Applications of Discrete Exterior Calculus on Exact Conservation FEM

Feng Ling*

*University of Texas at Austin.
e-mail: FLing@utexas.edu

This paper surveys the applications of De Rham cohomology and discrete exterior calculus on linear elliptic PDEs. A code for 2D stokes system constructed in this paradigm is tested with known results.

Introduction

why exact conservation FEM
why DEC
what's been done

Quick Survey to Differential Forms

In the de Rham chain complex of differential forms, we first and foremost note the duality between k-forms and $(n-k)$-forms. This duality is realized by the Hodge star operator $\star : \bigwedge^k \to \bigwedge^{n-k}$. By definition, we have $\star\star\alpha^k = (-1)^{k(n-k)} s\alpha^k$ for any k-form α. Here n is the dimension of our space, and s is the signature of the inner product on our space (sign of the determinant of the inner product tensor). Since we would be working in Euclidean space only, s always equals 1.

The differential operator $\mathrm{d} : \bigwedge^k \to \bigwedge^{k+1}$ applied to a k-form is defined to be

$$\mathrm{d} f_I \mathrm{d}x^I = \sum_{i=0}^{k+1} \frac{\partial f_i}{\partial x^i} \mathrm{d}x^i \wedge \mathrm{d}x^I$$

The differential operator is linear, satisfies Leibniz rule, metric free, and form the exact de Rham cohomology sequence.

It has a natural dual operator, appropriately named the codifferential $\mathrm{d}^* : \bigwedge^{k+1} \to \bigwedge^k$. On a k-form, it is defined to be

$$\mathrm{d}^* = (-1)^{n(k+1)+1} \star \mathrm{d} \star$$

where n is as usual the dimension of our space.

The harmonic Laplacian operator $\Delta : \bigwedge^k \to \bigwedge^k$ is defined to be

$$\Delta = \mathrm{d}\mathrm{d}^* + \mathrm{d}^*\mathrm{d}$$

For any 0-form, since its codifferential is always trivial, Δ reduce to $\mathrm{d}^*\mathrm{d}$. Similarly for any n-form, $\Delta = \mathrm{d}\mathrm{d}^*$.

The generalized Stokes' theorem extends integration by parts in 1D to k-forms. In its inner product form, we have

$$(\alpha^k, \mathrm{d}^*\beta^{k+1})_\Omega = (\mathrm{d}\alpha^k, \beta^{k+1})_\Omega - \int_{\partial\Omega} \alpha^k \wedge \star\beta^{k+1}$$

Discretization and Implementation

mesh geometry and topology
incidence matrices (discrete differential operators e.g. discrete gradient, curl, and divergence)
basis functions
mass matrix
boundary conditions

Results and Discussion

Problem 1 *we would like to find the solution to the equation*

$$\Delta u^{(0)} = \mathrm{d}^*\mathrm{d}u^{(0)} = f^{(0)}$$

where $u^{(0)}, f^{(0)}$ are 0-forms on 2D domain Ω

Answer. Introducing the intermediate 1-form p^1

$$\Rightarrow \begin{cases} \mathrm{d}u^{(0)} = p^1 \\ \mathrm{d}^*p^1 = f^{(0)} \end{cases}$$

Taking inner product with test functions $v^{(0)}, w^1$, we get

$$\begin{cases} (w^1, \mathrm{d}u^{(0)})_\Omega = (w^1, p^1)_\Omega \\ (v^{(0)}, \mathrm{d}^*p^1)_\Omega = (v^{(0)}, f^{(0)})_\Omega \end{cases}$$

Performing integration by parts (from generalized Stokes theorem)

$$\begin{cases} (w^1, \mathrm{d}u^{(0)})_\Omega = (w^1, p^1)_\Omega \\ (\mathrm{d}v^{(0)}, p^1)_\Omega - \int_{\partial\Omega} v^{(0)} \wedge \star p^1 = (v^{(0)}, f^{(0)})_\Omega \end{cases}$$

Next we approximate the forms in finite dimension spaces

$$
\begin{aligned}
u_h^{(0)}(\xi^1,\xi^2) &= \sum_{i,j} \mathrm{u}_{i,j} P_{i,j}(\xi^1,\xi^2) \\
v_h^{(0)}(\xi^1,\xi^2) &= \sum_{i,j} \mathrm{v}_{i,j} P_{i,j}(\xi^1,\xi^2) \\
p_h^{1}(\xi^1,\xi^2) &= \sum_{i,j} \mathrm{p}^1_{i,j} L^1_{i,j}(\xi^1,\xi^2) + \sum_{i,j} \mathrm{p}^2_{i,j} L^2_{i,j}(\xi^1,\xi^2) \\
w_h^{(0)}(\xi^1,\xi^2) &= \sum_{i,j} \mathrm{w}^1_{i,j} L^1_{i,j}(\xi^1,\xi^2) d\xi^1 + \sum_{i,j} \mathrm{w}^2_{i,j} L^2_{i,j}(\xi^1,\xi^2) d\xi^2
\end{aligned}
$$

where $P_{i,j} = N_i(x^1)N_j(x^2)$, $L^1_{i,j} = M_i(x^1)N_j(x^2)$, and $L^2_{i,j} = N_i(x^1)M_j(x^2)$ are the basis functions. Here $\mathrm{u}_{i,j}, \mathrm{v}_{i,j}, \mathrm{p}^k_{i,j}, \mathrm{w}^k_{i,j}$ are the nodal/edge values for each finite dimensional projections.

Here we can choose an appropriate reordering of the basis function for both P_i and L_i.

Substituting and rearranging terms give us

$$
\begin{cases}
\mathrm{M}_{(1)}\mathrm{D}_{(1,0)} \cdot \mathbf{u} - \mathrm{M}_{(1)} \cdot \mathbf{p} & = \mathbf{0} \\
\mathrm{D}_{(0,1)}\mathrm{M}_{(1)} \cdot \mathbf{p} - \int_{\partial\Omega} v^{(0)} \wedge \star p^1 & = \mathrm{M}_{(0)}(f^{(0)})
\end{cases}
$$

where $\mathrm{D}_{(1,0)}$ is the discrete matrix representation of divergence and $\mathrm{D}_{(0,1)} = \mathrm{D}^T_{(1,0)}$ is the dual operator. Furthermore, $\mathrm{M}_{(i)}$ is the mass matrix for the i-form basis functions integrated using (Gaussian) quadrature over the parent element domain. Specifically,

$$
\mathrm{M}_{(0)} = \sum_{i,j} \int_\Omega P_i P_j \det J dx^1 \wedge dx^2
$$

$$
\mathrm{M}_{(1)} = \sum_{i,j} \int_\Omega L_i L_j \det J dx^1 \wedge dx^2
$$

$$
\mathrm{M}_{(0)}(f^{(0)}) = \sum_{i} \int_\Omega f^{(0)} P_i^k \det J dx^1 \wedge dx^2
$$

■

Problem 2 *we would like to find the solution to the equation*

$$
\Delta u^2 = \mathrm{d}\mathrm{d}^* u^2 = f^2
$$

where u^2, f^2 are 2-forms on 2D domain Ω

Answer. Introducing the intermediate 1-form p^1

$$
\Rightarrow \begin{cases} \mathrm{d}^* u^2 = p^1 \\ \mathrm{d}p^1 = f^2 \end{cases}
$$

Taking inner product with test functions v^1, w^2, we get

$$\begin{cases}(v^1, \mathrm{d}^* u^2)_\Omega = (v^1, p^1)_\Omega \\ (w^2, \mathrm{d}p^1)_\Omega = (w^2, f^2)_\Omega\end{cases}$$

Performing integration by parts (from generalized Stokes theorem)

$$\begin{cases}(\mathrm{d}v^1, u^2)_\Omega - \int_{\partial\Omega} v^1 \wedge \star u^2 = (v^1, p^1)_\Omega \\ (w^2, \mathrm{d}p^1)_\Omega = (w^2, f^2)_\Omega\end{cases}$$

Next we approximate the forms in finite dimension spaces

$$\begin{aligned}
u_h^2(\xi^1, \xi^2) &= \sum_{i,j} \mathrm{u}_{i,j} S_{i,j}(\xi^1, \xi^2) d\xi^1 \wedge d\xi^2 \\
p_h^1(\xi^1, \xi^2) &= \sum_{i,j} \mathrm{p}_{i,j}^1 L_{i,j}^1(\xi^1, \xi^2) d\xi^1 + \sum_{i,j} \mathrm{p}_{i,j}^2 L_{i,j}^2(\xi^1, \xi^2) d\xi^2 \\
v_h^1(\xi^1, \xi^2) &= \sum_{i,j} \mathrm{v}_{i,j}^1 L_{i,j}^1(\xi^1, \xi^2) d\xi^1 + \sum_{i,j} \mathrm{v}_{i,j}^2 L_{i,j}^2(\xi^1, \xi^2) d\xi^2 \\
w_h^2(\xi^1, \xi^2) &= \sum_{i,j} \mathrm{w}_{i,j} S_{i,j}(\xi^1, \xi^2) d\xi^1 \wedge d\xi^2
\end{aligned}$$

where $P_{i,j} = N_i(\xi^1) N_j(\xi^2)$, $L_{i,j}^1 = M_i(\xi^1) N_j(\xi^2)$, and $L_{i,j}^2 = N_i(\xi^1) M_j(\xi^2)$ are the basis functions. Here $\mathrm{u}_{i,j}, \mathrm{w}_{i,j}, \mathrm{p}_{i,j}^k, \mathrm{v}_{i,j}^k$ are the nodal/edge values for each finite dimensional projections.

Here we can choose an appropriate reordering of the basis function for both P_i and L_i.

Substituting and rearranging terms give us

$$\begin{cases}\mathrm{D}_{(1,2)} \mathrm{M}_{(2)} \cdot \mathbf{u} - \mathrm{M}_{(1)} \cdot \mathbf{p} & = \int_{\partial\Omega} v^1 \wedge \star u^2 \\ \mathrm{M}_{(2)} \mathrm{D}_{(2,1)} \cdot \mathbf{p} & = \mathrm{M}_{(2)}(f^2)\end{cases}$$

where $\mathrm{D}_{(1,0)}$ is the discrete matrix representation of divergence and $\mathrm{D}_{(0,1)} = \mathrm{D}_{(1,0)}^T$ is the dual operator. Furthermore, $\mathrm{M}_{(i)}$ is the mass matrix for the i-form basis functions integrated using (Gaussian) quadrature over the parent element domain. Specifically,

$$\mathrm{M}_{(1)} = \sum_{i,j} \int_\Omega L_i L_j g^{i,j} \det J d\xi^1 \wedge d\xi^2$$

$$\mathrm{M}_{(2)} = \sum_{i,j} \int_\Omega S_i S_j \frac{1}{\det J} d\xi^1 \wedge d\xi^2$$

$$\mathrm{M}_{(0)}(f^{(0)}) = \sum_i \int_\Omega f^{(0)} P_i^k \det J d\xi^1 \wedge d\xi^2$$

■

Problem 3 *We would like to find the solution to the 2D incompressible stationary stokes flow problem in vorticity-velocity-pressure differential forms on a 2D domain* Ω

$$\begin{aligned}\Delta u^1 + \mathrm{d}^* p^2 &= f^1 \\ \mathrm{d}u^1 &= 0\end{aligned}$$

[. Analytic Solution] Here since $n = 2$, we have $\mathrm{d}^* = (-1)^{2k+1} \star \mathrm{d} \star = - \star \mathrm{d} \star$. Since $du^1 = 0$ the Laplacian is reduced to $\Delta = \mathrm{d}\mathrm{d}^*$. Thus we have

$$\begin{aligned}-\mathrm{d} \star \mathrm{d} \star u^1 - \star \mathrm{d} \star p^2 &= f^1 \\ \mathrm{d}u^1 &= 0 \\ \omega^0 &= \mathrm{d}^* u^1\end{aligned}$$

Given solutions

$$\begin{aligned}u^1 &= u^x \mathrm{d}x + u^y \mathrm{d}y \\ p^2 &= p \, \mathrm{d}x \wedge \mathrm{d}y\end{aligned}$$

we can calculate the corresponding vorticity and body force by simple substitution.

In 2D, the codifferential reduces to $\mathrm{d}^* = (-1)^{2(k+1)+1} \star \mathrm{d} \star = (-1)^{2k+1} \star \mathrm{d} \star = - \star \mathrm{d} \star$. And the Hodge star on the basis forms are

$$\begin{aligned}\star 1 &= \mathrm{d}x \wedge \mathrm{d}y \\ \star \mathrm{d}x &= \mathrm{d}y \\ \star \mathrm{d}y &= -\mathrm{d}x \\ \star \mathrm{d}x \wedge \mathrm{d}y &= 1\end{aligned}$$

Substituting into the stokes system, and using subscript to indicate partial derivatives, we get

$$\begin{aligned}f^1 &= -\mathrm{d} \star \mathrm{d}(u^x \mathrm{d}y - u^u \mathrm{d}x) + \star \mathrm{d}p \\ &= -\mathrm{d} \star (u^x_x \mathrm{d}x \wedge \mathrm{d}y - u^y_y \mathrm{d}y \wedge \mathrm{d}x) - \star(p_x \mathrm{d}x + p_y \mathrm{d}y) \\ &= -d(u^x_x + u^y_y) + p_x \mathrm{d}y - p_y \mathrm{d}x \\ &= -u^x_{xx}\mathrm{d}x - u^x_{xy}\mathrm{d}y - u^y_{yx}\mathrm{d}x - u^y_{yy}\mathrm{d}y + p_x \mathrm{d}y - p_y \mathrm{d}x \\ &= (-u^x_{xx} - u^y_{xy} - p_y)\mathrm{d}x + (-u^x_{xy} - u^y_{yy} + p_x)\mathrm{d}y\end{aligned}$$

Note that we have $\omega^0 = -(u^x_x + u^y_y)$ as an intermediate result.

■

[. Lid-driven cavity flow] content... ■

[. Manufactured Solution]

Specifically for $u = -\cos(2\pi x)\sin(2\pi y)\mathrm{d}x - \sin(2\pi x)\cos(2\pi y)\mathrm{d}y$, we get ■

Conclusion

Equivalence with mixed Galerkin finite-element method
can be extended "easily" for isoGeometric analysis.

Next Steps

curved basis and geometry
compressibility
toy problem with reentry heat flow
triangular mesh
distant: 3D problem, nonlinear problems

References

rene, paper, thesis
sahin owens, lid-cavity driven flow
keenan, dec

Appendix: MATLAB code

```
close all; clc;
%% problem definitions
% number of elements in x-, y- direction
n = 80; m = 80;
S = Surface(n,m);

% number of quadrature points in x-, y- direction
quadu = 2; quadv = 2;

%% lid driven cavity flow
% problem = 'lid driven cavity flow';
% % strong boundary conditions (normal velocity/flux)
% sbc.south = 0;
% sbc.north = 0;
% sbc.west = 0;
% sbc.east = 0;
%
% % weak boundary conditions (tangential velocity) [dy dx]
% ubc.south = @(x,y) [0 0];
% ubc.north = @(x,y) [1 0];
% ubc.east = @(x,y) [0 0];
% ubc.west = @(x,y) [0 0];
%
% % load (body force) f0[1], f1[dy dx], f2[dy^dx]
% f.f0 = 0;
% f.f1 = @(x,y) [0 0];
% f.f2 = 0;

%% mfg. solution
problem = 'mfg. solution';
% strong boundary conditions (normal velocity/flux)
sbc.south = 0;
sbc.north = 0;
sbc.west = 0;
sbc.east = 0;

% weak boundary conditions (tangential velocity) [dy dx]
u = @(x,y) [-sin(2*pi*x)*cos(2*pi*y), cos(2*pi*x)*sin(2*pi*y)];
ubc.south = @(x) u(x,0); ubc.north = @(x) u(x,1);
ubc.west = @(y) u(0,y); ubc.east = @(y) u(1,y);
% ubc.south = @(x) [-sin(2*pi*x) isnan(x)];
% ubc.north = ubc.south;
% ubc.west = @(y) [isnan(y) -sin(2*pi*y)];
% ubc.east = ubc.west;

% load (body force) f0[1], f1[dy dx], f2[dy^dx]
f.f0 = 0;
f.f1 = @(x,y) [-8*pi^2*sin(2*pi*x).*cos(2*pi*y) + pi*cos(pi*x).*sin(pi*y) ...
    -8*pi^2*cos(2*pi*x).*sin(2*pi*y) - pi*sin(pi*x).*cos(pi*y)];
% f.f1 = @(x,y) [0,0];
```

```
f.f2 = 0; % incompressible/divergence free flow

%% computation
% dimension of mesh primitives
a = S.numnodes; b = sum(S.numedges); c = S.numfaces;

% compute mass matrices and load vectors
[M0,M1,M2,F0,F1,F2] = assembly(S,f,quadu,quadv);

% compute discrete differential operator matrices
D10 = curl(S);
D21 = divergence(S);

% compute weak boundary conditions
W1 = weakbcs(S,quadu,quadv,ubc);

% assemble stokes system mass matrix
Mass = [-M0          D10'*M1      sparse(a,c)  ;
         M1*D10      sparse(b,b)  D21'*M2      ;
        sparse(c,a)  M2*D21       sparse(c,c)  ];

% assemble load vector
Load = [ W1 + F0; F1; F2];

% apply strong boundary conditions
% ii = S.getglobalboundaryedges;
% Mass(a + ii,:) = 0.0;
% Load(a + ii) = 0.0;
% for i=1:length(ii), Mass(a+ii(i),a+ii(i)) = 1.0; end
[Mass,Load] = strongbcs(S,Mass,Load,sbc);

% solving the system
xh = Mass\Load;

%% post processing
vort = reshape(xh(1:a),n+1,m+1);
velx = reshape(xh(a+(1:S.numedges(1))),n+1,m);
vely = reshape(xh(a+S.numedges(1)+(1:S.numedges(2))),n,m+1);
pres = reshape(xh(a+b+(1:c)),n,m);

% stream function
Psi = zeros(n+1,m+1);
Psi(:,2:end) = Psi(:,2:end) + cumsum(velx,2);
Psi(2:end,:) = Psi(2:end,:) - cumsum(vely,1);

% refine solution mesh
res = 2e2+1;
[XX,YY,VV,VX,VY,PP,DV] = femsol(S,vort,velx,vely,pres,res);
%% visualization
close all;

figure(); hold all; view(2); title('vorticity \omega')
surf(XX,YY,VV,'edgecolor','none','facecolor','interp')
% contour(XX,YY,VV,100);
```

```
xlabel('x');ylabel('y');colormap jet; colorbar
saveas(gcf,num2str([n,m],['vorticity - ' problem ' on %dby%d grid.png']))

figure(); hold all; view(2); title('x-velocity v_x')
surf(XX,YY,VX,'edgecolor','none','facecolor','interp')
% contour(XX,YY,VX,30);
xlabel('x');ylabel('y');colormap jet; colorbar
saveas(gcf,num2str([n,m],['x velocity - ' problem ' on %dby%d grid.png']))

figure(); hold all; view(2); title('y-velocity v_y')
surf(XX,YY,VY,'edgecolor','none','facecolor','interp')
% contour(XX,YY,VY,30);
xlabel('x');ylabel('y');colormap jet; colorbar
saveas(gcf,num2str([n,m],['y velocity - ' problem ' on %dby%d grid.png']))

figure(); hold all; view(2); title('pressure p')
surf(XX,YY,PP,'edgecolor','none','facecolor','interp')
% contour(XX,YY,PP,30);
xlabel('x');ylabel('y');colormap jet; colorbar
saveas(gcf,num2str([n,m],['pressure - ' problem ' on %dby%d grid.png']))

figure(); hold all; view(2); title('$\nabla\cdot\vec v$','interpreter','latex')
surf(XX,YY,DV,'edgecolor','none','facecolor','interp')
% contour(XX,YY,DV,50);
xlabel('x');ylabel('y');colormap jet; colorbar
saveas(gcf,num2str([n,m],['velocity divergence - ' problem ' on %dby%d grid.png']))

figure(); hold all; view(2); title('Stream function \Psi')
surf(S.xnodes,S.ynodes,Psi,'edgecolor','none','facecolor','interp')
% contour(S.xnodes,S.ynodes,Psi,100);
xlabel('x');ylabel('y');colormap jet; colorbar
saveas(gcf,num2str([n,m],['stream function - ' problem ' on %dby%d grid.png']))

% centerline velocities
figure(); hold all; title('centerline x-velocity')
plot(VX(XX == 0.5),YY(XX == 0.5))
xlabel('v_x(0.5,y)');ylabel('y');
saveas(gcf,num2str([n,m],['centerline x-velocity - ' problem ' on %dby%d grid.png']))

figure(); hold all; title('centerline y-velocity')
plot(XX(YY == 0.5),VY(YY == 0.5))
xlabel('x');ylabel('v_y(x,0.5)');
saveas(gcf,num2str([n,m],['centerline y-velocity - ' problem ' on %dby%d grid.png']))

%% analytic solutions for mfg. sol'n
vv = @(x,y) -4*pi*sin(2*pi*x).*sin(2*pi*y);
vx = @(x,y) -sin(2*pi*x).*cos(2*pi*y);
vy = @(x,y) -cos(2*pi*x).*sin(2*pi*y);
pp = @(x,y) sin(pi*x).*sin(pi*y);

VVa = vv(XX,YY);
VXa = vx(XX,YY);
VYa = vy(XX,YY);
PPa = pp(XX,YY);
```

```
figure(); hold all; view(2); title('vorticity \omega')
% surf(XX,YY,VVa,'edgecolor','none','facecolor','interp')
contour(XX,YY,VVa,100);
colormap jet; colorbar
figure(); hold all; view(2); title('x-velocity v_x')
% surf(XX,YY,VXa,'edgecolor','none','facecolor','interp')
contour(XX,YY,VXa,100);
colormap jet; colorbar
figure(); hold all; view(2); title('y-velocity v_y')
% surf(XX,YY,VYa,'edgecolor','none','facecolor','interp')
contour(XX,YY,VYa,100);
colormap jet; colorbar
figure(); hold all; view(2); title('pressure p')
% surf(XX,YY,PPa,'edgecolor','none','facecolor','interp')
contour(XX,YY,PPa,100);
colormap jet; colorbar

%% L2 errors

[L2vv(n),L2vy(n),L2vx(n),L2pp(n)] = femerr(S,vort,vely,velx,pres,vv,vy,vx,pp,quadu,quadv);
```