Computer code for solving the Motz problem with collocation method

Danial Amini Baghbadorani[1]

[1]Independent researcher, 3rd Floor, No. 34, Golestan Dead End., Dehghan Alley, Bazarche St., Shahrestan Bridge, Moshtagh 2 St., P.O. Box 81581-36647, Esfahan, Iran. Email: danial.amini@gmail.com

Abstract

In this brief note, the coefficients for the series describing the solution of the Motz problem is obtained via collocation method and the computer code is presented. The presented method showed sufficient accuracy up to the 28th term of the radial harmonic series.

Keywords: Collocation, Motz problem, Laplace equation

Problem statement

Motz's problem is a benchmark problem for Laplacian equation with boundary conditions that are "difficult" to impose. The known numerical solutions of this problems can be used to assess the performance of various numerical methods, such as the spectral collocation methods. Highly accurate solutions exist for solving the Motz problem with special refinements. However, these algorithms are tailored for solving this particular problem. It is also interesting to see how an algorithm performs without these refinements, because the algorithms tested with the Motz algorithm can be applied to other, more practical problems.

The Motz problem is expressed by $\nabla^2 u=0$ over the domain shown in Figure 1.

Benchmark solution with polar harmonics

A solution satisfying the boundary condition in the *x*-axis can be obtained by separation of variables and solving the eigenproblem. This leads to the series solution $u(r,\theta)=\sum_{n=0}^{M} d_n r^{n+1/2}\cos\left[(n+1/2)\theta\right]$ (Hong, 2016) with the first 40 coefficients listed in Table 1. Considering that $u_x=\cos\theta u_r-\sin\theta/r\,u_\theta$ and $u_y=\sin\theta u_r+\cos\theta/r\,u_\theta$, we obtain $u_x=\sum_{n=0}^{M} d_n(n+1/2)r^{n-1/2}\cos\left[(n-1/2)\theta\right]$ and $u_y=-\sum_{n=0}^{M} d_n(n+1/2)r^{n-1/2}\sin\left[(n-1/2)\theta\right]$ (Lu et al., 2004). Here ∇=Nabla operator, *u*=function value, *x*=horizontal coordinates, *y*=vertical coordinates, *r*=radius, θ=angle, *n*=term counter, *M*=number of terms, d_n=expansion coefficient, and subscripts denote differentiation with respect to variables.

Computer code

The computer code for the algorithm was written in Octave language, available in and in Table 1. Octave is a free and open source software with very similar syntax to Matlab and the same m file extension, although Octave is considerably slower. The slow speed is not a problem in linear, steady-state problems. Although, time dependent problems require better solvers.

Method 1 (Octave file: motz_runner2.m)

The problem is thus reduced to imposing boundary conditions on the right, top and left sides. We used uniform node spacing in Cartesian coordinates. For the right side, we have x=1 and y=j/N wherein 0≤j≤N, thus $\theta=\tan^{-1}j/N$. Here N=number of nodes. The boundary condition is $\sum_{n=0}^{M} d_n \left(1/\cos\theta\right)^{n+1/2}\cos\left[\left(n+1/2\right)\theta\right]=500$ with suitable substitution to remove r. For the top side, we have x=(N−j)/N and y=1 wherein 0≤j≤2N, thus $\theta=\pi/2-\tan^{-1}(1-j/N)$. The boundary condition is $-\sum_{n=0}^{M} d_n \left(n+1/2\right)\left(1/\sin\theta\right)^{n-1/2}\sin\left[\left(n-1/2\right)\theta\right]=0$. For the left side, we have x=−1 and y=1−j/N wherein 0≤j≤N, thus $\theta=\pi-\tan^{-1}(1-j/N)$. The boundary condition is $\sum_{n=0}^{M} d_n \left(n+1/2\right)\left(-1/\cos\theta\right)^{n-1/2}\cos\left[\left(n-1/2\right)\theta\right]=0$. Writing out these equations in matrix form leads to $[A]_{\hat{N}\times M}\,\{d\}_{M\times 1} = \{B\}_{\hat{N}\times 1}$, wherein [$A$] is the coefficient matrix, {d} is the vector containing d_n coefficients, and {B} is the right hand side vector containing values of Dirichlet and Neumann boundary conditions for right, top and left boundary conditions. Also, $\hat{N}$=4N+3= overall number of nodes. This implies that the nodes for right side are written in rows 1 until N+1; the nodes for top side are written in rows N+2 until 3N+2; and the nodes for the left side are written in rows 3N+3 until 4N+3. The rectangular system is converted to square system by premultiplying A^T (transpose of A), resulting in $[\hat{A}]_{M\times M}\{d\}_{M\times 1}=\{\hat{B}\}_{M\times 1}$ wherein $[\hat{A}]=[A^T][A]$ and $\{\hat{B}\}=[A^T]\{B\}$. In order to avoid large numbers, it is better to use $d_n=\left(\sqrt{2}\right)^{-(n+2)}c_n$ to scale the columns of the coefficient matrix to more comparable values.

Calculations were made using M=40 and N=10000, therefore, $\hat{N}$≈40000 nodes were used. Results are truncated to progressively fewer significant digits for higher terms because accuracy is less important in those terms. Comparison of results in done in Table 2, showing that the simple algorithm presented herein can obtain coefficients with reasonable accuracy up until 28th term using the truncated values. However, the algorithm is much worse than methods in Lu et al. (2004) and Hong (2016), although this is not surprising for the presented simple, crude and unrefined algorithm. Slight differences are observed for the last reported digits for terms n=8, 9 and 19. We finally remark although the choice of the fundamental solution for the Laplace equation is of the form cos$x$ cosh$y$, cos$x$ sinh$y$, etc. in rectangular domains, presence of singularities sometimes necessitates fundamental solutions of the form r^n cos$n\theta$. The presented algorithm showed sufficient capability for application in those problems.

Method 2

Method 2 is exactly the same as method 1, though written in a different formulation.

$$u=\sum_{j=0}^{M} d_j\phi_j, u_x=\sum_{j=0}^{M} d_j\phi_{xj}, u_z=\sum_{j=0}^{M} d_j\phi_{zj}, \frac{\partial u}{\partial d_j}=\phi_j$$

$$\phi_j=r^{j+\frac{1}{2}}\cos\left[\left(j+\frac{1}{2}\right)\theta\right], \phi_{jx}=\left(j+\frac{1}{2}\right)r^{j-\frac{1}{2}}\cos\left[\left(j-\frac{1}{2}\right)\theta\right],$$

$$\phi_{jy}=-\left(j+\frac{1}{2}\right)r^{j-\frac{1}{2}}\sin\left[\left(j-\frac{1}{2}\right)\theta\right]$$

$$F = \int_{Right} (u-500)^2\, dy + \int_{Top} u_y^2 dx + \int_{Left} u_x^2 dy$$

Minimize F $\Rightarrow \dfrac{\partial F}{\partial d_j} = 0$

$$\frac{\partial F}{\partial d_j} = \int_{Right} (u-500)\frac{\partial u}{\partial d_j} dy + \int_{Top} 2u_z \frac{\partial u_z}{\partial d_j} dx + \int_{Left} 2u_x \frac{\partial u_x}{\partial d_j} dy = 0$$

$$\int_{Right} u \frac{\partial u}{\partial d_j} dy + \int_{Top} 2u_z \frac{\partial u_z}{\partial d_j} dx + \int_{Left} 2u_x \frac{\partial u_x}{\partial d_j} dy = 500\int_{Right} u \frac{\partial u}{\partial d_j} dy$$

$$\sum_{k=0}^{M} d_k \int_{Right} \phi_j \phi_k dy + \sum_{k=0}^{M} d_k \int_{Top} \phi_{yj}\phi_{yk} dx + \sum_{k=0}^{M} d_k \int_{Top} \phi_{xj}\phi_{xk} dy = 500\int_{Right} \phi_{xj} dy \ \text{ for } 0 \le j \le M$$

Therefore

$$\left(\left[\int_{Right} \phi_j \phi_k dy\right] + \left[\int_{Top} \phi_{yj}\phi_{yk} dx\right] + \left[\int_{Top} \phi_{xj}\phi_{xk} dy\right]\right)\{d_k\} = 500\left\{\int_{Right} \phi_{xj} dy\right\}$$

In other words the following system is formed
$[A]_{(M+1)\times(M+1)}\{d\}_{(M+1)\times 1}=\{R\}_{(M+1)\times 1}$.
The integrals can be calculated using the Trapezoidal rule.

Comment about methods 1 & 2

The method is inefficient because it needs a lot of collocation (integration) points. Methods for stabilizing the matrix are: pre-conditioning, singular decomposition, and Tikhonov regularization, although, the accuracy of the solution will not improve.
Using more computational digits will alleviate the problem. Instead of double precision, use quadruple precision if condition number of coefficient matrix is 10^{16}, etc (personal communications with Prof. Alex Cheng).

References

Hong, W.-T., 2016. Enriched meshfree method for an accurate numerical solution of the Motz problem. Advances in Mathematical Physics 2016.
Lu, T.-T., Hu, H., Li, Z.-C., 2004. Highly accurate solutions of Motz's and the cracked beam problems. Engineering Analysis with Boundary Elements 28, 1387-1403.

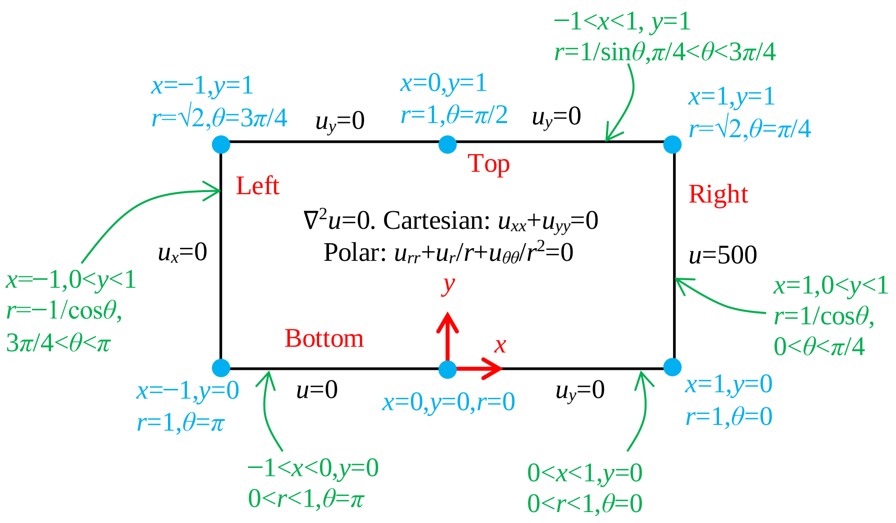

Figure 1. Motz problem along with important points (blue text) and boundary line segments (green text) in Cartesian and polar coordinates

Table 1. Code files in https://github.com/DanialAmini/Motz_Problem_Collocation

File	Language	Description
motz_runner2.m	Octave	Method 1
testing_octave_motz_save2.m	Octave	Method 2

Table 1. Octave code for method 1

```
clear;  format long;

M=40; N=10000; NN=4*N+3; A=zeros(NN,M); B=zeros(NN,1); V_xyt=zeros(NN,3);

for j=0:N
  V_xyt(j+1,1)=1;   V_xyt(j+1,2)=j/N;   t=atan(j/N);   V_xyt(j+1,3)=t;
  for n=0:M-1
    A(j+1,n+1)=((1/cos(t))^(n+1/2))*cos((n+1/2)*t)/sqrt(2.5)^(n+2);
  endfor
endfor
for j=0:N
  B(j+1)=500;
endfor

for j=0:2*N
  V_xyt(N+1+j+1,1)=(N-j)/N;   V_xyt(N+1+j+1,2)=1;   t=pi/2-atan((N-j)/N);
V_xyt(N+1+j+1,3)=t;
  for n=0:M-1
    A(N+1+j+1,n+1)=-(n+1/2)*((1/sin(t))^(n-1/2))*sin((n-
1/2)*t)/sqrt(2.5)^(n+2);
  endfor
endfor
for j=0:2*N
  B(N+1+j+1)=0;
endfor

for j=0:N
  V_xyt(3*N+2+j+1,1)=-1;    V_xyt(3*N+2+j+1,2)=1-j/N;    t=pi-atan(1-j/N);
V_xyt(3*N+2+j+1,3)=t;
  for n=0:M-1
    A(3*N+2+j+1,n+1)=(n+1/2)*((-1/cos(t))^(n-1/2))*cos((n-
1/2)*t)/sqrt(2.5)^(n+2);
  endfor
endfor
for j=0:N
  B(3*N+2+j+1)=0;
endfor

%C=A\B
C=(transpose(A)*A)\(transpose(A)*B)

DD=zeros(M,1);
for n=0:M-1
  DD(n+1)=C(n+1)/sqrt(2.5)^(n+2);
endfor
```

Table 2. Comparison of d_n (values for N=10000, M=40)

n	Hong (2016), no trunction	Lu et al. (2004), Significant digits	Octave
0	401.162453745234	401.1624537	401.1624537
1	87.6559201950879	87.6559202	87.6559202
2	17.2379150794467	17.23791508	17.23791508
3	-8.07121525968247	-8.07121526	-8.07121526
4	1.44027271702287	1.440272717	1.440272717
5	0.331054885920767	0.331054886	0.331054886
6	0.275437344508727	0.275437345	0.275437345
7	-8.69329946589323E-02	-0.086932995	-0.086932995
8	3.36048786193139E-02	0.033604878	0.033604878
9	1.53843742840408E-02	0.015384375	0.015384374
10	7.30230172821564E-03	0.007302302	0.007302302
11	-3.18411377492366E-03	-0.003184114	-0.003184114
12	1.22064638572558E-03	0.001220646	0.001220646
13	5.30965296814512E-04	0.000530965	0.000530965
14	2.71512155856615E-04	0.000271512	0.000271512
15	-1.20045575740674E-04	-0.00012005	-0.00012005
16	5.05397300846273E-05	0.00005054	0.00005054
17	2.31670163990000E-05	0.00002317	0.00002317
18	1.15353890475415E-05	0.000011535	0.000011535
19	-5.29397510035331E-06	-0.00000529	-0.00000530
20	2.29074482358489E-06	0.00000229	0.00000229
21	1.06369298555841E-06	0.00000106	0.00000106
22	5.31498056296845E-07	0.000000531	0.000000531
23	-2.45403558772047E-07	-0.00000025	-0.00000025
24	1.09425820081517E-07	0.00000011	0.00000011
25	5.20412323411083E-08	0.00000005	0.00000005
26	2.59454810138248E-08	0.00000003	0.00000003
27	-1.09175391022716E-08	-0.00000001	-0.00000001