Chapter-31-3--Domain-Decomposition

In mathematics, numerical analysis, and numerical partial differential equations, **domain decomposition** methods solve a boundary value problem by splitting it into smaller boundary value problems on subdomains and iterating to coordinate the solution between adjacent subdomains. A coarse problem with one or few unknowns per subdomain is used to further coordinate the solution between the subdomains globally. The problems on the subdomains are independent, which makes domain decomposition methods suitable for parallel computing. Domain decomposition methods are typically used as preconditioners for Krylov space iterative methods, such as the conjugate gradient method, GMRES, and LOBPCG.

In overlapping domain decomposition methods, the subdomains overlap by more than the interface. Overlapping domain decomposition methods include the Schwarz alternating method and the additive Schwarz method. Many domain decomposition methods can be written and analyzed as a special case of the abstract additive Schwarz method.

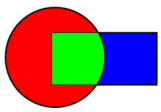
In non-overlapping methods, the subdomains intersect only on their interface. In primal methods, such as Balancing domain decomposition and BDDC, the continuity of the solution across subdomain interface is enforced by representing the value of the solution on all neighboring subdomains by the same unknown. In dual methods, such as FETI, the continuity of the solution across the subdomain interface is enforced by Lagrange multipliers. The FETI-DP method is hybrid between a dual and a primal method.

Non-overlapping domain decomposition methods are also called iterative substructuring methods.

Mortar methods are discretization methods for partial differential equations, which use separate discretization on nonoverlapping subdomains. The meshes on the subdomains do not match on the interface, and the equality of the solution is enforced by Lagrange multipliers, judiciously chosen to preserve the accuracy of the solution. In the engineering practice in the finite element method, continuity of solutions between non-matching subdomains is implemented by multiple-point constraints.

Finite element simulations of moderate size models require solving linear systems with millions of unknowns. Several hours per time step is an average sequential run time, therefore, parallel computing is a necessity. Domain decomposition methods embody large potential for a parallelization of the finite element methods, and serve a basis for distributed, parallel computations.

The picture below is a diagram representing the domain decomposition topology, as described in the above cell. A close analog of the diagram was originally used by Schwarz in 1870.



The Github repository of arielshao contains Matlab code for the DD25_Workshop. The object of the study is the construction of a 3D environmental temperature map of a certain indoor room.The code cell below shows the execution program for Solution3_5.

In [ ]:

```
%This code simulates an insulated wall on the right edge of the domain;
% that is, partial u/ partial n=0 for x=1;

RoomData                                % include problem parameters
f=[zeros(J,1) f zeros(J,1)];            % Robin solvers compute bc
xi=[0 xi 1];                            % thus need to increase size
pe=1e12;                                % to emulate Dirichlet condition
a=8;                                    % interface position
maxiter=200;
f1=f(:,2:a); f2=f(:,a+1:end);           % subdomain source terms
g=zeros(J,1);
```

```
x1=(0:h:a*h); x2=(a*h:h:1);                 % subdomain mesh in x
z1=zeros(1,a+1); z2=zeros(1,J-a+2);         % for plotting
u=Solve2dR(f,eta,0,J+1,gg*pe,gd,pe,0);      % global solve
err(1)=norm(u,'fro');                       % initial error starting with 0
th=0.7;                                     % relaxation parameter
e=ones(J,1);                                % construct normal derivative
Na=[speye(J) -spdiags([-e (eta*h^2+4)*e -e]/2,[-1 0 1],J,J)]/h;
figure(100);clf;
for i=1:maxiter
  u1=Solve2d(f1,eta,0,a,gg,g);              % solve left subdomain
  ta=Na*[u1(:,end-1);u1(:,end)]+f2(:,1)*h/2;% extract interface data
  u2=Solve2dR(f2,eta,a,J+1,ta,gd,0,0);     % solve right subdomain
  g=th*u2(:,1)+(1-th)*g;                    % relax Dirichlet trace
  ufin=[u1(:,1:a),(u1(:,a+1)+u2(:,1))/2,u2(:,2:end)];
  err(i+1)=norm(u-ufin,'fro');
  mesh(x1,y,[z1;u1;z1]); hold on;           % plot the two iterates
  mesh(x2,y,[z2;u2;z2]); hold off;
  xlabel('x');ylabel('y');zlabel('Dirichlet-Neumann iterates');
  pause
end
figure(101);clf;
semilogy(1:maxiter+1,err)                   % plot error decay
xlabel('Iterations');
ylabel('Error');


%% Observations and Analysis

% At iteration 20:

%           Homogeneous Neumann      vs    Homegeneous Dirichlet
%-------------------------------------------------------------------
%  error       1.52e-4                        1.06e-5
% solution  monotonically decreasing
%          to zero on the right domain         like a saddle
%-------------------------------------------------------------------
% The insulated wall case is slow in convergence.
% Graphically, it gives a more resonable simulation of the heating of a
% room.

```

The code cell below shows the execution program for Solution3_6.

In [ ]:

```
%This example solves on 3 subdomains [0, a], [a,3a],[3a,1]
% First solve a Dirichlet problem on the 2nd subdomain
% Next solve a mixed Dirchlet-Neumann problem on 1st and 3rd subdomains
% Then update the interface data and repeat the procedure

RoomData                                    % include problem parameters
f=[zeros(J,1) f zeros(J,1)];                % Robin solvers compute bc
xi=[0 xi 1];                                % thus need to increase size
pe=1e12;                                    % to emulate Dirichlet condition
a=8;                                        % interface position
maxiter=100;
f1=f(:,1:a+1); f2=f(:,a+2:2*a);
f3=f(:,2*a+1:end);                         % subdomain source terms
g1=zeros(J,1);
g2=zeros(J,1);
x1=(0:h:a*h); x2=(a*h:h:2*a*h);
x3=(2*a*h:h:1);                            % subdomain mesh in x
z1=zeros(1,a+1); z2=zeros(1,a+1);
z3=zeros(1,J-2*a+2);                        % for plotting
u=Solve2dR(f,eta,0,J+1,gg*pe,pe*gd,pe,pe);  % global solve
err(1)=norm(u,'fro');                       % initial error starting with 0
th=0.7;                                     % relaxation parameter
e=ones(J,1);                                % construct normal derivative
Na=[speye(J) -spdiags([-e (eta*h^2+4)*e -e]/2,[-1 0 1],J,J)]/h;
Nb=[-spdiags([-e (eta*h^2+4)*e -e]/2,[-1 0 1],J,J) speye(J)]/h;
u1=zeros(J,a+1);
u2=zeros(J,a+1);
u3=zeros(J,J-2*a+1);
figure(3);clf;
for i=1:maxiter
  u2=Solve2d(f2,eta,a,2*a,g1,g2);            % solve dirichlet problem on the 2nd domain
  tb=Nb*[u2(:,1);u2(:,2)]+f1(:,end)*h/2;
  ta=Na*[u2(:,end-1);u2(:,end)]+f3(:,1)*h/2; % extract interface data
```

```
  u1=Solve2dR(f1,eta,0,a,gg*pe,tb,pe, 0);
  u3=Solve2dR(f3,eta,2*a,J+1,ta,gd*pe,0,pe);    % solve dirichlet-neumann problem on 1st and 3rd domain
  g1=th*u1(:,end)+(1-th)*g1;                  % update the interface data
  g2=th*u3(:,1)+(1-th)*g2;
  ufin=[u1(:,1:a),(u1(:,a+1)+u2(:,1))/2,u2(:,2:end-1),(u2(:,end)+u3(:,1))/2, u3(:,2:end)];
  err(i+1)=norm(u-ufin,'fro');
  mesh(x1,y,[z1;u1;z1]); hold on;           % plot the two iterates
  mesh(x2,y,[z2;u2;z2]); hold on;
  mesh(x3,y,[z3;u3;z3]); hold off;
  xlabel('x');ylabel('y');zlabel('Dirichlet-Neumann iterates');
  %pause
end
figure(4);clf;
semilogy(1:maxiter+1,err)                   % plot error decay
xlabel('Iterations');
ylabel('Error');

%% Observations and Analysis

% a=8  th=0.5

%                     2  subdomains      vs             3 subdomains
%-------------------------------------------------------------------------
% # of iterations          14                                32
% error                    3.09e-11                          5e-15


% As the number of subdomains increases, more iterations are needed for
% convergence, but our approximate solution converges to a more accurate
% solution.
```

The plot for Solution3_5, (left), appears to show something awry with the calculation of subdomain function values for the boundary between two adjacent subdomains. The plot for Solution3_6, (right), shows smooth joins between its subdomains.