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

Method and Results

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. 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). 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.

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 $0\leq j\leq N$, thus $\theta=\tan^{-1}j/N$. Here N=number of nodes. The boundary condition is $\sum_{n=0}^{M} d_n(1/\cos\theta)^{n+1/2}\cos\left[(n+1/2)\theta\right]=500$ with suitable substitution to remove r. For the top side, we have $x=(N-j)/N$ and y=1 wherein $0\leq j\leq 2N$, thus $\theta=\pi/2-\tan^{-1}(1-j/N)$. The boundary condition is $-\sum_{n=0}^{M} d_n(n+1/2)(1/\sin\theta)^{n-1/2}\sin\left[(n-1/2)\theta\right]=0$. For the left side, we have $x=-1$ and $y=1-j/N$ wherein $0\leq j\leq N$, thus $\theta=\pi-\tan^{-1}(1-j/N)$. The boundary condition is $\sum_{n=0}^{M} d_n(n+1/2)(-1/\cos\theta)^{n-1/2}\cos\left[(n-1/2)\theta\right]=0$. 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.

The computer code for the algorithm was written in Octave language, available in https://github.com/DanialAmini/Motz_Problem_Collocation 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.

Calculations were made using M=40 and N=10000, therefore, $\hat{N}\approx$40000 nodes were used. Results are truncated to progressively fewer significant digits for higher terms because accuracy is less important in those terms. Comparison of results in done in Table 2, showing that the simple algorithm presented herein can obtain coefficients with reasonable accuracy up until 28th term using the truncated values. However, the algorithm is much worse than methods in Lu et al. (2004) and Hong (2016), although this is not our expectation for the presented simple, crude and unrefined algorithm. Slight differences are observed for the last reported digits for terms n=8, 9 and 19. We finally remark although the choice of the fundamental solution for the Laplace equation is of the form cos$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.

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.

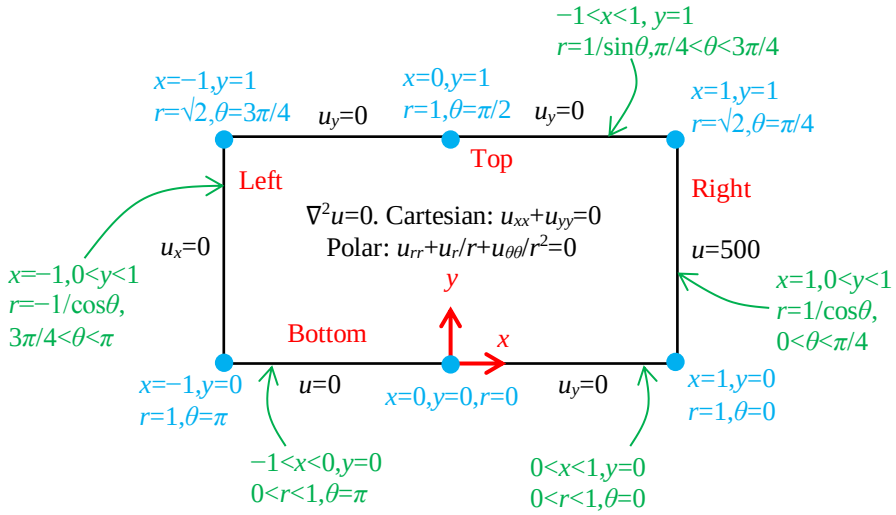

Figure 1. Motz problem along with important points (blue text) and boundary line segments (green text) in Cartesian and polar coordinates

Table 1. Motz problem Octave code.

```
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

n	Hong (2016), truncated	Lu et al. (2004), Significant digits	Octave
0	401.1624537	401.1624537	401.1624537
1	87.6559202	87.6559202	87.6559202
2	17.23791508	17.23791508	17.23791508
3	-8.07121526	-8.07121526	-8.07121526
4	1.440272717	1.440272717	1.440272717
5	0.331054886	0.331054886	0.331054886
6	0.275437345	0.275437345	0.275437345
7	-0.086932995	-0.086932995	-0.086932995
8	0.033604879	0.033604878	0.033604878
9	0.015384374	0.015384375	0.015384374
10	0.007302302	0.007302302	0.007302302
11	-0.003184114	-0.003184114	-0.003184114
12	0.001220646	0.001220646	0.001220646
13	0.000530965	0.000530965	0.000530965
14	0.000271512	0.000271512	0.000271512
15	-0.00012005	-0.00012005	-0.00012005
16	0.00005054	0.00005054	0.00005054
17	0.00002317	0.00002317	0.00002317
18	0.000011535	0.000011535	0.000011535
19	-0.00000529	-0.00000529	-0.00000530
20	0.00000229	0.00000229	0.00000229
21	0.00000106	0.00000106	0.00000106
22	0.000000531	0.000000531	0.000000531
23	-0.00000025	-0.00000025	-0.00000025
24	0.00000011	0.00000011	0.00000011
25	0.00000005	0.00000005	0.00000005
26	0.00000003	0.00000003	0.00000003
27	-0.00000001	-0.00000001	-0.00000001