# Modelling Sedimentation

By

C.H. Hesse
Department of Statistics
University of California, Berkeley

Technical Report No. 125
November 1987
(revised)

Department of Statistics
University of California
Berkeley, California

# Modelling Sedimentation

By

C.H. Hesse
Department of Statistics
University of California, Berkeley
Berkeley, California 94720

# Modelling Sedimentation

C.H. Hesse
Department of Statistics
University of California, Berkeley

Abstract

Particle sedimentation in viscuous fluids is a phenomenon of great practical importance. For a long time it has proved to be quite inaccessible to any modelling with relevance in applications. Building on earlier work of Pickard and Tory this paper presents some further developments for a stochastic model for sedimentation. Various aspects including modelfitting, parameter estimation, implementation, and simulation on a computer equipped with high resolution graphics are described in detail.

AMS (1980) subject classification: Primary 62M09 62P99; secondary 60J70.

Key words: Stochastic modelling, particle sedimentation, transit times, Ornstein-Uhlenbeck process, conditional maximum likelihood, simulation.

---

The present author became involved in this project as a graduate student of D. Pickard at Harvard University. Part of the work was done while I was visiting D. Pickard in 1986 at the Department of Mathematics and Statistics at Queens University in Kingston, Canada. I would like to thank the Department for generously funding the computational activities and the late Dr. David Pickard for his encouragement, support and friendship.

## 1. Overview.

This paper presents the present state of a stochastic model for sedimentation. Here and below by sedimentation is meant the collection of phenomena arising in a two-phase solids-fluid system that evolves from some initial state under the influence of gravity. In theoretical terms this is simply a problem of continuum mechanics and it is possible, at least in principle, to set up the equations of motion for all particles and fluid flows. However, since the number of particles involved is quite large the problem becomes very high dimensional and hence intractable.

To handle this intractability, Kynch (1952) has introduced several restrictive simplifying assumptions. These simplifications lead to solutions of the equations of motion not for individual particle trajectories but for average particle behavior. In this sense Kynch's theory is still fluid-dynamically sound but clearly incapable to interpret any phenomena arising from the variability of particle behavior.

A stochastic model was first introduced by Pickard and Tory (1977). It has subsequently been extended and improved upon (Pickard and Tory (1979), (1982), (1986)). The present author became involved in this project as a student of D. Pickard in 1986. Part of the work was done while I was visiting D. Pickard at Queens University in Kingston, Canada. The purpose of this paper is to present further developments and some recent work for a refined version of the model.

Section 2 introduces the present state of the stochastic model. The model attempts to steer an intermediate course between intractability (the fluid dynamics approach) and overly restrictive and unrealistic assumptions (Kynch's approach). It does so by using fluid dynamic methods to model individual particle trajectories and stochastic methods to analyze ensemble behavior.

Particle velocities are assumed to be governed by a parametrized equicontinuous family of Ornstein-Uhlenbeck processes. Since the Ornstein-Uhlenbeck parameters are taken to be functions of a more fundamental parameter, local particle concentration, this leads to a nested parametrization scheme. Since local particle concentration changes as the system evolves, the structure of the model is similar, in general, to that of random processes in random environments. In the special case considered here process parameters are functions of functionals over the state space of the system.

Section 3 introduces a modelfitting procedure. Due to the nested two-stage parametrization scheme, parameter estimation and modelfitting has been difficult. No satisfactory and non ad-hoc method existed previously. Section 3 gives such a method. It is based on recently developed approximations to the first-passage density of an integrated Ornstein-Uhlenbeck process to a straight line boundary (see Hesse (1990)

for the best available approximation). This procedure allows to make use of the extensive industrial data bases containing mostly transit-times of sedimenting particles (Johne (1965)).

Section 4 deals with computational aspects. The model easily lends itself to implementation and simulation. Incrementally, particles individually and simultaneously evaluate the relative (with respect to themselves) configuration of the entire particle ensemble and determine an update of the fundamental parameter local concentration. Then particles perform velocity transitions governed by Ornstein-Uhlenbeck processes with these updated parameters. This leads to new velocities, new positions, new configurations, and ultimately to new parameters for the Ornstein Uhlenbeck velocity processes. This scheme can be made into an algorithm for incremental system evolution.

The model has been implemented on an Apollo Domain DN 600 with high resolution colour graphics driven by the software package PRIMH. However, it can also be implemented on other devices with graphics tools such as Suns, etc. The implemented model constitutes an enormously powerful exploratory tool for hydrologists and rheologists. The FORTRAN source code for a working implementation of the model on an Apollo DN 600 appears in the Appendix.

## 2. A Stochastic Model for Sedimentation.

The stochastic model advanced in this section attempts to steer an intermediate course between intractability (hydrodynamics approach) and unrealistic, overly restrictive assumptions (Kynch's approach). In doing so it combines hydrodynamic as well as stochastic concepts: Individual particle trajectories are modeled according to fluid dynamic principles and ensemble behavior is treated stochastically. We proceed by explaining this fruitful symbiosis.

One of the facts of hydrodynamics is that identical particles in the same environment exhibit the same behavior. Environment of a particle is here interpreted in a very broad sense and in particular includes positions of other particles, fluid flows and possibly internal pressures and external forces. If a particle has exact ''knowledge'' of all of these parameters it invokes the laws of physics and ''computes'' its incremental trajectory.

But clearly, for the purpose of modelling, environment in the above sense is a much too high dimensional parameter. An attempt is therefore made to summarize environment by a few summary parameters which try to capture the main determinants of incremental particle behavior and to combine the remaining factors to velocity variation at these given parameter values. A particle equipped with only a partial

knowledge of its full environment (given by the values of the summary parameters) selects its velocity transitions from a distribution parametrized by these summary parameters.

This principle also conveys a great deal of insight into the evolution of the entire system during sedimentation. At time t, each particle individually and all particles simultaneously consult their environments and determine their individual summary parameters. In the family of stochastic processes governing their velocity transitions, particles then adjust the parameters to the updated values and perform velocity transitions according to these stochastic processes. Incrementally, this leads to new velocities, hence to new positions, new environments and ultimately to new summary parameters. Then, again, parameter values are adjusted and velocity transitions are performed, and so on.

The stochastic model utilizes the above mechanism with only one summary parameter: local particle concentration, interpreted as a kernel-smoothed version of relative configuration. Hence, the fundamentals of the model are simple and intuitive. It rests on only the following two concepts:

(a) Parametrization by a summarized version of environment (local concentration)

(b) A family of parametrized stochastic processes.

In addition to (a) and (b) there is a bridge connecting these two features of the model: equicontinuity.

These three pillars of the model are described in detail in (2.1), (2.2), and (2.3).

## 2.1. Parametrization by a Summary of Environment

In the absence of external forces (except gravity) and internal pressures, the sole determinant of incremental particle evolution is relative particle configuration. This is, for a given particle, the set of pairwise distance vectors to other particles. However, experiments confirm that relative configuration exerts influence on particle motion mainly through concentration and more especially through local concentration (Happel and Brenner (1965)), and hence justifies the choice of local particle concentration as the major summary parameter of environment.

In order to define local particle concentration it is convenient to consider the following space-time process $P(x,t)$ as a descriptive device for system evolution in the container D. Let

$$P(x,t) = \begin{cases} 1 \text{ if } x \in D \subset \mathbf{R}^3 \text{ is in solid phase at time } t \\ 0 \text{ if } x \in D \subset \mathbf{R}^3 \text{ is in fluid phase at time } t \end{cases}$$

Then $\{P(x,t): x \in D, t \geq 0\}$ describes the space-time evolution of the sedimenting system.

**Definition:** (Local Solids Concentration $c(x,t)$)

Local solids concentration is a kernel-smoothed version of relative configuration:

$$c(x,t): D \times [0,\infty) \rightarrow [0,1]$$

$$c(x,t) = \int_D K(x - x') P(x',t)\, dx'$$

for some kernel $K: \mathbf{R}^3 \rightarrow [0,\infty]$. K integrates to one and is usually taken to be unimodal with mode at the origin.

It is expected that many kernels K will lead to qualitatively similar results when implemented. The actually utilized kernel function in the implementation was chosen on the basis of computational considerations.

## 2.2. Stochastic Processes

A particle at $x_0$ at time $t_0$ samples its incremental velocity transitions from a stochastic process parametrized by $c(x_0, t_0)$. Here we motivate a reasonable choice for the family of stochastic processes. This is best understood from a consideration of the various forces acting upon each particle, namely friction $F_f = -\tau V(x,t)$, gravitation $F_g = mg$ and the Langevin force $F_s$.

Gravitation is proportional to particle mass m. Friction is taken to be proportional to particle velocity $V(x,t)$ but its direction is opposite to velocity (Stokes friction). In addition, there is a force which is best thought of as stochastic. It is due to thermal movements of the fluid molecules and becomes more and more relevant the smaller particle size and particle mass are. Typically, sedimentation deals with particle sizes of orders of magnitude where this stochastic force cannot be ignored. It will be modelled as white noise.

With the above-mentioned forces, Newtons law then translates into the following simple stochastic differential equation

$$m\, dV = m\, g - \tau\, V + dW \tag{2.1}$$

where W is a Wiener process. The solution is given by the Ornstein-Uhlenbeck process with drift. A parametrized family of drifting Ornstein-Uhlenbeck processes governing the velocity transitions of individual particle is therefore a reasonable choice for modelling purposes.

This reveals that the model employs a nested two-stage parametrization scheme: The Ornstein-Uhlenbeck processes are parametrized by drift $\mu$, frictional coefficient $\tau$, and

variance $\sigma^2$ but these parameters are themselves functions of the more fundamental parameter $c(x,t)$ which is a space-time functional. A particle remains under the influence of a single Ornstein-Uhlenbeck process as long as local concentration c remains constant along its trajectory.

2.3. Equicontinuity

At this point, clearly, a concept is necessary to provide the link between the family of Ornstein-Uhlenbeck processes as represented by their transition densities and the parametrization of these densities. This link is provided by the following

**Assumption:** The transition densities $f_c$ depend continuously on c, i.e. $\forall c, \varepsilon$ there is a $\delta(c,\varepsilon)$ such that $\forall u, v$ and $t \geq T_0$, for some $T_0$, the following holds

$$|c - c'| < \delta(c,\varepsilon) \Rightarrow |f_c(v,t,u) - f_{c'}(v,t,u)| < \varepsilon.$$

Hence, $f_c$ is an equicontinuous family on $\mathbf{V} \times \mathbf{V}$, the cross-product of the closed and bounded set of attainable velocities. The reason why uniformity is required for t larger than some $T_0$ only is that as $t \to 0$, $f_c$ approaches a delta-function and the above condition with $t \geq T_0$ replaced by $t \geq 0$ would be much too restrictive. As stated however, the above equicontinuity assumption is very mild. It can be justified on intuitive grounds: Small changes in local solids concentration of slurries are expected to effect only small changes in their behavior, in particular, they should cause only small perturbations of the velocity transition structure of particles.

Equicontinuity ties together the stochastic processes and their parametrization. It has far-reading implications which eventually make efficient simulation of the model possible. We list some immediate implications in a loose fashion. The proofs are left to the reader.

1. For all concentrations c and velocities u the transition densities $f_c(v,t,u)$ converge to a steady-state p.d.f. as $t \to \infty$: $\lim_{t\to\infty} f_c(v,t,u) = g_c(v)$, say.
2. The steady-state densities $g_c$ depend continuously on c.
3. The Ornstein-Uhlenbeck parameters $\mu(c)$, $\tau(c)$ and $\sigma^2(c)$ are continuous functions of c.
4. If local solids concentration remains essentially constant along the trajectory of a particle over a time interval $\Delta t$, then this particle can be thought of as remaining under the influence of a single Ornstein-Uhlenbeck process (as a uniform approximation) during $\Delta t$.

5. If in a region $D_0 \subset D$ of the sedimenting system local solids concentration remains sufficiently constant, then this region can be characterized (as a uniform approximation) by a single Ornstein-Uhlenbeck process.
6. Every sedimenting system can be characterized (as a uniform approximation) by finitely many Ornstein-Uhlenbeck processes.

The local structure of the stochastic model is very flexible and leaves considerable freedom for fine-tuning into particular applications. These degrees of freedom include the choice of stochastic processes (Ornstein-Uhlenbeck, Brownian motion, other diffusions) the resolution of the discretization of space and time, the choice of the kernel-smoother K and hence the exact definition of local solids concentration, and the shape of the parameter functions $\mu(c)$, $\sigma^2(c)$ and $\tau(c)$.

This flexibility makes it possible to design models which exhibit widely different qualitative behavior, such as the formation of dense particle layers, systematic or chaotic velocity variation, the formation of sharp or of diffuse interfaces, aso.

Due to this flexibility the usefulness of the model structure even goes beyond sedimentation and particle systems. Appropriately adjusted models of this type may also be useful in the study of traffic flow systems and in modelling the dynamics of population movements in demography.

## 3. Fitting the Model and Estimating Parameters.

In the context of the stochastic model for sedimentation, modelfitting amounts to determination of the parameter functions $\mu(c)$, $\tau(c)$ and $\sigma^2(c)$. Very little research has been done to investigate, even qualitatively, the shape of these functions. Some related work is due to Shannon et. al. (1982). Among other things this work explores the steady-state mean velocity as a function of *global* concentration $\gamma$ (the total proportion of particles).

As to $\tau$ and $\sigma^2$, their dependence on concentration does not seem to have been investigated at all. We propose, in this section, a modelfitting procedure that allows to make use of the existing extensive industrial data bases containing transit times of sedimenting particles, e.g. Johne (1965). Parameter estimation on the basis of particle transit times leads to the first-passage problem for an integrated Ornstein-Uhlenbeck process. The method is based on accurate global approximations of the first passage density to a straight line boundary for this process (Hesse 1990).

If a particle i starts at time t = 0 from $X_i(0) = 0$ with velocity $V_i(0) = v$ then

$$X_i(t) = \int_0^t V_i(s)\,ds \tag{3.1}$$

is the displacement process and

$$T_i(x) = \inf\{t \geq 0 : X_i(t) = x\} \tag{3.2}$$

is the first passage time to distance x. For the purpose of estimating the Ornstein-Uhlenbeck parameters from a sample of independent $T_i(x)$, $i = 1, \ldots, n$ it will be necessary to ensure (e.g. through the choice of x) that along the particles trajectory (until boundary crossing) local concentration is essentially constant (equal to c, say) so that in (3.1) $V_i(s)$ is an Ornstein-Uhlenbeck process with parameters $\mu(c)$, $\tau(c)$, $\sigma^2(c)$.

Hesse (1990) gives the following approximations to the density $f_{T(x)}(t)$ of T(x):

$$\hat{f}_{T(x)}(t) = x\tau(2\pi\sigma^2 t^3)^{-1/2} \exp(-\tau^2(x - \mu t)^2 / 2\sigma^2 t) \tag{3.3}$$

$$f^*_{T(x)}(t) = \left[\frac{3x - (v + \mu)t}{2t} - \frac{\tau}{8}(3(x - \mu t) - vt)\right]\phi(x) \tag{3.4}$$

where $\phi(x)$ is the density of the normal distribution with mean $= (v/\tau)(1 - e^{-\tau t}) + \mu t$ and variance $= (\sigma^2/2\tau^3)(2\tau t + 4e^{-\tau t} - e^{-2\tau t} - 3)$,

$$\tilde{f}_{T(x)}(t) = f^*_{T(x)}(t)(\Phi(\gamma(t)) + (\gamma(t))^{-1}\psi(\gamma(t))) \tag{3.5}$$

with $\psi$ and $\Phi$ being the standard normal density and distribution functions, respectively, and

$$\gamma(t) = [3x - (v + \mu)t - (\tau t/4)(3(x - \mu t) - vt)] / (\sigma t^{3/2} - \sigma\tau t^{5/2}/8).$$

$\hat{f}_{T(x)}(t)$ is accurate as an approximation to $f_{T(x)}(t)$ in the limit as the boundary becomes increasingly remote. Both $f^*_{T(x)}(t)$ and $\tilde{f}_{T(x)}(t)$ perform well for $(3x - vt)/2t$ large and $\beta$ small but $\tilde{f}_{T(x)}(t)$ is more accurate than $f^*_{T(x)}(t)$, see Hesse (1990).

Based on these approximations, Hesse (1990) also showed that the random variable

$$T^*(x) = \exp(T(x) + T^{-1}(x))$$

is approximately of algebraic tail-type in the sense that for t larger than some B

$$P(T^*(x) > t) \approx \text{const.} t^{-\delta}$$

with $\delta = 2\pi^{-1}\tau^2\sigma^{-2}\mu x$.

If $T_i(x)$, $i = 1, \ldots, n$ is an iid sample of passage times with order statistics

$$T_{(1)} \geq T_{(2)} \geq \cdots \geq T_{(n)}$$

and assuming that for some k

$$T_{(k+1)}(x) > B > T_{(k+2)}(x)$$

then

$$E(i) = i \ln(T_{(i)} / T_{(i+1)}) \quad i = 1, \ldots, k$$

are independent exponentially distributed with parameter $\delta$ and

$$\hat{\delta} = k / \sum_{i=1}^{k} E(i)$$

is the conditional maximum-likelihood estimator of $\delta$ given that the largest $(k + 1)$ order statistics of T(x) exceed B.

The number k of order statistics to be used, can be determined from the following consideration. If k has been chosen appropriately then $E(1), \ldots, E(k)$ is an independent sample from an exponential distribution with parameter $\delta$ and one may employ a standard goodness of fit test to test this hypothesis.

If k has been chosen too large then the null-hypotheses that the sample comes from an exponential distribution is false. Hence in practice one will start with a large k and repeatedly test for exponential distribution (reducing k by one in case of a rejection of the null-hypothesis). This procedure chooses k as the first such value for which the null-hypothesis is no longer rejected.

Hesse (1990) also demonstrates that $f^{*}_{T(x)}(t)$ is a very accurate approximation to $f_{T(x)}(t)$ in the lower tail and, in addition, that

$$T^{**}(x) = \exp((x - \mu T(x))^2 T^{-3}(x))$$

is also approximately of algebraic tail type with exponent $\delta = 3\pi / \sigma^2$. Hence the above method may be used to estimate $\sigma^2$.

Since $\mu$ may be estimated in some other fashion (Hesse (1990)) the conditional maximum likelihood procedure based on $T^{*}(x)$ and $T^{**}(x)$ allows to disentangle $\tau$ and $\sigma^2$.

## 4. Computational Issues.

Although conceptually the model is simple, it is a priori not obvious whether it lends itself to efficient implementation and simulation. The bottle-neck is the simultaneous updating of parameters, velocities, and positions. Local solids concentration updating is especially inefficient, since $c(x, t)$ is a function over a three-dimensional space domain. In view of these difficulties several simplifying features are introduced into the model structure.

1. It is convenient to transform to a standardized dimensionless version of the model which also reduces the number of parameter functions. This is achieved by measuring velocity v in multiples of steady-state drift velocity $\mu$ and time t in terms of correlation length $\tau t$, i.e.

$$v' = v\mu^{-1}, \quad t' = \tau t, \quad x' = \tau x \mu^{-1}$$

so that in this new coordinate system $\mu' = \tau' = 1$ and $\sigma' = \sigma / \mu\sqrt{\tau}$ is the only remaining relevant parameter

2. The original definition of local solids concentration,

$$c(x,t): \mathbf{R}^3 \times [0,\infty) \rightarrow [0,1]$$

with

$$c(x,t) = \int_D \mathbf{K}(x - x')\, P(x',t)\, dx'$$

is computationally troublesome for the above-mentioned reason. It is convenient to reduce the space coordinates to one dimension in the following way: Horizontal layers of the slurry are projected onto the vertical axis. Let $Q(y,t)$ denote the proportion of the layer (at height y above the bottom of the vessel) from y to $y + \Delta y$ that is occupied by particles. Then $Q(y,t)$ in the 1-D picture corresponds to $P(x,t)$ in the 3-D picture. We then revise the definition of local concentration to

$$c^*(y,t): \mathbf{R}^+ \times [0,\infty) \rightarrow [0,1]$$

$$c^*(y,t) = \int_{\mathbf{R}^+} K^*(y - y')\, dQ(y',t).$$

In this interpretation, $c^*$ is the convolution of Q with $K^*$. Consequently, it may be updated simultaneously throughout the slurry by a single pass of the Fast Fourier Transform algorithm and its inverse. In addition, we used Gaussian weight functions as kernels to further enhance computational speed. By changing the bandwidth of these kernel smoothers, systems that exhibit very different qualitative behavior may result. For example, in our simulations, narrow-bandwidth smoothing often lead to the formation of dense particle layers settling together.

3. Sedimentation is a three-dimensional phenomenon, but the implementation in the Appendix focuses on the vertical component of motion as the most important one. Horizontal motion is modelled by superimposing an independent zero-mean Ornstein-Uhlenbeck processes and particle trajectories are also allowed to overlap so as to indicate a third dimension.

4. Clearly, both space and time need to be discretized in simulations. We chose the space increment $\Delta y$ in such a way that the height of the sedimentation vessel was divided into 100 intervals. The time increment $\Delta t$ was chosen in such a way that on the order of a hundred steps were necessary for the particle to reach the bottom of the vessel.

5. The discretization of time necessitates an assumption that specifies the behavior of particles during the time intervals $(k\Delta t, (k+1)\Delta t)$. The only assumption that

makes sense both stochastically and numerically is the assumption of constant acceleration during these intervals. This amounts to approximation of the integral in

$$X(t + \Delta t) = X(t) + \int_{t}^{\Delta t} V(s)\, ds,$$

by the trapezoidal rule and leads to position being a quadratic spline. Hence, this assumption also implies the use of a quadratic interpolation scheme when determining boundary crossing times.

6. The number of displayed particles during simulations was limited to about 1000. This seemed to be a necessary compromise between realism and computational tractability as restricted by the capabilities of the Apollo DN600.

Further details about these issues and others such as smoothing, plotting, etc. can be found in the FORTRAN code in the appendix.

## APPENDIX

This Section contains (with minimal documentation) the FORTRAN code for an implementation of the stochastic model on an Apollo Domain DN 600 workstation equipped with colour graphics driven by the software package PRIMH. The program TESTIC is due to E. Ramos.

```
      program TESTIC
%nolist
%include '/sys/ins/base.ins.ftn'
%include '/sys/ins/gpr.ins.ftn'
%list
%include '//stat600/stat/ramos/pick/simul/src/colorinfo.ftn'
c       this program simulates particles settling in a slurry
c       using the exact stochastic formula for the increments.
%include '//stat600/stat/ramos/pick/simul/src/graf.com.ftn'
%include '//stat600/stat/ramos/pick/simul/src/graf3.com.ftn'
%include '//stat600/stat/ramos/pick/simul/src/param.com.ftn'
%include '//stat600/stat/ramos/pick/simul/src/param2.com.ftn'
%include '//stat600/stat/ramos/pick/simul/src/spline.com.ftn'

      logical ex,dd
      complex conc(1024),ssv(1024)
      real*8 seed
      real*4 xo(1000),dx(1000),s,co,ss
      real*4 sd,rmu,rmean,c,gsd,z
      real*4 vx(1000),xx(1000),vvx(1000)
      real*4 yx(1000),b
      real*4 sdy,covyv,corryv,corryv1,corryv2,fill,xk
      real*4 yy(1000),v(1000),ov(1000)
      real*4 vv(1000),con(1000),sv(1000)
      real*4 sdx,rmeanx,corrxv,gsdx,rrmu
      real*4 bx,cx,vwidth,fk
      integer*2 scx(1000),scy(1000),oscx(1000),center(2)
      integer*2 ooscx(1000),oscy(1000)
      integer*2 rad
      integer*4 in1(1000),in2(1000)
      integer*4 in3(1000),status,icol
      integer*4 ntime,i,j,m,nd,icol1,icol2,ndead
      integer*4 nstart
      character*1 ans,ans2


      open(unit=2,file='-stdin')  {apollo}
      open(unit=3,file='-stdout')  {apollo}
c   CHECK IF THERE IS A CONTINUING SIMULATION FILE
      inquire(file='//stat600/stat/ramos/pick/simul/data/barf8'
     +  ,exist=ex)
C   IF SO, DO WE WANT TO CONTINUE?
      if(ex)then
          write(3,*)'does this continue a process?'
          read(2,1001)ans
c          read(2,1001)ans2
          ans2=ans
1001      format(a1)
          if(ans.eq.'y')then
             open(unit=9,file=
     +       '//stat600/stat/ramos/pick/simul/data/barf8',recl=128)
             read(9,*)ntime,npart,xscl1,xscl2,yscl,ovlength,vwidth
             read(9,*)t,incon,pack,fk,xk,h,nbins,seed,vlength
             read(9,*)jdims,mdims
             read(9,889)((coef(j,m),m=1,mdims),j=1,jdims)
             read(9,888)(rknots(j),j=1,jdims)
             read(9,*)ban,rad,icol1,icol2,ndraw,nhi
             read(9,1000)(xo(i),yy(i),con(i),oscy(i),
```

```
     +                    ooscx(i),ov(i),v(i),vx(i),i=1,npart)
               close(9)
               nstart=ntime+1
               write(3,*)'enter number of steps to continue'
               read(2,*)ntime
               ntime=nstart+ntime-1
               go to 99
            else
               go to 98
            end if
         end if
c    GET PARAMETERS FOR PROCESS FROM FILES
98       open(unit=4,file='//stat600/stat/ramos/pick/simul/data/barf00')  {apollo}
         read(4,*)ntime,npart,xscl1,xscl2,yscl,ovlength,vwidth
         close(4)
         write(3,*)'This simulation runs for',ntime,' steps with',npart,
         write(3,*)'particles. Scaling is',xscl1,xscl2,' for the x-axis,'
         write(3,*)'and ',yscl,' for the y-axis.'
         write(3,*)
         open(unit=5,file='//stat600/stat/ramos/pick/simul/data/barf1')  {apollo}
         read(5,*)t,incon,pack,fk,xk,h,nbins,seed
         close(5)
         write(3,*)'time step length in sec ',t
         write(3,*)'initial concentration ',incon
         write(3,*)'concentration of packed bed ',pack
         write(3,*)'fill rate constant',fk
         write(3,*)'smoothing half-bandwidth in mm ',h
         write(3,*)'bins in histograms ',nbins
         write(3,*)

         open(unit=6,file='//stat600/stat/ramos/pick/simul/data/barf2')  {apollo}
         read(6,*)jdims,mdims
c                              read the spline coefficients
c                              for more info cf. De Boor,
c                              A practical Guide to splines,
c                              Springer-Verlag

         read(6,889)((coef(j,m),m=1,mdims),j=1,jdims)
         read(6,888)(rknots(j),j=1,jdims)
         close(6)
888      format(g16.8)
889      format(4g16.8)

c                                        get graphics info

         open(unit=7,file='//stat600/stat/ramos/pick/simul/data/barf3')  {apollo}
         read(7,*)ban,rad,icol1,icol2,ndraw,nhi
         close(7)
          ndraw=min(npart,ndraw)
          nhi=min(min(nhi,ndraw),npart)
         write(3,*)'number of highlighted particles ',nhi
         if (rad.lt.0)rad=int2(2*sqrt(incon/.02))
         write(3,*)'we draw ',ndraw,' particles of radius ',rad
         fill=min(incon/pack*fk,1.)
         write(3,890)(fill*100.)
890      format('fill rate is ',f6.2,'%')
c    initialize variables
         nstart=1
         vlength=ovlength
         do 11 i=1,npart
            dx(i)=0.
            xo(i)=0.
11         enddo

c    Get initial vertical position and velocities.
         call gunif(seed,ndraw,xo)
```

```
        call ggnpm(seed,npart,v)
        call gunif(seed,npart,yy)
        call ggnpm(seed,npart,vx)
c   parametrize initial step; first in the y-axis
99        call paramet(1.,rrmu,gsd,c,b,sd)

c   then in the x-axis
          call paramet(1.,rrmu,gsdx,cx,bx,sdx)
          sdx=sdx*xk
          gsdx=gsdx*xk
         corrxv=(1-cx)/sqrt(2*bx*t*(1+cx)/(1-cx)+cx-3)
         sdx=sqrt(2*bx*t+(1-cx)*(cx-3))*sdx/bx

       if(.not.ex.or.ans.ne.'y')then
c         get initial velocities
        do j=1,npart
           con(j)=1.
           v(j)=v(j)*sd+rrmu
c           vx(j)=vx(j)*sdx
           vx(j)=vx(j)*sd
           yy(j)=yy(j)*vlength
           xo(j)=xo(j)*vwidth
        enddo
       end if

c   initialize graphics
         call g_init
         call legend

c   draw initial picture
       if(.not.ex.or.ans.ne.'y')then
        call timer(0,t)
        do j=1,ndraw
           oscx(j)=xscl1+int2(xo(j)*xscl2/vwidth)
           scy(j)=int2(yy(j)*yscl/ovlength)
           center(1)=oscx(j)
           center(2)=scy(j)
           icol=icol1
           if (j.le.ndraw-nhi)icol=icol2
          call gpr_$set_fill_value(icol,status)
          call gpr_$circle_filled(center,rad,status)
           ooscx(j)=oscx(j)
           oscy(j)=scy(j)
        enddo
        end if
c   settling starts
          do j=nstart,ntime
           call ggnpm(seed,npart,vv)
           call ggnpm(seed,npart,yx)
           call ggnpm(seed,npart,vvx)
           call ggnpm(seed,npart,xx)

c   calculations for new view
           ndead=0

           do 20 i=1,npart
c
c    IF LAST PARTICLE POSITION EXCEEDS CURRENT VESSEL LENGTH

              if(yy(i).ge.vlength) then
c    MEANS PARTICLE IS PART OF THE PACKED BED

                 ndead=ndead+1

c    AND ITS CURRENT VELOCITY SHOULD BE SET TO ZERO
```

```
                vv(i)=0.
                vvx(i)=0.

              else

c   parametrization for mean and sd in terms of concentration
C      INSURE POSITIVE CONCENTRATIONS
                con(i)=max(0.,con(i))
                call paramet(con(i),rmu,gsd,c,b,sd)

c   get velocity and new position for particle
c   first the parameters to give two normals a given bivariate normal
c      corryv=(1-c)**2/sqrt((2*b*t+c*(4-c)-3)*(1-c**2))
                corryv=(1-c)/sqrt(2*b*t*(1+c)/(1-c)+c-3)
                sdy=sqrt(2*b*t+(1-c)*(c-3))*sd/b

c   next get the actual variables; first get position increment
c   by first: multiplying N(0,1) variable by appropiate sd,
                yx(i)=sdy*(corryv*vv(i)+sqrt(1-corryv**2)*yx(i))
                xx(i)=sdx*(corrxv*vvx(i)+sqrt(1-corrxv**2)*xx(i))
c                xx(i)=xk*sdy*(corryv*vvx(i)+sqrt(1-corryv**2)*xx(i))
              yy(i)=min(yy(i)+yx(i)+rmu*t+(v(i)-rmu)/b*(1-c),vlength)
c                xo(i)=xo(i)+xx(i)+vx(i)/b*(1-c)

c   then get mean for velocity and then multiply by sd and add mean
c   if position is negative then set equal to -position and set
c   velocity to 0
                rmean=rmu+(v(i)-rmu)*c
                rmeanx=vx(i)*cx
                xo(i)=xo(i)+rmean/rrmu*xx(i)+vx(i)/bx*(1-cx)
c                rmeanx=vx(i)*c
                vv(i)=gsd*vv(i)+rmean
                vvx(i)=gsdx*vvx(i)+rmeanx
              end if
              scx(i)=mod(int2(xo(i)*xscl2/vwidth+11*xscl2),xscl2)+xscl1
              scy(i)=int2(yy(i)*yscl/ovlength)
              ov(i)=v(i)
              v(i)=vv(i)
              vx(i)=vvx(i)
              oscx(i)=scx(i)
20           enddo
            vlength=ovlength*(1.-fill*real(ndead)/real(npart))

c   smoothing
            call kernel2(yy,conc,ov,ssv,in1,in2,in3)
c           call camera_shut
c   erasing and plotting
c                                first the particles
          call erase(1)
          call clip(1)
           do i=1,ndraw
c               center(1)=ooscx(i)
c               center(2)=oscy(i)
c               icol=red
c               if (i.le.ndraw-nhi)icol=black
c              call gpr_$set_fill_value(icol,status)
c              call gpr_$circle_filled(center,rad,status)
              center(1)=scx(i)
              center(2)=scy(i)
              icol=icol1
              if (i.le.ndraw-nhi)icol=icol2
             call gpr_$set_fill_value(icol,status)
             call gpr_$circle_filled(center,rad,status)
              oscy(i)=scy(i)
              ooscx(i)=scx(i)
            enddo
```

```
            call clip(2)
c                               then the curves
            call packed(yellow)
            call erase(0)
            call axes
            call histodraw(conc,ssv,j)
c                               finally the text
            call timer(j,t)
cd          call spielberg
            call pause
            call legend
c     get conc and vel for each particle from smooths
c     notice that for concentration we use the concentration of the
c     bin where the particle is headed for, as given by in3(i), as
c     opposed to the one where the particle currently is, given by
c     in1(i)
c
            do i=1,npart
               con(i)=conc(in1(i))+4*conc(in2(i))+conc(in3(i))
               con(i)=con(i)/6
               sv(i)=ssv(in1(i))
            enddo
          enddo
           call gpr_$terminate(dd,status)
c     SAVE SIMULATION INFO TO CONTINUE SIMULATING LATER
          write(3,*)'do you want to continue later?'
c          open(unit=2,file='-stdin')  {apollo}
          read(2,1001)ans2
          if(ans2.eq.'y')then
          open(unit=8,file='//stat600/stat/ramos/pick/simul/data/barf8'
       +  ,status='overwrite',recl=128)
          write(8,*)ntime,npart,xscl1,xscl2,yscl,ovlength,vwidth
          write(8,*)t,incon,pack,fk,xk,h,nbins,seed,vlength
          write(8,*)jdims,mdims
          write(8,889)((coef(j,m),m=1,mdims),j=1,jdims)
          write(8,888)(rknots(j),j=1,jdims)
          write(8,*)ban,rad,icol1,icol2,ndraw,nhi
          write(8,1000)(xo(i),yy(i),con(i),oscy(i),
       +  ooscx(i),ov(i),v(i),vx(i),i=1,npart)
1000      format(3g16.8,2i4,3g16.8)
          close(8)
          end if
999       stop
          end
          subroutine timer(j,t)
#include '//stat600/stat/ramos/pick/simul/src/colorinfo.ftn'
          integer*4 j,status
          integer*2 xpos,ypos,strl
          integer*2 namel,fid
          real*4 t
          character*8 time
#include '//stat600/stat/ramos/pick/simul/src/graf.com.ftn'
#include '//stat600/stat/ramos/pick/simul/src/graf2.com.ftn'
          write(time,10)(j*t)
10        format(f8.2)
          xpos=xscl1+(28*9)+int2(79)
          ypos=yscl+int2(3)*space-int2(10)
          call gpr_$set_text_value(cyan,status)
          call gpr_$set_text_background_value(black,status)
          call gpr_$move(xpos,ypos,status)
          strl=int2(8)
          call gpr_$text(time,strl,status)
          call gpr_$set_text_background_value(black,status)
          return
          end
```

```
      program simula3
%nolist
%include '/sys/ins/base.ins.ftn'
%include '/sys/ins/gpr.ins.ftn'
%list
c       this program simulates particles settling in a slurry

        double precision seed
        integer*2 scx(10000),scy(10000),oscx(10000)
        integer*2 ooscx(10000),oscy(10000),xscl,yscl
        integer*2 scon(10000),svcon(10000),oscon(10000),osvcon(10000)
        integer nbins,npart,in(10000),mm,ndraw
        real xo(10000),dx(10000),s,h,co,ss,ban,t
        real incon,sd,gsd,rmu,rmean,c,pack
        complex conc(1024),ssv(1024)
        real yy(10000),v(10000),vv(10000),con(10000),sv(10000)
        common /graf/ xscl,yscl
        common /param/ mm,incon,pack,npart,nbins,h,t

c   internal parameters

        data ntime/10000/
        data npart/1000/
        data xscl/500/
        data yscl/1020/
        write(*,*)'This simulation runs for',ntime,' steps with',npart,
        write(*,*)'particles. Scaling is',xscl,' for the x-axis, and'
        write(*,*)yscl,' for the y-axis.'
        write(*,*)

c   user input parameters

        write(*,*)'enter length of time step, rec (.01,5):'
        write(*,*)'enter initial concentration of particles, rec (.04)'
        write(*,*)'enter concentration of packed bed, rec (.6)'
        write(*,*)'enter bandwidth in position histogram, rec (5,100):'
        write(*,*)'enter power of 2 bins in histograms <=1024:'
        write(*,*)'enter scale for position histogram, rec(.3):'
        write(*,*)'enter number of particles to draw'
        write(*,*)'enter seed for random number generator (large):'
        read(*,*)t,incon,pack,h,nbins,ban,ndraw,seed

c   initialize variables

        do 1 i=1,npart
        dx(i)=0.
        xo(i)=0.
1       continue

c   get parameter for packed bed

        mm=nint((500.+4*h)/(500.+8*h)*nbins)
        write(*,*)'vessel ends at the',mm,'th  bin'

c   Get initial vertical position and velocities.

        call gunif(seed,ndraw,xo)
        call ggnpm(seed,npart,v)
        call gunif(seed,npart,yy)

c   parametrize initial step

        call paramet(1.,rmu,gsd,c)
        sd=.6*rmu
```

```
c   get initial velocities

        do 10 j=1,npart
        con(j)=1.
        v(j)=v(j)*sd+rmu
        yy(j)=yy(j)*500.
10      continue

c   initialize graphics

        call g_init

c   draw initial picture

        do 100 j=1,ndraw
        oscx(j)=xscl+int2(xo(j)*xscl)
        scy(j)=int2(yy(j)*yscl/500.)
        call plot(oscx(j),scy(j),1,int4(4))
        ooscx(j)=oscx(j)
        oscy(j)=scy(j)
100     continue

c   settling starts

        do 30 j=1,ntime
        call ggnpm (seed,npart,vv)
        call ggnpm(seed,ndraw,dx)

c   calculations for new view

        do 20 i=1,npart
        if(yy(i).ge.500.)                then
                yy(i)=500.
                vv(i)=0.
                scx(i)=oscx(i)
                scy(i)=yscl

        else

c   parametrization for mean and sd in terms of concentration

        call paramet(con(i),rmu,gsd,c)

c   get velocity and new position for particle

                rmean=rmu+(v(i)-rmu)*c
                vv(i)=gsd*vv(i)+rmean
                yy(i)=abs(yy(i)+(v(i)+vv(i))*(t/2.))
                scx(i)=mod(oscx(i)+int(dx(i)*2*t),xscl)+xscl
                scy(i)=mod(int(yy(i)*yscl/500.),yscl)
        end if
        v(i)=vv(i)
        oscx(i)=scx(i)
20      continue

c   smoothing

        call kernel2(yy,conc,v,ssv,in)

c   get conc and vel for each particle from smooths

        do 400 i=1,npart
        con(i)=conc(in(i))
        sv(i)=ssv(in(i))
400     continue
```

```
c   erasing and plotting

        call axes(ban)
        do 500 i=1,ndraw
        call plot(oscon(i),oscy(i),1,int4(0))
        call plot(osvcon(i),oscy(i),1,int4(0))
        call plot(ooscx(i),oscy(i),1,int4(0))
        if(yy(i).ge.500.)goto 500
        call plot(scx(i),scy(i),1,int4(4))
        ss=amod(con(i)*ban*xscl,real(xscl))
        scon(i)=mod(int(xscl*(.5-sign(.5,ss))+ss),xscl)
        ss=amod(sv(i)*ban*xscl,real(xscl))
        svcon(i)=mod(int(xscl*(.5-sign(.5,ss))+ss),xscl)
        call plot(scon(i),scy(i),1,int4(2))
        call plot(svcon(i),scy(i),1,int4(1))
        oscy(i)=scy(i)
        ooscx(i)=scx(i)
        oscon(i)=scon(i)
        osvcon(i)=svcon(i)
500     continue
30      continue
        stop
        end
         program simulaG
%nolist
%include '/sys/ins/base.ins.ftn'
%include '/sys/ins/gpr.ins.ftn'
%list
c       this program simulates particles settling in a slurry
c       using the exact stochastic formula for the increments.
c       formulas in notes

        real*8 seed
        real*4 sdy,covyv,corryv
        integer*2 scx(10000),scy(10000),oscx(10000)
        integer*2 ooscx(10000),oscy(10000),xscl,yscl
        integer*2 scon(10000),svcon(10000),oscon(10000),osvcon(10000)
        integer*4 nbins,npart,in(10000),mm,ndraw,status
        real*4 xo(10000),dx(10000),s,h,co,ss,ban,t
        real*4 incon,sd,rmu,rmean,c,pack,gsd
        real*4 yx(10000),b
        complex conc(1024),ssv(1024)
        real*4 yy(10000),v(10000),vv(10000),con(10000),sv(10000)
        common /graf/ xscl,yscl,ban
        common /param/ mm,incon,pack,npart,nbins,h,t

c   internal parameters

        data ntime/10000/
        data npart/1000/
        data xscl/500/
        data yscl/1020/
        open(unit=5,file='barf')  {apollo}
        write(*,*)'This simulation runs for',ntime,' steps with',npart,
        write(*,*)'particles. Scaling is',xscl,' for the x-axis, and'
        write(*,*)yscl,' for the y-axis.'
        write(*,*)

c   user input parameters

        write(*,*)'enter length of time step, rec (.01,5):'
        write(*,*)'enter initial concentration of particles, rec (.04)'
        write(*,*)'enter concentration of packed bed, rec (.6)'
        write(*,*)'enter bandwidth in position histogram, rec (5,100):'
        write(*,*)'enter power of 2 bins in histograms <=1024:'
        write(*,*)'enter scale for position histogram, rec(.3):'
```

```
        write(*,*)'enter number of particles to draw'
        write(*,*)'enter seed for random number generator (large):'
        read(5,*)t,incon,pack,h,nbins,ban,ndraw,seed

c   initialize variables

        do 1 i=1,npart
        dx(i)=0.
        xo(i)=0.
1       continue

c   get parameter for packed bed

        mm=nint(500./(500.+8*h)*nbins)
        write(*,*)'vessel ends at the',mm,'th  bin'

c   Get initial vertical position and velocities.

        call gunif(seed,ndraw,xo)
        call ggnpm(seed,npart,v)
        call gunif(seed,npart,yy)

c   parametrize initial step

        call paramet(1.,rmu,gsd,c,b,sd)

c   get initial velocities

        do 10 j=1,npart
        con(j)=1.
        v(j)=v(j)*sd+rmu
        yy(j)=yy(j)*500.
10      continue

c   initialize graphics

        call g_init

c   draw initial picture

        do 100 j=1,ndraw
        oscx(j)=xscl+int2(xo(j)*xscl)
        scy(j)=int2(yy(j)*yscl/500.)
        call plot(oscx(j),scy(j),1,int4(7))
        ooscx(j)=oscx(j)
        oscy(j)=scy(j)
100     continue

c   settling starts

        do 30 j=1,ntime
        call ggnpm (seed,npart,vv)
        call ggnpm(seed,ndraw,dx)
        call ggnpm(seed,npart,yx)

c   calculations for new view

        do 20 i=1,npart

        if(yy(i).ge.500.) then

          yy(i)=500.
          vv(i)=0.
          scx(i)=oscx(i)
          scy(i)=yscl
```

```
        else
c   parametrization for mean and sd in terms of concentration
            call paramet(con(i),rmu,gsd,c,b,sd)
c   get velocity and new position for particle
c   first the parameters to give two normals a given bivariate normal
c           covyv=dble((sd*(1-c))**2/b)
            sdy=sqrt(2*b*t+c*(4-c)-3)*sd/b
c           corryv=covyv/(dble(gsd)*sdy)
            corryv=(1-c)**2/sqrt((2*b*t+c*(4-c)-3)*(1-c**2))
c   next get the actual variables; first get sd for position
            yx(i)=sdy*(corryv*vv(i)*sqrt(1-corryv**2)*yx(i))
c   then add the mean to it
            yy(i)=abs(yy(i)+yx(i)+rmu*t+(v(i)-rmu)/b*(1-c))
c   then get mean for velocity and then multiply by sd and add mean
            rmean=rmu+(v(i)-rmu)*c
            vv(i)=gsd*vv(i)+rmean
            scx(i)=mod(oscx(i)+int(dx(i)*2*t),xscl)+xscl
            scy(i)=mod(int(yy(i)*yscl/500.),yscl)
        end if
        v(i)=vv(i)
        oscx(i)=scx(i)
20      continue
c   smoothing
        call kernel2(yy,conc,v,ssv,in)
c        call histodraw(conc,ssv)
c   get conc and vel for each particle from smooths
        do 400 i=1,npart
        con(i)=conc(in(i))
        sv(i)=ssv(in(i))
400     continue
c   erasing and plotting
        call axes(ban)
        nd=ndraw
        do 500 i=1,ndraw
        call plot(oscon(i),oscy(i),1,int4(0))
        call plot(osvcon(i),oscy(i),1,int4(0))
        call plot(ooscx(i),oscy(i),1,int4(0))
        if(yy(i).ge.500.)yy(i)=500
        call plot(scx(i),scy(i),1,int4(7))
        ss=amod(con(i)*ban*xscl,real(xscl))
        scon(i)=mod(int(xscl*(.5-sign(.5,ss))+ss),xscl)
        ss=amod(sv(i)*ban*xscl,real(xscl))
        svcon(i)=mod(int(xscl*(.5-sign(.5,ss))+ss),xscl)
        call plot(scon(i),scy(i),1,int4(2))
        call plot(svcon(i),scy(i),1,int4(1))
        oscy(i)=scy(i)
        ooscx(i)=scx(i)
        oscon(i)=scon(i)
        osvcon(i)=svcon(i)
```

```
500     continue
30      continue
        stop
        end
         program simulaGn
%nolist
%include '/sys/ins/base.ins.ftn'
%include '/sys/ins/gpr.ins.ftn'
%list
c       this program simulates particles settling in a slurry
c       using the exact stochastic formula for the increments.
c       formulas in notes

        real*8 seed
        real*4 sdy,covyv,corryv,corryv1,corryv2
        integer*2 scx(10000),scy(10000),oscx(10000),center(2)
        integer*2 ooscx(10000),oscy(10000),xscl,yscl
        integer*2 scon(10000),svcon(10000),oscon(10000),osvcon(10000)
        integer*4 nbins,npart,in(10000),in2(10000),mm,ndraw,status
        real*4 xo(10000),dx(10000),s,h,co,ss,ban,t
        real*4 incon,sd,rmu,rmean,c,pack,gsd,z
        real*4 yx(10000),b
        complex conc(1024),ssv(1024)
        real*4 yy(10000),ov(10000),v(10000),vv(10000),con(10000),sv(10000)
        common /graf/ xscl,yscl,ban
        common /param/ mm,incon,pack,npart,nbins,h,t

c   internal parameters

        data ntime/10000/
        data npart/1000/
        data xscl/511/
        data yscl/1023/
        open(unit=5,file='barf')  {apollo}
        write(*,*)'This simulation runs for',ntime,' steps with',npart,
        write(*,*)'particles. Scaling is',xscl,' for the x-axis, and'
        write(*,*)yscl,' for the y-axis.'
        write(*,*)

c   user input parameters

        write(*,*)'enter length of time step, rec (.01,5):'
        write(*,*)'enter initial concentration of particles, rec (.04)'
        write(*,*)'enter concentration of packed bed, rec (.6)'
        write(*,*)'enter bandwidth in position histogram, rec (5,100):'
        write(*,*)'enter power of 2 bins in histograms <=1024:'
        write(*,*)'enter scale for position histogram, rec(.3):'
        write(*,*)'enter number of particles to draw'
        write(*,*)'enter seed for random number generator (large):'
        read(5,*)t,incon,pack,h,nbins,ban,ndraw,seed

c   initialize variables

        do 1 i=1,npart
        dx(i)=0.
        xo(i)=0.
1       continue

c   get parameter for packed bed

        mm=int((500+4*h)/(500+8*h)*nbins)
        write(*,*)'vessel ends at the',mm,'th  bin'

c   Get initial vertical position and velocities.

        call gunif(seed,ndraw,xo)
```

```
        call ggnpm(seed,npart,v)
        call gunif(seed,npart,yy)

c   parametrize initial step

        call paramet(1.,rmu,gsd,c,b,sd)

c   get initial velocities

        do 10 j=1,npart
           con(j)=1.
           v(j)=v(j)*sd+rmu
           yy(j)=yy(j)*500.
10      continue

c   initialize graphics

        call g_init

c   draw initial picture

        do 100 j=1,ndraw
           oscx(j)=xscl+int2(xo(j)*xscl)
           scy(j)=int2(yy(j)*yscl/500.)
         center(1)=oscx(i)
         center(2)=scy(i)
         call gpr_$set_fill_value(2,status)
         call gpr_$circle_filled(center,int2(2),status)
c           call plot(oscx(j),scy(j),1,int4(7))
           ooscx(j)=oscx(j)
           oscy(j)=scy(j)
100     continue

c   settling starts

        do 30 j=1,ntime
        call ggnpm(seed,npart,vv)
        call ggnpm(seed,ndraw,dx)
        call ggnpm(seed,npart,yx)

c   calculations for new view

        do 20 i=1,npart

        if(yy(i).ge.500.) then

          yy(i)=500.
          vv(i)=0.
          scx(i)=oscx(i)
          scy(i)=yscl

        else

c   parametrization for mean and sd in terms of concentration

          call paramet(con(i),rmu,gsd,c,b,sd)

c   get velocity and new position for particle
c   first the parameters to give two normals a given bivariate normal

c          covyv=dble((sd*(1-c))**2/b)
          sdy=sqrt(2*b*t+c*(4-c)-3)*sd/b
c         corryv2=covyv/(dble(gsd)*sdy)
          if(c.gt..9)then
             z=1-c
             sumy=0
```

```
              do 1001 j=15,3,-1
1001          sumy=1./j+sumy*z
              corryv=sqrt(2*sumy*(2-z))
              sdy=sqrt(sumy*z**3*2)*sd/b
            else
              corryv=(1-c)/sqrt((2*b*t+c*(4-c)-3)*(1+c)/(1-c))
            end if
c     next get the actual variables; first get position increment
c     by first: multiplying N(0,1) variable by appropiate sd,

            yx(i)=sdy*(corryv*vv(i)+sqrt(1-corryv**2)*yx(i))

c     then add the mean to it

            yy(i)=yy(i)+yx(i)+rmu*t+(v(i)-rmu)/b*(1-c)

c     then get mean for velocity and then multiply by sd and add mean
c     if position is negative then set equal to -position and set
c     velocity to 0

c            if(yy(i).lt.0)then
c               yy(i)=-yy(i)
c               vv(i)=0.
c            else
               rmean=rmu+(v(i)-rmu)*c
               vv(i)=gsd*vv(i)+rmean
c            end if

            scx(i)=mod(oscx(i)+int(dx(i)*2*t),xscl)+xscl
            scy(i)=int2(yy(i)*yscl/500.)

          end if
          ov(i)=v(i)
          v(i)=vv(i)
          oscx(i)=scx(i)
20        continue

c     smoothing

          call kernel2(yy,conc,ov,ssv,in,in2)
          call erase
          call axes
          call histodraw(conc,ssv)
c     get conc and vel for each particle from smooths

          do 400 i=1,npart
          con(i)=conc(in(i))
          sv(i)=ssv(in2(i))
400       continue

c     erasing and plotting

          nd=ndraw
          do 500 i=1,ndraw
           center(1)=ooscx(i)
           center(2)=oscy(i)
           call gpr_$set_fill_value(0,status)
           call gpr_$circle_filled(center,int2(2),status)
c          call plot(ooscx(i),oscy(i),1,int4(0))
           if(yy(i).ge.500.)yy(i)=499.
           center(1)=scx(i)
           center(2)=scy(i)
           call gpr_$set_fill_value(2,status)
           call gpr_$circle_filled(center,int2(2),status)
c          call plot(scx(i),scy(i),1,int4(7))
          oscy(i)=scy(i)
```

```
        ooscx(i)=scx(i)
500     continue
30      continue
        stop
        end
         program simulaGn
%nolist
%include '/sys/ins/base.ins.ftn'
%include '/sys/ins/gpr.ins.ftn'
%list
c       this program simulates particles settling in a slurry
c       using the exact stochastic formula for the increments.
c       formulas in notes

        real*8 seed
        real*4 sdy,covyv,corryv,corryv1,corryv2
        integer*2 scx(10000),scy(10000),oscx(10000),center(2)
        integer*2 ooscx(10000),oscy(10000),xscl,yscl,rad
        integer*2 scon(10000),svcon(10000),oscon(10000),osvcon(10000)
        integer*4 nbins,npart,in(10000),in2(10000),mm,ndraw,status
        integer*4 icol
        real*4 xo(10000),dx(10000),s,h,co,ss,ban,t
        real*4 incon,sd,rmu,rmean,c,pack,gsd,z
        real*4 yx(10000),b,coef(20,4),rknots(20)
        complex conc(1024),ssv(1024)
        real*4 yy(10000),v(10000),ov(10000)
        real*4 vv(10000),con(10000),sv(10000)
        common /graf/ xscl,yscl,ban
        common /param/ mm,incon,pack,npart,nbins,h,t,rrmu
        common /param2/vlength
        common /spline/jdims,mdims,rknots,coef
c   internal parameters

        data ntime/10000/
        data npart/1000/
        data xscl/511/
        data yscl/780/
        data vlength/500./

        open(unit=6,file='barf2')  {apollo}
        read(6,*)jdims,mdims
c                               read the spline coefficients
c                               for more info cf. De Boor,
c                               A practical Guide to splines,
c                               Springer-Verlag
        read(6,889)((coef(j,m),m=1,mdims),j=1,jdims)
        read(6,888)(rknots(j),j=1,jdims)
888     format(g16.8)
889     format(4g16.8)
        write(*,*)'This simulation runs for',ntime,' steps with',npart,
        write(*,*)'particles. Scaling is',xscl,' for the x-axis, and'
        write(*,*)yscl,' for the y-axis.'
        write(*,*)

c   user input parameters

        write(*,*)'enter length of time step, rec (.01,5):'
        write(*,*)'enter initial concentration of particles, rec (.04)'
        write(*,*)'enter concentration of packed bed, rec (.6)'
        write(*,*)'enter bandwidth in position histogram, rec (5,100):'
        write(*,*)'enter power of 2 bins in histograms <=1024:'
        write(*,*)'enter scale for position histogram, rec(.3):'
        write(*,*)'enter number of particles to draw'
        write(*,*)'enter seed for random number generator (large):'
        write(*,*)'enter radius of particles in pixels'
        open(unit=5,file='barf')  {apollo}
```

```
      read(5,*)t,incon,pack,h,nbins,ban,ndraw,seed,rad

c     initialize variables

      do 1 i=1,npart
      dx(i)=0.
      xo(i)=0.
1     continue

c     get parameter for packed bed
      smm=vlength/(vlength+4*h)*nbins
      mm=int(smm)
      if(smm-mm .gt.0.)then
         mm=mm+1
      end if
      write(*,*)'vessel ends at the',mm,'th  bin'

c     Get initial vertical position and velocities.

      call gunif(seed,ndraw,xo)
      call ggnpm(seed,npart,v)
      call gunif(seed,npart,yy)

c     parametrize initial step

      call paramet(1.,rrmu,gsd,c,b,sd)

c     get initial velocities

      do 10 j=1,npart
         con(j)=1.
         v(j)=v(j)*sd+rrmu
         yy(j)=yy(j)*vlength
10    continue

c     initialize graphics

d      call g_init

c     draw initial picture

      do 100 j=1,ndraw
         oscx(j)=xscl+int2(xo(j)*xscl)
         scy(j)=int2(yy(j)*yscl/vlength)
       center(1)=oscx(j)
       center(2)=scy(j)
d       call gpr_$set_fill_value(3,status)
d       call gpr_$circle_filled(center,rad,status)
        ooscx(j)=oscx(j)
        oscy(j)=scy(j)
100   continue

c     settling starts

      do 30 j=1,ntime
      call ggnpm(seed,npart,vv)
      call ggnpm(seed,ndraw,dx)
      call ggnpm(seed,npart,yy)

c     calculations for new view

      do 20 i=1,npart

      if(yy(i).ge.vlength) then

        yy(i)=vlength
```

```
            vv(i)=0.
            dx(i)=0.
            scx(i)=oscx(i)
            scy(i)=yscl

          else

c     parametrization for mean and sd in terms of concentration

            call paramet(con(i),rmu,gsd,c,b,sd)

c     get velocity and new position for particle
c     first the parameters to give two normals a given bivariate normal

              corryv=(1-c)/sqrt((2*b*t+c*(4-c)-3)*(1+c)/(1-c))
              sdy=sqrt(2*b*t+(1-c)*(c-3))*sd/b
                 corryv=corryv/abs(corryv)*min(1.,abs(corryv))
c     next get the actual variables; first get position increment
c     by first: multiplying N(0,1) variable by appropiate sd,

               yx(i)=sdy*(corryv*vv(i)+sqrt(1-corryv**2)*yx(i))
               yy(i)=yy(i)+yx(i)+rmu*t+(v(i)-rmu)/b*(1-c)

c     then get mean for velocity and then multiply by sd and add mean
c     if position is negative then set equal to -position and set
c     velocity to 0

               rmean=rmu+(v(i)-rmu)*c
               vv(i)=gsd*vv(i)+rmean

            scx(i)=mod(oscx(i)+int(dx(i)*t*
     +              (1.-incon*con(i)/pack)),xscl)+xscl
            scy(i)=int2(yy(i)*yscl/vlength)

          end if
          ov(i)=v(i)
          v(i)=vv(i)
          oscx(i)=scx(i)
20        continue

c     smoothing

          call kernel2(yy,conc,ov,ssv,in,in2)
d          call erase(0)
d          call axes
d           call histodraw(conc,ssv)
cd          call pause
c     get conc and vel for each particle from smooths

          do 400 i=1,npart
          con(i)=conc(in(i))
          sv(i)=ssv(in2(i))
c          sv(i)=ssv(in(i))
400       continue

c     erasing and plotting

          nd=ndraw
          do 500 i=1,ndraw
           center(1)=ooscx(i)
           center(2)=oscy(i)
d           call gpr_$set_fill_value(1,status)
d           call gpr_$circle_filled(center,rad,status)
           if(yy(i).ge.vlength)yy(i)=vlength
           center(1)=scx(i)
           center(2)=scy(i)
```

```
          icol=3
          if (i.gt.10)icol=5
d         call gpr_$set_fill_value(icol,status)
d         call gpr_$circle_filled(center,rad,status)
        oscy(i)=scy(i)
        ooscx(i)=scx(i)
500     continue
30      continue
        stop
        end
        integer*4 jdims,mdims
        real*4 rknots(20),coef(20,4)
        common /spline/jdims,mdims,rknots,coef
         program simulaGn
%nolist
%include '/sys/ins/base.ins.ftn'
%include '/sys/ins/gpr.ins.ftn'
%include '//stat600/stat/ramos/pick/simul/src/colorinfo.ftn'
%list
c       this program simulates particles settling in a slurry
c       using the exact stochastic formula for the increments.

        real*8 seed
        real*4 sdy,covyv,corryv,corryv1,corryv2,fill
        integer*2 scx(10000),scy(10000),oscx(10000),center(2)
        integer*2 ooscx(10000),oscy(10000),xscl1,xscl2
        integer*2 yscl,yyscl,rad
        integer*2 scon(10000),svcon(10000),oscon(10000),osvcon(10000)
        integer*4 nbins,npart,in(10000),in2(10000),mm,ndraw,status,icol
        real*4 xo(10000),dx(10000),s,h,co,ss,ban,t
        real*4 incon,sd,rmu,rmean,c,pack,gsd,z
        real*4 yx(10000),b,coef(20,4),rknots(20)
        real*4 vx(10000),xx(10000),vvx(10000)
        complex conc(1024),ssv(1024)
        real*4 yy(10000),v(10000),ov(10000)
        real*4 vv(10000),con(10000),sv(10000)
        integer*4 jdims,mdims,ntime,nhi,i,j,m,nd,icol1,icol2,ndead
        real*4 sdx,ovlength,rmeanx,corrxv,gsdx,romu,rrmu
        real*4 bx,cx,vlength,vwidth,fk,rrmux
        common /graf/ xscl1,xscl2,yscl,ban
        common /param/mm,incon,pack,npart,nbins,h,t,romu
        common /param2/ovlength,vlength
        common /spline/jdims,mdims,rknots,coef

c   GET PARAMETERS FOR PROCESS FROM FILES
        open(unit=4,file='//stat600/stat/ramos/pick/simul/data/barf0')  {apoll
        read(4,*)ntime,npart,xscl1,xscl2,yscl,ovlength,vwidth

        write(*,*)'This simulation runs for',ntime,' steps with',npart,
        write(*,*)'particles. Scaling is',xscl1,xscl2,' for the x-axis,'
        write(*,*)'and ',yscl,' for the y-axis.'
        write(*,*)

c        write(*,*)'enter length of time step, rec (.01,5):'
c        write(*,*)'enter initial concentration of particles, rec (.04)'
c        write(*,*)'enter concentration of packed bed, rec (.6)'
c        write(*,*)'enter fill rate constant'
c        write(*,*)'enter tuning constant for lateral movement'
c        write(*,*)'enter bandwidth in position histogram, rec (5,100):'
c        write(*,*)'enter power of 2 bins in histograms <=1024:'
c        write(*,*)'enter seed for random number generator (large):'
        open(unit=5,file='//stat600/stat/ramos/pick/simul/data/barf1')  {apoll
        read(5,*)t,incon,pack,fk,xk,h,nbins,seed
        close(5)
        write(*,*)'time step length in sec ',t
        write(*,*)'initial concentration ',incon
```

```
      write(*,*)'concentration of packed bed ',pack
      write(*,*)'fill rate constant',fk
      write(*,*)'smoothing half-bandwidth in mm ',ban
      write(*,*)'bins in histograms ',nbins

      open(unit=6,file='//stat600/stat/ramos/pick/simul/data/barf2')  {apollo}
      read(6,*)jdims,mdims
c                              read the spline coefficients
c                              for more info cf. De Boor,
c                              A practical Guide to splines,
c                              Springer-Verlag

      read(6,889)((coef(j,m),m=1,mdims),j=1,jdims)
      read(6,888)(rknots(j),j=1,jdims)
      close(6)
888   format(g16.8)
889   format(4g16.8)

c                                        get graphics info

      open(unit=7,file='//stat600/stat/ramos/pick/simul/data/barf3')  {apollo}
c      write(*,*)'enter scale for position histogram, rec(.3):'
c      write(*,*)'enter radius of particles in pixels'
c      write(*,*)'enter colors (2) of particles'
c      write(*,*)'enter number of particles to highlight'
      read(7,*)ban,rad,icol1,icol2,nhi
      close(7)
c      write(*,*)'radius of particles in pixels ',rad
      write(*,*)'number of highlighted particles ',nhi
      ndraw=npart
      rad=int2(2*sqrt(incon/.02))
      write(*,*)'we draw ',ndraw,' particles of radius ',rad
      fill=min(incon/pack*fk,1.)
      write(*,*)'fill rate is ',fill

c   initialize variables
      vlength=ovlength
      do i=1,npart
         dx(i)=0.
         xo(i)=0.
      enddo

c   Get initial vertical position and velocities.
      call gunif(seed,ndraw,xo)
      call ggnpm(seed,npart,v)
      call gunif(seed,npart,yy)

c   parametrize initial step; first in the y-axis
      call paramet(0.,romu,gsd,c,b,sd)
      call paramet(1.,rrmu,gsd,c,b,sd)

c   then in the x-axis
c      open(unit=8,file='//stat600/stat/ramos/pick/simul/data/barf4')  {apol
c      write(*,*)'enter b for lateral movement [.05,.2]'
c      write(*,*)'enter c for sdx=c/t'
c      read(8,*)bx,cx
c      close(8)
c      sdx=cx/t
c      cx=exp(max(-30.,-bx*t))
c      gsdx=sqrt(1-cx**2)*sdx
c       call paramet(1.,rrmux,gsdx,cx,bx,sdx)
c       sdx=sdx*xk
c       gsdx=gsdx*xk
c      corrxv=(1-cx)/sqrt((2*bx*t+cx*(4-cx)-3)*(1+cx)/(1-cx))
c      sdx=sqrt(2*bx*t+(1-cx)*(cx-3))*sdx/bx
c       write(*,*)'correlation and sd for lateral movement'
```

```
c         write(*,*)corrxv,sdx

c   get initial velocities
        do j=1,npart
           con(j)=1.
           v(j)=v(j)*sd+rrmu
c           vx(j)=vx(j)*sdx
           vx(j)=vx(j)*sd
           yy(j)=yy(j)*vlength
           xo(j)=xo(j)*vwidth
        enddo

c   initialize graphics
d        call g_init

c   draw initial picture
        call legend
        call timer(0,t)
        do j=1,ndraw
           oscx(j)=xscl1+int2(xo(j)*xscl2/vwidth)
           scy(j)=int2(yy(j)*yscl/ovlength)
         center(1)=oscx(j)
         center(2)=scy(j)
          icol=icol1
          if (j.gt.nhi)icol=icol2
d         call gpr_$set_fill_value(icol,status)
d         call gpr_$circle_filled(center,rad,status)
          ooscx(j)=oscx(j)
          oscy(j)=scy(j)
        enddo

c   settling starts
        do j=1,ntime
           call ggnpm(seed,npart,vv)
           call ggnpm(seed,ndraw,yx)
           call ggnpm(seed,npart,vvx)
           call ggnpm(seed,npart,xx)

c   calculations for new view
           ndead=0

           do 20 i=1,npart
c
c    IF LAST PARTICLE POSITION EXCEEDS CURRENT VESSEL LENGTH

              if(yy(i).ge.vlength) then
C    MEANS PARTICLE IS PART OF THE PACKED BED

                 ndead=ndead+1

C    AND ITS CURRENT VELOCITY SHOULD BE SET TO ZERO

                 vv(i)=0.
                 vvx(i)=0.

              else

c   parametrization for mean and sd in terms of concentration
C      INSURE POSITIVE CONCENTRATIONS
                 con(i)=max(0.,con(i))
                 call paramet(con(i),rmu,gsd,c,b,sd)

c   get velocity and new position for particle
c   first the parameters to give two normals a given bivariate normal
c                  corryv=(1-c)**2/sqrt((2*b*t+c*(4-c)-3)*(1-c**2))
                 corryv=(1-c)/sqrt(2*b*t*(1+c)/(1-c)+c-3)
```

```
                sdy=sqrt(2*b*t+(1-c)*(c-3))*sd/b

c   next get the actual variables; first get position increment
c   by first: multiplying N(0,1) variable by appropiate sd,
                yx(i)=sdy*(corryv*vv(i)+sqrt(1-corryv**2)*yx(i))
c                xx(i)=sdx*(corrxv*vvx(i)+sqrt(1-corrxv**2)*xx(i))
                xx(i)=xk*sdy*(corryv*vvx(i)+sqrt(1-corryv**2)*xx(i))
              yy(i)=min(yy(i)+yx(i)+rmu*t+(v(i)-rmu)/b*(1-c),vlength)
c                xo(i)=xo(i)+xx(i)+vx(i)/bx*(1-cx)
                xo(i)=xo(i)+xx(i)+vx(i)/b*(1-c)

c   then get mean for velocity and then multiply by sd and add mean
c   if position is negative then set equal to -position and set
c   velocity to 0
                rmean=rmu+(v(i)-rmu)*c
c                rmeanx=vx(i)*cx
                rmeanx=vx(i)*c
                vv(i)=gsd*vv(i)+rmean
                vvx(i)=gsdx*vvx(i)+rmeanx
             end if
             scx(i)=mod(int2(xo(i)*xscl2/vwidth+11*xscl2),xscl2)+xscl1
             scy(i)=int2(yy(i)*yscl/ovlength)
             ov(i)=v(i)
             v(i)=vv(i)
             vx(i)=vvx(i)
             oscx(i)=scx(i)
20          enddo
          vlength=ovlength*(1.-fill*real(ndead)/real(npart))

c   smoothing
          call kernel2(yy,conc,ov,ssv,in,in2)

c   erasing and plotting
c                              first the particles
c         call erase(1)
          nd=ndraw
          do i=1,ndraw
             center(1)=ooscx(i)
             center(2)=oscy(i)
             icol=red
             if (i.le.npart-nhi)icol=black
d            call gpr_$set_fill_value(icol,status)
d            call gpr_$circle_filled(center,rad,status)
             center(1)=scx(i)
             center(2)=scy(i)
             icol=icol1
             if (i.le.npart-nhi)icol=icol2
d            call gpr_$set_fill_value(icol,status)
d            call gpr_$circle_filled(center,rad,status)
             oscy(i)=scy(i)
             ooscx(i)=scx(i)
          enddo
c                              then the curves
          call packed(yellow,ovlength,vlength)
d         call erase(0)
d         call axes
d         call histodraw(conc,ssv)
c                              finally the text
          call timer(j,t)
cd        call pause

c   get conc and vel for each particle from smooths
          do i=1,npart
             con(i)=conc(in(i))
             sv(i)=ssv(in2(i))
c            sv(i)=ssv(in(i))
```

```
          enddo
        enddo
999     stop
        end

        real function asinh (x)
        real*4 x
        x=max(x,10.**(-16))
        asinh=log(x+sqrt(1.+x**2.))
        return
        end
        subroutine axes
%include '//stat600/stat/ramos/pick/simul/src/colorinfo.ftn'
        integer*2 x(2),y(2)
        integer*4 m,col,status,j
%include '//stat600/stat/ramos/pick/simul/src/graf.com.ftn'
        m=int(1./ban)
        y(1)=0
        y(2)=yscl
        do 20 j=0,m
        col=red
        x(1)=int2(real(j)*xscl1/m)
        x(2)=x(1)
        if(j.eq.m/2.or.j.eq.0)col=white
        call line(x,y,int2(2),col,status)
20      continue
        return
        end

        subroutine clip(j)
c    sets clipping window to the left or right or full bitmap
c    (according to j=0 or j=1 or j=2 )
        integer*4 j,status
        integer*2 window(2,2)
%include '//stat600/stat/ramos/pick/simul/src/graf.com.ftn'
        window(1,1)=int2(0)
        if(j.eq.1)window(1,1)=xscl1
        window(2,1)=int2(0)
        if(j.eq.0)window(1,2)=xscl1-int2(1)
        if(j.eq.1)window(1,2)=xscl2-int2(1)
        if(j.eq.2)window(1,2)=xscl1+xscl2-int2(1)
        window(2,2)=yscl-int2(1)
        if(j.eq.2)window(2,2)=int2(1023)
        call gpr_$set_clip_window(window,status)
        call gpr_$set_clipping_active((.true.),status)
        return
        end
       integer*4
     +             black,
     +             red,
     +             green,
     +             blue,
     +             cyan,
     +             magenta,
     +             yellow,
     +             white
       parameter(
     +             black=0,
     +             red=1,
     +             green=2,
     +             blue=3,
     +             cyan=4,
     +             magenta=6,
     +             yellow=5,
     +             white=7    )
        subroutine erase(k)
```

```
%include '//stat600/stat/ramos/pick/simul/src/graf.com.ftn'
        integer*2 x(4),y(4)
        integer*4 n,k

        x(1)=xscl2*int2(k)+xscl1-int2(1)
        x(2)=x(1)
        x(3)=xscl1*int2(k)-int2(1)
        x(4)=x(3)
c
        y(1)=int2(0)
        y(2)=yscl-int2(1)
        y(3)=y(2)
        y(4)=int2(0)

        call fill(x,y,4,0)
        return
        end
C*****************************************************************
c    SUBROUTINE TO fill POLYGON
C
C        INPUTS: X,Y -- COORDINATES IN PIXEL VALUE
C                  N -- NUMBER OF SIDES OF POLYGON
C                COL -- COLOR TO USE IN FILLING
C
C
C*****************************************************************
        subroutine fill(x,y,n,col)
C
        integer*2 x(1),y(1)
        integer*4 st,col,n
        call gpr_$set_fill_value(col,st)
        call gpr_$start_pgon(x(n),y(n),st)
        call gpr_$pgon_polyline(x,y,int2(n),st)
        call gpr_$close_fill_pgon(st)
        return
        end
       subroutine g_init
C
C Allocate and initialize bitmap for graphics in borrowed mode
C
        integer*4 desc,bl_desc,status
        integer*2 mode,hi,unit,bsize(2)
        integer*2 window(2,2)
        data mode/0/
        data hi/3/
        data unit/1/
        data bsize/1024,1024/
        window(1,1)=int2(0)
        window(2,1)=int2(0)
        window(1,2)=int2(1023)
        window(2,2)=int2(1023)
        call gpr_$init(mode,unit,bsize,hi,desc,status)
        call gpr_$set_bitmap(desc,status)
        call gpr_$allocate_attribute_block(bl_desc,status)
        call gpr_$set_attribute_block(bl_desc,status)
        call gpr_$set_clip_window(window,status)
        call gpr_$set_clipping_active((.true.),status)
        return
        end

        integer*2 xscl1,xscl2,yscl
        real*4 ban
        common /graf/ xscl1,xscl2,yscl,ban
        integer*2 space
        common /graf2/ space
        integer*4 nhi,ndraw
```

```
      common /graf3/nhi,ndraw

C-----------------------------------------------------------------------
C
      SUBROUTINE GUNIF(DSEED,NR,R)
C                                     SPECIFICATIONS FOR ARGUMENTS
      INTEGER             NR
      REAL                R(NR)
      DOUBLE PRECISION    DSEED
C                                     SPECIFICATIONS FOR LOCAL VARIABLES
      INTEGER             I
      REAL                U
      DOUBLE PRECISION    D2P31M,D2PN31
C                                     D2P31M = (2**31)-1
C                                     D2PN31 = (2**31)
      DATA                D2P31M/2147483647.D0/
      DATA                D2PN31/2147483711.D0/
C                                     FIRST EXECUTABLE STATEMENT
      DO 10 I = 1,NR
         DSEED = DMOD(16807.D0*DSEED,D2P31M)
         R(I) = real(DSEED/D2PN31)
   10 CONTINUE
      RETURN
      END

        SUBROUTINE PAUSE
c   This subroutine holds the screen in borrowed mode so that
c   the plot does not immediately disappear.
c   The screen is cleared after any key is struck, with a printable
c   screen dump produced if the key is 'F1'
%no list
%include '/sys/ins/gpr.ins.ftn'
%list
        CHARACTER*1 e_d
        INTEGER*4 status
        INTEGER*2 pos(2),e_t,key_set(16),window(2,2)
        LOGICAL dummy

        data key_set/16*16#ffff/
        window(1,1) = 0
        window(2,1) = 0
        window(1,2) = 1023
        window(2,2) = 1023

        call GPR_$ENABLE_INPUT( GPR_$KEYSTROKE,key_set,status )
        dummy = GPR_$EVENT_WAIT( e_t,e_d,pos,status )

c If the key 'F1' is struck then a screen dump is made:
        if( e_d .eq. char(16#C0) ) then
        call savpic('//stat600/stat/ramos/tmp/scrdmp',
     +              int2(31),window)
        end if

        call GPR_$CLEAR( int4(0),status )
10      continue
        dummy = GPR_$COND_EVENT_WAIT( e_t,e_d,pos,status )
        if( e_t.ne.GPR_$NO_EVENT ) go to 10
        return
        end


        SUBROUTINE SAVPIC( filename,length,window )
c   This subroutine dumps the bitmap onto a file which can be
c   printed using  PRF ... -PLOT.
```

```
        CHARACTER*(*) filename
        INTEGER*2 window(2,2),length
        INTEGER*2 i,plane_0,hiplane,bit_or,str_id,line_wid,bpi,rop(8)
        INTEGER*2 gmfstatus
        INTEGER*4 status,bmdesc,pointer
        LOGICAL invert
        PARAMETER( invert   = .true.,
     +             bpi      = 100,
     +             plane_0  = 0,
     +             bit_or   = 7,
     +             hiplane  = 3,
     +             gmfstatus = 1 )

        call GPR_$INQ_BITMAP( bmdesc,status )
        call GPR_$INQ_RASTER_OPS( rop,status )

c   Copy all the planes into plane zero:

        call GPR_$SET_RASTER_OP( plane_0,bit_or,status )
        do i=1,hiplane
            call GPR_$BIT_BLT
     +      ( bmdesc,window,i,window(1,1),plane_0,status )
        end do

c   Dump the bitmap onto a file:
            call SHOW('4')
        call GPR_$INQ_BITMAP_POINTER( bmdesc,pointer,line_wid,status )
        call GMF_$OPEN( filename,length,gmfstatus,str_id,status )
        call GMF_$COPY_PLANE( str_id,invert,bpi,pointer,window(1,2),
     +      window(2,2),line_wid,status )
        call GMF_$CLOSE( str_id,status )

        call GPR_$SET_RASTER_OP( plane_0,rop(1),status )

        return
        end


        SUBROUTINE SHOW( ch )
%nolist
%include '/sys/ins/cal.ins.ftn'
%include '/sys/ins/time.ins.ftn'
%list
        character ch*1
        integer*2 clock(3),ixp,iyp
        integer*4 seconds,status
        data ixp,iyp,seconds/1000,20,1/
        call CAL_$SEC_TO_CLOCK( seconds,clock )
        call TIME_$WAIT( TIME_$RELATIVE,clock,status )
        return
        end
        subroutine pause
%no list
%include '/sys/ins/base.ins.ftn'
%include '/sys/ins/error.ins.ftn'
%include '/sys/ins/gpr.ins.ftn'
%list
c     This subroutine holds the screen in borrowed mode so that
c the plot does not immediately disappear.
c
        character*1 e_d
        integer*4 status
        integer*2 position(2),e_t,key_set(16),rectangle(4)
        logical unobscured
c
        data key_set/16*16#ffff/
```

```
        rectangle(1) = 0
        rectangle(2) = 0
        rectangle(3) = 1023
        rectangle(4) = 1023
c
        call gpr_$enable_input(gpr_$keystroke,key_set,status)
        unobscured = gpr_$event_wait(event_type,e_d,position,status)

c   If the key 'F1' is struck then a screen dump is made:
        if (e_d .eq. char(16#C0)) then
        call savpic('//stat600/stat/ramos/pick/simul/pic/scrdmp',
     +              int2(42),rectangle,status)
        end if

c        CALL GPR_$CLEAR(INT4(0),STATUS)
        return
        end


        subroutine savpic(filename,length,rectangle,status)
c
c       This subroutine dumps the bitmap onto a file which can be
c       printed using  PRF ... -PLOT.
c
%no list
%include '/sys/ins/base.ins.ftn'
%include '/sys/ins/gmf.ins.ftn'
%include '/sys/ins/gpr.ins.ftn'
%include '/sys/ins/error.ins.ftn'
%list
c
c
        character*(*) filename
        integer*2 rectangle(4),wpl,length,str_id,line_wid,bpi
        integer*2 i,plane_0,bit_or,hiplane
        integer*4 status,bmdesc,pointer
        logical invert
        parameter (invert   = .true.,
     +             bpi      = 100,
     +             plane_0  = 0,
     +             bit_or   = 7,
     +             hiplane  = 3 )

        call gpr_$inq_bitmap(bmdesc,status)

c   Copy all the planes into plane zero:

        call GPR_$SET_RASTER_OP( plane_0, bit_or, status )
        do i=1,hiplane
            call GPR_$BIT_BLT
     +      ( bmdesc,rectangle,i,rectangle,plane_0,status )
        end do

c   Dump the bitmap onto a file:

        call gpr_$inq_bitmap_pointer( bmdesc,pointer,line_wid,status )
c        wpl = (rectangle(3)+15)/16
        wpl = line_wid
        call gmf_$open( filename,length,gmf_$overwrite,str_id,status )
        call gmf_$copy_subplane( str_id,invert,bpi,pointer,rectangle(3),
     +      rectangle(4),rectangle(1),rectangle(2),wpl,status )
        call gmf_$close( str_id,status )

        return
        end
        subroutine histodraw(conc,smv,kk)
```

```
%include '//stat600/stat/ramos/pick/simul/src/colorinfo.ftn'
        integer*2 ycoord(1024),conarray(1024),velarray(1024)
        integer*2 fluxarray(1024)
        integer*4 status,jj,jk,j,kk,col1,col2,col3
        real*4 y,origin,half
        complex conc(1),smv(1)
%include '//stat600/stat/ramos/pick/simul/src/graf.com.ftn'
%include '//stat600/stat/ramos/pick/simul/src/param.com.ftn'
%include '//stat600/stat/ramos/pick/simul/src/param2.com.ftn'
c                               EXE STATR
        half=(vlength+8*h)/nbins/2
        origin=-4*h+half
c                                              ycoord(1)=int2(-4*h/vlength*yscl)
c                                              conarray(1)=int2(0)
        do 10 j=1,nbins

c                                              jj=j/2+mod(j,2)
c                                              jk=(j+1)/2-mod(j,2)
          conarray(j)=int2(min(real(conc(j))*ban*xscl1,xscl1))
          velarray(j)=int2(min(real(smv(j)+5)*ban*xscl1,xscl1))
      fluxarray(j)=int2(min(real(conc(j)*smv(j))*ban*xscl1,xscl1))
          y=(origin+(j-1)*2*half)/ovlength*yscl
c                                               y=(-4*h+(jj-1)*(vlength+8*h)/nbins)/c
          ycoord(j)=int2(y)
10      continue
c                                               velarray(1)=velarray(2)
c                                                fluxarray(1)=fluxarray(2)
         call clip(0)
c         col1=mod(kk,7)+1
c         col2=mod(kk*8+1,7)+1
c         col3=mod(kk*64+2,7)+1
         call line (conarray,ycoord,int2(nbins),blue,status)
         call line (velarray,ycoord,int2(nbins),magenta,status)
         call line (fluxarray,ycoord,int2(nbins),yellow,status)
          call clip(2)
        return
        end
        subroutine interv(xt.lxt,x,left,mflag)
c
cREFERENCE: DE BOOR, CARL. A PRACTICAL GUIDE TO
C           SPLINES. SPRINGER-VERLAG, 1978, PP. 89-93
C
         INTEGER LEFT, LXT,MFLAG,IHI,ILO,ISTEP,MIDDLE
         REAL X ,XT(LXT)
         DATA ILO /1/
         SAVE ILO
         IHI=ILO+1
         IF (IHI.LT.LXT)GOTO 20
         IF (X.GE.XT(LXT))GOTO 110
         IF (LXT.LE.1)GOTO 90
         ILO=LXT-1
20       IF (X.GE.XT(IHI))GOTO 40
         IF (X.GE.XT(ILO))GOTO 100
         ISTEP=1
31       IHI=ILO
         ILO=IHI-ISTEP
         IF(ILO.LE.1)GOTO 35
         IF(X.GE.XT(ILO))GOTO50
         ISTEP=ISTEP+2
35       ILO=1
         IF(X.LT.XT(1))GOTO 90
         GOTO 50
40       ISTEP=1
41       ILO=IHI
         IHI=ILO+ISTEP
         IF(IHI.GE.LXT)          GOTO 45
```

```
        IF(X.LT.XT(IHI))          GOTO 50
        ISTEP=ISTEP+2
                                  GOTO 41
45      IF(X.GE.XT(LXT))          GOTO 110
        IHI=LXT
```

```
      subroutine interv(xt,lxt,x,left,mflag)
c*****************************************************************
c
c  xt should be an increasing sequence
c
c*****************************************************************
      integer*4 left,lxt,mflag,j
      real*4 x,xt(20)
         do 10 j=1,lxt
            if(x.lt.xt(j))then
               left=max(1,j-1)
               return
            end if
10       continue
         left=lxt
       return
       end


      subroutine kernel2(x,counts,v,sv,kk,kkk,kkkk)

c     this subroutine estimates the density of data x
c     it returns for each x a value. The sum of values
c     is npart, i.e. the average value is 1.
c     It also estimates the smoothed drift sv, of velocities v
c
c     inputs
c             x array which has the positions of each particle
c
c
c     outputs
c             kk is a vector of length the number of
c              particles which at the jth position contains
c              the number of the horizontal strip in which
c             that particle fell, i.e., its bin number
c
c             kkk is simular, excepts it has the bin no.
c              of the bin where the particle is headed
c              for.
c             counts,
c                      array with smoothed concentration
c                      for each bin
c
      real*4 x(1),v(1),hi,rlo,norm
      integer*4 iwk(11),kkkk(1),i,j,k
      integer*4 kk(1),kkk(1),n
      real*4 fil(1024),rnx,smm,c
      complex counts(1),sv(1)

#include '//stat600/stat/ramos/pick/simul/src/param.com.ftn'
#include '//stat600/stat/ramos/pick/simul/src/param2.com.ftn'
```

```
c   calculate parameters
        hi=vlength+4*h
        rlo=-4*h
        c=(h*8*atan(1.)/(hi-rlo))**2
        norm=1.+8./vlength*h

c   get parameter for packed bed, finds out what bin the vessel
c   ends at, i.e. at bin 4.5 so it knows how to adjust for the
c   packed bed later on...in (*****)
        smm=(vlength/(vlength+8*h)+1)*real(nbins)/2.
        mm=int(smm)
        if(smm-mm.gt.000001)then
           mm=mm+1
        end if


c   get parameter for IMSL's fft
        n=nint(log(real(nbins))/log(2.))

c   initialize variables
        do 10 j=1,nbins
           counts(j)=(0.,0.)
           sv(j)=(0.,0.)
10      continue

c   get counts for each bin


                do 20 j=1,npart
                kk(j)=min(int((x(j)-rlo)/(hi-rlo)*nbins)+1,nbins)
                kk(j)=max(1,kk(j))
        kkk(j)=min(int((x(j)+v(j)*t-rlo)/(hi-rlo)*nbins)+1,nbins)
        kkkk(j)=min(int((x(j)+v(j)*t/2-rlo)/(hi-rlo)*nbins)+1,nbins)
                kkk(j)=max(1,kkk(j))
                kkkk(j)=max(1,kkkk(j))
                counts(kk(j))=counts(kk(j))+1.
                sv(kk(j))=sv(kk(j))+v(j)
20              continue

c   put in packed bed
c   (********)
        counts(mm+1)=counts(mm+1)+pack/incon*npart/nbins*(smm-mm)
        if(mm+2.le.nbins)then
           do 21 j=mm+2,nbins
              counts(j)=pack/incon*npart/nbins
21         continue
        end if
c   normalize sum of velocities at each bin
c                rnx=0.
                do 200 i=1,nbins
                if(real(counts(i)).lt.1.)then
                   sv(i)=1.
                else
                   sv(i)=sv(i)/counts(i)
                end if
c                rnx=rnx+real(counts(i))
200             continue

c   fft histograms
        call fft2c(sv,n,iwk)
        call fft2c(counts,n,iwk)

c   filtering and get conjugate for inverting fft
        do 30 k=1,nbins
           fil(k)=exp(-min(.5*c*min(k-1,nbins+1-k)**2.,38.))
```

```
            counts(k)=fil(k)*conjg(counts(k))/norm/npart
            sv(k)=fil(k)*conjg(sv(k))/nbins
30        continue

c    invert ffts
          call fft2c(counts,n,iwk)
          call fft2c(sv,n,iwk)
          return
          end

          subroutine legend
%include '//stat600/stat/ramos/pick/simul/src/colorinfo.ftn'
          integer*4 status
          integer*2 ypos,xpos
          integer*2 fid,namel,strl
          character*30 fname
          character*4 s_nbins,s_h,s_incon
          character*5 s_ndraw,s_npart,s_nhi
c          character*5 s_incon
%include '//stat600/stat/ramos/pick/simul/src/param.com.ftn'
%include '//stat600/stat/ramos/pick/simul/src/graf.com.ftn'
%include '//stat600/stat/ramos/pick/simul/src/graf2.com.ftn'
%include '//stat600/stat/ramos/pick/simul/src/graf3.com.ftn'
          data space/34/
          fname='/sys/dm/fonts/f9x15           '
          namel=int2(30)
          call gpr_$load_font_file(fname,namel,fid,status)
          call gpr_$set_text_font(fid,status)
          call gpr_$set_text_background_value(0,status)
          call gpr_$set_text_value(white,status)
          ypos=yscl+int2(15)
          xpos=0
          call gpr_$move(xpos,ypos,status)
          call gpr_$text('0',int2(1),status)
          xpos=int2(real(xscl1)/10.-8)
          call gpr_$move(xpos,ypos,status)
          call gpr_$text('Co',int2(2),status)
          xpos=xpos+int2(real(xscl1)/2.)-int2(45)
          call gpr_$move(xpos,ypos,status)
          call gpr_$text('0',int2(1),status)
          xpos=int2(real(xscl1)/2.+real(xscl1)/10.-8)
          call gpr_$move(xpos,ypos,status)
          call gpr_$text('Uo',int2(2),status)
          xpos=int2(15)
          ypos=yscl+2*space-int2(10)
          call gpr_$move(xpos,ypos,status)
          strl=int2(18)
          call gpr_$set_text_value(yellow,status)
          call gpr_$text('       FLUX       ',strl,status)
          ypos=ypos+space
          call gpr_$move(xpos,ypos,status)
          call gpr_$set_text_value(blue,status)
          call gpr_$text('   CONCENTRATION  ',strl,status)
          xpos=int2(real(xscl1)/2.)
          ypos=yscl+2*space-int2(10)
          call gpr_$move(xpos,ypos,status)
          call gpr_$set_text_value(magenta,status)
          call gpr_$text('       VELOCITY   ',strl,status)
          xpos=int2(100)
          ypos=yscl+int2(4)*space
          call gpr_$move(xpos,ypos,status)
          call gpr_$set_text_value(green,status)
          call gpr_$text('BINS IN HISTOGRAM ',strl,status)
          xpos=xpos+space
          write(s_nbins,10)nbins
10        format(i4)
```

```
      strl=4
      call gpr_$text(s_nbins,strl,status)
      strl=int2(18)
      xpos=int2(100)
      ypos=ypos+space
      call gpr_$move(xpos,ypos,status)
      call gpr_$text(' SMOOTHING WINDOW ',strl,status)
      write(s_h,11)(2*h)
11    format(f4.1)
      strl=4
      call gpr_$text(s_h,strl,status)
      strl=int2(18)
      xpos=xscl1+100
      ypos=yscl+2*space -int2(10)
      call gpr_$move(xpos,ypos,status)
      call gpr_$text('  INITIAL CONCENTRATION  ',int2(25),status)
      write(s_incon,12)incon
      call gpr_$text(s_incon,int2(4),status)
12    format(f4.3)
      ypos=ypos+space
      call gpr_$move(xpos,ypos,status)
      call gpr_$text('        ELAPSED TIME    ',int2(24),status)
      ypos=ypos+space
      call gpr_$move(xpos,ypos,status)
      call gpr_$text('PARTICLES IN SIMULATION ',int2(24),status)
      write(s_npart,13)npart
13    format(i5)
      call gpr_$text(s_npart,int2(5),status)
      ypos=ypos+space
      call gpr_$move(xpos,ypos,status)
      call gpr_$text('       PARTICLES DRAWN  ',int2(24),status)
      write(s_ndraw,13)ndraw
      call gpr_$text(s_ndraw,int2(5),status)
      ypos=ypos+space
      call gpr_$move(xpos,ypos,status)
      call gpr_$text('  PARTICLES HIGHLIGHTED ',int2(24),status)
      write(s_nHI,13)nhi
      call gpr_$text(s_nhi,int2(5),status)
      return
      end

      subroutine line (x,y,nd,col,status)
      integer*2 x(1),y(1),nd
      integer*4 status,col
      call gpr_$move(x(1),y(1),status)
      call gpr_$set_draw_value(col,status)
      call gpr_$polyline(x,y,nd,status)
      return
      end
      subroutine packed(icol)
%include '//stat600/stat/ramos/pick/simul/src/graf.com.ftn'
%include '//stat600/stat/ramos/pick/simul/src/param2.com.ftn'
      integer*2 x(4),y(4)
      integer*4 n,k,icol
      x(1)=xscl1+xscl2
      x(2)=x(1)
      x(3)=xscl1
      x(4)=x(3)
c
      y(1)=int2(vlength/ovlength*yscl)
      y(2)=yscl
      y(3)=yscl
      y(4)=y(1)

      call fill(x,y,4,icol)
      return
```

```
        end
        real*4 incon,pack,h,t
        integer*4 npart,nbins,mm
        common /param/mm,incon,pack,npart,nbins,h,t
        real*4 ovlength,vlength
        common /param2/ovlength,vlength
        subroutine paramet(con,rmu,gsd,c,b,sd)
c   parametrization for mean and sd in terms of concentration
c
c       inputs
c               con
c       outputs
c               rmu,gsd,c,b,sd
c

        real*4 con,rmu,co,b,sd,c
        real*4 gsd,conn
%include '//stat600/stat/ramos/pick/simul/src/param.com.ftn'
        conn=con*incon
        co=asinh(conn*10000.)
        if (co.le.3.69)then
        rmu=1-.0615*co+.03136*co**2.
        else
        co=co-3.69
        rmu=1.2001+2.295*co-1.447*co**2.+.2667*co**3.-.01547*co**4.
        end if
        rmu=max(rmu,0.)
        b=2.5*conn+.001/(conn+.000001)
        sd=.6*rmu*(1-exp(max(-30.,-100*conn**2)))
        c=exp(max(-30.,-b*t))
        gsd=sqrt(1-c**2)*sd
        return
        end
        subroutine paramet(con,rmu,gsd,c,b,sd)
c   parametrization for mean and sd in terms of concentration
c
c       inputs
c               con
c       outputs
c               rmu,gsd,c,b,sd
c

        real*4 con,rmu,co,b,sd,c
        real*4 gsd,conn,asinh,ppvalu
%include '//stat600/stat/ramos/pick/simul/src/param.com.ftn'
%include '//stat600/stat/ramos/pick/simul/src/spline.com.ftn'
        conn=con*incon
        co=asinh(conn*10000.)+3*log(1+20*conn)
        rmu=ppvalu (rknots,coef,jdims,mdims,co,0)
        sd=.807*rmu**2*conn**.33333333
        b=2.5*conn+.001/(conn+.000001)
        c=exp(max(-30.,-b*t))
        gsd=sqrt(1-c**2)*sd
        return
        end

         subroutine plot(x,y,n,ival)
c
c Plots curve through points (x(i),y(i)) , i=1,...,n
c
        integer*2 x(n),y(n)
        integer*4 ival,status
        call gpr_$set_draw_value(ival,status)
        do 10 i=1,n
        call gpr_$move(x(i),y(i),status)
        call gpr_$line(x(i),y(i),status)
```

```
10      continue
        return
        end


        real function ppvalu (break,coef,l,k,x,jderiv)

cREFERENCE: DE BOOR, CARL. A PRACTICAL GUIDE TO
C           SPLINES. SPRINGER-VERLAG, 1978, PP. 89-93
C

        integer*4 jderiv,k,l,i,m,ndummy
        real*4 break(20),coef(20,4),x,fmmjdr,h
        ppvalu=0.
        fmmjdr=k-jderiv
        call interv(break,l,x,i,ndummy)
        h=x-break(i)
        do 10 m=k,jderiv+1,-1
           ppvalu=ppvalu/fmmjdr*h
           ppvalu=ppvalu+coef(i,m)
10         fmmjdr=fmmjdr-1.
        return
        end


         real function asinh (x)
         real*4 x
         x=max(x,10.**(-16))
         asinh=log(x+sqrt(1.+x**2.))
         return
         end
         subroutine axes
%include '//stat600/stat/ramos/pick/simul/src/colorinfo.ftn'
         integer*2 x(2),y(2)
         integer*4 m,col,status,j
%include '//stat600/stat/ramos/pick/simul/src/graf.com.ftn'
         m=int(1./ban)
         y(1)=0
         y(2)=yscl
         do 20 j=0,m
         col=red
         x(1)=int2(real(j)*xscl1/m)
         x(2)=x(1)
         if(j.eq.m/2.or.j.eq.0)col=white
         call line(x,y,int2(2),col,status)
20       continue
         return
         end

         subroutine clip(j)
c     sets clipping window to the left or right or full bitmap
c     (according to j=0 or j=1 or j=2 )
         integer*4 j,status
         integer*2 window(2,2)
%include '//stat600/stat/ramos/pick/simul/src/graf.com.ftn'
         window(1,1)=int2(0)
         if(j.eq.1)window(1,1)=xscl1
         window(2,1)=int2(0)
         if(j.eq.0)window(1,2)=xscl1-int2(1)
         if(j.eq.1)window(1,2)=xscl2-int2(1)
         if(j.eq.2)window(1,2)=xscl1+xscl2-int2(1)
         window(2,2)=yscl-int2(1)
         if(j.eq.2)window(2,2)=int2(1023)
         call gpr_$set_clip_window(window,status)
         call gpr_$set_clipping_active((.true.),status)
         return
```

```
        end
       integer*4
     +            black,
     +            red,
     +            green,
     +            blue,
     +            cyan,
     +            magenta,
     +            yellow,
     +            white
       parameter(
     +            black=0,
     +            red=1,
     +            green=2,
     +            blue=3,
     +            cyan=4,
     +            magenta=6,
     +            yellow=5,
     +            white=7    )
        subroutine erase(k)
%include '//stat600/stat/ramos/pick/simul/src/graf.com.ftn'
        integer*2 x(4),y(4)
        integer*4 n,k

        x(1)=xscl2*int2(k)+xscl1-int2(1)
        x(2)=x(1)
        x(3)=xscl1*int2(k)-int2(1)
        x(4)=x(3)
c
        y(1)=int2(0)
        y(2)=yscl-int2(1)
        y(3)=y(2)
        y(4)=int2(0)

        call fill(x,y,4,0)
        return
        end
C*****************************************************************
c    SUBROUTINE TO fill POLYGON
C
C        INPUTS: X,Y -- COORDINATES IN PIXEL VALUE
C                  N -- NUMBER OF SIDES OF POLYGON
C               COL -- COLOR TO USE IN FILLING
C
C
C*****************************************************************
        subroutine fill(x,y,n,col)
C
        integer*2 x(1),y(1)
        integer*4 st,col,n
        call gpr_$set_fill_value(col,st)
        call gpr_$start_pgon(x(n),y(n),st)
        call gpr_$pgon_polyline(x,y,int2(n),st)
        call gpr_$close_fill_pgon(st)
        return
        end
       subroutine g_init
C
C Allocate and initialize bitmap for graphics in borrowed mode
C
        integer*4 desc,bl_desc,status
        integer*2 mode,hi,unit,bsize(2)
        integer*2 window(2,2)
        data mode/0/
        data hi/3/
        data unit/1/
```

```
        data bsize/1024,1024/
        window(1,1)=int2(0)
        window(2,1)=int2(0)
        window(1,2)=int2(1023)
        window(2,2)=int2(1023)
        call gpr_$init(mode,unit,bsize,hi,desc,status)
        call gpr_$set_bitmap(desc,status)
        call gpr_$allocate_attribute_block(bl_desc,status)
        call gpr_$set_attribute_block(bl_desc,status)
        call gpr_$set_clip_window(window,status)
        call gpr_$set_clipping_active((.true.),status)
        return
        end

        integer*2 xscl1,xscl2,yscl
        real*4 ban
        common /graf/ xscl1,xscl2,yscl,ban
        integer*2 space
        common /graf2/ space
        integer*4 nhi,ndraw
        common /graf3/nhi,ndraw



C-----------------------------------------------------------------------
C
      SUBROUTINE GUNIF(DSEED,NR,R)
C                                     SPECIFICATIONS FOR ARGUMENTS
      INTEGER             NR
      REAL                R(NR)
      DOUBLE PRECISION    DSEED
C                                     SPECIFICATIONS FOR LOCAL VARIABLES
      INTEGER             I
      REAL                U
      DOUBLE PRECISION    D2P31M,D2PN31
C                                     D2P31M = (2**31)-1
C                                     D2PN31 = (2**31)
      DATA                D2P31M/2147483647.D0/
      DATA                D2PN31/2147483711.D0/
C                                     FIRST EXECUTABLE STATEMENT
      DO 10 I = 1,NR
         DSEED = DMOD(16807.D0*DSEED,D2P31M)
         R(I) = real(DSEED/D2PN31)
   10 CONTINUE
      RETURN
      END

        SUBROUTINE PAUSE
c   This subroutine holds the screen in borrowed mode so that
c   the plot does not immediately disappear.
c   The screen is cleared after any key is struck, with a printable
c   screen dump produced if the key is 'F1'
%no list
%include '/sys/ins/gpr.ins.ftn'
%list
        CHARACTER*1 e_d
        INTEGER*4 status
        INTEGER*2 pos(2),e_t,key_set(16),window(2,2)
        LOGICAL dummy

        data key_set/16*16#ffff/
        window(1,1) = 0
        window(2,1) = 0
        window(1,2) = 1023
        window(2,2) = 1023
```

```
        call GPR_$ENABLE_INPUT( GPR_$KEYSTROKE,key_set,status )
        dummy = GPR_$EVENT_WAIT( e_t,e_d,pos,status )

c If the key 'F1' is struck then a screen dump is made:
        if( e_d .eq. char(16#C0) ) then
        call savpic('//stat600/stat/ramos/tmp/scrdmp',
     +              int2(31),window)
        end if

        call GPR_$CLEAR( int4(0),status )
10      continue
        dummy = GPR_$COND_EVENT_WAIT( e_t,e_d,pos,status )
        if( e_t.ne.GPR_$NO_EVENT ) go to 10
        return
        end


        SUBROUTINE SAVPIC( filename,length,window )
c   This subroutine dumps the bitmap onto a file which can be
c   printed using  PRF ... -PLOT.
        CHARACTER*(*) filename
        INTEGER*2 window(2,2),length
        INTEGER*2 i,plane_0,hiplane,bit_or,str_id,line_wid,bpi,rop(8)
        INTEGER*2 gmfstatus
        INTEGER*4 status,bmdesc,pointer
        LOGICAL invert
        PARAMETER( invert   = .true.,
     +             bpi      = 100,
     +             plane_0  = 0,
     +             bit_or   = 7,
     +             hiplane  = 3,
     +             gmfstatus = 1 )

        call GPR_$INQ_BITMAP( bmdesc,status )
        call GPR_$INQ_RASTER_OPS( rop,status )

c   Copy all the planes into plane zero:

        call GPR_$SET_RASTER_OP( plane_0,bit_or,status )
        do i=1,hiplane
            call GPR_$BIT_BLT
     +      ( bmdesc,window,i,window(1,1),plane_0,status )
        end do

c   Dump the bitmap onto a file:
            call SHOW('4')
        call GPR_$INQ_BITMAP_POINTER( bmdesc,pointer,line_wid,status )
        call GMF_$OPEN( filename,length,gmfstatus,str_id,status )
        call GMF_$COPY_PLANE( str_id,invert,bpi,pointer,window(1,2),
     +      window(2,2),line_wid,status )
        call GMF_$CLOSE( str_id,status )

        call GPR_$SET_RASTER_OP( plane_0,rop(1),status )

        return
        end


        SUBROUTINE SHOW( ch )
%nolist
%include '/sys/ins/cal.ins.ftn'
%include '/sys/ins/time.ins.ftn'
%list
        character ch*1
        integer*2 clock(3),ixp,iyp
        integer*4 seconds,status
```

```
        data ixp,iyp,seconds/1000,20,1/
        call CAL_$SEC_TO_CLOCK( seconds,clock )
        call TIME_$WAIT( TIME_$RELATIVE,clock,status )
        return
        end
        subroutine pause
%no list
%include '/sys/ins/base.ins.ftn'
%include '/sys/ins/error.ins.ftn'
%include '/sys/ins/gpr.ins.ftn'
%list
c     This subroutine holds the screen in borrowed mode so that
c the plot does not immediately disappear.
c
        character*1 e_d
        integer*4 status
        integer*2 position(2),e_t,key_set(16),rectangle(4)
        logical unobscured
c
        data key_set/16*16#ffff/
        rectangle(1) = 0
        rectangle(2) = 0
        rectangle(3) = 1023
        rectangle(4) = 1023
c
        call gpr_$enable_input(gpr_$keystroke,key_set,status)
        unobscured = gpr_$event_wait(event_type,e_d,position,status)

c   If the key 'F1' is struck then a screen dump is made:
        if (e_d .eq. char(16#C0)) then
        call savpic('//stat600/stat/ramos/pick/simul/pic/scrdmp',
     +              int2(42),rectangle,status)
        end if

c        CALL GPR_$CLEAR(INT4(0),STATUS)
        return
        end


        subroutine savpic(filename,length,rectangle,status)
c
c       This subroutine dumps the bitmap onto a file which can be
c       printed using  PRF ... -PLOT.
c
%no list
%include '/sys/ins/base.ins.ftn'
%include '/sys/ins/gmf.ins.ftn'
%include '/sys/ins/gpr.ins.ftn'
%include '/sys/ins/error.ins.ftn'
%list
c
c
        character*(*) filename
        integer*2 rectangle(4),wpl,length,str_id,line_wid,bpi
        integer*2 i,plane_0,bit_or,hiplane
        integer*4 status,bmdesc,pointer
        logical invert
        parameter (invert   = .true.,
     +             bpi      = 100,
     +             plane_0  = 0,
     +             bit_or   = 7,
     +             hiplane  = 3 )

        call gpr_$inq_bitmap(bmdesc,status)

c   Copy all the planes into plane zero:
```

```
        call GPR_$SET_RASTER_OP( plane_0, bit_or, status )
        do i=1,hiplane
            call GPR_$BIT_BLT
     +      ( bmdesc,rectangle,i,rectangle,plane_0,status )
        end do

c   Dump the bitmap onto a file:

        call gpr_$inq_bitmap_pointer( bmdesc,pointer,line_wid,status )
c        wpl = (rectangle(3)+15)/16
        wpl = line_wid
        call gmf_$open( filename,length,gmf_$overwrite,str_id,status )
        call gmf_$copy_subplane( str_id,invert,bpi,pointer,rectangle(3),
     +      rectangle(4),rectangle(1),rectangle(2),wpl,status )
        call gmf_$close( str_id,status )

        return
        end
        subroutine histodraw(conc,smv,kk)
%include '//stat600/stat/ramos/pick/simul/src/colorinfo.ftn'
        integer*2 ycoord(1024),conarray(1024),velarray(1024)
        integer*2 fluxarray(1024)
        integer*4 status,jj,jk,j,kk,col1,col2,col3
        real*4 y,origin,half
        complex conc(1),smv(1)
%include '//stat600/stat/ramos/pick/simul/src/graf.com.ftn'
%include '//stat600/stat/ramos/pick/simul/src/param.com.ftn'
%include '//stat600/stat/ramos/pick/simul/src/param2.com.ftn'
c                                EXE STATR
        half=(vlength+8*h)/nbins/2
        origin=-4*h+half
c                                                ycoord(1)=int2(-4*h/vlength*yscl)
c                                                conarray(1)=int2(0)
        do 10 j=1,nbins

c                                                jj=j/2+mod(j,2)
c                                                jk=(j+1)/2-mod(j,2)
           conarray(j)=int2(min(real(conc(j))*ban*xscl1,xscl1))
           velarray(j)=int2(min(real(smv(j)+5)*ban*xscl1,xscl1))
      fluxarray(j)=int2(min(real(conc(j)*smv(j))*ban*xscl1,xscl1))
           y=(origin+(j-1)*2*half)/ovlength*yscl
c                                                y=(-4*h+(jj-1)*(vlength+8*h)/nbins)/c
           ycoord(j)=int2(y)
10      continue
c                                                velarray(1)=velarray(2)
c                                                fluxarray(1)=fluxarray(2)
         call clip(0)
c         col1=mod(kk,7)+1
c         col2=mod(kk*8+1,7)+1
c         col3=mod(kk*64+2,7)+1
         call line (conarray,ycoord,int2(nbins),blue,status)
         call line (velarray,ycoord,int2(nbins),magenta,status)
         call line (fluxarray,ycoord,int2(nbins),yellow,status)
          call clip(2)
        return
        end
        subroutine interv(xt.lxt,x,left,mflag)
c
cREFERENCE: DE BOOR, CARL. A PRACTICAL GUIDE TO
C           SPLINES. SPRINGER-VERLAG, 1978, PP. 89-93
C
         INTEGER LEFT, LXT,MFLAG,IHI,ILO,ISTEP,MIDDLE
         REAL X ,XT(LXT)
         DATA ILO /1/
         SAVE ILO
```

```
          IHI=ILO+1
          IF (IHI.LT.LXT)GOTO 20
          IF (X.GE.XT(LXT))GOTO 110
          IF (LXT.LE.1)GOTO 90
          ILO=LXT-1
20        IF (X.GE.XT(IHI))GOTO 40
          IF (X.GE.XT(ILO))GOTO 100
          ISTEP=1
31        IHI=ILO
          ILO=IHI-ISTEP
          IF(ILO.LE.1)GOTO 35
          IF(X.GE.XT(ILO))GOTO50
          ISTEP=ISTEP+2
35        ILO=1
          IF(X.LT.XT(1))GOTO 90
          GOTO 50
40        ISTEP=1
41        ILO=IHI
          IHI=ILO+ISTEP
          IF(IHI.GE.LXT)            GOTO 45
          IF(X.LT.XT(IHI))          GOTO 50
          ISTEP=ISTEP+2
                                    GOTO 41
45        IF(X.GE.XT(LXT))          GOTO 110
          IHI=LXT
```

```
        subroutine interv(xt,lxt,x,left,mflag)
c*****************************************************************
c
c  xt should be an increasing sequence
c
c*****************************************************************
        integer*4 left,lxt,mflag,j
        real*4 x,xt(20)
           do 10 j=1,lxt
              if(x.lt.xt(j))then
                 left=max(1,j-1)
                 return
              end if
10         continue
           left=lxt
         return
         end
```

```
        subroutine kernel2(x,counts,v,sv,kk,kkk,kkkk)

c       this subroutine estimates the density of data x
c       it returns for each x a value. The sum of values
c       is npart, i.e. the average value is 1.
c       It also estimates the smoothed drift sv, of velocities v
c
c       inputs
c              x array which has the positions of each particle
c
c
c       outputs
```

```
c               kk is a vector of length the number of
c                particles which at the jth position contains
c                the number of the horizontal strip in which
c               that particle fell, i.e., its bin number
c
c               kkk is simular, excepts it has the bin no.
c                of the bin where the particle is headed
c                for.
c               counts,
c                        array with smoothed concentration
c                        for each bin
c
        real*4 x(1),v(1),hi,rlo,norm
        integer*4 iwk(11),kkkk(1),i,j,k
        integer*4 kk(1),kkk(1),n
        real*4 fil(1024),rnx,smm,c
        complex counts(1),sv(1)

%include '//stat600/stat/ramos/pick/simul/src/param.com.ftn'
%include '//stat600/stat/ramos/pick/simul/src/param2.com.ftn'

c   calculate parameters
        hi=vlength+4*h
        rlo=-4*h
        c=(h*8*atan(1.)/(hi-rlo))**2
        norm=1.+8./vlength*h

c   get parameter for packed bed, finds out what bin the vessel
c   ends at, i.e. at bin 4.5 so it knows how to adjust for the
c   packed bed later on...in (*****)
        smm=(vlength/(vlength+8*h)+1)*real(nbins)/2.
        mm=int(smm)
        if(smm-mm.gt.000001)then
           mm=mm+1
        end if


c   get parameter for IMSL's fft
        n=nint(log(real(nbins))/log(2.))

c   initialize variables
        do 10 j=1,nbins
           counts(j)=(0.,0.)
           sv(j)=(0.,0.)
10      continue

c   get counts for each bin


                do 20 j=1,npart
                kk(j)=min(int((x(j)-rlo)/(hi-rlo)*nbins)+1,nbins)
                kk(j)=max(1,kk(j))
        kkk(j)=min(int((x(j)+v(j)*t-rlo)/(hi-rlo)*nbins)+1,nbins)
        kkkk(j)=min(int((x(j)+v(j)*t/2-rlo)/(hi-rlo)*nbins)+1,nbins)
                kkk(j)=max(1,kkk(j))
                kkkk(j)=max(1,kkkk(j))
                counts(kk(j))=counts(kk(j))+1.
                sv(kk(j))=sv(kk(j))+v(j)
20              continue

c   put in packed bed
c   (********)
        counts(mm+1)=counts(mm+1)+pack/incon*npart/nbins*(smm-mm)
        if(mm+2.le.nbins)then
           do 21 j=mm+2,nbins
              counts(j)=pack/incon*npart/nbins
```

```
21          continue
         end if
c    normalize sum of velocities at each bin
c                 rnx=0.
                  do 200 i=1,nbins
                  if(real(counts(i)).lt.1.)then
                     sv(i)=1.
                  else
                     sv(i)=sv(i)/counts(i)
                  end if
c                  rnx=rnx+real(counts(i))
200               continue

c    fft histograms
         call fft2c(sv,n,iwk)
         call fft2c(counts,n,iwk)

c    filtering and get conjugate for inverting fft
         do 30 k=1,nbins
            fil(k)=exp(-min(.5*c*min(k-1,nbins+1-k)**2.,38.))
            counts(k)=fil(k)*conjg(counts(k))/norm/npart
            sv(k)=fil(k)*conjg(sv(k))/nbins
30       continue

c    invert ffts
         call fft2c(counts,n,iwk)
         call fft2c(sv,n,iwk)
         return
         end

         subroutine legend
%include '//stat600/stat/ramos/pick/simul/src/colorinfo.ftn'
         integer*4 status
         integer*2 ypos,xpos
         integer*2 fid,namel,strl
         character*30 fname
         character*4 s_nbins,s_h,s_incon
         character*5 s_ndraw,s_npart,s_nhi
c         character*5 s_incon
%include '//stat600/stat/ramos/pick/simul/src/param.com.ftn'
%include '//stat600/stat/ramos/pick/simul/src/graf.com.ftn'
%include '//stat600/stat/ramos/pick/simul/src/graf2.com.ftn'
%include '//stat600/stat/ramos/pick/simul/src/graf3.com.ftn'
         data space/34/
         fname='/sys/dm/fonts/f9x15          '
         namel=int2(30)
         call gpr_$load_font_file(fname,namel,fid,status)
         call gpr_$set_text_font(fid,status)
         call gpr_$set_text_background_value(0,status)
         call gpr_$set_text_value(white,status)
         ypos=yscl+int2(15)
         xpos=0
         call gpr_$move(xpos,ypos,status)
         call gpr_$text('0',int2(1),status)
         xpos=int2(real(xscl1)/10.-8)
         call gpr_$move(xpos,ypos,status)
         call gpr_$text('Co',int2(2),status)
         xpos=xpos+int2(real(xscl1)/2.)-int2(45)
         call gpr_$move(xpos,ypos,status)
         call gpr_$text('0',int2(1),status)
         xpos=int2(real(xscl1)/2.+real(xscl1)/10.-8)
         call gpr_$move(xpos,ypos,status)
         call gpr_$text('Uo',int2(2),status)
         xpos=int2(15)
         ypos=yscl+2*space-int2(10)
         call gpr_$move(xpos,ypos,status)
```

```
      strl=int2(18)
      call gpr_$set_text_value(yellow,status)
      call gpr_$text('       FLUX       ',strl,status)
      ypos=ypos+space
      call gpr_$move(xpos,ypos,status)
      call gpr_$set_text_value(blue,status)
      call gpr_$text('  CONCENTRATION  ',strl,status)
      xpos=int2(real(xscl1)/2.)
      ypos=yscl+2*space-int2(10)
      call gpr_$move(xpos,ypos,status)
      call gpr_$set_text_value(magenta,status)
      call gpr_$text('       VELOCITY   ',strl,status)
      xpos=int2(100)
      ypos=yscl+int2(4)*space
      call gpr_$move(xpos,ypos,status)
      call gpr_$set_text_value(green,status)
      call gpr_$text('BINS IN HISTOGRAM ',strl,status)
      xpos=xpos+space
      write(s_nbins,10)nbins
10    format(i4)
      strl=4
      call gpr_$text(s_nbins,strl,status)
      strl=int2(18)
      xpos=int2(100)
      ypos=ypos+space
      call gpr_$move(xpos,ypos,status)
      call gpr_$text(' SMOOTHING WINDOW ',strl,status)
      write(s_h,11)(2*h)
11    format(f4.1)
      strl=4
      call gpr_$text(s_h,strl,status)
      strl=int2(18)
      xpos=xscl1+100
      ypos=yscl+2*space -int2(10)
      call gpr_$move(xpos,ypos,status)
      call gpr_$text('  INITIAL CONCENTRATION  ',int2(25),status)
      write(s_incon,12)incon
      call gpr_$text(s_incon,int2(4),status)
12    format(f4.3)
      ypos=ypos+space
      call gpr_$move(xpos,ypos,status)
      call gpr_$text('        ELAPSED TIME    ',int2(24),status)
      ypos=ypos+space
      call gpr_$move(xpos,ypos,status)
      call gpr_$text('PARTICLES IN SIMULATION ',int2(24),status)
      write(s_npart,13)npart
13    format(i5)
      call gpr_$text(s_npart,int2(5),status)
      ypos=ypos+space
      call gpr_$move(xpos,ypos,status)
      call gpr_$text('       PARTICLES DRAWN  ',int2(24),status)
      write(s_ndraw,13)ndraw
      call gpr_$text(s_ndraw,int2(5),status)
      ypos=ypos+space
      call gpr_$move(xpos,ypos,status)
      call gpr_$text('  PARTICLES HIGHLIGHTED ',int2(24),status)
      write(s_nHI,13)nhi
      call gpr_$text(s_nhi,int2(5),status)
      return
      end

      subroutine line (x,y,nd,col,status)
      integer*2 x(1),y(1),nd
      integer*4 status,col
      call gpr_$move(x(1),y(1),status)
      call gpr_$set_draw_value(col,status)
```

```
        call gpr_$polyline(x,y,nd,status)
        return
        end
        subroutine packed(icol)
%include '//stat600/stat/ramos/pick/simul/src/graf.com.ftn'
%include '//stat600/stat/ramos/pick/simul/src/param2.com.ftn'
        integer*2 x(4),y(4)
        integer*4 n,k,icol
        x(1)=xscl1+xscl2
        x(2)=x(1)
        x(3)=xscl1
        x(4)=x(3)
c
        y(1)=int2(vlength/ovlength*yscl)
        y(2)=yscl
        y(3)=yscl
        y(4)=y(1)

        call fill(x,y,4,icol)
        return
        end
        real*4 incon,pack,h,t
        integer*4 npart,nbins,mm
        common /param/mm,incon,pack,npart,nbins,h,t
        real*4 ovlength,vlength
        common /param2/ovlength,vlength
        subroutine paramet(con,rmu,gsd,c,b,sd)
c   parametrization for mean and sd in terms of concentration
c
c       inputs
c               con
c       outputs
c               rmu,gsd,c,b,sd
c

        real*4 con,rmu,co,b,sd,c
        real*4 gsd,conn
%include '//stat600/stat/ramos/pick/simul/src/param.com.ftn'
        conn=con*incon
        co=asinh(conn*10000.)
        if (co.le.3.69)then
        rmu=1-.0615*co+.03136*co**2.
        else
        co=co-3.69
        rmu=1.2001+2.295*co-1.447*co**2.+.2667*co**3.-.01547*co**4.
        end if
        rmu=max(rmu,0.)
        b=2.5*conn+.001/(conn+.000001)
        sd=.6*rmu*(1-exp(max(-30.,-100*conn**2)))
        c=exp(max(-30.,-b*t))
        gsd=sqrt(1-c**2)*sd
        return
        end
        subroutine paramet(con,rmu,gsd,c,b,sd)
c   parametrization for mean and sd in terms of concentration
c
c       inputs
c               con
c       outputs
c               rmu,gsd,c,b,sd
c

        real*4 con,rmu,co,b,sd,c
        real*4 gsd,conn,asinh,ppvalu
%include '//stat600/stat/ramos/pick/simul/src/param.com.ftn'
%include '//stat600/stat/ramos/pick/simul/src/spline.com.ftn'
```

```
        conn=con*incon
        co=asinh(conn*10000.)+3*log(1+20*conn)
        rmu=ppvalu (rknots,coef,jdims,mdims,co,0)
        sd=.807*rmu**2*conn**.33333333
        b=2.5*conn+.001/(conn+.000001)
        c=exp(max(-30.,-b*t))
        gsd=sqrt(1-c**2)*sd
        return
        end

         subroutine plot(x,y,n,ival)
C
C Plots curve through points (x(i),y(i)) , i=1,...,n
C
        integer*2 x(n),y(n)
        integer*4 ival,status
        call gpr_$set_draw_value(ival,status)
        do 10 i=1,n
        call gpr_$move(x(i),y(i),status)
        call gpr_$line(x(i),y(i),status)
10      continue
        return
        end


       real function ppvalu (break,coef,l,k,x,jderiv)

cREFERENCE: DE BOOR, CARL. A PRACTICAL GUIDE TO
C           SPLINES. SPRINGER-VERLAG, 1978, PP. 89-93
C

       integer*4 jderiv,k,l,i,m,ndummy
       real*4 break(20),coef(20,4),x,fmmjdr,h
       ppvalu=0.
       fmmjdr=k-jderiv
       call interv(break,l,x,i,ndummy)
       h=x-break(i)
       do 10 m=k,jderiv+1,-1
          ppvalu=ppvalu/fmmjdr*h
          ppvalu=ppvalu+coef(i,m)
10        fmmjdr=fmmjdr-1.
       return
       end
```

## References


1. Happel, J. and Brenner, H. (1965). Low Reynolds Number Hydrodynamics. Prentice-Hall, Englewood Cliffs, N.J.

2. Hesse, C.H. (1990). The One-Sided Barrier Problem for an Integrated Ornstein-Uhlenbeck Process. Technical Report, Department of Statistics, University of California at Berkeley.

3. Johne, R. (1965). Einfluss der Konzentration einer monodispersen Suspension auf die Geschwindigkeit ihrer Teilchen. Dissertation Universität Karlsruhe.

4. Kynch, G.J. (1952). A Theory of Sedimentation. *Trans. Faraday Soc.* **48**, 166-176.

5. Pickard, D.K. and Tory, E.M. (1977). A Markov Model for Sedimentation. *J. Math. Anal. Appl.* **60**, 349-369.

6. Pickard, D.K. and Tory, E.M. (1979). Experimental Implications of a Markov Model for Sedimentation. *J. Math. Anal. Appl.* **72**, 150-176.

7. Pickard, D.K. and Tory, E.M. (1982). Extensions of a Markov Model for Sedimentation. *J. Math. Anal. Appl.* **86**, 442-470 (1982).

8. Pickard, D.K. and Tory, E.M. (1986). A Markov Model for Sedimentation: Fundamental Issues and Insights. Technical Report. Department of Mathematics and Statistics, Queens University, Kingston.

9. Shannon, P.T., DeHass, R.D., Stroupe, E.P. and Tory, E.M. (1964). Batch and Continuous Thickening: Prediction of Batch Settling Behavior from Initial Rate Data with Results for Rigid Spheres. *Ind. Eng. Chem. Fundam.* **3**, 250-260.

# TECHNICAL REPORTS

## Statistics Department

## University of California, Berkeley

1. BREIMAN, L. and FREEDMAN, D. (Nov. 1981, revised Feb. 1982). How many variables should be entered in a regression equation? Jour. Amer. Statist. Assoc., March 1983, 78, No. 381, 131-136.

2. BRILLINGER, D. R. (Jan. 1982). Some contrasting examples of the time and frequency domain approaches to time series analysis. Time Series Methods in Hydrosciences, (A. H. El-Shaarawi and S. R. Esterby, eds.) Elsevier Scientific Publishing Co., Amsterdam, 1982, pp. 1-15.

3. DOKSUM, K. A. (Jan. 1982). On the performance of estimates in proportional hazard and log-linear models. Survival Analysis, (John Crowley and Richard A. Johnson, eds.) IMS Lecture Notes - Monograph Series, (Shanti S. Gupta, series ed.) 1982, 74-84.

4. BICKEL, P. J. and BREIMAN, L. (Feb. 1982). Sums of functions of nearest neighbor distances, moment bounds, limit theorems and a goodness of fit test. Ann. Prob., Feb. 1982, 11. No. 1, 185-214.

5. BRILLINGER, D. R. and TUKEY, J. W. (March 1982). Spectrum estimation and system identification relying on a Fourier transform. The Collected Works of J. W. Tukey, vol. 2, Wadsworth, 1985, 1001-1141.

6. BERAN, R. (May 1982). Jackknife approximation to bootstrap estimates. Ann. Statist., March 1984, 12 No. 1, 101-118.

7. BICKEL, P. J. and FREEDMAN, D. A. (June 1982). Bootstrapping regression models with many parameters. Lehmann Festschrift, (P. J. Bickel, K. Doksum and J. L. Hodges, Jr., eds.) Wadsworth Press, Belmont, 1983, 28-48.

8. BICKEL, P. J. and COLLINS, J. (March 1982). Minimizing Fisher information over mixtures of distributions. Sankhyā, 1983, 45, Series A, Pt. 1, 1-19.

9. BREIMAN, L. and FRIEDMAN, J. (July 1982). Estimating optimal transformations for multiple regression and correlation.

10. FREEDMAN, D. A. and PETERS, S. (July 1982, revised Aug. 1983). Bootstrapping a regression equation: some empirical results. JASA, 1984, 79, 97-106.

11. EATON, M. L. and FREEDMAN, D. A. (Sept. 1982). A remark on adjusting for covariates in multiple regression.

12. BICKEL, P. J. (April 1982). Minimax estimation of the mean of a mean of a normal distribution subject to doing well at a point. Recent Advances in Statistics, Academic Press, 1983.

14. FREEDMAN, D. A., ROTHENBERG, T. and SUTCH, R. (Oct. 1982). A review of a residential energy end use model.

15. BRILLINGER, D. and PREISLER, H. (Nov. 1982). Maximum likelihood estimation in a latent variable problem. Studies in Econometrics, Time Series, and Multivariate Statistics, (eds. S. Karlin, T. Amemiya, L. A. Goodman). Academic Press, New York, 1983, pp. 31-65.

16. BICKEL, P. J. (Nov. 1982). Robust regression based on infinitesimal neighborhoods. Ann. Statist., Dec. 1984, 12, 1349-1368.

17. DRAPER, D. C. (Feb. 1983). Rank-based robust analysis of linear models. I. Exposition and review. Statistical Science, 1988, Vol. 3 No. 2 239-271.

18. DRAPER, D. C. (Feb 1983). Rank-based robust inference in regression models with several observations per cell.

19. FREEDMAN, D. A. and FIENBERG, S. (Feb. 1983, revised April 1983). Statistics and the scientific method, Comments on and reactions to Freedman, A rejoinder to Fienberg's comments. Springer New York 1985 Cohort Analysis in Social Research, (W. M. Mason and S. E. Fienberg, eds.).

20. FREEDMAN, D. A. and PETERS, S. C. (March 1983, revised Jan. 1984). Using the bootstrap to evaluate forecasting equations. J. of Forecasting, 1985, Vol. 4, 251-262.

21. FREEDMAN, D. A. and PETERS, S. C. (March 1983, revised Aug. 1983). Bootstrapping an econometric model: some empirical results. JBES, 1985, 2, 150-158.

22. FREEDMAN, D. A. (March 1983). Structural-equation models: a case study.

23. DAGGETT, R. S. and FREEDMAN, D. (April 1983, revised Sept. 1983). Econometrics and the law: a case study in the proof of antitrust damages. Proc. of the Berkeley Conference, in honor of Jerzy Neyman and Jack Kiefer. Vol I pp. 123-172. (L. Le Cam, R. Olshen eds.) Wadsworth, 1985.

24. DOKSUM, K. and YANDELL, B. (April 1983). Tests for exponentiality. Handbook of Statistics, (P. R. Krishnaiah and P. K. Sen, eds.) 4, 1984, 579-611.

25. FREEDMAN, D. A. (May 1983). Comments on a paper by Markus.

26. FREEDMAN, D. (Oct. 1983, revised March 1984). On bootstrapping two-stage least-squares estimates in stationary linear models. Ann. Statist., 1984, 12, 827-842.

27. DOKSUM, K. A. (Dec. 1983). An extension of partial likelihood methods for proportional hazard models to general transformation models. Ann. Statist., 1987, 15, 325-345.

28. BICKEL, P. J., GOETZE, F. and VAN ZWET, W. R. (Jan. 1984). A simple analysis of third order efficiency of estimate Proc. of the Neyman-Kiefer Conference, (L. Le Cam, ed.) Wadsworth, 1985.

29. BICKEL, P. J. and FREEDMAN, D. A. Asymptotic normality and the bootstrap in stratified sampling. Ann. Statist. 12 470-482.

30. FREEDMAN, D. A. (Jan. 1984). The mean vs. the median: a case study in 4-R Act litigation. JBES. 1985 Vol 3 pp. 1-13.

31. STONE, C. J. (Feb. 1984). An asymptotically optimal window selection rule for kernel density estimates. Ann. Statist., Dec. 1984, 12, 1285-1297.

32. BREIMAN, L. (May 1984). Nail finders, edifices, and Oz.

33. STONE, C. J. (Oct. 1984). Additive regression and other nonparametric models. Ann. Statist., 1985, 13, 689-705.

34. STONE, C. J. (June 1984). An asymptotically optimal histogram selection rule. Proc. of the Berkeley Conf. in Honor of Jerzy Neyman and Jack Kiefer (L. Le Cam and R. A. Olshen, eds.), II, 513-520.

35. FREEDMAN, D. A. and NAVIDI, W. C. (Sept. 1984, revised Jan. 1985). Regression models for adjusting the 1980 Census. Statistical Science. Feb 1986, Vol. 1, No. 1, 3-39.

36. FREEDMAN, D. A. (Sept. 1984, revised Nov. 1984). De Finetti's theorem in continuous time.

37. DIACONIS, P. and FREEDMAN, D. (Oct. 1984). An elementary proof of Stirling's formula. Amer. Math Monthly. Feb 1986, Vol. 93, No. 2, 123-125.

38. LE CAM, L. (Nov. 1984). Sur l'approximation de familles de mesures par des familles Gaussiennes. Ann. Inst. Henri Poincaré, 1985, 21, 225-287.

39. DIACONIS, P. and FREEDMAN, D. A. (Nov. 1984). A note on weak star uniformities.

40. BREIMAN, L. and IHAKA, R. (Dec. 1984). Nonlinear discriminant analysis via SCALING and ACE.

41. STONE, C. J. (Jan. 1985). The dimensionality reduction principle for generalized additive models.

42. LE CAM, L. (Jan. 1985). On the normal approximation for sums of independent variables.

43. BICKEL, P. J. and YAHAV, J. A. (1985). On estimating the number of unseen species: how many executions were there?

44. BRILLINGER, D. R. (1985). The natural variability of vital rates and associated statistics. Biometrics, to appear.

45. BRILLINGER, D. R. (1985). Fourier inference: some methods for the analysis of array and nonGaussian series data. Water Resources Bulletin, 1985, 21, 743-756.

46. BREIMAN, L. and STONE, C. J. (1985). Broad spectrum estimates and confidence intervals for tail quantiles.

47. DABROWSKA, D. M. and DOKSUM, K. A. (1985, revised March 1987). Partial likelihood in transformation models with censored data. Scandinavian J. Statist., 1988, 15, 1-23.

48. HAYCOCK, K. A. and BRILLINGER, D. R. (November 1985). LIBDRB: A subroutine library for elementary time series analysis.

49. BRILLINGER, D. R. (October 1985). Fitting cosines: some procedures and some physical examples. Joshi Festschrift, 1986. D. Reidel.

50. BRILLINGER, D. R. (November 1985). What do seismology and neurophysiology have in common? - Statistics! Comptes Rendus Math. Rep. Acad. Sci. Canada. January, 1986.

51. COX, D. D. and O'SULLIVAN, F. (October 1985). Analysis of penalized likelihood-type estimators with application to generalized smoothing in Sobolev Spaces.

52. O'SULLIVAN, F. (November 1985). A practical perspective on ill-posed inverse problems: A review with some new developments. To appear in Journal of Statistical Science.

53. LE CAM, L. and YANG, G. L. (November 1985, revised March 1987). On the preservation of local asymptotic normality under information loss.

54. BLACKWELL, D. (November 1985). Approximate normality of large products.

55. FREEDMAN, D. A. (June 1987). As others see us: A case study in path analysis. Journal of Educational Statistics. 12, 101-128.

56. LE CAM, L. and YANG, G. L. (January 1986). Replaced by No. 68.

57. LE CAM, L. (February 1986). On the Bernstein - von Mises theorem.

58. O'SULLIVAN, F. (January 1986). Estimation of Densities and Hazards by the Method of Penalized likelihood.

59. ALDOUS, D. and DIACONIS, P. (February 1986). Strong Uniform Times and Finite Random Walks.

60. ALDOUS, D. (March 1986). On the Markov Chain simulation Method for Uniform Combinatorial Distributions and Simulated Annealing.

61. CHENG, C-S. (April 1986). An Optimization Problem with Applications to Optimal Design Theory.

62. CHENG, C-S., MAJUMDAR, D., STUFKEN, J. & TURE, T. E. (May 1986, revised Jan 1987). Optimal step type design for comparing test treatments with a control.

63. CHENG, C-S. (May 1986, revised Jan. 1987). An Application of the Kiefer-Wolfowitz Equivalence Theorem.

64. O'SULLIVAN, F. (May 1986). Nonparametric Estimation in the Cox Proportional Hazards Model.

65. ALDOUS, D. (JUNE 1986). Finite-Time Implications of Relaxation Times for Stochastically Monotone Processes.

66. PITMAN, J. (JULY 1986, revised November 1986). Stationary Excursions.

67. DABROWSKA, D. and DOKSUM, K. (July 1986, revised November 1986). Estimates and confidence intervals for median and mean life in the proportional hazard model with censored data. Biometrika, 1987, 74, 799-808.

68. LE CAM, L. and YANG, G.L. (July 1986). Distinguished Statistics, Loss of information and a theorem of Robert B. Davies (Fourth edition).

69. STONE, C.J. (July 1986). Asymptotic properties of logspline density estimation.

71. BICKEL, P.J. and YAHAV, J.A. (July 1986). Richardson Extrapolation and the Bootstrap.

72. LEHMANN, E.L. (July 1986). Statistics - an overview.

73. STONE, C.J. (August 1986). A nonparametric framework for statistical modelling.

74. BIANE, PH. and YOR, M. (August 1986). A relation between Lévy's stochastic area formula, Legendre polynomial, and some continued fractions of Gauss.

75. LEHMANN, E.L. (August 1986, revised July 1987). Comparing Location Experiments.

76. O'SULLIVAN, F. (September 1986). Relative risk estimation.

77. O'SULLIVAN, F. (September 1986). Deconvolution of episodic hormone data.

78. PITMAN, J. & YOR, M. (September 1987). Further asymptotic laws of planar Brownian motion.

79. FREEDMAN, D.A. & ZEISEL, H. (November 1986). From mouse to man: The quantitative assessment of cancer risks. Statistical Science, 1988, Vol. 3 No. 1, 3-56.

80. BRILLINGER, D.R. (October 1986). Maximum likelihood analysis of spike trains of interacting nerve cells.

81. DABROWSKA, D.M. (November 1986). Nonparametric regression with censored survival time data.

82. DOKSUM, K.J. and LO, A.Y. (Nov 1986, revised Aug 1988). Consistent and robust Bayes Procedures for Location based on Partial Information.

83. DABROWSKA, D.M., DOKSUM, K.A. and MIURA, R. (November 1986). Rank estimates in a class of semiparametric two-sample models.

84. BRILLINGER, D. (December 1986). Some statistical methods for random process data from seismology and neurophysiology.

85. DIACONIS, P. and FREEDMAN, D. (December 1986). A dozen de Finetti-style results in search of a theory. Ann. Inst. Henri Poincaré, 1987, 23, 397-423.

86. DABROWSKA, D.M. (January 1987). Uniform consistency of nearest neighbour and kernel conditional Kaplan - Meier estimates.

87. FREEDMAN, D.A., NAVIDI, W. and PETERS, S.C. (February 1987). On the impact of variable selection in fitting regression equations. T.K. Dijkstra, ed. On Model Uncertainty and Its Statistical Implications. Lecture Notes in Econometric and Mathematical Systems, No. 307 Springer, Berlin pp.1-16.

88. ALDOUS, D. (February 1987, revised April 1987). Hashing with linear probing, under non-uniform probabilities.

89. DABROWSKA, D.M. and DOKSUM, K.A. (March 1987, revised January 1988). Estimating and testing in a two sample generalized odds rate model. J. Amer. Statist. Assoc., 1988, 83, 744-749.

90. DABROWSKA, D.M. (March 1987). Rank tests for matched pair experiments with censored data.

91. DIACONIS, P and FREEDMAN, D.A. (April 1988). Conditional limit theorems for exponential families and finite versions of de Finetti's theorem. Journal of Theoretical Probability, Oct. 1988, Vol. 1 No. 4, 381-410.

92. DABROWSKA, D.M. (April 1987, revised September 1987). Kaplan-Meier estimate on the plane.

92a. ALDOUS, D. (April 1987). The Harmonic mean formula for probabilities of Unions: Applications to sparse random graphs.

93. DABROWSKA, D.M. (June 1987, revised Feb 1988). Nonparametric quantile regression with censored data.

94. DONOHO, D.L. & STARK, P.B. (June 1987). Uncertainty principles and signal recovery. SIAM J. Appl. Math., June, 1989.

95. CANCELLED

96. BRILLINGER, D.R. (June 1987). Some examples of the statistical analysis of seismological data. To appear in Proceedings, Centennial Anniversary Symposium, Seismographic Stations, University of California, Berkeley.

97. FREEDMAN, D.A. and NAVIDI, W. (June 1987). On the multi-stage model for carcinogenesis. Environmental Health Perspectives, 1989, 81, 169-188.

98. O'SULLIVAN, F. and WONG, T. (June 1987). Determining a function diffusion coefficient in the heat equation.

99. O'SULLIVAN, F. (June 1987). Constrained non-linear regularization with application to some system identification problems.

100. LE CAM, L. (July 1987, revised Nov 1987). On the standard asymptotic confidence ellipsoids of Wald.

101. DONOHO, D.L. and LIU, R.C. (July 1987). Pathologies of some minimum distance estimators. Annals of Statistics, June, 1988.

102. BRILLINGER, D.R., DOWNING, K.H. and GLAESER, R.M. (July 1987). Some statistical aspects of low-dose electron imaging of crystals.

103. LE CAM, L. (August 1987). Harald Cramér and sums of independent random variables.

104. DONOHO, A.W., DONOHO, D.L. and GASKO, M. (August 1987). Macspin: Dynamic graphics on a desktop computer. IEEE Computer Graphics and applications, June, 1988.

105. DONOHO, D.L. and LIU, R.C. (August 1987). On minimax estimation of linear functionals.

106. DABROWSKA, D.M. (August 1987). Kaplan-Meier estimate on the plane: weak convergence, LIL and the bootstrap.

107. CHENG, C-S. (Aug 1987, revised Oct 1988). Some orthogonal main-effect plans for asymmetrical factorials.

108. CHENG, C-S. and JACROUX, M. (August 1987). On the construction of trend-free run orders of two-level factorial designs.

109. KLASS, M.J. (August 1987). Maximizing $E \max_{1 \le k \le n} S_k^+ / ES_n^+$: A prophet inequality for sums of I.I.D. mean zero variates.

110. DONOHO, D.L. and LIU, R.C. (August 1987). The "automatic" robustness of minimum distance functionals. Annals of Statistics, June, 1988.

111. BICKEL, P.J. and GHOSH, J.K. (August 1987, revised June 1988). A decomposition for the likelihood ratio statistic and the Bartlett correction — a Bayesian argument.

112. BURDZY, K., PITMAN, J.W. and YOR, M. (September 1987). Some asymptotic laws for crossings and excursions.

113. ADHIKARI, A. and PITMAN, J. (September 1987). The shortest planar arc of width 1.

114. RITOV, Y. (September 1987). Estimation in a linear regression model with censored data.

115. BICKEL, P.J. and RITOV, Y. (Sept. 1987, revised Aug 1988). Large sample theory of estimation in biased sampling regression models I.

116. RITOV, Y. and BICKEL, P.J. (Sept.1987, revised Aug. 1988). Achieving information bounds in non and semiparametric models.

117. RITOV, Y. (October 1987). On the convergence of a maximal correlation algorithm with alternating projections.

118. ALDOUS, D.J. (Oct. 1987, revised Aug. 1990). Meeting times for independent Markov chains.

119. HESSE, C.H. (Oct. 1987, revised Nov. 1990). An asymptotic expansion for the mean of the passage-time distribution of integrated Brownian Motion.

120. DONOHO, D. and LIU, R. (Oct. 1987, revised Mar. 1988, Oct. 1988). Geometrizing rates of convergence, II.

121. BRILLINGER, D.R. (October 1987). Estimating the chances of large earthquakes by radiocarbon dating and statistical modelling. Statistics a Guide to the Unknown, pp. 249-260 (Eds. J.M. Tanur et al.) Wadsworth, Pacific Grove.

122. ALDOUS, D., FLANNERY, B. and PALACIOS, J.L. (November 1987). Two applications of urn processes: The fringe analysis of search trees and the simulation of quasi-stationary distributions of Markov chains.

123. DONOHO, D.L., MACGIBBON, B. and LIU, R.C. (Nov.1987, revised July 1988). Minimax risk for hyperrectangles. *Ann. Statist.* **18**, 1416-1437, (1990).

124. ALDOUS, D. (November 1987). Stopping times and tightness II.

125. HESSE, C.H. (Nov. 1987, revised Sept. 1990). Modelling Sedimentation.

126. DALANG, R.C. (December 1987, revised June 1988). Optimal stopping of two-parameter processes on nonstandard probability spaces.

127. Same as No. 133.

128. DONOHO, D. and GASKO, M. (December 1987). Multivariate generalizations of the median and trimmed mean II.

129. SMITH, D.L. (December 1987). Exponential bounds in Vapnik-Červonenkis classes of index 1.

130. STONE, C.J. (Nov.1987, revised Sept. 1988). Uniform error bounds involving logspline models.

131. Same as No. 140

132. HESSE, C.H. (Dec. 1987, revised June 1989). A Bahadur - Type representation for empirical quantiles of a large class of stationary, possibly infinite - variance, linear processes. *Ann. Statist.* 18, 1188-1202.

133. DONOHO, D.L. and GASKO, M. (December 1987). Multivariate generalizations of the median and trimmed mean, I.

134. CANCELLED

135. FREEDMAN, D.A. and NAVIDI, W. (December 1987). Ex-smokers and the multistage model for lung cancer. Epidemiology, Jan. 1990, Vol. 1 No. 1, 21-29

136. LE CAM, L. (January 1988). On some stochastic models of the effects of radiation on cell survival.

137. DIACONIS, P. and FREEDMAN, D.A. (April 1988). On the uniform consistency of Bayes estimates for multinomial probabilities. To appear in Ann. Statist.

137a. DONOHO, D.L. and LIU, R.C. (1987). Geometrizing rates of convergence, I.

138. DONOHO, D.L. and LIU, R.C. (Jan. 1988, revised Jan 1990). Geometrizing rates of convergence, III.

139. BERAN, R. (January 1988). Refining simultaneous confidence sets.

140. HESSE, C.H. (Dec. 1987, revised Oct. 1990). Statistical aspects of neural networks.

141. BRILLINGER, D.R. (Jan. 1988). Two reports on trend analysis: a) An elementary trend analysis of Rio negro levels at Manaus, 1903-1985. b) Consistent detection of a monotonic trend superposed on a stationary time series.

142. DONOHO, D.L. (Jan. 1985, revised Jan. 1988). One-sided inference about functionals of a density. Ann. Stat., December, 1988.

143. DALANG, R.C. (Feb. 1988, revised Nov. 1988). Randomization in the two-armed bandit problem.

144. DABROWSKA, D.M., DOKSUM, K.A. and SONG, J.K. (February 1988). Graphical comparisons of cumulative hazards for two populations.

145. ALDOUS, D.J. (February 1988). Lower bounds for covering times for reversible Markov Chains and random walks on graphs.

146. BICKEL, P.J. and RITOV, Y. (Feb.1988, revised August 1988). Estimating integrated squared density derivatives.

147. STARK, P.B. (March 1988). Strict bounds and applications.

148. DONOHO, D.L. and STARK, P.B. (March 1988). Rearrangements and smoothing.

149. NOLAN, D. (Sept. 1989, revised Oct. 1990). On min-max majority and deepest points.

150. SEILLIER, F. (March 1988). Sequential probability forecasts and the probability integral transform.

151. NOLAN, D. (Mar. 1988, revised Feb. 1990). Asymptotics for multivariate trimming.

152. DIACONIS, P. and FREEDMAN, D.A. (April 1988). Cauchy's equation and de Finetti's theorem. To appear in Scand. J. Statist.

153. DIACONIS, P. and FREEDMAN, D.A. (April 1988). On the problem of types.

154. DOKSUM, K.A. and GASKO, M. (May 1988). On the correspondence between models in binary regression analysis and survival analysis.

155. LEHMANN, E.L. (May 1988). Jerzy Neyman, 1894-1981.

156. ALDOUS, D.J. (May 1988). Stein's method in a two-dimensional coverage problem.

157. FAN, J. (June 1988). On the optimal rates of convergence for nonparametric deconvolution problem.

158. DABROWSKA, D. (June 1988). Signed-rank tests for censored matched pairs.

159. BERAN, R.J. and MILLAR, P.W. (June 1988). Multivariate symmetry models.

160. BERAN, R.J. and MILLAR, P.W. (June 1988). Tests of fit for logistic models.

161. BREIMAN, L. and PETERS, S. (June 1988, revised March 1990). Comparing automatic smoothers (A public service enterprise).

162. FAN, J. (June 1988). Optimal global rates of convergence for nonparametric deconvolution problem.

163. Cancelled

164. BICKEL, P.J. and KRIEGER, A.M. (July 1988). Confidence bands for a distribution function using the bootstrap.

165. HESSE, C.H. (July 1988, revised Sept. 1990). On a class of stable process models in economics.

166. FAN, JIANQING (July 1988). Nonparametric estimation of quadratic functionals in Gaussian white noise.

167. BREIMAN, L., STONE, C.J. and KOOPERBERG, C. (August 1988). Confidence bounds for extreme quantiles.

168. LE CAM, L. (Aug. 1988, revised Jan. 1990). Maximum likelihood an introduction.

169. BREIMAN, L. (Aug.1988, revised Dec. 1990). The little bootstrap and other methods for dimensionality selection in regression: X-fixed prediction error.

170. LE CAM, L. (September 1988). On the Prokhorov distance between the empirical process and the associated Gaussian bridge.

171. STONE, C.J. (September 1988). Large-sample inference for logspline models.

172. ADLER, R.J. and EPSTEIN, R. (September 1988). Intersection local times for infinite systems of planar brownian motions and for the brownian density process.

173. MILLAR, P.W. (October 1988). Optimal estimation in the non-parametric multiplicative intensity model.

174. YOR, M. (October 1988). Interwinings of Bessel processes.

175. ROJO, J. (October 1988). On the concept of tail-heaviness.

176. ABRAHAMS, D.M. and RIZZARDI, F. (September 1988). BLSS - The Berkeley interactive statistical system: An overview.

177. MILLAR, P.W. (October 1988). Gamma-funnels in the domain of a probability, with statistical implications.

178. DONOHO, D.L. and LIU, R.C. (October 1988). Hardest one-dimensional subproblems.

179. DONOHO, D.L. and STARK, P.B. (October 1988). Recovery of sparse signal when the low frequency information is missing.

180. FREEDMAN, D.A. and PITMAN, J.A. (Nov. 1988). A measure which is singular and uniformly locally uniform. To appear in Proc. Amer. Math Soc.

181. DOKSUM, K.A. and HOYLAND, ARNLJOT (Nov. 1988, revised Jan. 1989, Aug. '89). Models for variable stress accelerated life testing experiments based on Wiener processes and the inverse Gaussian distribution.

182. DALANG, R.C., MORTON, A. and WILLINGER, W. (November 1988). Equivalent martingale measures and no-arbitrage in stochastic securities market models.

183. BERAN, R. (November 1988). Calibrating prediction regions.

184. BARLOW, M.T., PITMAN, J. and YOR, M. (Feb. 1989). On Walsh's Brownian Motions.

185. DALANG, R.C. and WALSH, J.B. (Dec. 1988). Almost-equivalence of the germ-field Markov property and the sharp Markov property of the Brownian sheet.

186. Cancelled.

187. NEVEU, J. and PITMAN, J.W. (Feb. 1989). Renewal property of the extrema and tree property of the excursion of a one-dimensional brownian motion.

188. NEVEU, J. and PITMAN, J.W. (Feb. 1989). The branching process in a brownian excursion.

189. PITMAN, J.W. and YOR, M. (Nov. 1990). Arcsin laws and interval partitions derived from a stable subordinator.

190. STARK, P.B. (Dec. 1988). Duality and discretization in linear inverse problems.

191. LEHMANN, E.L. and SCHOLZ, F.W. (Jan. 1989). Ancillarity.

192. PEMANTLE, R. (Feb. 1989). A time-dependent version of Pólya's urn.

193. PEMANTLE, R. (Feb. 1989). Nonconvergence to unstable points in urn models and stochastic approximations.

194. PEMANTLE, R. (Feb. 1989, revised May 1989). When are touchpoints limits for generalized Pólya urns.

195. PEMANTLE, R. (Feb. 1989). Random walk in a random environment and first-passage percolation on trees.

196. BARLOW, M., PITMAN, J. and YOR, M. (Feb. 1989). Une extension multidimensionnelle de la loi de l'arc sinus.

197. BREIMAN, L. and SPECTOR, P. (Mar. 1989, revised June 1990). Submodel selection and evaluation in regression — the X-random case.

198. BREIMAN, L., TSUR, Y. and ZEMEL, A. (Mar. 1989). A simple estimation procedure for censored regression models with known error distribution.

199. BRILLINGER, D.R. (Mar. 1989). Two papers on bilinear systems: a) A study of second- and third-order spectral procedures and maximum likelihood identification of a bilinear system. b) Some statistical aspects of NMR spectroscopy, *Actas del 2° congreso lantinoamericano de probabilidad y estadistica matematica*, Caracas, 1985.

200. BRILLINGER, D.R. (Mar. 1989). Two papers on higher-order spectra: a) Parameter estimation for nonGaussian processes via second and third order spectra with an application to some endocrine data. b) Some history of the study of higher-order moments and spectra.

201. DE LA PENA, V. and KLASS, M.J. (April 1989). L bounds for quadratic forms of independent random variables.

202. FREEDMAN, D.A. and NAVIDI, W.C. (April 1989). Testing the independence of competing risks.

203. TERDIK, G. (May 1989). Bilinear state space realization for polynomial stochastic systems.

204. DONOHO, D.L. and JOHNSTONE, I.M. (May 1989). Minimax risk over $l_p$-Balls.

205. PEMANTLE, R., PROPP, J. and ULLMAN, D. (May 1989). On tensor powers of integer programs.

206. MILASEVIC, P. and NOLAN, D. (May 1989). Estimation on the sphere: A geometric approach.

207. SPEED, T.P. and YU, B. (July 1989, rev. Oct. 1989). Stochastic complexity and model selection: normal regression.

208. DUBINS, L.E. (June 1989). A group decision device: Its pareto-like optimality.

209. BREIMAN, L. (July 1989, revised Aug. 1990). Fitting additive models to regression data: Diagnostics and alternative views.

210. PEMANTLE, R. (July 1989). Vertex-reinforced random walk.

211. LE CAM, L. (August 1989). On measurability and convergence in distribution.

212. FELDMAN, R.E. (July 1989). Autoregressive processes and first-hit probabilities for randomized random walks.

213. DONOHO, D.L., JOHNSTONE, I.M., HOCH, J.C. and STERN, A.S. (August 1989). Maximum entropy and the nearly black object.

214. DONOHO, D.L. (Aug. 1989, revised Sept., Nov. 1989). Statistical estimation and optimal recovery.

215. STONE, C. (Aug. 1989, revised Dec. 1990). Asymptotics for doubly-flexible logspline response models.

216. NOLAN, D. (August 1989). The excess mass ellipsoid.

217. FREEDMAN, D.A. (August 1989). Statistical models and shoe leather.

218. BICKEL, P.J., NAIR, V.N. and WANG, P.C.C. (August 1989). Nonparametric inference under biased sampling from a finite population.

219. DALANG, R.C. and WALSH, J.B. (September 1989). The sharp Markov property of the brownian sheet and related processes.

220. BURMAN, P. and NOLAN, D. (Oct. 1989). Location-adaptive density estimation and nearest-neighbor distance.

221. LOW, M. (Oct. 1989). Non-existence of an adaptive estimator for the value of an unknown probability density.

222. LOW, M.G. (Oct. 1989). Invariance and rescaling of infinite dimensional Gaussian shift experiments.

223. LOW, M.G. (Oct. 1989). Lower bounds for the integrated risk in nonparametric density and regression estimation.

224 BREIMAN, L. and CUTLER, A. (Oct. 1989). A deterministic algorithm for global optimization.

225. LOW, M.G. (Oct. 1989). Local convergence of nonparametric density estimation problems to Gaussian shift experiments on a Hilbert space.

226. LEHMANN, E.L. (Oct. 1989). Model specification: The views of Fisher and Neyman, and later developments.

227. GOLUBEV, G.K. and NUSSBAUM, M. (Nov. 1989). A risk bound in Sobolev class regression.

228. STARK, P.B. (Dec. 1989). Rigorous computer solutions of infinite-dimensional inverse problems.

229. EVANS, S.N. and PERKINS, E. (December 1989). Measure-valued Markov branching processes conditioned on non-extinction.

230. EVANS, S.N. (December 1989). The entrance space of a measure-valued Markov branching process conditioned on non-extinction.

231. BREIMAN, L. (Dec. 1989, revised July 1990). The $\Pi$-method for estimating multivariate functions from noisy data.

232. SHAFFER, J.P. (December 1989). Probability of directional errors with disordinal (qualitative) interaction.

233. BRODSKY, M. and PANAKHOV, E. (January 1990). Concerning a priori estimates of solution of the inverse logarithmic potential problem.

234. SHAFFER, J.P. (January 1990). The Gauss-Markov theorem and random regressors.

235. SPLAWA-NEYMAN, J. (January 1990). On the application of probability theory to agricultural experiments. Essay on principles by Jerzy Splawa-Neyman. (Roczniki Nauk. Bol. Tom. X (1922)). Translation from the Polish original of §9 pp.29-42 by D.M. Dabrowska, edited by T.P. Speed.

236. DONOHO, D.L. and NUSSBAUM, M. (January 1990). Minimax quadratic estimation of a quadratic functional.

237. DONOHO, D.L., JOHNSTONE, I.M., STERN, A.S. and HOCH, J.C. (January 1990). Does the maximum entropy method improve sensitivity? *Proc. Nat. Acad. Sci. USA* **87**, 5066-5068, July 1990.

238. KOOPERBERG, C. and STONE, C.J. (Feb. 1990, revised Dec. 1990). A study of logspline density estimation.

239. HESSE, C.H. (March 1990). Hitting-time densities of a two-dimensional Markov process.

240. EVANS, S.N. (March 1990). Trapping a measure-valued Markov branching process conditioned on non-extinction.

241. YU, B. and SPEED, T.P. (March 1990). Stochastic complexity and model selection II. Histograms.

242. BICKEL, P.J. and MILLAR, P.W. (March 1990). Uniform convergence of probability measures on classes of functions.

243. DALANG, R.C. and WALSH, J.B. (April 1990). The sharp Markov property of Lévy sheets.

244. BRILLINGER, D.R. (March 1990). Two reports on the analysis of spatially aggregate data:
a) Mapping aggregate birth data
b) Spatial-temporal modelling of spatially aggregate birth data.

245. ZHANG, P. (April 1990). Variable selection in non-parametric regression with continuous covariates.

246. BICKEL, P.J. and ZHANG, P. (April 1990). Variable selection in non-parametric regression with categorical covariates.

247. EVANS, S.N. (April 1990). Association and infinite divisibility for the wishart distribution and its diagonal marginals.

248. FREEDMAN, D.A., KLEIN, S., SACKS, J., EVERETT, C. and SMYTH, C. (April 1990). Ecological regression and voting rights.

249. DONOHO, D.L. and LOW, M.G. (May 1990). Renormalization exponents and optimal pointwise rates of convergence.

250. HESSE, C.H. (May 1990). The one-sided barrier problem for an integrated Ornstein-Uhlenbeck process.

251. HESSE, C.H. (May 1990). Rates of convergence for the empirical distribution function and the empirical characteristic function of a broad class of linear processes. To appear - *J. Multivariate Anal.* (Nov. 1990).

252. EVANS, S.N. (May 1990). Equivalence and perpendicularity of local field Gaussian measures.

253. SPEED, T.P. (May 1990). Introduction to "The Arrangement of Field Experiments" by R.A. Fisher ( J. Min. Agric. Gr. Br., 33, 503-513; Collected Papers II, 38: 83-94).

254. BRODSKY, M. (May 1990). Physically meaningful results on the uniqueness of inversion of Newtonian potential.

255. BICKEL, P.J. and RITOV, Y. (May 1990). Estimating linear functionals of a PET image.

256. BICKEL, P.J., RITOV, Y. and WELLNER, J.A. (May 1990). Efficient estimation of linear functionals of a probability measure P with known marginal distributions.

257. ZHANG, P. (June 1990). Model selection via multi-fold cross validation.

258. BICKEL, P.J. and FAN, J. (June 1990). Some problems on the estimation of densities under shape restrictions.

259. DONOHO, D.L. and LOW, M.G. (June 1990). White noise approximation for sampled data.

260. YU, B. (June 1990). Rates of convergence and central limit theorems for empirical processes of stationary mixing sequences.

261. STARK, P.B. (July 1990). Inference in infinite dimensional inverse problems: Discretization and duality.

262. EVANS, S.N. (August 1990). Polar and non-polar sets for a tree indexed process.

263. LE CAM, L. (August 1990). Some recent results in the asymptotic theory of statistical estimation.

264. BARLOW, M.T., EVANS, S.N., and PERKINS, E.A. (August 1990). Collision local times and measure-valued processes.

265. LE CAM, L. (August 1990). Some special results of measure theory.

266. BASAK, G.K. (August 1990). A class of limit theorems for singular diffusions.

267. BASAK, G.K. and BHATTACHARYA, R.N. (August 1990). Stability in distribution for a class of singular diffusions.

268. STONE, C.J. (August 1990). $L_2$ rate of convergence for interaction spline regression.

269. LE CAM, L. (September 1990). An infinite dimensional convolution theorem.

270. BICKEL, P.J., RITOV, Y. and WELLNER, J.A. (Sept. 1990). Efficient estimation of linear functionals of a probability measure P with known marginal distributions.

271. DONOHO, D.L. and LOGAN, B.F. (September 1990). Signal recovery and the large sieve.

272. BICKEL, P.J. (October 1990). Some theory for the stringer bound of auditing practice.

273. BICKEL, P.J. (October 1990). Theoretical comparison of bootstrap t confidence bounds.

274. HESSE, C.H. (October 1990). Further refinements of a stochastic model for particle sedimentation in fluids.

275. LOW, M.G. (October 1990). Renormalizing upper and lower bounds for integrated risk in the white noise model.

276. KLEIN, S., SACKS, J. and FREEDMAN, D.A. (October 1990). Ecological regression versus the secret ballot.

277. GUERRA, R. and SPEED, T.P. (November 1990). Statistical methods for analyzing DNA-DNA hybridization data.

278. LEHMANN, E.L. and ROJO, J. (November 1990). Invariant location orderings.

279. FREEDMAN, D.A. (December 1990). The 1990 census and statistical adjustment.

Copies of these Reports plus the most recent additions to the Technical Report series are available from the Statistics Department technical typist in room 379 Evans Hall or may be requested by mail from:

Department of Statistics
University of California
Berkeley, California 94720

Cost: $1 per copy.