Spring 4-1999

A Computational Process-Response Model of Hillslope Evolution Applied to Undercut Slopes on Abandoned Incised Meanders in the Eastern Highland Rim of Tennessee USA

Richard Tran Mills
University of Tennessee - Knoxville

UNIVERSITY HONORS PROGRAM

SENIOR PROJECT - APPROVAL

Name: Richard Tran Mills

College: Arts & Sciences Department: College Scholars, Geology, Physics

Faculty Mentor: Dr. William E. Blass

PROJECT TITLE: A Computational Process-Response Model of Hillslope Evolution Applied to Undercut Slopes on Abandoned Incised Meanders in the Eastern Highland Rim of Tennessee, USA

I have reviewed this completed senior honors thesis with this student and certify that it is a project commensurate with honors level undergraduate research in this field.

Signed: [signature], Faculty Mentor

WILLIAM E. BLASS

Date: 4/26/99

Comments (Optional):

A COMPUTATIONAL PROCESS-RESPONSE MODEL OF HILLSLOPE EVOLUTION APPLIED TO UNDERCUT SLOPES ON ABANDONED INCISED MEANDERS IN THE EASTERN HIGHLAND RIM OF TENNESSEE, USA

Richard Tran Mills
Department of Geological Sciences and
Department of Physics and Astronomy
University of Tennessee
Knoxville, Tennessee 37996, USA

INTRODUCTION

Evolution of hillslopes on resistant bedrock takes place so slowly that direct observation of change in most cases is impossible. A traditional way to study this evolution is to substitute space for time by ordering modern-day hillslope profiles according to their relative age, and then considering their forms to represent stages of a developmental sequence. In the unglaciated Appalachians and Interior Plateaus of southeastern North America, landscapes are poorly dated, and finding a chronosequence of hillslope profiles is difficult. One opportunity to do so is provided by incised meandering streams that show "ingrown" meanders, characterized by gentle slip-off slopes on the inside of the meanders and steep undercut slopes on the outside. Some of these meanders become abandoned when stream erosion cuts through their narrow necks. Once the meander is abandoned, the hillslope on the outside of the meander is no longer actively eroded and its profile evolves into a new form with lower slope angles. The higher the cutoff meander above the modern stream level (AML), the older the meander. Thus, the height AML of the meander can be used as a proxy for hillslope age.

Here, a newer method was used in conjunction with this traditional approach to complement and extend it. A general computational process-response model of hillslope evolution based on the mass-balance equation was developed and used to simulate the transition of actively undercut slopes into slopes on abandoned meanders. The combination of space for time substitution and computer modeling was used to study hillslope development on the Eastern Highland Rim, Tennessee, where streams are incised as much as 100 m below the plateau surface (Figure 1). This thesis focuses mainly on the modeling portion of the study; the full details of the study are presented elsewhere (Mills and Mills, 1997).

PHYSICAL SETTING

The eastern Highland Rim of Tennessee is a plateau standing at an elevation of about 300 m, situated between the Cumberland Plateau to the southeast (elevation about 550 m) and the Central Basin to the northwest (elevation about 200 m) (Figure 1). This area is underlain, for the most part, by five formations. From oldest to youngest these are the Leipers and Catheys Formations (Ordovician), commonly mapped as one unit; the Chattanooga Shale (Devonian and Mississippian); the Fort Payne Formation (Mississippian); and the Warsaw Formation (Mississippian). The Leipers-Catheys unit contains coarse-grained, fine-grained, and argillaceous limestone, and has a maximum exposed thickness of 45 m. In the incised stream valleys, this unit crops out at the base of the slopes and on the valley floor. The Chattanooga Shale is a carbonaceous, fissile shale about 8 meters thick. It crops out in settings similar to the Leipers-Catheys. The Fort Payne Formation contains silicastone, calcareous siltstone, argillaceous limestone, and bands and nodules of dense chert. Much of its silica apparently has formed by replacement of limestone. Fourteen samples from the Fort Payne in the Cane Creek area (north of Burgess Falls in southern Putnam County) were dissolved in formic acid to determine the percent of insoluble materials by weight. This percentage ranged from 28.7 to 91.1, with a mean of 55.0. The Fort Payne thickness ranges from 50 to 75 m. Of the units described here, the Fort Payne is by far the most resistant to erosion and generally forms the steep valley walls along the incised streams. The Warsaw Formation is a limestone with various concentrations of sand, calcareous siltstone, calcareous shale, and argillaceous limestone. In the study areas it occurs mainly on the surface of the Highland Rim. Its thickness ranges from 25 to

35 m (Wilson and Marcher, 1968). Regional dip is a fraction of 1° to the southeast, but small local structures also occur.

The present mean annual temperature on the northern Eastern Highland Rim is about 15°C and the mean annual rainfall about 1320 mm. However, pollen records on the Rim demonstrate much colder temperatures during the last glacial maximum. For example, in a record from Anderson Pond (36° 02' N, 85° 30' W) at an elevation of about 300 m, Delcourt (1979) reported vegetation patterns for 18 kyr that indicate mean annual temperatures near 0°C. Thus, periglacial conditions have existed here during parts of the Pleistocene.

Many streams near the western margin of the Highland Rim plateau are deeply incised, flowing in gorges as much as 100 m deep. Generally these valleys show "ingrown" meanders, characterized by gentle slip-off slopes on the inside of the meanders and steep undercut slopes on the outside. Some of these meanders have been abandoned when stream erosion cut through the narrow neck of the meanders (Figures 2-3). The floors of these "cutoff" meanders range in height from 2 m to as much as 43 m above the modern stream. The age of abandonment thus varies from recent to ancient; some idea of the time involved can be gained by considering regional denudation rates and stream incision rates. Based on dissolved stream loads, Reesman and Godfrey (1981) found that the chemical denudation of the Central Basin is about 40 m myr^{-1}. Since the incised streams on the Highland Rim are graded to the Central Basin, a downcutting rate of this amount would seem reasonable. Also germane to this question is the stream incision rate determined by Sasowsky *et al.* (1995) for the East Fork of the Obey River near the western edge of the Cumberland Plateau (site A in Figure 1). Based on heights of paleomagnetically dated cave passages above the present stream, they estimated this rate to be 60 m myr^{-1}. Although the formations involved are stratigraphically higher than those of the valleys considered here, the climatic and neotectonic settings of this river are similar to those of the incised Highland Rim streams, so that this rate is probably applicable. Taking the 40 m myr^{-1} and the 60 m myr^{-1} as a probable range of incision rates, the highest cutoff meander was abandoned between 1.08 and 0.72 Ma.

METHODS

Twelve cutoff meanders that most closely matched in regard to stratigraphy and depth of stream incision were selected from about twice that number on the Eastern Highland Rim. Several approaches were then used to study the changes in the form of the undercut slope as a function of the vertical height of the abandoned meander floor above the modern stream. Briefly, these were:

- First, on topographic maps (scale 1:24,000, contour interval 20 ft [6.1 m]), the maximum slope angle over a vertical distance of 100 ft (30.5 m) was measured on the outside of each meander. These angles were then plotted against the maximum height AML of the abandoned meander floor.

- Second, 21 hillslope profiles were surveyed on both active and abandoned undercut slopes, by means of tape and clinometer. The actively eroded slopes were examined to determine the effect of stratigraphy on the form of slope profile, and to determine the probable original form of hillslopes on the abandoned meanders. Then the form and steepness of the abandoned undercut slopes were examined and related to the height AML.

- Third, a small number of seismic refraction lines were run in order to determine approximate thicknesses of colluvium on floors and valley walls of abandoned meanders.

- Fourth, for selected profiles, a computer model based on work by Kirkby (1971, 1984, 1987, 1992, and unpub. data, 1991) and Kirkby *et al.* (1992) was written and used to simulate the evolution of hillslope profiles over time.

The results of the first three approaches are documented in full in Mills and Mills (1997); here, only those results pertinent to the computer modeling approach are mentioned.

The model was utilized by taking one of the actively undercut slopes as the initial profile, and then running the model until the profile had "evolved" into a form approaching that of a particular profile on an abandoned meander. As process rates were not measured in this study, the approach was to use appropriate rates from the literature. Different values of rates were tried in order to make the simulated profile most closely approach the form of the actual profile, subject to the restriction that only rates that seemed reasonable could be used. The best-fitting simulated profile was taken to be the one with the smallest mean absolute difference between elevations along the simulated and actual profiles. The model time necessary to reach this best-fit profile was then recorded. Hundreds of simulations were run in the course of the study.

THE PROCESS-RESPONSE HILLSLOPE EVOLUTION MODEL

The computer model developed is based heavily on work by Kirkby (1971, 1984, 1987, 1992, and unpub. data, 1991) and Kirkby *et al.* (1992). The essentials of the model are as follows. The hillslope profile is divided into a series of equally spaced cells (51 in the simulations we ran), with the storage in each cell representing the elevation at a point on the hillslope. Between each time step, sediment fluxes into and out of each cell are calculated from empirical process laws, and from these the accompanying changes in the elevation of each cell are determined. Climate and lithology are held constant, so process rates depend largely upon the slope topography; i.e., distance from the divide and downslope gradient. A fixed time step of small enough size to prevent numerical instabilities is used.

Process laws

"Creep" includes a group of processes which depend on gradient but not on collecting area, and have no lower threshold. In the present setting it consists mainly of soil creep and solifluction. Creep is assumed to carry sediment at a rate directly proportional to the downslope gradient. "Wash" refers to overland flow that is able to entrain and carry soil particles on the surface. Unlike creep, wash depends on collecting area (i.e., distance from the divide) as well as on gradient. The sediment flux S out of a cell from creep and wash processes combined is given by (Kirkby *et al.*, 1992):

$$S = K[1 + (x/u)^2]g \quad (1)$$

where x is the distance from the divide, g is the downslope gradient, K is a constant giving the rate of creep, and u is the distance in meters beyond which the wash term, $K(x/u)^2g$, becomes larger than the creep term.

Landslides are modeled as a continuous process; hence the sediment flux due to landslides represents a long-term average, rather than individual slides. The use of these average rates assumes that individual slides are small enough not to change the slope profile significantly. The flux due to landslides is controlled by four parameters, which are discussed in more detail in Kirkby (1984, 1987). Two of these are thresholds: a lower, stable gradient g_ϕ below which there is no landslide activity, and an upper gradient g_t above which slides will never come to rest. The first depends on the angle of internal friction and whether pore pressure can develop. Likely values range from 0.14 (8°) for clays up to about 0.58 (30°) for some sandstones. The second may usually be related to the talus angle of repose of 0.7 (35°). The third is a rate constant α which governs the rate of free degradation, or unconstrained lowering, D which is given by

$$D = \alpha g (g - g_\phi). \tag{2}$$

α may range from 0.001 m yr^{-1} for sandstones to as much as 10 m yr^{-1} for clays (Kirkby, 1987). The fourth parameter h_0 indicates the average height from which blocks fall from cliffs, which should be roughly their mid-height. It is in a sense used to represent the momentum of the falling blocks in the expression for the mean horizontal distance h traveled by the moving material:

$$h = h_0/(g_t - g). \tag{3}$$

The value of h_0 influences how far a slide can run out across gently sloping ground at the base of a slope, but generally has only a slight effect on the slope profile elsewhere. Combining the expressions for detachment rate and travel distance, the sediment flux S_i out of cell i due to landslides is given by (Kirkby *et al.*, 1992)

$$S_i = \frac{Ddx + S_{i-1}}{1 + (1/h)dx} \tag{4}$$

where dx is the spacing between cells, and S_{i-1} is the slide flux out of cell $i - 1$.

Solution is modeled as a rate of uniform vertical lowering; with each iteration of the model, each cell is lowered by an amount determined by

$$dz = -r_s dt \tag{5}$$

where dz is the change in elevation of the cell, r_s is the rate of solution, and dt is the time step used. Unlike the other processes modeled, solution does not contribute to the flux of sediment being transported to cells downslope; rather, it is assumed that any material freed by solution immediately leaves the system. This assumption is reasonable for all but very arid climates, where reprecipitation of dissolved minerals can become significant.

The basis for the hillslope model--the mass-balance equation

For each iteration of the model, sediment fluxes out of each cell (and hence into the adjacent cell downslope) due to creep and landslides are calculated. These are then grouped together into a total sediment outflux for each cell, and then converted into the resultant changes in elevation. The basis for this is the mass-balance, or continuity equation, which may be written as

$$\frac{\partial z}{\partial t}=\frac{\partial S}{\partial x} \qquad \textbf{(6)}$$

where S is the downslope flux of sediment and z the elevation at distance x from the divide, and t is the elapsed time. In more concrete terms, the elevation changes are determined from the expression:

$$\frac{\partial z}{\partial t}=\frac{S_{i-1}-S_i}{\partial x} \qquad \textbf{(7)}$$

where S_i is the sediment flux out of cell i, and S_{i-1} the flux out of cell i-1 and, hence, the flux into cell i. Once the rate of elevation change $\partial z/\partial t$ due to downslope sediment transport for a cell is calculated, the change in elevation is determined by multiplying by the time step dt. Combining this rate of elevation change with that due to solution, the explicit expression for the elevation of a cell can be written:

$$z_{t+dt}=z_t+(\frac{\partial z}{\partial t}-r_s)dt \qquad \textbf{(8)}$$

where z_t is the elevation of a cell at time t.

Additional assumptions

Due to the inherent limitations of computer models, a few somewhat artificial assumptions must be made. At the divide, the calculated downslope sediment flux is doubled, because it is assumed that an equal amount of sediment leaves in each direction (the rate of solution is *not* doubled). At the final basal cell, provision is made for the user to choose whether all sediment transported in is to be removed or whether a fixed percentage of entering material is to be retained. In the interest of numerical stability, negative elevations are not allowed; if a cell's calculated elevation comes out negative, it is set to 0. Generally, solution is the only process which could cause negative elevations, were this requirement not imposed.

Implementation

The computer model was written entirely in the Microsoft Visual Basic 5 programming language. Although Visual Basic (VB) is not commonly used to write numerical modeling code, it was chosen for several reasons. First, since we initially did not know the specifics of how we wished to model to operate, VB seemed like a good choice because of its suitability for rapid application prototyping. Second, it was important for the program to have an intuitive graphical user interface that could be easily understood by other users, and VB excells at interface design. Third, it was important that others be able to easily understand and modify the program code. The forgiving syntax and widespread use of VB allow the language to be easily learned. VB does have the disadvantages of being proprietary and somewhat slow, but these are not real problems, as the software is relatively cheap and even the longest model runs should be completed in a few minutes on an Intel 486-class machine.

The program code was written keeping ease of future modification in mind. The erosional processes (with the exception of uniform vertical solution, which is discussed in the program documentation) are implemented in a modular fashion, with each process being implemented in separate function that returns the downslope sediment flux due to the action of that process. Thus one can easily modify how a process is modeled or add new a one to the program without having to make changes spread throughout the program code.

A freeware public distribution of the modeling program, titled *Richard's n-store Hillslope Dynamics Model* (*HDS* for short), has been prepared. The model will be distributed with full source code, and is licensed under the terms of the GNU General Public License, version 2, which allows redistribution and/or modification of the program as long as the modified versions remain free and also licensed under the terms of the GNU General Public License. Efforts have been made to insure that the documentation and program commenting are adequate enough to allow users to easily modify and extend the program code. The documentation for the program is provided in Appendix A, and the full source code of the model is in Appendix B.

RESULTS AND DISCUSSION

The hillslope evolution model was applied to one profile from each of the Cane Creek cutoff meanders (Figure 4). For each of these profiles, the most appropriate profile from the actively undercut hillslopes (Figure 5) was used as an initial profile. The appropriateness of a profile was determined on the basis of how well its stratigraphy and relief matches that of the profile on an abandoned meander that we wished to "evolve" the initial profile into. For example, for the oldest cutoff, where the meander floor had not yet cut through the base of the Fort Payne Formation, the profile U4 was used as the initial profile.

The rationale for the selected process rates was as follows. For the vertical solution rate (r_s), a value of 50 μm yr^{-1} was used. This is a high rate for silicate rocks, but a low rate for carbonate rocks. As the Fort Payne Formation, the chief formation with which we are concerned, consists of more than half insolubles (mostly silica), but does contain some limestone beds, its solution rate is probably intermediate between the two types of rocks.

The landslide threshold angle (g_{ϕ}) was assumed to be 23°, based on the angles of internal friction estimated from the particle-size distributions (Table 1) using the triangular diagram of Kirkby (1973, Figure 5). We chose to estimate rather than experimentally measure the angles of internal friction because incorporation of the large clasts present in the slope debris is difficult using a shear box of typical size. Furthermore, it seems that slope angles in the study area are only partly controlled by the mechanics of the surficial mantle, as the correlation between angle of internal friction and slope angle is poor. For example, the angles of the straight segments in profile A3 are 36° and 38° , and are 34° and 35° for A2. The 34° - 38° range is typical of talus slopes; however, as shown in Table 1, there seems to be too much silt and clay in the debris for it to behave as talus. Talus can stand at an angle close to its angle of internal friction (ϕ) because its interstices are too large to allow significant pore pressure to develop even during intense rainfalls. However, the amount of fine material in the debris mantles of A3 and A2 should be sufficient to produce complete saturation, which would produce a maximum slope angle θ such that, approximately, $\tan \theta = \frac{1}{2} \tan \phi$ (Skempton, 1964), which, for the ϕ values shown in Table 1, would yield θ equal to 21°-25°. Yet, the observed maximum angles are much closer to the ϕ values than they are to these angles. This finding is difficult to explain, except by assuming that slope angle is at least partly controlled by factors other than the mechanics of the surficial mantle. One possible explanation is that bedrock ledges act to "dam" debris and thereby increase the slope angle from what it would be if the bedrock lacked ledges.

The talus angle (g_t) was assumed to be 35°, a typical value. No data were available to estimate the rate of free degradation above threshold (α). Given that the Fort Payne is a highly resistant unit, however, it was assumed to have a rate comparable to sandstone, so that the value of 1 mm yr^{-1} used by Kirkby (1984, 1987) for sandstone was used. Creep rate (*K*) generally ranges from 10 to 100 cm^2 yr^{-1}, with the former typical of normal soil creep and the latter of periglacial solifluction. Therefore, the former was used for time intervals during the Holocene and the latter for intervals of glacial climates. That periglacial conditions existed during glacial times is strongly suggested by the Anderson Pond pollen record (Delcourt, 1979), located only 11 km ESE of the Cane Creek sites at a similar elevation.

The distance at which wash becomes greater than creep (*u*) also varies with climate. Generally, the main factor is the effect of vegetation cover, with distances being greater where vegetation cover is greater (i.e., humid climate) and lesser where the cover is lesser (i.e., dry climate). However, in the present setting the main climatic variation over time is temperature. Under periglacial conditions, because of the great increase in creep rate, we assumed that the effect of wash would be somewhat less. Therefore, we used 200 m for glacial climates and 50 m for interglacial climates. In the humid climate of Tennessee, it might be expected that *u* would be far longer than slope length. However, as shown by the presence of gullies on some of the slopes (Figure 3), the effect of wash has been significant. The above process rates were also ones that resulted in relatively good fits of the model profiles to the actual profiles.

The best results obtained for the three sites are shown in Figure 6. For the climate-sensitive parameters, a decision about what rates to use was made as follows. Preliminary runs showed the approximate model ages of the three profiles. Creep and wash rates were then assigned according to how much of the profile's age was during glacial times and how much during post-glacial times, with the boundary set at 15 kyr. Because the model age of the youngest hillslope (29.6 kyr; Figure 6A) fell about equally in each interval, values of K (50 cm^2 yr^{-1}) and u (100 m), intermediate between those of glacial and postglacial conditions, were used. For the intermediate hillslope (98.4 kyr; Figure 6B) and the oldest hillslope (330.2 kyr; Figure 6C), since the ages fell mainly in the Pleistocene, glacial-age values were used (K = 100 cm^2 yr^{-1} and u = 200 m). (For the oldest hillslope, this usage ignores the presence of interglacials during the time span. However, the 100 cm^2 yr^{-1} value produced a substantially better fit than did lower K values, and so was used despite this problem).

As Figure 6 shows, the best fit was obtained for the oldest profile (Figure 6C), and the fit for the youngest profile (Figure 6A) is also good. The fit for the intermediate profile (Figure 6B) is poorer than desired, but was the best that could be done using reasonable process rates.

To determine the colluvium thickness on the lower ends of the hillslopes associated with the abandoned meanders, seismic lines were run along slope on four profiles. However, signals proved to be severely attenuated in this loose material, and only *minimum* thicknesses could be obtained. These thicknesses were >7.9 and >9.4 m at A3, >8.8 m at A2, and >9.7 m at A1. These values establish the presence of thick colluvium on lower slopes, demonstrating that they are at least substantially depositional. The talus thicknesses indicated by seismic refraction are compatible with the modeling results for the younger two profiles, but not for the oldest, where most of the talus deposited earlier in the slope evolution is subsequently removed by erosion. A possible explanation for this discrepancy is that seismic velocity interpreted as talus here is, in fact, residuum.

The sensitivity of model profile age to changes in rate values was determined by repeatedly running the model using 4-6 different values for each parameter while holding the values of other parameters constant. The effect on both fit and model age was examined. This showed that, for the younger two profiles, by far the most important factor was the rate of free degradation α. For example, by increasing the rate from 1 to 5 mm yr^{-1}, the age of the youngest profile decreased from 29.6 kyr to about 6 kyr, and that of the intermediate profile decreased from 98.4 to about 30 kyr. Variation in the other rates had much less effect, however, with the age of the youngest profile showing changes of no more than about 15% and that of the intermediate profile no more than about 50%.

In contrast, for the oldest profile, since most erosion is accomplished by transport-limited rather than weathering-limited processes, the effect of variation in the free-degradation rate is relatively small. For example, increasing α from 1 to 5 mm yr^{-1} decreases age of the best-fit profile from 330.2 kyr to about 276 kyr, and decreasing it to as small as 10^{-6} mm yr^{-1} increases the age only to about 366 kyr. On the other hand, variation in some of the other parameters has somewhat greater effects than for the younger profiles. For example, decreasing creep rate K from 100 to 10 cm^2 yr^{-1} increases age about 60%, and decreasing wash distance u from 200 m to 50 m more than halves the age. The particular values chosen provide either the best fit or close to the best fit.

A model parameter of interest is the amount of material retained in the most-downslope cell. Presuming that the hillslope declines passively after abandonment of the meander, a large amount of hillslope debris would be expected to pile up at the base of the slope. In fact, however, retaining even several percent of the flux into the basal cell produces a profile that, because of the prominence of its footslope, matches the actual profile much more poorly than when the cell is set to retain none of the flux into it. (Retaining large percentages generally leads to model instability.) Therefore, all of the simulations reported here retained no sediment in the basal cell. A partial explanation of this finding may involve the process of meander abandonment. Rather than being a simple on-off switch, abandonment probably is a gradual process, with a progressive reduction in the frequency and size of flows through the meander loop while the cut-off course is being established. Even though the decreasing flows may not be sufficient to undermine the hillslope, they may still be capable of removing part of the debris shed by the declining slope. (Some evidence for this interpretation may be provided by the topographic profiles across the floors of cutoff valley reaches [Figure 7]. These indicate valley floors that are narrower than those of active valleys, suggesting that they may have been adjusted, before they were completely abandoned, to smaller flows than those typical of the active valley reaches.) In addition, basal debris could be removed by solution, and, even after total cessation of stream flow, fine material could also be removed from the base of the slope by wash.

As another check on the results of the modeling, the second profile at each of the three Cane Creek cutoffs was modeled using the same process rates and initial profiles as used for the first profile. Although fits were not as good, the ages of the best-fit model profiles were similar. For A3c, the age was 25.8 kyr (vs. 29.2 kyr for A3b); for A2b the age was 105.5 kyr (vs. 98.4 kyr); and for A1c the age was 385.9 kyr (vs. 330.2 kyr for A1b). This finding shows that the results are not significantly affected by small differences between individual profiles.

Assuming a uniform rate of downcutting for Cane Creek, age of the cutoffs (as given by age of the hillslope profiles) should be proportional to height AML. A plot of the former against the latter (Figure 8) thus allows a test of internal consistency of simulated ages, although of course this tells us nothing about the accuracy of ages. As Figure 8 shows, at least on logarithmic scales, the relationship is reasonably good.

CONCLUSIONS

The application of a hillslope evolution model allows several insights that otherwise would not have been possible. One finding is that high creep rates, closer to those of a periglacial than a temperate climate, are required to produce the upper convexities of the slope profiles developed on the abandoned meander walls. This suggests that hillslopes in the region have been strongly influenced by Pleistocene climates. Another result is the ability to compare the incision rate of 60 mm kyr^{-1} determined by Sasowsky *et al.* (1995) for the East Fork of the Obey River with those determined by slope modeling. If the slope profile associated with the 43-m high meander, for example, has an age of 330.2 kyr, a downcutting rate of 130 mm kyr^{-1} subsequent to abandonment is implied. Of course, this age is very approximate, but it would have to be increased to 716.7 kyr to yield the Obey River incision rate of 60 mm kyr^{-1}. Such an increase would require unreasonably low values of the transport-limited process rates. Hence, it appears probable that the incision rate of Cane Creek has been somewhat greater than that of the Obey River, although exactly how much faster cannot be specified with confidence.

A third insight concerns the process rates that need to be determined in the field to allow more precise modeling to be done. For young hillslopes, the free degradation rate is by far the most important to determine; creep, wash, and solution rates have much less effect on slope evolution. On the other hand, for old hillslopes, the free degradation rate is not very important, but the rates of creep, wash, and solution become critical. Determination of modern rates, however, is not sufficient, as the effect of Pleistocene periglacial climate on hillslopes appears to be strong. Pleistocene rates might be approximated by determining volumes of late-glacial hillslope deposits between dated deposits.

A fourth insight concerns the amount of rock debris on the foot of the slope. Both the model slope and the actual slope have far less talus than would be expected from simple passive decline after abandonment. Possible explanations for this finding are removal of debris during a gradual process of abandonment, removal of debris in solution, and removal of fine debris by wash to points distant from the base of the slope.

REFERENCES CITED

Delcourt, H. R., 1979, Late-Quaternary vegetation history of the eastern Highland Rim and adjacent Cumberland Plateau of Tennessee: Ecological Monographs, v. 49, p. 255-280.

Kirkby, M. J., 1971, Hillslope process-response models based on the continuity equation, in Brunsden, D. (ed.), Slopes: form and process: London, Institute of British Geographers, Special Publication 3, p. 15-30.

Kirkby, M. J., 1973, Landslides and weathering rates: Geologia Applicata e Idrogeologica, Bari, v. 8, p. 171-183.

Kirkby, M. J., 1984, Modelling cliff development in South Wales: Savigear reviewed: Zeitschrift für Geomorphologie, v. 28, p. 405-426.

Kirkby, M. J., 1987, General models of long-term slope evolution through mass movement, in Anderson, M. G., and Richards, K. S., Slope stability: London, Wiley & Sons Ltd, p. 359-379.

Kirkby, M. J., 1992, An erosion-limited hillslope erosion model, in Schmidt, K.-H., and de Ploey, J., eds., Functional geomorphology: landform analysis and models: Catena Supplement no. 23, p. 157-187.

Kirkby, M. J., Naden, P.S., Burt, T. P., and Butcher, D. P., 1992, Computer simulation in physical geography: Chichester, Wiley: p. 85-90.

Mills, H. H., and Mills, R. T., 1997, Evolution of undercut slopes on abandoned incised meanders in the Eastern Highland Rim of Tennessee, USA (under review).

Reesman, A. L., and Godfrey, A. E., 1981, Development of the Central Basin of Tennessee by chemical denudation: Zeitschrift für Geomorphologie, v. 25, p. 437-456.

Sasowsky, I. D., White, W. B., Schmidt, V. A., 1995, Determination of stream-incision rate in the Appalachian plateaus by using cave-sediment magnetostratigraphy: Geology, v. 23, p. 415-418.

Skempton, A. W., 1964, The long-term stability of clay slopes: Geotechnique, v. 2, p. 75-102.

Wilson, C. W., Jr., and Marcher, M. V., 1968, Geologic map and mineral resources summary of the Burgess Falls quadrangle, Tennessee: State of Tennessee Division of Geology, GM 326-SE and MRS 326-SE.

Table 1. Particle-size analysis and estimated angle of internal friction

Slope profile	% gravel	% sand	% silt	% clay	Estimated ϕ
A1b	50	12	31	7	38°
A2a	56	17	26	1	38°
A2b	62	9	23	6	42°
A3b	67	10	18	5	43°
A3c	53	10	31	6	38°

The ϕ values were estimated from the triangular diagram in Kirkby (1973, his Figure 5) relating ϕ to particle-size distribution.

FIGURE CAPTIONS

Figure 1. Location map of study area. Numbers 1-12 show locations of studied abandoned meanders; A shows location of stream valley with incision rate determined by Sasowsky *et al.* (1995). Quadrangle and stream names are as follows: 1-3, Burgess Falls quadrangle, Cane Creek; 4, Burgess Falls quadrangle, Falling Water River; 5-6, Dodson Branch quadrangle, Blackburn Fork; 7-9, Windle quadrangle, Roaring River; 10, Riverton quadrangle, Obey River East Fork; 11-12, Moodyville quadrangle, Wolf River.

Figure 2. Oblique aerial photograph of large cutoff meander on Eastern Highland Rim (location 4 on Figure 1). Width of meander floor is about 100 m.

Figure 3. Map showing area of concentrated study along Cane Creek. U indicates profiles on actively undercut slopes and A indicates profiles on undercut slopes of abandoned meanders. Meander A1 is 43 m above the modern stream level, meander A2 is 14 m above, and meander A3 is 2 m above. Grid squares are 1-km on a side. Eastings and northings are for UTM Zone 16.

Figure 4. Profiles on abandoned undercut slopes at Cane Creek, showing geological contacts. Locations of profiles are shown on Figure 3.

Figure 5. Profiles on actively undercut slopes at Cane Creek, showing geological contacts. Locations of profiles are shown on Figure 3.

Figure 6. Comparison of assumed original profile, modern profile, and best-fit profile produced by modeling, for the three abandoned meanders along Cane Creek. For this modeling, all profiles were adjusted to the same horizontal length of 150 m. As all surveyed profiles were less than 150 m, this was done by extending the convex slope at the top of the profile. The following rates were the same for all three of the shown simulations: solution rate (r_s) = 50 μ yr^{-1}, landslide threshold angle (g_ϕ) = 23°, talus angle (g_t) = 35°, and rate of free degradation above threshold (α) = 1 mm yr^{-1}. A. Lowest cutoff meander (A3). Creep/solifluction rate (K) = 50 cm^2 yr^{-1}, distance at which wash becomes greater than creep (u) = 100 m. The mean absolute difference between the best-fit model profile and the actual profile is 1.66 m and the model age is 29.6 kyr. B. Intermediate cutoff meander (A2). Creep/solifluction rate (K) = 100 cm^2 yr^{-1}, distance at which wash becomes greater than creep (u) = 200 m. The mean absolute difference between the best-fit model profile and the actual profile is 3.76 m and the model age is 98.4 kyr. (None of the actively-undercut profiles were suitable to use as an initial profile here, because immediately above the intermediate cutoff the Rim surface is unusually low (about 10 m lower than elsewhere), so that the actively-undercut profiles are higher than the original profile actually was. To produce a more reasonable initial profile, the upper part of U1 was lowered about 10 m.) C. Highest cutoff meander (A1). Creep/solifluction rate (K) = 100 cm^2 yr^{-1}, distance at which wash becomes greater than creep (u) = 200 m. The mean absolute difference between the best-fit model profile and the actual profile is 0.66 m and the model age is 330.2 kyr.

Figure 7. Profiles across floors of abandoned meanders. Note vertical exaggeration is greater than on Figures 4 and 5.

Figure 8. Plot of hillslope ages estimated by modeling vs. relative age of hillslopes indicated by height of associated meander floor above modern stream.

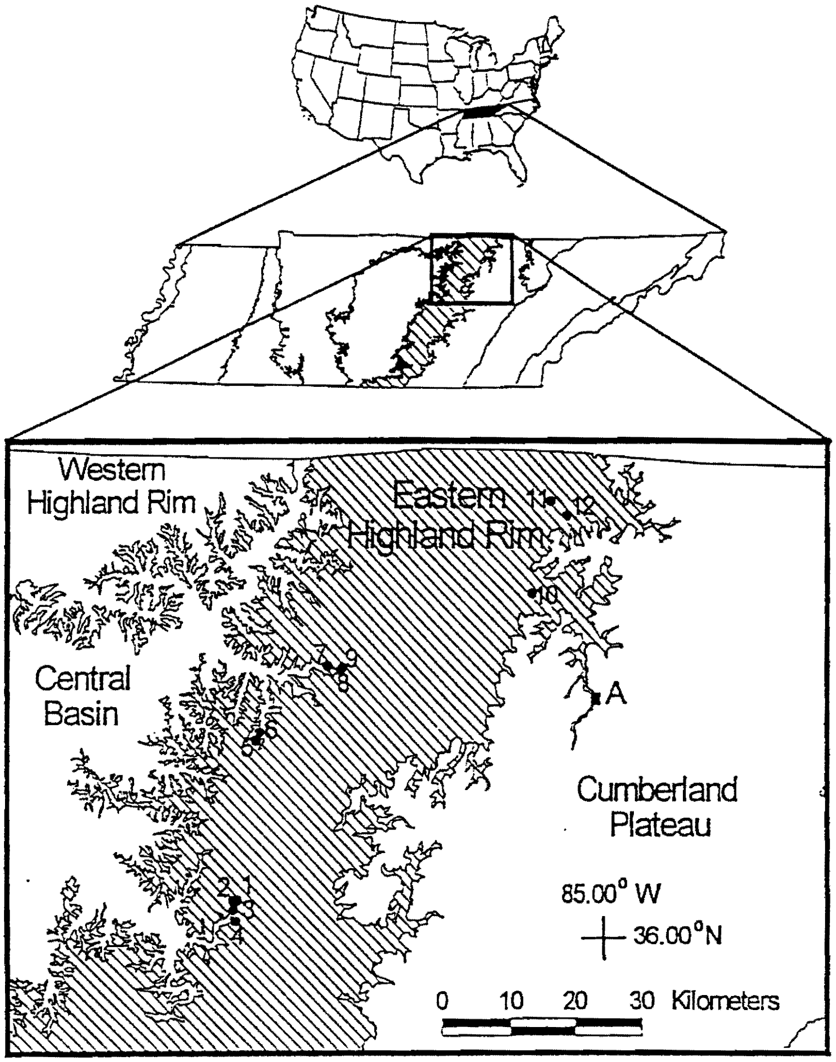

Western
Highland Rim
Eastern
Highland Rim
Central
Basin
Cumberland
Plateau
85.00° W
36.00°N
0 10 20 30 Kilometers
A

Fig. 1

Fig. 2

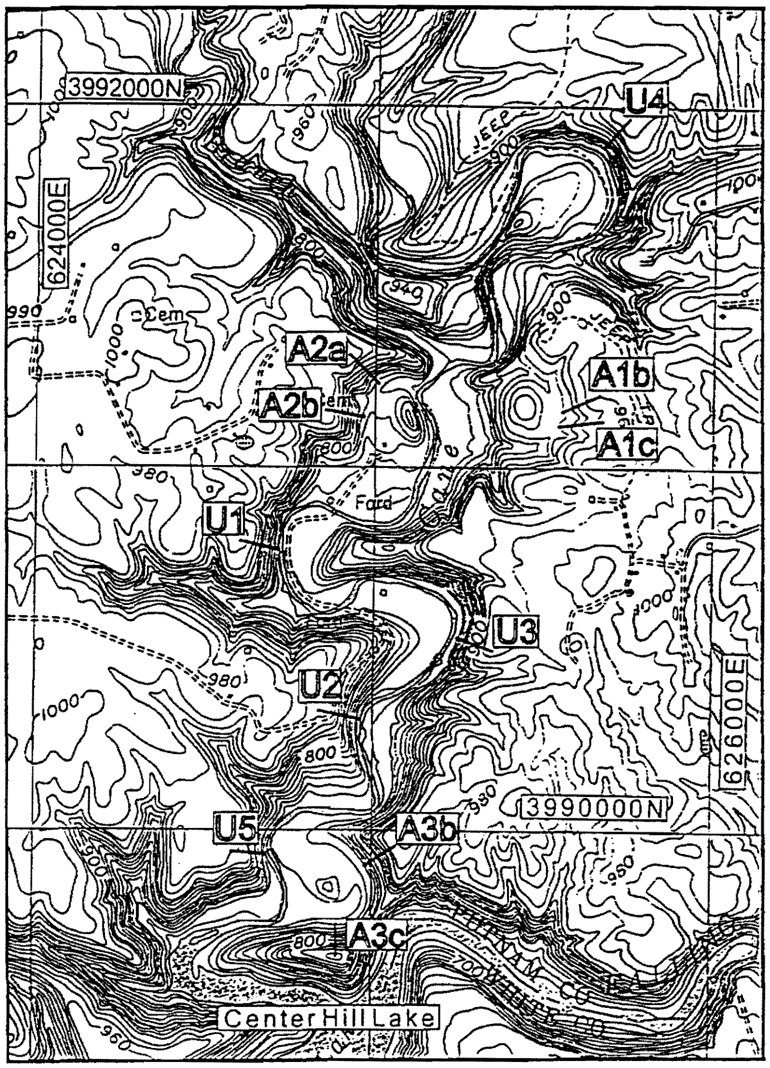

3992000N
624000E
U4
A2a
A2b
A1b
A1c
U1
Ford
U3
U2
626000E
3990000N
U5
A3b
A3c
Center Hill Lake

Fig. 3

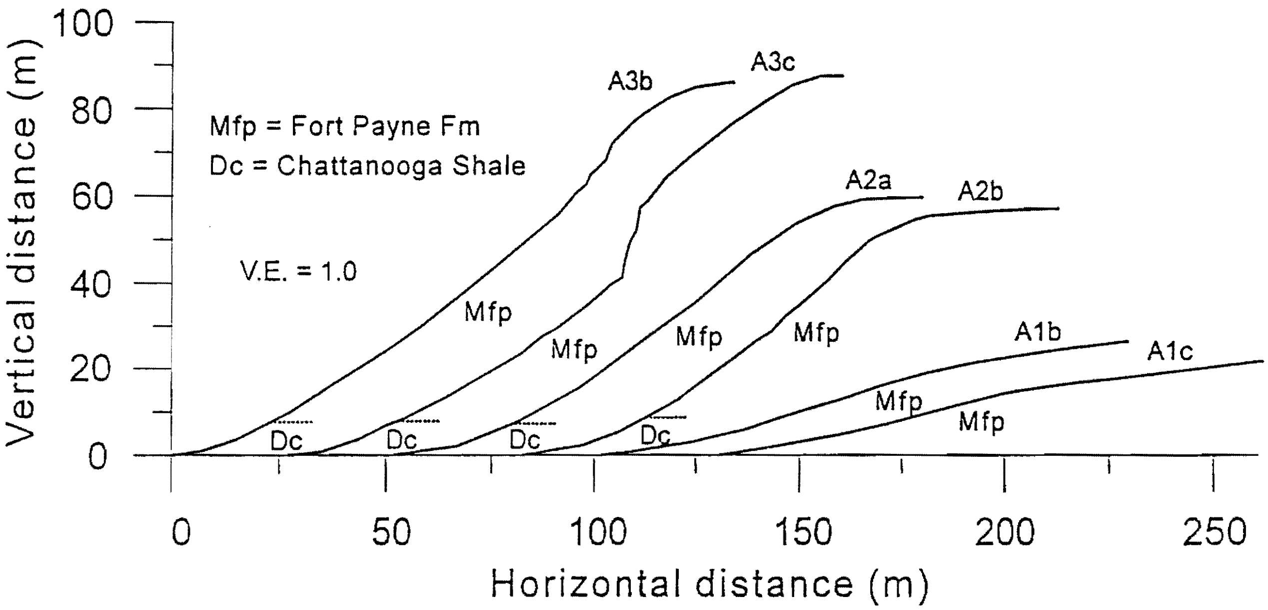
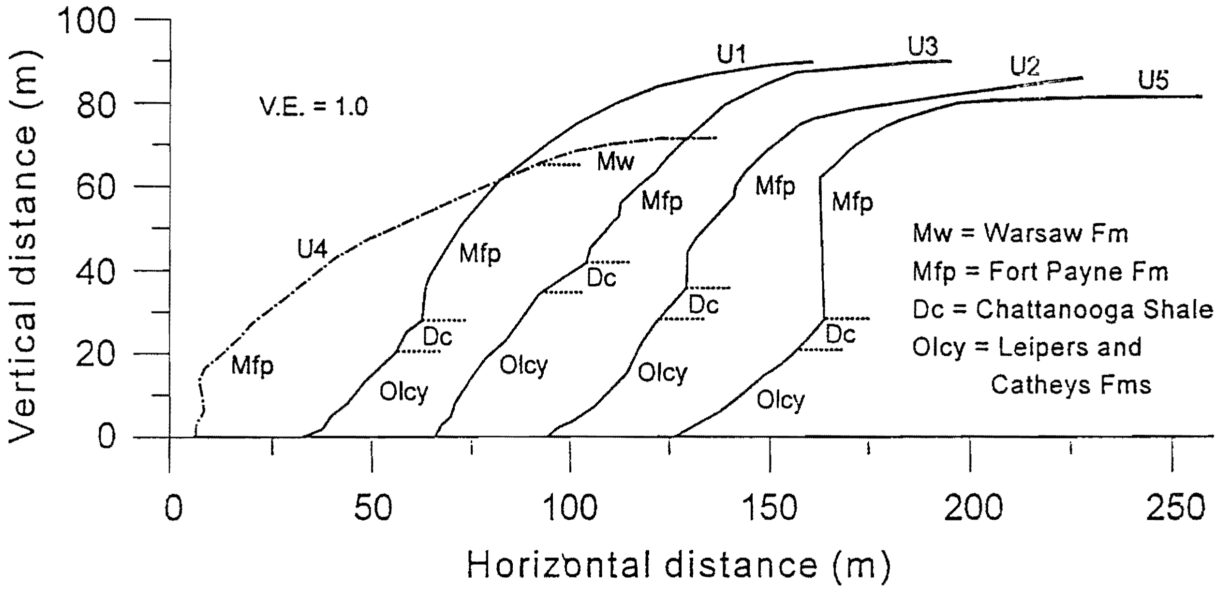
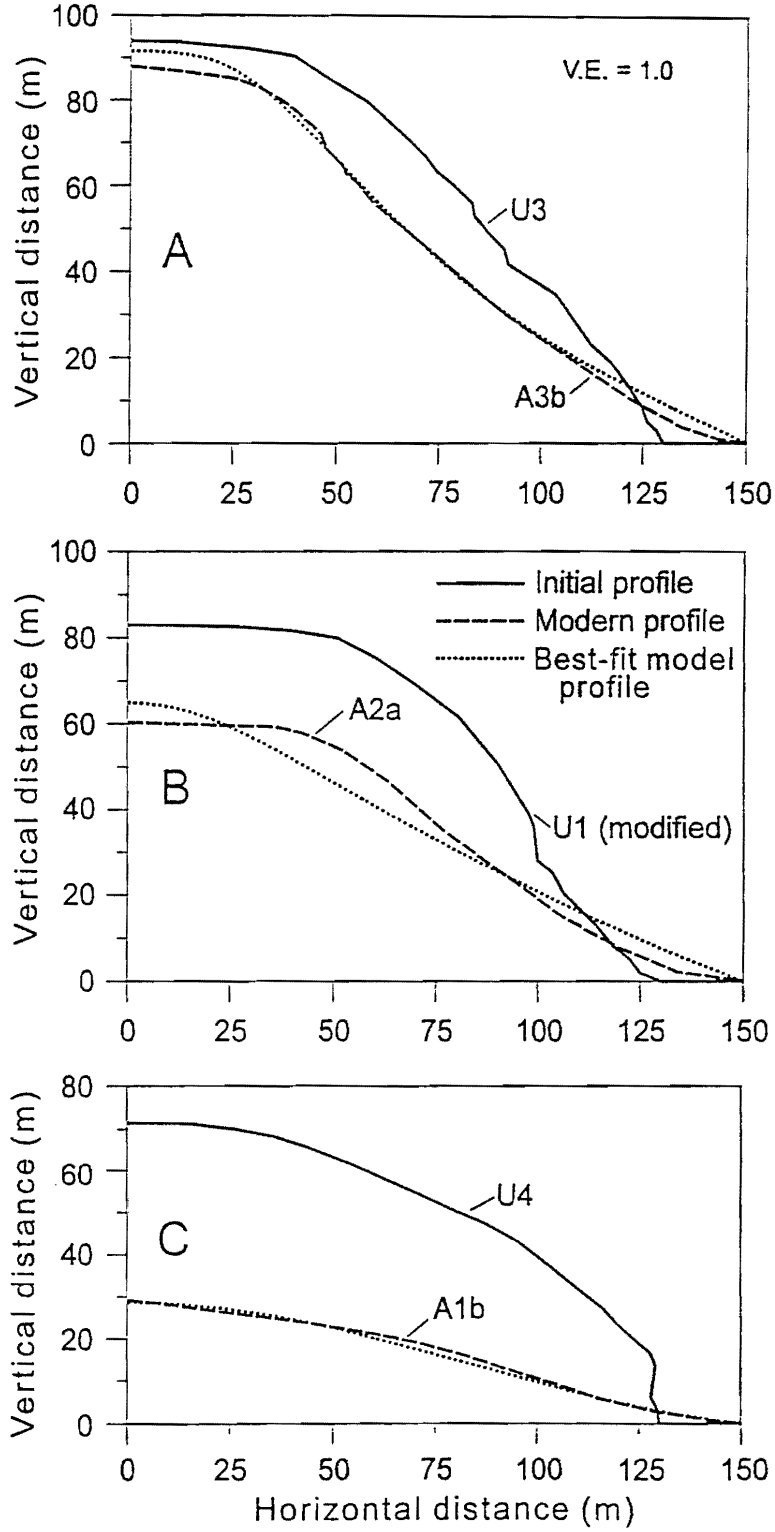

Vertical distance (m)
100
80
60
40
20
0
Mfp = Fort Payne Fm
Dc = Chattanooga Shale
V.E. = 1.0
A3b
A3c
A2a
A2b
A1b
A1c
Mfp
Dc
0
50
100
150
200
250
Horizontal distance (m)

Fig. 4



100
80
60
40
20
0
Vertical distance (m)
V.E. = 1.0
U1
U3
U2
U5
U4
Mw
Mfp
Dc
Olcy
Mw = Warsaw Fm
Mfp = Fort Payne Fm
Dc = Chattanooga Shale
Olcy = Leipers and
Catheys Fms
0
50
100
150
200
250
Horizontal distance (m)

Fig. 5



100
80
60
40
20
0
Vertical distance (m)
V.E. = 1.0
A
U3
A3b
0
25
50
75
100
125
150
100
80
60
40
20
0
Vertical distance (m)
Initial profile
Modern profile
Best-fit model profile
B
A2a
U1 (modified)
0
25
50
75
100
125
150
80
60
40
20
0
Vertical distance (m)
C
U4
A1b
0
25
50
75
100
125
150
Horizontal distance (m)

Fig. 6

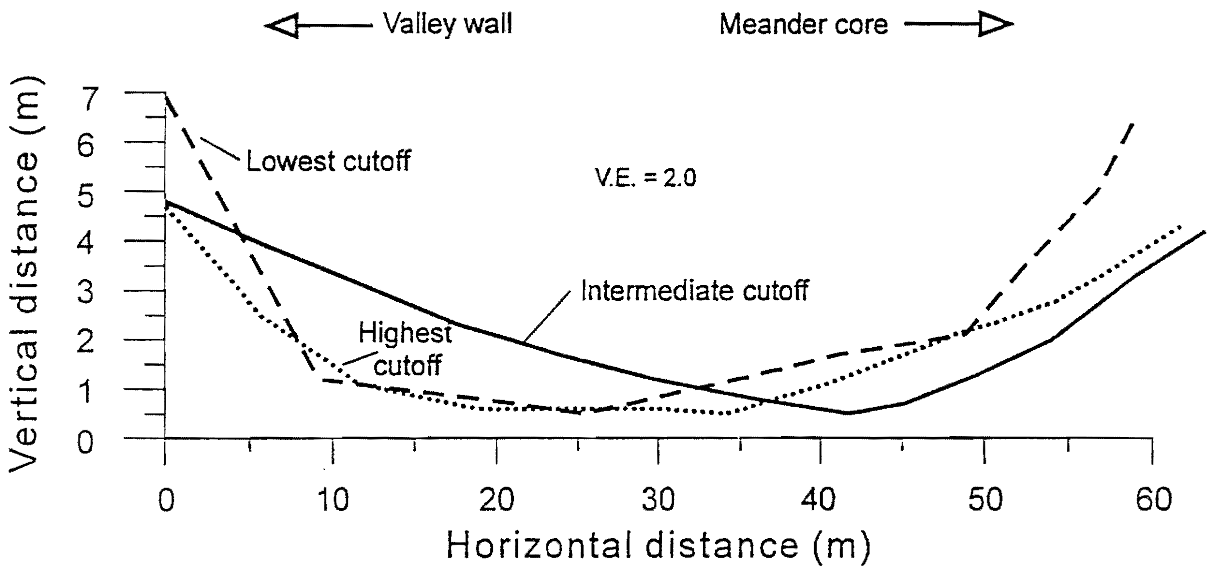

Valley wall
Meander core
Vertical distance (m)
7
6
5
4
3
2
1
0
Lowest cutoff
V.E. = 2.0
Intermediate cutoff
Highest cutoff
0
10
20
30
40
50
60
Horizontal distance (m)

Fig. 7

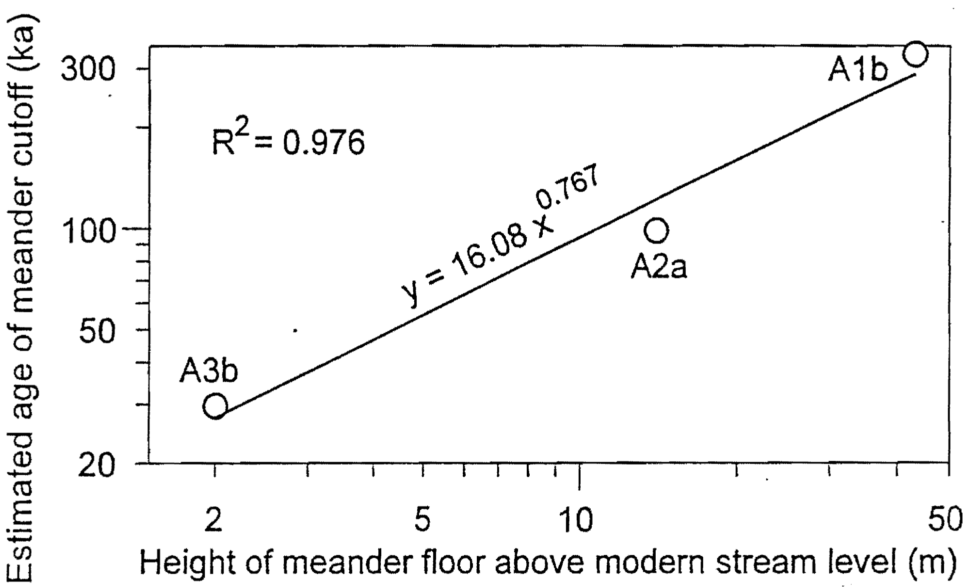

Estimated age of meander cutoff (ka)
300
100
50
20
R^2 = 0.976
y = 16.08 x^0.767
A1b
A2a
A3b
2
5
10
50
Height of meander floor above modern stream level (m)

Fig. 8

APPENDIX A:

HILLSLOPE DYNAMICS SIMULATOR

DOCUMENTATION

Richard's n-store Hillslope Dynamics Simulator (HDS) version 1.1 documentation

Contents

Richard's n-store Hillslope Dynamics Simulator (*HDS* for short) simulates the evolution of hillslope profiles through time using a simple linear-store model based on the mass balance equation. It utilizes a fully graphical user interface, and allows real-time visualization of evolving hillslopes.

Below are links to documentation for this release.

GNU GENERAL PUBLIC LICENSE
Version 2, June 1991

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

GNU GENERAL PUBLIC LICENSE
TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program"

means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY

FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

```
<one line to give the program's name and a brief idea of what it does.>
Copyright (C) 19yy  <name of author>

This program is free software; you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation; either version 2 of the License, or
(at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program; if not, write to the Free Software
Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
```

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

```
Gnomovision version 69, Copyright (C) 19yy name of author
Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type `show w'.
This is free software, and you are welcome to redistribute it
under certain conditions; type `show c' for details.
```

The hypothetical commands `show w' and `show c' should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than `show w' and `show c'; they could even be mouse-clicks or menu items--whatever suits your program.

```
You should also get your employer (if you work as a programmer) or your
school, if any, to sign a "copyright disclaimer" for the program, if
necessary.  Here is a sample; alter the names:

  Yoyodyne, Inc., hereby disclaims all copyright interest in the program
  'Gnomovision' (which makes passes at compilers) written by James Hacker.

  <signature of Ty Coon>, 1 April 1989
  Ty Coon, President of Vice

This General Public License does not permit incorporating your program into
proprietary programs.  If your program is a subroutine library, you may
consider it more useful to permit linking proprietary applications with the
library.  If this is what you want to do, use the GNU Library General
Public License instead of this License.
```

Introduction

Richard's n-store Hillslope Dynamics Model (*HDS*) is a fully 32-bit, Windows95/NT application for simulating the evolution of hillslope profiles over time. It implements a relatively simple linear-store model of hillslope evolution based on the mass-balance, or continuity, equation, which it "solves" by approximation with finite differences. The program has a fully graphical user interface and allows real-time visualization of the evolving hillslope, as well as output of numerical data to ASCII files. It models the effects of landslides, wash, and creep/solifluction/rainsplash using empirical process laws.

HDS is written entirely in Microsoft Visual Basic 5 (most of the code should be backwards compatible). I made a conscious effort to make the code as well-commented and readable as possible, so that it can be easily altered by others. All of the source-code is freely available. The program executable can be run without a copy of the Visual Basic environment, but the program can be much more useful if it is run from within the Visual Basic environment (see Appendix B).

What it does:

HDS requires two ASCII input files: one specifying the process rates that will be used in a model run, and one describing the initial geometry of the hillslope profile to be "evolved." A third ASCII profile, specifying the geometry of a "target" hillslope profile that we hope to match by evolving the initial profile, is optional. The process rates file can be written in *HDS*'s proprietary .par format, or .ev files from M. J. Kirkby's (Kirby*et al.*, 1992) SLOPEN program can be imported. Initial and target hillslope profile's can be read from two column text files specifying (x,y) coordinates of points along the profile, or they can be imported from the .ev files using Kirkby's percent of slope length vs. percent of slope height system.

The program breaks the hillslope into a series of equally spaced cells, with the storage in each cell representing the elevation at a point on the hillslope. For each iteration of the model, the sediment fluxes into and out of each cell are calculated from empirical process laws using the various rates specified by the user, and the elevation of each cell is adjusted accordingly. The model is moved forward in time using a fixed time step that is specified by the user. If the user has chosen to use a target profile, a comparison of the evolving hillslope profile and the target one is made at each iteration, in order to determine the best fit.

The user is given several different output options. The evolving hillslope profile can be redrawn continuously, or successive profiles can be overlaid at user-specified intervals. The user can choose to write the evolved profile to disk at given intervals, and the current state of the evolving hillslope can be dumped to disk at any time by clicking the appropriate button. If a target slope profile has been specified, when a model run is terminated, the user is given the option of saving the best fit profiles to disk.

The Hillslope Model

The model implemented in this program is based largely on work by M. J. Kirkby (1971, 1984, 1987, 1992, and unpub. data, 1991; Kirkby *et al.*, 1992) of Leeds University. The essentials of the model are as follows. The hillslope profile is divided into a series of equally spaced cells, with the storage in each cell representing the elevation at a point on the hillslope. Between each time step, sediment fluxes into and out of each cell are calculated from empirical process laws, and from these the accompanying changes in the elevation of each cell are determined. Climate and lithology are held constant, so process rates depend largely upon the slope topography; i.e., distance from the divide and downslope gradient. A fixed time step of small enough size to prevent numerical instabilities is used.

"Creep" includes a group of processes which depend on gradient but not on collecting area, and have no lower threshold. In the present setting it consists mainly of soil creep and solifluction. Creep is assumed to carry sediment at a rate directly proportional to the downslope gradient. "Wash" refers to overland flow that able to entrain and carry soil particles on the surface are grouped together as "wash." Unlike creep, wash depends on collecting area (i.e., distance from the divide) as well as on gradient. The sediment flux S out of a cell from creep and wash processes combined is given by (Kirkby et al., 1992):

$$S = K[1 + (x/u)^2]g$$

where x is the distance from the divide, g is the downslope gradient, K is a constant giving the rate of creep, and u is the distance in meters beyond which the wash term, $K(x/u)^2g$, becomes larger than the creep term.

Landslides are modeled as a continuous process; hence the sediment flux due to landslides represents a long-term average, rather than individual slides. The use of these average rates assumes that individual slides are small enough not to change the slope profile significantly. The flux due to landslides is controlled by four parameters, which are discussed in more detail in Kirkby (1984, 1987). Two of these are thresholds: a lower, stable gradient g_φ below which there is no landslide activity, and an upper gradient g_t above which slides will never come to rest. The first depends on the angle of internal friction and whether pore pressure can develop. Likely values range from 0.14 (8^o) for clays up to about 0.58 (30^o) for some sandstones. The second may usually be related to the talus angle of repose of 0.7 (35^o). The third is a rate constant α which governs the rate of free degradation, or unconstrained lowering, D which is given by

$$D = \alpha g(g - g_\phi).$$

α may range from 0.001 m yr^{-1} for sandstones to as much as 10 m yr^{-1} for clays (Kirkby, 1987). The fourth parameter h_0 indicates the average height from which blocks fall from cliffs, which should be roughly their mid-height. It is in a sense used to represent the momentum of the falling blocks in the expression for the mean horizontal distance h traveled by the moving material:

$$h = h_0/(g_t - g).$$

The value of h_0 influences how far a slide can run out across gently sloping ground at the base of a slope, but generally has only a slight effect on the slope profile elsewhere. Combining the expressions for detachment rate and travel distance, the sediment flux S_i out of cell i due to landslides is given by (Kirkby *et al.*, 1992)

$$S_i = \frac{Ddx + S_{i-1}}{1 + (1/h)dx}$$

where *dx* is the spacing between cells, and S_i - 1 is the slide flux out of cell *i* - 1.

Solution is modeled in two ways. Kirkby (1991, unpub. data; et al., 1992) modeled the sediment flux out of a cell due to solution as being linearly proportional to the distance x from the divide:

$$S = sol \cdot x$$

Where *sol* is the rate constant governing Kirkby-type solution. In Kirkby's 1991 and 1992 programs, solution does not operate unless the downslope gradient exceeds 0. In HDS, we allow the user to specify the "solution gradient" which the downslope gradient must exceed in order for Kirkby-type solution to operate. We do this because often the downslope gradient is incredibly close to 0, but is just ever so slightly greater, leading to operation of Kirkby-type solution where it should not really be occurring. Also, in Kirkby's 1991 and 1992 programs, the sediment outflux due to solution is included with the sediment influx into the cell below; in effect, sediment can build a "talus." This seems unrealistic, so we instead assume that dissolved material is almost immediately carried out of the system, and therefore is *not* included as part of the sediment influx into the lower neighboring cell.

Experimentation with the model using Kirkby-type solution alone often yielded unsatisfactory results. Certainly, the rate of solution should somehow increase with distance from the divide, but the linear relationship seems to give poor results. Additionally, at least some sort of solution should occur around the divide as well. Hence, we also modeled solution as a rate of uniform vertical lowering; with each iteration of the model, each cell is lowered by an amount determined by

$$dz = -r_s dt$$

where *dz* is the change in elevation of the cell, r_s is the rate of solution, and *dt* is the time step used. Again, this type of solution does not contribute to the flux of sediment being transported to cells downslope; rather, it is assumed that any material freed by solution immediately leaves the system.

For each iteration of the model, sediment fluxes out of each cell (and hence into the adjacent cell downslope) due to creep, landslides, and (optionally) Kirkby-type solution are calculated. These are then grouped together into a total sediment outflux for each cell, and then converted into the resultant changes in elevation. The basis for this is the mass-balance, or continuity equation, which may be written as

$$\frac{\partial z}{\partial t} = \frac{\partial S}{\partial x}$$

where *S* is the downslope flux of sediment and *z* the elevation at distance *x* from the divide, and *t* is the elapsed time. In more concrete terms, the elevation changes are determined from the expression:

$$\frac{\partial z}{\partial t} = \frac{S_{i-1} - S_i}{\partial x}$$

dependS_i is the sediment flux out of cell *i*, and S_{i-1} the flux out of cell *i*-1 and, hence, the flux into cell i (actually, if one is using Kirkby-type solution, the flux due to it must be subtracted from S_{i-1}). Once the rate of elevation change ∂z/∂t due to downslope sediment transport for a cell is calculated, the change in elevation is determined by multiplying by the time step dt. Combining this rate of elevation change with that due to solution, the explicit expression for the elevation of a cell can be written:

$$z_{t+dt} = z_t + \left(\frac{\partial z}{\partial t} - r_s\right)dt$$

where z_t is the elevation of a cell at time t.

What's really going on -- a "pseudocode" explanation of the program

For those of you who would like some more concrete details of the program's operation, here's an something close to a pseudocode explanation of what the program does (Kirkby-type solution is left out of this explanation, but it's easy to figure out how it would fit in):

GUTS OF THE HILLSLOPE MODEL

For each iteration, the computer calculates sediment fluxes out of each cell, beginning with the uppermost cell and ending with the lowest one. Excluding the special cases used at the top and bottom of each cell, for cell i (see the paper for explanations of the variables):

1) The sediment flux due to creep and wash is calculated. This flux is *independent* of the flux upslope. The expression used is:

$$S = K[1 + (x/u)^2]g$$

2) The sediment flux due to landslides is calculated. This flux is *dependent* on the landslide flux upslope. The expression used is:

$$S_i = \frac{Ddx + S_{i-1}}{1 + (1/h)dx}$$

where S_i is the flux due to landslides out of cell i, and S_{i-1} is the flux due to landslides out of cell i-1, and hence into cell i.

Once the fluxes have been calculated for each cell, **a total sediment outflux for each cell is calculated by simply adding the fluxes due to creep/wash and landslides together**. Now changes in elevation due to these fluxes can be determined by the continuity equation. The rate of change in elevation is given by the expression:

$$\frac{\partial z}{\partial t} = \frac{S_{i-1} - S_i}{\partial x}$$

Note that in the above equation, the S's are *total* sediment outfluxes, i.e. sums of the fluxes due to creep/wash and landslides. To get from $\partial z/\partial t$ (due to processes *other* than solution) to actual changes in elevation with a timestep, one just multiplies by the timstep dt. That is, the change in elevation $dz_{non\text{-}solution}$ (due only to processes other than solution) with timestep dt is

$$dz_{non\text{-}solution} = \frac{\partial z}{\partial t} dt$$

And since the elevation change $dz_{solution}$ due to solution is given by

$$dz_{solution} = -r_s dt$$

We find that the total change in elevation of a cell must be given by the sums of these, viz.

$$dz = dz_{non\text{-}solution} + dz_{solution}$$

Which, substituting, is written

$$dz = (\frac{\partial z}{\partial t} - r_s)dt$$

And since the elevation z_{t+dt} of a cell at time $t + dt$ must be equal to the cell's elevation at time t plus the change in elevation that happens between the two times, we may write the cell's elevation

$$z_{t+dt} = z_t + (\frac{\partial z}{\partial t} - r_s)dt$$

which is in fact the explicit expression for the elevation of a cell from the preceding section. Once the new elevation of each cell has been determined from the this equation, a new iteration begins.

Using the Program

The basics (a very abbreviated overview)

Hopefully, the user interface is intuitive enough to allow users to figure things out, but, just in case, I'll specify the basic steps in performing a model run. (Try running with some of the example files included with the program, if you'd like):

1) Creating a file that contains the process rates that you want to use (either a .par or a .ev file).
2) Creating a file that specifies the initial geometry of the hillslope profile you wish to evolve.
3) (Optional) Creating a file that specifies the geometry of the target hillslope profile you want to try to match.
4) Specifying the input filenames in the text boxes in the Main window.
5) Specifying the program options you desire in the Options window.
6) Starting the model run by pressing the "Run Model" button.

Now, provided that all of the filenames you specified exist the model should start running. When it's running you can do several things by clicking the buttons in the "Model Running..." window that pops up.

- You can show and hide various program windows by clicking the appropriate checkboxes on the left side of the "Model Running" window.
- You can pause the model by clicking the "Pause model" button in the "Model Running..." window. You are then presented with the option to write a snapshot of the current profile to disk.
- You can pause the model and display the best fit profiles by clicking the "Pause and display best fit profiles" button. The profile shown in blue is the best fit as determined by absolute differences, while the one in green is that determined by squared differences. (More on how the best fits are determined later).
- You can terminate the model run by clicking the "Terminate model run" button. If you are using a target profile for comparison, you are given the option of saving the best fit profiles to disk at this time.

While the model is running, it's also possible to change the timestep, process rates, etc., by changing the values in the textboxes in the "Program Options" and "Model Parameters" window. However, if you really need to change these while the model is running, it's best to do so within the Visual Basic environment (see Appendix B for details).

If you'd like to work with the graphical output displayed in the "Hillslope profile view:" window, it's possible to write it's current state to the Windows clipboard by clicking the mouse on it and pressing [Alt]-[Print Screen]. (This works for all windows applications.) More on this later.

Below you'll find a much more detailed explanation of how to use the program:

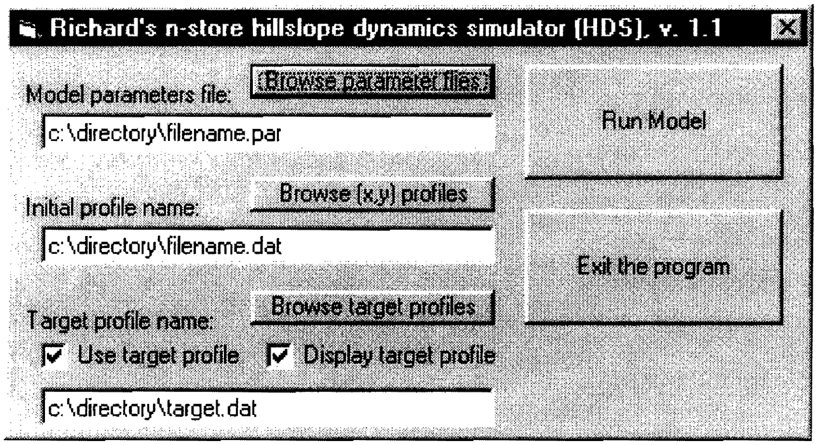
The details

- Part I: A window by window explanation of the program interface.
- Part II: An explanation of the file formats used by the program.

Part I: A window by window explanation of the program interface.

The main program window

The main program window is the first thing that is displayed when the program initially loads and when a model run is completed. It is from this window that you specify which input files to use, start a model run, and exit the program.



Specifying the input files:

There are **three text boxes** present in the window--these are where the names of the input files to be used are specified.

- The model parameters file -- This is the file that contains the process rates to be used in a model run. It can be in .ev or .par format. If you are using a .par format file, it is possible to specify the other input files to be used within the .par file itself (to learn how this works, see the Understanding and writing the program input files section).
- The initial profile name -- This is the file that specifies the geometry of the initial profile to be evolved. This can be a .dat or a .ev file.
- The target profile name -- This is the file that specifies the geometry of the hillslope profile with which you are comparing the initial profile's evolution. This can be a .dat file or a .ev file. The use of a target profile is optional.

There are two ways to specify the names of the input files to be used. The first and most direct method is to select the approriate text box by left-clicking on it, and then typing in the path and name of the file to be used. However, if you don't know this information (or would simply prefer not to type it), it is easier to use the second method: click on the appropriate "Browse [whatever]" to bring up Window's standard Open dialog box.

Choosing the target profile options:

There are **two checkboxes** present above the name of the target profile file.

- The "Use target profile" checkbox -- Put a check in this box if you intend to use a target profile in

your model run. If this box is not checked, then you don't have to specify a filename in the "target profile name" text box.
- The "Display target profile" checkbox -- Put a check in this box if you will be using a target profile and you want it to be displayed in the window in which the evolving hillslope is shown.

Running the model:

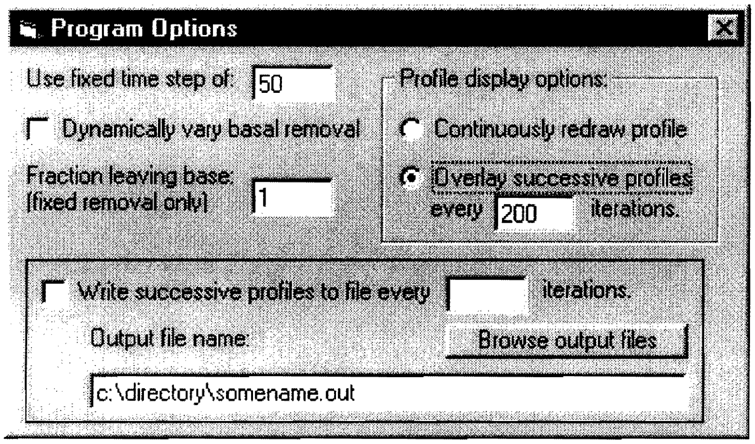
Once you have specified the names of the input files and have selected the options that you wish to use, press the "Run Model" button to start the run.

Exiting the program:

When you are finished using the program and wish to exit, click the "Exit the program" button or the Windows close gadget to end the program.

The Program Options window

The Program Options window is where you specify several different program options.



The size of the fixed timestep -- You must specify the size of the time step (in years) that the model advances by with each interation. Too large a timestep leads to numerical instabilities (further explanation is provided in the technical details section).

Dynamically vary basal removal -- If this option is enabled, an attempt is made by the program to adjust the sediment outflux in the last cell by extrapolating from the outfluxes of the cells immediately above. This is an experimental feature, and, in my judgement, usually gives unsatisfactory results. See the technical details section for more information.

Fraction leaving base -- If the "Dynamically vary basal removal" option is not selected, then the a fixed proportion of the sediment entering the basal cell is removed with each iteration. This proportion ranges between 0 and 1. Setting this proportion anywhere much below one usually yields an unnaturally large buildup of sediment at the base and leads to numerical instabilities.

Profile display options -- You can choose between two options for displaying the hillslope profile as it evolves.

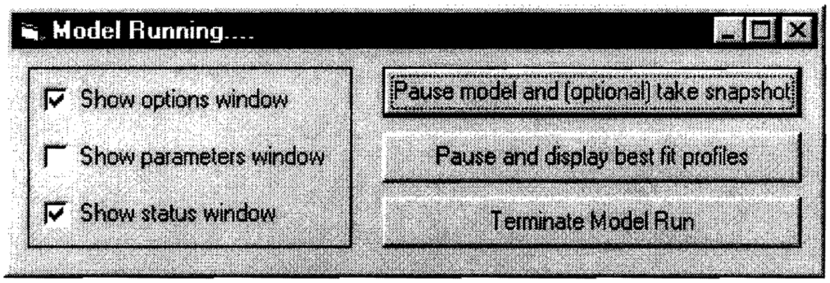
- Continuosly redraw profile -- If this option is selected, then the profile will be redrawn after every

iteration. This slows down the model a bit because of the time consumed by freqently redrawing.)
- Overlay successive profiles every *x* iterations-- If this option is selected, the hillslope profile is drawn over any previous profiles every *x* iterations. This is somewhat faster than continuously redrawing, and allows the user to more easily see exactly how the profile evolves over time.

Writing successive profiles to file every *x* iterations -- If this option is enabled, the output file specified will be created, and every *x* iterations, the profile will be written to this file in (x,y) coordinates. (Be careful with your disk space when using this option.)

The Model Running.... Window

This window is what you control the program from when a model run is in progress.



Showing and hiding program windows -- The three checkboxes on the left allow you to show and hide the options, parameters, and status windows while a model run is in progress. Just click to hide/show a window.

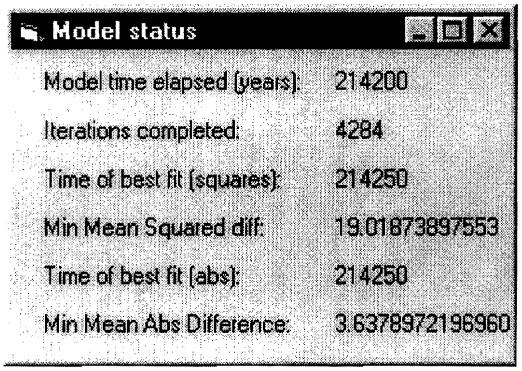
Pause the model and (optional) take snapshot -- Clicking this button pauses the model and gives you the option of taking a snapshot of the profile's current state. When you take a "snapshot," the current hillslope profile is dumped to a disk file in (x,y) coordinates along with the number of years elapsed and interations completed. The program will prompt you for the file name to use.

Pause and display best fit profiles -- Clicking this button pauses the model and displays the current best fit profiles in the "Hillslope profile view:" window. The best fit as determined by absolute differences is drawn in green, while that determined by differences of squares is drawn in blue. The target profile is also drawn, in red. (Note that the if the best fit profiles are perfectly coincident, only a green best fit profile will be drawn. This is because the best-fit profile determined by absolute differences is drawn after the best-fit profile determined by squared differences.)

Terminate Model Run -- Clicking this button terminates the current model run. When you do so, you are given the choice of saving the best fit information to a file. If you choose yes, the times of best fits, iterations and years elapsed, the minimum mean absolute and squared differences, and the best fit profiles are written to disk (the program will prompt you for the file name). Once the model run is terminated, all of the program variables are reset and another run can be started from the main window.

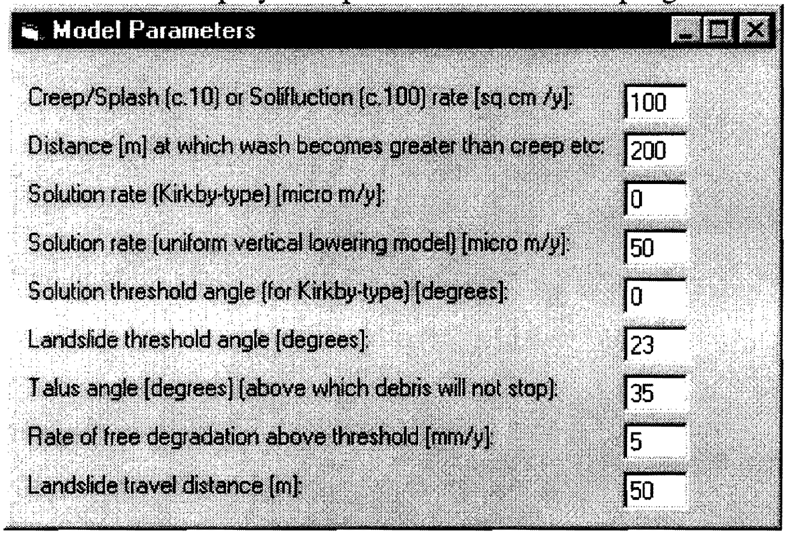
The Model status window

This window displays various data about the current state of the model. For an explanation of the information regarding best fits, see Appendix A.



The Model Parameters window

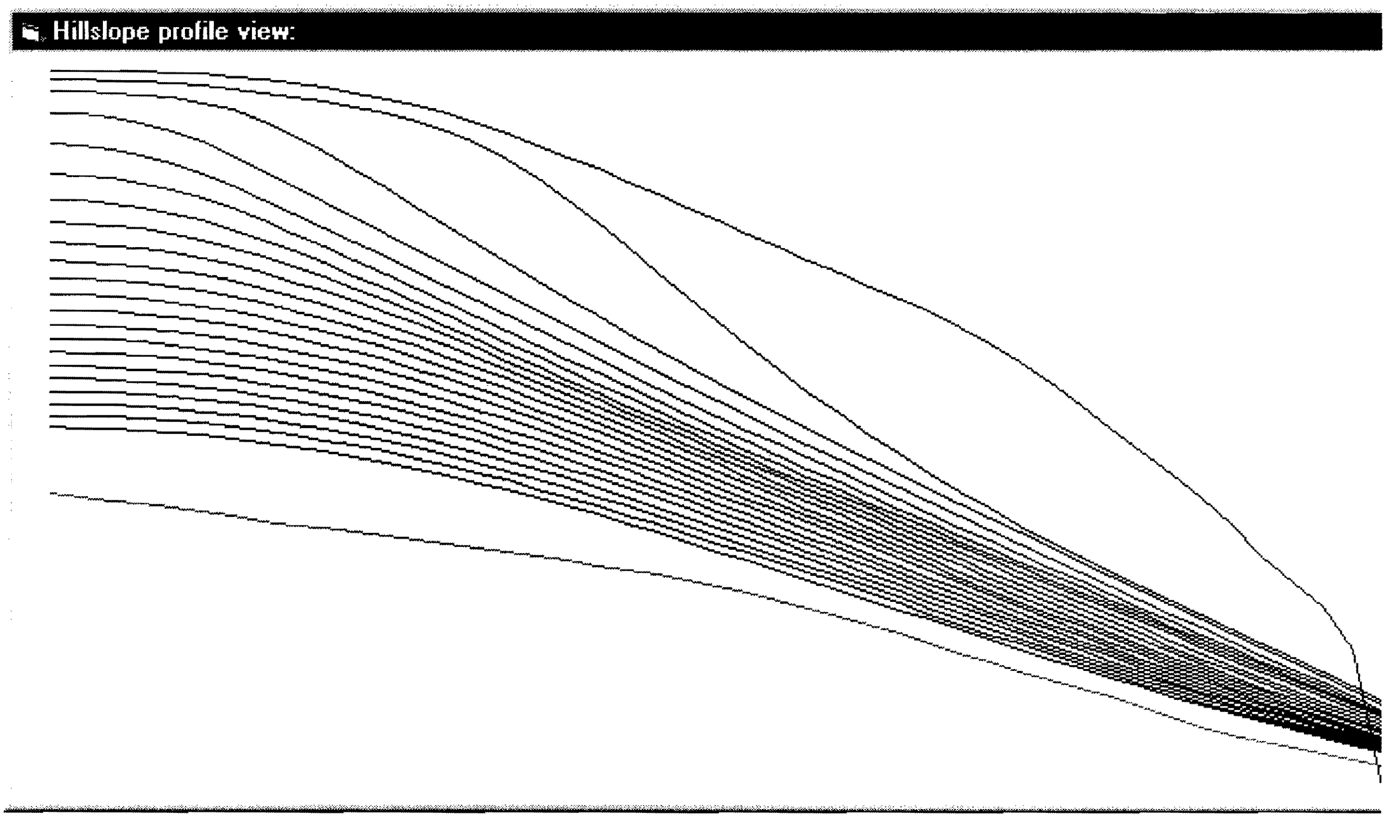
This window displays the process rates that the program in textboxes.



Modifying process rates at run-time -- It is possible to modify the process rates by clicking in the appropriate textbox and typing a new value. (Refer to the technical details section if you plan to do this.)

The Hillslope profile view window

This window is where the program draws all of its graphical output.



Copying the contents of the Hillslope profile view window -- It is possible to copy the contents of the Hillslope profile view window to the windows clipboard. To do so, click on the window to select it and press [Alt]-[Print Screen] on your keyboard. (This is a general feature of Windows, not HDS.)

Part II: An explanation of the file formats used by the program.

HDS requires at least two files for input: a file specifying the rate constants to use, and a file specifying the initial hillslope profile to start from. A third file, specifying a target profile, is optional. The program can use input files in its proprietary format, or it can import the .ev files used in hillslope modeling programs by Kirkby.

The Parameters file (.par)

HDS uses .par files to specify the various parameters that the model will use. Here's an example:

```
    Example .par parameter file
    Number of cells down length of slope:
    50
    PROCESS RATES
    Creep/Splash (c.10) or Solifluction (c.100) rate [sq.cm /y]
        Distance [m] at which wash becomes greater than creep etc
            Solution rate (Kirkby-type) [micro m/y] (250 for limestone, 5-50 for
others)
                Solution rate (uniform vertical lowering model) [micro m/y]
```

```
                  Solution threshold angle (for Kirkby-type) [degrees]
    100   200   0   50     1
    Landslide threshold angle [degrees] {8 to 30}
        Talus angle [degrees] (above which debris will not stop){c. 35x}
            Rate of free degradation above threshold [mm/y] (<2500mm/y)
                Landslide travel distance [m]
    23   35    5   50
    PROGRAM OPTIONS (Optional.  "NS" keyword indicates that a parameter is not
specified.)
    Use fixed time step of (years)
    NS
    Dynamically vary basal removal (0 or anything else = no, 1 = yes)
    1
    Fraction leaving base (fixed removal only)
    1
    Continuously redraw profile (0 or anything else = no, 1 = yes)
    0
    Overlay successive profiles every ____ iterations (ignored if the above is 1)
    200
    ASSOCIATED FILES (Optional.  "NS" keyword indicates that a parameters is not
specified.)
    Initial profile file (.ev or .dat)
    NS
    Target profile file (.ev or .dat)
    NS
```

How do I go about writing one of these?

Well, it's simpler than it looks--most of the lines in the above example are just placeholders. When the program loads a .par file, it doesn't use any of those lines of English text--those are only there for your benefit. The program only pays attention to those lines with numbers for program parameters in them. However, the lines of English text DO need to be there, because HDS is programmed to skip a specified number of lines before reading certain sets of parameters. For example, when HDS reads the following lines from the input file:

```
    PROCESS RATES
    Creep/Splash (c.10) or Solifluction (c.100) rate [sq.cm /y]
        Distance [m] at which wash becomes greater than creep etc
            Solution rate (Kirkby-type) [micro m/y] (250 for limestone, 5-50 for
others)
                Solution rate (uniform vertical lowering model) [micro m/y]
                    Solution threshold angle (for Kirkby-type) [degrees]
    100   200   0   50     1
```

it simply skips 6 lines down and then reads 5 values into the appropriate numeric variables. If you replaced the first six lines with six lines of nonsense characters, HDS wouldn't care or even know. You probably wouldn't want to do this, however, because those lines tell you what each of the 5 numbers listed represent. '100' represents the creep constant, '200' represents the distance at which wash becomes greater than creep, and so on.

The best way to write a .par file is to modify an existing one. Use "example.par", which is included with this program, or one of your own, and simply modify the values of the various parameters. Leave the lines of English text alone, unless you want to modify the comments that they contain.

What's specified in a .par file

The first thing that's specified is the "number of cells down length of slope." This is the number of discrete cells that the program will break the hillslope profile into. Using more cells results in greater detail in the hillslope profile, as well as more more computations for the computer to carry out. 50 has worked well for the simulations that I have run, but feel free to experiment and see what gives the best results.

After the number of cells to use is specified, the .par file is divided into three sections, two of which are optional:

The PROCESS RATES section:

This section is required. It's where you specify the values of the process rates and other constants that appear in the process laws that govern the evolution of the slope profile. For information about these parameters, refer to the documentation on the hillslope model itself.

The PROGRAM OPTIONS section:

This section is optional, but if it is included, it must immediately follow the "process rates" section. Here you can specify most of the program options that can be specified in the Program Options window when the program is running. See the documentation on the Program Options window for information on the various options. Note that you don't have to specify all of the paramters in this section. Place the keyword "NS" on any line where you prefer to use the program's default for that option.

The ASSOCIATED FILES section:

This section is also optional. If it is included, it must immediately follow the "program options" section, unless that section has been omitted, in which case it must immediately follow the "process rates" section. Here you can specify the initial and target profile files to use with the .par file, and thus avoid having to specify those files manually in the main program window.

Profile Data files (.dat)

These files are used to describe the geometries of the initial and target hillslope profiles. A .dat file consists of two columns specifying the (x,y) coordinates of points along the hillslope. The x and y coordinates on each line are separated by tabs or several spaces. An example file, example.dat, is included with the program.

.ev files

The .ev format is that used by Kirkby's SLOPEN hillslope modeling program. An .ev file specifies both hillslope process rates and hillslope geometry. In HDS, however, if an .ev file is specified in one of the input filename textboxes, only the type of input associated with that textbox is loaded from the .ev file. For example, if you specify example.ev in the textbox for the model parameters file, the program will load the parameters specified in example.ev, but it will only load the hillslope geometry contained in example.ev if that file is specified in the initial profile name textbox as well. Thus it is possible to use only the model parameters, only the hillslope geometry, or both from a given .ev file.

Here's an example .ev file (example.ev):

```
Example .ev file
Slope length (m), initial height(m) & number of points down length of slope
150   100   50
PROCESS RATES
Creep/Splash (c.10) or Solifluction (c.100) rate (sq.cm /y)
    Distance (m) at which wash becomes greater than creep etc
        Solution rate (fm/y) [250 for limestone, 5-50 for others]
100   200   0   50
Minimum insoluble residue in soil (0.4-0.5)
    Scale soil depth for initial deepening (c. 100cm)
        Soil depth at which transport rate is halved (1-1000cm)
0.5   100   50
Landslide threshold angle (x) {8x to 30x}
    Talus angle (above which debris will not stop){c. 35x}
        Rate of free degradation above threshold (<2500mm/y)
23    35     5
Landslide travel distance (m)
    Basal condition controlled by : 1 = Elev'n:  2 = Sediment Removal
        Base level elevation (m)
50    1     0
Absolute Rate of 1: Lowering  or 2: Sedi increment/unit fp width (<<2E4fm/y)
    Relative rate/m above baselevel 1: Lowering or 2: Sedi Inc (<<100fm/y)
        Flood Plain width (1-1000m) {only relevant in case 2}
0     0    10
TECTONICS (non interactive)
Uplift in fm/y (<20,000fm/y) at divide
    Uplift in fm/y at slope base
        Proportion p quadratic (+ for doming:  1-p= proportion linear)
0     0    0
INITIAL SOIL DEPTHS
At top of slope (m)
    At base of slope (m)
1     0
INITIAL FORM PROFILE for each slope section
Percent of slope length (adding to 100% or final point joined to baselevel)
    Percent of slope height (above base level)
        Proportion p quadratic (+ for convex; 1-p=prop linear)
17.56  2.13   0
11.58  5.90   0
 7.13  5.90   0
11.13 10.15   0
10.60  9.51   0
11.41 15.35   0
 8.09 13.04   0
 5.45 11.27   0
 3.10  7.84   0
 0.19 10.22   0
```

```
 0.42  8.69   0
If L.H. column adds to less than 100%, last point is joined to base level
```

The first portion of an .ev file specifies various model parameters. Only those parameters that also appear in .par files are used. Kirkby's SLOPEN program does some extra things that HDS does not, and the parameters that are specific to SLOPEN are simply ignored by HDS. As with .par files, the lines of text that identify the various parameters are simply place holders. One modification in the .ev file format is allowed by the program. If one places a fourth number on the same line as the "Solution rate" number (this is Kirkby-type), that fourth number will be interpreted as the rate of solution by uniform vertical lowering. If this number is not given, then the program assumes that no solution by uniform vertical lowering should occur.

The second portion of an .ev file specifies the initial hillslope geometry according to the following format: the first column specifies the percentage of the slope's horizontal length between two points, the second column specifies the percentage of the slope's height between two points, and the last column specifies the "proportion p quadratic", which is not used by HDS. I find the format of .par files much more intuitive for specifying initial hillslope geometry.

Some technical details

Selecting a time step

It is important to select a sufficiently small time step. Otherwise, numerical instabilities will appear, causing the model to exhibit increasingly erratic behavior (showing hillslope evolution that clearly defies physical law) until numerical overflow will cause the program to crash. An appropriate time step can be found by a moderate amount of trial and error. For most hillslopes, a time step on the order of tens of years usually works. The more dramatically the profile changes over time, the smaller the time step needs to be. For instance, young, steep profiles subject to dramatic change by landslides will require much smaller time steps than old, gentle profiles that change very gradually over time. Sometimes it is desirable to change the time step during the course of a model run, keeping it very small while a profile remains fairly steep, and then increasing it significantly when the profile is gentle and changes only very gradually over time. If you desire to do this, it is best to do so from within the Visual Basic environment (see Appendix B for details).

Modifying program parameters at run time

It is possible to modify the various program parameters while the model is running by clicking in the appropriate textbox and typing a new value. This can cause problems, though, because, at each iteration, the program grabs the values of the parameters from the text boxes. So, if you delete what is in one of the parameter text boxes and then the program tries to read the value from it, it will encounter an error and break. You can avoid such problems by cleverly making sure that there is always a value in the text box (example: to change time step from 50 to 100 you change it to 5, then 15, then 1, then 100). A much better way to make changes in the model parameters is to run the program code from within the Visual Basic development environment, however (see the details in Appendix B).

Note that a time step can only be too small insofar as it may require lots of computer (actual) time for the model to run.

Dynamically varying basal removal

This feature was added as more of an experiment than anything else. Like retaining too much sediment at the bottom of the slope when using fixed removal, dynamically varying the basal removal seems to result in unnaturally large sediment buildups at the base and numerical instabilities.

The procedure for determining the outflux of sediment from the basal cell when using this option is simple. The program simply determines a linear rate of increase of sediment outflux with increasing position downslope by calculating the rate of increase between the cell two positions before the basal cell and the cell immediately before the basal cell. The program then extrapolates to determine the outflux for the basal cell.

Appendix A: Determination of best fit profiles

The program determines best fit profiles by two slightly different methods. According to the method based on the minimum mean squared difference, the best-fitting simulated profile is taken to be the one with the smallest mean squared difference between elevations along the simulated and target profiles. The best fit profile based on the minimum mean absolute difference is determined in a similar fashion, but absolute values of differences between elevations along the simulated and target profiles are used, rather than squared differences.

I do not know which method really yields a closer fit. In the research that I have done, I have ended up using the absolute differences method, but both methods usually yield very similar results.

Appendix B: Using the program in the Visual Basic environment

Although it is possible to run HDS from the executable files included in the program distribution, the program is more useful when it is run from within the Visual Basic environment. From within the Visual Basic environment it is possible to pause the program (break) during execution, modify program parameters, and then continue execution from where the program left off. The ability to do this can be very valuable. For instance, if one knows that significant climatic change has taken place during a hillslope's evolution, one can begin a model run with parameters appropriate to earlier conditions, pause the model run after an appropriate amount of time has passed, modify the process rates to reflect later climatic conditions, and then resume the model run with those rate changes in place.

There are two good ways to pause a model run. The first is to manually issue a break command while the program is running, either by pushing the break button on the Visual Basic toolbar, issuing a CNTRL-BREAK from the keyboard, or by selecting "Break" from the "Run" menu of the Visual Basic environment. The second way is to instruct Visual Basic to break on a watched expression. To do this, one selects "Add Watch" from the "Debug" menu. When the "Add Watch" window pops up, one types the expression that Visual Basic is to watch in the "Expression:" textbox, and then clicks the "Break When Value Is True" option. This is the best way to pause the model when you want to stop it at a specific point in model time, such as when you wish to change the process rates after a certain number of years have passed. The model keeps track of how many years have elapsed in the model run in the global variable "Time". So, for example, to have the model break execution after 10000 years, the watched expression to specify is "Time = 10000".

Once the model has been stopped, parameters values may be changed by issuing assignment statements (of the form `variable = new_value`) in the "Immediate" code window. To modify most parameters, one actually needs to modify the contents of the textboxes in the program windows--this is because the program internally represents some of these values in different units than the user specifies them in. Code is attached to the parameter textboxes in the program that will convert these to the units that the program uses internally and then update the internal variables. This conversion and updating occurs anytime that the values of the textboxes are changed, even if the program is in break mode at the time.

Inconveniently, when the program is in break mode, the values of the textboxes cannot be changed by clicking on them and then editing their contents. The values must be changed via an assignment statement issued in the "Immediate" code window. The variable names for the important program parameters are listed below:

`OptionsForm.Timestep` -- timestep (in years)
`ParametersForm.creep_constant` -- the creep/rainsplash/solifluction constant
`ParametersForm.u_wash` -- distance at which wash becomes greater than creep, etc.
`ParametersForm.solution_rate` -- solution rate for Kirkby-type solution
`ParametersForm.vertical_solution` -- solution rate for the uniform vertical lowering model
`ParametersForm.solution_threshold` -- threshold angle for Kirkby-type solution
`ParametersForm.lower_threshold` -- landslide threshold angle
`ParametersForm.higher_threshold` -- talus angle (above which debris will not stop)
`ParametersForm.detachment_constant` -- rate of free degradation above threshold
`ParametersForm.travel_constant` -- landslide travel distance

To change one of these parameters in the middle of a model run, one puts the program in break mode, issues an assignment statement in the "Immediate" code window (e.g. `OptionsForm.Timestep = 100`), and then continues execution of the program.

Appendix C: Specifics of altering the model

The program code was written keeping ease of future modification in mind. The erosional processes (with the exception of uniform vertical solution, which will also be discussed) are implemented in a modular fashion, with each process being implemented in separate function that returns the downslope sediment flux due to the action of that process. Thus one can easily modify how a process is modeled or add new a one to the program without having to make changes spread throughout the program code.

All of the code that performs the actual calculations in the model are contained in `Module1.bas`. Code contained in the form objects only implement the user interface. `Module1.bas` contains many functions/subroutines; we will discuss those which one needs to be familiar with in order to alter the model.

The first procedure that gets executed when the program is started is the `Main()` procedure, which simply loads the appropriate forms at startup. The procedure that should really be considered the main part of the program is the `MainLoop()` subroutine, which really launches everything else in the model. It is what is called when the "Run Model" button is pressed (after a few initializations associated with the user interface are performed). When `MainLoop()` begins, it carries out some initializations, and then enters the main `while` loop in the program, which terminates when something sets the `model_stop` flag to `True`. Each time through the loop, the necessary graphics are drawn, the model is taken through another iteration, the goodness of fit is calculated, and, finally, the iteration counter is incremented.

`MainLoop()` calls the `Step()` subroutine to perform an actual iteration of the numerical model. The first thing that the `Step()` subroutine does is enter a `for` loop that, for each cell along the hillslope profile, does the following:

1. Calculates the downslope gradient.
2. Calculates and sums the downslope sediment fluxes due to each erosional process being modeled.
3. Calculates the time derivative of elevation *dz/dt* by subtracting the total flux out of the current cell from that of the previous cell, and then dividing by the distance between cells *dx*.

Once that loop has completed, the program enters another `for` loop. This loop uses the *dz/dt* value calculated for each cell to determine the elevation change that should occur over the length of the time step, and then the elevation of each cell is updated accordingly. It is here that the effects of uniform vertical lowering by solution are incorporated, instead of in the preceding `for` loop. This is because solutional uniform vertical lowering is not modeled as a downslope sediment flux.

There are, of course, some special cases that have not been discussed above, such as what to do at the hillslope divide or the basal cell. These were omitted for the sake of clarity, and they can be easily understood by examining the code in the `step()` subroutine.

Fluxes due to various erorsional processes are calculated by the following functions:

`SlowFlux()` -- Returns the sediment flux due to creep/splash/solifluction and wash.
`SlideFlux()` -- Returns the sediment flux due to landslides.
`SolutionFlux()` -- Returns the sediment flux due to Kirkby-type solution.

To modify the operation of these processes, one needs only to modify the the code within each function.

To add new erosional processes to the model, one first needs to write a function that returns the downslope sediment flux due to that process. Then the code within the first `for` loop of the `Step()` subroutine that sums the fluxes due to all of the erosional processes needs to be modified to include the flux due to the added process in the summation. The code that does so consists of the following two lines:

```
Total_transport(i) = (Slow_transport + Slide_Transport)
dz_dt(i) = (Total_transport(i - 1) - (Total_transport(i) + Solution_transport)) / dx
```

The reason that `Solution_transport` is not included in the `Total_transport(i)` sum is that Kirkby-type solution does not build a talus, viz., material freed by solution transport is removed immediately from the system. That is why `Solution_transport` is only added in on the second line. (It is, of course, very easy to change things so that the flux due to Kirby-type solution does build a talus. There is code commented out in the procedure that implements Kirby-type solution in such fashion.) If one desires a new erosional process to build a talus, simply add the flux calculated for that function into the `Total_transport(i)` sum. If not, only add it in the second line, where `Solution_transport` is added.

References Cited

Kirkby, M. J., 1971, Hillslope process-response models based on the continuity equation, in Brunsden, D. (ed.), Slopes: form and process: London, Institute of British Geographers, Special Publication 3, p. 15-30.

Kirkby, M. J., 1984, Modelling cliff development in South Wales: Savigear reviewed: Zeitschrift für Geomorphologie, v. 28, p. 405-426.

Kirkby, M. J., 1987, General models of long-term slope evolution through mass movement, in Anderson, M. G., and Richards, K. S., Slope stability: London, Wiley & Sons Ltd, p. 359-379.

Kirkby, M. J., 1992, An erosion-limited hillslope erosion model, in Schmidt, K.-H., and de Ploey, J., eds., Functional geomorphology: landform analysis and models: Catena Supplement no. 23, p. 157-187.

Kirkby, M. J., Naden, P.S., Burt, T. P., and Butcher, D. P., 1992, Computer simulation in physical geography: Chichester, Wiley: p. 85-90.

APPENDIX B:

HILLSLOPE DYNAMICS SIMULATOR

SOURCE CODE

```
'A multiple linear store model for hillslope evolution.
'Based loosely on a model formulated by M. J. Kirkby.
'Copyright (c) 1997, 1999 by Richard Tran Mills.
'Initial alpha version completed 7/11/97.
'First semblance of an actual user interface appeared 7/16/97.
'Limited .ev import ability also added 7/16/97.
'7/19/97--Completed fixing some small problems with .ev importing.
'7/24/97--Finished adding target profile comparison, (x,y) format initial profiles.
'7/25/97--Added simple support for a partial sediment flux out of the last cell.  Added a text bo
x for the user to specify the fixed time step.
'7/26/97--Added some different options for handling solution.
'7/29/97--Fixed a slight error that was causing large problems with model time: landslide thresho
lds were being left in degrees, rather than being converted into slopes.
'7/30/97--Fixed an error in yesterday's bug fix! (It was due to a typo.)
'7/31/97--Added a box in the options window to allow the user to input the solution threshold ang
le (in degrees).
'8/01/97--Added several features that we decided were necessary after starting modeling yesterday
: Program displays best fit profiles, means differences; separate text boxes for .ev and .dat fil
e names; program defaults to solution which produces no talus.
'8/03/97--Added a tally of the initial amount of sediment and the amount of sediment (talus) whic
h leaves the system.
'8/06/97--Added support for a uniform vertical solutional lowering rate, as well as code to preve
nt negative elevations (these were leading to numerical problems).
'8/07/97--Added support for a dynamically varying removal rate in the last cell, based on a linea
r extrapolation from the removal rates of the second- and first to last cells.
'         With this change in place, declared version 1.0 of the program.
'4/15/99--After numerous improvements upon the user interface, released version 1.1
'         of the program.

Option Explicit

'Represents the horizontal (x) and vertical (y) position of a hillslope cell.
Public Type Cell
    x As Double
    y As Double
End Type

Public Const PI As Double = 3.14159265359

Public hillslope_cell() As Cell    'Dynamic array of cells that make up the hillslope profile
Public target_profile() As Cell    'Dynamic array of cells that make up the target hillslope prof
ile

Public model_stop As Integer    'Flag to stop model.  Model stops when model_stop = True
Public iteration As Long    'Counts the number of iterations elapsed.
Public number_of_cells As Integer   'Holds the number of cells in the hillslope profile

Public dt As Double     'Time step
Public dx As Double     'Cell spacing along profile

Public Time As Double   'Absolute "time" since the model started iterating

Public creep_constant As Double 'Constant governing the rate of creep
Public u_wash As Double 'The distance in meters beyond which the wash becomes greater than creep
Public higher_threshold As Double, lower_threshold As Double    'Gradient thresholds for landslid
es
Public travel_constant As Double    'Constant governing landslide travel distance
Public detachment_constant As Double    'Constant governing landslide detachment
Public solution_rate As Double      'Rate of denudation (Kirkby-type solution) (this is a constan
t)
Public vertical_solution As Double      'Rate of uniform vertical lowering by solution
Public solution_threshold As Double     'Angle (in degrees) which the local gradient must equal o
r exceed for Kirkby-type solution to occur

Public MinSumAbs As Double  'Minimum sum of absolute differences
Public MinSumSquares As Double  'Minimum sum of squared differences
Public BestAbsFitTime As Double    'Time at which the best fit to the target profile has occurred
, according to the sum of absolute differences
Public BestAbsProfile() As Cell   'The modeled profile which best fits the target profile, accord
ing to the sum of absolute differences
```

```
Public BestSquaresFitTime As Double 'Time of best fit, according to difference of squares.
Public BestSquaresProfile() As Cell 'Modeled profile associated with BestSquaresFitTime.

Public Initial_area As Double   'Holds the area of the initial profile; used to indicate the tota
l amount of sediment that the model starts with.
Public Flux_out As Double   'Total flux of talus into the last cell.  This allows us to calculate
 how much talus leaves the system when we have 100% removal at the slope base.


'ImportEv() imports parameters and initial profiles from a .ev file whose name
'is contained in infile$.  It returns True upon success.

Public Function ImportEv(infile$) As Integer
    Dim i As Integer, j As Integer
    Dim test$
    Dim junk As Variant     'This really is just a variable for holding junk.
    Dim x As Double, y As Double, m As Double
    Dim slope_length As Double, initial_height As Double
    Dim xpercent() As Double, ypercent() As Double
    Dim xoriginal() As Double, yoriginal() As Double
    Dim number_of_original_points As Integer
    Open infile$ For Input As #1

    'Read in the data from the .ev file

    For i = 1 To 2
        Line Input #1, junk
    Next i
    Input #1, slope_length, initial_height, number_of_cells
    'The next line of code needs a bit of explaining.  When Kirkby's program says it is using,
    'say, 20 cells, it is modelling the hillslope with 21 points.  On the other hand, when my
    'program says it is using 20 cells, it uses 20 points to model the hillslope.  To correct
    'for this difference, I add 1 to the number of cells specified in the .ev file.
    number_of_cells = number_of_cells + 1
    For i = 1 To 4
        Line Input #1, junk
    Next i
    Input #1, creep_constant, u_wash, solution_rate
    creep_constant = creep_constant * 0.0001
    solution_rate = solution_rate * 0.000001
    Input #1, junk
    If VarType(junk) >= 2 And VarType(junk) <= 5 Then
        vertical_solution = junk * 0.000001
    Else
        vertical_solution = 0
    End If
    For i = 1 To 7
        Line Input #1, junk
    Next i
    Input #1, lower_threshold, higher_threshold, detachment_constant
    lower_threshold = Tan(PI / 180 * lower_threshold)   'Put the threshold's in terms of slope.
    higher_threshold = Tan(PI / 180 * higher_threshold)
    detachment_constant = detachment_constant * 0.001   'Not sure if this is the correct thing to
 do.  I think it is.
    For i = 1 To 3
        Line Input #1, junk
    Next i
    Input #1, travel_constant, junk, junk
    For i = 1 To 17
        Line Input #1, junk
    Next i

    'Read in the original profile in terms of percentages.
    number_of_original_points = 0
    Do
        test$ = Input(1, #1)
        If (Asc(test$) > 47 And Asc(test$) < 58) Or Asc(test$) = 46 Then   'i.e., if the characte
r is a digit or decimal
            Seek #1, (Seek(1) - 1)  'Move read/write position back one character.
            Input #1, x, y, junk
```

```
            If Not (x = 0 And y = 0) Then
                number_of_original_points = number_of_original_points + 1
                ReDim Preserve xpercent(number_of_original_points)
                ReDim Preserve ypercent(number_of_original_points)
                xpercent(number_of_original_points) = x
                ypercent(number_of_original_points) = y
            End If
        End If
    Loop While (Asc(test$) > 47 And Asc(test$) < 58) Or Asc(test$) = 46
    'Convert the slope percentages into (x,y) coordinates.
    ReDim xoriginal(number_of_original_points + 1)
    ReDim yoriginal(number_of_original_points + 1)
    xoriginal(1) = 0
    yoriginal(1) = initial_height
    For i = 1 To number_of_original_points
        xoriginal(i + 1) = xoriginal(i) + slope_length * xpercent(i) / 100
        yoriginal(i + 1) = yoriginal(i) - initial_height * ypercent(i) / 100
    Next i
    If xoriginal(number_of_original_points + 1) <> slope_length Then    'If the x percentages don
't add up to 100, we must add a final point as baselevel.
        ReDim Preserve xoriginal(number_of_original_points + 2)
        ReDim Preserve yoriginal(number_of_original_points + 2)
        xoriginal(number_of_original_points + 2) = slope_length
        yoriginal(number_of_original_points + 2) = 0
    End If
    'Initialize the array of equally spaced cells composing the initial hillslope profile
    ReDim hillslope_cell(number_of_cells)
    dx = slope_length / (number_of_cells - 1)
    hillslope_cell(1).x = xoriginal(1)
    hillslope_cell(1).y = yoriginal(1)
    For i = 2 To (number_of_cells)
        x = hillslope_cell(i - 1).x + dx
        hillslope_cell(i).x = x
        j = 1
        'While xoriginal(j) < hillslope_cell(i).x
        'Note that the above line of code which is commented out SHOULD work, but doesn't.  I thi
nk there's a bug in VB.
        'Hence I use the line below as a substitute.
        While CSng(xoriginal(j)) < CSng(hillslope_cell(i).x)
            j = j + 1
        Wend
        m = (yoriginal(j) - yoriginal(j - 1)) / (xoriginal(j) - xoriginal(j - 1))   'm = slope of
 the line
        y = m * (x - xoriginal(j)) + yoriginal(j)
        hillslope_cell(i).y = y
    Next i
    Close #1
    ImportEv = True
End Function


'LoadXYTargetProfile () loads a target hillslope profile from an ASCII file of (x,y) points.
'The target profile is then broken down into cells with the same spacing as the ones in the initi
al profile.
'The target and initial profiles must have the same initial length.
'The first point in the target profile must have x = 0.
'This procedure will only work correctly after a starting profile has been loaded.
'
'This function returns True is successful, False if not.

Public Function LoadXYTargetProfile(ProfileName$) As Variant
    Dim i As Integer, j As Integer, filenumber As Integer
    Dim number_of_points As Integer
    Dim x As Double, y As Double, m As Double
    Dim xoriginal() As Double, yoriginal() As Double
    ReDim target_profile(number_of_cells)

    filenumber = FreeFile()
    Open ProfileName$ For Input As #filenumber
    While EOF(filenumber) = 0        'Read in the (x,y) data
```

```
        number_of_points = number_of_points + 1
        ReDim Preserve xoriginal(number_of_points)
        ReDim Preserve yoriginal(number_of_points)
        Input #filenumber, xoriginal(number_of_points), yoriginal(number_of_points)
    Wend

    'Exit and return False if initial and target profiles are not of the same length.
    If xoriginal(number_of_points) <> hillslope_cell(number_of_cells).x Then
        LoadXYTargetProfile = False
        Exit Function
    End If

    'Now break the target profile into cells with spacing identical to that of the initial profil
e
    target_profile(1).x = xoriginal(1)
    target_profile(1).y = yoriginal(1)
    For i = 2 To number_of_cells
        x = target_profile(i - 1).x + dx
        target_profile(i).x = x
        j = 1
        'See the comments in ImportEv() for why I use the CSng() function here.
        While CSng(xoriginal(j)) < CSng(target_profile(i).x)
            j = j + 1
        Wend
        m = (yoriginal(j) - yoriginal(j - 1)) / (xoriginal(j) - xoriginal(j - 1))   'm = slope of
 the line
        y = m * (x - xoriginal(j)) + yoriginal(j)
        target_profile(i).y = y
    Next i

    Close #filenumber

    'Exit procedure and return True; target profile has been successfully loaded.
    LoadXYTargetProfile = True
End Function


'LoadXYInitialProfile loads an initial hillslope profile from an ASCII file of
'(x,y) points.  ProfileName$ specifies the name of the file to be loaded.
'The function returns True upon success.

Public Function LoadXYInitialProfile(ProfileName$) As Variant
    Dim i As Integer, j As Integer, filenumber As Integer
    Dim number_of_points As Integer
    Dim x As Double, y As Double, m As Double
    Dim xoriginal() As Double, yoriginal() As Double
    'The Preserve keyword is not used here, so the profile loaded from a .ev
    'file is overwritten.
    ReDim hillslope_cell(number_of_cells)

    filenumber = FreeFile()
    Open ProfileName$ For Input As #filenumber
    While EOF(filenumber) = 0        'Read in the (x,y) data
        number_of_points = number_of_points + 1
        ReDim Preserve xoriginal(number_of_points)
        ReDim Preserve yoriginal(number_of_points)
        Input #filenumber, xoriginal(number_of_points), yoriginal(number_of_points)
    Wend
    dx = (xoriginal(number_of_points) - xoriginal(1)) / (number_of_cells - 1)
    'Now break the target profile into cells with equal spacing.
    hillslope_cell(1).x = xoriginal(1)
    hillslope_cell(1).y = yoriginal(1)
    For i = 2 To number_of_cells
        x = hillslope_cell(i - 1).x + dx
        hillslope_cell(i).x = x
        j = 1
        'See the comments in ImportEv() for why I use the CSng() function here.
        While CSng(xoriginal(j)) < CSng(hillslope_cell(i).x)
            j = j + 1
        Wend
```

```
        m = (yoriginal(j) - yoriginal(j - 1)) / (xoriginal(j) - xoriginal(j - 1))   'm = slope of
 the line
        y = m * (x - xoriginal(j)) + yoriginal(j)
        hillslope_cell(i).y = y
    Next i
    Close #filenumber
    'Exit procedure and return True; (x,y) initial profile has been
    'successfully loaded.
    LoadXYInitialProfile = True
End Function


'Mainloop() launches everything else in the model, basically.

Public Sub MainLoop()
    Dim i As Integer, test As Variant, junk As Integer
    ReDim BestAbsProfile(number_of_cells)
    ReDim BestSquaresProfile(number_of_cells)
    GraphForm.Show
    'Put this here so OptionsForm won't be partially obscured by GraphForm
    OptionsForm.Show
    Call ScaleToFit(hillslope_cell, GraphForm)
    If MainForm.chkUseTargetProfile = 1 Then
        test = LoadTargetProfile(MainForm.TargetProfileName)
        If test = False Then
            junk = MsgBox("The target profile you selected could not be used.", vbExclamation, "W
arning")
        ElseIf MainForm.ChkDisplayTargetProfile = 1 Then
            Call ProfilePlot(target_profile, GraphForm, RGB(255, 0, 0))
        End If
    End If
    StatusForm.Show
    If MainForm.chkUseTargetProfile = 1 Then
        'We have to do all this to prime the loop.
        MinSumAbs = SumAbsDifferences(hillslope_cell, target_profile)
        MinSumSquares = SumSquaredDifferences(hillslope_cell, target_profile)
        BestAbsFitTime = 0
        BestSquaresFitTime = 0
        StatusForm.BestAbsFitTime = BestAbsFitTime
        StatusForm.MinMeanAbsDifference = MinSumAbs / number_of_cells
        StatusForm.BestSquaresFitTime = BestSquaresFitTime
        StatusForm.MinMeanSquaredDifference = MinSumSquares / number_of_cells
        For i = 1 To number_of_cells
            BestAbsProfile(i) = hillslope_cell(i)
        Next i
        For i = 1 To number_of_cells
            BestSquaresProfile(i) = hillslope_cell(i)
        Next i
    End If

    Initial_area = ProfileArea(hillslope_cell)

    'This is the main program loop.
    While model_stop <> True
        If OptionsForm.OptRedraw = True Then
            GraphForm.Cls
            Call ProfilePlot(hillslope_cell, GraphForm)    'Plot the current hillslope profile
            If MainForm.ChkDisplayTargetProfile = 1 Then Call ProfilePlot(target_profile, GraphFo
rm, RGB(255, 0, 0))
        ElseIf OptionsForm.OptNoRedraw = True And iteration Mod OptionsForm.UpdateFrequency = 0 T
hen
            Call ProfilePlot(hillslope_cell, GraphForm)
        End If
        If OptionsForm.WriteFrequency <> Empty Then
            If OptionsForm.chkWriteSuccessiveProfiles = 1 And iteration Mod CInt(OptionsForm.Writ
eFrequency) = 0 Then
                test = AppendProfile(OptionsForm.OutputFilename)
            End If
        End If
```

```
        StatusForm.DisplayStatus
        Step    'Move the model through one time step.
        'Check for best fit if a target profile is being used.
        If MainForm.chkUseTargetProfile = 1 Then
            If SumAbsDifferences(hillslope_cell, target_profile) < MinSumAbs Then
                MinSumAbs = SumAbsDifferences(hillslope_cell, target_profile)
                BestAbsFitTime = Time
                StatusForm.BestAbsFitTime = Time
                StatusForm.MinMeanAbsDifference = MinSumAbs / number_of_cells
                For i = 1 To number_of_cells
                    BestAbsProfile(i) = hillslope_cell(i)
                Next i
            End If
            If SumSquaredDifferences(hillslope_cell, target_profile) < MinSumSquares Then
                MinSumSquares = SumSquaredDifferences(hillslope_cell, target_profile)
                BestSquaresFitTime = Time
                StatusForm.BestSquaresFitTime = Time
                StatusForm.MinMeanSquaredDifference = MinSumSquares / number_of_cells
                For i = 1 To number_of_cells
                    BestSquaresProfile(i) = hillslope_cell(i)
                Next i
            End If
        End If
        'DoEvents passes control to the operating system to let it handle
        'routine tasks, etc.
        DoEvents
        iteration = iteration + 1
    Wend
End Sub


'ProfileArea() calculates the area under a slope profile by the trapezoidal rule.
'Note, of course, that this will yield somewhat inaccurate areas for slopes
'with overhangs.  Though I do not think it is possible to get slopes with
'overhangs in the model.
'This function isn't actually used anywhere in the model, but it is useful to
'have if one wants to calculate how much sediment has been eroded out of the
'system.
'Arguments:
'   profile() -- Array of Cell's that specify a hillslope profile.

Public Function ProfileArea(profile() As Cell) As Double
    Dim i As Integer
    Dim area As Double

    area = 0
    For i = 1 To (number_of_cells - 1)
        area = area + (1 / 2) * (profile(i).y + profile(i + 1).y) * (profile(i + 1).x - profile(i
).x)
    Next i
    ProfileArea = area
End Function


'SumAbsDifferences() calculates the sum of the absolute differences in elevation
'between two hillslope profiles.  It is used to determine best fits by absolute
'differences.
'Arguments:
'   profile1() -- Array of Cells representing a hillslope profile
'   profile2() -- Ditto

Public Function SumAbsDifferences(profile1() As Cell, profile2() As Cell) As Double
    Dim i As Integer
    Dim sum As Double

    For i = 1 To number_of_cells
        sum = sum + Abs(profile1(i).y - profile2(i).y)
    Next i
    SumAbsDifferences = sum
End Function
```

```
'ScaleToFit() sets up the GraphForm coordinate system such that the hillslope
'contained in profile() fits inside the form.

Public Sub ScaleToFit(profile() As Cell, GraphForm)
    Dim i As Integer
    Dim xmax As Double, xmin As Double
    Dim ymax As Double, ymin As Double
    Dim xlength As Double, ylength As Double

    'Find the max and min x and y values
    For i = 1 To number_of_cells
        If profile(i).x > xmax Then xmax = profile(i).x
        If profile(i).x < xmin Then xmin = profile(i).x
        If profile(i).y > ymax Then ymax = profile(i).y
        If profile(i).y < ymin Then ymin = profile(i).y
    Next i

    'Set up the coordinate system with some extra room around the edges
    xlength = xmax - xmin
    ylength = ymax - ymin
    GraphForm.Scale (xmin - 0.025 * xlength, ymax + 0.025 * ylength)-(xmax + 0.025 * xlength, ymi
n - 0.025 * ylength)
End Sub


'ProfilePlot() plots an array of hillslope cells to a plot object with an
'optionally specified color value.
'Arguments:
'    data -- an array of Cell data structures with x and y variables.
'    plot object -- the name of the form, picture box, etc., within which to plot
'                   the graph.
'    ColorValue -- the color value that the profile is to be drawn with.  This
'                  argument is optional.

Public Sub ProfilePlot(data() As Cell, plot_object, Optional ColorValue)
    Dim i As Integer

    'The following code which has been commented out plots only the points
    'in the hillslope profile.
    'For i = 1 To number_of_cells
    '    plot_object.PSet (data(i).x, data(i).y)
    'Next i

    'This code connects the points in the hillslope profile with lines.
    If VarType(ColorValue) = vbError Then
        plot_object.PSet (data(1).x, data(1).y)
        For i = 2 To number_of_cells
            plot_object.Line -(data(i).x, data(i).y)
        Next i
    Else
        plot_object.PSet (data(1).x, data(1).y), ColorValue
        For i = 2 To number_of_cells
            plot_object.Line -(data(i).x, data(i).y), ColorValue
        Next i
    End If
End Sub


'SlideFlux() calculates the sediment flux downslope due to landslides.
'Arguments:
'    x -- Distance from the divide
'    gradient -- Downslope gradient
'    previous_slide_transport -- the slide flux for this iteration out of the
'        adjacent upslope cell

Public Function SlideFlux(x As Double, gradient As Double, previous_slide_transport As Double) As
 Double
    Dim Detachment As Double
```

```
    'Reciprocal_height is the reciprocal of the slide travel distance
    '(Kirkby's h)
    Dim Reciprocal_height As Double
    Dim Flux As Double

    Reciprocal_height = (higher_threshold - gradient) / travel_constant
    'The line below is just to prevent any numerical problems.
    If Reciprocal_height < 0 Then Reciprocal_height = 0
    Detachment = detachment_constant * gradient * (gradient - lower_threshold)
    Flux = (Detachment * dx + previous_slide_transport) / (1 + Reciprocal_height * dx)
    If Flux < 0 Then Flux = 0
    SlideFlux = Flux
End Function


'SolutionFlux() calculates sediment flux due to Kirkby-type solution.  The
'downslope gradient must exceed the "solution threshold gradient"
'(solution_threshold)in order for solution to occur.
'Arguments:
'   x -- Distance from the divide
'   gradient -- Downslope gradient

Public Function SolutionFlux(x As Double, gradient As Double) As Double
    If gradient > Tan(PI / 180 * solution_threshold) Then SolutionFlux = solution_rate * x Else S
olutionFlux = 0
End Function


'SlowFlux() calculates the combined downslope sediment flux for "creep"
'processes and wash.
'Arguments:
'   x -- Distance from the divide
'   gradient -- Downslope gradient

Public Function SlowFlux(x As Double, gradient As Double) As Double
    SlowFlux = creep_constant * gradient * (1 + (x / u_wash) ^ 2)
End Function


'Step() steps the model through one iteration.

Public Sub Step()

Dim i As Integer
Dim gradient As Double       'Local downslope gradient
Dim dz_dt() As Double  'Change in elevation z with time t
Dim Total_transport() As Double 'Total downslope sediment flux
Dim Slow_transport As Double, Solution_transport As Double, Slide_Transport As Double
Dim P1 As Double, P2 As Double, P3 As Double, m As Double

ReDim dz_dt(number_of_cells)
ReDim Total_transport(number_of_cells)

'Calculate rates of denudation, etc., for each cell.

For i = 1 To number_of_cells
    If i = number_of_cells Then     'viz., if we are at the base of the slope
        If OptionsForm.chkVaryRemoval = 0 Then
            'Assume that the flux out of the cell is a given percent of the flux out of the cell
directly above.
            dz_dt(number_of_cells) = (Total_transport(number_of_cells - 1) - CDbl(OptionsForm.Fra
ctionLeaving) * Total_transport(number_of_cells - 1)) / dx
        ElseIf OptionsForm.chkVaryRemoval = 1 Then
            'P1 = percent of incoming talus leaving second to last cell
            P1 = Total_transport(number_of_cells - 2) / Total_transport(number_of_cells - 3)
            'P2 = percent of incoming talus leaving first  to last cell
            P2 = Total_transport(number_of_cells - 1) / Total_transport(number_of_cells - 2)
            'm = slope of the line relating amount of incoming talus leaving a cell to the cell's
 x coordinate.
            m = (P2 - P1) / (hillslope_cell(number_of_cells - 1).x - hillslope_cell(number_of_cel
```

```
ls - 2).x)
            P3 = m * (hillslope_cell(number_of_cells).x - hillslope_cell(number_of_cells - 1).x)
+ P2
            dz_dt(number_of_cells) = (Total_transport(number_of_cells - 1) - P3 * Total_transport
(number_of_cells - 1)) / dx
        End If

    Else
        'This assumes that the divide is on the left and the slope base on the right.
        gradient = (hillslope_cell(i).y - hillslope_cell(i + 1).y) / dx
        'The [name]_transport variables are all RATES (fluxes) of transport.
        Slow_transport = SlowFlux(hillslope_cell(i).x, gradient)
        Solution_transport = SolutionFlux(hillslope_cell(i).x, gradient)
        Slide_Transport = SlideFlux(hillslope_cell(i).x, gradient, Slide_Transport)

        'Do not allow solution to build a talus.
        Total_transport(i) = (Slow_transport + Slide_Transport)
        dz_dt(i) = (Total_transport(i - 1) - (Total_transport(i) + Solution_transport)) / dx

        'Allow solution to build a talus.
        'Total_transport(i) = (Slow_transport + Slide_Transport + Solution_transport)
        'dz_dt(i) = (Total_transport(i - 1) - (Total_transport(i))) / dx

        If i = 1 Then   'i.e., if we are at the divide
            'We assume symmetry such that an equal amount of sediment
            'leaves each side of the divide.
            dz_dt(1) = dz_dt(1) * 2
        End If

    End If
Next i

'Choose the value of the time increment dt.
'For right now, I'm simply using a fixed time increment.

dt = CDbl(OptionsForm.Timestep)

'Update each cell

Time = Time + dt
For i = 1 To number_of_cells
    'We have to incorporate the vertical lowering by solution here, so it will
    'not be multiplied by 2 at the first cell.
    hillslope_cell(i).y = hillslope_cell(i).y + dz_dt(i) * dt - vertical_solution * dt
    'This is to prevent negative elevations, which can lead to numerical problems
    If hillslope_cell(i).y < 0 Then hillslope_cell(i).y = 0
Next i

'Flux_out is the total flux into the basal cell.  It isn't actually used by the
'program, but it allows us to keep track of the amount of debris that leaves
'the system when we are retaining no sediment at the slope bottom.
Flux_out = Flux_out + Total_transport(number_of_cells - 1) * dt

End Sub


'SumSquaredDifferences() calculates the sum of the difference of squares between
'two hillslope profiles.
'Arguments:
'   profile1() -- An array of Cell's representing a hillslope profile
'   profile2() -- ditto

Public Function SumSquaredDifferences(profile1() As Cell, profile2() As Cell) As Double
    Dim i As Integer
    Dim sum As Double

    For i = 1 To number_of_cells
        sum = sum + (profile1(i).y - profile2(i).y) ^ 2
    Next i
    SumSquaredDifferences = sum
```

```
End Function


'Writefitdata() saves the data on best fits to the file whose name is contained
'in filename$.  It returns True when successful.

Public Function WriteFitData(filename$) As Variant
    Dim i As Integer, filenumber As Integer

    filenumber = FreeFile()

    Open filename$ For Output As #filenumber
    Print #filenumber, "Initial Profile Name: " + MainForm.InitialProfileFileName
    Print #filenumber, "Target Profile Name: " + MainForm.TargetProfileName
    Print #filenumber, "Model time elapsed (years):", Time
    Print #filenumber, "Iterations completed:", iteration
    Print #filenumber, "Time of best fit (squared differences):", BestSquaresFitTime
    Print #filenumber, "Minimum mean squared difference:", (MinSumSquares / number_of_cells)
    Print #filenumber, "Time of best fit (absolute differences:", BestAbsFitTime
    Print #filenumber, "Minimum mean absolute difference:", (MinSumAbs / number_of_cells)
    Print #filenumber, ""
    Print #filenumber, "Best fit profile (squared differences):"
    For i = 1 To number_of_cells
        Print #filenumber, BestSquaresProfile(i).x, BestSquaresProfile(i).y
    Next i
    Print #filenumber, ""
    Print #filenumber, "Best fit profile (absolute differences):"
    For i = 1 To number_of_cells
        Print #filenumber, BestAbsProfile(i).x, BestAbsProfile(i).y
    Next i

    Close #filenumber

    WriteFitData = True
End Function


'LoadParameters() loads parameters from .par or .ev file whose name is contained
'in filename$.  It does so by calling ImportEvParameters() or LoadParFile().
'It returns True upon success.

Public Function LoadParameters(filename$) As Variant
    Dim extension$, char$
    Dim i As Integer
    Dim test As Variant


    'Get the file extension of the parameters file
    i = 1
    While char$ <> "." And i <= Len(filename$)
        extension$ = Right$(filename$, i)
        char$ = Left$(extension$, 1)
        i = i + 1
    Wend

    'Read in the parameters
    If extension$ = ".ev" Or extension = ".EV" Then
        'import the parameters from an .ev file
        test = ImportEvParameters(filename$)
    ElseIf extension$ = ".par" Or ".PAR" Then
        'Read in the contents of a .par file
        test = LoadParFile(filename$)
    End If

    'Update the textboxes in ParametersForm to correspond with what's been loaded
    '(Conversion of the parameters into the correct units is handled by methods
    'in the ParametersForm form.)
    ParametersForm.creep_constant = creep_constant
    ParametersForm.u_wash = u_wash
    ParametersForm.solution_rate = solution_rate
```

```
    ParametersForm.vertical_solution = vertical_solution
    ParametersForm.solution_threshold = solution_threshold
    ParametersForm.lower_threshold = lower_threshold
    ParametersForm.higher_threshold = higher_threshold
    ParametersForm.detachment_constant = detachment_constant
    ParametersForm.travel_constant = travel_constant

    LoadParameters = True

End Function


'LoadParFile() reads in the contents of the .par parameters file whose name
'is contained in filename$

Public Function LoadParFile(filename$) As Variant
    Dim i As Integer, filenumber As Integer
    Dim buffer As Variant
    Dim junk$

    filenumber = FreeFile()
    Open filename$ For Input As #filenumber
    Line Input #filenumber, junk$
    Line Input #filenumber, junk$
    Input #filenumber, number_of_cells
    For i = 1 To 6
        Line Input #filenumber, junk$
    Next i
    Input #filenumber, creep_constant, u_wash, solution_rate, vertical_solution, solution_thresho
ld
    For i = 1 To 4
        Line Input #filenumber, junk$
    Next i
    Input #filenumber, lower_threshold, higher_threshold, detachment_constant, travel_constant
    If EOF(filenumber) = True Then  'Exit this function if the end of file has been reached.
        LoadParFile = True
        Exit Function
    End If
    Line Input #filenumber, junk$
    'I start using a buffer variable below since values read can be either numeric or strings.
    If Left$(junk$, 15) = "PROGRAM OPTIONS" Then
        'If the .par file contains a section specifying program options.
        Line Input #filenumber, junk$
        Input #filenumber, buffer
        If VarType(buffer) <> vbString Then
            'If the buffer variable does not hold a string.
            dt = CDbl(buffer)
        End If
        Line Input #filenumber, junk$
        Input #filenumber, buffer
        If VarType(buffer) <> vbString Then     'ditto
            OptionsForm.chkVaryRemoval = buffer
        End If
        Line Input #filenumber, junk$
        Input #filenumber, buffer
        If VarType(buffer) <> vbString Then     'ditto
            OptionsForm.FractionLeaving = buffer
        End If
        Line Input #filenumber, junk$
        'Below, read whether or not to continuously redraw profile.
        Input #filenumber, buffer
        If buffer = 1 Then
            OptionsForm.OptRedraw = True
        Else
            OptionsForm.OptRedraw = False
        End If
        Line Input #filenumber, junk$
        'read how often to overlay successive profiles.
        Input #filenumber, buffer
        If VarType(buffer) <> vbString Then
```

```
            OptionsForm.UpdateFrequency = buffer
        End If
        'Exit this function if the end of file has been reached.
        If EOF(filenumber) = True Then
            LoadParFile = True
            Exit Function
        End If
        Line Input #filenumber, junk$
    End If
    If Left$(junk$, 16) = "ASSOCIATED FILES" Then
        Line Input #filenumber, junk$
        Input #filenumber, buffer   'Read initial profile filename.
        If buffer <> "NS" Then
            MainForm.InitialProfileFileName = buffer
        End If
        Line Input #filenumber, junk$
        Input #filenumber, buffer   'Read the target profile filename.
        If buffer <> "NS" Then
            MainForm.TargetProfileName = buffer
            MainForm.chkUseTargetProfile = 1
        End If
    End If
    Close #filenumber
    LoadParFile = True
End Function


'ImportEvParameters loads parameters from the .ev file whose name is contained
'in filename$.  It returns True upon success.

Public Function ImportEvParameters(filename$) As Variant
    Dim i As Integer, filenumber As Integer
    Dim junk As Variant     'This really is just a variable for holding junk.

    filenumber = FreeFile()
    Open filename$ For Input As filenumber

    'Read in the data from the .ev file
    For i = 1 To 2
        Line Input #filenumber, junk
    Next i
    Input #filenumber, junk, junk, number_of_cells
    'The next line of code needs a bit of explaining.  When Kirkby's program
    'says it is using, say, 20 cells, it is modelling the hillslope with 21
    'points.  On the other hand, when my program says it is using 20 cells, it
    'uses 20 points to model the hillslope.  To correct for this difference, I
    'add 1 to the number of cells specified in the .ev file.
    number_of_cells = number_of_cells + 1
    For i = 1 To 4
        Line Input #filenumber, junk
    Next i
    Input #filenumber, creep_constant, u_wash, solution_rate
    Input #filenumber, junk
    'Below: if a rate of vertical solution has been specified in the
    '.ev file (not actually supported by the original .ev file format
    If VarType(junk) >= 2 And VarType(junk) <= 5 Then
        vertical_solution = CDbl(junk)
    Else
        vertical_solution = 0
    End If
    For i = 1 To 7
        Line Input #filenumber, junk
    Next i
    Input #filenumber, lower_threshold, higher_threshold, detachment_constant
    For i = 1 To 3
        Line Input #filenumber, junk
    Next i
    Input #filenumber, travel_constant
    Close #filenumber
```

```
    ImportEvParameters = True

End Function


'LoadInitialProfile() loads an initial slope profile from either a .ev or a .par
'file.  It does so by calling ImportEvInitialProfile() or LoadXYInitialProfile().
'It returns True upon success.

Public Function LoadInitialProfile(filename$) As Variant
    Dim i As Integer, filenumber As Integer, test As Variant
    Dim char$, extension$

    'Get the file extension of the initial profile filename
    i = 1
    While char$ <> "." And i <= Len(filename$)
        extension$ = Right$(filename$, i)
        char$ = Left$(extension$, 1)
        i = i + 1
    Wend

    If extension$ = ".ev" Or extension$ = ".EV" Then
        test = ImportEvInitialProfile(filename$)
    Else
        test = LoadXYInitialProfile(filename$)
    End If

    LoadInitialProfile = True
End Function


'ImportEvInitialProfile() loads an initial profile from the .ev file whose
'name is specified in filename$.  It returns True upon success.

Public Function ImportEvInitialProfile(filename$) As Variant
    Dim i As Integer, j As Integer, filenumber As Integer
    Dim test$
    Dim junk As Variant     'This really is just a variable for holding junk.
    Dim x As Double, y As Double, m As Double
    Dim slope_length As Double, initial_height As Double
    Dim xpercent() As Double, ypercent() As Double
    Dim xoriginal() As Double, yoriginal() As Double
    Dim number_of_original_points As Integer

    filenumber = FreeFile()
    Open filename$ For Input As #filenumber

    'Read in the data from the .ev file
    For i = 1 To 2
        Line Input #filenumber, junk
    Next i
    Input #filenumber, slope_length, initial_height, junk
    For i = 1 To 34
        Line Input #filenumber, junk
    Next i

    'Read in the original profile in terms of percentages.
    number_of_original_points = 0
    Do
        test$ = Input(1, #filenumber)
        'If the character is a digit or decimal.
        If (Asc(test$) > 47 And Asc(test$) < 58) Or Asc(test$) = 46 Then
            'The line below moves read/write position back one character.
            Seek #filenumber, (Seek(1) - 1)
            Input #filenumber, x, y, junk
            If Not (x = 0 And y = 0) Then
                number_of_original_points = number_of_original_points + 1
                ReDim Preserve xpercent(number_of_original_points)
                ReDim Preserve ypercent(number_of_original_points)
                xpercent(number_of_original_points) = x
```

```
                ypercent(number_of_original_points) = y
            End If
        End If
    Loop While (Asc(test$) > 47 And Asc(test$) < 58) Or Asc(test$) = 46

    'Convert the slope percentages into (x,y) coordinates.
    ReDim xoriginal(number_of_original_points + 1)
    ReDim yoriginal(number_of_original_points + 1)
    xoriginal(1) = 0
    yoriginal(1) = initial_height
    For i = 1 To number_of_original_points
        xoriginal(i + 1) = xoriginal(i) + slope_length * xpercent(i) / 100
        yoriginal(i + 1) = yoriginal(i) - initial_height * ypercent(i) / 100
    Next i
    'If the x percentages don't add up to 100, we must add a final point at
    'baselevel.
    If xoriginal(number_of_original_points + 1) <> slope_length Then
        ReDim Preserve xoriginal(number_of_original_points + 2)
        ReDim Preserve yoriginal(number_of_original_points + 2)
        xoriginal(number_of_original_points + 2) = slope_length
        yoriginal(number_of_original_points + 2) = 0
    End If

    'Initialize the array of equally spaced cells composing the initial
    'hillslope profile.
    ReDim hillslope_cell(number_of_cells)
    dx = slope_length / (number_of_cells - 1)
    hillslope_cell(1).x = xoriginal(1)
    hillslope_cell(1).y = yoriginal(1)
    For i = 2 To (number_of_cells)
        x = hillslope_cell(i - 1).x + dx
        hillslope_cell(i).x = x
        j = 1
        'While xoriginal(j) < hillslope_cell(i).x
        'Note that the above line of code which is commented out SHOULD work,
        'but doesn't.
        'I think there's a bug in VB.
        'No, I KNOW that there is.
        'Hence I use the line below as a substitute.
        While CSng(xoriginal(j)) < CSng(hillslope_cell(i).x)
            j = j + 1
        Wend
        m = (yoriginal(j) - yoriginal(j - 1)) / (xoriginal(j) - xoriginal(j - 1))   'm = slope of
 the line
        y = m * (x - xoriginal(j)) + yoriginal(j)
        hillslope_cell(i).y = y
    Next i

    Close #filenumber

    ImportEvInitialProfile = True
End Function


'LoadTargetProfile() loads a target profile from either a .ev or a .par file.  It
'does so by calling ImportEvTargetProfile() or LoadXYTargetProfile().  It returns
'true upon success.

Public Function LoadTargetProfile(filename$) As Variant
    Dim i As Integer, test As Variant
    Dim char$, extension$

    'Get the file extension of the parameters file
    i = 1
    While char$ <> "." And i <= Len(filename$)
        extension$ = Right$(filename$, i)
        char$ = Left$(extension$, 1)
        i = i + 1
    Wend
```

```
    If extension$ = ".ev" Or extension$ = ".EV" Then
        test = ImportEvTargetProfile(filename$)
    Else
        test = LoadXYTargetProfile(filename$)
    End If

    LoadTargetProfile = True
End Function


'ImportEvTargetProfile() loads a target profile from the .ev file whose name is
'specified in filename$.  It returns True upon success.

Public Function ImportEvTargetProfile(filename$) As Variant
    Dim i As Integer, j As Integer, filenumber As Integer
    Dim test$
    Dim junk As Variant     'This really is just a variable for holding junk.
    Dim x As Double, y As Double, m As Double
    Dim slope_length As Double, initial_height As Double
    Dim xpercent() As Double, ypercent() As Double
    Dim xoriginal() As Double, yoriginal() As Double
    Dim number_of_original_points As Integer
    Dim number_of_new_points

    Open filename$ For Input As #filenumber

    'Read in the data from the .ev file
    For i = 1 To 2
        Line Input #filenumber, junk
    Next i
    Input #filenumber, slope_length, initial_height, junk
    For i = 1 To 34
        Line Input #filenumber, junk
    Next i

    'Read in the original profile in terms of percentages.
    number_of_original_points = 0
    Do
        test$ = Input(1, #filenumber)
        'i.e., if the character is a digit or decimal.
        If (Asc(test$) > 47 And Asc(test$) < 58) Or Asc(test$) = 46 Then
            'The line below moves read/write position back one character.
            Seek #filenumber, (Seek(1) - 1)
            Input #filenumber, x, y, junk
            If Not (x = 0 And y = 0) Then
                number_of_original_points = number_of_original_points + 1
                ReDim Preserve xpercent(number_of_original_points)
                ReDim Preserve ypercent(number_of_original_points)
                xpercent(number_of_original_points) = x
                ypercent(number_of_original_points) = y
            End If
        End If
    Loop While (Asc(test$) > 47 And Asc(test$) < 58) Or Asc(test$) = 46

    'Convert the slope percentages into (x,y) coordinates.
    ReDim xoriginal(number_of_original_points + 1)
    ReDim yoriginal(number_of_original_points + 1)
    xoriginal(1) = 0
    yoriginal(1) = initial_height
    For i = 1 To number_of_original_points
        xoriginal(i + 1) = xoriginal(i) + slope_length * xpercent(i) / 100
        yoriginal(i + 1) = yoriginal(i) - initial_height * ypercent(i) / 100
    Next i

    number_of_new_points = number_of_original_points + 1

    'If the x percentages don't add up to 100, we must add a final point as
    'baselevel.
    If xoriginal(number_of_original_points + 1) <> slope_length Then
        ReDim Preserve xoriginal(number_of_original_points + 2)
```

```
        ReDim Preserve yoriginal(number_of_original_points + 2)
        xoriginal(number_of_original_points + 2) = slope_length
        yoriginal(number_of_original_points + 2) = 0
        number_of_new_points = number_of_original_points + 2
    End If

    'Exit and return False if initial and target profiles are not of the same
    'length.
    If xoriginal(number_of_new_points) <> hillslope_cell(number_of_cells).x Then
        ImportEvTargetProfile = False
        Exit Function
    End If

    'Initialize the array of equally spaced cells composing the initial
    'hillslope profile.
    ReDim target_profile(number_of_cells)
    dx = slope_length / (number_of_cells - 1)
    target_profile(1).x = xoriginal(1)
    target_profile(1).y = yoriginal(1)
    For i = 2 To (number_of_cells)
        x = target_profile(i - 1).x + dx
        target_profile(i).x = x
        j = 1
        'While xoriginal(j) < target_profile(i).x
        'Note that the above line of code which is commented out SHOULD work,
        'but doesn't.
        'I think there's a bug in VB.
        'No, I KNOW that there is.
        'Hence I use the line below as a substitute.
        While CSng(xoriginal(j)) < CSng(target_profile(i).x)
            j = j + 1
        Wend
        m = (yoriginal(j) - yoriginal(j - 1)) / (xoriginal(j) - xoriginal(j - 1))    'm = slope of
 the line
        y = m * (x - xoriginal(j)) + yoriginal(j)
        target_profile(i).y = y
    Next i

    Close #filenumber

    ImportEvTargetProfile = True
End Function


'WriteSnapshot() writes a snapshot of the current state of the modeled hillslope
'to the file specified by filename$.  It returns True upon success.

Public Function WriteSnapshot(filename$) As Variant
    Dim i As Integer, filenumber As Integer

    filenumber = FreeFile()

    Open filename$ For Output As #filenumber
    Print #filenumber, "Initial Profile Name: " + MainForm.InitialProfileFileName
    Print #filenumber, "Model time elapsed (years):", Time
    Print #filenumber, "Iterations completed:", iteration
    Print #filenumber, ""
    For i = 1 To number_of_cells
        Print #filenumber, hillslope_cell(i).x, hillslope_cell(i).y
    Next i

    Close #filenumber

    WriteSnapshot = True
End Function


'AppendProfile() write the current iteration number, the time elapsed, and the
'current modeled hillslope profile to the disk file whose name is specified in
'$filename.  If filename$ does not exist already, it is created.  Otherwise, the
```

```
'data are appended onto the end of the file.
'
'Note that I never explicitly close the file opened in this function.
'This is to avoid having to close and reopen it several times, which
'could severely limit performance.  The file opened in this function
'SHOULD be closed when the program executes a Reset statement after
'a model run is terminated.
Public Function AppendProfile(filename$) As Variant
    Dim i As Integer
    Static filenumber As Integer

    If iteration = 0 Then
        filenumber = FreeFile()
        Open filename$ For Output As #filenumber
    End If
    Print #filenumber, "Iteration "; iteration, "Years elapsed: "; Time
    For i = 1 To number_of_cells
        Print #filenumber, hillslope_cell(i).x, hillslope_cell(i).y
    Next i
    Print #filenumber, ""

    AppendProfile = True
End Function


'The ResetPublicVariables() procedure resets all of the Public variables declared
'in module1, with the exception of the model_stop flag.

Public Sub ResetPublicVariables()
    Erase hillslope_cell
    Erase target_profile
    Erase BestAbsProfile
    Erase BestSquaresProfile

    iteration = 0
    number_of_cells = 0
    dt = 0
    dx = 0
    Time = 0
    creep_constant = 0
    u_wash = 0
    higher_threshold = 0
    lower_threshold = 0
    travel_constant = 0
    detachment_constant = 0
    solution_rate = 0
    vertical_solution = 0
    solution_threshold = 0
    MinSumAbs = 0
    MinSumSquares = 0
    BestAbsFitTime = 0
    BestSquaresFitTime = 0
    Initial_area = 0
    Flux_out = 0
End Sub


'Main() is the procedure called when the program is opened.

Public Sub Main()
    MainForm.Show
    OptionsForm.Show
End Sub
```

```
VERSION 5.00
Begin VB.Form GraphForm
   AutoRedraw      =   -1  'True
   BackColor       =   &H00FFFFFF&
   Caption         =   "Hillslope profile view:"
   ClientHeight    =   5580
   ClientLeft      =   60
   ClientTop       =   345
   ClientWidth     =   11865
   LinkTopic       =   "Form1"
   PaletteMode     =   1  'UseZOrder
   ScaleHeight     =   5580
   ScaleWidth      =   11865
End
```

```
Private Sub input_file_name_Change()

End Sub

Private Sub BrowseInitialProfileFiles_Click()
    CommonDialog1.Filter = ".dat files (*.dat)|*.dat|.ev files (*.ev)|*.ev|All files (*.*)|*.*"
    CommonDialog1.ShowOpen
    MainForm.InitialProfileFileName = CommonDialog1.filename
End Sub

Private Sub BrowseParameterFiles_Click()
    CommonDialog1.Filter = ".par files (*.par)|*.par|.ev files (*.ev)|*.ev|All files (*.*)|*.*"
    CommonDialog1.ShowOpen
    MainForm.ParametersFileName = CommonDialog1.filename
End Sub

Private Sub BrowseTargetProfiles_Click()
    CommonDialog1.Filter = ".dat files (*.dat)|*.dat|.ev file (*.ev)|*.ev|All files (*.*)|*.*"
    CommonDialog1.ShowOpen
    MainForm.TargetProfileName = CommonDialog1.filename
End Sub

Private Sub chkUseTargetProfile_Click()
    If chkUseTargetProfile = 1 Then
        TargetProfileName.SetFocus
    End If
End Sub

Private Sub ExitProgram_Click()
    End
End Sub

'This is to insure that the entire application is killed off if
'MainForm is terminated.
Private Sub Form_Unload(Cancel As Integer)
    End
End Sub

Private Sub RunModel_Click()
    'v. 1.0 code:
    'OptionsForm.Show
    'Dim test as integer     'This is for error-trapping.
    'test = ImportEv(MainForm.EvFileName)
    'test = LoadXYInitialProfile(MainForm.DatFileName)

    Dim test As Variant     'This will eventually be used for error-trapping.

    iteration = 0
    model_stop = False
    test = LoadParameters(MainForm.ParametersFileName)
    test = LoadInitialProfile(MainForm.InitialProfileFileName)
    test = LoadTargetProfile(MainForm.TargetProfileName)
    ModelRunningForm.Show
    MainLoop
End Sub

Private Sub ShowOptions_Click()
    If OptionsForm.Visible = False Then
        OptionsForm.Show
        ShowOptions.Caption = "Hide Options"
    ElseIf OptionsForm.Visible = True Then
        OptionsForm.Hide
        ShowOptions.Caption = "Show Options"
    End If
End Sub

'This will not work correctly with the time step that must be used with the .ev files,
'due to differences in the various model parameters used here.
Private Sub TestModel_Click()
    'Initialize program variables:
```

```
    'I'm mostly using the values from the file slope1.ev that comes with Kirkby's book.
    dx = 1
    number_of_cells = 50
    creep_constant = 10 * 0.0001
    u_wash = 200
    solution_rate = 10 * 0.000001
    lower_threshold = Tan(PI / 180 * 22)
    higher_threshold = Tan(PI / 180 * 35)
    detachment_constant = 50 * 0.001 'I *think* this is right.
    travel_constant = 20

    ReDim hillslope_cell(number_of_cells)

    'Set up the initial slope profile:
    'This one is a normal fault scarp.
    For i = 1 To 50
        If i <= 15 Then
            hillslope_cell(i).y = 25
        ElseIf i > 36 Then
            hillslope_cell(i).y = 0
        Else
            hillslope_cell(i).y = -(25 / 21) * i + 300 / 7
        End If
        hillslope_cell(i).x = i
    Next i

    MainLoop

End Sub
```

```
VERSION 5.00
Object = "{F9043C88-F6F2-101A-A3C9-08002B2F49FB}#1.1#0"; "COMDLG32.OCX"
Begin VB.Form MainForm
   BorderStyle     =   1  'Fixed Single
   Caption         =   "Richard's n-store hillslope dynamics simulator (HDS), v. 1.1"
   ClientHeight    =   2865
   ClientLeft      =   6045
   ClientTop       =   2265
   ClientWidth     =   5955
   LinkTopic       =   "Form1"
   MaxButton       =   0   'False
   MinButton       =   0   'False
   PaletteMode     =   1  'UseZOrder
   ScaleHeight     =   2865
   ScaleWidth      =   5955
   Begin VB.CommandButton ExitProgram
      Caption         =   "Exit the program"
      Height          =   855
      Left            =   3840
      TabIndex        =   12
      Top             =   1200
      Width           =   1935
   End
   Begin VB.CommandButton BrowseTargetProfiles
      Caption         =   "Browse target profiles"
      Height          =   255
      Left            =   1800
      TabIndex        =   11
      Top             =   1800
      Width           =   1815
   End
   Begin MSComDlg.CommonDialog CommonDialog1
      Left            =   0
      Top             =   0
      _ExtentX        =   847
      _ExtentY        =   847
      _Version        =   327681
   End
   Begin VB.TextBox InitialProfileFileName
      Height          =   285
      Left            =   240
      TabIndex        =   10
      Top             =   1320
      Width           =   3375
   End
   Begin VB.CommandButton BrowseInitialProfileFiles
      Caption         =   "Browse (x,y) profiles"
      Height          =   255
      Left            =   1800
      TabIndex        =   8
      Top             =   960
      Width           =   1815
   End
   Begin VB.CheckBox ChkDisplayTargetProfile
      Caption         =   "Display target profile"
      Height          =   255
      Left            =   1920
      TabIndex        =   7
      Top             =   2160
      Value           =   1  'Checked
      Width           =   1815
   End
   Begin VB.CheckBox chkUseTargetProfile
      Caption         =   "Use target profile"
      Height          =   255
      Left            =   240
      TabIndex        =   6
      Top             =   2160
      Value           =   1  'Checked
      Width           =   1575
```

```
   End
   Begin VB.TextBox TargetProfileName
      Height          =   285
      Left            =   240
      TabIndex        =   5
      Top             =   2520
      Width           =   3375
   End
   Begin VB.CommandButton RunModel
      Caption         =   "Run Model"
      Height          =   855
      Left            =   3840
      TabIndex        =   3
      Top             =   120
      Width           =   1935
   End
   Begin VB.TextBox ParametersFileName
      Height          =   285
      Left            =   240
      TabIndex        =   1
      Top             =   480
      Width           =   3375
   End
   Begin VB.CommandButton BrowseParameterFiles
      Caption         =   "Browse parameter files"
      Height          =   255
      Left            =   1800
      TabIndex        =   0
      Top             =   120
      Width           =   1815
   End
   Begin VB.Label Label3
      Caption         =   "Initial profile name:"
      Height          =   255
      Left            =   120
      TabIndex        =   9
      Top             =   1080
      Width           =   1575
   End
   Begin VB.Label Label2
      Caption         =   "Target profile name:"
      Height          =   255
      Left            =   120
      TabIndex        =   4
      Top             =   1920
      Width           =   1455
   End
   Begin VB.Label Label1
      Caption         =   "Model parameters file:"
      Height          =   255
      Left            =   120
      TabIndex        =   2
      Top             =   240
      Width           =   1575
   End
End
```

```
Private Sub chkShowOptionsForm_Click()
    If chkShowOptionsForm.Value = 1 Then
        OptionsForm.Show
    ElseIf chkShowOptionsForm.Value = 0 Then
        OptionsForm.Hide
    End If
End Sub

Private Sub chkShowParametersForm_Click()
    If chkShowParametersForm.Value = 1 Then
        ParametersForm.Show
    ElseIf chkShowParametersForm.Value = 0 Then
        ParametersForm.Hide
    End If
End Sub

Private Sub chkShowStatusForm_Click()
    If chkShowStatusForm.Value = 1 Then
        StatusForm.Show
    ElseIf chkShowStatusForm.Value = 0 Then
        StatusForm.Hide
    End If
End Sub

Private Sub DisplayBestFitProfiles_Click()
    Dim junk As Integer
    If MainForm.chkUseTargetProfile = 1 Then
        Call ProfilePlot(BestSquaresProfile, GraphForm, RGB(0, 0, 255))
        Call ProfilePlot(BestAbsProfile, GraphForm, RGB(0, 255, 0))
        Call ProfilePlot(target_profile, GraphForm, RGB(255, 0, 0))
    End If
    junk = MsgBox("Model paused.  Click OK to continue.", vbOKOnly, "Model paused")
End Sub

Private Sub PauseModel_Click()
    Dim response As Integer, test As Variant
    response = MsgBox("Would you like to take a snapshot?", vbYesNo, "Model paused")

    'Now, hopefully, the below code will not end up writing the contents of the
    'hillslope_cell() array in the middle of an iteration.  It should only write
    'after an iteration has been completed, because that is when the DoEvents()
    'function is called.  On my computer, under both Win95 and WinNT 4.0, the
    'writing only occurs after DoEvents() is called.
    If response = vbYes Then
        Dim filename$
        Dim i As Integer
        While char$ <> "." And i <> Len(MainForm.InitialProfileFileName)
            i = i + 1
            char$ = Mid(MainForm.InitialProfileFileName, i, 1)
        Wend
        If char$ = "." Then
            filename$ = Mid(MainForm.InitialProfileFileName, 1, i) + "snp"
        ElseIf Len(MainForm.InitialProfileFileName) = i Then
            filename$ = Mid(MainForm.InitialProfileFileName, 1, i) + ".snp"
        End If
        CommonDialog1.filename = filename$
        CommonDialog1.Filter = ".snp files (*.snp)|*.snp|All files (*.*)|*.*"
        CommonDialog1.ShowSave
        test = WriteSnapshot(CommonDialog1.filename)
    End If
End Sub



Private Sub TerminateModelRun_Click()
    Dim test As Variant

    If MainForm.chkUseTargetProfile = 0 Then
        Dim quit_model As Integer
        quit_model = MsgBox("Are you sure you want to terminate this model run?", vbYesNo, "End m
```

```
odel run?")
        If quit_model = vbNo Then
            model_stop = False
            Exit Sub
        ElseIf quit_model = vbYes Then
            model_stop = True
        End If
    ElseIf MainForm.chkUseTargetProfile = 1 Then
        Dim SaveInfo As Integer
        SaveInfo = MsgBox("Save the best fit information?", vbYesNoCancel, "Model Prompt")
        If SaveInfo = vbCancel Then
            model_stop = False
            Exit Sub
        ElseIf SaveInfo = vbNo Then
            model_stop = True
        ElseIf SaveInfo = vbYes Then        'I should eventually prompt the user for a filename h
ere, instead of just assigning one.
            model_stop = True
            Dim filename$
            Dim i As Integer
            While char$ <> "." And i <> Len(MainForm.InitialProfileFileName)
                i = i + 1
                char$ = Mid(MainForm.InitialProfileFileName, i, 1)
            Wend
            If char$ = "." Then
                filename$ = Mid(MainForm.InitialProfileFileName, 1, i) + "fit"
            ElseIf Len(MainForm.InitialProfileFileName) = i Then
                filename$ = Mid(MainForm.InitialProfileFileName, 1, i) + ".fit"
            End If
            CommonDialog1.filename = filename$
            CommonDialog1.Filter = ".fit files (*.fit)|*.fit|All files (*.*)|*.*"
            CommonDialog1.ShowSave
            test = WriteFitData(CommonDialog1.filename)
        End If
    End If

    Reset   'Close any files that are still open.

    'Now reset everything so a new model run can be performed.
    'First, unload all of the forms except MainForm.
    '(Unloading MainForm would kill the application.)
    Unload GraphForm
    Unload StatusForm
    Unload ParametersForm
    Unload OptionsForm
    Unload ModelRunningForm

    'Now reload and show OptionsForm, whose properties have been
    'reset to their initial values by the Unload statement.
    Load OptionsForm
    OptionsForm.Show

    'Now reset all of the public variables.
    Call ResetPublicVariables

End Sub
```

```
VERSION 5.00
Object = "{F9043C88-F6F2-101A-A3C9-08002B2F49FB}#1.1#0"; "COMDLG32.OCX"
Begin VB.Form ModelRunningForm
   Caption         =   "Model Running...."
   ClientHeight    =   1620
   ClientLeft      =   150
   ClientTop       =   6435
   ClientWidth     =   6015
   LinkTopic       =   "Form1"
   ScaleHeight     =   1620
   ScaleWidth      =   6015
   Begin MSComDlg.CommonDialog CommonDialog1
      Left            =   2160
      Top             =   960
      _ExtentX        =   847
      _ExtentY        =   847
      _Version        =   327681
   End
   Begin VB.CheckBox chkShowStatusForm
      Caption         =   "Show status window"
      Height          =   255
      Left            =   240
      TabIndex        =   5
      Top             =   1080
      Value           =   1  'Checked
      Width           =   1815
   End
   Begin VB.CheckBox chkShowParametersForm
      Caption         =   "Show parameters window"
      Height          =   375
      Left            =   240
      TabIndex        =   4
      Top             =   600
      Width           =   2175
   End
   Begin VB.CheckBox chkShowOptionsForm
      Caption         =   "Show options window"
      Height          =   255
      Left            =   240
      TabIndex        =   3
      Top             =   240
      Value           =   1  'Checked
      Width           =   1935
   End
   Begin VB.CommandButton TerminateModelRun
      Caption         =   "Terminate Model Run"
      Height          =   375
      Left            =   2760
      TabIndex        =   2
      Top             =   1080
      Width           =   3135
   End
   Begin VB.CommandButton DisplayBestFitProfiles
      Caption         =   "Pause and display best fit profiles"
      Height          =   375
      Left            =   2760
      TabIndex        =   1
      Top             =   600
      Width           =   3135
   End
   Begin VB.CommandButton PauseModel
      Caption         =   "Pause model and (optional) take snapshot"
      Height          =   375
      Left            =   2760
      TabIndex        =   0
      Top             =   120
      Width           =   3135
   End
   Begin VB.Shape Shape1
      Height          =   1335
```

```
      Left            =   120
      Top             =   120
      Width           =   2415
   End
End
```

```
Private Sub BrowseOutputFiles_Click()
    CommonDialog1.Filter = ".out files (*.out)|*.out|All files (*.*)|*.*"
    CommonDialog1.ShowOpen
    OptionsForm.OutputFilename = CommonDialog1.filename
End Sub

Private Sub chkWriteSuccessiveProfiles_Click()
    If chkWriteSuccessiveProfiles = 1 Then
        WriteFrequency.SetFocus
    End If
End Sub
```

```
VERSION 5.00
Object = "{F9043C88-F6F2-101A-A3C9-08002B2F49FB}#1.1#0"; "COMDLG32.OCX"
Begin VB.Form OptionsForm
   BorderStyle     =   1  'Fixed Single
   Caption         =   "Program Options"
   ClientHeight    =   2850
   ClientLeft      =   6480
   ClientTop       =   5625
   ClientWidth     =   5475
   LinkTopic       =   "Form1"
   MaxButton       =   0   'False
   MinButton       =   0   'False
   PaletteMode     =   1  'UseZOrder
   ScaleHeight     =   2850
   ScaleWidth      =   5475
   Begin MSComDlg.CommonDialog CommonDialog1
      Left            =   480
      Top             =   1200
      _ExtentX        =   847
      _ExtentY        =   847
      _Version        =   327681
   End
   Begin VB.CommandButton BrowseOutputFiles
      Caption         =   "Browse output files"
      Height          =   255
      Left            =   3240
      TabIndex        =   16
      Top             =   2040
      Width           =   1815
   End
   Begin VB.TextBox OutputFilename
      Height          =   285
      Left            =   600
      TabIndex        =   15
      Top             =   2400
      Width           =   4455
   End
   Begin VB.TextBox WriteFrequency
      Height          =   285
      Left            =   3240
      TabIndex        =   12
      Top             =   1680
      Width           =   615
   End
   Begin VB.CheckBox chkWriteSuccessiveProfiles
      Caption         =   "Write successive profiles to file every"
      Height          =   255
      Left            =   240
      TabIndex        =   11
      Top             =   1680
      Width           =   3015
   End
   Begin VB.CheckBox chkVaryRemoval
      Caption         =   "Dynamically vary basal removal"
      Height          =   255
      Left            =   120
      TabIndex        =   10
      ToolTipText     =   "This feature is experimental.  I don't think it works quite right."
      Top             =   480
      Width           =   2535
   End
   Begin VB.TextBox FractionLeaving
      Height          =   285
      Left            =   1800
      TabIndex        =   9
      Text            =   "1"
      ToolTipText     =   "Proportion of the flux into the basal cell that is carried away."
      Top             =   960
      Width           =   615
   End
```

```
   Begin VB.TextBox Timestep
      Height          =   285
      Left            =   1800
      TabIndex        =   7
      Text            =   "50"
      ToolTipText     =   "(Too large a time step leads to numerical instabilities.)"
      Top             =   120
      Width           =   615
   End
   Begin VB.Frame Frame1
      Caption         =   "Profile display options:"
      Height          =   1335
      Left            =   2760
      TabIndex        =   0
      Top             =   120
      Width           =   2535
      Begin VB.TextBox UpdateFrequency
         Height          =   285
         Left            =   840
         TabIndex        =   4
         Text            =   "200"
         Top             =   960
         Width           =   615
      End
      Begin VB.OptionButton OptNoRedraw
         Caption         =   "Overlay successive profiles"
         Height          =   255
         Left            =   120
         TabIndex        =   2
         Top             =   720
         Value           =   -1  'True
         Width           =   2295
      End
      Begin VB.OptionButton OptRedraw
         Caption         =   "Continuously redraw profile"
         Height          =   255
         Left            =   120
         TabIndex        =   1
         Top             =   360
         Width           =   2295
      End
      Begin VB.Label Label2
         Caption         =   "iterations."
         Height          =   255
         Left            =   1560
         TabIndex        =   5
         Top             =   960
         Width           =   735
      End
      Begin VB.Label Label1
         Caption         =   "every"
         Height          =   255
         Left            =   360
         TabIndex        =   3
         Top             =   960
         Width           =   495
      End
   End
   Begin VB.Shape Shape1
      Height          =   1215
      Left            =   120
      Top             =   1560
      Width           =   5055
   End
   Begin VB.Label Label6
      Caption         =   "Output file name:"
      Height          =   255
      Left            =   600
      TabIndex        =   14
      Top             =   2040
```

```
      Width           =   1335
   End
   Begin VB.Label Label5
      Caption         =   "iterations."
      Height          =   255
      Left            =   3960
      TabIndex        =   13
      Top             =   1680
      Width           =   735
   End
   Begin VB.Label Label4
      Caption         =   "Fraction leaving base: (fixed removal only)"
      Height          =   375
      Left            =   120
      TabIndex        =   8
      ToolTipText     =   "Proportion of the flux into the basal cell that is carried away."
      Top             =   840
      Width           =   1575
   End
   Begin VB.Label Label3
      Caption         =   "Use fixed time step of:"
      Height          =   255
      Left            =   120
      TabIndex        =   6
      ToolTipText     =   "(Too large a time step leads to numerical instabilities.)"
      Top             =   120
      Width           =   1575
   End
End
```

```
Private Sub creep_constant_Change()
    Module1.creep_constant = CDbl(ParametersForm.creep_constant) * 0.0001
End Sub

Private Sub detachment_constant_Change()
    Module1.detachment_constant = CDbl(ParametersForm.detachment_constant) * 0.001
End Sub

Private Sub higher_threshold_Change()
    Module1.higher_threshold = Tan(PI / 180 * CDbl(ParametersForm.higher_threshold))
End Sub

Private Sub lower_threshold_Change()
    Module1.lower_threshold = Tan(PI / 180 * CDbl(ParametersForm.lower_threshold))
End Sub

Private Sub solution_rate_Change()
    Module1.solution_rate = CDbl(ParametersForm.solution_rate) * 0.000001
End Sub

Private Sub solution_threshold_Change()
    Module1.solution_threshold = CDbl(ParametersForm.solution_threshold)
End Sub

Private Sub travel_constant_Change()
    Module1.travel_constant = CDbl(ParametersForm.travel_constant)
End Sub

Private Sub u_wash_Change()
    Module1.u_wash = CDbl(ParametersForm.u_wash)
End Sub

Private Sub vertical_solution_Change()
    Module1.vertical_solution = CDbl(ParametersForm.vertical_solution) * 0.000001
End Sub
```

```
VERSION 5.00
Begin VB.Form ParametersForm
   Caption         =   "Model Parameters"
   ClientHeight    =   3495
   ClientLeft      =   150
   ClientTop       =   2370
   ClientWidth     =   5700
   LinkTopic       =   "Form1"
   ScaleHeight     =   3495
   ScaleWidth      =   5700
   Begin VB.TextBox travel_constant
      Height          =   285
      Left            =   4560
      TabIndex        =   17
      Top             =   3120
      Width           =   495
   End
   Begin VB.TextBox detachment_constant
      Height          =   285
      Left            =   4560
      TabIndex        =   16
      Top             =   2760
      Width           =   495
   End
   Begin VB.TextBox higher_threshold
      Height          =   285
      Left            =   4560
      TabIndex        =   15
      Top             =   2400
      Width           =   495
   End
   Begin VB.TextBox lower_threshold
      Height          =   285
      Left            =   4560
      TabIndex        =   14
      Top             =   2040
      Width           =   495
   End
   Begin VB.TextBox solution_threshold
      Height          =   285
      Left            =   4560
      TabIndex        =   13
      Top             =   1680
      Width           =   495
   End
   Begin VB.TextBox vertical_solution
      Height          =   285
      Left            =   4560
      TabIndex        =   12
      Top             =   1320
      Width           =   495
   End
   Begin VB.TextBox solution_rate
      Height          =   285
      Left            =   4560
      TabIndex        =   11
      Top             =   960
      Width           =   495
   End
   Begin VB.TextBox u_wash
      Height          =   285
      Left            =   4560
      TabIndex        =   10
      Top             =   600
      Width           =   495
   End
   Begin VB.TextBox creep_constant
      Height          =   285
      Left            =   4560
      TabIndex        =   9
```

```
      Top             =   240
      Width           =   495
   End
   Begin VB.Label Label9
      Caption         =   "Landslide travel distance [m]:"
      Height          =   255
      Left            =   120
      TabIndex        =   8
      Top             =   3120
      Width           =   2175
   End
   Begin VB.Label Label8
      Caption         =   "Rate of free degradation above threshold [mm/y]:"
      Height          =   255
      Left            =   120
      TabIndex        =   7
      Top             =   2760
      Width           =   3615
   End
   Begin VB.Label Label7
      Caption         =   "Solution threshold angle (for Kirkby-type) [degrees]:"
      Height          =   255
      Left            =   120
      TabIndex        =   6
      Top             =   1680
      Width           =   3615
   End
   Begin VB.Label Label6
      Caption         =   "Talus angle [degrees] (above which debris will not stop):"
      Height          =   255
      Left            =   120
      TabIndex        =   5
      Top             =   2400
      Width           =   4095
   End
   Begin VB.Label Label5
      Caption         =   "Landslide threshold angle [degrees]:"
      Height          =   255
      Left            =   120
      TabIndex        =   4
      Top             =   2040
      Width           =   2655
   End
   Begin VB.Label Label4
      Caption         =   "Solution rate (uniform vertical lowering model) [micro m/y]:"
      Height          =   255
      Left            =   120
      TabIndex        =   3
      Top             =   1320
      Width           =   4095
   End
   Begin VB.Label Label3
      Caption         =   "Solution rate (Kirkby-type) [micro m/y]:"
      Height          =   255
      Left            =   120
      TabIndex        =   2
      Top             =   960
      Width           =   2775
   End
   Begin VB.Label Label2
      Caption         =   "Distance [m] at which wash becomes greater than creep etc:"
      Height          =   255
      Left            =   120
      TabIndex        =   1
      Top             =   600
      Width           =   4335
   End
   Begin VB.Label Label1
      Caption         =   "Creep/Splash (c.10) or Solifluction (c.100) rate [sq.cm /y]:"
      Height          =   255
```

```
      Left             =   120
      TabIndex         =   0
      Top              =   240
      Width            =   4095
   End
End
```

```
Public Sub DisplayStatus()
    StatusForm.Years.Caption = Time
    StatusForm.Iterations.Caption = iteration
End Sub
```

```
VERSION 5.00
Begin VB.Form StatusForm
   Caption         =   "Model status"
   ClientHeight    =   2280
   ClientLeft      =   8235
   ClientTop       =   435
   ClientWidth     =   3705
   LinkTopic       =   "Form1"
   PaletteMode     =   1  'UseZOrder
   ScaleHeight     =   2280
   ScaleWidth      =   3705
   Begin VB.Label MinMeanSquaredDifference
      Caption         =   "N/A"
      Height          =   255
      Left            =   2400
      TabIndex        =   11
      Top             =   1200
      Width           =   1215
   End
   Begin VB.Label MinMeanAbsDifference
      Caption         =   "N/A"
      Height          =   255
      Left            =   2400
      TabIndex        =   10
      Top             =   1920
      Width           =   1335
   End
   Begin VB.Label Label6
      Caption         =   "Min Mean Abs Difference:"
      Height          =   255
      Left            =   240
      TabIndex        =   9
      Top             =   1920
      Width           =   1935
   End
   Begin VB.Label Label5
      Caption         =   "Min Mean Squared diff:"
      Height          =   255
      Left            =   240
      TabIndex        =   8
      Top             =   1200
      Width           =   1935
   End
   Begin VB.Label BestAbsFitTime
      Caption         =   "N/A"
      Height          =   255
      Left            =   2400
      TabIndex        =   7
      Top             =   1560
      Width           =   1215
   End
   Begin VB.Label BestSquaresFitTime
      Caption         =   "N/A"
      Height          =   255
      Left            =   2400
      TabIndex        =   6
      Top             =   840
      Width           =   1215
   End
   Begin VB.Label Label4
      Caption         =   "Time of best fit (abs):"
      Height          =   255
      Left            =   240
      TabIndex        =   5
      Top             =   1560
      Width           =   1935
   End
   Begin VB.Label Label3
      Caption         =   "Time of best fit (squares):"
      Height          =   255
```

```
      Left            =   240
      TabIndex        =   4
      Top             =   840
      Width           =   1935
   End
   Begin VB.Label Iterations
      Caption         =   "Iterations"
      Height          =   255
      Left            =   2400
      TabIndex        =   3
      Top             =   480
      Width           =   1215
   End
   Begin VB.Label Years
      Caption         =   "Years"
      Height          =   255
      Left            =   2400
      TabIndex        =   2
      Top             =   120
      Width           =   1215
   End
   Begin VB.Label Label2
      Caption         =   "Iterations completed:"
      Height          =   255
      Left            =   240
      TabIndex        =   1
      Top             =   480
      Width           =   1935
   End
   Begin VB.Label Label1
      Caption         =   "Model time elapsed (years):"
      Height          =   255
      Left            =   240
      TabIndex        =   0
      Top             =   120
      Width           =   1935
   End
End
```