```
function [x,xdot,z] = RKDAE(ButcherArray, F, dFdxdot, dFdx, dFdz, T,
 x0, z0_est)
    % Returns the iterations of a RK method using Newton's method
    % ButcherArray: Struct with the RK's Butcher array
    % F: Function handle.
    %    Implicit function for DAE: F(xdot,x,z,t)=0
    % dFdxdot, dFdx, dFdz: Function handles
    %                       Jacobians of f w.r.t. x_dot,x,z,
 respectively
    % T: Vector of time points
    % x0: Initial state
    % z0_est: Estimate of algebraic variables at initial time
    % x: RK iterations for x, Nx x Nt
    % xdot, z: xdot and z values for x and T, Nx x Nt

    % Definitions and allocations
    Nt = length(T);
    Nx = length(x0);
    Nz = length(z0_est);
    dT = diff(T);
    x = zeros(Nx,Nt);
    xdot = zeros(Nx,Nt);
    z = zeros(Nz,Nt);
    A = ButcherArray.A;
    b = ButcherArray.b(:);
    c = ButcherArray.c(:);
    Nstage = size(A,1);
    % Make sure that the vector function handles can act on
    % concatenations of vectors
    FVec = @(xdot,x,z,t) ExpandFunction2Concatenations(F,xdot,x,z,t);
    dFdxdotVec = @(xdot,x,z,t)
 ExpandFunction2Concatenations(dFdxdot,xdot,x,z,t);
    dFdxVec = @(xdot,x,z,t)
 ExpandFunction2Concatenations(dFdx,xdot,x,z,t);
    dFdzVec = @(xdot,x,z,t)
 ExpandFunction2Concatenations(dFdz,xdot,x,z,t);
    % Start integration
    [xdot0,z0] =
 SolveForXdotAndZGivenX(x0,T(1),FVec,dFdxdotVec,dFdzVec,zeros(Nx,1),z0_est);
    x(:,1) = x0;
    xdot(:,1) = xdot0;
    z(:,1) = z0;
    xt = x0;
    xdott = xdot0;
    zt = z0;
    w = [repmat(xdott,Nstage,1); repmat(zt,Nstage,1)]; % initial guess
    % Integrate
    for nt=2:Nt
        t = T(nt-1);
        dt = dT(nt-1);
        G = @(w) RKDAEResidual(w,xt,t,dt,A,c,FVec);
        JG = @(w)
 RKDAEJacobianResidual(w,xt,t,dt,A,c,dFdxdotVec,dFdxVec,dFdzVec);
        w = NewtonsMethod(G,JG,w);
        K = reshape(w(1:Nx*Nstage),Nx,Nstage);
        xt = xt + dt*(K*b);
        x(:,nt) = xt;
        [xdott,zt] = SolveForXdotAndZGivenX(xt,t
+dt,FVec,dFdxdotVec,dFdzVec,xdott,zt);
        xdot(:,nt) = xdott;
        z(:,nt) = zt;
    end
end
function [xdot,z] =
 SolveForXdotAndZGivenX(x,t,F,dFdxdot,dFdz,xdotest,zest)
    % Given x and t, returns xdot and z value such that
 F(xdot,x,z,t)=0
    % y = [xdot;z]
    Nx = length(x);
    G = @(y) F(y(1:Nx),x,y(Nx+1:end),t);
    JG = @(y) [dFdxdot(y(1:Nx),x,y(Nx+1:end),t) dFdz(y(1:Nx),x,y(Nx
+1:end),t)];
    y = NewtonsMethod(G,JG,[xdotest;zest]);
    xdot = y(1:Nx);
    z = y(Nx+1:end);
end
function g = RKDAEResidual(w,xt,t,dt,A,c,F)
    % Returns the residual function for the RK scheme iteration
    % w = [K1;K2;...;Knstages;z1;z2;...;znstages];
    Nx = length(xt);
    Nstage = size(A,1);
    K = reshape(w(1:Nx*Nstage),Nx,Nstage);
    Z = reshape(w(Nx*Nstage+1:end),[],Nstage);
    Tg = t+dt*c';
    Xg = xt+dt*K*A';
    g = reshape(F(K,Xg,Z,Tg),[],1);
end
function G = RKDAEJacobianResidual(w,xt,t,dt,A,c,dFdxdot,dFdx,dFdz)
    % Returns the Jacobian of the residual function
    % for the RK scheme iteration
    % w = [K1;K2;...;Knstages;z1;z2;...;znstages];
    Nx = length(xt);
    Nstage = size(A,1);
    K = reshape(w(1:Nx*Nstage),Nx,Nstage);
    Z = reshape(w(Nx*Nstage+1:end),[],Nstage);
    Nz = size(Z,1);
    TG = t+dt*c';
    XG = xt+dt*K*A';
    dFdxdotG = cell2mat(arrayfun(@(i)
 dFdxdot(K(:,i),XG(:,i),Z(:,i),TG(:,i))',...
        1:Nstage,'UniformOutput',false))';
    dFdxG = cell2mat(arrayfun(@(i)
 dFdx(K(:,i),XG(:,i),Z(:,i),TG(:,i))',...
        1:Nstage,'UniformOutput',false))';
    dFdzG = cell2mat(arrayfun(@(i)
 dFdz(K(:,i),XG(:,i),Z(:,i),TG(:,i))',...
        1:Nstage,'UniformOutput',false))';
    G = [repmat(dFdxdotG,1,Nstage).*kron(eye(Nstage),ones(Nz
+Nx,Nx)) ...
         + repmat(dFdxG,1,Nstage).*kron(dt*A,ones(Nz+Nx,Nx)) ...
         repmat(dFdzG,1,Nstage).*kron(eye(Nstage),ones(Nx+Nz,Nz))];
end
function fVec = ExpandFunction2Concatenations(f,xdot,x,z,t)
    % Returns the concatenation [f(xdot(:,i),x(:,i),z(:,i),t(i)):
 i=1...N]
    % f, function handle that returns column vector
    % xdot, matrix [] x N
    % x, matrix [] x N
    % z, matrix [] x N
    % t, matrix 1 x N
    N = size(t,2);
    fVec = cell2mat(arrayfun(@(i) f(xdot(:,i),x(:,i),z(:,i),t(i))',...
        1:N,'UniformOutput',false))';
end
```

```
Not enough input arguments.

Error in RKDAE (line 15)
    Nt = length(T);
```

Published with MATLAB® R2019a