```
1…function sol = bvp5c(ode, bc, solinit, options)

% Check input arguments
% Adjust status of warnings
% Validate arguments and options
[neqn,nparam,nregions,atol,rtol,Nmax,xyVectorized,printstats] = ...
    bvparguments(solver_name,ode,bc,solinit,options);

% Modify equations to accommodate unknown parameters

% Deal with a singular BVP.
[ode,bc,jac,bcjac,Joptions,dBCoptions] = ...
    bvpfunctions(solver_name,ode,bc,options,neqn,nparam,nregions);

141    bvpfunctions(solver_name,ode,bc,options,neqn,nparam,nregions);

% Deal with a singular BVP.
[singularBVP,ode,jac,solinit,PBC] = ...
    bvpsingular(solver_name,solinit,ode,jac,options,neqn,nparam,nregions);

151  % Adjust the problem to accommodate unknown parameters

164 % Four-stage Lobatto IIIa collocation formula (non-trivial coefficients
only.)

175    % Constant matrices for the collocation Jacobian

182 % Interpolate solution at collocation points
[X,Y] = interpGuess(solinit);

188 %% Algebraic solver parameters
maxNewtIter = 4;
maxProbes = 4;     % weak line search
needGlobalJacobian = true;
refinedMesh = false;
203 % THE MAIN LOOP:




%--------------------------------------------------------------------------
370 % Nested functions
%--------------------------------------------------------------------------
  function [X,Y] = interpGuess(sol)
  %INTERP_GUESS  Evaluate/interpolate the initial guess at collocation
points.


396: end  % interpGuess
%--------------------------------------------------------------------------

  function [X,Y] = interpGuess_region(sol,sol_xreg,sol_yreg,region)
  % In a region, evaluate/interpolate the initial guess at collocation
points.

450... end  % interpGuess_region
  %-------------------------------------------------------------------------

454  function F = odeFcn(X,Y)
  %ODE_FCN  Evaluate the ODE function for all points in X,Y.

467  end  % odeFcn

  %-------------------------------------------------------------------------

471   function F = odeFcn_region(X,Y,region)
  % In a region, evaluate the ODE function for all points in X,Y.

486   end  % odeFcn_region

  %-------------------------------------------------------------------------

490  function [Jn,Jnc1,Jnc2,Jnp1] = odeJac_region(x,y,f,Jpropagated,region)
  % In a region, compute the ODE Jacobian at points in a mesh interval.
528  end  % odeJac_region

  %-------------------------------------------------------------------------

532  function res = bcaux(Ya,Yb)
  %BCAUX  Reshape the arguments for the BC function.
537   end  % bcaux
  %-------------------------------------------------------------------------

541   function [dBCdya,dBCdyb] = bcJac(ya,yb,bcVal)
  %BC_JAC  Compute the BC Jacobian.
560   end  % bcJac

  %-------------------------------------------------------------------------

  function [RHS,RHSbc]= colloc_RHS(X,Y,F)
  %COLLOC_RHS  Evaluate the system of collocation equations.
  %   Separately return the residual in the boundary conditions.
590   end  % colloc_RHS

  %-------------------------------------------------------------------------

594   function RHSode = colloc_RHS_region(X,Y,F)
  % In a region, evaluate the system of collocation equations.
616   end  % colloc_RHS_region

  %-------------------------------------------------------------------------

620  function [Jac,doSeparateBCs] = colloc_Jac(X,Y,F,bcVal)
  %COLLOC_JAC  Form the global Jacobian of the collocation equations.
723  end  % colloc_Jac
```

```
  %--------------------------------------------------------------------------

      function [JacI,JacJ,JacV] = colloc_JacODE_region(X,Y,F,region)
  % In a region, form the Jacobian of collocation equations.
       end  % colloc_JacODE_region

  %--------------------------------------------------------------------------

      function maxInterpResidual = colloc_maxresidual(X,Y,F)
  %COLLOC_MAXRESIDUAL  Compute max residual in the collocation equations.
     end  % colloc_maxresidual

  %--------------------------------------------------------------------------

     function maxInterpResidual = colloc_maxresidual_region(X,Y,F)
  % In a region, compute max residual in the collocation equations.
      end  % colloc_maxresidual_region

  %--------------------------------------------------------------------------

     function ymid = interpolateYmid(X,Y,F)
  %INTERPOLATE_YMID  Interpolate Y at the midpoints of mesh subintervals.
      end  % interpolateYmid

  %--------------------------------------------------------------------------

      function ymid = interpolateYmid_region(X,Y,F)
  % In a region, interpolate Y at the midpoints of mesh subintervals.
    h = diff(X(1:nstages:end));
      end  % interpolateYmid_region

  %--------------------------------------------------------------------------

      function res = residualEstimate(X,Y,Ymid,F,nsamples)
  %RESIDUAL_ESTIMATE  Estimate the residual in each mesh subinterval.
     end  % residualEstimate

  %--------------------------------------------------------------------------

      function res = residualEstimate_region(X,Y,ymid,F,nsamples,region)
  % In a region, estimate the residual in each mesh subinterval.
      end  % residualEstimate_region

  %--------------------------------------------------------------------------

     function [XX,YY,FF] = newSolutionProfile(X,Y,Ymid,F,errEst)
  %NEW_SOLUTION_PROFILE  Redistribute mesh points and approximate the
solution.

    % Detect mesh oscillations: Was there a mesh with
    % the same number of nodes and a similar residual?
       % If so, only allow for adding mesh points.
% modify the mesh, interpolate the solution
    [xreg,yreg,freg,ymidreg,errEstReg] = getRegionData(1,X,Y,F,Ymid,errEst);
    [XX,YY,FF] =
newSolutionProfile_region(xreg,yreg,ymidreg,freg,errEstReg,...
                                          {1},canRemovePoints);
      end  % newSolutionProfile

  %--------------------------------------------------------------------------

      function [XX,YY,FF] =
newSolutionProfile_region(X,Y,ymid,F,errEst,region,...
                                                 canRemovePoints)% In a
region, redistribute mesh points and approximate the solution.
       end  % newSolutionProfile_region

  %--------------------------------------------------------------------------

      function sol = outputSol(X,Y,F,ymid)
  %OUTPUT_SOL  Assembly the solution structure.
       end  % outputSol

  %--------------------------------------------------------------------------

      function [xout,yout,ypout,ymidout] = outputData(x,y,f,ymid)
  %OUTPUT_DATA  Prepare data for the solution structure.
      end  % outputData

  %--------------------------------------------------------------------------

       function MBVP = updateMBVP(X)
  %UPDATE_MBVP  Update indices of the internal boundary points for MBVPs.
end  % updateMBVP

  %--------------------------------------------------------------------------

      function [xreg,yreg,freg,ymidreg,errest] =
getRegionData(region,X,Y,F,Ymid,ErrEst)
  %GET_REGION_DATA  Extract mesh points and solution data for a given region.
       end  % getRegionData

%--------------------------------------------------------------------------

    end  % bvp5c
```