```
function x = IRK(ButcherArray, f, dfdx, T, x0)
    % Returns the iterations of an IRK method using Newton's method
    % ButcherArray: Struct with the IRK's Butcher array
    %   A: Nstage x Nstage
    %   b: Nstage x 1
    %   c: Nstage x 1
    % f: Function handle
    %    Vector field of ODE, i.e., x_dot = f(t,x)
    % dfdx: Function handle
    %       Jacobian of f w.r.t. x
    % T: Vector of time points, 1 x Nt
    % x0: Initial state, Nx x 1
    %
    % x: IRK iterations, Nx x Nt
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Define variables
    % Allocate space for iterations (x) and k1,k2,...,ks
    A = ButcherArray.A;
    b = ButcherArray.b;
    c = ButcherArray.c;

    Nx = size(x0,1);
    Nt = size(T,2);
    Nstage = size(A,1);

    x = zeros(Nx, Nt);
    x(:,1) = x0;

    % K: Nx x Nstage matrix of Ki's
    % k: Nx*Nstage x 1 column vector of Ki
    K = zeros(Nx, Nstage);
    K = repmat(x0, 1, Nstage);

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    xt = x0; % initial iteration
    k = reshape(K,Nx*Nstage,1); % initial guess
    % Loop over time points
    for nt=2:Nt
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        % Update variables
        % Get the residual function for this time step
        % and its Jacobian by defining adequate functions
        % handles based on the functions below.
        % Solve for k1,k2,...,ks using Newton's method
        % Calculate and save next iteration value x_t
        dt = T(nt) - T(nt-1);
        t = T(nt);

        r  = @(k) IRKODEResidual(k,xt,t,dt,A,c,f);
        dr = @(k) IRKODEJacobianResidual(k,xt,t,dt,A,c,dfdx);

        k = NewtonsMethod(r,dr,k);
        K = reshape(k,Nx,Nstage);

        xt = xt + dt*K*b;
        x(:,nt) = xt;
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    end
```

```
59 end
60
61 function g = IRKODEResidual(k,xt,t,dt,A,c,f)
62     % Returns the residual function for the IRK scheme iteration
63     % k: Column vector with k1,...,ks, Nstage*Nx x 1
64     % xt: Current iteration, Nx x 1
65     % t: Current time
66     % dt: Time step to next iteration
67     % A: A matrix of Butcher table, Nstage x Nstage
68     % c: c matrix of Butcher table, Nstage x 1
69     % f: Function handle for ODE vector field
70     Nx = length(xt);
71     Nstage = size(A,1);
72     K = reshape(k,Nx,Nstage);
73     Tg = t+dt*c';
74     Xg = xt+dt*K*A';
75     g = reshape(K-f(Tg,Xg),[],1);
76 end
77
78 function G = IRKODEJacobianResidual(k,xt,t,dt,A,c,dfdx)
79     % Returns the Jacobian of the residual function
80     % for the IRK scheme iteration
81     % k: Column vector with k1,...,ks, Nstage*Nx x 1
82     % xt: Current iteration, Nx x 1
83     % t: Current time
84     % dt: Time step to next iteration
85     % A: A matrix of Butcher table, Nstage x Nstage
86     % c: c matrix of Butcher table, Nstage x 1
87     % dfdx: Function handle for Jacobian of ODE vector field
88     Nx = length(xt);
89     Nstage = size(A,1);
90     K = reshape(k,Nx,Nstage);
91     TG = t+dt*c';
92     XG = xt+dt*K*A';
93     dfdxG = cell2mat(arrayfun(@(i) dfdx(TG(:,i),XG(:,i))',1:Nstage,...
94         'UniformOutput',false))';
95     G = eye(Nx*Nstage)-repmat(dfdxG,1,Nstage).*kron(dt*A,ones(Nx));
96 end
```