MANCHESTER 1824
Manchester University Press

Finite difference solution of the Falkner–Skan wedge flow equation

S. Han
Department of Mechanical Engineering, Tennessee Technological University, Cookeville, TN 38505, USA
E-mail: shan@tntech.edu

Abstract A simple numerical method for the solution of the Falkner–Skan boundary layer flow is presented. A third-order ordinary differential equation is recast into two ordinary differential equations, one first order and the other second order. Finite difference approximations of both equations are sequentially solved iteratively with any arbitrary initial conditions. Numerical solutions were found to be in good agreement with published data over a wide range of wedge flows.

Keywords solutions; Falkner–Skan; difference scheme; Matlab

Introduction

The Falkner–Skan wedge flow equation is obtained by combining two partial differential equations that describe the conservation of mass and linear momentum of a steady two-dimensional laminar boundary layer flow with a non-zero pressure gradient in terms of a similarity variable. The details of its derivation and solutions are usually part of a first-year graduate fluid mechanics course.

The Falkner–Skan boundary layer equation is given by [1]:

$$f''' + \frac{m+1}{2} ff'' + m(1 - f'f') = 0 \tag{1}$$

where the differentiation is with respect to the similarity variable, η, and m is related to the wedge angle. Boundary conditions at the wall ($\eta = 0$) and at the free stream ($\eta = \infty$) are:

$$f(0) = 0, \quad f'(0) = 0, \quad f'(\infty) = 1 \tag{2}$$

The usual method of solution presented in the literature is to replace equation 1, which is third order, with three first-order ordinary differential equations for f, f' and f''. Integration of these equations using any ordinary differential equation solver, such as the Runge–Kutta method, requires the boundary condition for $f''(0)$, which is unknown. The 'shooting method' employs a guessed boundary condition, $f''(0) = \lambda$, to initiate the integration; the accuracy of the solution is then tested by comparing it with the known boundary condition $f'(\infty) = 1$, i.e.,

$$F(\lambda) = |f'_{\lambda}(\infty) - 1| \approx 0 \tag{3}$$

http://dx.doi.org/10.7227/IJMEE.41.1.1

where $f'_\lambda(\infty)$ is the value obtained with the choice of $f''(0) = \lambda$. The proper value of λ is then obtained by varying λ values until equation 3 is satisfied. Either linear interpolation [2] or the Newton–Raphson method [3] are frequently used for this purpose. Unless an initial choice of λ is sufficiently close to the true value, however, these approaches often fail because $F(\lambda) = 0$ is usually a multi-valued function of λ, except with $m = 0$. Another method, which is more robust but complicated, is to use a minimization procedure with a perturbation equation [4].

In this paper, a standard finite difference scheme well known to senior undergraduate students in engineering is used to avoid the complications associated with the shooting method.

Finite difference equation

Equation 1 can be recast into two ordinary differential equations:

$$f' = z \tag{4}$$

$$z'' + \frac{m+1}{2} f z' + m(1 - z^2) = 0 \tag{5}$$

Equations 4 and 5 are to be solved for two dependent variables, f and z. The calculation domain is equally divided into N nodes with uniform grid spacing, as shown in Fig. 1. The value of η_{max} should be large enough to reach the free stream condition.

Equation 4 is approximated at the mid-point between nodes i and $i + 1$ by using the second-order central difference scheme,

$$\frac{f_{i+1} - f_i}{2\frac{\Delta\eta}{2}} = \frac{1}{2}(z_i^* + z_{i+1}^*)$$

where * indicates unknown z_i values. Rearranging this:

$$f_{i+1} = f_i + \frac{1}{2}(z_i^* + z_{i+1}^*)\Delta\eta; i = 1, 2, \ldots, N-1 \tag{6}$$

With the known boundary condition $f_1 = 0$ and guessed values for z_i^*, *all* f_i are calculated.

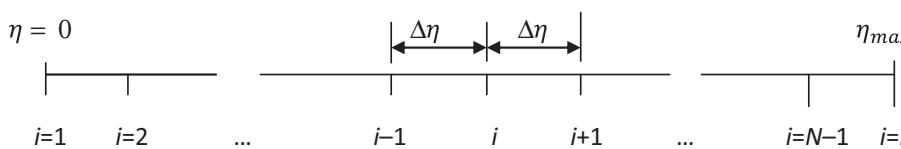

Fig. 1 *Grid system with uniform grid spacing.*

Next, a second-order central difference scheme applied to equation 5 at the regular node i results in:

$$\frac{z_{i-1}-2z_i+z_{i+1}}{\Delta\eta^2}+\left(\frac{m+1}{2}\right)f_i\frac{z_{i+1}-z_{i-1}}{2\Delta\eta}+m\left(1-z_i^2\right)=0$$

Rearranging this,

$$a_i z_i = b_i z_{i+1} + c_i z_{i-1} + d_i; i = 2, 3, \ldots, N-1 \tag{7}$$

where the coefficients and source terms are

$$a_i = \left(\frac{2}{\Delta\eta^2} + m z_i^*\right) \tag{8}$$

$$b_i = \left(\frac{1}{\Delta\eta^2} + \frac{m+1}{4\Delta\eta} f_i\right) \tag{9}$$

$$c_i = \left(\frac{1}{\Delta\eta^2} - \frac{m+1}{4\Delta\eta} f_i\right) \tag{10}$$

$$d_i = m \tag{11}$$

With the known boundary conditions, $z_1 = 0$ and $z_N = 1$, equation 7 provides a set of simultaneous equations in tri-diagonal matrix form for (N–2) unknown z_i.

Equations 6 and 7 are solved repeatedly in sequence until the solution converges within a prescribed tolerance, $\left|\frac{z_i^* - z_i}{z_i}\right| \le \varepsilon$. The computational procedures are:

Step 1. Assign initially guessed values for z_i^*;
Step 2. Calculate f_i using equation 6;
Step 3. Calculate z_i using equation 7;
Step 4. Does the convergence, $\left|\frac{z_i^* - z_i}{z_i}\right| \le \varepsilon$? If yes, the solution is obtained. If not, set $z_i^* = z_i$ and go back to step 2. Repeat steps 2–4 until the solution converges.

The accuracy of the solution is checked by comparing the skin friction coefficient expressed in terms of $f''(0)$ with the available data. In the present formulation, dependent variables are f and f'. Therefore, $f''(0)$ is estimated by using a second-order forward difference scheme at $\eta = 0$:

$$f''(0) = z'(0) = \frac{-z_3 + 4z_2 - 3z_1}{2\Delta\eta} = \frac{1}{2} C_{fx} \sqrt{Rex} \tag{12}$$

TABLE 1 *Comparison of results in skin friction* ($\frac{1}{2}C_{fx}\sqrt{Rex}$)

m	Published data [7]	Present results	Number of iterations
10	n/a	3.7552	14
5	2.6344	2.6784	13
1	1.2326	1.2326	9
1/3	0.75746	0.75752	6
0	0.33206	0.3321	12
–0.06542	0.16372	0.16395	24
–0.09000	n/a	0.0195	160
–0.09041	0	0.00624	430

Results

All the numerical results shown in Table 1 are obtained with $\eta_{max} = 10$ and an equal grid spacing, $\Delta\eta = 0.1$. This choice covers a wide range of boundary layer thicknesses and makes all coefficients appearing in equation 7 positive for a guaranteed convergence [5]. The initial guessed value for dependent variable z_i is also fixed at $z_i = 1$. The solution is assumed to have converged when the convergence criterion $\varepsilon = 10^{-6}$ is met. The number of iterations to reach the converged solution for each case is included in Table 1 for reference. Numerical code is written in Matlab [6], which is routinely used nowadays in engineering classroom settings. The code is reproduced in the Appendix for reference.

Conclusions

Application of a standard finite difference scheme and Matlab for the solution of the Falkner–Skan boundary layer flow equation is presented. This method provides an alternative way to solve an important fundamental fluid mechanics problem without the tedious steps required in other methods. The straight forward nature of the method is particularly suitable for senior and first-year graduate students.

References

[1] H. Schlichting, *Boundary-Layer Theory*, 6th edition (McGraw-Hill, New York, 1968), p. 150.
[2] F. M. White, *Viscous Fluid Flow* (McGraw-Hill, New York, 1974), pp. 175–178.
[3] S. R. Otto and J. P. Denier, *An Introduction to Programming and Numerical Methods in Matlab* (Springer, London, 2005), pp. 274–276.
[4] A. Bejan, *Convection Heat Transfer* (Wiley Interscience, New York, 1984), pp. 429–434.
[5] S. V. Patankar, *Numerical Heat Transfer and Fluid Flow* (McGraw-Hill, New York, 1980), p. 37.
[6] Matlab version R2008a (MathWorks Inc., Natick, MA, 2008).
[7] S. Kakac and Y. Yener, *Convective Heat Transfer*, 2nd edition (CRC Press, Abingdon, 1995), p. 75.

Appendix

```
%Falkner-Skan Wedge boundary layer flow is solved by
%central difference scheme
%
% f'''+0.5(m+1)ff''+m(1-f'f')=0 is transformed
% to z=f' and z''+0.5(m+1)fz'+m(1-zz)=0
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
close all
clear all
clc
%select geometry: geo=m
geo=10; % 0=blasius, 1=planar stagnation flow, etc
%x=eta, te=z in this program
n=101;%number of grid points in eta direction
maxiter=200;%maximum iteration number allowed
%define calculation domain
tl=10; %maximum eta value where free stream condition is found (it
is about 10)
delx=tl/(n-1);%increment of eta
dx=ones(1,n-1);% initialization
dx=delx*dx;%
%assign eta-coordinate
x=zeros(1,n);%initialization
x(1)=0;
for m=1:n-1
x(m+1)=x(m)+dx(m);
end
%prescribe initial z values
te=zeros(1,n);%initialization of z
tep=te;%initialization of temporary z
for i=1:n
te(i)=1.0;% any arbitrary value
tep(i)=te(i);
end
%initialize f
f=zeros(1,n);%initialization of f
%iteration for convergence
iter=0;%iteration counter
iflag=1;%flag to indicate convergence, iflag=0 means converged solution
%iteration loop for the convergence
while iflag==1 % end is at the end of program marked by
****************
%
for i=1:n-1
```

```
f(i+1)=f(i)+0.5*(te(i)+te(i+1))*dx(i);%calculate f integrating f'=z
end
%prescribe boundary conditions
te(1)=0;%at eta=0 f'=0
te(n)=1;%at eta=infinite, f'=1
%initialize the coefficients for tdma matrix
%evaluate the diffusion conductance and source terms
for i=2:n-1
% diffusion conductance
ae=1/(dx(i)^2)+0.5*(geo+1)*f(i)/(2*dx(i));
aw=1/(dx(i)^2)-0.5*(geo+1)*f(i)/(2*dx(i));
%source term evaluation
sc=geo;
ap=2/(dx(i)^2)+geo*te(i);
b=sc;
%setting coefficients for tdma matrix
ta(i)=ap;
tb(i)=ae;
tc(i)=aw;
td(i)=b;
%modify using boundary conditions
if i==2
td(i)=td(i)+aw*te(1);
elseif i==n-1
td(i)=td(i)+ae*te(n);
end
end
%Begining TDMA (solve the simultaneous equations by Thomas
Algorithm)
nq=n-2;%number of equations to be solved
nqp1=nq+1;
nqm1=nq-1;
%forward substitution
beta(2)=tb(2)/ta(2);
alpa(2)=td(2)/ta(2);
for i=3:nqp1
beta(i)=tb(i)/(ta(i)-tc(i)*beta(i-1));
alpa(i)=(td(i)+tc(i)*alpa(i-1))/...
(ta(i)-tc(i)*beta(i-1));
end
%backward substitution
dum(nqp1)=alpa(nqp1);
for j=1:nqm1
i=nqp1-j;
dum(i)=beta(i)*dum(i+1)+alpa(i);
```

```
end
%end of TDMA routine
%update the dependent variable, new z
for i=2:n-1
te(i)=dum(i);
end
%check the convergence
for i=1:n
errote(i)=abs(te(i)-tep(i))/te(i);
end
error=1.0e-6;%error tolerance
if (max(errote)>error)
iter=iter+1;
tep=te;
iflag=1;
else
iflag=0;
end
if iter>maxiter
break
end
end % this end goes with the while iflag==1 at the
top*************************
%
%print the results
fprintf('iteration number is %i \n',iter)
%plot the result eta.vs.u/u_inf
plot(x,te,'-')
grid on
title('Falkner Skan Wedge Flow'),xlabel('eta'),ylabel('u/u_inf')
%
table=[x',f',te'];%save results in a table form
disp('eta f f_prime ')
disp([num2str(table)]);
%numerically calculated f''(0)
gradient_z=(-te(3)+4*te(2)-3*te(1))/(2*dx(2));
disp([,f"(0)= ,, num2str(gradient_z)])
```