Uncertainty Quantification

Specifying Uncertainty in Dakota

Bob Cochran
Applied Computational Heat Transfer
Seattle, WA
rjc@heattransfer.org

May 26, 2017
Revised Version: June 2, 2017

ME 498/599 Uncertainty
University of Washington
Spring Quarter 2017

Outline

- Specifying uncertain variables in Dakota
- Probability distribution functions (pdf and cdf)
- Latin hypercube sampling (LHS) algorithm

Specifying Uncertain Variables in Dakota

Dakota Input File

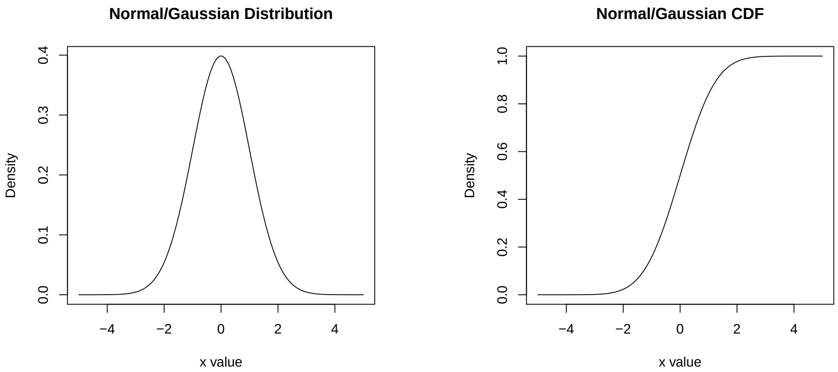
Using the distribution function which best fits your understanding of the uncertainty in the model parameters defined in the `variables` block.

- Normal/Gaussian: `normal_uncertain`
- Uniform: `uniform_uncertain`
- Lognormal: `lognormal_uncertain`
- Beta: `beta_uncertain`
- Triangle: `triangular_uncertain`

See slides 10 to 15 in Uncertainty Quantification Dakota training materials.
See [SW04] for LHS theory and distributions used in Dakota.

Normal or Gaussian Distribution

Probability Distribution Functions



$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\left(\frac{(x-\mu)^2}{2\sigma^2}\right)} \qquad -\infty < x < \infty$$

where μ is the mean and σ is the standard deviation

Specifying a Normal Distribution

Dakota Input File

```
variables
  normal_uncertain = integer
    descriptors     = 'string'
    means           = real
    std_deviations  = real
```

Specifying a Truncated Normal Distribution

Dakota Input File

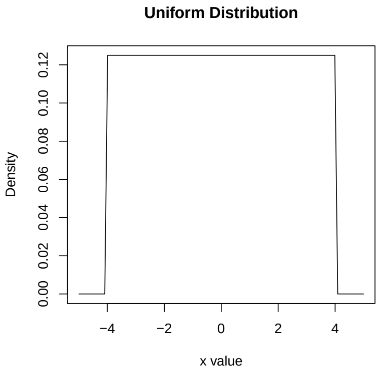
Usually you will only have a *feeling* for the amount of uncertainty in a variable, in lieu of actual data. In this case, if you want a 99% confidence (3σ) that the values will fall within a $\pm$% amount, use $\sigma = \frac{\mu \times (\%/100)}{3}$, for example:

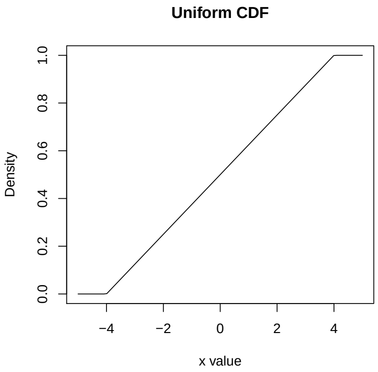
$$\mu = 12 \pm 15\% \begin{cases} \mu_{\text{upper}} = 13.8 \\ \mu_{\text{lower}} = 10.2 \end{cases} \Rightarrow \quad \sigma = \frac{12 \times 0.15}{3} = 0.6$$

```
variables
  normal_uncertain = integer
    descriptors      = 'string'
    means            = real
    std_deviations   = real
    upper_bounds     = real
    lower_bounds     = real
```

Uniform Distribution

Probability Distribution Functions





$$f(x) = \frac{1}{B - A} \qquad A \leq x \leq B$$

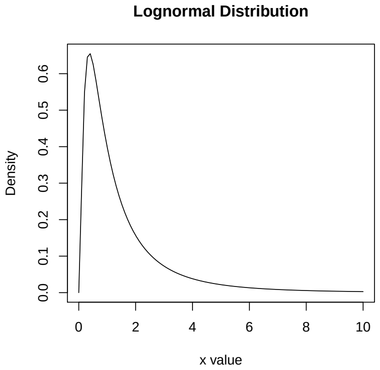
Specifying a Uniform Distribution

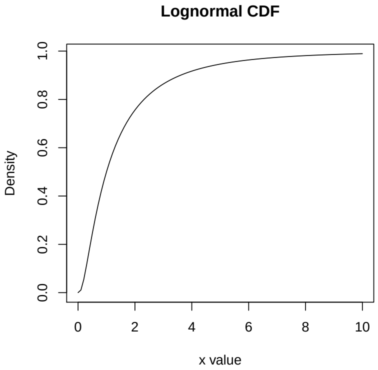
Dakota Input File

```
variables
  uniform_uncertain = integer
    descriptors     = 'string'
    lower_bounds    = real          # A
    upper_bounds    = real          # B
```

Lognormal Distribution

Probability Distribution Functions





$$f(x) = \frac{1}{x\sigma\sqrt{2\pi}} e^{-\left(\frac{(\ln x - \mu)^2}{2\sigma^2}\right)} \qquad 0 < x$$

where μ is the mean and σ is the standard deviation

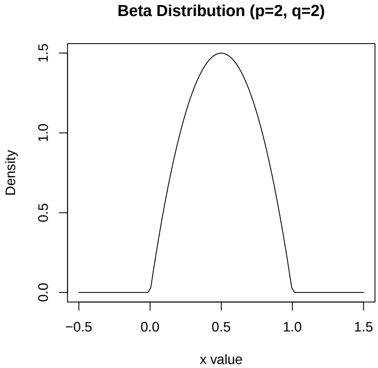
Specifying a Lognormal Distribution

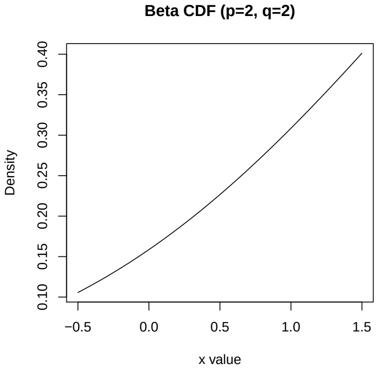
Dakota Input File

```
variables
  lognormal_uncertain = integer
    descriptors         = 'string'
    lambdas             = real
    means               = real
```

Beta Distribution

Probability Distribution Functions





$$f(\beta) = \frac{\beta}{\int_A^B \beta} \qquad \beta = x^{(p-1)}(1-x)^{(q-1)}$$

where p and q are shape parameters and A and B are the endpoints

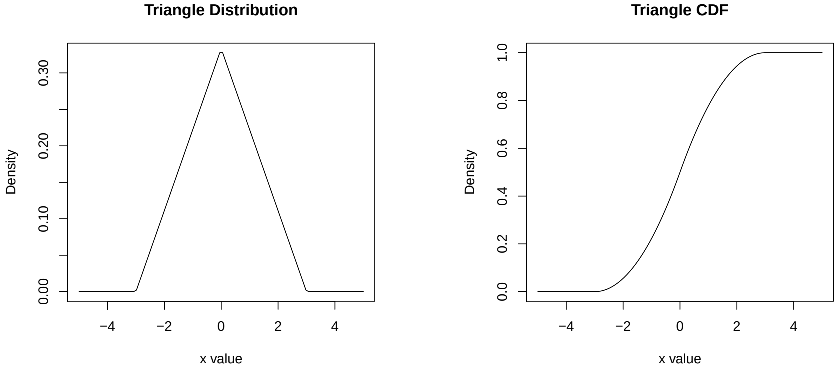
Specifying a Beta Distribution

Dakota Input File

```
variables
  beta_uncertain = integer
    descriptors          = 'string'
    alphas               = real      # p
    betas                = real      # q
    lower_bounds         = real      # A
    upper_bounds         = real      # B
```

Triangle Distribution

Probability Distribution Functions



$$f(x) = \frac{2(x-a)}{(c-a)(b-a)} \qquad a \leq x \leq b$$

$$f(x) = \frac{2(c-x)}{(c-a)(c-b)} \qquad b \leq x \leq c$$

Specifying a Triangle Distribution

Dakota Input File

```
variables
  triangular_uncertain = integer
    descriptors        = 'string'
    modes              = real      # b
    lower_bounds       = real      # a
    upper_bounds       = real      # c
```

Note that `modes` is the apex value of the triangular distribution and must fall between `lower_bounds` and `upper_bounds`.

Latin Hypercube Sampling (LHS)

Appendix

R Code for Normal Distribution Plots

```
x <- seq(-5, 5, length=100)
pdfx <- dnorm(x, mean = 0, sd = 1)

pdf("normal_dist.pdf", width=5, height=5)
plot(x, pdfx, type="l", lty=1,
     xlab="x value", ylab="Density",
     main="Normal/Gaussian Distribution")
dev.off()

cdfx <- pnorm(x, mean = 0, sd = 1)
pdf("normal_cdf.pdf", width=5, height=5)
plot(x, cdfx, type="l", lty=1,
       xlab="x value", ylab="Density",
       main="Normal/Gaussian CDF")
dev.off()
```

R Code for Uniform Distribution Plots

```
x <- seq(-5, 5, length=100)
pdfx <- dunif(x, min = -4, max = 4)

pdf("unif_dist.pdf", width=5, height=5)
plot(x, pdfx, type="l", lty=1,
        xlab="x value", ylab="Density",
        main="Uniform Distribution")
dev.off()

cdfx <- punif(x, min = -4, max = 4)
pdf("unif_cdf.pdf", width=5, height=5)
plot(x, cdfx, type="l", lty=1,
        xlab="x value", ylab="Density",
        main="Uniform CDF")
dev.off()
```

R Code for Lognormal Distribution Plots

```
x <- seq(0, 10, length=100)
pdfx <- dlnorm(x, meanlog = 0, sdlog = 1)

pdf("lnorm_dist.pdf", width=5, height=5)
plot(x, pdfx, type="l", lty=1,
     xlab="x value", ylab="Density",
     main="Lognormal Distribution")
dev.off()

cdfx <- plnorm(x, meanlog = 0, sdlog = 1)
pdf("lnorm_cdf.pdf", width=5, height=5)
plot(x, cdfx, type="l", lty=1,
       xlab="x value", ylab="Density",
       main="Lognormal CDF")
dev.off()
```

R Code for Beta Distribution Plots

```
x <- seq(-0.5, 1.5, length=100)
pdfx <- dbeta(x, 2, 2)

pdf("beta_dist.pdf", width=5, height=5)
plot(x, pdfx, type="l", lty=1,
     xlab="x value", ylab="Density",
     main="Beta Distribution (p=2, q=2)")
dev.off()

cdfx <- pnorm(x, 2, 2)
pdf("beta_cdf.pdf", width=5, height=5)
plot(x, cdfx, type="l", lty=1,
        xlab="x value", ylab="Density",
        main="Beta CDF (p=2, q=2)")
dev.off()
```

R Code for Triangle Distribution Plots

```
library(triangle)

x <- seq(-5, 5, length=100)
hx <- dtriangle(x, a=-3, b=3)

pdf("tri_dist.pdf", width=5, height=5)
plot(x, hx, type="l", lty=1,
        xlab="x value", ylab="Density",
        main="Triangle Distribution")
dev.off()

cdfx <- ptriangle(x, a=-3, b=3)
pdf("tri_cdf.pdf", width=5, height=5)
plot(x, cdfx, type="l", lty=1,
        xlab="x value", ylab="Density",
        main="Triangle CDF")
dev.off()
```

References I

[SW04] Laura P. Swiler and Gregory D. Wyss.
A user's guide to Sandia's latin hypercube sampling software: LHS UNIX library/standalone version.
Technical Report SAND04-2439, Sandia National Laboratories, Albuquerque, NM, 2004.