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 viscous 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.


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).
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.


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

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

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

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

for some kernel \( K: \mathbb{R}^3 \to [0,\infty] \). \( K \) 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

\[
(2.1) \quad m \, dV = m \, g - \tau \, V \, dW
\]

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:

**Assumption:** The transition densities $f_c$ depend continuously on $c$, i.e.

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

Hence, $f_c$ is an equicontinuous family on $V \times V$, 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, etc.

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

\begin{equation}
X_i(t) = \int_0^t V_i(s) \, ds
\end{equation}
is the displacement process and

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

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 \(T(x)\):

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

$$\tilde{f}_{T(x)}(t) = [\frac{3x - (v + \mu)t}{2t} - \frac{\tau}{8}(3(x - \mu t) - vt)]\phi(x)$$

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

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

\(\hat{f}_{T(x)}(t)\) is accurate as an approximation to \(f_{T(x)}(t)\) in the limit as the boundary becomes increasingly remote. Both \(\tilde{f}_{T(x)}^*(t)\) 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 \(\tilde{f}_{T(x)}^*(t)\), 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 \(B\)

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

with \(\delta = 2\pi^{-1/2}\sigma^{-2}\mu x\).

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

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

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

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

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 $\sigma^2$.


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,

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

with

$$c(x, t) = \int_{\mathcal{D}} K(x - x') P(x', t) \, dx'$$

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

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

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

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

%nlolist
%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, covvy, corrvy, corrvy1, corrvy2, fill, xx
real*4 yy(1000), v(1000), ov(1000)
real*4 vv(1000), con(1000), sv(1000)
real*4 sdx, rmeanx, corrxv, gsdx, rmu
real*4 bx, cx, vwidth, fk
integer*2 scx(1000), scy(1000), oscx(1000), center(2)
integer*2 ooscx(1000), ooscy(1000)
integer*2 rad
integer*4 in1(1000), in2(1000)
integer*4 in3(1000), status, icol
integer*4 ntime, i, j, m, nd, icoll, 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
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=129)
read(9, *) ntime, npart, xscll, xscl2, yscl, ovlength, vwidth
read(9, *) t, incon, pack, fk, xx, 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, icoll, icol2, ndraw, nhi
read(9, 1000) (xo(i), yy(i), con(i), ocy(i),
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
read(6,889)((coef(j,m),m=1,mdims),j=1,jdims)
read(6,888)(rknots(j),j=1,jdims)
close(6)
888 format(gl6.8)
889 format(4g16.8)
c read the spline coefficients
for more info cf. De Boor,
A practical Guide to splines,
Springer-Verlag
c read(6,889)((coef(j,m),m=1,mdims),j=1,jdims)
read(6,888)(rknots(j),j=1,jdims)
close(6)
format(g16.8)
format(4g16.8)
c get graphics info
open(unit=7,file='//stat600/stat/ramos/pick/simul/data/barf3') {apollo}
read(7,*),ban,rad,icoll,icol2,ndraw,nhi
close(7)
ndraw=min(npart,ndraw)
nhi=min(min(nhi,ndraw),npart)
write(3,*),'number of highlighted particles ',nhi
if (rad.1t.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
   CALL PARAMET(1., RRMU, GSD, C, B, SD)

C THEN IN THE X-AXIS
   CALL PARAMET(1., RRMU, GSDX, CX, BX, SDX)
   SDX = SDX*XX
   GSDX = GSDX*XX
   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
      GET INITIAL VELOCITIES
      DO J=1, NPART
         CON(J) = 1.
         V(J) = V(J)*SD+RRMU
         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) = XSC1 + INT2(XO(J)*XSC2/VWIDTH)
         SCY(J) = INT2(YY(J)*YSC1/OVLENGTH)
         CENTER(1) = OSCX(J)
         CENTER(2) = SCY(J)
         ICOL = IC0L
         IF (J.LE.NDRAW-NHI) ICOL = IC0L2
         CALL GPR_$SET_FILL_VALUE(ICOL, STATUS)
         CALL GPR_$CIRCLE_FILLED(CENTER, RAD, STATUS)
         OSSCX(J) = OSCX(J)
         OSSY(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, VXX)
      CALL GGNPM(SEED, NPART, XX)

C CALCULATIONS FOR NEW VIEW
   NDEAD = 0

   DO 20 I=1, NPART

   IF LAST PARTICLE POSITION EXCEEDS CURRENT VESSEL LENGTH
      IF(YY(I) .GE. VLENGTH) THEN
         MEANS PARTICLE IS PART OF THE PACKED BED
         NDEAD = NDEAD + 1
      END IF

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

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

next get the actual variables; first get position increment
by first: multiplying N(0,1) variable by appropriate sd,

yx(i)=sdy*(corryv*vv(i)+sqrt(1-corryv**2)*yx(i))
xx(i)=sdx*(corrxv*vvx(i)+sqrt(1-corrxxv**2)*xx(i))

yy(i)=min(yy(i)+yx(i)+rmu*t+(v(i)-rmu)/b*(1-c),vlength)
ox(i)=xo(i)+xx(i)+vx(i)/b*(1-c)

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

rmean=rmu+(v(i)-rmu)*c
rmeanx=vx(i)*c
xo(i)=xo(i)+rmean/rrmu*xx(i)+vx(i)/bx*(1-cx)

rmeanx=vx(i)*c
vv(i)=gsd*vv(i)+rmean
vvx(i)=gsd*xv(i)+rmeanx

end if

scx(i)=mod(int2(xo(i)*xscl2/vwidth+ll*xscl2),xscl2)+xscll
scy(i)=int2(yy(i)*yscl/ovlength)

ov(i)=v(i)
v(i)=vv(i)
vx(i)=vvx(i)
oscy(i)=scy(i)
ooscx(i)=scx(i)

vlength=ovlength*(1.-fill*real(ndead)/real(npart))

smoothing
call kernel2(yy,conc,ov,ssv,inl,in2,in3)
call camera_shut
erasing and plotting
f irst the particles
call erase(l)
call clip(l)
do i=1,ndraw

center(1)=ooscx(i)
center(2)=oscy(i)
icol=red
if (i.le.ndraw-nhi)icol=black
call gpr_$set_fill_status(icol,status)
call gpr_$circle_filled(center,rad,status)
center(1)=scx(i)
center(2)=scy(i)
icol=icol1
if (i.le.ndraw-nhi)icol=icol2
call gpr_$set_fill_status(icol,status)
call gpr_$circle_filled(center,rad,status)
scy(i)=scy(i)
ooscx(i)=scx(i)
endo
call clip(2)  
then the curves

  call packed(yellow)
call erase(0)
call axes
call histodraw(conc,ssv,j)
call timer(j,t)
call spielberg
call pause
call legend
cd

get conc and vel for each particle from smooths
notice that for concentration we use the concentration of the bin where the particle is headed for, as given by in3(i), as opposed to the one where the particle currently is, given by in1(i)

do i=1,npart
  con(i)=conc(inl(i))+4*conc(in2(i))+conc(in3(i))
  con(i)=con(i)/6
  ssv(i)=ssv(inl(i))
endo
call gpr_$terminate(dd,status)
cSAVE SIMULATION INFO TO CONTINUE SIMULATING LATER
write(3,*)'do you want to continue later?'
copen(unit=2,file='stdin') {apollo}
cread(2,1001)ans2
cif(ans2.eq.'y') then
  open(unit=8,file='stat600/stat/ramos/pick/simul/data/barf8'
+,'overwrite',recl=128)
cwrite(8,*)ntime,npart,xscll,xscl2,yscl,ovlength,vwidth
cwrite(8,*)incon,pack,fk,xk,h,nbins,seed,vlength
write(8,*)jdims,mdims
write(8,889)((coef(j,m),m=1,mdims),j=1,jdims)
cwrite(8,888)(rknots(j),j=1,jdims)
cwrite(8,*)ban,rad,icoll,icol2,ndraw,nhi
write(8,1000)(xo(i),yy(i),con(i),oscy(i),
+        ooscx(i),ov(i),v(i),vx(i),i=1,npart)
c1000 format(3g16.8,2i4,3g16.8)
close(8)
cend if
999 stop
cend subroutine timer(j,t)
include '//stat600/stat/ramos/pick/simul/src/colorinfo.ftn'
integer*4 j,status
integer*2 xpos,ypos,strl
integer*2 namel,fid
real*4 t
character*8 time
include '//stat600/stat/ramos/pick/simul/src/graf.com.ftn'
include '//stat600/stat/ramos/pick/simul/src/graf2.com.ftn'
cwrite(time,10)(j*t)
c10 format(f8.2)
xpos=xscl1+(28*9)+int2(79)
ypos=yscl+int2(3)*space-int2(10)
call gpr_$set_text_value(cyan,status)
call gpr_$set_text_backround_value(black,status)
call gpr_$move(xpos,ypos,status)
strl=int2(8)
call gpr_$text(time,strl,status)
call gpr_$set_text_background_value(black,status)
return
cend
program simula3

%no list
%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 scon(10000), svcon(10000), oscon(10000)
integer*2 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, ysc1
common /param/ mm, incon, pack, npart, nbins, h, t

c internal parameters

data ntime/10000/
data npart/1000/
data xscl/500/
data ysc1/1020/
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, y)
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

   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

   call g_init

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
            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(0))
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)
osscon(i)=svcon(i)

500 continue
30 continue
stop
end

program simulaG

%list
%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/10000/
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

initialize variables
  do 1 i=1,npart
data(i)=0.
oxo(i)=0.
1  continue

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

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

parametrize initial step
call paramet(l.,rmu,gsd,c,b,sd)

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

initialize graphics
call g_init

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

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

calculations for new view
  do 20 i=1,npart
if(yy(i).ge.500.) then
  yy(i)=500.
vv(i)=0.
sx(i)=oscx(i)
sy(i)=yscl
20 continue
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
  covyv=dble((sd*(1-c))**2/b)
  sdy=sqrt(2*b*t+c*(4-c)-3)*sd/b
  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)

v(i)=vv(i)
oscx(i)=scx(i)

20 continue

c smoothing

  call kernel2(yy,conc,v,ssv,in)
  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),i, ,int4(0))
    call plot(osvcon(i),oscy(i),i, ,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),i, ,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)
    oscon(i)=scon(i)
    osvcon(i)=svcon(i)
  500 continue
program simulaGn

#include '/sys/ins/base.ins.ftn'
#include '/sys/ins/gpr.ins.ftn'
%list

c this program simulates particles settling in a slurry
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), oscy(10000), xscl, yscl
integer*2 scon(10000), svcon(10000), oscon(10000), osvcon(10000)
integer*4 npbins, 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, npbins, h, t

%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, 'particles. Scaling is', xscl, 'for the x-axis, and', yscl, 'for the y-axis.'
write(**)

%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, npbins, ban, ndraw, seed

%initialize variables

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

%get parameter for packed bed

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

%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),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
  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
  next get the actual variables; first get position increment
  by first: multiplying N(0,1) variable by appropriate sd,
  yx(i)=sdy*(corryv*vv(i)+sqrt(1-corryv**2)*yx(i))
  then add the mean to it
  yy(i)=yy(i)+yx(i)+rmu*t+(v(i)-rmu)/b*(1-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
  if(yy(i).lt.0)then
    yy(i)=-yy(i)
    v(i)=0.
  else
    rmean=rmu+(v(i)-rmu)*c
    v(i)=gsd*vv(i)+rmean
  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

smoothing
  call kernel2(yy,conc,ov,ssv,in,in2)
  call erase
  call axes
  call histodraw(conc,ssv)
  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

erasing and plotting
  nd=ndraw
  do 500 i=1,ndraw
    center(1)=oscx(i)
    center(2)=oscy(i)
    call gpr_set_fill_value(1,status)
    call gpr_circle_filled(center,int2(2),status)
    call plot(oscx(i),oscy(i),1,int4(0))
    if(yy(i).ge.500.)yy(i)=500.
    center(1)=scx(i)
    center(2)=scy(i)
    call gpr_set_fill_value(2,status)
    call gpr_circle_filled(center,int2(2),status)
    call plot(scx(i),scy(i),i,int4(7))
    oscy(i)=scy(i)
ooscx(i) = scx(i)
continue
30 continue
stop
end

program simulaGn
%no list
%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,corryvi,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,rd,rmu,rmean,c,pack,gsd,z
real*4 yx(10000),b,coef(20,4),rknots(20)
complex conc(1024),ssv(1024)
real*4 yy(l0000),v(l0000),con(10000)
real*4 yx(l0000),b,coef(20,4),rknots(20)
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

read the spline coefficients
for more info cf. De Boor,
A practical Guide to splines,
Springer-Verlag

read(6,889)((coef(j,m),m=1,mdims),j=1, jdims)
read(6,888)(rknots(j),j=1, jdims)

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(l.,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

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, yx)

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
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 appropriate 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
  if position is negative then set equal to -position and set
  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

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))
c 400 continue
c erasing and plotting
  nd=ndraw
  do 500 i=1,ndraw
    center(1)=oscx(i)
    center(2)=oscy(i)
d  call gpr_$set_fill_value(i,status)
d  call gpr_$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

call gpr_$set_fill_value(icol,status)
call gpr_$circle_filled(center,rad,status)

osci(i)=scy(i)
ooscx(i)=scx(i)

continue
30 continue

stop

end

integer*4 jdims,mdims
real*4 rknots(20),coef(20,4)
common /spline/jdims,mdims,rknots,coef

program simulaGn

this program simulates particles settling in a slurry
using the exact stochastic formula for the increments.

real*8 seed
real*4 sdy,covyv,corryv,corryv1,corryv2,fil
integer*2 scx(10000),scy(10000),oscx(10000),center(2)
integer*2 ooscx(10000),oscy(10000),xsc11,xsc12
integer*2 yscl,yscl2,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 vy(10000),v(10000),ov(10000)
real*4 vv(10000),con(10000),sv(10000)
integer*4 jdims,mdims,ntime,nhi,i,j,m,nd,icoll,icol2,ndead
real*4 sx,ovlength,rmeanx,corrxv,gsdx,romu,rmux
integer*4 nbins,npart,mm,incon,pack,npart,nbins,h,t,romu
common /graf/xscl1,xsc12,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,xsc12,yscl,ovlength,vwidth

write(*,*)'This simulation runs for',ntime,' steps with',npart,
write(*,*)' particles. Scaling is',xscl1,xsc12,' for the x-axis,'
write(*,*)' and ',yscl,' for the y-axis.'
write(*,*)

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 fill rate constant'
write(*,*)'enter tuning constant for lateral movement'
write(*,*)'enter bandwidth in position histogram, rec (5,100):'
write(*,*)'enter power of 2 bins in histograms <=1024:'
write(*,*)'enter seed for random number generator (large):'
open(unit=5,file='//stat600/stat/ramos/pick/simul/data/barfl') (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
read(6,889)((coef(j,m), m=1, mdims), j=1, jdims)
read(6,888)(rknots(j), j=1, jdims)
close(6)
format(g16.8)
888 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, icoll, 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.
oxo(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
open(unit=8, file='//stat600/stat/ramos/pick/simul/data/barf4') {apollo}
write(*,*)'enter b for lateral movement [.05,.2]'
write(*,*)'enter c for sdx=c/t'
read(8,*) bx, cx
close(8)
sdx=cx/t
cx=exp(max(-30. , -bx*t))
gsd=sqrt((1-cx**2)*sdx
call paramet(1., rrmux, gsdx, cx, bx, sdx)
sdx=sdx*xk
gsdx=gsdx*xk
corr(xv)=(1-cx)/sqrt((2*bx*t+cx*(4-cx)-3)*(1+cx)/(1-cx))
sdx=sqrt((2*bx*t+(1-cx)*(cx-3))*sdx/bx
c
write(*,*)'correlation and sd for lateral movement'}
write(*,*)corrxy, sdx

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

c initialize graphics
  call g_init

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

c calculations for new view
  ndead=0
  do 20 i=1,npart
  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
    corryv=(1-c)**2/sqrt((2*b*t+c*(4-c)-3)*(1-c**2))
    corrv=(1-c)/sqrt(2*b*t*(1+c)/(1-c)+c-3)
\[ sdy = \sqrt{2b \cdot t + (1-c) \cdot (c-3) \cdot sd/b} \]

next get the actual variables; first get position increment
by first: multiplying N(0,1) variable by appropriate sd,
\[ yx(i) = sdy \cdot (corrvv \cdot vv(i) + \sqrt{1-corrvv^2}) \cdot yx(i) \]
\[ xx(i) = sdx \cdot (corrvx \cdot vvx(i) + \sqrt{1-corrvx^2}) \cdot xx(i) \]
\[ yy(i) = \min(yy(i) + yx(i) + rmu \cdot t + (v(i) - rmu) / b \cdot (1-c), vlength) \]
\[ xo(i) = xo(i) + xx(i) + vx(i) / bx \cdot (1-cx) \]
\[ xo(i) = xo(i) + xx(i) + vx(i) / b \cdot (1-c) \]

then get mean for velocity and then multiply by sd and add mean
if position is negative then set equal to -position and set
velocity to 0
\[ rmean = rmu + (v(i) - rmu) \cdot c \]
\[ rmeanx = vx(i) \cdot cx \]
\[ vv(i) = gsd \cdot vv(i) + rmean \]
\[ vvx(i) = gsdx \cdot vvx(i) + rmeanx \]
end if
\[ scx(i) = \text{mod}\left(\text{int2}(xo(i) \cdot xsc12 / vwidth + 11 \cdot xsc12), xsc12 + xsc1l \right) \]
\[ scy(i) = \text{int2}(yy(i) \cdot ysc1 / ovlength) \]
\[ ov(i) = v(i) \]
\[ vx(i) = vv(i) \]
\[ sv(i) = ssv(i) \]
\[ oscy(i) = scy(i) \]
\[ ooscx(i) = scx(i) \]
\[ vlength = ovlength \cdot (1. - fill \cdot \text{real}(ndead) / \text{real}(npart)) \]

smoothing
\[ \text{call kernel2}(yy, conc, ov, ssv, in, in2) \]

erasing and plotting
first the particles
\[ \text{call erase}(1) \]
\[ n = ndraw \]
do i = 1, ndraw
\[ \text{center}(1) = ooscx(i) \]
\[ \text{center}(2) = oscy(i) \]
\[ \text{icol} = \text{red} \]
if (i.le.npart-nhi)icol=black
\[ \text{call gpr}$\_set$\_fill$\_value$(icol, status) \]
\[ \text{call gpr}$\_circle$\_filled$(center, rad, status) \]
\[ \text{center}(1) = scx(i) \]
\[ \text{center}(2) = scy(i) \]
\[ \text{icol} = \text{icol1} \]
if (i.le.npart-nhi)icol=icol2
\[ \text{call gpr}$\_set$\_fill$\_value$(icol, status) \]
\[ \text{call gpr}$\_circle$\_filled$(center, rad, status) \]
\[ oscy(i) = scy(i) \]
\[ ooscx(i) = scx(i) \]
enddo
then the curves
\[ \text{call packed}(yellow, ovlength, vlength) \]
do i = 1, npart
\[ \text{con}(i) = \text{conc}(in(i)) \]
\[ \text{sv}(i) = ssv(in2(i)) \]
\[ \text{sv}(i) = ssv(in(i)) \]

get conc and vel for each particle from smooths
\[ \text{call timer}(j, t) \]
\[ \text{call pause} \]
finally the text
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
t (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) = xsc12*int2(k) + xscll - int2(1)
x(2) = x(1)
x(3) = xsc1 + 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

******************************************************************************************
SUBROUTINE TO fill POLYGON

INPUTS: X, Y -- COORDINATES IN PIXEL VALUE
N -- NUMBER OF SIDES OF POLYGON
COL -- COLOR TO USE IN FILLING

******************************************************************************************

subroutine fill(x, y, n, col)
intrinsic

integer*2 x(1), y(1)
intrinsic
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 Allocate and initialize bitmap for graphics in borrowed mode

integer*4 desc, bl_desc, status
integer*2 mode, hi, unit, bsize(2)
intrinsic
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----------------------------------------------------------
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
D2P31M = (2**31)-1
D2PN31 = (2**31)
DATA D2P31M/2147483647.DO/
DATA D2PN31/2147483711.DO/
C FIRST EXECUTABLE STATEMENT
DO 10 I = 1,NR
   DSEED = DMOD(16807.DO*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 'Fl'
  %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)
  if( e_d.eq. char(16#C0) ) then
     call savpic('//stat600/stat/ramos/tmp/scrdmp',
               int2(31),window)
  end if

  call GPR_$CLEAR(int4()),status
  continue
  dummy = GPR_$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.
SUBROUTINE SHOW(ch)

%no list
%include '/sys/ins/cal.ins.ftn'
%include '/sys/ins/time.ins.ftn'
%list
c character ch*1
integer*2 clock(3),i xp,i yp
integer*4 seconds,status
data i xp,i yp,seconds/1000,20,1/
call CAL$_SEC_TO_CLOCK( seconds,clock )
call TIME$_WAIT( TIME$_RELATIVE,clock,status )
return
end

%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

call gpr$_$enable_input(gpr$_$keystroke,key_set,status)
unobscured = gpr$_$event_wait(event_type,e_d,position,status)

c If the key 'Fl' 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

call GPR$_$CLEAR(INT4(0),STATUS)
return
end

subroutine savpic(filename,length,rectangle,status)

c This subroutine dumps the bitmap onto a file which can be
printed using PRF ... -PLOT.

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 )
wpl = (rectangle(3)-16)/16
wpl = line_wid
call gmf$_$open( filename,length,gmf$_$overwrite,str_id,status )
call gmf$_$copy_subplar( 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)
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/colorinfo.ftn'
%include '//stat600/stat/ramos/pick/simul/src/graf.com.ftn'
%include '//stat600/stat/ramos/pick/simul/src/param.com.ftn'
linclude '//stat600/stat/ramos/pick/simul/src/param2.com.ftn'

EXE STATR

half=(vlength+8*h)/nbins/2
origin=-4*h+half

ycoord(1)=int2(-4*h/vlength*yscl)
conarray(1)=int2(0)
do 10 j=1,nbins

jj=j/2+mod(j,2)
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

ycoord(j)=int2(y)
10 continue

velarray(1)=velarray(2)
fluxarray(1)=fluxarray(2)

y=(-4*h+(jj-1)*(vlength+8*h)/nbins)/;
call clip(0)
coll=mod(kk,7)+1
col2=mod(kk*8+1,7)+1
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)

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
10 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))GOTO 50
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

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

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

subroutine interv(xt,lxt,x,left,mflag)
***
xt should be an increasing sequence
***
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)
this subroutine estimates the density of data x
it returns for each x a value. The sum of values
is npart, i.e. the average value is 1.
It also estimates the smoothed drift sv, of velocities v
inputs
   x array which has the positions of each particle
outputs
   kk is a vector of length the number of
      particles which at the jth position contains
      the number of the horizontal strip in which
      that particle fell, i.e., its bin number
   kkk is similar, excepts it has the bin no.
      of the bin where the particle is headed for.
   counts,      array with smoothed concentration
               for each bin
real*4 x(l),v(l),hi,rio,.crr
integer*4 iw(i),kkk(l),j,k
integer*4 k(k),k(k),n
real*4 fill(1024),rnx,smm,c
complex counts(l),sv(l)

#include './stat600/stat/ramos/pick/simul/src/param.com.ftn'
#include './stat600/stat/ramos/pick/simul/src/param2.com.ftn'
calculate parameters
  hi=vlength+4*h
  rlo=-4*h
c=(h*8*atan(1.))/(hi-rlo)**2
norm=1.+8./vlength*h

get parameter for packed bed, finds out what bin the vessel ends at, i.e. at bin 4.5 so it knows how to adjust for the packed bed later on...in (*****)
smm=(vlength/(vlength+8*h)+1)*real(nbins)/2.
mm=int(smm)
if(smm-mm.gt.0.00001)then
  mm=mm+1
end if

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

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

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)
    kkkk(j)=max(1,kkkk(j))
    counts(kk(j))=counts(kk(j))+1.
    sv(kk(j))=sv(kk(j))+v(j)
  20 continue

cut in packed bed
(*****)
  counts(mm+1)=counts(mm+1)+pack/incon*npart/nbins*(smm-mm)
  if(mm+2.l.e.nbins)then
    do 21 j=mm+2,nbins
      counts(j)=pack/incon*npart/nbins
    21 continue
  end if

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

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

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

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,name1,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'
dinclude '//stat600/stat/ramos/pick/simul/src/graf4.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(xscll)/10.-8)
call gpr_$move(xpos,ypos,status)
call gpr_$text('Co',int2(2),status)
xpos=xpos+int2(real(xscll)/2.)-int2(45)
call gpr_$move(xpos,ypos,status)
call gpr_$text('0',int2(1),status)
xpos=int2(real(xscll)/2.+real(xscll)/10.-8)
call gpr_$move(xpos,ypos,status)
call gpr_$text('Uo',int2(2),status)
xpos=int2(15)
ypos=ypos+space
strl=int2(18)
call gpr_$move(xpos,ypos,status)
call gpr_$set_text_value(yellow,status)
call gpr_$text(' ',strl,status)
ypos=ypos+space
call gpr_$move(xpos,ypos,status)
call gpr_$set_text_value(blue,status)
call gpr_$text(' ',strl,status)
xpos=int2(real(xscll)/2.)
ypos=yscl+2*space-int2(10)
call gpr_$move(xpos,ypos,status)
call gpr_$set_text_value(magenta,status)
call gpr_$text(' ',strl,status)
xpos=int2(100)
ypos=yscl+int2(4)*space
call gpr_$move(xpos,ypos,status)
call gpr_$set_text_value(green,status)
call gpr_$text(' ',strl,status)
xpos=xpos+space
write(s_nbins,10)nbins
format(14)
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)
format(f4.1)
strl=4
call gpr $text(s_h,strl,status)
strl=int2(18)
xpos=xsc11+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
format(f4.3)
ypos=ypos+space
call gpr $move(xpos,ypos,status)
call gpr $set_draw_value(col,status)
call gpr $polyline(x,y,nd,status)
return
end

subroutine line (x,y,nd,icol,status)
integer*2 x(1),y(1),nd
integer*4 status,icol
call gpr $move(x(1),y(1),status)
call gpr $set_draw_value(icol,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)=xsc11+xsc12
x(2)=x(1)
x(3)=xsc11
x(4)=x(3)
y(1)=int2(vlength/ovlen:n relative scl)
y(2)=yscl
y(3)=yscl
y(4)=y(1)
call fill(x,y,4,icol)
return
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)  

cparametrization for mean and sd in terms of concentration

inputs  
con  
outputs  
rmu, gsd, c, b, sd  

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. - 0.0615 * co + 0.03136 * co**2.  
else  
co = co - 3.69  
rmu = 1.2001 + 2.295 * co - 1.447 * co**2. + 0.2667 * co**3. - 0.01547 * co**4.  
end if  
rmu = max (rmu, 0.)  
b = 2.5 * conn + 0.001 / (conn + 0.000001)  
sd = 0.6 * 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)  

cparametrization for mean and sd in terms of concentration

inputs  
con  
outputs  
rmu, gsd, c, b, sd  

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 = 0.807 * rmu**2 * conn**.33333333  
b = 2.5 * conn + 0.001 / (conn + 0.000001)  
c = exp (max (-30., -b*t))  
gsd = sqrt (1 - c**2) * sd  
return  
end  

subroutine plot (x, y, n, ival)  

cPlots curve through points (x(i), y(i)) , i=1, ..., n

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
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, 1, 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) = ysc1
do 20 j = 0, m
    col = red
    x(1) = int2 (real (j) * xsc1 / 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) = xsc1
window (2, 1) = int2 (0)
if (j .eq. 0) window (1, 2) = xsc1 - int2 (1)
if (j .eq. 1) window (1, 2) = xsc2 - int2 (1)
if (j .eq. 2) window (1, 2) = xsc1 + xsc2 - int2 (1)
window (2, 2) = ysc1 - 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******************************************************************************

SUBROUTINE TO fill POLYGON

INPUTS: X,Y -- COORDINATES IN PIXEL VALUE
N -- NUMBER OF SIDES OF POLYGON
COL -- COLOR TO USE IN FILLING

******************************************************************************

subroutine fill(x,y,n,col)

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

C******************************************************************************

Allocate and initialize bitmap for graphics in borrowed mode

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
INTEGER NR
REAL R(NR)
DOUBLE PRECISION DSEED

C SPECIFICATIONS FOR LOCAL VARIABLES
C
INTEGER I
REAL U
DOUBLE PRECISION D2P31M,D2PN31

D2P31M = (2**31)-1
D2PN31 = (2**31)

DATA D2P31M/2147483647.DO/
DATA D2PN31/2147483647.DO/

DO 10 I = 1,NR
   DSEED = DMOD(16807.DO*DSEED,D2P31M)
   R(I) = real(DSEED/D2PN31)
10 CONTINUE
RETURN
END

SUBROUTINE PAUSE
C
This subroutine holds the screen in borrowed mode so that
the plot does not immediately disappear.
C
The screen is cleared after any key is struck, with a printable
screen dump produced if the key is 'F1'

share
*include '/sys/ins/gpr.ins.ftn'

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$_{\text{ENABLE\_INPUT}}$( GPR$_{\text{KEYSTROKE}}$, key_set, status )
dummy = GPR$_{\text{EVENT\_WAIT}}$( e_t, e_d, pos, status )

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

10 continue
dummy = GPR$_{\text{COND\_EVENT\_WAIT}}$( e_t, e_d, pos, status )
if( e_t.ne.GPR$_{\text{NO\_EVENT}}$ ) 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 )

c Call the inq bitmap and inq raster ops:
call GPR$_{\text{INQ\_BITMAP}}$( bmdesc,status )
call GPR$_{\text{INQ\_RASTER\_OPS}}$( rop,status )

Copy all the planes into plane zero:
call GPR$_{\text{SET\_RASTER\_OP}}$( plane_0,bit_or,status )
do i=1,hiplane
call GPR$_{\text{BIT\_BLT}}$( bmdesc,window,i,window(1,1),plane_0,status )
end do

Dump the bitmap onto a file:
call SHOW('4')
call GPR$_{\text{INQ\_BITMAP\_POINTER}}$( bmdesc,pointer,line_wid,status )
call GMF$_{\text{OPEN}}$( filename,length,gmfstatus,str_id,status )
call GMF$_{\text{COPY\_PLANE}}$( str_id,invert,bpi,pointer,window(1,2),
 + window(2,2),line_wid,status )
call GMF$_{\text{CLOSE}}$( str_id,status )
call GPR$_{\text{SET\_RASTER\_OP}}$( plane_0,rop(1),status )
return
end

SUBROUTINE SHOW( ch )
%list
%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_$RELATIVE,clock,status )
return
end
subroutine pause
%no list
%include '/sys/ins/base.ins.ftn'
%include '/sys/ins/error.ins.ftn'
%include '/sys/ins/gpr.ins.ftn'
%list
c This subroutine holds the screen in borrowed mode so that
the plot does not immediately disappear.
c
character*1 e_d
integer*4 status
integer*2 position(2),e_t,key_set(16),rectangle(4)
logical unobscured
c
data key_set/16*16#ffff/
rectangle(1) = 0
rectangle(2) = 0
rectangle(3) = 1023
rectangle(4) = 1023
c
call gpr$_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#0C)) 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)

This subroutine dumps the bitmap onto a file which can be
printed using PRF ... -PLOT.

%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
character*(*) filename
integer*2 rectangle(4),wpi,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 = 1),
+ 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$inq_bitmap_pointer( bmdesc, pointer, line_wid, status )
wpl = (rectangle(3)+15)/16
wpl = line_wid
call gmf$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, coll, col2, col3
real*4 y, origin, half
complex conc(l), smv(l)
%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
    half=(vlength+8*h)/nbins/2
    origin=-4*h+half
    ycoord(j)=int2(-4*h/vlength*yscl)
    conarray(1)=int2(0)
do 10 j=1,nbins
    jj=j/2+mod(j,2)
    jk=(j+1)/2-mod(j,2)
    conarray(j)=int2(min(real(conc(j))*ban*xscll, xscll))
    velarray(j)=int2(min(real(smv(j)+5)*ban*xscll, xscll))
    fluxarray(j)=int2(min(real(conc(j)*smv(j))*ban*xscll, xscll))
    y=(origin+(j-1)*2*half)/ovlength*yscl
    y=(origin+(j-1)*2*half)/ovlength*yscl
10 continue

c ycoord(j)=int2(y)

do 10 continue

c call clip(0)
c coll=mod(kk,7)+1
    col2=mod(kk*8+1,7)+1
    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)

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 90
ILO=LXT-1
35 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 IHI=LXT
GOTO 110

subroutine interv(xt,lxt,x,left,mflag)

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

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

this subroutine estimates the density of data x
it returns for each x a value. The sum of values
is npart, i.e. the average value is 1.
It also estimates the smoothed drift sv, of velocities v
inputs
x array which has the positions of each particle
outputs
kk is a vector of length the number of particles which at the jth position contains the number of the horizontal strip in which that particle fell, i.e., its bin number.

kkk is similar, excepts it has the bin no. of the bin where the particle is headed for.

counts, array with smoothed concentration for each bin

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 ends at, i.e. at bin 4.5 so it knows how to adjust for the packed bed later on...in (*****)
    smm=(vlength/(vlength+8*h)+1)*real(nbins)/2.
    mm=int(smm)
    if(smm-mm.gt.0.00001)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)
        kkkk(j)=max(1,kk(j))
        kkkkk(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
(******)
    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
continue
end if

; c normalize sum of velocities at each bin
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
   rnx=rnx+real(counts(i))
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
   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(’O’,int2(1),status)
xpos=int2(real(xscl1)/10.-8)
call gpr_$move(xpos,ypos,status)
call gpr_$text(’Co’,int2(1),status)
xpos=xpos+int2(real(xscl1)/10.-int2(45)
call gpr_$move(xpos,ypos,status)
call gpr_$text(’O’,int2(1),status)
xpos=int2(real(xscl1)/10.-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_$move(xpos,ypos,status)
subroutine line (x, y, nd, col, status)
integer*2 x(1), y(1), nd
integer*4 status, col
call gpr_$move(x(1), y(1), status)
call gpr_$set_draw_value(col, status)
return
end
call gpr_$polyline(x,y,nd,status)
return
end

subroutine packed(icol)
%include '//stat600/stat/ramos/pick/simul/src/graf.com.f90'
%include '//stat600/stat/ramos/pick/simul/src/param2.com.f90'
integer*2 x(4),y(4)
integer*4 n,k,icol
x(1)=xsc11+xsc12
x(2)=x(1)
x(3)=xsc11
x(4)=x(3)
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
inputs
con
c
outputs
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.f90'
conn=con*incon
do =asinh(conn*10000.)
if (do.le.3.69) then
rmu=1-.0615*do+.03136*do**2.
else
co=do-3.69
rmu=1.2001+2.295*co-1.447*co**2.+2.667*co**3.-.01547*co**4.
end if
rmu=max(rmu,0.)
b=2.5*do+.001/(do+.000001)
sd=.6*rmu*(1-exp(max(-30.,-100*do**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
inputs
con
c
outputs
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.f90'
%include '//stat600/stat/ramos/pick/simul/src/spline.com.f90'
conn = conn * 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)
C
C REFERENCE: 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


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


32. BREIMAN, L. (May 1984). Nail finders, edifices, and Oz.
43. BICKEL, P. J. and YAHAV, J. A. (1985). On estimating the number of unseen species: how many executions were there?
56. LE CAM, L. and YANG, G. L. (January 1986). Replaced by No. 68.
64. O'SULLIVAN, F. (May 1986). Nonparametric Estimation in the Cox Proportional Hazards Model.
71. BICKEL, P.J. and YAHAV, J.A. (July 1986). Richardson Extrapolation and the Bootstrap.
77. O'SULLIVAN, F. (September 1986). Deconvolution of episodic hormone data.


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


95. CANCELLED


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


127. Same as No. 133.

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


131. Same as No. 140


134. CANCELLED


163. Cancelled


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


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.
186. Cancelled.


234. 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.


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


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


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


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


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.