Unsorting: an easy way of generating random permutations of an array

Les Hatton
CISM, Kingston University*

07 Apr 2007

1 Overview

This short paper was prompted by a need to generate random permutations of an array whilst trying to reduce the global error on a series of N coupled ordinary differential equations. Each equation was related to a different geometric point and the order in which equations were differenced could cause a geometric error to grow so a partially successful attempt to control this was based around choosing a different permutation order for each equation at each time step.

This in turn led to the need to generate such a permutation in a reasonably efficient way. This must have been done before somewhere but I couldn't find it so note that a very simple way of doing this involves taking your favourite array sorting program and randomising the way it re-orders pairs of elements. This is guaranteed to generate a permutation of the original array whose degree of disorder depends on the sorting algorithm and also the degree of randomness in the re-ordering.

1.1 Bubble unsort

A C implementation of a standard bubble sort is shown below for an array of n integers, a[i], i = 0,..,n-1.

```
#include <stdlib.h>

void
bubbleunsort(
     int  a[],
     int  n
)
{
     int  i, j, t;

     for ( i = (n-1); i >= 0; i-- )
```

*L.Hatton@kingston.ac.uk, lesh@oakcomp.co.uk

```
    {
         for ( j = 1; j <= i; j++ )
         {
              if ( a[j-1] > a[j] )
              {
                   t         = a[j-1];
                   a[j-1]    = a[j];
                   a[j]      = t;
              }
         }
    }
}
```

To turn this into a permutation generator, simply randomise the ordering line

```
if ( a[j-1] > a[j] )
```

into something like:-

```
if ( rand() > (RAND_MAX/2) )
```

recalling that the standard C function rand() generates a random integer between 0 and RAND_MAX. Now the bubble sort normally moves array elements which are out of position, one place at a time into their final sorted position. When unsorting, this means that elements will not normally move very far from their original position. If a greater spread is required, the above can be biased by introducing (RAND_MAX/3) to force more change. Alternatively, a sorting method which moves elements more than one place in any exchange could be used, to achieve the same end more efficiently, for example the heapsort, [1].

Finally it should be noted that rand() may not indeed be very random on older implementations which use low-order bits. As we are reminded by [2], the high-order bits should always be used.

References

[1] Sedgewick, R (1990) *Algorithms in C*, Addison Wesley, Reading, Mass ISBN 0-201-51425-7

[2] Press, W.H., Flannery B.P., Teukolsky S.A. and Vetterling, W.T. (1992) *Numerical Recipes in C: the art of scientific computing*, Cambridge University Press, ISBN 0-521-43108-5, 2nd edition, p. 277.