



NIKHEF-H/93-24

October 1993

Generation of pseudo-random variates with fixed sum and product

Jiri Hoogland and Ronald Kleiss

NIKHEF-H, P.O. Box 41882, 1009 DB Amsterdam, The Netherlands

Abstract

We discuss algorithms for generating n positive (pseudo)random numbers that sum to 1 and have a fixed product s . Two explicit examples are given, and the probability density they generate on the $(n - 2)$ -dimensional space defined by the sum and product constraints are derived. The efficiency of these algorithms as a function of s and n is discussed.

Generation of pseudo-random variates with fixed sum and product

Jiri Hoogland and Ronald Kleiss

NIKHEF-H, P.O. Box 41882, 1009 DB Amsterdam, The Netherlands

Abstract

We discuss algorithms for generating n positive (pseudo)random numbers that sum to 1 and have a fixed product s . Two explicit examples are given, and the probability density they generate on the $(n-2)$ -dimensional space defined by the sum and product constraints are derived. The efficiency of these algorithms as a function of s and n is discussed.

1 Introduction

In this paper we discuss the generation of variates with fixed sum and product. This problem arose in an investigation of the possibility of calculating complicated Feynman loop integrals numerically, by Monte Carlo sampling of the Feynman parameters. For instance, in the fusion process of two gluons into a Higgs boson, mediated by a quark loop, the effective gluon-gluon-Higgs coupling contains the following integral [1]:

$$\int dx_1 dx_2 dx_3 \frac{1 - x_1 x_2}{1 - \alpha x_1 x_2 - \epsilon} \delta(x_1 + x_2 + x_3 - 1),$$

where α is a positive number, and ϵ is taken to be infinitesimally small. For such wildly fluctuating integrands, it is of course important that one be able to control the product $x_1 x_2$. Although this integral is a simple case, it is easily seen that more complicated integrands, containing sums of products of x_i 's, appear in higher-order corrections, or loops with more external legs. We believe that the task of generating x_i values under such a nontrivial constraint must be well under control if a Monte Carlo integration of loop integrals is to be feasible, and the relatively simple constraint on the total product provides a good starting point for the development of the necessary techniques.

For the moment, then, the problem is the following: given n and s , and a source of iid (pseudo)random numbers ρ that are uniform in the unit interval (this is the usual format of standard 'random number generators'), we have to construct variates x_1, x_2, \dots, x_n that satisfy

$$x_i \geq 0, \quad \sum_{i=1}^n x_i = 1, \quad \prod_{i=1}^n x_i = s. \quad (1)$$

This is only possible if $0 \leq s \leq n^{-n}$, so that we may put

$$s = \sigma^{-n}, \quad \sigma \geq n. \quad (2)$$

Note that trivial scaling can accommodate any other fixed value for the sum. The optimal algorithm is of course the one that yields a distribution of points $\vec{x} = (x_1, x_2, \dots, x_n)$ that is uniform on the $n-2$ -dimensional subspace of $[0, 1]^n$ defined by the constraints (1). For instance, such an algorithm is

known if only either the sum or the product of the x_i is fixed [2]. Other such examples are that of generating points throughout the $(3n - 4)$ -dimensional relativistic phase space of n massless particles, where the RAMBO algorithm [3] is optimal, and that of generating points throughout the non-relativistic phase space of n massive particles, where the non-relativistic alternative of RAMBO is optimal. In all such cases, however, knowledge of the optimal generation algorithm is concomitant with the ability to evaluate the corresponding constrained integral exactly in (essentially) closed form. For the present problem, however, this would imply that we are able to compute the integral

$$\Phi_n(s) = \int_0^1 \prod_{i=1}^n dx_i \delta \left(\sum_j x_j - 1 \right) \delta \left(\prod_k x_k - s \right) \quad (3)$$

in some closed form, using known functions. Now, from Eq.(3) we can derive the following recursion relation:

$$\Phi_{n+1}(s) = \int_0^1 dx \frac{1}{x(1-x)} \Phi \left(\frac{s}{x^n(1-x)} \right) \quad (4)$$

We can simply find

$$\Phi_1(s) = \delta(s-1) \quad , \quad \Phi_2(s) = \frac{2\theta(1-4s)}{\sqrt{1-4s}} \quad , \quad (5)$$

but already $\Phi_3(s)$ involves an elliptic integral. It is fairly easy to prove, however, that

$$\begin{aligned} \Phi_3(s) &\sim 3 \log \frac{1}{s} \quad (s \rightarrow 0) \quad , \\ \Phi_n(s) &\propto \left(n^{-n} - s \right)^{(n-3)/2} \quad (s \rightarrow n^{-n}) \quad , \end{aligned} \quad (6)$$

and we shall use this information to inspect our results later on. For the moment, however, we conjecture that there is no optimal algorithm (i.e. with constant density) that consists of relatively simple manipulations such as multiplications, logarithms, etcetera. We therefore settle for the next best thing and only ask for an algorithm leading to a density that does not fluctuate too much over the $(n-2)$ -dimensional subspace.

2 The general algorithm

We shall now discuss a general class of algorithms that generate points \vec{x} with the constraints of Eq.(1). Since n variables are required, it is reasonable to start with n pseudo-random numbers ρ_i ($i = 1, \dots, n$), and then transform these so as to satisfy Eq.(1): this transformation, then, must depend on 2 parameters representing the two degrees of freedom that are to be eliminated. We therefore put

$$x_i = \frac{1}{t} f(\rho_i, \nu) \quad i = 1, 2, \dots, n \quad , \quad (7)$$

which is essentially the most general form. The two parameters t and ν are then given by

$$\begin{aligned} t &= \sum_{i=1}^n f(\rho_i, \nu) \quad , \\ Q(\nu) &\equiv \prod_i f(\rho_i, \nu) \left(\sum_{i=1}^n f(\rho_i, \nu) \right)^{-n} = s \quad . \end{aligned} \quad (8)$$

The only nontrivial point in this algorithm is the determination of ν . We shall of course only be interested in mappings f that in fact allow Eq.(8) to be solved, and preferably in a simple manner.

We must now calculate the (non-uniform) density with which the points \vec{x} are generated. This density is reflected into a weight which will come with each generated event \vec{x} . This calculation is done by writing the algorithm in unitary form:

$$\begin{aligned} 1 &= \int_0^1 \prod_{i=1}^n d\rho_i \\ &= \int \prod_i [d\rho_i dx_i \delta(x_i - f(\rho_i, \nu)/t)] \\ &\quad \times dt \delta \left(t - \sum_j f(\rho_j, \nu) \right) d\nu \delta(Q(\nu) - s) |Q(\nu)| \quad , \end{aligned} \quad (9)$$

$$\delta \left(t - \sum_j f(\rho_j, \nu) \right) = \frac{1}{t} \delta \left(\sum_i x_i - 1 \right) \quad . \quad (10)$$

where the implied boundaries on the integrals are determined later on. The Dirac functions may be manipulated as follows. First, we have, trivially:

Next,

$$\begin{aligned} \delta(x_i - f(\rho_i, \nu)/t) &= |t| \delta(x_i t - f(\rho_i, \nu)) \\ &= \left| \frac{\partial f(\rho_i, \nu)}{\partial \rho_i} \right| \delta(\rho_i - \phi(\nu, x_i t)) \\ &= \left| \frac{1}{x_i \frac{\partial f(\rho_i, \nu)}{\partial \rho_i}} \right| \delta(\rho_i - \phi(\nu, x_i t)) , \end{aligned} \quad (11)$$

where ϕ is the inverse function of f , and

$$g(\rho_i, \nu) = \log f(\rho_i, \nu) . \quad (12)$$

Finally,

$$\begin{aligned} |Q'(\nu)| &= \left| Q(\nu) \left\{ \sum_i \frac{\partial f(\rho_i, \nu)}{f(\rho_i, \nu)} - \frac{n}{\sum_j f(\rho_j, \nu)} \sum_k \frac{\partial f(\rho_k, \nu)}{\partial \nu} \right\} \right| \\ &= \left| \sum_i (1 - nx_i) \frac{\partial}{\partial \nu} g(\rho_i, \nu) \right| . \end{aligned} \quad (13)$$

We can now eliminate the ρ_i from Eq.(9):

$$\begin{aligned} 1 &= \int_0^1 \prod dx_i \delta \left(\sum_j x_j - 1 \right) \delta \left(\prod_k x_k - s \right) \\ &\quad \times \int d\nu dt \left| \prod_i \frac{\partial}{\partial \rho_i} g(\rho_i, \nu) \right|^{-1} \left| \sum_j (1 - nx_j) \frac{\partial}{\partial \nu} g(\rho_j, \nu) \right| \end{aligned} \quad (14)$$

The expression in the last line is the probability density of \vec{x} on the $(n-2)$ -dimensional subspace. The integration limits on ν and t are given by the condition $0 < \rho_i = \phi(\nu, x_i t) < 1$, and hence depend on the particular mapping that was chosen. Before examining special choices for f , we want to remark that the above algorithms are invariant under reparametrizations of ν and t ; that is, if we denote by $v_1(\nu)$ and $v_2(t)$ two strictly monotonic functions of ν and t , then the choice

$$x_i = \frac{1}{v_2(t)} f(\rho_i, v_1(\nu)) \quad (15)$$

results in *exactly* the same density, that of Eq.(14), as does the choice of Eq.(7). It therefore makes no difference if we replace, say, ν by $\log \nu$ or so.

3 Two particular algorithms

3.1 Algorithm A

We shall now discuss two different choices for f . In the first place, we may want to simplify the dependence of the density on \vec{x} as much as possible, keeping only the sum with the factor $1 - nx_i$, which is presumably unavoidable. It is, then, attractive to choose g such that $\frac{\partial}{\partial \nu} g(\rho_i, \nu)$ is independent of ρ_i . Keeping in mind the reparametrization invariance, we can therefore write

$$g(\rho_i, \nu) = -\nu \rho_i \Rightarrow f(\rho_i, \nu) = \exp(-\nu \rho_i) . \quad (16)$$

This leads to

$$\rho_i = -\frac{\log(x_i t)}{\nu} , \quad (17)$$

and the integration limits on ν and t are seen to be

$$\frac{\exp(-\nu)}{x_{\min}} < t < \frac{1}{x_{\max}} , \quad \nu > \log \frac{x_{\max}}{x_{\min}} . \quad (18)$$

Here $x_{\min} = \min_i x_i$ and $x_{\max} = \max_i x_i$. The density is then

$$\int_{\log r}^{\infty} d\nu \int_{e^{-\nu/x_{\min}}}^{1/x_{\max}} t^{-1} \nu^{-n-1} \sum_i (nx_i - 1) \log x_i = \frac{\sum_i (nx_i - 1) \log x_i}{n(n-1)(\log r)^{n-1}} . \quad (19)$$

The last point of importance for this algorithm is the determination of ν . The equation (8) for ν is equivalent, in this case, to

$$\begin{aligned} F(\nu) &\equiv \sum_i e^{\nu \alpha_i} = \sigma , \\ \alpha_i &\equiv -\beta_i + \bar{\beta} , \quad \bar{\beta} = \sum_j r_j / n . \end{aligned} \quad (20)$$

Putting $\alpha_{\max} = \max_i \alpha_i > 0$, we can easily see that the solution for ν is bracketed by

$$0 < \frac{\log \sigma - \log n}{\alpha_{\max}} < \nu < \frac{\log \sigma}{\alpha_{\max}} . \quad (21)$$

Moreover,

$$\begin{aligned} F'(\nu) &= \sum_i \alpha_i e^{\nu \alpha_i} > 0, \\ F''(\nu) &= \sum_i \alpha_i^2 e^{\nu \alpha_i} > 0, \end{aligned} \quad (22)$$

so that $F(\nu)$ is nicely concave. Simple Newton-Raphson iteration is therefore guaranteed to converge quickly to the root, provided we start at the 'high end' with $\nu = \log \sigma / \alpha_{\max}$.

In the above example, g was chosen such as to simplify the dependence on \bar{x} : however, due to the integration limits, a power of $\log(x_{\max}/x_{\min})$ creeps in again.

3.2 Algorithm B

The 'next simplest' algorithm is probably the following choice for f :

$$f(\rho_i, \nu) = \rho_i^\nu \Rightarrow g(\rho_i, \nu) = \nu \log \rho_i. \quad (23)$$

We find that $\rho_i = (x_i t)^{1/\nu}$, and the integration limits on t and ν become quite simple:

$$0 < t < \frac{1}{x_{\max}}, \quad \nu > 0. \quad (24)$$

The density, in this case, is

$$\int_0^\infty d\nu \int_0^{1/x_{\max}} d\nu^{-n-1} (st^\nu)^{1/\nu} \sum_i (nx_i - 1) \log x_i = \frac{(n-2)! \sum_i (nx_i - 1) \log x_i}{n^n (\log \sigma x_{\max})^{n-1}}. \quad (25)$$

We see that the second algorithm yields a density that is quite similar to that from the first algorithm: in particular, for large n , we have, roughly, $(n-2)!n^{-n} \sim n^{-2} \exp(-n)$ so that the difference between algorithms A and B is essentially the replacement of x_{\min} by $1/\sigma$ times a factor somewhat smaller than 1: this we may easily believe to be a reasonable numerical approximation.

Moreover, we note that the form of Eq.(8) in this case is in fact the same as that of algorithm A, with the same $F(\nu)$ provided we replace $\alpha_i = \bar{p} - \rho_i$

by

$$\alpha_i = \log \rho_i - \sum_k \log \rho_k / n; \quad (26)$$

the same remarks on the Newton-Raphson method also hold in this case.

Before proceeding we also point out the important observation that the event weights are bounded, i.e. the weight distribution may have a tail but it is always finite. A proof of this statement, together with an upper bound on the weight, is given in the Appendix.

This finishes our discussion of the algorithms. Apart from trivial reparametrizations of ν and t , we have not been able to find mappings f that lead to density functions of simplicity comparable to algorithms A and B.

4 Numerical results

We now turn to the performance of the algorithms, where we shall be interested in correctness and the distribution of the generated weights, rather than speed. In the first place, a nice and powerful check on the correctness of our computation of the density of points on the $(n-2)$ -dimensional subspace is the following. If, for given n , we pick s random between 0 and n^{-n} before generating \bar{x} , we are effectively computing integrals with only the sum constraint, which are relatively easy. In particular, we have

$$\begin{aligned} \int_0^1 \prod_i (dx_i x_i^{c_i}) \delta \left(\sum_j x_j - 1 \right) &= \\ &= \beta_n(c_1, \dots, c_n) \equiv \frac{\Gamma(c_1 + 1) \dots \Gamma(c_n + 1)}{\Gamma(c_1 + \dots + c_n + n)}. \end{aligned} \quad (27)$$

This suggests the following test:

1. generate an s value between 0 and n^{-n} ;
2. generate a vector \bar{x} for this s , using either algorithm A or B; this point \bar{x} comes with a weight $w_{A < B}(\bar{x})$, the reciprocal of the density;
3. pick a set of numbers c_1, \dots, c_n at random (they should all be larger than -1);

4. compute

$$W_{A,B} = \frac{w_{A,B}(\vec{x}) \prod_{i=1}^n z_i^{c_i}}{n^n \beta(c_1, \dots, c_n)}; \quad (28)$$

then, the expectation value (average) of the numbers W must be unity. In this manner we test a whole, large parameter space (in terms of s and the c_i) at once. An example of such a test for n up to 10 and integer c_i up to 4, is given in the table, which is based on samples of 100,000 events each. Here, the error estimate is the standard Monte Carlo one.

n	$\langle W_A \rangle$	$\langle W_B \rangle$
2	0.9988 ± 0.0067	0.9931 ± 0.0065
3	0.9978 ± 0.0041	0.9960 ± 0.0045
4	0.9827 ± 0.0074	1.0072 ± 0.0078
5	1.0085 ± 0.0145	1.0196 ± 0.0131
6	0.9745 ± 0.0205	1.0101 ± 0.0215
7	0.9430 ± 0.0336	0.9422 ± 0.0285
8	1.2751 ± 0.2003	1.0084 ± 0.0677
9	0.9877 ± 0.0939	0.9263 ± 0.0681
10	0.7572 ± 0.0611	0.9737 ± 0.2206

From the table, we see that the expected value of 1 for $\langle W \rangle$ is reasonably well reproduced, except possible for algorithm A and $n = 10$. Note that the estimated errors are a good deal larger than the 'natural' value for 10^6 points, which is about 3×10^{-3} . This is a reflection of the enormous fluctuations in the weight: this is obviously not a very efficient method of computing the number 1.

Another point of interest is that, for given n and s , the expected value of $w(\vec{x})$ must be equal to $\Phi_n(s)$: this gives us the means of computing $\Phi_n(s)$, for given s , by Monte Carlo. Note that this is distinct from generating s values distributed according to $\Phi_n(s)$, for which the algorithm is simply to generate n numbers z_i that sum to one, and taking s to be their product. Here is one more example of how a horribly transcendental function can actually be generated very simply! In fig.1 we give the simulated shape of $\Phi_n(s)$ between $s = 0$ and $s = n^{-n}$, for n up to 10. The behaviour of $\Phi_n(s)$ with n is seen to be in accord with the results mentioned in the introduction. Finally, we give

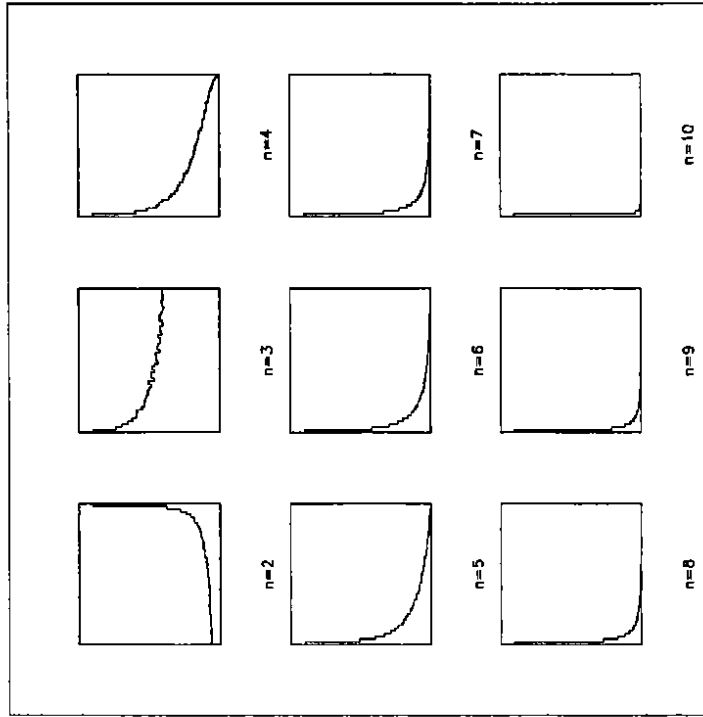


Figure 1: The probability densities $\Phi_n(s)$ for various values of n , and s between 0 and n^{-n} . All histograms are normalized to unity.

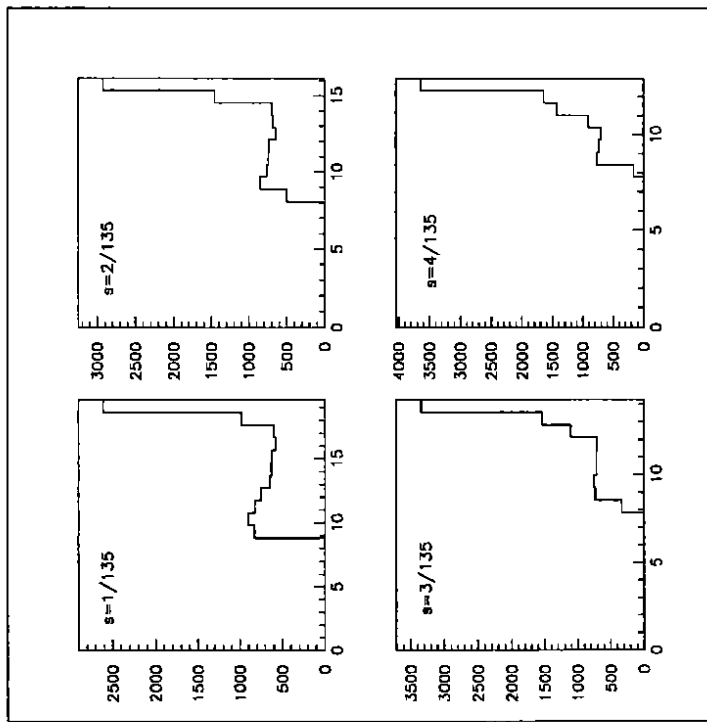


Figure 2: Weight distributions, following from algorithm A, for samples of 10,000 events, for $n = 3$. The four histograms correspond to s values of, respectively, 0.2, 0.4, 0.6, and 0.8 of the maximum s value for this n .

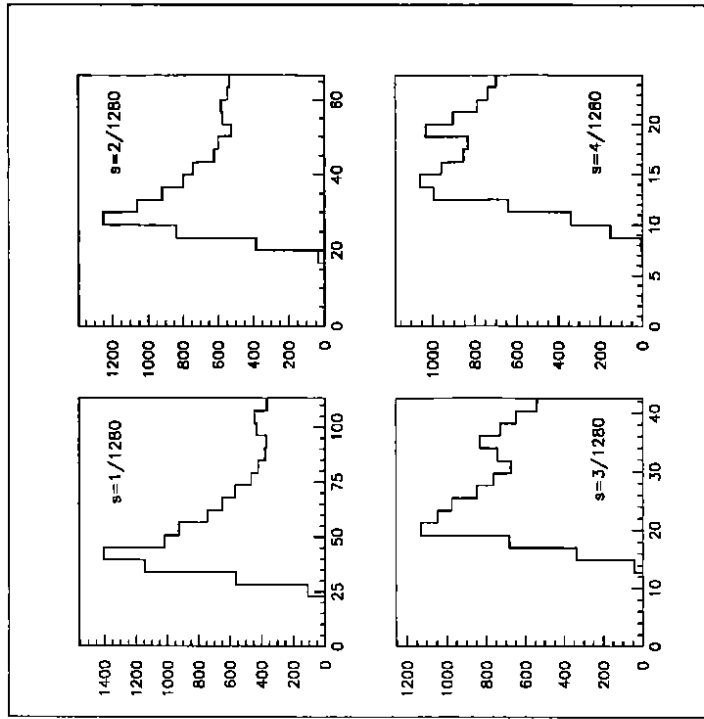


Figure 3: Weight distributions for $n = 4$.

in figs. 2, 3 and 4 the distribution of the weights $w(\bar{x})$ for some chosen values of s and n , for algorithm A. Distributions such as these are quite acceptable. Note that for increasing n the weight distribution deteriorates a bit in the sense that a tail towards high values develops. Nonetheless we conclude that the algorithms described above provide an efficient way of generating variates with fixed sum and product for values of n that are not too high.

Appendix: An upper bound on the weight

We shall provide an upper bound on the weight as given by algorithm B. Of course, this is equivalent to a lower bound on the density. This is done by minimizing the numerator in Eq.(25), and maximizing the denominator. The numerator contains the factor

$$R \equiv \sum_i (nx_i - 1) \log x_i = -\log s + n \sum_i x_i \log x_i, \quad (29)$$

so that we are actually looking for the minimum of $\sum_i x_i \log x_i$. The method of Lagrange multipliers then gives

$$\frac{\partial}{\partial x_k} \left(\sum_i x_i \log x_i - \lambda_1 \sum_i x_i - \lambda_2 \prod_i x_i \right) = \log x_k + 1 - \lambda_1 - \frac{\lambda_2 s}{x_k} = 0. \quad (30)$$

Now, this equation may have either one root (for $\lambda_2 > 0$) or two roots (for $\lambda_2 < 0$). If the root is unique, we must necessarily have $s = n^{-n}$, so for lower s values the x_k must be distributed over either one of two different values. We therefore put

$$\begin{aligned} x_i &= \frac{a}{n_1}, & i &= 1, \dots, n_1, \\ x_i &= \frac{a}{n_2}, & i &= n_1 + 1, \dots, n, \quad n_2 = n - n_1; \end{aligned} \quad (31)$$

note that $n_1 = 0$ or $n_1 = n$ is not allowed here. The x_i then sum automatically to 1, and the value of a is determined by

$$a^{n_1} (1 - a)^{n_2} = sn_1^{n_1} n_2^{n_2}, \quad (32)$$

where we may take the highest of the two roots without loss of generality. We now have reduced the minimization problem from one over an $(n-2)$ -dimensional space to one over just the set $n_1 = 1, 2, \dots, n$. A computer search

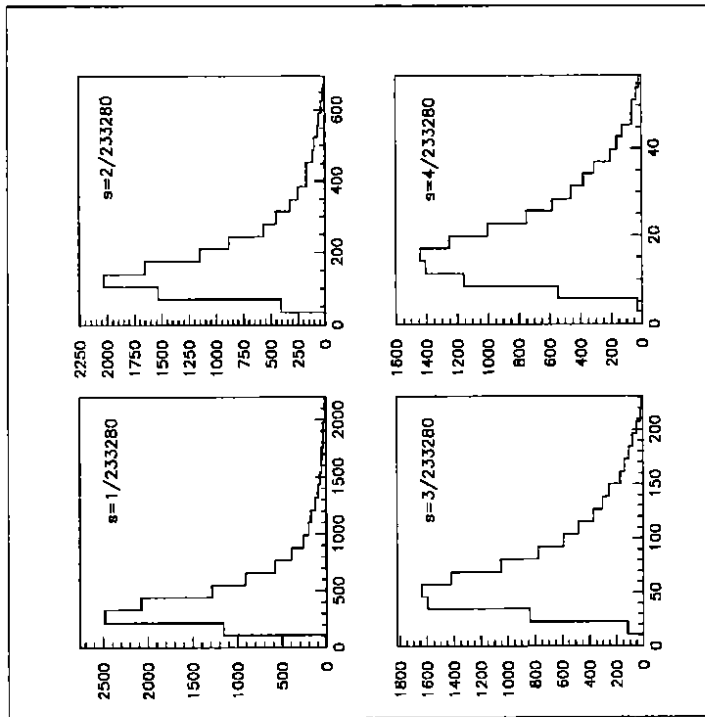


Figure 4: Weight distributions for $n = 6$.

indicates that, for any n and σ , the minimum is obtained for $n_1 = n - 1$. We conclude that

$$R \geq (na - n - 1) \log \left(\frac{a}{(1-a)(n-1)} \right), \quad (33)$$

where a is the largest solution in $(0,1)$ of

$$a(1-a)^{n-1} = s(n-1)^{n-1}. \quad (34)$$

The denominator contains the factor $\log \sigma x_{\max}$, and is maximized if x_{\max} is maximal. Now, we have the inequality

$$(1 - x_{\max})^{n-1} > x_{\max}(1 - x_{\max})^{n-1} \geq s(n-1)^{n-1}, \quad (35)$$

We therefore find

$$x_{\max} < 1 - (n-1)s^{1/(n-1)}, \quad (36)$$

and this provides an upper bound on the denominator of Eq.(25). By combining the two bounds, we find an absolute upper bound on the weight for algorithm B. Note, however, that the numerator is minimal for $(n-1)$ equal, 'large' values and one 'small' one, and the numerator is maximal for the different situation where one value is 'large' and the other ones equal and 'small'. Therefore, the real upper limit on the weight is again smaller than the above bound. An upper bound on the density (lower bound on the weight) can be derived by similar arguments.

To round off the argument, we have to discuss what happens around the 'symmetrical point' $x_i = 1/n$ for all i , $s = n^{-n}$, since there the numerator vanishes. Therefore, put

$$x_i = \frac{1 + \epsilon_i}{n}, \quad |\epsilon_i| < \epsilon, \quad \sum_i \epsilon_i = 0. \quad (37)$$

Working to leading order in ϵ , we then see that

$$\begin{aligned} \sum_i (nx_i - 1) \log x_i &\sim \sum_i \epsilon_i^2 + \mathcal{O}(\epsilon^3), \\ \log \sigma &\sim \log n + \frac{1}{2n} \sum_i \epsilon_i^2 + \mathcal{O}(\epsilon^3), \\ \log x_{\max} &\sim -\log n + \epsilon_{\max} + \mathcal{O}(\epsilon^2), \end{aligned} \quad (38)$$

and so we see that, when $\epsilon \rightarrow 0$, the density will actually diverge for $n > 3$, and the weight go to zero rather than to infinity, which would make our algorithms less attractive.

References

- [1] F. Wilczek, *Phys. Rev. Lett.* **39** (1977) 1304; H. Georgi, S.L. Glashow, M.E. Mahacek, and D.V. Nanopoulos, *Phys. Rev. Lett.* **40** (1978) 692; see also Z. Kunszt and W.J. Stirling, Proceedings of ECFA Large Hadron Collider Workshop, Aachen, 1990 (ECFA 90-133 and CERN 90-10), p.427.
- [2] L. Devroye, *Non-Uniform Random Variate Generation*, Springer (New York) 1986.
- [3] S.D. Ellis, R. Kleiss and W.J. Stirling, *Comp. Phys. Comm.* **40** (1986) 359.