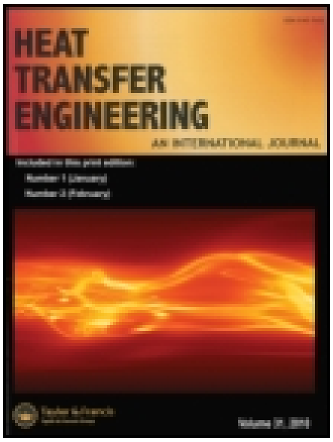

Heat Transfer Engineering
Publication details, including instructions for authors and subscription information:
http://www.tandfonline.com/loi/uhte20

Spreadsheet Solution of a Two-Dimensional Stefan Problem Using an Approximate Method

A. Kharab
[a] Mathematics Department, King Fahd University of Petroleum & Minerals, Dhahran, Saudi Arabia
Published online: 29 Oct 2010.

Heat Transfer Engineering, 21:65–71, 2000

0145–7632/00 $12.00 + .00

Spreadsheet Solution of a Two-Dimensional Stefan Problem Using an Approximate Method

A. KHARAB

Mathematics Department, King Fahd University of Petroleum & Minerals, Dhahran, Saudi Arabia

This article describes the use of a spreadsheet program for the numerical solution of a two-dimensional Stefan problem. A complete Lotus 1-2-3 macro is presented and numerical results are given and compared with other known results.

A large number of technically important problems involve solution of the governing equation subject to the existence of a moving boundary. Such problems are referred as moving or free boundary-value problems. Particularly important problems of this type are Stefan problems which arise as models of heat transfer processes involving change of phase, i.e., melting or solidification (see [1]). The interface on which melting or solidification occurs is called the free boundary. In this article, an approximate analytical method is presented to find the numerical solution of a two-dimensional Stefan problem using a Lotus 1-2-3 spreadsheet program.

Recently, spreadsheet programs have been widely used to find the numerical solution of many science and engineering problems. Spreadsheets programs can provide rapid and simple numerical solution for initial- and boundary-value problems in one or multiple dimensions. A method of using spreadsheets to perform the numerical solution of a two-dimensional heat conduction problem is presented by Kharab [2]. In [3], Kerk used it to obtain the numerical solution for transient fluid flow in a porous medium with a moving drainage boundary. Spreadsheets have been used in engineering for analyzing logical networks [4] and solving differential equations [5]. Unlike the approach adopted in [3–5], where the solutions consisted of a number of steps which the user had to execute, our approach proposes a solution which is executed in a single Lotus 1-2-3 macro program. Input data and initial temperature distributions can be entered into the spreadsheet by just moving the cursor to the appropriate cell and then typing the entry. This would help give engineers a better understanding of the numerical algorithm and experience the method more easily.

The author wishes to acknowledge the support of King Fahd University of Petroleum & Minerals, Dhahran, Saudi Arabia.

Address correspondence to Dr. Abdelwahab Kharab, Mathematics Department, King Fahd University of Petroleum & Minerals, Dhahran 31261, Saudi Arabia. E-mail: awkharab@kfupm.edu.sa

THE TWO-DIMENSIONAL STEFAN PROBLEM

The one-phase Stefan problem of conductive heat transfer arises as water and ice occupying a given domain D with prescribed temperature or flux on the boundary of the domain. The domain D contains two regions, one occupied by water and the other by ice at 0°C separated by a moving boundary $s(x, t)$. The one-phase Stefan problem in two space variables is given by

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \quad \text{in } \Omega(s,t), \quad t > 0 \tag{1}$$

where $\Omega(s,t) = \{(x,y) \in R^2 \mid 0 \leq x \leq 1, 0 \leq y \leq s(x,t)\}$, and subject to the boundary conditions

$$\frac{\partial u}{\partial x} = 0 \quad x = 0 \quad x = 1 \tag{2}$$

$$u = 1 \quad y = 0 \tag{3}$$

$$u = 0 \quad y = s(x,t) \tag{4}$$

$$\frac{\partial s}{\partial t} = -\frac{\partial u}{\partial y} + \left(\frac{\partial u}{\partial x}\right)\left(\frac{\partial s}{\partial x}\right) \quad y = s(x,t) \tag{5}$$

and the initial temperature distribution

$$u(x,y,0) = 1 - \frac{y}{g(x,0)} \tag{6}$$

Here $u(x,y,t)$ denotes the temperature of the water at a point (x,y) in the region $\Omega(s,t)$ at any time t. The region $\Omega'(s,t) \equiv \{(x,y) \in R^2 \mid 0 \leq x \leq 1, y \geq s(x,t)\}$ is assumed to be occupied by ice at 0°C. $y = s(x,t)$ defines the moving boundary and $g(x)$ is a given function such that $s(x,0) = g(x)$.

SOLUTION METHOD

The approximate method used to solve to this problem was discussed in detail in [6]. So, following [6], we assume that the temperature profile for constant x is given by

$$u(x,y,t) = A\left(\frac{y}{s}\right)^2 + B\left(\frac{y}{s}\right) + 1 \tag{7}$$

where A and B are parameters, usually functions of x and t, to be determined and s is the moving boundary. Note that Eq. (7) satisfies condition (3). Further, by differentiating (7) with respect to y and setting $y = 0$ we obtain the relation

$$s\,\frac{\partial u(x,0,t)}{\partial y} = B \tag{8}$$

By applying condition (4) in (7), we get the relation $A = -(1+B)$. Thus (7) can be written as

$$u(x,y,t) = -(1+B)\left(\frac{y}{s}\right)^2 + B\left(\frac{y}{s}\right) + 1 \tag{9}$$

Now u is in terms of s and B only, which are still to be found. We proceed by taking the zeroth and first moments of Eq. (1) with respect to y. There are respectively defined by

$$\int_0^s \frac{\partial u}{\partial t}\,dy = \int_0^s \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right)dy \tag{10}$$

and

$$\int_0^s y\,\frac{\partial u}{\partial t}\,dy = \int_0^s y\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right)dy \tag{11}$$

By using (5) and (8) and making some simplifications, (10) takes the form

$$s(10+B)s_t + s^2 B_t = s(4+B)s_{xx} - 6B + s^2 B_{xx} + 2ss_x B_x \tag{12}$$

Similarly, using (5) and (8) in (11) we obtain

$$s(9+B)s_t + \frac{1}{2}s^2 B_t = (3+B)(ss_{xx} + s_x^2) + \frac{1}{2}s^2 B_{xx} + 2ss_x B_x + 6 \tag{13}$$

By solving for s_t and B_t the system of equations defined by (12) and (13), we finally obtain

$$\frac{\partial s}{\partial t} = \frac{2(3+B)s_x^2 + s(2+B)s_{xx} + 6(2+B) + 2ss_x B_x}{s(8+B)} \tag{14}$$

and

$$\frac{\partial B}{\partial t} = \frac{-2(30+13B+B^2)s_x^2 + 12ss_{xx} - 12(B^2+10B+10) - 4ss_x B_x}{s^2(8+B)} + \frac{\partial^2 B}{\partial x^2} \tag{15}$$

where

$$\frac{\partial s}{\partial t} = s_t \qquad \frac{\partial s}{\partial x} = s_x \qquad \frac{\partial^2 s}{\partial x^2} = s_{xx},$$

$$\frac{\partial B}{\partial t} = B_t \qquad \frac{\partial B}{\partial x} = B_x$$

The solution of the simultaneous differential equations (14) and (15) gives the position of the moving boundary $s(x, t)$.

NUMERICAL METHOD AND SPREADSHEET DESCRIPTION

We now develop a numerical scheme for Eqs. (14) and (15) as follows. Let $\delta x = 1/N$ for a positive integer N, and $\delta t = TMAX/M$ for a positive integer M and an arbitrary upper time boundary $TMAX$. Values of s are calculated along a $x = i\delta x$, $i = 0, 1, \ldots, N$ mesh for successive time steps δt.

By approximating the space derivative by a central difference and the time derivative by a forward difference, Eqs. (14) and (15) take the forms

$$\begin{aligned} s_i^{k+1} = s_i^k + \frac{\lambda}{4s_i^k(8+B_i)}\Big[&2(3+B_i^k)(s_{i+1}^k - s_{i-1}^k)^2 \\ &+ 4s_i^k(2+B_i^k)(s_{i+1}^k - 2s_i^k + s_{i-1}^k) \\ &+ 24(\delta x)^2(2+B_i^k) + 2s_i^k(s_{i+1}^k - s_{i-1}^k) \\ &\times (B_{i+1}^k - B_{i-1}^k)\Big] \end{aligned} \tag{16}$$

and

$$\begin{aligned} B_i^{k+1} = B_i^k + \frac{\lambda}{4(s_i^k)^2(8+B_i^k)}\Big\{&-2\left[30 + 13B_i^k + (B_i^k)^2\right] \\ &\times (s_{i+1}^k - s_{i-1}^k)^2 + 48s_i^k(s_{i+1}^k - 2s_i^k + s_{i-1}^k) \\ &- 48(\delta x)^2\left[(B_i^k)^2 + 10B_i^k + 10\right] \\ &- 4s_i^k(s_{i+1}^k - s_{i-1}^k)(B_{i+1}^k - B_{i-1}^k) \\ &+ \lambda(B_{i+1}^k - 2B_i^k + B_{i-1}^k) \end{aligned} \tag{17}$$

for $i = 1, 2, \ldots, N-1$ and $k = 1, 2, \ldots$.

$$\begin{aligned} s_0^{k+1} = s_0^k + \frac{2\lambda}{s_0^k(8+B_0)}\Big[&s_0^k(2+B_0^k)(s_1^k - s_0^k) \\ &+ 3(\delta x)^2(2+B_0^k)\Big] \end{aligned} \tag{18}$$

$$\begin{aligned} s_N^{k+1} = s_N^k + \frac{2\lambda}{s_N^k(8+B_N)}\Big[&s_N^k(2+B_N^k)(s_{N-1}^k - s_N^k) \\ &+ 3(\delta x)^2(2+B_N^k)\Big] \end{aligned} \tag{19}$$

$$\begin{aligned} B_0^{k+1} = B_0^k + \frac{12\lambda}{(s_0^k)^2(8+B_0^k)}\Big\{&2s_0^k(s_1^k - s_0^k) - (\delta x)^2 \\ &\times\left[(B_0^k)^2 + 10B_0^k + 10\right]\Big\} + 2\lambda(B_1^k - B_0^k) \end{aligned} \tag{20}$$

$$\begin{aligned} B_N^{k+1} = B_N^k + \frac{12\lambda}{(s_N^k)^2(8+B_N^k)}\Big\{&2s_N^k(s_{N-1}^k - s_N^k) \\ &- (\delta x)^2\left[(B_N^k)^2 + 10B_N^k + 10\right]\Big\} \\ &+ 2\lambda(B_{N-1}^k - B_N^k) \end{aligned} \tag{21}$$

for $k = 1, 2, \ldots$. Here $\lambda = \delta t/(\delta x)^2$, $s_i^k = (i\delta x, k\delta t)$, and $B_i^k = (i\delta x, k\delta t)$.

For the initial conditions we have, from (8),

$$B(x, 0) = s(x, 0)\frac{\partial u(x, 0, 0)}{\partial y} = s(x, 0)\frac{-1}{s(x, 0)} = -1$$

Thus, for $k = 0$,

$$B_i^0 = -1 \qquad \text{and} \qquad s_i^0 = g(i\delta x)$$

The spreadsheet structure is quite well suited for finite-difference methods, since each nodal point can be represented by a cell. The Lotus 1-2-3 macro that gives the numerical solution using the above method is shown in Figure 1, with some comments given on the left-hand side of the macro range. The program follows the same format as programs written in FORTRAN or BASIC. In order to understand how the 1-2-3 macro runs, we now describe the main parts of it.

The commands in cells ID1 ... ID19 are already explained in Figure 1 and can be found in the Lotus 1-2-3 user manual. There main use is to define all variables used in the 1-2-3 macro and set the values of λ (LAMDA), δx (DX) and M.

As shown in Figure 1, there are three FOR loops in the program. Loop 1 is used to compute the initial position $s(x, t)$ of the moving boundary and the initial values of $B(x, t)$. The loop starts in cell ID20 and terminates in cell IE28. At the end of loop 1, all values of $s(x, 0)$ and $B(x, 0)$ will be computed and written in the range B6 ... C16. as shown in Figure 2.

The two other loops, loops 2 and 3, are used to compute the values of $s(x, t)$ and $B(x, t)$ at different times. Loop 2 sets the counter for the time steps and loop 3

A	IB	IC	ID	IE	IF	IG	IH	II	IJ	IK	IL	IM
1	Set screen display off		{HOME}{WINDOWSOFF}									
2	Reset all range names		/rnr~/rnc\A~ID1~									
3	Erase range of output		{BLANK A5..HA200}~									
4	Write label "x \ y"		{GOTO}A5~^x \ t~									
5	Write labels S(t), B(t)		{GOTO}B4~^S(t)~{RIGHT}^B(t)~{RIGHT}^...~{RIGHT}^...~									
6	Define the variables:		/rncDT~B2~/rncN~A2~									
7	DT, N, TIME, G(X),		/rncTIME~B5~/rncG(X)~D2~									
8	TMAX,		/rncTMAX~C2~									
9	DX, LAMDA		/rncDX~IA1~/rncLAMDA~IA2~									
10	I, K, M, X		/rncI~IA3~/rncK~IA4~									
11			/rncM~IA5~/rncX~IA6~									
12	Create range "U"		{GOTO}B6~/rncU~{RIGHT}{DOWN N}~									
13	Set t=0 and x=0.		{LET TIME,0}~{LET X,0}~									
14	Set DX=1/N		{LET DX,1/N}~									
15	Compute lambda		{LET LAMDA,DT/(DX*DX)}~									
16	Set M=TMAX/DT		{LET M,TMAX/DT}~									
17	Write x0,x1,..		{GOTO}A6~+0~{DOWN}+A6+$DX~/c~.{DOWN N-1}~									
18	Set sceen display on.		{WINDOWSON}~									
19	Move cursor to "U"		{GOTO}U~									
20	FOR Loop 1 ----->		{FOR I,0,N,1,IE24}~									
21	FOR Loop 2 ----->		{FOR K,1,M,1,IE30}~									
22			{QUIT}~									
23												
24				+G(X)~/rv~~								
25		Compute initial		{RIGHT}-1~{LEFT}								
26		s(x,0)=g(x) and		{DOWN}								
27		B(x,0)=1 and write		{LET X,X+DX}~								
28		it in range B5..C16		{RETURN}~								
29												
30		Move cursor to "U"		{GOTO}U~{RIGHT 2}								
31		FOR Loop 3 ----->		{FOR I,0,N,1,IF40}~								
32				{IF @MOD(K,G2)=0#OR#K=1}{BRANCH IE35}~								
33		Control the output		{GOTO}U~{RIGHT 2}/m.{END}{DOWN}{RIGHT}~{LEFT 2}~								
34		frequency of s and B.		{LEFT 2}/rncU~.{RIGHT}{DOWN N}~{BRANCH IE38}~								
35				/rncU~{ESC}{RIGHT 2}.{DOWN N}{RIGHT}~								
36		Write the actual		{IF K>1}/rncTIME~{RIGHT 2}~								
37		time.		{IF K>1}{LET TIME,K*DT}~								
38				{RETURN}~								
39												
40			Write the B.C. for		{IF I=0}{LET IM1,0}~{LET IM5,2*@INDEX(U,0,1)}~{BRANCH IF44}~							
41			s(x,t) at x=0 and x=1.		{IF I=N}{LET IM1,0}~{LET IM5,2*@INDEX(U,0,N-1)}~{BRANCH IF44}~							
42					{LET IM1,@INDEX(U,0,I+1)-@INDEX(U,0,I-1)}~							
43					{LET IM5,@INDEX(U,0,I+1)+@INDEX(U,0,I-1)}~							
44			Write the B.C. for		{IF I=N}{LET IN1,0}~{LET IN5,2*@INDEX(U,1,N-1)}~{BRANCH IF48}~							
45			B(x,t) at x=0 and x=1.		{IF I=0}{LET IN1,0}~{LET IN5,2*@INDEX(U,1,1)}~{BRANCH IF48}~							
46					{LET IN1,@INDEX(U,1,I+1)-@INDEX(U,1,I-1)}~							
47					{LET IN5,@INDEX(U,1,I+1)+@INDEX(U,1,I-1)}~							
48					{LET IM2,@INDEX(U,0,I)}~							
49					{LET IN2,@INDEX(U,1,I)}~							
50					{LET IM3,2*(3+IN2)*IM1^2+4*IM2*(2+IN2)*(IM5-2*IM2)}~							
51					{LET IM4,24*DX^2*(2+IN2)+2*IM2*IM1*IN1}~							
52			Compute s from (16)		{LET IM6,IM2+LAMDA/(4*IM2*(8+IN2))*(IM3+IM4)}~+IM6~/rv~~							
53					{LET IN3,-2*(30+13*IN2+IN2^2)*IM1^2+48*IM2*(IM5-2*IM2)}~							
54					{LET IN4,-48*DX^2*(IN2^2+10*IN2+10)-4*IM2*IM1*IN1}~							
55			Compute B from (17)		{LET IN6,IN2+LAMDA*((IN3+IN4)/(4*IM2^2*(8+IN2))+IN5-2*IN2)}~							
56					{RIGHT}+IN6~/rv~~							
57					{LEFT}{DOWN}							
58					{RETURN}~							
59												

Figure 1 Spreadsheet program.

for the space variables x. Note that at each time step $t = (k+1)\,\delta t$, $s[i\delta x, (k+1)\,\delta t]$, and $B[i\delta x, (k+1)\,\delta t]$, $i = 0, 1, 2, \ldots, n$, are computed using the values at the previous time step, that is, at $t = k\delta t$. At $t = k\delta t$ the values of s and B are saved in the range "U" defined by two columns and the values of s and B at the actual time step are written, respectively, in the two columns to the right of range "U" (see Figure 3).

The values of s and B are computed at each time level using the formulas residing in cells IF40 . . . IF58. These formulas are written using Eqs. (16)–(21) and the 1-2-3 function @INDEX, which plays the same role as arrays do in FORTRAN. In order to use it, one has to define first a range in the spreadsheet. In our model problem, the function @INDEX takes its values from the range "U" containing the values of s and B at the previous time step, @INDEX(U,0,I) = s_i^k and @INDEX(U,1,I) = B_i^k ($I = 0, 1, \ldots, N$). The formulas residing in cells IF40 . . . IF55 are defined as follows.

A	A	B	C	D
4		S(t)	B(t)	
5	x \ t	0	...	
6	0	0.3	-1	
7	0.1	0.302462	-1	
8	0.2	0.309789	-1	
9	0.3	0.321799	-1	
10	0.4	0.338197	-1	
11	0.5	0.358579	-1	
12	0.6	0.382443	-1	
13	0.7	0.409202	-1	
14	0.8	0.438197	-1	
15	0.9	0.468713	-1	
16	1	0.5	-1	
17				

Figure 2 Initial values of $s(x, t)$ and $B(x, t)$.

(IF40): If $i = 0$ set $IM1 = 0$ and $IM5 = s_1^k$.
(IF41): If $i = n$ set $IM1 = 0$ and $IM5 = s_{n-1}^k$.
(IF42): If $i \neq 0$ and $i \neq n$ set $IM1 = s_{i+1}^k - s_{i-1}^k$.
(IF43): If $i \neq 0$ and $i \neq n$ set $IM5 = s_{i+1}^k + s_{i-1}^k$.
(IF44): If $i = n$ set $IN1 = 0$ and $IN5 = B_{n-1}^k$.
(IF45): If $i = 0$ set $IN1 = 0$ and $IN5 = B_1^k$.
(IF46): If $i \neq 0$ and $i \neq n$ set $IN1 = B_{i+1}^k - B_{i-1}^k$.
(IF47): If $i \neq 0$ and $i \neq n$ set $IN5 = B_{i+1}^k + B_{i-1}^k$.
(IF48): Set $IM2 = s_i^k$.
(IF49): Set $IN2 = B_i^k$.
(IF50): Set $IM3 = 2(3 + IN2)IM1^2 + 4(IM2)(2 + IN2)(IM5 - 2IM2)$.
(IF51): Set $IM4 = 24\delta x^2(2 + IN2) + 2(IM2)(IM1)(IN1)$.
(IF52): Set $IM6 = IM2 + \lambda/(4IM2(8 + IN2))(IM3 + IM4)$, $IM6$ represents Eq. (16).
(IF53): Set $IN3 = -2(30 + 13IN2 + IN2^2)IM1^2 + 48(IM2)(IM5 - 2IM2)$.
(IF54): Set $IN4 = -48\delta x^2(IN2^2 + 10IN2 + 10) - 4(IM2)(IM1)(IN1)$.
(IF55): Set $IN6 = IN2 + \lambda((4IN3 + IN4)/(4IM2^2(8 + IN2)) + IN5 - 2IN2)$, $IN6$ represents Eq. (17).

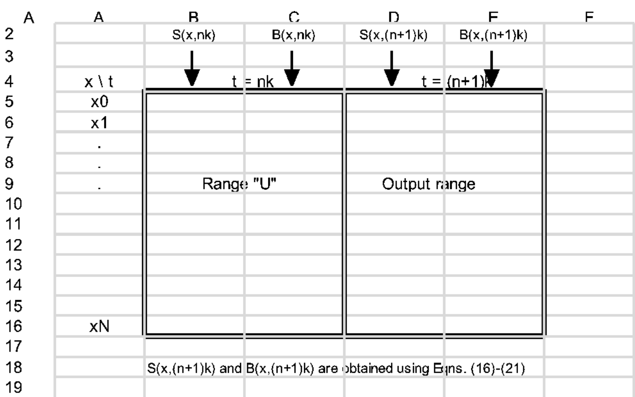

Figure 3 Position of range "U" and the output range.

A	A	B	C	D	E	F
3			Range U			
4		s(x,t)	B(x,t)	s(x,t)	B(x,t)	
5	x \ t	t=kDT	t=kDT	t=(k+1)DT	t=(k+1)DT	
6	i=0					
7	i=1					
8	.					
9	.					
10	i-1	@INDEX(U,0,I-1)	@INDEX(U,1,I-1)			
11	i	@INDEX(U,0,I)	@INDEX(U,1,I)	s(iDX,(k+1)DT)	B(iDX,(k+1)DT)	
12	i+1	@INDEX(U,0,I+1)	@INDEX(U,1,I+1)			
13						
14						
15						
16	i=n					
17						

Figure 4 Computational molecule of Eqs. (16) and (17).

Figure 4 shows the computational molecule of Eqs. (16) and (17). At the end of loop 3 the macro branches to cell IE32. The IF statement in cell IE32 controls the output frequency of $s(x, t)$ and $B(x, t)$. If K is a multiple of the value in cell G2, then the range "U" is moved 2 columns to the right, otherwise the range containing the actual values of s and B are moved in the range "U." Finally, when $K = M$, loop 2 terminates and the 1-2-3 macro stops.

NUMERICAL RESULTS

We shall give here numerical results of the application of the present 1-2-3 macro taking $\delta x = 0.1$, $\delta t = 0.001$, and $TMAX = 0.4$. In order to check the accuracy of the results obtained from the 1-2-3 macro, we take the same initial position of the moving boundary used in [6] so that we can compare our results with the those obtained in [6]. The initial positions of $s(x, t)$ to be used are

1. $g_1(x) = 0.5 - 0.2\cos(\pi x/2)$
2. $g_2(x) = 1.0 - 0.2e^{-25x^2}$
3. $g_3(x) = 0.5 - 0.2\cos(\pi x)$, $0 \leq x \leq 1, t = 0$

One of the main advantages of using spreadsheets is that data can be easily entered. For our example enter the input data as follows.

Enter N in cell A2, δt in cell B2, $TMAX$, the largest value of t to be considered, in cell C2, the initial position $g(x)$ of s in cell D2, and in cell G2 the number of time steps between successive printings of $s(x, t)$ and $B(x, t)$. For example, if $g(x) = 0.5 - 0.2\cos(\pi x/2)$, move the cursor to cell D2 and type 0.5-0.2*@cos (@pi*x/2).

To run the 1-2-3 macro one needs first to name it by moving the cursor to the first cell where the macro starts and using the command /rnc. In our case, since the 1-2-3 macro starts in cell ID1, we use the command /rnc\A~ID1~ to name it \A. Invoke now the 1-2-3 macro by pressing Alt-A.

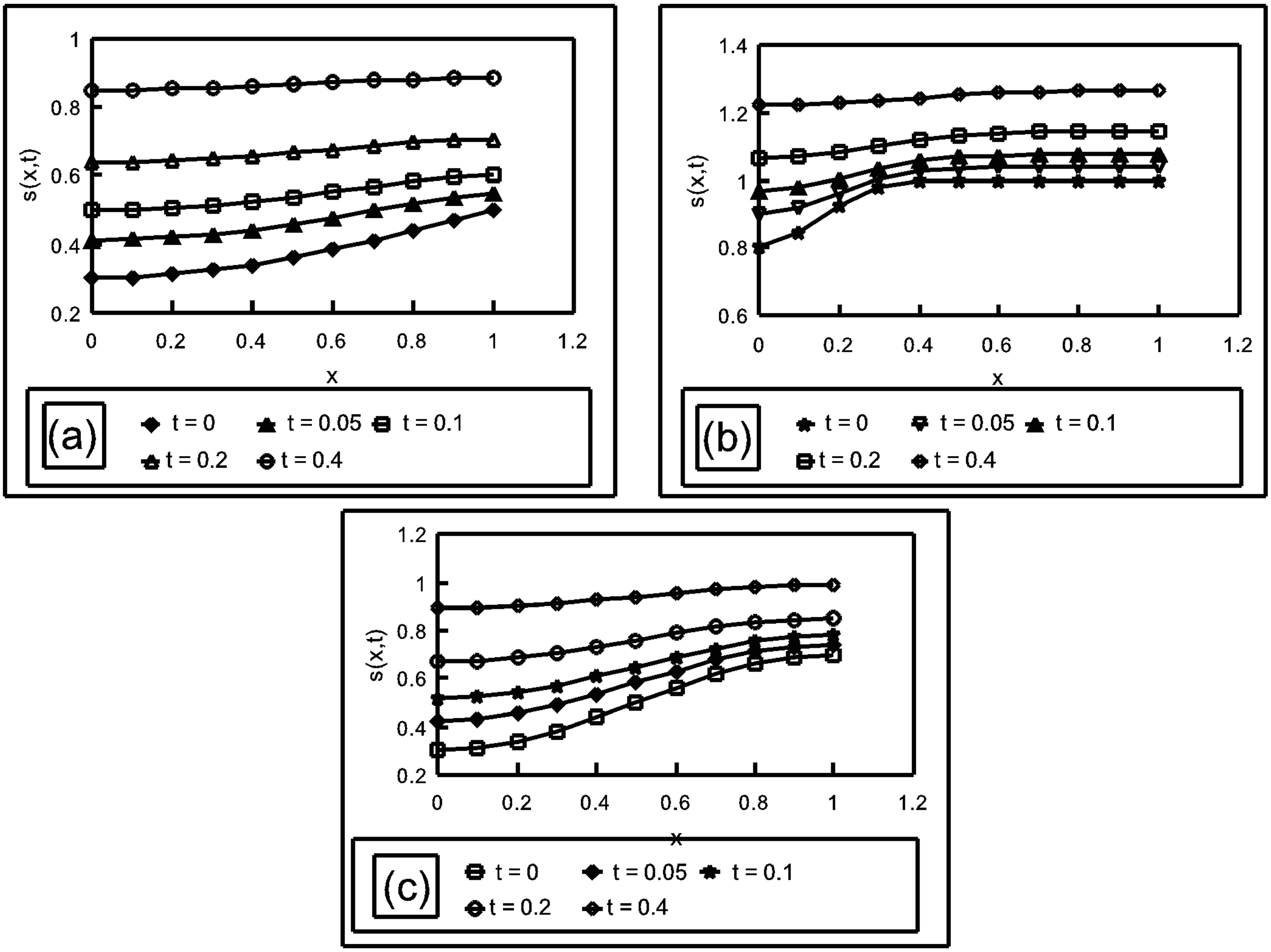

Figure 5 Position of the moving boundary at various times: (*a*) $g_1(x)$; (*b*) $g_2(x)$; (*c*) $g_3(x)$.

Figure 5 shows the plot of the output of the 1-2-3 macro applied to the initial positions $g_1(x)$, $g_2(x)$, and $g_3(x)$ of $s(x, t)$, and Figure 6 shows the position of the moving boundary along $x = 0$ and $x = 1$ at different time steps, obtained from our results and [6]. The error between the results obtained from the present macro and [6] represents the difference between the round-off errors of the two results, which comes from the type of computers used to find the numerical solution. I would expect our results to be more accurate than the ones in [6], since Lotus 1-2-3 carries up to 18 decimal digits of accuracy.

The 1-2-3 macro was run using Lotus 1-2-3 Version 3.4 for DOS. The total execution time was for each example around 4 min and 55 s using a Gateway 2000 486/50 personal computer.

CONCLUSION

A spreadsheet program has been used to find the numerical solution of a two-dimensional Stefan Problem using an approximate method. The sample program used in this article shows that spreadsheets are not limited to basic computations but, through user "macros," they can function in the same way as programs written in high-level language without the need of compiling the program. Data and functions are easily entered by simply moving the cursor to the appropriate cells. Moreover, spreadsheets include built-in graphics capabilities that permit graphs to be viewed on the screen or printed with little effort.

The use of spreadsheet programs and personal computers has resulted in an analysis tool with many applications. As shown in this article, they may include the use of spreadsheets in numerical analysis to solve multidimensional problems in engineering and sciences.

The spreadsheet model presented here has been developed using Lotus 1-2-3 for Windows. The spreadsheet program can be obtained from the author by sending him a formated (IBM or compatible) $5\frac{1}{4}$-in. diskette. For Microsoft Excel users, Excel enables you to open Lotus 1-2-3 worksheets and run the macros they contain.

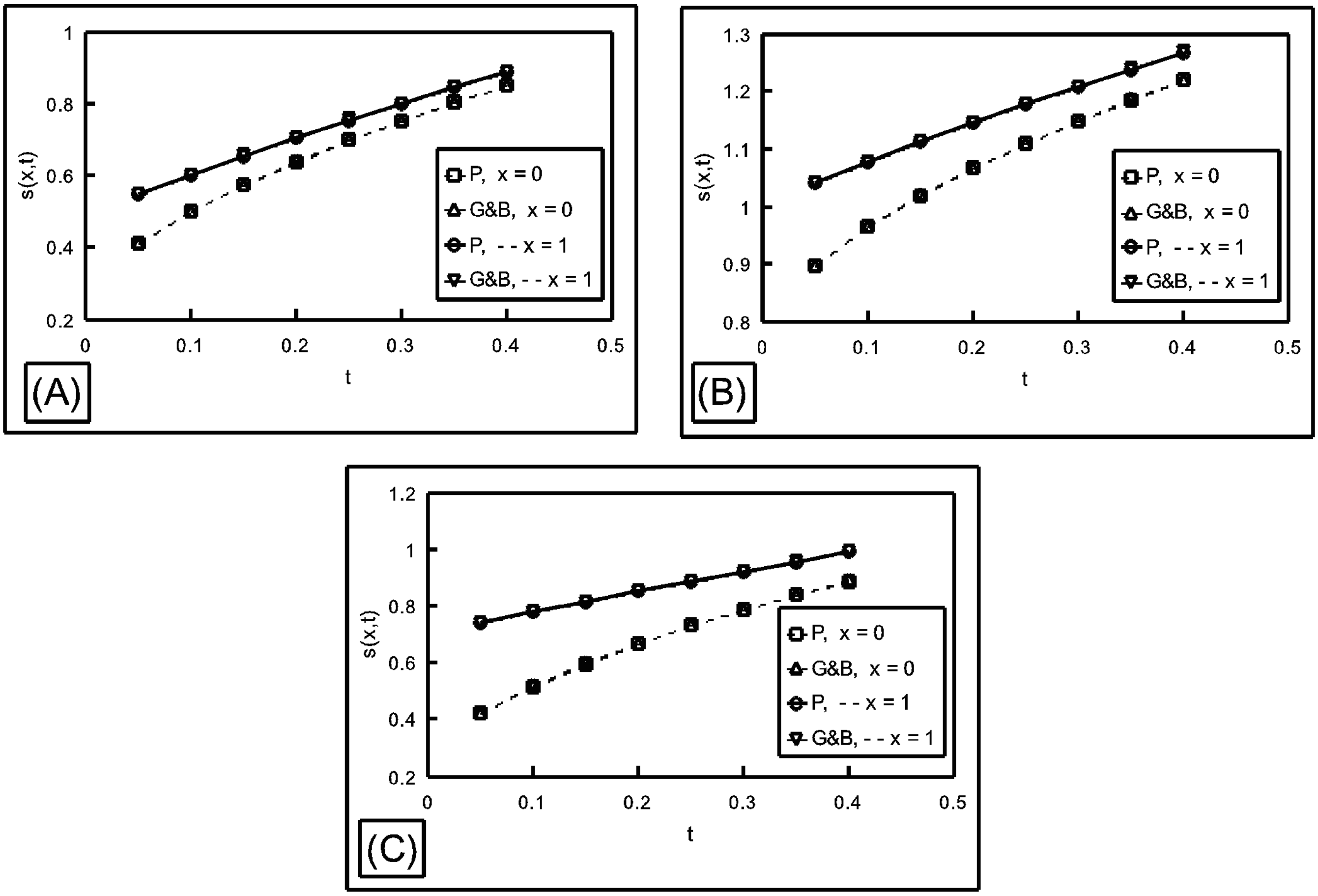

Figure 6 Comparison of the position of the moving boundary along $x = 0$ and $x = 1$ at various time steps: (A) $g_1(x)$; (B) $g_2(x)$; (C) $g_3(x)$.

REFERENCES

[1] Crank, J., *Free and Moving Boundary Value Problems*, Oxford University Press, New York, 1984.

[2] Kharab, A., Use of a Spreadsheet Program in a Two-Dimensional Heat Conduction Problem, *Comput. Meth. Appl. Mech. Eng.*, vol. 122, pp. 173–181, 1995.

[3] Kerkes, D. J., Spreadsheet Solution for Transient Fluid Flow in a Porous Media with a Moving Drainage Boundary, *Computational Mechanics Publications. Proc. First Int. Conf.*, Southampton, U.K., vol. 1, pp. 49–62, 1991.

[4] Rao, N. D., Typical Applications of Microcomputer Spreadsheets to Electrical Engineering Problems, *IEEE Trans. Educ.*, vol. E-27, pp. 237–242, 1984.

[5] Hagler, M., Spreadsheet Solution of Partial Differential Equations, *IEEE Trans. Educ.*, vol. E-30, pp. 130–134, 1987.

[6] Gupta, R. S., and Banik, N. C., Solution of a Weakly Two-Dimensional Melting Problem by an Approximate Method, *J. Comput. Appl. Math.*, vol. 31, pp. 351–356, 1990.

Abdelwahab Kharab is an Associate Professor of Mathematics at King Fahd University of Petroleum & Minerals. He received his B.S. from University of Constantine, Algeria, in 1976, and the M.Sc. and Ph.D. degrees in Applied Mathematics from Oregon State University at Corvallis, OR, in 1980 and 1984, respectively. Since 1985 he has been with the Department of Mathematics at King Fahd University of Petroleum & Minerals, Dhahran, Saudi Arabia. He was a Visiting Professor at Oregon State University at Corvallis during 1991–1992. His research interests are numerical analysis and free boundary-value problems.