Explanation of algorithm for sampling from $s_k(x) = u_k(x)/\int_D u_k(x')dx' = c\ u_k(x)$. First we need to find the normalizing constant $c := 1/\int_D u_k(x')dx'$:

$$\int_D u_k(x')dx' = \sum_{j=0}^{k+1} I_j,$$

where

$$I_j := \int_{z_{j-1}}^{z_j} \exp(h(x_j) + (x' - x_j)h'(x_j))dx'.$$

Notice that the explicit form of I_j depends on whether $h'(x_j) = 0$ or not. Therefore we have,

$$I_j = \begin{cases} (z_j - z_{j-1})\exp(h(x_j)) & \text{if} \quad h'(x_j) = 0, \\ \frac{\exp u_k(z_j) - \exp u_k(z_{j-1})}{h'(x_j)} & \text{otherwise.} \end{cases}$$

Next, in order to use the inverse CDF method for sampling, we must find the CDF for $s_k(x)$, $S_k(x) = c\int_{z_0}^{x} u_k(x')dx'$:

$$S_k(x) = c\left(\sum_{j=0}^{t-1} I_j + \int_{z_{t-1}}^{x} \exp(h(x_t) + (x' - x_t)h'(x_t))dx')\right),$$

where t is the index of which interval of z's that x lies in. Formally, it is $t(x) = \{1 \le i \le k+1 : x \in (z_{i-1}, z_i)\}$. For convenience, let

$$\texttt{partialSums[t-1]} := \sum_{j=1}^{t-1} I_j$$

and notice that our normalizing constant can be expressed as $c = \texttt{1/partialSums[k]}$. Moreover, let us define

$$J_{t-1}(x) := \int_{z_{t-1}}^{x} \exp(h(x_t) + (x' - x_t)h'(x_t))dx').$$

Then we have,

$$S_k(x) = c\left(\texttt{partialSums[t-1]} + J_{t-1}(x)\right),$$

where

$$J_{t-1}(x) = \begin{cases} \exp\big(h(x_t)\big)(x - z_{t-1}) & \text{if} \quad h'(x_t) = 0, \\ \frac{\exp u_k(x) - \exp u_k(z_{t-1})}{h'(x_t)} & \text{otherwise.} \end{cases}$$

Now, we can determine the inverse transform $S_k^{-1}(U)$, where $U \sim$ Uniform[0,1]. Because t is actually a function of x, it too must be inverted. Intuitively, we want to pick the biggest t such that $S_k(z_{t-1}) < U$. Formally,

$t(U) = \{1 \leq i \leq k+1 : U \in (c \times \texttt{partialSums[i - 1]}, c \times \texttt{partialSums[i]})\}$. Solving for the inverse, we have:

$$S_k^{-1}(U) = \begin{cases} \frac{\frac{U}{c} - \texttt{partialSums[t-1]}}{\exp\big(h(x_t)\big)} + z_{t-1} & \text{if} \quad h'(x_t) = 0, \\ \frac{\log\big(h'(x_t)(\frac{U}{c} - \texttt{partialSums[t-1]}) + \exp u_k(z_{t-1})\big) - h(x_t)}{h'(x_t)} + x_t & \text{otherwise.} \end{cases}$$

Contributions of Birce: Birce wrote the codes for finding initial set of abscissae points if not provided by the user, finding and updating the intersection points of the envelope function, updating abscissae and vectors `hx` and `dhx` and finally the accept/reject function that takes returns the decision regarding a new sampled point.