REFERENCES

Bhutra, S.,and Payatakes, A.C., J. Aerosol Sci., Vol. 10, (1979).

Billing, C.E., Ph.D. Dissertation, Calif. Int. Tech., Pasadena (1966).

Davies, C.N., Air Filtration, 1-80, (1973).

Edgar, T.F.,and Himmelblau, D.M., Optimization of Chemical Process, 34-63, 188-202, (1989).

Emi, H., Okuyama, K.,and Yoshioka, N., J. Chem. Eng. Japan, Vol. 6, (1973).

Gene, H.H., Mohammed, S.S.,and Paul, D.C., Analytical, Numerical and Computational Methodesfor Science and Engineering, 69-107, 412-470, (1991).

Gordon M.B.,and Bruce, M.P., Ind. Eng. Chem. Process Des. Dev., Vol. 18, (1979).

Kanaoka, C., Emi, H.,and Myojo, T., J. Aerosol Sci., Vol. 11, (1980).

______, Emi, H.,and Tanthapanichakoon, W., AIChE Journal, Vol. 29, No. 6, (1983).

______,and Hiragi, S., J. Aerosol Sci., Vol. 21, (1990).

Klaus, W.,and Paul, A.B., Aerosol Measurement, 1-38, 179-198, 843-857, (1993).

Kuwabara, S., J. Phy. Soc. Japan, Vol. 14, (1959).

Lee, K.W.,and Gieseke, J.A., J. Aerosol Sci., Vol 11, (1980).

Myojo, T., Kanaoka, C.,and Emi, H., J. Aerosol Sci., Vol. 15, (1984).

Payatakes, A.C.,and Tien, C., J. Aerosol Sci., Vol 7, (1976a).

______,and Tien, C., Filtration and Separation, Vol. 14, (1976b).

______,and Tien, C., Powder Tech., Vol. 14, (1976c).

______,and Tien, C., AIChE Journal, Vol. 23, No. 2, (1977).

______,and Gradon, L., AIChE Journal, Vol. 26, No. 3, (1980a).

______,and Gradon, L., Chem. Eng. Sci., Vol. 35, (1980b).

Raduskvich, L.V., Colloid. J. U.S.S.R., Vol. 26, (1964).

Stechkina, I.B.,and Fuchs, N.A., Ann. Occup. Hyg., Vol. 9, (1966).

______, Kirst, A.A.,and Fuchs, N.A., Ann. Occup. Hyg., Vol. 12, (1969).

Tanthapanichakoon, W., Ph.D. Dissertation, University of Texas at Austin, Texas, (1978).

_______,and Kanaoka, C., Proceedings Fifth World Filtration Congress Japan, (1993).

Tien, C., Wang, C.S.,and Barot, D.T., Science, Vol. 196, (1977).

William, H.P., Brian, P.F., Saul, A.T.,and William, T.V., Numerical Recipes: The Art of Scientific Computing (FORTRAN Version), University of Cambridge, 289-293, (1990).

Wongsri, M., Tanthapanichakoon, W., Kanaoka, C.,and Emi, H., Advanced Powder Technol. Japan, Vol. 2, No. 1, (1991).

Yoshioka, N., Emi, H.,and Sone, H., Kagaku Kogaku (Chem. Eng. Japan), Vol. 33, (1969).

APPENDIX

APPENDIX A
RANDOM NUMBERS GENERATION

1. Methods of generating uncorrelated random numbers

When a computer subprogram is used, as in this study, to generate a sequence of random numbers, each particular sequence will be completely defined by its starting value or see. The term "pseudo-random numbers" is sometimes used to describe such numbers.

The algorithms used to generate the uncorrected uniform and normal random numbers in this study were developed by Pike and Hill (1966). The general procedure was: 1) select a nonrepetitive starting value, 2) generate a uniformly distributed random number lying between 0 and 1 and with a mean of 0.5, 3) if appropriate, use two uniform random numbers from 2) to generate a normally distributed random number with a mean of 0.0 and a standard deviation of 1.0, 4) repeat steps 2) and 3) until a sequence of random numbers have been obtained.

Pike and Hill recommended that the starting value should be an odd integer between 1 and 67,108,863. Subroutine START breaks up a starting value into two integral numbes in order to prevent overflowing without having to use double-precision integers on the IBM compatible personal computer. To generate several starting values for separate sequence of random numbers, the following procedure was adopted: 1) an odd number between 1 and 67,108,863 was selected from a random number table, 2) the number was used to generate a sequence of uniform random numbers, 3) subroutine RESTRT was used to transform each uniform random number in the sequence in to a new starting value later used.

To break up the cyclic nature of a very long sequence, a new starting value is used for each record of the Monte-Carlo simulation.

2. Random number generators

2.1 Starting value

```
      SUBROUTINE START(ISTRT)
C
C     SUBROUTINE START TRANSFORMS THE STARTING VALUE ISTRT
C     SO THAT IT MAY BE USED BY SUBROUTINE RANDOM OR
C     SUBROUTINE RND WITH OUT CAUSING OVERFLOWING OF THE
C     INTEGER NUMBER THE VALUE OF ISTRT MUST BE AND ODD
C     NUMBER  OBTAINED FROM A TABLE OF RANDOM NUMBER,
C     PREFERABLY BETWEEN 42758321 AND 67108863
C     ISTRT=STARTING VALUE
C
      COMMON/CMN2/ RUNF,RNRM,IY1,IY2
      I=2*(ISTRT/2)+1
      IY2=I/16384
      IY1=I-16384*IY2
      RETURN
      END

      SUBROUTINE RESTRT
C
C     SUBROUTINE RESTRT YIELDS A NON-REPETITIVE STARTING
C     VALUE  FOR THE RANDOM NUMBER GENERATORS
C
      COMMON/CMN2/ RUNF,RNRM,IY1,IY2
      I=0
      J=0
      DO 100 K=1,3
      CALL RANDOM
      I=I+IY1
      J=J+IY2
100   CONTINUE
      IY1=MOD(123*I,16384)
      IY2=MOD(789*J,4096)
      RETURN
      END
```

2.2 Uniform random number generator

Subroutine RANDOM was used to generate a uniform random numbers, whose values lie between 0 and 1.

```
      SUBROUTINE RANDOM
C
C     SUBROUTINE RANDOM GENERATES A UNIFORM RANDOM
C     NUMBER WITH A MEAN OF 0.5 AND A RANGE OF 0.0 TO 1.0
C     THE SEQUENCE OF RANDOM NUMBERS HAS A LIMITING CYCLE
C     LENGTH  OF 16,777,216
C
      COMMON/CMN2/ RUNF,RNRM,IY1,IY2
      IY=3125*IY1
      N=IY/16384
      IY1=IY-16384*N
      IY=3125*IY2+N
      IY2=IY-4096*(IY/4096)
      RUNF=DBLE(16384*IY2+IY1)/67108864.0D0
      RETURN
      END
```

2.3 Normal random numbe generator

Subroutine RND was used to generate a normal random number with a mean of 0.0 and a standard deviation of 1.0.

```
      SUBROUTINE RND
C
C     SUBROUTINE RND GENERATES A STANDARD NORMAL RANDOM
C     NUMBER USING  TWO UNIFORM RANDOM NUMBERS
C
      COMMON/CMN2/ RUNF,RNRM,IY1,IY2
C
C     CALL RANDOM(FIRST TIME)
C
      IX=3125*IY1
      N=IX/16384
      IX=IX-16384*N
      IY=3125*IY2+N
      IY=IY-4096*(IY/4096)
      R=DBLE(16384*IY+IX)/67108864.0D0
      X=SQRT(-2.0*ALOG(R))
C
C   CALL RANDOM(SECOND TIME)
C
      IX=3125*IX
      N=IX/16384
      IY1=IX-16384*N
      IY=3125*IY+N
      IY2=IY-4096*(IY/4096)
      R=DBLE(16384*IY2+IY1)/67108864.0D0
      RNRM=X*COS(6.283185*R)
      RETURN
      END
```

APPENDIX B

DOWHILL SIMPLEX METHOD IN MULTIDIMENSIONS

The downhill simplex method is due to Nelder and mead (1965). The method requires only function evaluations, not derivatives. It is not very efficient in term of the number of function evaluations that it requires. However, the downhill simplex method may frequently be the best method to use if the figure of merit is "get something working quickly" for a problem whose computational burden is small.

A simplex is the geometrical figure consisting, in N dimensions, of N+1 points (or vertices) and all their interconnecting line segments, polygonal faces, etc. In two dimensions, a simplex is a triangle. In three dimensions it is a tetrahedron, not necessarily the regular tetrahedron. In general we are only interested in simplexes that are nondegenerate, i.e. which enclose a finite inner N-dimensional volume. If any point of a nondegenerate simplex is taken as the origin, then the N other points define vector directions that span the N-dimensional vector space.

For multidimensional minimization, the best we can do is give our algorithm a starting guess, that is, an N-vector of independent variables as the first point to try. The algorithm is then supposed to makes its own way downhill through the unimaginable complexity of an N-dimensional topography, until it encounters a (at least local) minimum.

The downhill simplex method must be started not just with a single point, but with N+1 points, defining an initial simplex. If you think of one of these points (it matters not which) as being your initial starting point A_0, then you can take the other N points to be

$$A_i=A_0+\sigma\, i_i \qquad \text{(B.1)}$$

where the i_i's are N unit vectors, and where σ is constant which is your guess of the problem's characteristic length scale.

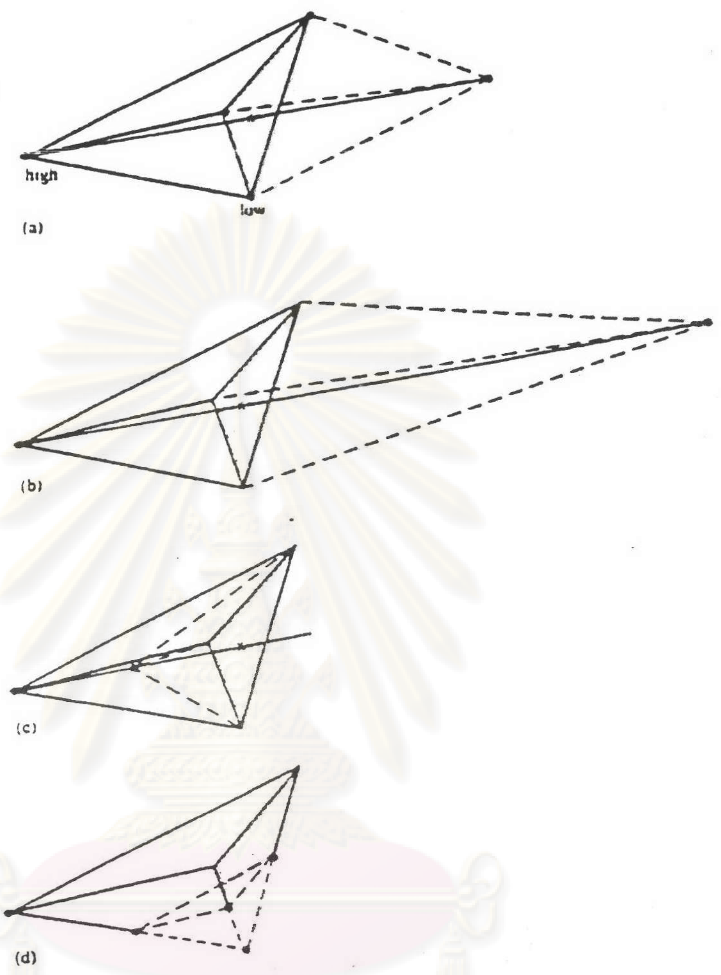

Figure B.1. Possible outcomes for a step in the downhill simplex method. The simplex at the begining of the step, here a tetrahedron, is drawn with solid lines. The simplex at the end of the step (drawn dashed) can be either (a) a reflection away from the high point, (b) a reflecton and expansion away from the high point, (c) a contraction along one dimension from the high point, or (d) a contraction along all dimensions toward the low point. An appropriate sequence of such steps will always converge to a minimum of the function.

The downhill simplex method now takes a series of steps, most steps just moving the point of the simplex where the function is largest ("highest point") through the opposite face of the simplex to a lower point. These steps are called reflections. and they are constructed to conserve the volume of the simplex (hence maintain its nondegeneracy). When it can do so, the method expands the simplex in one or another direction to take larger steps. When it reaches a "valley floor", the method contracts itself in the transverse direction and tries to ooze down the valley. If there is a situation where the simplex is trying to "pass through the eye of a needle", it contracts itself in all directions, pulling itself in around its lowest (best) point. The routine name SIMPLEX is intended to be descriptive of this kind of behavior; the basic moves are summarized in Figure B.1.

Termination criteria can be delicate in any multidimensional minimization routine. Without bracketing, and with more than one independent variable, we no longer the option of requiring a certain tolerance for a single independent variable We typically can identify one "cycle" or "step" of our multidimensional algorithm. It is then possible to terminate when the vector distance moved in that step is fractionally smaller in magnitude than some tolerance TOL. Alternatively, we could require that the decrease in the function value in the terminating step be fractionally smaller than some tolerance FTOL. Note that while TOL should not usually be smaller than the square root of the machine precision, it is perfectly appropriate to let FTOL be of order the machine precision (or perhaps slightly larger so as not to bediddled by roundoff).

Note well that either of the above criteria might be fooled by a single anomalous step that, for one reason or another, failed to get anywhere. Therefore, it is frequently a good idea to restart a multidimensional minimization routine at a point where it claims to have found a minimum. For this restart, you should reinitialize any nillary input quantities. In the downhill simplex method, for example method, for example, you should reinitialize N of the N+1 vertices of the simplex again by equation B.1, with A_0 being one of the vertices of the claimed minimum.

Restarts should never be very expensive; your algorithm did, after all, converge to the restart point once, and now you are starting the algorithm already there.

Consider, them, our N-dimensional SIMPLEX:

```
void SIMPLEX(double eni[]) {
long a;
struct  time t,t1;
double p[mp][np],y[mp],pr[nmax],prr[nmax],pbar[nmax];
double alpha,beta,gamma;
double ndim,mpts,rtol,ftol=0.001,ypr,yprr,y1,y2,avg;
int     i,j,iter,itmax,ihi,inhi,ilo,county;
alpha=1.0;     beta=0.2;      gamma=2.0;
y1=0;y2=0;county=0;avg=0.5;
itmax=500;
ndim=np;
mpts=ndim+1;
iter=0;
for(i=0;i<ndim;i++)  p[0][i]=eni[i];
for(i=0;i<mpts;i++)  if(i!=0) for(j=0;j<ndim;j++) p[i][j]=p[0][j]+0.1*(i+j);
for(i=0;i<mpts;i++)  y[i]=FUNCT(p[i]);
do{
        ilo=1;
        if(y[0]>y[1]){
                ihi=1;
                inhi=2;
        }
        else {
                ihi=2;
                inhi=1;
        }
        for(i=0;i<mpts;i++){
                if(y[i]<y[ilo])  ilo=i;
                if(y[i]>y[ihi]){
                        inhi=ihi;
                        ihi=i;
                }
                else if(y[i]>y[inhi]) {
                        if(i!=ihi) inhi=i;
                }
        }
        y1=y[ilo];
        if(y1==y2){
                county++;
                if(county>5){
                        avg=avg*0.9;
                        county=0;
                }
        }
        else{
                y2=y[ilo];
                county=0;
                avg=0.5;
        }
        rtol=fabs(y[ihi]-y[ilo]);
        if(iter==itmax) exit(1);
        if(rtol>ftol) {
                iter=iter+1;
                for(i=0;i<ndim;i++)     pbar[i]=0;
                for(i=0;i<mpts;i++) {
                        if(i!=ihi) for(j=0;j<ndim;j++) pbar[j]=pbar[j]+p[i][j];
                }
                for(i=0;i<ndim;i++){
                        pbar[i]=pbar[i]/ndim;
                        pr[i]=(1+alpha)*pbar[i]-alpha*p[ihi][i];
                }
                ypr=FUNCT(pr);
```

```
if(ypr<=y[ilo]){
        for(i=0;i<ndim;i++) prr[i]=gamma*pr[i]+(1-gamma)*pbar[i];
        yprr=FUNCT(prr);
        if(yprr<y[ilo]){
                for(i=0;i<ndim;i++) p[ihi][i]=prr[i];
                y[ihi]=yprr;
        }
        else{
                for(i=0;i<ndim;i++) p[ihi][i]=pr[i];
                y[ihi]=ypr;
        }
}
else if(ypr>=y[inhi]){
        if(ypr<y[ihi]){
                for(i=0;i<ndim;i++) p[ihi][i]=pr[i];
                y[ihi]=ypr;
        }
        for(i=0;i<ndim;i++) prr[i]=beta*p[ihi][i]+(1-beta)*pbar[i];
        yprr=FUNCT(prr);
        if(yprr<y[ihi]){
                for(i=0;i<ndim;i++) p[ihi][i]=prr[i];
                y[ihi]=yprr;
        }
        else {
                for(i=0;i<mpts;i++){
                        if(i!=ilo){
                                for(j=0;j<ndim;j++){
                                        pr[j]=avg*(p[i][j]+p[ilo][j]);
                                        p[i][j]=pr[j];
                                }
                                y[i]=FUNCT(pr);
```

```
                                        }
                                }
                        }
                }
                else {
                        for(i=0;i<ndim;i++) p[ihi][i]=pr[i];
                        y[ihi]=ypr;
                }
        }
}while(rtol>ftol);
}
```

APPENDIX C
CLEAN FIBER EFFICIENCY

R	Stoke number St										
	0	0.2	0.4	0.6	0.8	1	1.2	1.4	1.6	1.8	2
0.05	0.0022701	0.0024462	0.0032455	0.0047477	0.0077754	0.0165395	0.0632363	0.1381846	0.1967505	0.244112	0.2834522
0.06	0.0032478	0.003495	0.0046171	0.0067172	0.0109235	0.0228367	0.0732569	0.1443636	0.2018938	0.2489612	0.2882491
0.07	0.0043923	0.0047207	0.0062084	0.0089811	0.0144973	0.0296699	0.0823701	0.1505501	0.2071566	0.2539321	0.2931538
0.08	0.0057004	0.0061186	0.0080111	0.0115225	0.0184518	0.0368496	0.0908671	0.1567446	0.2125229	0.2590135	0.2981597
0.09	0.0071689	0.0076849	0.0100175	0.0143236	0.022743	0.0442242	0.0989374	0.1629505	0.2179804	0.2641957	0.303259
0.1	0.0087953	0.0094154	0.0122191	0.0173681	0.027332	0.051682	0.1066984	0.1691702	0.2235212	0.26947	0.308447
0.11	0.0105765	0.0113072	0.0146084	0.0206412	0.0321822	0.0591501	0.1142319	0.1754052	0.2291385	0.2748306	0.3137158
0.12	0.0125097	0.0133555	0.0171788	0.0241277	0.0372582	0.0665815	0.1215928	0.1816592	0.234826	0.2802708	0.3190648
0.13	0.0145927	0.0155577	0.019924	0.0278142	0.0425298	0.0739554	0.1288243	0.187932	0.2405813	0.2857882	0.3244873
0.14	0.0168221	0.0179101	0.0228367	0.0316887	0.0479707	0.0812629	0.1359549	0.1942256	0.2463961	0.2913755	0.3299809
0.15	0.019196	0.0204096	0.0259116	0.0357391	0.0535586	0.0885033	0.1430087	0.2005426	0.2522712	0.2970334	0.3355427
0.16	0.0217121	0.0230527	0.0291424	0.039955	0.0592722	0.0956834	0.1500036	0.2068833	0.2582029	0.3027543	0.3411685
0.17	0.0243669	0.0258368	0.0325244	0.0443262	0.0650975	0.102811	0.1569539	0.2132485	0.2641878	0.308537	0.3468593
0.18	0.0271597	0.028759	0.036052	0.0488435	0.0710187	0.1098934	0.1638705	0.2196402	0.2702249	0.3143811	0.3526084
0.19	0.0300868	0.0318156	0.0397203	0.053498	0.0770255	0.1169391	0.1707637	0.2260577	0.2763116	0.3202814	0.3584177
0.2	0.033147	0.0350052	0.0435254	0.0582843	0.0831086	0.1239559	0.1776414	0.2325036	0.2824469	0.3262385	0.3642822

Table C.1. The clean fiber efficiency of inertial impaction solving by Runge Kutta 4th method based on the limitting trajectory theory

R	Stoke number St										
	0	0.2	0.4	0.6	0.8	1	1.2	1.4	1.6	1.8	2
0.05	0.0019311	0.0038341	0.0057371	0.0076401	0.0095431	0.0114462	0.0133492	0.0152522	0.0171552	0.0190582	0.0209613
0.06	0.0027633	0.0054586	0.0081539	0.0108492	0.0135445	0.0162398	0.0189351	0.0216304	0.0243257	0.027021	0.0297163
0.07	0.0037378	0.0073471	0.0109564	0.0145657	0.018175	0.0217843	0.0253936	0.029003	0.0326123	0.0362216	0.0398309
0.08	0.004852	0.0094909	0.0141299	0.0187688	0.0234078	0.0280467	0.0326857	0.0373246	0.0419636	0.0466025	0.0512415
0.09	0.0061034	0.0118817	0.01766	0.0234383	0.0292165	0.0349948	0.0407731	0.0465514	0.0523297	0.0581079	0.0638862
0.1	0.0074897	0.0145112	0.0215327	0.0285543	0.0355758	0.0425973	0.0496188	0.0566404	0.0636619	0.0706834	0.077705
0.11	0.0090084	0.0173715	0.0257345	0.0340975	0.0424606	0.0508236	0.0591867	0.0675497	0.0759127	0.0842758	0.0926388
0.12	0.0106574	0.0204547	0.0302519	0.0400492	0.0498465	0.0596438	0.069441	0.0792383	0.0890356	0.0988328	0.1086301
0.13	0.0124344	0.0237532	0.035072	0.0463908	0.0577096	0.0690283	0.0803471	0.0916659	0.1029847	0.1143035	0.1256223
0.14	0.0143374	0.0272596	0.0401818	0.053104	0.0660263	0.0789485	0.0918707	0.1047929	0.1177152	0.1306374	0.1435596
0.15	0.0163641	0.0309665	0.0455688	0.0601711	0.0747735	0.0893758	0.1039781	0.1185804	0.1331828	0.1477851	0.1623874
0.16	0.0185128	0.0348667	0.0512206	0.0675745	0.0839284	0.1002823	0.1166361	0.13299	0.1493439	0.1656978	0.1820517
0.17	0.0207813	0.0389531	0.0571249	0.0752967	0.0934685	0.1116403	0.1298121	0.1479839	0.1661557	0.1843275	0.2024993
0.18	0.0231679	0.0432189	0.0632699	0.0833209	0.1033719	0.1234228	0.1434738	0.1635248	0.1835758	0.2036268	0.2236778
0.19	0.0256707	0.0476571	0.0696436	0.09163	0.1136165	0.1356029	0.1575894	0.1795758	0.2015623	0.2235487	0.2455352
0.2	0.0282878	0.0522611	0.0762343	0.1002076	0.1241808	0.1481541	0.1721273	0.1961005	0.2200738	0.244047	0.2680203

Table C.2. The clean fiber efficiency of inertial impaction using Stechkina's equation (1969)

R	Peclet number Pe										
	200	500	1000	1500	2000	2500	3000	3500	4000	4500	5000
0.05	0.0994	0.0577	0.0393	0.0319	0.0277	0.0249	0.0229	0.0214	0.0203	0.0193	0.0185
0.06	0.1030	0.0608	0.0422	0.0346	0.0303	0.0275	0.0255	0.0240	0.0228	0.0218	0.0210
0.07	0.1067	0.0640	0.0451	0.0375	0.0331	0.0303	0.0282	0.0267	0.0255	0.0245	0.0237
0.08	0.1106	0.0674	0.0483	0.0405	0.0361	0.0332	0.0312	0.0296	0.0283	0.0273	0.0265
0.09	0.1145	0.0709	0.0516	0.0437	0.0392	0.0363	0.0342	0.0326	0.0314	0.0303	0.0295
0.1	0.1186	0.0745	0.0550	0.0470	0.0425	0.0396	0.0374	0.0358	0.0345	0.0335	0.0326
0.11	0.1227	0.0783	0.0586	0.0505	0.0459	0.0429	0.0408	0.0391	0.0378	0.0368	0.0359
0.12	0.1270	0.0822	0.0622	0.0541	0.0495	0.0464	0.0443	0.0426	0.0413	0.0402	0.0393
0.13	0.1314	0.0862	0.0660	0.0578	0.0531	0.0501	0.0479	0.0462	0.0449	0.0438	0.0429
0.14	0.1359	0.0903	0.0700	0.0617	0.0569	0.0538	0.0516	0.0499	0.0486	0.0475	0.0466
0.15	0.1405	0.0945	0.0740	0.0656	0.0609	0.0577	0.0555	0.0537	0.0524	0.0513	0.0504
0.16	0.1452	0.0989	0.0782	0.0697	0.0649	0.0617	0.0594	0.0577	0.0563	0.0552	0.0543
0.17	0.1500	0.1033	0.0824	0.0739	0.0690	0.0658	0.0635	0.0618	0.0604	0.0592	0.0583
0.18	0.1549	0.1079	0.0868	0.0782	0.0733	0.0700	0.0677	0.0659	0.0645	0.0634	0.0624
0.19	0.1598	0.1125	0.0913	0.0826	0.0776	0.0744	0.0720	0.0702	0.0688	0.0676	0.0667
0.2	0.1649	0.1172	0.0959	0.0871	0.0821	0.0788	0.0764	0.0746	0.0732	0.0720	0.0710

Table C.3. The clean fiber efficiency of convective diffusion using Stechkina's equation (1969)

APPENDIX D

The diffusion coefficient D_{BM}, Cunningham slip correction fatcor C_m, and mobility β of pariticles in air at 20°C

Radius of particle (μm)	D_{BM} (cm²/sec)	C_m	β
0.001	1.3×10^{-2}	110.2	3.19×10^{11}
0.005	5.3×10^{-4}	22.5	1.31×10^{10}
0.01	1.4×10^{-4}	11.56	3.35×10^{9}
0.02	3.6×10^{-5}	6.10	8.84×10^{8}
0.025	2.4×10^{-5}	5.03	5.83×10^{8}
0.05	6.8×10^{-6}	2.89	1.68×10^{8}
0.1	2.2×10^{-6}	1.88	5.45×10^{7}
0.15	1.24×10^{-6}	1.57	3.08×10^{7}
0.2	8.4×10^{-7}	1.42	2.06×10^{7}
0.225	7.2×10^{-7}	1.375	1.77×10^{7}
0.25	6.3×10^{-7}	1.334	1.55×10^{7}
0.5	2.76×10^{-7}	1.166	6.75×10^{6}
1.0	1.3×10^{-7}	1.083	3.13×10^{6}
2.0	6.16×10^{-8}	1.042	1.51×10^{6}
5.0	2.4×10^{-8}	1.017	5.88×10^{5}

VITA

Mr.Adisorn Areephant was born on November 28, 1972 in Bangkok, Thailand. He attended Yothinburana High school in Bangkok and graduated in 1990. In April 1994, he recieved his Bachelor Degree of Engineering in Chemical Engineering from Faculty of Engineering, Chulalongkorn University, Bangkok, Thailand. He continued his Master's study at Chulalongkorn University in 1990. He was granted the degree in April, 1996.