A Personal Retrospection of Reservoir Simulation

Donald W. Peaceman

I discuss here some early history of reservoir simulation, from a personal point of view. I will give some history, some philosophy, and some numerical analysis. I will try to stress the interrelationship of the type of computing equipment that we had available at any given time, the kinds of calculations we were able to make, and the kinds of problems we were able to solve. So, if you will indulge me, we will take a look at what computing and numerical analysis were like 20 to 35 years ago, in the not-so-good old days.

I started in 1951 with Humble Oil and Refining Company, which at that time was a subsidiary of Standard Oil of New Jersey. Standard Oil of New Jersey became Exxon, and the Research Division of Humble Oil evolved into the present day Exxon Production Research Company (EPR).

When I came to work in 1951, we did not have any real computers available to us. Yet there was some reservoir modeling going on. I found some old pictures that illustrate how physical models were used.

Figure 1 shows the earliest one that I found. It was made in 1933, and it shows a sand-packed model that was used to study water coning. On top is an oil layer, with a water layer underneath it. You can see that wells were drilled just into the upper oil layer. The production of oil causes the pressure around the well to decrease, and that causes the water to cone up and be produced with the oil. Though oil fields have been produced since 1860, it was not until the 1930s that people in the oil industry started looking at reservoir mechanics in any kind of a scientific way. This was therefore one of the first attempts to understand why water starts to be produced with oil and why the produced water-oil ratio increases with time.

Sand-packed models were still in use some 25 years later—before computers took over. They became more sophisticated; for example, wedge shapes were used to take into account the radial geometry around a well.

The analogy between electric current flow and Darcy flow through sand had been recognized for quite a while, and electrolytic models were used in the late thirties and the forties to solve Laplace's equation for various geometries. The photograph on page 54 shows an example of how elaborate an electrolytic model could get. This was a model
of the East Texas Field, which is still one of the largest fields in the United States. The model was made of plastic and covered with an electrolyte solution. The plastic was contoured to represent the shape and permeability distribution in the field, with the depth of the solution above the plastic being proportional to the thickness times permeability that was actually measured in the field. The object was to measure the potential distribution in the field in order to predict the water influx from the aquifer surrounding the field.

But electrolytic models were steady-state models. To get a better representation of unsteady-state flow, the reservoir analyzer shown in Figure 2 was devised, involving a scaled electrical network of resistors and capacitors. Voltages represented pressure, current flow represented fluid flow, resistors corresponded to permeability times thickness, and the capacitors corresponded to porosity times thickness times compressibility. With this representation, unsteady-state compressible flow could be taken into account. The electrical network corresponded, of course, to a finite-difference equation solved continuously in time.

In addition to these physical analog models, some mathematical methods were available in 1951. In the thirties and forties, three authors made the most significant contributions to applying the methods of mathematical physics to reservoir engineering. Muskat, of Gulf, wrote a book in 1937 that summarized his work, and that book is
still very useful [1]. Hurst at Humble [2,3], and later Hurst and van Everdingen at Shell [4] also made significant contributions. Their methods were based primarily on infinite series solutions to Laplace’s equation and the heat conduction equation. Although these methods were elegant, they suffered from serious limitations in their application to real reservoir problems—they assumed uniform properties and ideal geometries and could only be used on linear differential equations. Also, these methods required the tedious evaluation of infinite series, which had to be computed by hand.

So that was the state of reservoir modeling when I came to work at Humble in 1951. We had nothing that you could call a computer. We did have access to some accounting machines that the accounting department would let us use, but only at night. Henry Rachford had come to work a year before me and was already playing with an accounting machine called the IBM 604. He, along with the managers of the Production Research Division of Humble, had the vision to see that digital computation was going to be the way to do reservoir modeling and that by using finite-difference methods to solve partial differential equations, we could overcome the limitations of the analytical methods. We wanted to be able to include nonuniform properties, arbitrary geometry, and nonlinearities in the differential equations.
A Personal Retrospection of Reservoir Simulation

But that vision was still faint. The first partial differential equation that we tried to solve was one-dimensional gas flow, and the first limitation that we were trying to overcome was that of nonlinearity. If one assumes a perfect gas, then the equation for linear one-dimensional gas flow is

$$\frac{\partial^2 p}{\partial x^2} = \frac{2\mu \partial p}{K \partial t}$$  \hspace{1cm} (1)

It looks a lot like the linear one-dimensional heat conduction equation, except that the second derivative term has $p^2$ in it instead of $p$, making that equation nonlinear. Because of that, there is no known analytical solution. The initial condition is uniform pressure; at one end is a fixed production rate $q$ giving the nonlinear boundary condition

$$q = \frac{KA}{\mu RT} p \frac{\partial p}{\partial x} \hspace{1cm} x = 0$$  \hspace{1cm} (2)

At the other end, the system is closed, so we have a no-flow boundary condition, with a zero derivative.

Of more practical interest was the radial problem, corresponding to the depletion of a circular gas reservoir with a well at the center:

$$\frac{1}{r} \frac{\partial}{\partial r} \left( r \frac{\partial p}{\partial r} \right) = \frac{2\mu \partial p}{K \partial t}$$  \hspace{1cm} (3)

The initial and boundary conditions are similar.

When I came to work, Henry Rachford and John Rice were already at work on this problem, trying to use the accounting machine, the IBM 604. (See Figure 3.) Let me try to describe this gadget to you. It was called a multiplying punch—the only input/output that it had was a card reader and card punch. It could only handle fixed-point decimal numbers, with no alphabetic information. The way the accountants used it, say for a payroll application, would be to have some numeric data already punched on each card, such as an employee’s identification number and salary. Each card would be read at the read station and would then travel to the punch station. On the way, the marvelous electronic multiply unit would calculate the withholding and social security taxes, subtract them, and then punch the taxes and take-home pay into the blank space on the card. After being punched, the card would then travel to the stacker. To see the results printed, an operator would have to carry the deck of cards to another machine to print the results.

The electronic multiply unit was really quite flexible, but programming it was done in a way that we would now consider quaint. There was a board with a lot of holes, and this board could be placed into a holder with terminals in the back. Programming was done by plug-
ing wires from one hole to another. Down one side of the board was a series of holes called program steps. On the other side were holes for various functions, such as reading from a card into electronic registers, adding or multiplying the contents of two registers together, and punching the contents of the registers onto the card. Remember that in those days, all the electronics were done with vacuum tubes.

This device was certainly unsuitable for scientific computations, yet it was all we had. John Rice and Henry Rachford programmed it to solve the one-dimensional gas flow problem. They were already familiar with the paper by O'Brien, Hyman, and Kaplan [5], published in 1951, which discussed the finite-difference solution to the linear heat conduction problem. That paper introduced us to the von Neumann stability analysis as well as to the use of implicit equations. From the stability analysis, they knew they could not use an explicit method for the radial problem, so they attempted to solve it by this implicit equation:

$$\frac{p_{i-1}^2 - 2p_i^2 + p_{i+1}^2}{\Delta x^2} = K \frac{p_i - p_i^{\text{old}}}{\Delta t}$$

(4)
A Personal Retrospection of Reservoir Simulation

We all were pretty naive in those days, so Rice and Rachford attempted to solve this equation sequentially from left to right using

$$p_{i+1}^2 = 2p_i^2 - p_{i-1}^2 + \frac{K\Delta x^2}{\Delta t}(p_i - p_i^{\text{old}})$$

(5)

This required guessing the slope at the well and seeing if the slope came out to be zero at the closed end. If not, they would adjust the initial slope and try again (a shooting method). We found out the hard way that this sequence of calculations from one end to the other is unstable and must blow up. In retrospect, this is obvious from an error analysis. But as I said, we were pretty naive, so that was one of our first experiences with an unstable calculation.

The fix, of course, is to solve for all of the pressures at all of the nodes simultaneously. We saw how to do that when we came upon an unpublished note by L. H. Thomas of IBM. In that note, he outlined what we now know as the tridiagonal algorithm. I believe our paper on gas flow was one of the first to present this algorithm in the published literature.

In order to use this algorithm, the nonlinear difference equation (4) had to be linearized. We did that by factoring the second-difference term into the form

$$\frac{(p_{i-1}^k + p_i^k)(p_{i-1}^{k+1} - p_{i+1}^{k+1}) - (p_i^k + p_{i+1}^k)(p_i^{k+1} - p_{i+1}^{k+1})}{\Delta x^2} = K\frac{p_{i+1}^{k+1} - p_i^{\text{old}}}{\Delta t}$$

(6)

and iterating on each time step. Although the iteration converged only linearly, it did converge very rapidly—usually five iterations were sufficient. At that time, we were not aware of the Newton-Raphson method, which would have given quadratic convergence.

By the time we had this new approach worked out, we were onto our next machine. It was our own machine now, and not one that we had to borrow. This was the IBM CPC, or Card-Programmed Calculator, shown in Figures 4 and 5. That is Henry Rachford in Figure 4. IBM did not really develop the CPC. Several computing groups at various aircraft companies modified and hooked together some existing IBM accounting machines, which IBM adopted and marketed as the CPC. It was a real kludge. The 418 (in the foreground of Figure 4) was an electromechanical accounting machine that could read cards, perform simple additions and subtractions, and print results at 150 lines per minute. It had the capacity to store eight 10-digit numbers. The box at the rear was the card punch. The 605 (at the right rear) was an electronic calculator, an extension of the 604 that I discussed before. All of these required the wiring of large boards. In addition, there were three boxes (not shown in Figure 4) we called ice boxes, which could each hold 16 ten-digit numbers in electromechanical counter wheels, like the odometer on a car. We could open the top
FIGURE 4.
IBM card-programmed calculator (CPC).

FIGURE 5.
Components of IBM CPC.
A Personal Retrospection of Reservoir Simulation

and actually read out the numbers while debugging. All of this was
decimal. It was designed to be fixed-point, but Henry and I wired the
machines to do floating-decimal arithmetic (two floating-point op-
erations per card), so that we achieved the magnificent rate of five
floating-point operations per second. We could store a total of 56
numbers. This was not a stored-program machine—instead the pro-
gram had to be punched onto cards—so, in effect, we had an unlimited
amount of storage available for programs, but only 56 words available
for temporary data.

This, then, was the device on which we solved the gas flow
problem for both the linear and radial cases [6]. Iteration was carried
out by reading the same deck of program cards over and over, and the
iteration was monitored by looking at the printed output. When the
iteration converged, we switched to a new deck to start a new time
step.

Although we were successful in solving the one-dimensional non-
linear problem, we were aware that further progress toward solving
realistic field problems would require going to higher dimensions—at
least to two dimensions. At the very least, we knew that we needed
to be able to solve the finite-difference analogs of Laplace’s equation
and the heat conduction equation (or better yet, the variable-coefficient
versions of those equations) in an arbitrary geometry. To do this, we
had to be able to solve a large system of simultaneous linear equations.
Direct solution of these equations by Gaussian elimination was out of
the question on the machines that were then available.

We had access in those days to several eminent consultants. One
of them was John von Neumann, and he visited us a couple of times.
He was very interested in the work we were doing, but when we
asked him how to go about solving two-dimensional problems of this
sort, he had no more to offer than the so-called extrapolated Liebman
method, now known as successive overrelaxation, or SOR. We already
knew about that method.

A breakthrough came, not while we were thinking about solving
a problem in x-y coordinates, but while we were working on a flow
problem in cylindrical coordinates:

\[
\frac{1}{r} \frac{\partial}{\partial t} \left( r \frac{\partial p}{\partial r} \right) + \frac{\partial^2 p}{\partial y^2} = \frac{\partial p}{\partial t} \tag{7}
\]

If we let \( x = \ln r \), then the differential equation takes the somewhat
simpler form

\[
e^{-2x} \frac{\partial^2 p}{\partial x^2} + \frac{\partial^2 p}{\partial y^2} = \frac{\partial p}{\partial t} \tag{8}
\]

Solving this equation implicitly would have had the same difficulty as
solving the heat conduction equation implicitly. But the inherent dif-
Problems

difference between the radial and vertical directions suggested another approach. Suppose we make the difference equation implicit in just one direction, say the radial direction, and explicit in the other, vertical, direction. The half-implicit, half-explicit difference analog would then be

$$e^{-2\pi i} \frac{p_{i,j}^{n+1} - 2p_{i,j}^n + p_{i,j+1}^n}{\Delta x^2} + \frac{p_{i,j-1}^n - 2p_{i,j}^n + p_{i,j+1}^n}{\Delta y^2} = \frac{p_{i,j}^{n+1} - p_{i,j}^n}{\Delta t}$$  \hspace{1cm} (9)

The advantage, of course, is that on each line we have just a tridiagonal system of equations, which is easy to solve. The von Neumann stability analysis is simple to do—we substitute this Fourier representation

$$p_{i,j}^n = \gamma^n e^{i\alpha x} e^{i\beta y}$$  \hspace{1cm} (10)

into the difference equation and examine the growth of $\gamma$. The amplification factor for $\gamma$ is given by the ratio

$$\frac{\gamma^{n+1}}{\gamma^n} = \frac{1 - 4(\Delta t/\Delta x^2) \sin^2(\beta \Delta y/2)}{1 + e^{-2\pi i(\Delta t/\Delta x^2)} \sin^2(\alpha \Delta x/2)}$$ \hspace{1cm} (11)

For the difference equation to be stable, the magnitude of this ratio has to be less than one for all $\alpha$ and $\beta$. Unless $\Delta t$ is very small, that will not be true, so we have here a difference equation that is not much better than a fully explicit one.

Suppose we do the opposite by making the equation explicit in the radial direction and implicit in the vertical direction.

$$e^{-2\pi i} \frac{p_{i-1,j}^n - 2p_{i,j}^n + p_{i+1,j}^n}{\Delta x^2} + \frac{p_{i,j-1}^n - 2p_{i,j}^n + p_{i,j+1}^n}{\Delta y^2} = \frac{p_{i,j}^{n+1} - p_{i,j}^n}{\Delta t}$$ \hspace{1cm} (12)

In that case, we would get the following ratio for the amplification factor

$$\frac{\gamma^{n+1}}{\gamma^n} = \frac{1 - e^{-2\pi i(\Delta t/\Delta x^2)} \sin^2(\alpha \Delta x/2)}{1 + 4(\Delta t/\Delta y^2) \sin^2(\beta \Delta y/2)}$$ \hspace{1cm} (13)

and again, unless $\Delta t$ is sufficiently small, this ratio will be bigger than one in magnitude for some $\alpha$ and $\beta$.

Somehow, and we do not remember exactly how, though it seemed natural enough, Henry Rachford and I came up with the idea of doing it one way for one time step, and then the other way for the next time step—a two-step procedure. We would then repeat the two-step procedure over and over.

It was not immediately obvious that this would be stable. We analyzed it independently overnight, however, and came to the same conclusion, that it is stable. It was necessary to recognize that we wanted to look at the amplification factor, not for either step alone,
but for the entire process of going from step \( n \) to step \( n + 2 \). Now the second ratio (13) is really \( \gamma^{n+2} / \gamma^{n+1} \), and we can multiply the two ratios (11) and (13) together and rearrange them to get

\[
\gamma^{n+2} = \frac{1 - e^{-2x_i (\Delta t / \Delta x^2)} \sin^2(\alpha \Delta x / 2)}{1 + e^{-2x_i (\Delta t / \Delta x^2)} \sin^2(\alpha \Delta x / 2)} \frac{1 - 4(\Delta t / \Delta y^2) \sin^2(\beta \Delta y / 2)}{1 + 4(\Delta t / \Delta y^2) \sin^2(\beta \Delta y / 2)}
\]

(14)

Now a remarkable thing happens. The first ratio is always less than one in magnitude, no matter what the values of \( \Delta t \), \( \Delta x \), or \( \alpha \) are. Similarly, the second ratio is always less than one in magnitude, no matter what the values of \( \Delta t \), \( \Delta y \), or \( \beta \) are. Hence, the product must be less than one in magnitude, and the two-step procedure must be stable. So that was how alternating direction was born.

Henry and I remember well the dates of this discovery, 30 and 31 December 1953. The reason we remember it so well is that we celebrated New Year’s Eve at our house, along with Jim Douglas and his wife. Naturally we were very excited and could hardly talk about anything else. This shop talk was very distressing to the hostess, my wife. I think she finally forgave us a few years later.

There were several implications to the discovery that were immediately apparent. First, of course, was the fact that the asymmetry of the cylindrical problem had nothing to do with the success of the method, even though that was what triggered the idea. In particular, of course, it could be applied directly to the heat conduction problem in ordinary \( x-y \) coordinates.

The second implication, of even greater significance, was the fact that the alternating-direction method can also be used to solve a steady-state problem. The solution to Laplace’s equation is, after all, the solution to the heat conduction equation at infinite time. We can imagine that if we take enough time steps, we will get the solution to Laplace’s equation. We can think of accelerating the process by taking some short time steps and then some longer ones, and then if we are not close enough to the solution, repeating the sequence of short and long time steps. What that amounts to, of course, is nothing more than using alternating direction as an iterative method, with \( \Delta t \) serving as an iteration parameter. Well, Jim Douglas ran with that idea. He carried out an analysis that permits one to calculate an almost optimum sequence of iteration parameters. He also demonstrated convergence of the ADI method to the solution of the heat conduction problem. His results were published in a companion paper [7] to the paper that Rachford and I published in the SIAM Journal early in 1955 [8].

The first tests of ADI were on the Card-Programmed Calculator. For the SIAM paper [8], we solved both an unsteady-state and a steady-state problem on a 14 by 14 square grid. Why 14 by 14? Well, we had 56 words of data storage. As we calculated for each line, we needed to keep four numbers internally for each point. Thus, the
longest line we could handle was 14. Most of the temporary data storage was punched out onto cards, and the direction was alternated using a card sorter. Because the data that were punched onto the cards had to be read back in reverse order, we punched the cards backwards and upside down, then turned them over, in order to facilitate the sorting process.

Jim Douglas and I wrote a second paper on ADI in which we solved some steady-state problems on geometries other than a square [9]. These were also done on the CPC. The first one, shown in Figure 6, was for heat flow around a corner, with the temperatures zero and one at two boundaries, and with no-flow boundaries elsewhere. Also shown is the grid numbering scheme that we used, in which the numbers correspond to the register numbers on the CPC.

Figure 7 shows a problem involving radiation from a square pipe. The inside of the pipe is at temperature $T_1$; the outside of the pipe has a nonlinear radiation boundary condition. We took advantage of the symmetry to solve the system in one-eighth of the cross-section. The computing grid is also shown.

The third problem, in Figure 8, was more related to reservoir engineering. We assumed an elliptical reservoir with no flow at the external boundary, with input wells at points A and B and output wells at points C and D. Again, we took advantage of symmetry and solved only one-quarter of the system. The computing grid for that problem is also shown. In all of these cases, the longest line was eight points long, which turned out to be very convenient on the CPC.

In 1955 we acquired a Bendix G-15, shown in Figure 9. This also had vacuum tube electronics, but its storage was almost completely on a magnetic drum. It came with fixed-point binary arithmetic, which was of limited scientific use, so I spent several months programming a floating-point interpreter for it. With that, we had the fantastic capability of doing 10 floating-point operations per second. It had 864 words

---

FIGURE 6. Heat flow around corner.
of memory available, for both data and program. Its input/output was paper tape, typewriter, and magnetic tape, none of which were particularly reliable by today’s standards.

In addition, within the next few years we started using IBM’s first widely used scientific computer, the 704, shown in Figure 10. It was a binary machine, with built-in floating-point hardware. Its electronics were based on thousands of vacuum tubes; its central memory was magnetic core; its secondary storage was magnetic tape. We never
FIGURE 8.
Flow in elliptical reservoir.

FIGURE 9.
Bendix G-15 drum computer.
acquired a 704 of our own. The first one that we used was at the IBM Service Center in New York City, starting about 1956; after several years we started using 704s at various aircraft companies throughout the country that were selling excess time.

As you can see from Table 1, the 704 marked a major advance in our computing capability, and with it we were able to solve our first

---

TABLE 1
Humble/EPR Computing Equipment

<table>
<thead>
<tr>
<th>Computer</th>
<th>Date Acquired</th>
<th>Storage (words)</th>
<th>Speed</th>
</tr>
</thead>
<tbody>
<tr>
<td>IBM 604</td>
<td>Before 1950</td>
<td>8(?)+ Cards</td>
<td>5 FLOPS</td>
</tr>
<tr>
<td>IBM CPC</td>
<td>1952</td>
<td>56 + Cards</td>
<td>10 FLOPS</td>
</tr>
<tr>
<td>BENDIX G-15</td>
<td>1955</td>
<td>864</td>
<td></td>
</tr>
<tr>
<td>IBM 704</td>
<td>1956</td>
<td>8,192</td>
<td>10,000 FLOPS</td>
</tr>
<tr>
<td>(away)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BENDIX G-20</td>
<td>1961</td>
<td>8,192</td>
<td>20,000 FLOPS</td>
</tr>
<tr>
<td>IBM 704/7044</td>
<td>1962</td>
<td>16,384</td>
<td>40,000 FLOPS</td>
</tr>
<tr>
<td>IBM 360/65</td>
<td>1967</td>
<td>256K</td>
<td>400,000 FLOPS</td>
</tr>
<tr>
<td>IBM 370/165</td>
<td>1971</td>
<td>1 M</td>
<td>1 MFLOPS</td>
</tr>
<tr>
<td>IBM 370/168</td>
<td>1975</td>
<td>2 M</td>
<td>1.2 MFLOPS</td>
</tr>
<tr>
<td>AMDAHL V8</td>
<td>1978</td>
<td>4 M</td>
<td>2 MFLOPS</td>
</tr>
<tr>
<td>IBM 3033</td>
<td>1979</td>
<td>4 M</td>
<td>2 MFLOPS</td>
</tr>
<tr>
<td>IBM 3081</td>
<td>1982</td>
<td>4 M</td>
<td>6 MFLOPS</td>
</tr>
<tr>
<td>CRAY 1S</td>
<td>1982</td>
<td>4 M</td>
<td>20–160 MFLOPS</td>
</tr>
</tbody>
</table>
Problems

real reservoir problem. That involved solving this variable coefficient version of the steady-state Laplace's equation:

\[
\frac{\partial}{\partial x} \left[ k(x, y) \frac{\partial p}{\partial x} \right] + \frac{\partial}{\partial y} \left[ k(x, y) \frac{\partial p}{\partial y} \right] = q(x, y)
\]  

(15)

\( k \) was the value of permeability times thickness, which was known as a function of \( x \) and \( y \). (See Figure 11.) As we still do today, we introduced the shape of the reservoir merely by setting \( k \) to zero for all grid points outside the reservoir. \( q \) is the input/output term. It was zero for most grid points and equal to the flow rate in or out wherever there was a well.

With internal storage of 8000 words, some of which was occupied by program, we could solve this problem core-contained on a 50 by 30 grid, or 1500 points. Alternating direction was used to iterate to the solution, and it took about five or ten minutes to converge. We could monitor the progress of the iteration on the slow on-line printer and modify the iteration parameter sequence by hand interactively, using switches on the control panel. Detailed output was taken on magnetic tape, which we took to an off-line printer for display of our results.

Fortran was not yet available. In fact, we saw the creators of Fortran using the same 704 in New York while we were there. Our program was written in assembly language, but that did not bother us because we were already used to dealing with machines that were much less user-friendly.

After the successful solution of this first field problem, the reservoir engineers wanted us to tackle a problem that required more defi-

FIGURE 11.
First field problem.
A Personal Retrospection of Reservoir Simulation

nition, one with 6000 grid points. We could not do this using only the
8000 words of central core memory, so we had to rewrite the program
to use magnetic tape for secondary storage. Alternating direction using
magnetic tape is very interesting to watch; you can actually visualize
the calculations taking place. For each horizontal line, the tape moves
forward slowly one block, as one line of the x-sweep and the begin-
ning of the y-sweep takes place. This is repeated for each horizontal
line. When all the lines are processed, the tape then quickly moves
backward as the back solution of the y-sweep is carried out.

This illustrates one aspect of the computing environment that has
changed considerably. In those days we loaded our own jobs and
watched the calculations take place, with magnetic tapes and the on-
line printer making the computations visible. Then, about 20 years
ago, computing became much more invisible. Professional operators
took over the actual operation of the machines. Secondary storage
moved from tape to disk, which is not very visible. So, although we
have gained in reliability, I think we have lost a feel for the computa-
tion process.

The next milestone of my career was the 1959 paper by Douglas,
Rachford, and myself, on two-dimensional, two-phase immiscible dis-
placement [10]. This involved the solution, in two dimensions, of these
two simultaneous nonlinear differential equations:

\[
\nabla \cdot \left[ \frac{K_{r_w}}{\mu_o} \nabla p_o \right] = \phi_o \frac{\partial S_o}{\partial t} \quad (16a)
\]
\[
\nabla \cdot \left[ \frac{K_{r_w}}{\mu_w} \nabla p_w \right] = \phi_w \frac{\partial S_w}{\partial t} \quad (16b)
\]

Without going into too much detail, these equations model the dis-
placement, in porous media, of oil by water. \( S \) is saturation, the frac-
tion of the void space occupied by each fluid. \( S_o \) plus \( S_w \) add up to
one. \( K \) is the rock permeability, a function of \( x \) and \( y \). When oil and
water flow together, they get in each other's way. To account for this,
relative permeabilities, \( k_{r_o} \) and \( k_{r_w} \) are used. They are assumed to be
known functions of saturation. In addition, the difference between
the oil pressure and water pressure is the capillary pressure, also a
function of saturation:

\[
p_o - p_w = P_c(S)
\]

These equations, (16a) and (16b), are coupled very nonlinearly. Again
without going into detail, in that paper we presented two finite-
difference methods for solving these equations that form the basis for
reservoir simulation that is still in use today. These finite-difference
methods have been extended to three dimensions and also to three
phases: oil, gas, and water. Furthermore, the methods have been
generalized to include fluid compressibility.
Problems

In looking back over this paper [10], however, I can see how naive we were. In particular, we did two things that we now recognize as unsatisfactory. First, we used midpoint weighting of relative permeability. In the difference equations, one needs values for relative permeability for each interval between grid points. It seemed reasonable to use an average value for this,

\[
(k_r)^{1/2} = \frac{((k_r)^{-1} + (k_r)^{1})}{2}
\]  

(17)

and that is what we did. The other thing we did was to use old values for relative permeabilities.

We might ask two questions—what was naive about it, and why were we so naive? Let me answer the second question first. On the face of it, (16a) and (16b) look parabolic in nature—in other words, like extensions of the heat conduction equation. For the heat conduction equation, such a treatment of the nonlinear coefficients is usually satisfactory.

The fact is, for an incompressible system, these equations are not parabolic in nature; rather, they are elliptic and hyperbolic in nature. To see this, if we add (16a) and (16b), the time derivatives of saturation drop out, and we get this pressure equation:

\[
\nabla \cdot [(KK_{ro}/\mu_o) + (KK_{rw}/\mu_o)] \nabla (p_o + p_w) \\
+ \nabla \cdot [(KK_{ro}/\mu_o) - (KK_{ro}/\mu_o)] \nabla (p_o - p_w) = 0
\]  

(18)

The first term involves the sum of the two phase pressures, and the second involves their difference. But that difference is capillary pressure, which is usually very small. So the first term dominates, and (18) looks just like the variable coefficient Laplace equation, which is elliptic in nature. We get another equation by subtracting (16a) and (16b) and also by making these substitutions:

\[
\tilde{v}_t = -(KK_{ro}/\mu_o) \nabla p_o - (KK_{ro}/\mu_o) \nabla p_w
\]  

(19)

\[
f(S) = (KK_{rw}/\mu_o)/(KK_{ro}/\mu_o) + (KK_{ro}/\mu_o)
\]  

(20)

where \( \tilde{v}_t \) is a total velocity, and \( f(S) \) is a fractional flow function. This gives us

\[
-\nabla \cdot \left[\frac{KK_{ro} \frac{dP_c}{dS_w}}{\mu_o} \nabla S_w \right] - \tilde{v}_t \frac{df}{dS_w} \cdot \nabla S_w = \phi \frac{\partial S_w}{\partial t}
\]  

(21)

On the face of it, (21) looks parabolic, but the first term involves \( P_c \), the capillary pressure, and it usually is small. The second term is the convection term, with velocity times a first order derivative of saturation, and it dominates. So (21) is really almost first-order hyperbolic in nature.
A Personal Retrospection of Reservoir Simulation

It was not until much later that we realized that the appropriate differential equation to analyze for stability is

\[ -v_1 \frac{\partial f(S)}{\partial x} = \phi \frac{\partial S}{\partial t} \]  

(22)

where we have assumed one dimension and zero capillary pressure. For midpoint weighting of relative permeability, the difference equation simplifies to

\[ v_i \left( f_i^{n+1} - f_i^n \right) \frac{1}{2\Delta x} = \phi \frac{S_i^{n+1} - S_i^n}{\Delta t} \]  

(23)

A von Neumann stability analysis shows that (23) is unstable for any size time step. Indeed, people who started using our method discovered empirically that they were getting oscillatory solutions that could be avoided by using upstream weighting for relative permeability. In upstream weighting, the relative permeability at the upstream grid point is used for each interval between grid points. Within a few years, it became standard practice in the industry to use upstream weighting. In that case, the appropriate difference equation to look at is

\[ v_i \left( f_i^{n+1} - f_i^n \right) \frac{1}{\Delta x} = \phi \frac{S_i^{n+1} - S_i^n}{\Delta t} \]  

(24)

A stability analysis shows that (24) is stable, provided the time step is small enough, according to the criterion

\[ v_i \frac{\partial f}{\partial S} \frac{\Delta t}{\Delta x} \leq 1 \]  

(25)

With the use of upstream weighting, the methods proposed in our paper and variations on them became quite popular for the solution of two- and three-dimensional problems. General-purpose reservoir simulators were developed by a number of companies over the next 10 years. But there was one class of problems that these simulators could not handle. These are coning problems, such as the one shown in Figure 12. Because of the radial geometry and the converging flow, the velocity is very high near the well. For any reasonable time step, inequality (25) is violated, and the calculated water-oil ratio produced into the well oscillates wildly.

About 1970, three papers were published almost simultaneously [11,12,13] that proposed essentially the same solution, which we call the semi-implicit approach. Instead of using the old value of relative permeability, an approximation for the new one is used:

\[ k_r^{n+1} \approx k_r^n + \frac{dk_r}{dS}(S_i^{n+1} - S_i^n) \]  

(26)
When this approximation is introduced into the saturation equation, in effect it makes the saturation equation implicit. The equation to analyze now looks like this:

$$\frac{f_{i+1}^n - f_i^{n+1}}{\Delta x} = \frac{S_i^{n+1} - S_i^n}{\Delta t}$$

(27)

It is stable for any size time step.

It might be useful to look now at how our computing equipment changed over the past 25 years. Refer again to Table 1. We finally got our own large-scale computer in 1961, a Bendix G-20 with 8192 words of core storage. It was twice the speed of the 704, with 20,000 floating-point operations per second. Bendix never came through with a Fortran compiler, so we continued to do all our programming in assembly language. We programmed a general-purpose, two-dimensional, two-phase reservoir simulator, with magnetic tape for secondary storage. Figure 13 shows Henry and me in front of the G-20 tape units, looking at some output. Bendix finally got out of the computer business, but before they did, we had sent back their G-20 and obtained an IBM 7040, a transistorized version of the 704, again with some increase in speed. Then, in the late sixties, we started with the new IBM 360/370 series, with disk storage and much faster arithmetic speeds. EPR now has several IBM machines, along with the IBM-compatible Amdahl, and the speed has been increasing significantly into the megaflops range, along with increases in the amount of central memory. EPR now also has a Cray 1-S, with four million words of storage. It is a vector computer with a theoretical maximum speed of 160 megaflops,
A Personal Retrospection of Reservoir Simulation

although, like most users, we would get a sustained rate in the range of 20 to 40 megaflops. Later versions of the Cray have even higher speeds and larger memories. Now that I am retired from Exxon, however, I am reduced to having my own personal computer at home. It runs at about 30,000 floating-point operations per second, which puts it where the main-frame computers were 25 years ago. But, undoubtedly, we are seeing similar increases in speed and memory for personal computers.

One point I would like to make is that in designing a general purpose reservoir simulator, it was never safe to assume that the central random access memory would be big enough. Even though we have seen very large increases in memory size, the computation speed has also been going up drastically, and the unit cost of computation has been going down. As a result, the reservoir engineers who use our simulators keep wanting to make their models bigger and bigger, with more and more definition. Or, to put it another way, they tend

FIGURE 13.
Bendix G-20 tape units.
Problems
to run out of central memory before they run out of money. (At least, they did before the price of oil went down.) Consequently, we have found it necessary to program our simulators to use secondary storage to supplement the central memory. We used to use magnetic tape for that; now, of course, we use disk storage, which is faster and much more reliable.

But secondary memory is always much slower than central memory, so it has been necessary to learn how to use it in the most efficient way possible. This requires paying attention to the characteristics of the hardware. So you can see, we started out 30 years ago in a hardware-oriented way, and we have never gotten completely away from it. Even with the most modern machines, the effective use of the equipment still requires paying attention to the hardware. And with the advent of vector computers and other kinds of parallel computers, this has become even more true.

What has happened to the alternating-direction method? Our first paper on two-dimensional immiscible displacement [10] used alternating direction to solve for the two phase pressures on each time step. Although this worked fairly well, as we got into more difficult field problems with highly variable permeability distributions, it became more and more difficult to find a sequence of parameters that would make it converge quickly. And frequently it would diverge. In 1968, Herb Stone of Exxon published a new method [14] called SIP. This method also requires a sequence of parameters, but it is much more robust, and it is easier to make it work. So, at Exxon, SIP pretty much superseded ADI, while other companies tended to go more for successive line overrelaxation. At the present time, the trend is toward preconditioned conjugate gradient methods. However, the search for good iteration methods is far from over, and there is still a lot of research going on in the area of iterative solution of equations.

In earlier times, direct elimination methods were out of the question, but that is no longer true. For two-dimensional problems, they are now quite competitive with iterative methods. This also is an active area of research, with people looking particularly at sparse matrix methods as well as studying how best to make use of vector computers and other types of parallel machines.

I have just touched on the simplest of the reservoir flow equations that involve the flow of oil, gas, and water. These we now solve routinely, even in three dimensions, with thousands or sometimes tens of thousands of grid points. But now the industry is looking more and more at the simulation of enhanced recovery processes, which involve the injection of carbon dioxide, high-pressure nitrogen, steam, chemicals, or polymers. The calculations required to simulate these processes are much more demanding, so the needs and the opportunities for research in these areas are tremendous.
A Personal Retrospection of Reservoir Simulation

I will say a little bit about numerical methods in general, as applied to reservoir simulation. We started with finite-difference methods 25 years ago, and they still continue to be used throughout the industry today. There are a number of problems that arise from the use of finite-difference methods. The chief one is probably numerical dispersion, which smears the solution for saturation and concentration. Finite-element and other variational methods for solving reservoir problems are being studied by a lot of people, but the results they have obtained have not been impressive enough to cause the industry to stop using finite-difference methods. So the finite-element people continue doing research to try to improve on their methods, though it seems clear that the finite-difference methods will continue for quite a while to be the mainstay of reservoir simulation. As long as that is the case, there needs to be more research on finite-difference methods, to understand them better and to improve on them.

I have tried to do my own little part in the study of finite-difference methods used in reservoir simulation. Three papers illustrate what I have tried to do. The first [15] is a detailed study of the stability of difference equations that use semi-implicit relative permeability. The last two [16,17] discuss how to relate the finite-difference solution for the pressure of a grid block containing a well to the actual pressure at the well itself.

In conclusion, I hope I have conveyed some of the excitement of the early days of reservoir simulation, when we had to fight against the limitations of primitive computing equipment as well as overcome our naiviness about numerical methods. There are still plenty of challenges left today, and I think they can be just as exciting.

References


Problems


Operation of the ENIAC.