g., \((\mu_0, \Sigma_0))\).
- Calculate \( \hat{\theta}_0 \), map into \( a_1 \).
- Repeat until \( a_i \) yields an acceptable level of numerical accuracy.
Improving Efficiency, cont.

Returning to the example, RNEs and $1/RNEs$ evolve as follows:

**Iteration 0:**

<table>
<thead>
<tr>
<th>RNE</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0023</td>
<td>428.4143</td>
</tr>
<tr>
<td>0.0053</td>
<td>189.6681</td>
</tr>
</tbody>
</table>

**Iteration 1:**

<table>
<thead>
<tr>
<th>RNE</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0575</td>
<td>17.4021</td>
</tr>
<tr>
<td>0.0331</td>
<td>30.1702</td>
</tr>
</tbody>
</table>

**Iteration 2:**

<table>
<thead>
<tr>
<th>RNE</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.8018</td>
<td>1.2472</td>
</tr>
<tr>
<td>0.7302</td>
<td>1.3694</td>
</tr>
</tbody>
</table>

**Iteration 3:**

<table>
<thead>
<tr>
<th>RNE</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.9790</td>
<td>1.0215</td>
</tr>
<tr>
<td>0.9535</td>
<td>1.0488</td>
</tr>
</tbody>
</table>
Improving Efficiency, cont.

Weight plots, Iteration 3:
Caveats regarding sequential updating:

- Convergence to target is not guaranteed
- Performance can be sensitive to starting values
- Initial sampler should be sufficiently diffuse to ensure coverage of appropriate range for targeted integrand
Aside regarding coverage:
The multivariate-t density is an attractive sampler relative to the normal density: it has similar location and shape parameters, but has \textbf{thicker tails}. Tail thickness controlled by degrees-of-freedom parameter $\nu$ (smaller $\nu$, thicker tails).

- Parameters of the multivariat-t: $(\gamma, V, \nu)$.
- Mean and second-order moments: $\gamma, \left(\frac{\nu}{\nu-2}\right)V^{-1}$
Improving Efficiency, cont.

Algorithm for obtaining drawings $\mu_i$ from multivariate-t ($\gamma, V, v$):

- Obtain $s_i$ from a $\chi^2(v)$ distribution:

$$s_i = \sum_{j=1}^{v} x_j^2, \quad x_j \sim N(0, 1)$$

- Use $s_i$ to construct the scaling factor

$$\sigma_i = (s_i / v)^{-1/2}$$

- Obtain $\mu_i$ as

$$\mu_i = \gamma \pm \sigma_i V^{-1/2} w_i, \quad w_{i,j} \sim N(0, 1), \quad j = 1, \ldots, k$$

$$V^{-1/2} = \text{chol}(V^{-1})'$$

- Note: use of $\pm$ yields antithetic acceleration (Geweke, 1988, JoE)
**Exercise:** Replace the normal densities used as Importance Samplers in the exercise above with multi-t densities, with $\gamma = \mu$ and $V^{-1} = \Sigma$. Experiment with alternative $\nu'$s to assess the impact on numerical efficiency.
**Goal:** Tailor \( g(θ|a) \) (via the specification of \( a \)) to minimize the n.s.e. associated with the approximation of

\[
G(Y) = \int_{Θ} ϕ(θ|Y) dθ.
\]

Write \( g(θ|a) \) as

\[
g(θ|a) = \frac{k(θ;a)}{χ(a)}
\]

\[
χ(a) = \int_{Θ} k(θ;a) dθ.
\]
Details regarding the tailoring of $g(\theta|a)$ are distinct for two special cases:

- $g(\theta|a)$ is parametric (i.e., a normal distribution)
- $g(\theta|a)$ is piecewise-linear
When \( g(\theta | a) \) is fully parametric, n.s.e. is (approximately) minimized via iterations on

\[
(\hat{a}_{l+1}, \hat{c}_{l+1}) = \arg \min_{a,c} \overline{Q}_N(a, c; Y|\hat{a}_l),
\]

\[
\overline{Q}_N(a, c; Y|\hat{a}_l) = \frac{1}{N} \sum_{i=1}^{N} d^2(\theta_i^l, a, c, Y) \omega(\theta_i^l; Y, \hat{a}_l),
\]

\[
d(\theta_i^l, a, c, Y) = \ln \varphi(\theta_i^l | Y) - c - \ln k(\theta_i^l; a),
\]

\[
\omega(\theta_i^l; Y, \hat{a}_l) = \frac{\varphi(\theta_i^l | Y)}{g(\theta_i^l | \hat{a}_l)}.
\]

The term \( c \) is a normalizing constant that controls for factors in \( \varphi \) and \( g \) that do not depend upon \( \theta \). Typically, it suffices to set \( \omega(\theta_i^l; Y, \hat{a}_l) = 1 \; \forall \; i \).
When \( g(\theta|a) \) is piecewise-linear, the parameters \( a \) are grid points:

\[
a' = (a_0, ..., a_R), \quad a_0 < a_1 < ... < a_R.
\]

In this case, the kernel \( k(\theta; a) \) is given by

\[
\ln k_j(\theta; a) = \alpha_j + \beta_j \theta \quad \forall \theta \in [a_{j-1}, a_j],
\]

\[
\beta_j = \frac{\ln \varphi(a_j) - \ln \varphi(a_{j-1})}{a_j - a_{j-1}}, \quad \alpha_j = \ln \varphi(a_j) - \beta_j a_j.
\]

Optimization is achieved by selecting \( \hat{a} \) as an equal-probability division of the support of \( \varphi(\theta|Y) \).
Given final estimates \((\hat{a}, \hat{c})\), the EIS estimate of \(G(Y)\) is given by

\[
G(Y)_N = \frac{1}{N} \sum_{i=1}^{N} \omega(\theta_i; Y, \hat{a}) .
\]

N.S.E. is computed as indicated above.
Implementation, Gaussian Sampler

To simplify notation, denote the targeted integrand as ϕ(θ|Y) ≡ ϕ(θ). We’ll take θ as k-dimensional, with elements (x₁, x₂, …xₖ).

With g(θ|a) Gaussian, a consists of the k × 1 vector of means μ and the k × k covariance matrix Σ. Since the covariance matrix is symmetric, the number of auxiliary parameters reduces to k + k(k + 1)/2.

The precision matrix H = Σ⁻¹.
Implementation, Gaussian Sampler, cont.

Our goal is to choose \((\mu, H)\) to approximate optimally \(\ln \varphi(s)\) by a Gaussian kernel:

\[
\ln \varphi(\theta) \propto -\frac{1}{2} (\theta - \mu)' H (\theta - \mu)
\]

\[
\propto -\frac{1}{2} (\theta' H \theta - 2 \theta' H \mu).
\]

Recall that by ‘optimally’, we refer to the weighted-squared-error minimization introduced above.
The term $\theta' H \theta$ can be written as

$$
\begin{pmatrix}
  x_1 & x_2 & \ldots & x_k \\
  h_{11} & h_{21} & \ldots & h_{1k} \\
  h_{21} & h_{22} & \ldots & h_{2k} \\
  \vdots & \vdots & \ddots & \vdots \\
  h_{k1} & h_{k2} & \ldots & h_{kk}
\end{pmatrix}
\begin{pmatrix}
  x_1 \\
  x_2 \\
  \vdots \\
  x_k
\end{pmatrix}
$$
Expanding $\theta' H \theta$, we obtain

$$
\theta' H \theta = h_{11} (x_1^2) + h_{22} (x_2^2) + \ldots + h_{jj} (x_{kj}^2) \\
+ 2h_{21} (x_2 x_1) + 2h_{31} (x_3 x_1) + \ldots + 2h_{k1} (x_k x_1) \\
+ 2h_{32} (x_3 x_2) + 2h_{42} (x_4 x_2) + \ldots + 2h_{k2} (x_k x_2) \\
\ldots \\
+ 2h_{k(k-1)} (x_k x_{k-1}) .
$$

This expression indicates that the coefficients of the squares, pairwise products and the individual components of $\theta$ are in one-to-one correspondence with the $\mu$ and $H$. 
Given this correspondence, the EIS optimization problem amounts to a weighted-least-squares problem involving the regression of $\ln \varphi(s)$ on

$$[1, x_1, \ldots, x_k, x_1 x_2, x_1 x_3, \ldots, x_{k-1} x_k, x_1^2, x_2^2, \ldots, x_k^2] .$$

The number of regressors is $K = \left( 1 + k + \frac{k(k+1)}{2} \right)$.
Implementation, Gaussian Sampler, cont.

Implementation Algorithm

- Specify $a_0$, generate

\[
y = \begin{pmatrix} \ln \varphi(\theta_1) \\ \vdots \\ \ln \varphi(\theta_M) \end{pmatrix}, \quad w = \begin{pmatrix} \omega_1 \\ \vdots \\ \omega_M \end{pmatrix}, \quad \frac{\varphi(\theta \mid Y)}{g(\theta \mid a_0)},
\]

\[
X = \begin{pmatrix} \kappa_1 \\ \vdots \\ \kappa_M \end{pmatrix}, \quad \kappa_i = \left[1^\sim \theta_i^\sim \text{vech}(\theta_i \cdot \theta_i')\right].
\]

(Note: $M << N$)
Implementation Algorithm, cont.

- Construct

\[ \tilde{y} = y \cdot (w \cdot 2), \quad \tilde{X} = X \cdot (w \cdot 2). \]

(Caution: set \( w = 1 \) when using a poor initial sampling density.)

- Estimate

\[ \hat{\beta} = (\tilde{X}' \tilde{X})^{-1} (\tilde{X} \tilde{y}). \]

- Map \( \hat{\beta} \) into \((\hat{\mu}, \hat{\Sigma})\). Jointly, these constitute \( a_1 \).

- Replacing \( a_0 \) above with \( a_1 \), repeat until convergence.
Implementation Algorithm, cont.

To map $\widehat{\beta}$ into $\left(\widehat{\mu}, \widehat{\Sigma}\right)$:

- Map the $k + 2$ through $K$ elements of $\widehat{\beta}$ into a symmetric matrix $\tilde{H}$, with $j^{th}$ diagonal element corresponding to the coefficient associated with the squared value of the $j^{th}$ element of $\theta$, and $(j, k)^{th}$ element corresponding to the product of the $j$ and $k^{th}$ element of $\theta$. I.E.,

$$\tilde{H} = \text{xpnd}(\widehat{\beta}[k+2:\text{rows(beta)])};$$

- Construct $\tilde{H}$ by multiplying all elements of $\tilde{H}$ by $-1$, then multiplying the diagonal elements by 2.

- $\widehat{\Sigma} = \tilde{H}^{-1}$
- $\widehat{\mu} = \widehat{\Sigma} \cdot \widehat{\beta}[2 : k + 1]$
Exercise: Return to the example outlined above. Using an initial sampler specified with

\[
\begin{align*}
\mu_0 &= \mu + 3 \cdot \sqrt{\text{diag}(\Sigma)}, \\
\Sigma_0 &= 10 \cdot \text{diagrv}(\Sigma, \sqrt{\text{diag}(\Sigma)}),
\end{align*}
\]

show that the EIS algorithm yields

\[
\hat{\mu} = \mu, \quad \hat{\Sigma} = \Sigma
\]

in one step using \( M = 50, \ w = 1 \). Experiment with alternative initial samplers.
Implementation, Piecewise-Linear Approximation

**Context:** Effective for use in univariate cases featuring obvious deviations from normality.
(Curse of dimensionality renders implementation problematic in high-dimensional cases.)
Implementation, PW-L Approx., cont.

Motivation 1. Mixture of Normals, \((\mu, \sigma) = (-0.5, 0.25), (0.5, 0.5)\), equal weights
Approximation via Gaussian samplers (handmade and EIS):

RNEs for calculating \( E(x) \): 0.61, 0.78.
Implementation, PW-L Approx., cont.

Motivation 2. \((\mu, \sigma) = (-1.5, 0.25), (1.5, 0.5)\), equal weights

RNE: 0.37
Implementation, PW-L Approx., cont.

Motivation 3. \((\mu, \sigma) = (-1.5, 0.05), (1.5, 0.5),\) equal weights

RNE: 0.11
Reboot: Recall that when \( g(\theta|a) \) is piecewise-linear, the parameters \( a \) are grid points:

\[
a' = (a_0, \ldots, a_R), \quad a_0 < a_1 < \ldots < a_R.
\]

In this case, the kernel \( k(\theta; a) \) is given by

\[
\ln k_j(\theta; a) = \alpha_j + \beta_j \theta \quad \forall \theta \in [a_{j-1}, a_j],
\]

\[
\beta_j = \frac{\ln \varphi(a_j) - \ln \varphi(a_{j-1})}{a_j - a_{j-1}}, \quad \alpha_j = \ln \varphi(a_j) - \beta_j a_j.
\]

Optimization is achieved by selecting \( \hat{a} \) as an equal-probability division of the support of \( \varphi(\theta|Y) \).
In order to achieve equal-probability division, and to implement the distribution as a sampler, we must

- construct its associated CDF
- invert the CDF.
Implementation, PW-L Approx., cont.

To gain intuition behind implementation and inversion, consider the CDF associated with Case 3:

Inversion involves inducing a mapping from the $y$ to the $x$ axis. Implementation involves mapping drawings obtained from a $U [0, 1]$ distribution onto the $x$ axis.
Implementation, PW-L Approx., cont.

Letting $s$ denote the $x$–axis variable, the CDF of $k$ can be written as

$$K_j(s; a) = \frac{\chi_j(s; a)}{\chi_n(a)}, \quad \forall s \in [a_{j-1}, a_j],$$

$$\chi_j(s; a) = \chi_{j-1}(a) + \frac{1}{\beta_j} [k_j(s; a) - k_j(a_{j-1}; a)],$$

$$\chi_0(a) = 0, \quad \chi_j(a) = \chi_j(a_j; a).$$

Note: $\chi_n(a)$ is the integrating constant associated with the pdf.
Implementation, PW-L Approx., cont.

Inversion/implementation:

\[
\begin{align*}
 s &= \frac{1}{\beta_j} \left\{ \ln \left[ k_j(a_{j-1}; a) + \beta_j \left( u \cdot \chi_R(a) - \chi_{j-1}(a) \right) \right] - \alpha_j \right\}, \\
 u &\in [0,1] \quad \text{and} \quad \chi_{j-1}(a) < u \cdot \chi_R(a) < \chi_j(a).
\end{align*}
\]
Implementation, PW-L Approx., cont.

Unrefined (equal spacing over x range) piecewise approximation for case 1:

RNE: 0.99
Implementation, PW-L Approx., cont.

Unrefined (equal spacing over x range) piecewise approximation for case 2:

RNE: 0.99
Implementation, PW-L Approx., cont.

Unrefined (equal spacing over x range) piecewise approximation for case 3:

RNE: 0.24
To understand the source of the problem in Case 3, consider again the CDF of the unrefined sampler:

Note: relatively few points along steep portions of the CDF,
To address the problem: **equal probability division** of the range for $s$. I.e., divide the vertical axis of the CDF into equal portions, then map into $s$:

$$u_i = \varepsilon + (2 - \varepsilon) \frac{i}{R}, \quad i = 1, \ldots, R - 1,$$

with $\varepsilon$ sufficiently small (typically $\varepsilon = 10^{-4}$) to avoid tail intervals of excessive length.
Iterative construction:
Given the step-$l$ grid $\hat{a}^l$, construct the density kernel $k$ and its CDF $K$ as described above. The step-$l + 1$ grid is then computed as

$$\hat{a}_{i+1} = K^{-1}(u_i), \quad i = 1, \ldots, R - 1.$$  

Iterate until (approximate) convergence.
Implementation, PW-L Approx., cont.

Refined approximation for Case 3:

RNE: 0.95
Implementation, PW-L Approx., cont.

Refined CDF: