AIAA JOURNAL
Vol. 47, No. 6, June 2009

Sensitivity Derivatives for Plasma Discharge Simulations

Kyle J. Lange* and W. Kyle Anderson†
University of Tennessee at Chattanooga, Chattanooga, Tennessee 37403

DOI: 10.2514/1.41730

The problem of applying sensitivity analysis to plasma discharge simulations is considered. A fluid model of a low-pressure radio frequency helium discharge is used to demonstrate how sensitivity derivatives can be computed for an unsteady periodic solution. The derivations of forward mode direct differentiation and the reverse mode adjoint method are presented. Good agreement is shown between sensitivity derivatives computed by these methods and those computed by finite differences. It is then demonstrated how sensitivity derivatives can be used as part of a design cycle to increase or decrease a given cost function. Finally, it is demonstrated how sensitivity derivatives can be used to compute error bounds for a given cost function.

Nomenclature

$\tilde{\mathbf{A}}$	=	Roe matrix for ion equations
C_1	=	first cost function, total average voltage in the discharge
C_2	=	second cost function, average peak electron density
C_3	=	third cost function, electron power deposition
C_4	=	fourth cost function, ion power deposition
C_5	=	fifth cost function, total electron production
C_6	=	sixth cost function, ionization power losses
C_7	=	seventh cost function, elastic collision power losses
C_8	=	eighth cost function, charge exchange collision power losses
E	=	electric field
E_a	=	activation energy
$\mathbf{F}$	=	flux vector
f	=	frequency of driving voltage
h_{iz}	=	electron energy loss per ionizing collision
I^g	=	global cost function
I^n	=	local cost function computed at each time step
j_e	=	electron flux
k_B	=	Boltzmann constant
k_{i0}	=	ionization preexponential factor
k_{mt}	=	electron momentum transfer coefficient
l	=	discharge gap length
m_e	=	electron mass
m_{He}	=	helium mass
m_i	=	ion mass
N	=	number of time steps in a period for computing I^g
N_{He}	=	neutral helium density
n	=	total number of design variables
n_e	=	electron density
n_i	=	ion density
p	=	discharge pressure
$p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8, p_9$	=	nondimensional coefficient
$\mathbf{Q}$	=	time-dependent variable vector
$\mathbf{Q}$	=	complete variable vector
q_e	=	energy flux
$\mathbf{R}$	=	residual vector
$\mathbf{S}$	=	source term vector
$\tilde{\mathbf{T}}$	=	matrix of right eigenvectors of $\tilde{\mathbf{A}}$
$\tilde{\mathbf{T}}^{-1}$	=	matrix of left eigenvectors of $\tilde{\mathbf{A}}$
T_e	=	electron temperature
T_{gas}	=	neutral gas temperature
T_0	=	reference temperature
V	=	electric potential
V_{max}	=	amplitude of driving voltage
$\bar{v}$	=	Roe-averaged velocity
v_i	=	ion velocity
α	=	optimization coefficient
β	=	design variable
Γ	=	cell surface
Δx	=	distance between two cell centers
θ	=	reflection coefficient
$\mathbf{\Lambda}$	=	first flow adjoint vector
$\tilde{\mathbf{\Lambda}}$	=	diagonal matrix of eigenvalues of $\tilde{\mathbf{A}}$
$\boldsymbol{\lambda}$	=	second flow adjoint vector
λ_D	=	debye length
σ_{cx}	=	charge exchange collision frequency
Ω	=	control volume
ω_e	=	electron frequency

Presented as Paper 5930 at the 12th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, Victoria, British Columbia, Canada, 10–12 September 2008; received 20 October 2008; accepted for publication 18 March 2009.

*Graduate Student, SimCenter: National Center for Computational Engineering; kyle-lange@utc.edu. AIAA Member.

†Professor, SimCenter: National Center for Computational Engineering; kyle-anderson@utc.edu. AIAA Member.

I. Introduction

PLASMA discharges are used in a wide variety of physical applications [1]. For instance, they are used in plasma display panels [2], in the manufacturing of integrated circuits [3], in electrodeless lamps [4], and for ozone generation [5]. To improve the performance of plasma discharges, a better understanding of the underlying physics is needed. Experiments and computational models can both be used to gain this understanding. Experiments, although providing valuable data, can be costly and are limited by the measuring capabilities of the apparatus. Computations can be used in conjunction with experiments to provide more information about the physics.

For instance, one can choose to include or omit a given reaction in a computational simulation. If similar results are obtained when the reaction is included and when it is omitted, it can be determined that the reaction has minimal importance in the discharge. If there is a significant change in results when a reaction is included or not included in the computational model, then the chemical reaction is obviously very important in the plasma.

With a view of obtaining a greater understanding of the physics, many authors have used computational simulations to model the physics of plasma discharges. Many different types of plasma discharges have been modeled, including coplanar-electrode plasma display panel cells [6], low-pressure radio frequency glow discharges [7–9], atmospheric pressure direct current microplasmas [10], and atmospheric pressure radio frequency glow discharges [11]. These simulations generally focus on the physics pertaining to fluids and electric forces or the chemical reactions involved in the discharge. However, although qualitative sensitivity analysis has been performed on plasma discharges [12], to the knowledge of the authors, there has been no dedicated effort to compute sensitivity derivatives.

Sensitivity derivatives may be controllable parameters and can be used to minimize or maximize a given plasma property (e.g., electron power deposition, peak electron density, electron production, etc.). They can also be used to vary experimentally variable quantities such as voltage, frequency, and the shape of vessels to obtain desired characteristics. These characteristics could be composition or power requirements, but could also be the uniformity of the distribution of material over a sample.

Sensitivity derivatives may also be used to assist in determining error bounds of plasma properties. Sensitivities of experimentally variable quantities or physical modeling constants (e.g., reaction rates, cross sections) can be used to determine quantitative error bands for computed quantities.

This paper develops sensitivity analysis methods and describes their use. The time-dependent reverse mode adjoint method and time-dependent forward mode direct differentiation are presented as methods to compute sensitivity derivatives for a low-pressure radio frequency glow discharge simulation. Sensitivity derivatives are used to change the design of the discharge for the purpose of changing a given cost function. It is also demonstrated how uncertainties in the model are computed from the sensitivity derivatives. Although this paper presents sensitivity analysis that is applied to an existing model, it is applicable to other models as well.

The paper is organized as follows. Section II presents the fundamental equations and boundary conditions used in the model. Section III discusses the discretization and implicit time advancement. Section IV discusses the method for computing time-dependent sensitivity derivatives and uncertainties. Section V presents numerical results from the simulation along with uncertainty estimates and sensitivity derivatives. The paper is concluded in Sec. VI.

II. Equations and Boundary Conditions

A. Governing Equations

Fluid models of plasma discharges contain continuity, momentum, and energy equations for each species. For this simulation, three species are considered: electrons, neutral helium atoms, and positive helium ions. The neutral helium atoms are assumed to be in thermal equilibrium with a constant density. The helium atom density is computed from the pressure and gas temperature using the ideal gas law, and thus no equations are used for the neutral helium atoms. The equations in this model are very similar to those used in previous papers [8,9].

Time-dependent partial differential equations are used to solve for the ion density, electron density, ion momentum, and the electron energy. In addition, Poisson's equation for the potential is solved at each time step. The drift-diffusion approximation is used to explicitly compute the electron momentum.

The electron density equation contains a production term that comes from the ionization of helium atoms by electrons. Because the electron frequency is much higher than the radio frequency applied to the gas, a drift-diffusion approximation is used in the electron momentum equation to solve for the electron flux. The drift-diffusion approximation assumes that the forces due to the electric field are balanced by diffusion and collisions with neutral particles. The electron energy equation includes a joule heating term, energy losses due to ionizing collisions, and energy losses from nonionizing collisions between electrons and neutral helium atoms.

The continuity and momentum equations used for the ions are cast in conservative form to use characteristic-based methods to compute the ion fluxes. In addition, the ion pressure term, which has been previously neglected in fluid simulations, is included in the ion momentum flux. Although small in magnitude, including the ion pressure term allows for the formulation of distinct eigenvectors, which allows characteristic-based schemes to be implemented [13]. The ions are assumed to be in thermal equilibrium with the neutral helium atoms, and thus no energy equation is used for the ions.

Poisson's equation is used to solve for the potential, from which the electric field is computed. It is assumed that there are no external magnetic fields.

The full set of equations is given here:

$$\frac{\partial n_e}{\partial t} + \frac{\partial j_e}{\partial x} = p_1 n_e e^{-p_2/T_e} \tag{1}$$

$$\frac{\partial n_i}{\partial t} + \frac{\partial}{\partial x}(n_i v_i) = p_1 n_e e^{-p_2/T_e} \tag{2}$$

$$j_e = -p_7\left(n_e E - \frac{\partial}{\partial x}(n_e T_e)\right) \tag{3}$$

$$\frac{\partial}{\partial t}(n_i v_i) + \frac{\partial}{\partial x}(n_i v_i v_i + p_9 n_i) = p_1 v_i n_e e^{-p_2/T_e} + p_3 n_i E - p_4 n_i |v_i| v_i \tag{4}$$

$$\frac{\partial}{\partial t}\left(\frac{3}{2} n_e T_e\right) + \frac{\partial q_e}{\partial x} = -j_e E - p_5 n_e e^{-p_2/T_e} - p_6 n_e (T_e - T_{\text{gas}}) \tag{5}$$

$$q_e = \frac{5}{2} j_e T_e - p_8 n_e T_e \frac{\partial T_e}{\partial x} \tag{6}$$

$$\frac{\partial^2 V}{\partial x^2} = -(n_i - n_e) \tag{7}$$

$$E = -\frac{\partial V}{\partial x} \tag{8}$$

$$p_1 = N_{\text{He}} k_{i0}/\omega_e \qquad p_2 = E_a/k_B T_0$$
$$p_3 = m_e/m_i \qquad p_4 = \frac{\pi}{2} N_{\text{He}} \sigma_{\text{cx}} \lambda_D$$
$$p_5 = h_{iz} N_{\text{He}} k_{i0}/k_B T_0 \omega_e \qquad p_6 = 3 m_e N_{\text{He}} k_{\text{mt}}/m_{\text{He}} \omega_e$$
$$p_7 = \omega_e/k_{\text{mt}} N_{\text{He}} \qquad p_8 = \frac{5}{2} p_7 \qquad p_9 = \frac{T_{\text{gas}}}{T_0}\frac{m_e}{m_i}$$

The equations are nondimensionalized in a similar to way to those of a previous work [9]. The values of the parameters used in the simulation are the same as those used in previous simulations [8,9] and are shown in Table 1.

Table 1 Parameters of RF helium discharge

Parameter	Symbol	Value
Amplitude of driving voltage	$V_{\max}$	500 V
Frequency of driving voltage	f	12 MHz
Length of discharge gap	l	4 cm
Neutral gas temperature	T_{gas}	300 K
Pressure of discharge	p	0.25 torr
Reflection coefficient	θ	0.25
Electron momentum transfer coefficient	k_{mt}	3.0×10^{-14} m^3/s
Ion charge exchange cross section	σ_{cx}	3.0×10^{-19} m^2
Electron energy loss per ionizing collision	h_{iz}	24.6 eV
Activation energy	E_a	25.1 eV
Ionization preexponential factor	k_{i0}	9.75×10^{-15} m^3/s

Bringing all of the equations together allows one to express the equations in conservative form as

$$\frac{\partial \mathbf{Q}}{\partial t} + \nabla \cdot \mathbf{F} = \mathbf{S} \tag{9}$$

$$\mathbf{Q} = \begin{pmatrix} n_e \\ n_i \\ n_i v_i \\ \frac{3}{2} n_e k T_e \\ 0 \end{pmatrix}, \qquad \mathbf{F} = \begin{pmatrix} j_e \\ n_i v_i \\ n_i v_i^2 + p_9 m_i \\ q_e \\ \nabla V \end{pmatrix} \tag{10}$$

$$\mathbf{S} = \begin{pmatrix} p_1 n_e e^{-p_2/T_e} \\ p_1 n_e e^{-p_2/T_e} \\ p_1 v_i n_e e^{-p_2/T_e} + p_3 n_i E - p_4 n_i |v_i| v_i \\ -j_e E - p_5 n_e e^{-p_2/T_e} - p_6 n_e (T_e - T_{\text{gas}}) \\ -(n_i - n_e) \end{pmatrix} \tag{11}$$

B. Boundary Conditions

The boundary conditions used in the simulation are the same as those used in previous simulations [8,9]. The electron flux at the boundaries is designated as the thermal electron flux. A percentage of the incident electrons at the electrode are reflected back into the discharge, as represented by the reflection coefficient θ. Thus, the electron flux boundary conditions are expressed in nondimensional form as

$$j_e = -\frac{1}{\sqrt{2\pi}}(1-\theta) n_e \sqrt{T_e} \quad \text{at } x = 0 \tag{12}$$

$$j_e = \frac{1}{\sqrt{2\pi}}(1-\theta) n_e \sqrt{T_e} \quad \text{at } x = L \tag{13}$$

The electron energy flux is assumed to be entirely convective, and thus is expressed in nondimensional form as

$$q_e = \tfrac{5}{2} j_e T_e \quad \text{at } x = 0, L \tag{14}$$

It is assumed that the ion density and ion momentum do not change spatially near the boundaries. Thus, the nondimensional ion flux and continuity boundary conditions are

$$\frac{\partial n_i}{\partial x} = 0 \quad \text{at } x = 0, L, \qquad \frac{\partial}{\partial x}(n_i v_i) = 0 \quad \text{at } x = 0, L \tag{15}$$

The potential at the left-hand boundary is set to a sinusoidally varying value, whereas the potential at the right boundary is grounded. The nondimensional potential boundary conditions are expressed as

$$V = V_{\max} \sin 2\pi f t \quad \text{at } x = 0, \qquad V = 0 \quad \text{at } x = L \tag{16}$$

III. Numerical Method

A. Mesh Discretization

The domain is discretized into control volumes by distributing mesh points across the domain according to an algorithm that clusters the points close to the boundaries. The faces of the control volumes are designated to be halfway in between the mesh points except for at the boundaries, where the mesh points are coincident with the cell faces. The mesh setup is shown in Fig. 1.

The spatial discretization is represented with subscripts by the index k. Integer values such as k and $k+1$ denote a value within a given cell. Half-integer values such as $k+\frac{1}{2}$ and $k-\frac{1}{2}$ denote the values at cell faces. The distance between points k and $k+1$ is denoted as Δx_k, whereas the distance between faces $k+\frac{1}{2}$ and $k-\frac{1}{2}$, the cell volume, is represented by Ω_k.

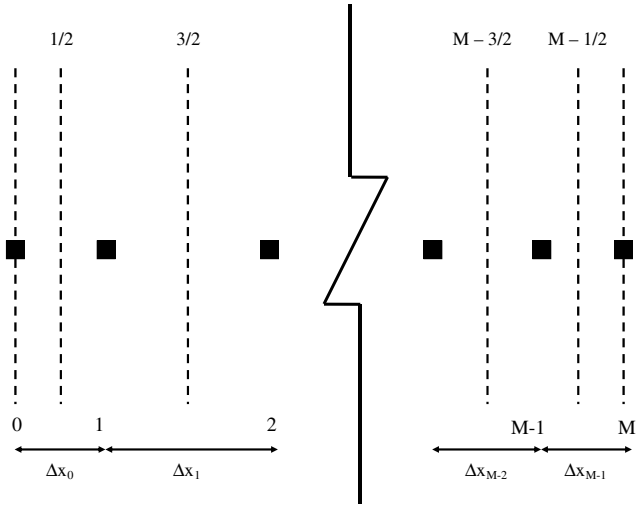

Fig. 1 Discretization of domain into M control volumes with $M+1$ grid points.

B. Spatial Discretization

In contrast to previous plasma discharge simulations that use finite differences or the finite elements to spatially discretize the equations [8,9,14], the equations are discretized in space using a finite volume scheme. Green's theorem is used to convert the volume integrals involving convective terms into surface integrals. The surface integrals are then discretized based on the mesh structure. For the governing equations in a given control volume Ω_k, this results in

$$\iiint_{\Omega_k} \left(\frac{\partial \mathbf{Q}}{\partial t} + \nabla \cdot \mathbf{F} \right) \mathrm{d}\Omega = \iiint_{\Omega_k} \mathbf{S}\, \mathrm{d}\Omega \qquad \Omega_k \frac{\partial \mathbf{Q}_k}{\partial t} + \iint_{\Gamma_k} \mathbf{F} \cdot \hat{\mathbf{n}} = \Omega_k \mathbf{S}_k \qquad \Omega_k \frac{\partial \mathbf{Q}_k}{\partial t} + \sum_{\Gamma_k} \mathbf{F} \cdot \hat{\mathbf{n}} = \Omega_k \mathbf{S}_k \tag{17}$$

Because the simulation is one-dimensional, the sum of the fluxes results in the sum of the fluxes at the cell faces at each end of the cell. The normal vector at each cell is defined as outward pointing, resulting in

$$\Omega_k \frac{\partial \mathbf{Q}_k}{\partial t} + \mathbf{F}_{k+\frac{1}{2}} - \mathbf{F}_{k-\frac{1}{2}} = \Omega_k \mathbf{S}_k \tag{18}$$

As discussed in Sec. III.E, the solution at each time step is updated using Newton's method, where the logarithms of the electron density and the electron energy are used as fundamental variables. By solving for the logarithms of these variables, the electron density and energy will always remain positive, because they are computed by exponentiating the fundamental variables.

In addition, solving for the logarithms allows for a robust second-order computation of the electron flux. In this numerical scheme, upwind-biased methods are used to compute electron fluxes at the cell faces. If one attempts to extrapolate the electron density values to the cell faces, it requires a very fine mesh to ensure that the density is never extrapolated to a negative value. However, this is not a problem when extrapolating the logarithms of the electron density. Thus, in the following equations, $\tilde{Q}_1 = \log(n_e)$ and $\tilde{Q}_2 = \log(\frac{3}{2} n_e T_e)$.

The electron continuity equation is discretized as

$$\frac{\partial e^{\tilde{Q}_1,k}}{\partial t} = -(j_{e,k+\frac{1}{2}} - j_{e,k-\frac{1}{2}}) + \Omega_k p_1 n_{e,k} e^{-p_2/T_{e,k}} \tag{19}$$

In Eq. (19), extrapolated logarithms of electron densities are used to compute the electron flux terms. Electron densities computed using values extrapolated from the left side of the face are denoted as $n_{e,L}$, whereas those computed using values extrapolated from the

right side of the face are denoted as $n_{e,R}$. The electron flux equation is discretized as

$$j_{e,k+\frac{1}{2}} = \begin{cases} -p_7\left[n_{e,L}E_{k+1/2} + \frac{n_eT_{e,k+1}-n_eT_{e,k}}{\Delta x_k}\right], & \text{if } E_{k+1/2} < 0 \\ -p_7\left[n_{e,R}E_{k+1/2} + \frac{n_eT_{e,k+1}-n_eT_{e,k}}{\Delta x_k}\right], & \text{if } E_{k+1/2} > 0 \end{cases} \tag{20}$$

where

$$E_{k+1/2} = -\left(\frac{V_{k+1} - V_k}{\Delta x_k}\right) \tag{21}$$

$$n_{e,L} = \exp\left[\tilde{Q}_{1,k} + \frac{\Delta x}{2}\left(\frac{\partial \tilde{Q}_1}{\partial x}\right)_k\right]$$
$$n_{e,R} = \exp\left[\tilde{Q}_{1,k+1} - \frac{\Delta x}{2}\left(\frac{\partial \tilde{Q}_1}{\partial x}\right)_{k+1}\right] \tag{22}$$

The electron energy equation is discretized as

$$\frac{\partial}{\partial t}(e^{\tilde{Q}_{2,k}}) = -(q_{e,k+1/2} - q_{e,k-1/2}) - \tfrac{1}{2}\Delta x_{k-1} j_{e,k-1/2}E_{k-1/2} + -\tfrac{1}{2}\Delta x_k j_{e,k+1/2}E_{k+1/2} - \Omega_k[p_5 n_{e,k}e^{-p_2/T_{e,k}} - p_6 n_{e,k}(T_{e,k} - T_{\text{gas}})] \tag{23}$$

where the energy flux is discretized as

$$q_{e,k+1/2} = \tfrac{5}{2} j_{e,k+1/2}T_{e,k+1/2} - p_8\left(\frac{n_eT_{e,k+1} + n_eT_{e,k}}{2}\right)\left(\frac{T_{e,k+1} - T_{e,k}}{\Delta x}\right) \tag{24}$$

and

$$T_{e,k+1/2} = \left(\frac{T_{e,k+1} + T_{e,k}}{2}\right) \tag{25}$$

The ion continuity and momentum equations are discretized as follows:

$$\frac{\partial n_{i,k}}{\partial t} = -[(n_iv_i)_{k+1/2} - (n_iv_i)_{k-1/2}] + \Omega_k p_1 n_{e,k}e^{-p_2/T_{e,k}} \tag{26}$$

$$\frac{\partial}{\partial t}(n_iv_i)_k = -[(n_iv_iv_i + p_9n_i)_{k+1/2} - (n_iv_iv_i + p_9n_i)_{k-1/2}] + \Omega_k(p_1 v_{i,k}n_{e,k}e^{-E_a/kT_{e,k}} + p_3 n_{i,k}E_{i,k} - p_4 n_{i,k}|v_{i,k}|v_{i,k}) \tag{27}$$

The ion continuity and momentum fluxes are discretized using a Roe scheme [15]. The Roe matrix $\tilde{\mathbf{A}}$ is expressed as $\tilde{\mathbf{T}}|\tilde{\mathbf{\Lambda}}|\tilde{\mathbf{T}}^{-1}$. Each matrix is computed using a Roe-averaged velocity $\bar{v}$, where

$$\bar{v} = \frac{\sqrt{n_{i,L}}v_{i,L} + \sqrt{n_{i,R}}v_{i,R}}{\sqrt{n_{i,L}} + \sqrt{n_{i,R}}} \tag{28}$$

The ion density and ion momentum terms are computed using extrapolated values at the cell faces. Using extrapolated values of the ion density and the ion momentum for the ion fluxes eliminates numerical jumps induced by using values that are not extrapolated [16]. It also allows for second-order spatial accuracy in the solution [13]. Densities and velocities extrapolated from the left are denoted by $n_{i,L}$ and $v_{i,L}$, whereas densities and velocities extrapolated from the right are denoted as $n_{i,R}$ and $v_{i,R}$. Using $\bar{v}$ to compute the Roe matrix gives the exact jump across a discontinuity. Thus, the ion continuity and momentum fluxes are given as

$$\begin{pmatrix} (n_iv_i)_{k+\frac{1}{2}} \\ (n_iv_iv_i + p_9n_i)_{k+\frac{1}{2}} \end{pmatrix} = \frac{1}{2}\begin{pmatrix} (n_iv_i)_L + (n_iv_i)_R \\ (n_iv_iv_i)_L + p_9n_{i,L} + (n_iv_iv_i)_R + p_9n_{i,R} \end{pmatrix} - \frac{1}{2}\tilde{\mathbf{T}}|\tilde{\mathbf{\Lambda}}|\tilde{\mathbf{T}}^{-1}\begin{pmatrix} n_{i,R} - n_{i,L} \\ (n_iv_i)_R - (n_iv_i)_L \end{pmatrix} \tag{29}$$

$$\tilde{\mathbf{\Lambda}} = \begin{pmatrix} \bar{v} + \sqrt{p_9} & 0 \\ 0 & \bar{v} - \sqrt{p_9} \end{pmatrix} \tag{30}$$

$$\tilde{\mathbf{T}} = \begin{pmatrix} 1 & 1 \\ \bar{v} + \sqrt{p_9} & \bar{v} - \sqrt{p_9} \end{pmatrix}, \quad \tilde{\mathbf{T}}^{-1} = \begin{pmatrix} \frac{-\bar{v}+\sqrt{p_9}}{2\sqrt{p_9}} & \frac{1}{2\sqrt{p_9}} \\ \frac{\bar{v}+\sqrt{p_9}}{2\sqrt{p_9}} & \frac{-1}{2\sqrt{p_9}} \end{pmatrix} \tag{31}$$

Poisson's equation is satisfied at each time step as

$$\frac{V_{k+1} - V_k}{\Delta x_k} - \frac{V_k - V_{k-1}}{\Delta x_{k-1}} = -\Omega_k(n_{i,k} - n_{e,k}) \tag{32}$$

C. Temporal Discretization

The temporal derivatives are evaluated using a second-order expression in the following way:

$$\left(\frac{\partial a}{\partial t}\right)_i = \frac{3a_i^{n+1} - 4a_i^n + a_i^{n-1}}{2\Delta t} \tag{33}$$

$$\left(\frac{\partial e^{\tilde{Q}}}{\partial t}\right)_i = \frac{3e^{\tilde{Q}^{n+1}} - 4e^{\tilde{Q}^n} + e^{\tilde{Q}^{n-1}}}{2\Delta t} \tag{34}$$

D. Discrete Boundary Conditions

The discrete boundary conditions, explained previously in Sec. III.B, are given for a mesh with $M + 1$ points as

$$n_{i,0} = n_{i,1} \qquad n_{i,M} = n_{i,M-1} \qquad (n_iv_i)_0 = (n_iv_i)_1$$
$$(n_iv_i)_M = (n_iv_i)_{M-1} \qquad j_{e,0} = -\frac{1}{\sqrt{2\pi}}(1-\theta)n_{e,0}\sqrt{T_{e,0}}$$
$$j_{e,M} = -\frac{1}{\sqrt{2\pi}}(1-\theta)n_{e,M}\sqrt{T_{e,M}} \qquad q_{e,0} = \frac{5}{2}j_{e,0}T_{e,0}$$
$$q_{e,M} = \frac{5}{2}j_{e,M}T_{e,M} \qquad V_0 = V_{\max}\sin(\omega t) \qquad V_M = 0$$

E. Implicit Time Advancement

1. Newton's Method

The discretized equations are solved simultaneously at each time step, using Newton's method. The residual function, using Eq. (18), is defined for a given cell as

$$\mathbf{R}_i(\bar{\mathbf{Q}}) = \Omega_i\left(\frac{\partial \mathbf{Q}}{\partial t}\right)_i - \mathbf{F}_{i-\frac{1}{2}}(\bar{\mathbf{Q}}) + \mathbf{F}_{i+\frac{1}{2}}(\bar{\mathbf{Q}}) - \mathbf{S}_i(\bar{\mathbf{Q}})\Omega_i \tag{35}$$

where $\bar{\mathbf{Q}}$ is defined as the variable vector of mass, momentum, energy, and potential:

$$\bar{\mathbf{Q}} = \begin{pmatrix} \tilde{Q}_1 \\ n_i \\ n_iv_i \\ \tilde{Q}_2 \\ V \end{pmatrix} \tag{36}$$

Note that the expression for $\bar{\mathbf{Q}}$ in Eq. (36) differs from the time-dependent variable vector $\mathbf{Q}$ that was given in Eq. (10). The time-dependent variable vector $\mathbf{Q}$ does not include the potential V, because Poisson's equation has no temporal dependence.

For a given time step n, successive Newton iterations are applied to compute the variable vector $\mathbf{Q}^n$. At each Newton iteration, the linearization of the residual with respect to the dependent variables is computed. This is known as the Jacobian matrix and is represented as $\partial\mathbf{R}/\partial\bar{\mathbf{Q}}$. For a given Newton iteration $m+1$ for a given time step n, the update $(\Delta\bar{\mathbf{Q}}^{m,n})$ to the variable vector from the previous Newton iteration m is computed as

$$\left(\frac{\partial\mathbf{R}}{\partial\bar{\mathbf{Q}}}\right)^{m+1,n}(\Delta\bar{\mathbf{Q}})^{m,n} = -\mathbf{R}(\bar{\mathbf{Q}}^{m+1,n}) \tag{37}$$

$$\Delta\bar{\mathbf{Q}}^{m,n} = \bar{\mathbf{Q}}^{m+1,n} - \bar{\mathbf{Q}}^{m,n} \tag{38}$$

The Newton iterations continue within a given time step until the norm of the residual in a given Newton iteration falls below a specified tolerance. This indicates that the updated value $\mathbf{Q}^n$ has been computed, and the solution is advanced to the next time step. This process is repeated until the relative change in peak ion density from one cycle to the next goes below a specified tolerance, indicating that the solution has converged to a periodic state.

2. *Computing the Jacobian Using Complex Perturbations*

The Jacobian matrix is computed at each Newton step using complex perturbations of the components of $\mathbf{Q}$. The usefulness of complex perturbations is shown in the following example. Consider a scalar function f that depends on x that is perturbed by $i\Delta x$. The Taylor series expansion gives

$$f(x+i\Delta x) = f(x) + i\frac{\partial f}{\partial x}\Delta x - \frac{(\Delta x)^2}{2}\frac{\partial^2 f}{\partial x^2} - i\frac{(\Delta x)^3}{6}\frac{\partial^3 f}{\partial x^3} + \cdots \tag{39}$$

Equation (39) can be rearranged to compute $\partial f/\partial x$ as

$$\frac{\partial f}{\partial x} = \mathrm{Im}\left(\frac{f(x+i\Delta x)}{\Delta x}\right) + \frac{(\Delta x)^2}{6}\frac{\partial^3 f}{\partial x^3} + \cdots \tag{40}$$

This gives a second-order accurate solution for $\partial f/\partial x$ and does not involve subtractive cancellation errors. This example can be extended to vector problems and can be used to compute $\partial\mathbf{R}/\partial\bar{\mathbf{Q}}$. More details can be found in the appropriate literature [17,18].

For this one-dimensional problem, the Jacobian matrix structure is block pentadiagonal due to the use of extrapolated values for the fluxes.

IV. Computing Sensitivity Derivatives

Two general methods are used to compute sensitivity derivatives. The first method is forward mode direct differentiation, which computes the sensitivity of an unlimited number of cost functions with respect to one design variable. The storage requirements of direct differentiation are not large because the sensitivity is computed as the solution advances in time. The second method is the reverse mode adjoint method, which computes the sensitivity of one cost function to an unlimited number of design variables. In this case, after the periodic solution has been reached, the sensitivity derivatives are computed using solution values at each previous time step, going all the way back to the initial conditions.

There are two types of sensitivity methods that can be used: continuous and discrete. The continuous sensitivity method differentiates the governing equations first with respect to the design variables and then discretizes them, whereas a discrete sensitivity method discretizes the governing equations first and then differentiates them with respect to design variables. Discrete sensitivity methods are used in this work. Both the adjoint method and direct differentiation are implemented from a linearization of the discretized equations. Although there are hyperbolic and elliptic equations that have been discretized accordingly, these methods of computing sensitivity derivatives are generally applicable.

Sensitivity derivatives for time-dependent problems have previously been computed for aerodynamic applications for periodic solutions that evolve over time [19,20]. However, although aerodynamic simulations typically reach periodic state in a few cycles, it takes approximately 500 cycles for this particular glow discharge model to reach a periodic state. This is because the helium ion mass is 3 orders of magnitude larger than the electron mass, and, subsequently, the helium ions take a longer time to respond to the variations in the electric field.

Much of the following nomenclature for time-dependent sensitivity analysis has been expressed in a previous paper [21]. However, the current work takes into account the contributions from residual functions and cost functions that have an explicit dependence on the design variables.

A. Forward Mode Direct Differentiation

For a periodic solution, a given cost function depends on data at each time step during the periodic cycle. For this reason, it is convenient to define a global cost function I^g along with a local cost function I^n for each time step, where

$$I^g = \frac{1}{N}\sum_{n=1}^{n=N} I^n \tag{41}$$

and there are N time steps in the given period. Note that the local cost function I^n is only computed at time steps during the periodic cycle when the cost function is computed and is not computed at every time step during the solution process. For a time-dependent simulation that runs for n_{tot} time steps, the global sensitivity derivative with respect to a design variable (ignoring mesh sensitivity) can be expressed in terms of the local sensitivity derivatives as

$$\frac{\mathrm{d}I^g}{\mathrm{d}\beta} = \sum_{n=1}^{n=n_{\mathrm{tot}}}\frac{\mathrm{d}I^g}{\mathrm{d}I^n}\left[\frac{\partial I^n}{\partial\mathbf{Q}^n}\frac{\partial\mathbf{Q}^n}{\partial\beta} + \frac{\partial I^n}{\partial\beta}\right] \tag{42}$$

At each time step, the residual function $\mathbf{R}$ is driven down to zero. Differentiating the residual function with respect to β, one obtains

$$\frac{\mathrm{d}\mathbf{R}^n}{\mathrm{d}\beta} = \frac{\partial\mathbf{R}^n}{\partial\mathbf{Q}^n}\frac{\partial\mathbf{Q}^n}{\partial\beta} + \frac{\partial\mathbf{R}^n}{\partial\mathbf{Q}^{n-1}}\frac{\partial\mathbf{Q}^{n-1}}{\partial\beta} + \frac{\partial\mathbf{R}^n}{\partial\mathbf{Q}^{n-2}}\frac{\partial\mathbf{Q}^{n-2}}{\partial\beta} + \frac{\partial\mathbf{R}^n}{\partial\beta} = 0 \tag{43}$$

Thus, one can solve for $\partial\mathbf{Q}^n/\partial\beta$ as

$$\frac{\partial\mathbf{Q}^n}{\partial\beta} = -\left[\frac{\partial\mathbf{R}^n}{\partial\mathbf{Q}^n}\right]^{-1}\left(\frac{\partial\mathbf{R}^n}{\partial\mathbf{Q}^{n-1}}\frac{\partial\mathbf{Q}^{n-1}}{\partial\beta} + \frac{\partial\mathbf{R}^n}{\partial\mathbf{Q}^{n-2}}\frac{\partial\mathbf{Q}^{n-2}}{\partial\beta} + \frac{\partial\mathbf{R}^n}{\partial\beta}\right) \tag{44}$$

Substituting Eq. (44) into Eq. (42) results in the following expression for the global sensitivity derivative:

$$\frac{\mathrm{d}I^g}{\mathrm{d}\beta} = \sum_{n=1}^{n=n_{\mathrm{tot}}}\frac{\mathrm{d}I^g}{\mathrm{d}I^n}\left\{-\frac{\partial I^n}{\partial\mathbf{Q}^n}\left[\frac{\partial\mathbf{R}^n}{\partial\mathbf{Q}^n}\right]^{-1}\left(\frac{\partial\mathbf{R}^n}{\partial\mathbf{Q}^{n-1}}\frac{\partial\mathbf{Q}^{n-1}}{\partial\beta} + \frac{\partial\mathbf{R}^n}{\partial\mathbf{Q}^{n-2}}\frac{\partial\mathbf{Q}^{n-2}}{\partial\beta} + \frac{\partial\mathbf{R}^n}{\partial\beta}\right) + \frac{\partial I^n}{\partial\beta}\right\} \tag{45}$$

Forward mode direct differentiation computes sensitivity derivatives by computing $\partial I^n/\partial\mathbf{Q}^n$, $\partial I^n/\partial\beta$, and $\partial\mathbf{Q}^n/\partial\beta$ at each time step as the solution is advanced in time. Once the solution has converged to a periodic state, the sensitivity derivatives of a vector of cost functions $\mathbf{I}^g$ with respect to one design variable can be computed over the course of one period. It is most useful in situations where there are many cost functions and only a few design variables.

At each time step, the quantity $\partial\mathbf{Q}/\partial\beta$ can be computed using Eq. (44), allowing the sensitivity derivative to be computed as the algorithm marches forward in time. The steps are as follows:

1) Compute the sensitivity of initial conditions to the design variable $\partial\mathbf{Q}^0/\partial\beta$.

2) Advance the solution by one time step.

3) Compute $\partial\mathbf{R}^n/\partial\beta$ at the current time step.

4) Compute $\partial\mathbf{Q}^n/\partial\beta$ for the current time step using Eq. (44).

5) Repeat steps 2–4 until the solution has converged to a periodic state. At convergence, begin the cycle to compute the cost function and sensitivity derivatives.

6) Advance the solution by one time step.

7) Compute $\partial \mathbf{I}^n/\partial\beta$ for the current time step.

8) Compute $\partial \mathbf{I}^n/\partial \mathbf{Q}^n$ for the current time step.

9) Compute $\partial \mathbf{Q}^n/\partial\beta$ for the current time step using Eq. (44).

10) Compute $\mathrm{d}\mathbf{I}^g/\mathrm{d}\mathbf{I}^n$ for the current time step.

11) Using Eq. (42), add the contribution

$$\frac{d\mathbf{I}^g}{d\mathbf{I}^n}\left[\frac{\partial \mathbf{I}^n}{\partial \mathbf{Q}^n}\frac{\partial \mathbf{Q}^n}{\partial \beta}+\frac{\partial \mathbf{I}^n}{\partial \beta}\right]$$

to the global sensitivity derivative $\mathrm{d}\mathbf{I}^g/\mathrm{d}\beta$.

12) Repeat steps 6–12 until the period has been completed.

Direct differentiation does not require much storage for saving the complete time history of $\partial \mathbf{Q}/\partial\beta$ because the quantities $\partial \mathbf{Q}^n/\partial\beta$, $\partial \mathbf{Q}^{n-1}/\partial\beta$, and $\partial \mathbf{Q}^{n-2}/\partial\beta$ can be overwritten as the algorithm proceeds. However, if there are multiple design variables, direct differentiation must be repeated for each design variable.

B. Reverse Mode Adjoint

Reverse mode adjoint has been used to compute sensitivity derivatives in steady and unsteady flows for a wide variety of applications [22–25]. For time-dependent problems with cost functions that are computed over a given period, reverse mode adjoint involves converging the solution and, from there, stepping backward in time to compute the sensitivity derivatives. It allows for the efficient computation of sensitivity derivatives where there is one cost function I^g and numerous design variables expressed in the vector $\boldsymbol{\beta}$. Thus, the reverse mode adjoint method is useful in applications where many design variables are present.

Taking the transpose of Eqs. (42) and (44), one obtains

$$\frac{\mathrm{d}I^{g^T}}{\mathrm{d}\boldsymbol{\beta}}=\sum_{n=0}^{n=n_f}\left[\frac{\partial \mathbf{Q}^{n^T}}{\partial \boldsymbol{\beta}}\frac{\partial I^{n^T}}{\partial \mathbf{Q}^n}+\frac{\partial I^{n^T}}{\partial \boldsymbol{\beta}}\right]\frac{\mathrm{d}I^{g^T}}{\mathrm{d}I^n} \tag{46}$$

$$\frac{\partial \mathbf{Q}^{n^T}}{\partial \boldsymbol{\beta}}=-\left(\frac{\partial \mathbf{Q}^{n-1^T}}{\partial \boldsymbol{\beta}}\frac{\partial \mathbf{R}^{n^T}}{\partial \mathbf{Q}^{n-1}}+\frac{\partial \mathbf{Q}^{n-2^T}}{\partial \boldsymbol{\beta}}\frac{\partial \mathbf{R}^{n^T}}{\partial \mathbf{Q}^{n-2}}+\frac{\partial \mathbf{R}^{n^T}}{\partial \boldsymbol{\beta}}\right)\left[\frac{\partial \mathbf{R}^n}{\partial \mathbf{Q}^n}\right]^{-T} \tag{47}$$

Because reverse mode differentiation starts at the final time step and goes backward, it is convenient to rewrite Eq. (46) as

$$\frac{\mathrm{d}I^{g^T}}{\mathrm{d}\boldsymbol{\beta}}=\sum_{n=n_{\mathrm{tot}}}^{n=1}\left[\frac{\partial \mathbf{Q}^{n^T}}{\partial \boldsymbol{\beta}}\frac{\partial I^{n^T}}{\partial \mathbf{Q}^n}+\frac{\partial I^{n^T}}{\partial \boldsymbol{\beta}}\right]\frac{\mathrm{d}I^{g^T}}{\mathrm{d}I^n} \tag{48}$$

Substituting Eq. (47) into Eq. (48) results in

$$\frac{\mathrm{d}I^{g^T}}{\mathrm{d}\boldsymbol{\beta}}=\sum_{n=n_{\mathrm{tot}}}^{n=1}\left\{-\left(\frac{\partial \mathbf{Q}^{n-1^T}}{\partial \boldsymbol{\beta}}\frac{\partial \mathbf{R}^{n^T}}{\partial \mathbf{Q}^{n-1}}+\frac{\partial \mathbf{Q}^{n-2^T}}{\partial \boldsymbol{\beta}}\frac{\partial \mathbf{R}^{n^T}}{\partial \mathbf{Q}^{n-2}}+\frac{\partial \mathbf{R}^{n^T}}{\partial \boldsymbol{\beta}}\right)\left[\frac{\partial \mathbf{R}^n}{\partial \mathbf{Q}^n}\right]^{-T}\frac{\partial I^{n^T}}{\partial \mathbf{Q}^n}+\frac{\partial I^{n^T}}{\partial \boldsymbol{\beta}}\right\}\frac{\mathrm{d}I^{g^T}}{\mathrm{d}I^n} \tag{49}$$

Because of the considerable expense of computing each term in Eq. (49) at each time step, the terms are computed recursively. The algorithm is as follows:

1) Converge the solution to a periodic state, storing the solution vector $\mathbf{Q}$ at each time step.

2) Starting with the final time step, compute the first flow adjoint vector $\boldsymbol{\Lambda}^n$ as

$$\boldsymbol{\Lambda}^{\,n}=-\left[\frac{\partial \mathbf{R}^n}{\partial \mathbf{Q}^n}\right]^{-T}\frac{\partial I^{n^T}}{\partial \mathbf{Q}^n}\frac{\mathrm{d}I^{g^T}}{\mathrm{d}I^n} \tag{50}$$

3) Compute the vector $\partial I^{n^T}/\partial\boldsymbol{\beta}$ for the current time step.

4) Add the contribution

$$\left(\frac{\partial \mathbf{R}^{n^T}}{\partial \boldsymbol{\beta}}\boldsymbol{\Lambda}^n+\frac{\partial I^{n^T}}{\partial \boldsymbol{\beta}}\frac{\partial I^g}{\partial I^n}\right)$$

from the current time step to the global sensitivity derivative $\mathrm{d}I^g/\mathrm{d}\boldsymbol{\beta}$.

5) Compute the second flow adjoint vector $\boldsymbol{\lambda}^n$ as

$$\boldsymbol{\lambda}^{\,n}=\frac{\partial \mathbf{R}^{n^T}}{\partial \mathbf{Q}^{n-1}}\boldsymbol{\Lambda}^n+\frac{\partial I^{n-1^T}}{\partial \mathbf{Q}^{n-1}}\frac{\mathrm{d}I^{g^T}}{\mathrm{d}I^{n-1}} \tag{51}$$

6) Compute the first flow adjoint vector for the previous time step as

$$\boldsymbol{\Lambda}^{\,n-1}=-\left[\frac{\partial \mathbf{R}^{n-1}}{\partial \mathbf{Q}^{n-1}}\right]^{-T}\boldsymbol{\lambda}^n \tag{52}$$

7) Compute $\partial I^{n-1^T}/\partial\boldsymbol{\beta}$.

8) Add the contribution

$$\left(\frac{\partial \mathbf{R}^{n-1^T}}{\partial \boldsymbol{\beta}}\boldsymbol{\Lambda}^{n-1}+\frac{\partial I^{n-1^T}}{\partial \boldsymbol{\beta}}\frac{\partial I^g}{\partial I^{n-1}}\right)$$

from the current time step to the global sensitivity derivative.

9) Compute the second flow adjoint vector for the previous time step as

$$\boldsymbol{\lambda}^{\,n-1}=\frac{\partial \mathbf{R}^{n-1^T}}{\partial \mathbf{Q}^{n-2}}\boldsymbol{\Lambda}^{n-1}+\frac{\partial \mathbf{R}^{n^T}}{\partial \mathbf{Q}^{n-2}}\boldsymbol{\Lambda}^n+\frac{\partial I^{n-2^T}}{\partial \mathbf{Q}^{n-2}}\frac{\mathrm{d}I^{g^T}}{\mathrm{d}I^{n-2}} \tag{53}$$

10) Continue stepping backward, computing $\boldsymbol{\Lambda}$ and $\boldsymbol{\lambda}$ at each time step as well as adding the contribution to the global sensitivity derivative. Note that, once the time step number is less than the time step number at which the computation of the cost function began, the sensitivity of the global cost function with respect to the local cost function ($\mathrm{d}\mathbf{I}^g/\mathrm{d}\mathbf{I}^n$) is equal to zero.

11) Once $\boldsymbol{\lambda}^1$ has been computed, add $(\partial \mathbf{Q}^{0^T}/\partial\boldsymbol{\beta})\boldsymbol{\lambda}^1$ to the global sensitivity derivative $\mathrm{d}\mathbf{I}^g/\mathrm{d}\beta$. Note that $\partial \mathbf{Q}^0/\partial\boldsymbol{\beta}$ is computed from the known initial conditions.

Thus, the expression for the total sensitivity derivative is computed as

$$\frac{\mathrm{d}I^g}{\mathrm{d}\boldsymbol{\beta}}=\frac{\partial \mathbf{Q}^{\mathrm{init}^T}}{\partial \boldsymbol{\beta}}\boldsymbol{\lambda}^1+\sum_{n=1}^{n=n_{\mathrm{tot}}}\left[\frac{\partial \mathbf{R}^{n^T}}{\partial \boldsymbol{\beta}}\boldsymbol{\Lambda}^n+\frac{\partial I^{n^T}}{\partial \boldsymbol{\beta}}\frac{\partial I^{g^T}}{\partial I^n}\right] \tag{54}$$

The reverse mode adjoint method is good for problems that have many design variables. However, it requires more storage than forward mode direct differentiation because the solution vector must be stored at every single time step in the solution process. This can be accomplished either by storing the solution vectors in memory or by writing out the solution vectors to a file.

C. Applications of Sensitivity Derivatives

1. *Design Modification*

After the sensitivity derivatives have been obtained using forward mode direct differentiation or reverse mode adjoint, the vector of design variables can then be changed by an amount $\Delta\boldsymbol{\beta}$, so as to increase or decrease a global cost function. The most elementary method to compute $\Delta\boldsymbol{\beta}$ is presented here for its simplicity. Once the design variables have been changed, the global cost function can be expressed using a Taylor series approximation as

$$I^g(\boldsymbol{\beta}+\Delta\boldsymbol{\beta})=I^g(\boldsymbol{\beta})+\left[\frac{\mathrm{d}I^g}{\mathrm{d}\boldsymbol{\beta}}\right]^T\Delta\boldsymbol{\beta}+\cdots \tag{55}$$

If the change in the design variables $\Delta\boldsymbol{\beta}$ is chosen appropriately, one can guarantee that the cost function will increase or decrease accordingly with each design cycle. If $\Delta\boldsymbol{\beta}=\alpha(\mathrm{d}I^g/\mathrm{d}\boldsymbol{\beta})$, where α is a small number, this results in

$$I^g(\boldsymbol{\beta} + \Delta\boldsymbol{\beta}) = I^g(\boldsymbol{\beta}) + \alpha\left[\frac{\mathrm{d}I^g}{\mathrm{d}\boldsymbol{\beta}}\right]^T\left[\frac{\mathrm{d}I^g}{\mathrm{d}\boldsymbol{\beta}}\right] + \cdots \tag{56}$$

Equation (56) shows that if α is positive, the cost function will always increase, whereas, if α is negative, the cost function will always decrease. This relationship will hold so long as small enough values of α are used.

This method, although very simple, can be inefficient and laborious to use for finding the maximum or minimum values of a cost function. There is no way of determining a priori a good value of α for a particular design problem. First, if α is too small, the cost function will change very slowly, leading to inefficiency in computations. Second, because Eq. (55) is essentially an explicit algorithm for updating the design variables, α cannot be too big, so as to ensure the stability of the algorithm.

Although more robust algorithms exist, the present technique is easily implemented and is suitable for demonstrating the use of sensitivity derivatives for design modifications.

2. *Computation of Error Bounds*

Sensitivity derivatives can also be used to compute error bounds for given cost functions [26]. Each design variable involved in a computation will have an experimental uncertainty and will contribute to the total error bound of the given cost function. The maximum possible total modeling uncertainty of a cost function, where n design variables are present, can be computed as

$$\max|\hat{I}^g - I^g| = \max|\Delta I^g| = \sum_{i=1}^{i=n}\left|\frac{\mathrm{d}I^g}{\mathrm{d}\beta_n}\Delta\beta_n\right| \tag{57}$$

where $\Delta\beta_n$ is the experimental uncertainty for a given design variable. It should be noted that, although error bounds can be computed using Eq. (57), this equation does not take into account the error that comes from using a model that does not take into account the appropriate species and reactions. However, if a reaction rate is computed from a particle-in-cell simulation with an appropriate uncertainty, and is then used in a fluid simulation, Eq. (57) can be used to estimate the error bound on the cost function due to the uncertainty in the reaction rate. Examples of how sensitivity derivatives can be used to compute physical uncertainties are found in the appropriate literature [26–28].

V. Results

The simulation parameters previously shown in Table 1 were used as design variables in the current simulation. The local cost functions computed for each simulation are given in Table 2. In each case, the global cost function is computed as in Eq. (41). In this particular case, the simulation was run for 500 RF cycles, and the cost functions were computed during the final cycle.

The sensitivity of each cost function with respect to each of the design variables listed in Table 1 were computed using finite differences, forward mode direct differentiation, and reverse mode adjoint. They were computed on a 161-point evenly spaced grid using 1000 time steps per cycle. The finite differences were computed by perturbing the design variable by 10^{-6} of its original value and using a central difference formula. The sensitivity derivatives were computed after 500 cycles and are given in Tables 3 and 4. Each method took approximately 12 h on an Intel server with a single 3.2 GHz 32-bit Pentium processor and 1 GB of RAM. It is expected that direct differentiation and the adjoint method would take a similar amount of time because, in each method, an additional linear solve is required at each time step. However, this process must be repeated for each design variable when using direct differentiation and repeated for each cost function when using the adjoint method.

Sensitivity derivatives computed using direct differentiation and adjoint methods differed from those computing using finite differences by an rms of 3.4×10^{-6}. Sensitivity derivatives computed using direct differentiation differed from those computed using adjoint methods by an rms of 6.5×10^{-10}. The similarity between derivatives computed using direct differentiation and those computed using the adjoint method is expected, because both of these computations used the same solution data, the same matrices, and the same residuals. They differ from the derivatives computed using finite differences because the computations are not subject to the same cancellation error that comes from using finite differences.

In computing sensitivity derivatives, significant computational savings are obtained by using forward mode direct differentiation and reverse mode adjoint as opposed to finite differences. For a

Table 3 Sensitivity derivatives for first four cost functions

	C_1	C_2	C_3	C_4
l, m	8.03×10^{2}	1.33×10^{17}	5.77×10^{2}	4.16×10^{3}
$V_{\max}$, V	3.24×10^{-1}	6.51×10^{12}	1.35×10^{-1}	1.91×10^{0}
p, torr	5.33×10^{1}	2.08×10^{16}	1.80×10^{2}	1.50×10^{3}
f, Hz	1.56×10^{-6}	5.57×10^{8}	8.77×10^{-6}	5.91×10^{-5}
T_{gas}, K	-4.44×10^{-2}	-1.73×10^{13}	-1.50×10^{-1}	-1.25×10^{0}
k_{mt}, m^3/s	2.53×10^{14}	1.05×10^{14}	1.76×10^{15}	1.21×10^{16}
k_{i0}, m^3/s	-1.51×10^{14}	5.28×10^{28}	2.77×10^{13}	2.27×10^{15}
σ_{cx}, m^2	2.40×10^{19}	5.12×10^{33}	-2.64×10^{19}	-3.91×10^{19}
E_a, eV	3.65×10^{-1}	-1.29×10^{14}	-6.25×10^{-2}	-5.38×10^{0}
h_{iz}, eV	-2.15×10^{-1}	-7.72×10^{13}	5.73×10^{-1}	-8.92×10^{0}
θ	-5.11×10^{0}	7.62×10^{14}	1.35×10^{1}	7.62×10^{1}

Table 4 Sensitivity derivatives for last four cost functions

	C_5	C_6	C_7	C_8
l, m	9.42×10^{19}	3.71×10^{2}	3.03×10^{2}	4.00×10^{3}
$V_{\max}$, V	2.32×10^{16}	9.16×10^{-2}	8.04×10^{-3}	1.86×10^{0}
p, torr	3.05×10^{19}	1.20×10^{2}	4.39×10^{1}	1.48×10^{3}
f, Hz	1.52×10^{12}	5.97×10^{-6}	8.30×10^{-7}	5.68×10^{-5}
T_{gas}, K	-2.53×10^{16}	-9.99×10^{-2}	-3.67×10^{-2}	-1.23×10^{0}
k_{mt}, m^3/s	2.82×10^{32}	1.11×10^{15}	3.04×10^{14}	1.17×10^{16}
k_{i0}, m^3/s	6.09×10^{31}	2.40×10^{14}	-6.84×10^{12}	2.17×10^{15}
σ_{cx}, m^2	-4.76×10^{36}	-1.88×10^{19}	6.41×10^{18}	-1.22×10^{18}
E_a, eV	-1.46×10^{17}	-5.73×10^{-1}	1.45×10^{-2}	-5.14×10^{0}
h_{iz}, eV	-2.06×10^{17}	9.53×10^{-1}	-1.16×10^{-1}	-8.58×10^{0}
θ	2.06×10^{18}	8.10×10^{0}	1.18×10^{0}	7.29×10^{1}

Table 2 Cost functions analyzed for sensitivity derivatives

Cost function	Number	I^n
Average total voltage, V	C_1	$\frac{1}{l}\sum_{i=0}^{i=n_c} V_i\Omega_i$
Average peak electron density, /m^3	C_2	$\bar{n}_e$
Electron power deposition, W/m^2	C_3	$\frac{1}{l}\sum_{i=0}^{i=n_c-1} -ej_eE\Delta x_i$
Ion power deposition, W/m^2	C_4	$\frac{1}{l}\sum_{i=0}^{i=n_c} en_iv_iE\Omega_i$
Total electron production, /m^3-s	C_5	$\frac{1}{l}\sum_{i=0}^{i=n_c} N_{\text{He}}n_ek_{i0}e^{-E_a/kT_e}\Omega_i$
Ionization power losses, W/m^2	C_6	$\frac{1}{l}\sum_{i=0}^{i=n_c} h_{iz}N_{\text{He}}n_ek_{i0}e^{-E_a/kT_e}\Omega_i$
Elastic collision power losses, W/m^2	C_7	$\frac{1}{l}\sum_{i=0}^{i=n_c} 3\frac{m_e}{m_{\text{He}}}N_{\text{He}}k_{\text{mt}}n_ek(T_e - T_{\text{gas}})\Omega_i$
Charge exchange collision power losses, W/m^2	C_8	$\frac{1}{l}\sum_{i=0}^{i=n_c} \frac{\pi}{2}m_in_iN_{\text{He}}\sigma_{\text{cx}}\lvert v_i\rvert v_i^2\Omega_i$

Table 5 Changes in design variables to increase electron power deposition

Design cycle	l, cm	p, mTorr	Electron power deposition, W/m^2
0	4.000	250.0	63.6
1	4.058	250.2	63.9
2	4.115	250.4	64.3
3	4.173	250.5	64.6
4	4.231	250.7	64.9

Table 6 Changes in design variables to decrease electron power deposition

Design cycle	l, cm	p, mTorr	Electron power deposition, W/m^2
0	4.000	250.0	63.6
1	3.942	249.8	63.2
2	3.885	249.6	62.8
3	3.827	249.5	62.4
4	3.769	249.3	61.9

problem with n_c cost functions and n_d design variables, computing sensitivity derivatives using finite differences requires one to run $n_d + 1$ solutions for the numerical solver and is subject to cancellation error. Forward mode direct differentiation requires n_d solutions of the numerical solver and n_d direct differentiation solutions. Reverse mode adjoint requires n_c solutions of the numerical solver and n_c adjoint solutions.

To test the effectiveness of using the sensitivity derivatives for the purpose of design modification, the electron power deposition is used as the cost function, while the physical design variables (T_{gas}, f, p, $V_{\max}$, and l) are modified at each design cycle. In the first optimization test, changes to the design variables are for the purposes of increasing the electron power deposition. A value of 1.0×10^{-6} is used for α. It is evident in Table 5 that the electron power deposition increases with each design cycle. It was observed that the gas temperature, frequency, and voltage amplitude changed very little with each design cycle and thus their values were not included in Table 5.

In the second optimization test, changes to the design variables are for the purpose of decreasing the electron power deposition, and thus a value of -1.0×10^{-6} is used for α. Table 6 shows the electron power deposition decreasing with successive design cycles. As before, T_{gas}, f, and $V_{\max}$ changed very little with each design cycle and are not shown in Table 6.

VI. Conclusions

The current work shows that forward mode direct differentiation and reverse mode adjoint can be used to accurately compute sensitivity derivatives. Sensitivity derivatives computed using finite differences compare very well with those using forward mode direct differentiation and reverse mode adjoint. The current work also shows how sensitivity derivatives can be used to modify the design variables so as to increase or decrease a given cost function. Finally, it is shown how error bounds can be computed for a given cost function using sensitivity derivatives and experimental uncertainties.

In the future, it is hoped that sensitivity analysis can be applied to simulations that include more chemical species in the model. In addition, it is hoped that this sensitivity analysis can be extended to two-dimensional simulations. This will present some significant challenges. For instance, the equations for a given node require information from nodes that are not immediately adjacent to that node. In a one-dimensional case, this results in a block pentadiagonal matrix but, for a two-dimensional case, this will increase the complexity of the matrix problem. One way of getting around this would be to use Jacobian-free Krylov methods [29], which only require evaluations of the residual and do not require one to store the Jacobian matrix.

Another issue to face in doing a two-dimensional simulation is the computational runtime required for the simulation. Currently, it takes approximately 15 h to run 500 cycles for a simulation with 161 evenly spaced points and 1000 time steps per cycle. This means that extending this work to two dimensions will necessitate running the computer code on parallel processors.

Finally, using reverse mode adjoint requires extensive memory storage requirements, even for one dimension. In these cases, five variables at 161 points for a total of 500,000 time steps must be stored. This could be accomplished for a two-dimensional simulation through the use of portable memory storage devices.

Acknowledgment

Funding for this research has been provided by the Tennessee Higher Education Commission.

References

[1] Bogaerts, A., Neyts, E., Gijbels, R., and van der Mullen, J., "Gas Discharge Plasmas and Their Applications," *Spectrochimica Acta, Part B: Atomic Spectroscopy*, Vol. 57, No. 4, 2002, pp. 609–658. doi:10.1016/S0584-8547(01)00406-2

[2] Sobel, A., "Plasma Displays," *IEEE Transactions on Plasma Science*, Vol. 19, No. 6, 1991, pp. 1032–1047. doi:10.1109/27.125029

[3] Lieberman, M., and Lichtenberg, A., *Principles of Plasma Discharges and Materials Processing*, Wiley, Hoboken, NJ, 2005.

[4] Wharmby, D., "Electrodeless lamps for lighting: a review," *IEE Proceedings, A: Science, Measurement and Technology*, Vol. 140, No. 6, 1993, pp. 465–473.

[5] Kogoma, M., and Okazaki, S., "Raising of Ozone Formation Efficiency in a Homogeneous Glow Discharge Plasma at Atmospheric Pressure," *Journal of Physics D: Applied Physics*, Vol. 27, No. 9, 1994, pp. 1985–1987. doi:10.1088/0022-3727/27/9/026

[6] Rauf, S., and Kushner, M. J., "Dynamics of a Coplanar-Electrode Plasma Display Panel Cell, I: Basic Operation," *Journal of Applied Physics*, Vol. 85, No. 7, 1999, pp. 3460–3469. doi:10.1063/1.369703

[7] Lymberopoulos, D., and Economou, D. J., "Two-Dimensional Self-Consistent Radio Frequency Plasma Simulations Relevant to the Gaseous Electronics Conference RF Reference Cell," *Journal of Research of the National Institute of Standards and Technology*, Vol. 100, No. 4, 1995, pp. 473–494.

[8] Nitschke, T., and Graves, D., "A Comparison of Particle in Cell and Fluid Model Simulations of Low-Pressure Radio Frequency Discharges," *Journal of Applied Physics*, Vol. 76, No. 10, 1994, pp. 5646–5660. doi:10.1063/1.358435

[9] Hammond, E., Mahesh, K., and Moin, P., "A Numerical Method to Simulate Radio-Frequency Plasma Discharges," *Journal of Computational Physics*, Vol. 176, No. 2, 2002, pp. 402–429. doi:10.1006/jcph.2001.6994

[10] Wang, Q., Economou, D., and Donnelly, V., "Simulation of a Direct Current Microplasma Discharge in Helium at Atmospheric Pressure," *Journal of Applied Physics*, Vol. 100, No. 2, 2006, p. 023301. doi:10.1063/1.2214591

[11] Yuan, X., and Raja, L., "Computational Study of Capacitively Coupled High-Pressure Glow Discharges in Helium," *IEEE Transactions on Plasma Science*, Vol. 31, No. 4, 2003, pp. 495–503. doi:10.1109/TPS.2003.815479

[12] Rauf, S., and Kushner, M. J., "Dynamics of a Coplanar-Electrode Plasma Display Panel Cell, II: Cell Optimization," *Journal of Applied Physics*, Vol. 85, No. 7, 1999, pp. 3470–3476. doi:10.1063/1.369704

[13] Lange, K., "A Fully Implicit Characteristic-Based Algorithm for a One Dimensional Radio Frequency Glow Discharge Fluid Simulation," M.S. Thesis, Univ. of Tennessee at Chattanooga, TN, July 2007.

[14] Graves, D., and Jensen, K., "A Continuum Model of DC and RF Discharges," *IEEE Transactions on Plasma Science*, Vol. 14, No. 2, 1986, pp. 78–91. doi:10.1109/TPS.1986.4316510

[15] Roe, P., "Approximate Riemann Solvers, Parameter Vectors, and Difference Schemes," *Journal of Computational Physics*, Vol. 43,

No. 2, 1981, pp. 357–372.
doi:10.1016/0021-9991(81)90128-5

[16] Tang, H., "On the Sonic Point Glitch," *Journal of Computational Physics*, Vol. 202, No. 2, 2005, pp. 507–532.
doi:10.1016/j.jcp.2004.07.013

[17] Whitfield, D., and Taylor, L., "Variants of a Two-Level Method for the Approximate Numerical Solution of Field Simulation Equations," Mississippi State Univ. Engineering Research Center, TR 98-09, July 1998.

[18] Anderson, W., Newman, J., Whitfield, D., and Nielsen, E., "Sensitivity Analysis for the Navier-Stokes Equations on Unstructured Meshes Using Complex Variables," AIAA Paper 1999-3294, 1999.

[19] Nadarajah, S., and Jameson, A., "Optimal Control of Unsteady Flows Using a Time Accurate Method," AIAA Paper 2002-5436, 2002.

[20] Ghayour, K., and Baysal, O., "Limit-Cycle Shape Optimization using Time-Dependent Transonic Equation," AIAA Paper 1999-3375, 1999.

[21] Mani, K., and Mavriplis, D., "An Unsteady Discrete Adjoint Formulation for Two-Dimensional Flow Problems with Deforming Meshes," AIAA Paper 2007-60, Jan. 2007.

[22] Pironneau, O., "On Optimum Design in Fluid Mechanics," *Journal of Fluid Mechanics*, Vol. 64, No. 1, 1974, pp. 97–110.
doi:10.1017/S0022112074002023

[23] Jameson, A., "Aerodynamic Design via Control Theory," *Journal of Scientific Computing*, Vol. 3, No. 3, 1988, pp. 233–260.
doi:10.1007/BF01061285

[24] Taylor, A. I., Hou, G., and Korivi, V., "Methodology for Calculating Aerodynamic Sensitivity Derivatives," *AIAA Journal*, Vol. 30, No. 10, 1992, pp. 2411–2419.
doi:10.2514/3.11241

[25] Kapadia, S., Anderson, W., and Elliott, L., "Design and Sensitivity Analysis of Solid Oxide Fuel Cells Using Discrete Adjoint Method," AIAA Paper 2007-4832, June 2007.

[26] Blackwell, B., Dowding, K., Cochran, R., and Dobranich, D., "Utilization of Sensitivity Coefficients to Guide the Design of a Thermal Battery," *Proceedings of the 1998 ASME International Mechanical Engineering Congress and Exposition*, Vol. 361, American Society of Mechanical Engineers, Fairfield, NJ, Nov. 1998, pp. 73–82.

[27] Putko, M., Newman, P., Taylor, A., III., and Green, L., "Approach for Uncertainty Propagation and Robust Design in CFD Using Sensitivity Derivatives," AIAA Paper 2001-2528, June 2001.

[28] Turgeon, E., Pelletier, D., Etienne, S., and Borggaard, J., "Sensitivity and Uncertainty Analysis for Turbulent Flows," AIAA Paper 2002-0985, Jan. 2002.

[29] Chan, T., and Jackson, K., "Nonlinearly Preconditioned Krylov Subspace Methods for Discrete Newton Algorithms," *SIAM Journal on Scientific and Statistical Computing*, Vol. 5, No. 3, 1984, pp. 533–542.
doi:10.1137/0905039

D. Gaitonde
Associate Editor