N87-26534

PROGRAMMING A HILLSLOPE WATER MOVEMENT MODEL ON THE MPP

J. E. Devaney, A. R. Irving
Science Applications Research
Lanham, Maryland

P. J. Camillo, R. J. Gurney
NASA Goddard Space Flight Center
Greenbelt, Maryland

ABSTRACT

We have developed a physically based numerical model of heat and moisture flow within a hillslope on a parallel architecture computer, as a precursor to a model of a complete catchment. Moisture flow within a catchment includes evaporation, overland flow, flow in unsaturated soil, and flow in saturated soil. Because of the empirical evidence that moisture flow in unsaturated soil is mainly in the vertical direction, flow in the unsaturated zone can be modelled as a series of one-dimensional columns. This initial version of the hillslope model includes evaporation and a single column of one-dimensional unsaturated zone flow. This case has already been solved on an IBM 3081 computer and is now being applied to the MPP architecture so as to make the extension to the one-dimensional case easier and to check the problems and benefits of using a parallel-architecture machine.

Keywords: Hydrology, Hillslopes, Parallel Processing, Unsaturated Flow, Evaporation.

INTRODUCTION

One important part of the global hydrological system is a catchment, which separates rainfall into three parts: evaporation, overland flow, which goes directly to a stream, and infiltration, which flows at a much slower rate vertically through the soil to a saturated zone, where the water then flows horizontally and merges into the stream or is stored as groundwater. This involves four components: evaporation, overland flow, flow in unsaturated soil and flow in saturated soil. If we are to understand the way in which spatial variations of hydrological parameters affect the water balance of a catchment, including evaporation, runoff, erosion and transport of minerals, we have to model the flow over and within a hillside explicitly.

Several workers have written hillslope models (Ref. 6). All of these have had limitations mainly because they could not be executed in a reasonable amount of time. This computer limitation is much reduced if a parallel processor is used, as it is possible to write the model in such a way that it can be solved in parallel at many points at one time.

The first stage of the work is to verify that such a model may be efficiently executed on a parallel processor. To this end we have coded and tested a model which already exists (Ref. 2) on a serial computer, an IBM 3081, and which contains two of the four previously identified components, namely evaporation and unsaturated flow. We have transferred an existing model to the MPP so that we may verify the accuracy of the parallel computations. We also plan to estimate the computational efficiency of the full catchment model in a parallel machine over the equivalent model on a serial machine,

if such a serial model were to be successfully created.

In this paper we discuss the physics of the one dimensional model, the method of solution, and its adaptability to the parallel architecture.

Model Description

The complete soil model is described in Ref. 2. The soil moisture profile in the unsaturated zone is the solution of the continuity equation

$$\frac{\partial \theta}{\partial t} = - \frac{\partial q_m}{\partial z} \qquad (1)$$

where $\theta(z,t)$ is the volumetric soil moisture (water volume/soil volume) at depth z at time t, and q_m is the vertical soil moisture flux, modelled by (Ref. 8)

$$q_m = K - D_\theta \left(\frac{\partial \theta}{\partial z}\right) \qquad (2)$$

K is the hydraulic conductivity, and D_θ is the moisture diffusion coefficient, which depends on soil moisture θ, the soil matric potential ψ and other physical constants. K and ψ are estimated with the parameterization (Ref. 3)

$$K(\theta) = K_s(\theta/\theta_s)^{2b+3} \qquad (3a)$$

$$\psi(\theta) = \psi_s(\theta/\theta_s)^{-b} \qquad (3b)$$

in which θ_s, K_s, and ψ_s are moisture content, conductivity, and potential at saturation. The value of b depends on soil texture.

The temperature profile in the soil may be modelled with Fourier's equations and is one option in the serial version. However, we have chosen to implement the computationally simpler yet physically adequate force-restore equations (Ref. 7). The surface and deep soil temperatures T_s and $\overline{T}$ are modelled by

$$\frac{\partial T_s}{\partial t} = \frac{2G}{a} - \frac{2\pi}{\tau}(T_s - \overline{T}) \qquad (4a)$$

$$\frac{\partial \overline{T}}{\partial t} = \frac{G}{a\sqrt{365\,\pi}} \qquad (4b)$$

$$a = \sqrt{\lambda c \tau/\pi} \qquad (4c)$$

where G is the soil surface heat flux, λ and c are respectively the soil thermal conductivity and heat capacity, and τ is the length of the day. $\overline{T}$ is the temperature at the depth where fluctuations are seasonal rather than diurnal. For most soils this depth is about 2 meters. The conductivity and heat capacity are modelled as functions of soil moisture and soil type with the model of Ref. 4.

To solve these equations for θ, T_s, and $\overline{T}$, boundary conditions must be supplied for moisture and temperature both at the air/soil interface and in the bottom layer of the profile. In principle, either the fluxes q_θ and G or the variables θ and T could be specified. In the model, surface heat and moisture fluxes are computed to model the effects of the environment (i.e., rainfall, evapotranspiration, radiation, etc.) on the profile evolution. At the bottom of the moisture profile a choice of flux or moisture boundary condition is used. One can specify constant moisture, a downward moisture flux equal to the hydraulic constant of the bottom layer, or any constant value.

The energy balance equation provides the surface fluxes:

$$G = R + LE + H \qquad (5)$$

All fluxes are positive downward. G is the heat absorbed by the soil, R is

the net radiation flux, LE is the evapotranspiration energy flux, and H is the sensible heat. After finding the solution, the surface moisture flux q_θ is set equal to E and G is used in the force-restore equations.

The net radiation R is divided into average short- and long-wavelength components:

$$R = R_{short} + R_{long} \tag{6}$$

Either or both components may be either estimated or measured. All four options are allowable within the computer program, standard models such as the Brunt model for long-wave radiation being available as options to estimate either or both components.

A standard model for the latent heat flux under neutral atmospheric stability is (Ref. 5)

$$LE = -\frac{\rho c_p k^2 \overline{U}_a}{\gamma \ln^2(z/z_0)}(e_s - e_a) \tag{7}$$

where ρ is the density of air (1.15×10^{-3} g cm^{-3}), c_p the air specific heat, k the von Karman constant (0.4), z_0 the surface roughness parameter, $\overline{U}_a$ the wind velocity (centimeters per second) at height z averaged over a suitable time period (~ 1 hour), e_a the vapor pressure at height z, e_s the vapor pressure at the soil surface, and γ the psychrometric constant (0.61808 mbar K^{-1}).

This may be expressed as

$$LE = -C_1 \overline{U}_a (e_s - e_a) \tag{8a}$$

$$\text{where } C_1 = \frac{\rho c_p k^2}{\gamma \ln^2(z/z_0)} \tag{8b}$$

Input to the program includes the constant C_1 and both $\overline{U}_a$ and e_a as functions of time.

The sensible heat flux H in the continuity equation (21) is calculated by

$$H = -\gamma C_1 \overline{U}_a (T_s - T_a) \tag{9}$$

The terms of heat balance equations are functions of the known surface temperature T_s and the meteorological variables e_a, $\overline{U}_a$, and T_a.

METHOD OF SOLUTION

The continuity equations (1 and 4) are solved by expressing the spatial derivatives of the moisture fluxes as finite differences, and then using a fourth order predictor-corrector for the time integration.

The soil is divided into N layers of varying thicknesses Δz_i, where N and Δz_i are input variables. At a particular time the moisture fluxes at the N-1 interior boundaries are calculated by evaluating Eq. 2. The surface and bottom fluxes are computed by evaluating the boundary condition equations at the top and bottom boundaries. This gives the flux at the N+1 layer boundaries, and the derivative with respect to depth is approximated for the ith layer by

$$\left(\frac{\partial q_m}{\partial z}\right) \cong \frac{q_{m_{i+1}} - q_{m_i}}{\Delta z_i} \tag{10}$$

The continuity equations (1 and 4) are of the form

$$\frac{\partial \vec{y}}{\partial t} = \vec{f}(t, \vec{y}) \tag{11}$$

where the state vector $\vec{y}$ has N+2 elements, soil moisture θ_i in each of N layers, T_s and $\overline{T}$. The first N elements of $\vec{f}$ are Eq. 10, and f_{n+1}

and f_{n+2} are the right hand sides of the force restore equations, 4a and 4b.

This is readily solved by an Adams-Bashforth predictor-corrector algorithm (Ref. 1,9). We first introduce the backward difference operator ∇ such that

$$\nabla \vec{f}(t) = \vec{f}(t) - \vec{f}(t - \nabla t)$$

Higher order backward differences are evaluated by successive applications of the operator;

$$\nabla^2 \vec{f}(t) = \nabla(\nabla \vec{f}(t)) = \vec{f}(t) - 2\vec{f}(t-\Delta t) - \vec{f}(t-2\Delta t)$$

$$\nabla^3 \vec{f}(t) = \vec{f}(t) - 3\vec{f}(t-\Delta t) + 3\vec{f}(t-2\Delta t) - \vec{f}(t-3\Delta t)$$

$$\nabla^4 \vec{f}(t) = \vec{f}(t) - 4\vec{f}(t-\Delta t) + 6\vec{f}(t-2\Delta t) - 4\vec{f}(t-3\Delta t) + \vec{f}(t-4\Delta t)$$

Using only the state vector $\vec{y}$ and the physical model $\vec{f}$, at time t, an estimate (called the predictor) at time $t+\Delta t$ is

$$\vec{y}^{(p)}(t+\Delta t) = \vec{y}(t) + \Delta t\left[1 + \frac{1}{2}\nabla + \frac{5}{12}\nabla^2 + \frac{3}{8}\nabla^3 + \frac{251}{720}\nabla^4\right]\vec{f}(t)$$

Then, using $\vec{y}^{(p)}(t+\Delta t)$ to evaluate the model $\vec{f}(t+\Delta t)$ at the $t+\Delta t$, one may compute another estimate of the state vector called the corrector;

$$\vec{y}^{(c)}(t+\Delta t) = \vec{y}(t) + \frac{251}{720}\Delta t\, \vec{f}(t+\Delta t) + \frac{\Delta t}{720}[469 + 109\nabla + 49\nabla^2 + 19\nabla^3]\vec{f}(t) \qquad (13)$$

The physical model equations are evaluated only once each time step to calculate $\vec{f}(t+\Delta t)$. The difference between $\vec{y}^{(p)}$ and $\vec{y}^{(c)}$ is a reliable estimate of the error, and the software determines if each element of this difference lies within a user specified window. If all differences are smaller than this window, the integration step size (Δt) is doubled, leading to increased computational efficiency. If any difference is too large, the step size is halved. This halving and doubling requires no re-evaluation of the model equations. The values of the four backward differences for the new integrator time (whether for halving, doubling, integration) are calculated as linear combinations of the four back values for the old integrator time. The fact that the same calculations are done on many pieces of data at the same time as well as the boolean nature of the error window checks make the Hillslope Model an ideal application for a Single Instruction Multiple Data (SIMD) Massively Parallel Processor (MPP).

Mapping the Hillslope Model to the MPP Architecture

The initial mapping the Hillslope model to the MPP involved three phases:

1. Determination of scalar and parallel components of the calculations
2. Data initialization on the MPP
3. Data output to the VAX

Both calculations within the layers

of the hillslope and at the boundaries are required at each time step of the model integrations. The calculations within the hillslope are exactly the same at each layer, so they were set up so that they could be done in unison by means of parallel arrays in the MPP Array Unit. These parallel values consisted of the soil moisture, temperature, depth, layer thicknesses, fluxes, derivatives, backward differences, as well as predicted and corrected values. Because the model was set up with a view to extension to a two dimensional model, each row in the Array Unit was to represent one vertical column in the hillslope, with the components of the row representing the layers in it. This initial version of the model was set up with 14 layers in the soil column to match the serial calculations. Thus in this one-dimensional rendition, only part of the first row of the Array Unit was used for each parallel array of data. In addition, since the temperature and moisture values constitute the state vector $(\vec{y})$ and thus require the same type of computations, they were placed together in the first row of the array unit. While this is initially wasteful in terms of the computational power of the MPP, it allows an easy extension to a two dimensional model which will use the other rows and hence the full capabilities of the MPP.

The boundary conditions involve only single values at the top and bottom of the soil column modified with scalar input data, and so they may be more efficiently done serially with the Main Control Unit (MCU). Thus, implementing the model on the MPP involved scalar and parallel components as well as communication between the scalar and parallel components for boundary condition values. The MPP architecture is very efficient at passing scalar values back and forth between the MCU and the Array Unit as it has a register designed for this purpose. Two special purpose assembly routines were written to take advantage of this. One took a user specified row and column in a parallel array and placed a given scalar value there. Another took a user specified row and column in a parallel array and retrieved a value into a scalar in the MCU. These were used in the boundary condition calculations.

Data intialization on the MPP therefore involved initialization of parallel arrays as well as scalar data. These arrays were initialized in FORTRAN arrays and scalars on the VAX and transferred over to the MPP scalars and arrays via the DR780 and DR11b buffers. The capability of the stager to permute the data bits between the VAX and the Array Unit was used to change the format of the floating point data from the VAX format to the MPP format while the data was being transferred between the VAX and the MPP. Because of this, parallel data transmission between the VAX and the MPP appeared transparent. Explicit bit swapping of scalar integer data between the VAX and MCU to accomodate the two separate integer formats was still necessary, however, as these values were transmitted across the DR11b which has no data permuting capability.

Special purpose routines were written to enable data output to the VAX of the information in the parallel arrays. These routines passed parallel arrays into predefined VAX FORTRAN arrays. MPP Pascal callable VAX FORTRAN routines were written which could write out the data in these arrays for user examination of intermediate and final results. In addition, some high level I/O equivalent to Pascal 'writeln' was written to speed the debugging process by bypassing the CAD debugger entirely.

The next step in mapping the Hillslope model to the MPP architecture involved implementing the physical model

equations in MPP Pascal.

The predicted and corrected state vector values (soil moisture and temperature) involve a calculation of the type (see Eqs. 12 and 13)

$$\vec{y}_{n+1} = \vec{y}_n + \Delta t \sum_i W_i * tp_i$$

where

W_i are the backward differences for each layer

tp_i are the constant scalar coefficients in the predictor equation

By assigning the backward differences to individual parallel arrays so that order ith backward differences for each layer are stored in the first row of the ith parallel array, the above calculation can be solved for each layer with a series of parallel operations. $W_i * (tp_i * \Delta t)$ involves only multiplication of a parallel array by a scalar as the coefficients of the predictor equation would be the same for each layer with the same order backward differences. The remaining sums can be done for each layer in parallel. Once the predicted and corrected values of each layer are available, their differences can be simultaneously calculated. Comparisons with user input error windows can be done all at once with simple boolean tests on the parallel arrays. Recalculation of the backward differences for each layer can be done in unison with the backward difference arrays.

There are additional calculations needed at each layer and each time step in order to include the dependencies of the heat and moisture fluxes on the moisture and temperature profiles through the column. These involve the matric potential, the hydraulic conductivity, the moisture fluxes as well as the derivatives of the moisture fluxes and temperatures. Special calculations must be done at the boundaries in order to include the effects of the air/soil interface at the top as well as the effects of saturated soil at the bottom of the column.

The matric potential and hydraulic conductivity calculations involve only parallel computations. They both depend on the calculation of the type:

```
result := a * A^b
```

where 'a' and 'b' are scalars and 'A' is a parallel array (which is the same for both). A parallel version of this was calculated by:

```
temp := ln( A )
result := a * exp( b * temp )
```

where 'temp' is a temporary variable.

The savings in time by only calculating the natural logarithm once and doing the calculations in parallel will be considerable as the model is extended to more dimensions. The derivatives are also easily available through parallel operations as the change between layers can be obtained easily through the 'shift' operator and the thickness of each layer is stored in a parallel array. The same is true of the fluxes. Computation of thermal parameters involved mixing some scalar values with information from various points in parallel arrays and storing the results back in the parallel arrays. Here is where the special purpose routines were used. Simple 'get' and 'put' routines were written to get/put a value out of/into a user specified row and column in a parallel array. This could be done quickly using the special capabilities of the MPP architecture. Moreover, as the needed scalar computations were being done, parallel operations could be

performed concurrently in the Array Unit.

Minimizing the Program Execution Time by Using the Capabilities of the MPP Through MPP Pascal

The multiprocessing capabilities of the MPP are easily available through MPP Pascal. Scalar calculations are performed in the Main Control Unit. This is a special purpose microcoded 16-bit processor which has a 16 bit hardware multiplier. Parallel calcuations are performed in the Array Unit with 16384 specially designed processors. The scalar Main Control Unit calculations and the parallel Array Unit calculations are done simultaneously except when the Main Control unit is expecting a scalar result from the Array Unit. This would be the case, for example, when doing a maximum, minimum or sum operation on a parallel array. MPP Pascal produces a code which runs in the Main Control Unit and makes calls to library and special purpose routines which run in the Array Unit. There is also a call queue which enables the Main Control Unit to stack its calls (including register transfer data) to the Array Unit. These calls may be stacked up to 15 deep. Thus a parallel operation, such as an assign, in MPP Pascal translates to a single call by the Main Control Unit to the Array Unit to begin its processing with its own processors. The Main Control Unit is then free to do either scalar calculations or send another parallel operation request to the Array Unit. By recognizing that the Main Control Unit is a serial processor it becomes apparent that sending requests to the Array Unit to perform parallel operations and then doing scalar operations in the Main Control Unit allows the scalar and parallel calculations to be done at the same time.

This feature of the MPP was used extensively in the boundary condition calculations to reduce program execution time. It proved to be the single most important tool for reduction of program running time. Other techniques involved setting up masks at initialization time and reusing them instead of regenerating them with 'WHERE' statements, and also the use of temporary stores for the results of natural logarithm and exponent functions which were to be used in more than one calculation.

A 24 hour simulation on the MPP with 14 layers used 30 seconds of computer time, whereas the identical simulation on the IBM 3081 serial machine used 4 seconds. Adding more layers to the MPP model would use virtually the same amount of time, whereas the execution time on the serial machine is approximately linear with the number of layers. We expect, then, that the break even point is approximately 115 layers. Modelling an entire catchment could easily require ten times this number, so we expect that the parallel architecture of the MPP will provide significant savings in computer time over a similar model on a serial machine.

CONCLUSION

We have coded a one-dimensional hydrological model of the surface energy and moisture balance and moisture flow in the unsaturated zone, as a precursor to a complete catchment model.

By comparing to an identical model on an IBM 3081 serial machine, we have shown that it is feasible to use the MPP for numerical models such as this one, and that the parallel architecture makes such calculations more efficient when the physical model includes modelling the same processes at many different points in space.

ACKNOWLEDGEMENT

Two of the authors, J. E. Devaney and A. R. Irving, were partially supported by NASA contract NAS5-28200.

REFERENCES

1. Booth, A.D., Numerical Methods, Academic Press, New York, 1957.

2. Camillo, P.J., R.J. Gurney and T.J. Schmugge, A Soil and Atmospheric Boundary Layer Model for Evapotranspiration and Soil Moisture Studies, Water Resour. Res., 1983, 19, pp. 371-380.

3. Clapp, R.B. and G.M. Hornberger, Empirical Equations for Some Soil Hydraulic Properties, Water Resour. Res., 1978, 14, pp. 601-604.

4. de Vries, D.A., Heat Transfer in Soils, Heat and Mass Transfer in the Biosphere, Scripa, Washington, D.C., 1975.

5. Eagleson, P.S., Dynamic Hydrology, McGraw-Hill, New York, 1970.

6. Freeze, R.A., Three-Dimensional, Transient, Saturated-Unsaturated Flow in a Groundwater Basin, Water Resour. Res., 1971, 7, pp. 929-941.

7. Lin, J.D., On the Force-Restore Method for Prediction of Ground Surface Temperature, J. Geophys. Res., 1980, 85, pp. 3251-3254.

8. Philip, J.R. and D.A. de Vries, Moisture Movement Porous Materials Under Temperature Gradients, EOS Trans. AGU, 1957, 38, pp. 222-228.

9. Teddington, A., Modern Computing Methods, Philosophical Library, NY, 1958.