

AD-A258 647



2

SPACECRAFT INTERACTION WITH AMBIENT AND
SELF-GENERATED PLASMA / NEUTRAL ENVIRONMENT

AFOSR Contract F49620-90-C-0051

Final Report 31 August 1992

DTIC
ELECTE
NOV 25 1992
S A D

Prepared by:

T. S. Mogstad
Advanced Space Science & Technology
Design & Technology Center

Approved by:

R. J. Reck, Director
Design & Technology Center

*Original contains color
plates; All DTIC reproductions
will be in black and
white.

Prepared for:

Air Force Office of Scientific Research
Bolling Air Force Base
D. C. 20332-6448

This document has been approved
for public release and sale; its
distribution is unlimited.

MCDONNELL DOUGLAS SPACE SYSTEMS COMPANY
DESIGN & TECHNOLOGY CENTER
5301 Bolsa Avenue, Huntington Beach, CA 92647, (714) 896-3311

92 11 24 018

92-30156



10402

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY (Leave blank)		2. REPORT DATE 31 August 1992		3. REPORT TYPE AND DATES COVERED Final Report 1 July 90 - 31 August 92	
4. TITLE AND SUBTITLE Spacecraft Interaction with Ambient and Self-Generated Plasma / Neutral Environment				5. FUNDING NUMBERS \$53,807	
6. AUTHOR(S) T. S. Mogstad				7. PERFORMING ORGANIZATION REPORT NUMBER F49620-90-C-0051	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) McDonnell Douglas Space Systems Company 5301 Bolsa Avenue Huntington Beach, CA 92647				8. PERFORMING ORGANIZATION REPORT NUMBER F49620-90-C-0051	
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) Air Force Office of Scientific Research AFOSR-TR Dolling AFB SC 29505-5010				10. SPONSORING / MONITORING AGENCY REPORT NUMBER F49620-90-C-0051DEF	
11. SUPPLEMENTARY NOTES Torkil S. Mogstad A3-Y881-13/3 Tel (714) 896-3437					
12a. DISTRIBUTION / AVAILABILITY STATEMENT				12b. DISTRIBUTION CODE	
13. ABSTRACT (Maximum 200 words) This report presents the results of our study of Spacecraft Interaction with Ambient and Self-Generated Plasma/Neutral Environment. Various neutral effluent release scenarios in low Earth orbit were defined and modeled with direct simulation Monte Carlo (DSMC) methods. The simulated environments included H ₂ venting from an SDI-type chemical power system, and Space Station Freedom (SSF) mono and bipropellant hydrazine thruster plume distributions at high-voltage solar arrays. After the neutral distributions were characterized, various ionization mechanisms (charge exchange, critical velocity ionization effects, photo-ionization) were included to determine the local plasma production. Neutral and ion continuity equations were solved to obtain the local ion distribution in the vicinity of high-voltage surfaces. It was shown that the thruster effluents can generate self-induced neutral and plasma density distributions several orders of magnitude greater than the natural ambient environment. For a bipropellant MMH-N ₂ O ₄ thruster firing into the ram, our simulations indicate that the local plasma density may increase as much as four orders of magnitude over the ambient. The effects of these self-induced environments on high-voltage surface arcing were assessed by using a microscopic model of the arcing process developed at MIT. High-voltage surface arcing thresholds, probabilities, and frequencies for current and future space platforms were discussed.					
14. SUBJECT TERMS				15. NUMBER OF PAGES 99	
				16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT Unclassified		18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified		19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	
20. LIMITATION OF ABSTRACT					

DISCLAIMER NOTICE



THIS DOCUMENT IS BEST QUALITY AVAILABLE. THE COPY FURNISHED TO DTIC CONTAINED A SIGNIFICANT NUMBER OF COLOR PAGES WHICH DO NOT REPRODUCE LEGIBLY ON BLACK AND WHITE MICROFICHE.

Table of Contents

List of Figures.....	iv
List of Acronyms.....	v
INTRODUCTION.....	1
RESULTS.....	2
<u>Neutral Distribution</u>	2
Chemical Power System Exhaust.....	2
SSF Hydrazine Thrusters.....	3
25 lbf Monopropellant Hydrazine Thruster.....	5
25 lbf Bipropellant MMH-N ₂ O ₄ Thruster.....	7
2.5 lbf Monopropellant Hydrazine Thruster.....	7
2.5 lbf Bipropellant Hydrazine Thruster.....	7
Surface Accommodation.....	7
<u>Ionization of Neutral Effluents</u>	10
Ion-Molecule Reactions.....	10
Photo-Ionization.....	11
Critical Ionization Velocity (CIV).....	12
Numerical Simulation of the Ionization Process.....	15
25 lbf Monopropellant Hydrazine Thruster.....	17
25 lbf Bipropellant MMH-N ₂ O ₄ Thruster.....	19
<u>Arcing at High-Voltage Surfaces</u>	23
CONCLUSION.....	25
BIBLIOGRAPHY.....	26
APPENDIX A: The Direct Simulation Monte Carlo Method.....	28
APPENDIX B: Flux Solver for Computing Local Plasma Density.....	30
Figures.....	32
Flux Solver Code FLUXES.....	62

	☒
	☐
	☐

DTIC QUALITY INSPECTED 4

Distribution/	
Availability Codes	
Dist	Avail and/or Special
A-1	

List of Figures

Figure		Page
1	Neutral Particle Beam (NPB) Platform	32
2	Generic Cylindrical Spacecraft with H2 Nozzle	33
3	H2 Number Density Around Chemical Power System	34
4	H2 Static Pressure Around Chemical Power System	35
5	SSF Final Configuration	36
6	Stage 2 of SSF Assembly	37
7	Stage 4 of SSF Assembly	38
8	SSF Arrow Mode Configuration	39
9	Neutral Density 25 lbf Monoprop Hydrazine Thruster into Wake	40
10	Static Pressure 25 lbf Monoprop Hydrazine Thruster into Wake	41
11	Temperature 25 lbf Monoprop Hydrazine Thruster into Wake	42
12	Absolute Velocity 25 lbf Monoprop Hydrazine Thruster into Wake	43
13	Axial Velocity 25 lbf Monoprop Hydrazine Thruster into Wake	44
14	Radial Velocity 25 lbf Monoprop Hydrazine Thruster into Wake	45
15	Neutral Density 25 lbf Biprop Hydrazine Thruster into Wake	46
16	Neutral Density 2.5 lbf Monoprop Hydrazine Thruster into Wake	47
17	Temperature 2.5 Monoprop Hydrazine Thruster into Wake	48
18	Neutral Density 2.5 Biprop Hydrazine Thruster into Wake	49
19	Temperature 25 lbf Monoprop Hydrazine Thruster into Wake	50
	0% Thermal Accommodation (Specular Reflection)	
20	Neutral Density 25 lbf Monoprop Hydrazine Thruster into Wake	51
	0% Thermal Accommodation (Specular Reflection)	
21	Ion Density 25 lbf Monoprop Hydrazine Thruster into Wake	52
22	O⁺ Density 25 lbf Monoprop Hydrazine Thruster into Wake	53
23	Ion Density 25 lbf Monoprop Hydrazine Thruster into Ram	54
24	N₂⁺ Density 25 lbf Monoprop Hydrazine Thruster into Ram	55
25	Ion Density 25 lbf Biprop Hydrazine Thruster into Wake	56
26	CO₂⁺ Density 25 lbf Biprop Hydrazine Thruster into Wake	57
27	Ion Density 25 lbf Biprop Hydrazine Thruster into Ram	58
28	CO₂⁺ Density 25 lbf Biprop Hydrazine Thruster into Ram	59
29	Schematic View of Solar Array Charging/Arcing Process	60
30	Total Arc Rate vs Bias Voltage for Different Densities	61

List of Acronyms

CIV.....	Critical Ionization Velocity
DSMC.....	Direct Simulation Monte Carlo
EFEE.....	Enhanced Field Emission Electrons
ESD.....	Electron Stimulated Desorption
EUV.....	Extreme Ultra Violet
IRAD.....	Independent Research and Development
LEO.....	Low Earth Orbit
MIT.....	Massachusetts Institute of Technology
MMH.....	Mono Methyl Hydrazine
MOC.....	Method of Characteristics
NPB.....	Neutral Particle Beam
OTV.....	Orbital Transfer Vehicle
PIC.....	Particle in Cell
PIX.....	Plasma Interaction Experiment
SDI.....	Strategic Defense Initiative
SSF.....	Space Station Freedom
UV.....	Ultra Violet
VRCS.....	Vernier Reaction Control System

INTRODUCTION

Any active space system will generate a self-induced environment due to the release of various effluents and the effluents' interactions with the ambient space environment. The perturbed local environment may interfere negatively with space system operations, and it is therefore important to know what environmental perturbations to expect, and predict the effect of these perturbations on space system operations. Space systems include space platforms such as Space Station Freedom (SSF) and Strategic Defense Initiative (SDI) type weapons' platforms, satellites, orbital transfer vehicles (OTVs), the Shuttle Orbiter, and future space transportation systems. The altitude regime we have focused on is low Earth orbit (LEO), roughly spanning the altitudes from 200 to 1000 km.

The main effluent source is the operation of various thrusters, which include attitude control and reboost operations, as well as venting from extremely high mass flow rate chemical power systems. Even though the thruster plume exhaust is initially neutral, it can become ionized through a series of interactions with the ambient environment. Such ionization mechanisms include charge exchange with the ambient oxygen ions, photo-ionization due to the solar extreme ultra violet (EUV) photon flux, and Critical Ionization Velocity (CIV) processes. So in addition to an ambient background plasma, there may be a local plasma enhancement due to ionization of the neutral effluents. The perturbed plasma provides an artificial environment for any high-voltage surface on the spacecraft, such as solar arrays. The high-voltage surfaces may be exposed to the space environment either through design or by erosion of insulation. For certain plasma and neutral density regimes, arcing and electric breakdown may be triggered at the high-voltage surfaces, causing sputtering and erosion of surfaces such as protective and thermal coatings. The sputtered material can redeposit on other sensitive surfaces such as solar arrays, radiators, sensors, and windows. Erosion of thermal coatings will affect the global thermal management of the space system, and the loss of protective coatings exposes the surfaces to both harmful ultra violet (UV) radiation from the Sun and to the chemically active ambient atomic oxygen, which can wear down surfaces by reacting with them. All these processes reduce the efficiency and lifetimes of space systems, and must be understood in order to effectively design a complex space system.

In this effort we have concentrated on modeling the self-induced environments expected at high-voltage solar arrays. This macroscopic environment serves as input to a microscopic model of the arcing process, developed by Professor Daniel Hastings and Dr. Mengu Cho at the Massachusetts

Institute of Technology (MIT). The goal is to provide a predictive capability for the occurrence and prevention of arcing on high-voltage surfaces.

The contract objective expressed in the Statement of Work is to "develop a model of the neutral and charged particle self-induced environment around a space platform / vehicle". To keep the analysis relevant, we have focused on two realistic effluent scenarios for space systems in LEO: (1) Massive H_2 release (kg/s mass flow rate) from an SDI-type chemical power system (e. g., Neutral Particle Beam Platform), and (2) SSF hydrazine thrusters.

After the structure geometry and thruster configuration were defined, the neutral effluent flowfields were modeled. The effluent density regimes range from continuum inside and immediately outside the thrusters, via transitional to rarefied flow beyond that. The continuum part of the flowfields were modeled with a Method of Characteristics (MOC) code under company Internal Research and Development (IRAD). The rarefied part of the plumes were characterized with direct simulation Monte Carlo (DSMC) methods, developed by Dr. G. A. Bird. The ionization of the neutral plumes was determined by solving a set of continuity equations for the neutral and plasma species.

The total calculated neutral and plasma environment was then coupled with the microscopic arcing model developed at MIT to predict arcing probability and characteristics for the modeled space system.

RESULTS

Neutral Distribution

Two realistic effluent releases have been modeled; one is the venting out of a high mass flow rate SDI-type chemical power system, and the other is the operation of various hydrazine thrusters in the SSF environment.

Chemical Power System Exhaust

One of the concepts within the SDI program is a space based neutral particle beam weapons platform (see figure 1). Such a device would accelerate negative hydrogen ions through an electron stripper, hence generating a beam of neutral hydrogen. Power levels on the order of megawatts are required to accelerate the hydrogen, and one concept is powered by a chemical power system fueled by hydrogen and oxygen. The combustion products are water and hydrogen, where the water is contained in an internal cycle, but the hydrogen is released to the ambient

ionosphere. The hydrogen gas would be vented through a series of nozzles at mass flow rates on the order kg/s for several minutes. The hydrogen molecules will become partly ionized after exit and perturb the local plasma environment. The perturbed environment can interact with the beam itself (e. g., attenuation, deflection), and with high-voltage surfaces on the space platform.

We have modeled such a release through a supersonic nozzle at a mass flow rate of 0.83 kg/s. The continuum part of the flow was modeled by an MOC code, and the transition line between continuum and rarefied flow was defined at a Bird number of 0.04. The Bird number, which is a condition for transition from continuum to rarefied flow, is discussed in more detail in the DSMC overview in Appendix A.

A schematic view of the nozzle and spacecraft is showed in figure 2. The nozzle radius is 28 cm and the nozzle half angle is 25 degrees. For the case we present here, the spacecraft radius was 50 cm, and the high-voltage solar array surfaces would be located around the structure upstream of the nozzle exit plane. The continuum properties along the Bird transition line were input to a 2-D DSMC code. The DSMC concept is described in more detail in Appendix A. Required inputs are composition, molecular number density, velocities, flow angles, and temperature. The DSMC runs were executed on a Stardent super mini-computer, and run times were on the order of 15-20 CPU hours.

The H_2 constant number density contours are shown in figure 3. For this case, the thruster was assumed to be venting into a vacuum. The DSMC code does have provisions for including the effects of an ambient free stream. The free stream has little effect on high density thruster plumes (like here), and to keep the analysis general and independent of altitude and ambient conditions, a vacuum boundary condition was implemented. The nozzle is located to the right along the z-axis, and the number densities vary from 10^{21} m^{-3} at the nozzle lip to 10^{18} m^{-3} at the arrays upstream of the nozzle exit. Typical ambient neutral densities in LEO are of the order 10^{13} to 10^{15} m^{-3} . Figure 4 depicts the static pressure around the space system, and these range from 10^{-7} to 10^{-4} torr at the structure walls. This is greater than the ambient pressure of approximately 10^{-8} to 10^{-7} torr.

SSF Hydrazine Thrusters

The SSF reboost and attitude control will be provided by 25 lbf (~111 N) and 55 lbf (~245 N) monopropellant hydrazine (N_2H_4) thrusters. In addition, the SSF environment will be affected by the Shuttle's 25 lbf bipropellant hydrazine Vernier Reaction Control System (VRCS) thrusters during Shuttle docking and berthing maneuvers. The bipropellants are fueled by MMH- N_2O_4 .

where MMH stands for monomethyl hydrazine (CH_6N_2) and N_2O_4 is the oxidizer. We have simulated both monoprop and biprop activity for a series of operating and geometrical parameters including various mass flow rates and locations with respect to high voltage surfaces. The SSF will be assembled over a total of 17 Shuttle missions, and the station orbital attitude will vary. In the final configuration (figure 5), the SSF will orbit with the arrays facing the ram and the truss perpendicular to the orbital velocity vector. During the first assembly phases shown in figures 6 and 7, however, the station will be orbiting in a so-called arrow mode, with the truss oriented along the velocity vector. To maintain orbital altitude, the station will need to reboost a few times per year during the arrow mode phase. The reboost will be provided by 25 lbf monoprops located on propulsion modules, which will be operating either over and across the solar arrays, or along the truss in the opposite direction. Figure 8 shows a scenario with the thrusters operating across one bank of solar arrays, with the approximately 10 by 30 m arrays located about 5 m downstream of the thruster exit plane. The reboost operations, which will last a few hours every few months, will generate a strongly perturbed local neutral / plasma environment at the high-voltage (160 V) solar arrays. In our analysis we have computed the expected environment to determine if arcing on the arrays is a concern.

The thrusters that were simulated have an exit radius of 1.625 in. (4.13 cm) and an expansion ratio of 100:1. To demonstrate the characteristics of biprop versus monoprop hydrazine thruster flowfields, we have also simulated SSF 25 lbf hydrazine thrusters operating on bipropellant MMH- N_2O_4 , even though this fuel mix is not used on the SSF (only on the Shuttle). The theoretical exhaust compositions for the monoprop and biprop hydrazine thrusters are shown in the following table 1:

Table 1: Theoretical Exhaust Composition of Hydrazine Thrusters	
Monopropellant Hydrazine (N_2H_4)	
Exhaust Specie	Molar Fraction
Hydrogen (H_2)	0.48
Nitrogen (N_2)	0.29
Ammonia (NH_3)	0.22
Bipropellant Hydrazine (MMH-N_2O_4)	
Exhaust Specie	Molar Fraction
Hydrogen (H_2)	0.20
Nitrogen (N_2)	0.31
Water (H_2O)	0.32
Carbon Monoxide (CO)	0.11
Carbon Dioxide (CO_2)	0.06

The hydrazine thrusters are located on propulsion modules that are replaced when the fuel is expended. Towards the end of the module lifetime, the mass flow rate decreases, and to simulate this development, thrust levels of 17, 9, and 2.5 lbf were modeled in addition to the nominal 25 lbf. Only the 25 and 2.5 lbf flowfields will be presented here; the 17 and 9 lbf levels were presented in the annual technical report of 31 July, 1991.

The thruster locations with respect to the solar arrays are illustrated in figure 8. The arrays will constantly be tracking the Sun, however, during reboost, the arrays will be feathered to some degree to avoid having the thrusters firing directly at the arrays. The arrays rotate both in alpha and beta directions, and the local neutral densities at the surface of the arrays will be a function of array attitude. The case that was looked at here has the array located in a horizontal position with an angle alpha of 90 degrees, and an angle beta of 0 degrees.

The SSF is traveling at an orbital velocity of approximately 7.7 km/s. This is expressed in the DSMC simulation as a free stream of atomic oxygen flowing past the arrays along the y-axis in figure 8. At LEO altitudes, atomic oxygen is the predominant neutral species with a number density of about $10^8 - 10^9 \text{ cm}^{-3}$. The nominal temperature of the arrays varies between 199 and 333 K, depending on the solar heating; most of the simulations were done for a nominal array temperature of 250 K. For most of the cases full thermal accommodation at the array surface has been assumed. The effect of varying the accommodation coefficient is addressed later on.

25 lbf Monopropellant Hydrazine Thruster

This is the thruster that will actually be used to provide reboost for the SSF during the arrow mode. The continuum part of the flowfield was calculated using MOC methods starting inside the thruster. The flow properties (composition, density, vector velocity, temperature) along a line separating continuum and transitional flow were used as input to the DSMC simulation. Figure 9 shows the constant number density contours for this thruster. The ambient free stream is from left to right, so the thruster is operating into the wake of the SSF. The simulated area is axis-symmetric with the thruster center line being the symmetry axis. The continuum part of the flow immediately outside the thruster exit shows up as a white region, and the white lines and streaks in the color plots only signify various computational regions in the DSMC simulation; the separation between regions is not physical, but purely a result of the way the color graphics software interpolates the data (or rather the lack of interpolation). In the simulation itself, the regions are continuous.

The exhaust plume initially exhibits a nominal expansion and density drop, but downstream, close to the solar array (which is oriented out of the paper plane), a density "pile-up" is evident, causing the local neutral molecular number density to reach levels of 10^{20} m^{-3} , which is six orders of magnitude greater than the ambient density. Note that this is the *total* neutral density. The monoprop hydrazine exhaust consists of hydrogen, nitrogen, and ammonia, and the relatively lighter hydrogen will show a greater expansion into the backflow region than the other two species. The DSMC code does keep track of the various species, but for our analysis, we have just assumed the species molar fraction to be fixed at all regions of the flowfield. Figures 10 and 11 show the static pressure and the temperature for the exhaust plume. The static pressure is just $p = 1/2 \rho V^2$, where ρ is the mass density, and V is the velocity of the flow. The static pressure can be seen to first decrease out of the nozzle to levels of 10^{-5} torr, before building up towards the array, reaching values of 0.01 torr. This compares to ambient pressures of about 10^{-7} torr. The temperature is really an average temperature averaged over the translational, rotational, and vibrational temperatures of the molecules. For this particular case, full thermal accommodation was assumed, which means that the flow incident to the array surface is brought to the array surface temperature. The flow initially cools off as it expands out of the thruster nozzle, but as the flow slows down and stagnates towards the array surface, it heats up, due to the conversion of molecular translational kinetic energy to internal energy. The flow heats up to about 1700 K in a layer along the array, before it cools off again, this time being accommodated to the array surface temperature fixed at 250 K. The net heat flux to the solar array surface varies from 26 W/m^2 (3 m downstream of the thruster) to 191 W/m^2 (8 m downstream of the thruster).

Figure 12 shows the absolute exhaust velocity for an array temperature of 333 K. The flow initially expands out of the nozzle at a speed of approximately 3 km/s. The orbital velocity of the SSF is 7.7 km/s, which means that the exhaust travels at $7.7 - 3 = 4.7 \text{ km/s}$ with respect to the ambient free stream. The flow slows down as it approaches the array, becoming stagnant (with respect to the array) about 2 m downstream of the nozzle exit plane. Figures 13 and 14 show the axial and radial components of the flow, respectively. Right outside the thruster exit plane, 4 m off the symmetry axis, one can see how the axial velocity component is reversed, causing the exhaust to expand into the backflow at speeds as high as 0.8 km/s; this will impact the CIV ionization in this region, as will be seen later. The radial velocity component is zero along the thruster axis, and about 3 km/s off the nozzle lip. Towards the array the axial component is suppressed again, showing reversal and negative axial velocity close to the surface.

25 lbf Bipropellant MMH-N₂O₄ Thruster

This next case is a simulation of an SSF thruster fueled by bipropellant hydrazine, which is what the Shuttle VRCS thrusters run on. The flowfield number densities shown in figure 15 resemble the ones for the monoprop, showing a density "pile-up" at the array 8 m downstream of the thruster exit plane. The colored streaks in this flowfield are attributed to numerical non-physical phenomena. The exhaust number densities for the biprop are generally lower than for the monoprop by almost half an order of magnitude. This can be attributed to the difference in the mass of the exhaust species; the biprop mixture has a lower mass per mole than the monoprop, which causes it to expand faster and which ultimately leads to a less dense plume.

2.5 lbf Monopropellant Hydrazine Thruster

This is basically a SSF thruster running at a mass flow rate one order of magnitude less than nominal. Figure 16 shows the constant number density contours for this case; the solar array inhibits the free expansion of the thruster plume, but there is no substantial density build-up at the array as demonstrated for the 25 lbf case. The flow temperatures are displayed in figure 17 and they are, as one would expect, significantly lower than for the 25 lbf case. A one order of magnitude decrease in the mass flow rate causes the temperature to be cut in half, reaching approximately 750 K in a layer 7 m downstream of the thruster exit plane.

2.5 lbf Bipropellant MMH-N₂O₄ Thruster

Figure 18 shows the constant number density contours for this case. Again, the overall flow density is reduced compared to the 25 lbf case. By comparing with the 2.5 lbf monoprop densities shown in figure 16, we see the effect of the lower plume molecular mass, causing the lighter biprop plume to expand faster and become more rarefied.

Surface Accommodation

The effect of varying the array surface accommodation was also looked at. The concept of surface accommodation is still not completely understood, and any conservative analysis should check the effect the accommodation coefficient on the flow properties.

A gas in contact with a solid surface is one of the most frequently occurring boundary conditions. Unfortunately, it is also one of the least well understood. Despite a great deal of theoretical and experimental study, the simple models of specular and diffuse reflection that were originally put forward by Maxwell in 1879 remain the most generally useful models for practical applications [Ref. 1].

Specular reflection is perfectly elastic with the molecular velocity component normal to the surface being reversed, while that parallel to the surface remains unchanged. This is a useful reference state for analytical studies, but does not occur for real gas-solid combinations. With this model, there is no thermal or viscous boundary layer adjacent to a smooth solid surface.

In diffuse reflection the velocity of each molecule after reflection is independent of its incident velocity. However, the velocities of the reflected molecules as a whole are distributed according to the half-range equilibrium Maxwellian distribution for the molecules that are directed away from the surface. This distribution is for the temperature T_r which may differ from the temperature T_w of the surface. The extent to which the reflected molecules have their temperature adjusted toward that of the surface is indicated by the accommodation coefficient a_c , defined by

$$a_c = \frac{q_i - q_r}{q_i - q_w}$$

Here, q_i and q_r are respectively the incident and reflected energy fluxes, while q_w is the energy flux that would be carried away in diffuse reflection with $T_r = T_w$. The range of a_c is from zero for no accommodation to unity for complete thermal accommodation.

Experiments with engineering surfaces in contact with gases at temperatures on the order of the standard temperature indicate that the reflection process approximates diffuse reflection with complete thermal accommodation. This may be a consequence of such surfaces being microscopically rough with the incident molecule suffering multiple scattering, or of the molecules being momentarily trapped or adsorbed on the surface (e. g., if the molecules react chemically with the surface). On the other hand, experiments with carefully prepared and cleaned surfaces indicate that the accommodation coefficient can be significantly less than one. As the translational energy of the incident molecules relative to the surface becomes large in comparison with the value corresponding to the surface temperature, complete thermal accommodation becomes less readily achieved.

In figure 11 we saw the flow temperature for full thermal accommodation, i. e., an accommodation coefficient of one, at the array. The solar array surface is at 250 degrees K, whereas the flow temperature is much hotter, reaching peaks of approximately 1700 K. The molecules hitting the surface are cooled down to the surface temperature (250 K) - this is evident in figure 11 where there is a relatively cool temperature layer next to the array. For a specular reflection case, the molecules hit and leave the surface without changing temperature, and the gas

heats up towards the surface as it is being slowed down, converting kinetic energy to internal energy. This is demonstrated in figure 19.

The effect of thermal accommodation on the local density is displayed in figures 9 and 20. According to the perfect gas law, $pV = nRT$, when the temperature drops, the density increases - and vice versa. So for the case with complete thermal accommodation, the relatively lower temperature along the array leads to a higher local density. The density "pile-up" in figure 9 can thus be seen to be about a factor of two greater than in figure 20. Subsequently, if the local density at the array was barely enough to trigger arcing on the high-voltage array, a factor of two decrease in density could help keep the array from arcing. In that sense, it would be preferable to have a surface of thermal accommodation coefficient zero, which is a surface exhibiting specular reflection. It should also be noted that for full thermal accommodation, varying the array temperature from 199 to 333 K (min and max due to solar heating), had no detectable effect on the local flow number density.

Ways to keep a surface as specularly reflecting as possible are:

(1) Maintain smooth surface.

- One could presumably use only surfaces that are smooth, however, one must also remember that in space surfaces may be subject to micrometeoroid impacts, chemical erosion, and UV deformation - which all tend to make the surface pitted and non-uniform.

(2) Chemically inactive or inert surfaces.

- This obviously depends on the incident molecules too. Atomic oxygen is very reactive and would be hard to keep inactive. However, the solar array is made up of solar cells with cover glass and protective coating, metallic interconnects, and supporting structure. Both copper and aluminum interconnects rapidly form oxides with the atomic oxygen, and this should halt further chemical activity involving oxygen on the surface. The cover glass coatings also form oxides.

However, for the case with the thruster operating along the array, the incident molecules are predominantly the exhaust species, which are H_2 , N_2 , and NH_3 for monoprop hydrazine, and H_2 , N_2 , H_2O , CO , and CO_2 for biprop MMH- N_2O_4 . Neither N_2 nor H_2 are very reactive, which should indicate a greater degree of specular reflection than for a pure atomic oxygen environment. Water and ammonia have chemical reactivities somewhere between atomic oxygen and nitrogen / hydrogen.

In conclusion, if the spacecraft designer expects density conditions that are borderline arc triggering, it is worthwhile to employ specular reflecting materials. Research and experiments to determine the accommodation coefficients for relevant materials would be needed in order to pick optimum materials. Thermal accommodation furthermore causes drag, which is another reason for studying the issue. I am not aware of any such optimization taking place in the community today.

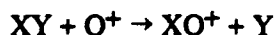
Ionization of the Neutral Effluents

Having computed the neutral flowfield distributions, we can now turn on the various possible ionization processes. Ionization of the neutral effluents will perturb the local plasma distribution and possibly affect arcing thresholds and arcing frequencies at the high-voltage solar arrays.

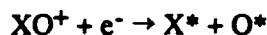
The neutrals can become ionized through a series of interactions with the ambient environment. The most important ionization mechanisms are:

Ion-Molecule Reactions

These reactions lead to the exchange of charge between species. Charge exchange reactions include charge transfer processes in which one or more electrons are transferred, as well as ion-atom or ion-molecule interchange (charged rearrangement) processes in which a charged or neutral atomic or molecular species is transferred similar to an ordinary chemical reaction [Ref. 2]. Molecules such as H_2 , H_2O , and CO_2 react with ionospheric oxygen ions in a charge transfer process as follows [Ref. 3]:



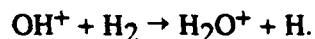
where XY is the released molecule, and XO^+ is a positive molecular ion which can rapidly recombine with electrons. Dissociative recombination is represented by the reaction



where X^* and O^* are electronically excited states which will radiate visible light. There is of course also straightforward recombination such as:



An example of an ion-molecule interchange is [Ref. 4]:



There is no momentum exchange or sharing of kinetic energy in the charge transfer process. Furthermore, charge exchange does not provide net ionization gain in a global sense, however, if a fast (7.7 km/s orbital velocity) ambient O^+ plasma is replaced by a slower (effluent exit velocity ~ 3 km/s) local plasma of ionized effluents, the local plasma density will increase.

Photo-Ionization

A solar UV photon of energy $h\nu$, where h is the Planck constant and ν is the photon frequency, can be absorbed by an atom or molecule, exciting it to a higher state, or photo-ionizing it when $h\nu$ is greater than the ionization energy [Ref. 5]:



Dissociative photo-ionization is an additional source of atomic ions [Ref. 6]:



In the latter case, the photons must have enough energy to both ionize and dissociate the molecule. The ion production rate, q , depends on the number density of the ionizable constituent, n , its ionization cross section for the particular photon wavelength interval, σ , and the local photon flux, ϕ :

$$q = \sigma n \phi$$

The solar EUV flux as a function of wavelength is well documented. The ionizing photon flux decreases from its value outside the atmosphere as the result of absorption. The optical depth, $\tau(\lambda)$, is a parameter that specifies the attenuation of solar irradiance by the atmosphere, and the ion production rate, adjusted for absorption, can be expressed as:

$$q = \sigma n(z) \phi \exp(-\tau)$$

where z depicts the altitude. Hence, the ionization rate decreases by a factor e for an optical depth of 1. The ion production rate has a maximum where $dq/dz = 0$, which corresponds to the condition $\tau = 1$ [Ref. 2]

The photo-ionization process is usually slower than the charge exchange processes, and all species with a photo lifetime of 10^5 s or greater will in general be depleted more rapidly by chemical or transport processes than by photolysis [Ref. 4]. Species like H_2 and CO_2 have photo-ionization rates of about $10^{-7} s^{-1}$, which corresponds to photo lifetimes of $1/10^{-7} = 10^7$ s. Shorter lifetimes are observed for H_2O (10^5 s) and $Ni(CO)_4$ (100 s).

Critical Ionization Velocity (CIV)

The concept of CIV was first introduced by *Alfvén* [Ref. 7] in 1954 as a component of his theory for the formation of the solar system. As a neutral gas moves through a magnetized plasma, a rapid ionization of the neutrals takes place if the kinetic energy of the neutrals relative to the plasma exceeds the ionization potential $e\phi$ of the neutrals. This then defines a critical velocity, V_{civ} .

$$\frac{1}{2} M V_{civ}^2 = e\phi$$

where M is the neutral mass, above which CIV ionization will take place. Although equating the neutral kinetic energy to the ionization potential in the calculation of a threshold for ionization of the neutrals may appear at first glance to be an obvious step, it is important to keep in mind that the kinetic energy needs to be at least twice that given in the equation above for an ionizing ion-neutral *collision* to occur (as is easily seen in the center of mass frame). This is because in collisions between ions and neutrals, only a fraction (one-half for equal mass ion and neutral) of the kinetic energy of the collision is available for ionization of the neutral [Ref. 8]. However, strong evidence has been put forward in lab experiments that CIV ionization does indeed occur and that the threshold is quite near or only slightly above that predicted by the above equation.

The CIV process depends on several steps [Ref. 9]. Some mechanism, such as charge exchange between the fast neutrals and cold plasma ions, or photo-ionization of a fraction of the neutrals, creates energetic ions. The relative motion between these ions and the plasma electrons creates an unstable situation which gives rise to the growth of plasma waves. These waves serve to remove energy from the fast ions and transfer it to the electrons. The electrons in the tail of the energy distribution with energy higher than $e\phi$ can ionize neutrals. This creates additional ions moving at the neutral velocity. These newborn ions feed energy into the unstable waves, which in turn further heat the electrons, thereby allowing the cyclic energy transfer to be sustained. The exponential increase in plasma density is a consequence of this energy transfer cycle.

Attempts to reproduce the CIV effect in space experiments have provided mixed success, but research is still going strong trying to demonstrate the CIV process in space and unravel the process. The most convincing evidence that the CIV effect does occur in a space plasma is from the Project Porcupine flight reported by *Haerendel* [Ref. 10]. A neutral barium ($V_{civ} = 2.7$ km/s) release demonstrated initial ionization of $30 \pm 10\%$ of the total neutral population that had a transverse velocity greater than the critical velocity. Other ionization mechanisms than CIV were ruled out as insufficient.

The velocities required to trigger CIV ionization are often too high for the effect to be important in LEO. This is because the critical velocities often exceed the orbital velocity (7-8 km/s) of the space system and its effluents. However, by thrusting into the ram direction, the exhaust plume velocities with respect to the ambient magnetized plasma becomes the sum of the orbital and the thruster exit velocities which may well exceed the critical velocity. Table 2 shows the critical ionization velocities for a few common exhaust species:

Table 2: Critical Velocities for a few Common Hydrazine Exhaust Species	
Species	V_{civ} [km/s]
H2	38.5 [Ref. 11]
N2	10.3 [Ref. 12]
NH3	10.7 [Ref. 12]
H2O	11.6 [Ref. 12]
CO	9.7 [Ref. 11]
CO2	7.7 [Ref. 12]

For the numerical simulations of the ionization CIV, we have taken advantage of ionization rates calculated by Dr. Rodger Biasca, formerly of MIT, but currently at the Phillips Lab/Hanscom Air Force Base [Ref. 12,13]. These rates were obtained from a one-dimensional implicit Particle-in-Cell (PIC) simulation assuming a constant density, infinity background of neutrals. The implicit PIC code is different from previous PIC simulations reported in the literature in that the electron-ion mass ratios are real. The simulations assume a neutral beam (corresponding to the thruster plume) propagating across a magnetic field in a background plasma. It was found that the electron number density increases as roughly

$$N_e(t)/N_e(0) = e^{v_{civ} t}$$

where v_{civ} is the anomalous or CIV ionization rate. This can be rewritten as:

$$\Delta N_e = N_{e(0)}(e^{v_{civ} t} - 1)$$

Currently, only Xe, Ne, CO₂, NO, and N₂ have been simulated, of which CO₂ and N₂ are applicable to the hydrazine thruster plumes we are modeling. The ionization rates computed by Biasca have been plotted as a function of both neutral velocity and neutral density, and we have determined numerical rates based on functional fitting to the plots. Obviously, no CIV ionization occurs at velocities below the critical ionization velocity, and the rates increase quadratically or faster with increasing neutral velocity. The rates are also proportional to the neutral density up to a point where the neutral number density is about 2×10^7 times the initial electron density. For greater neutral densities, the electron collision rates become on the order of the lower hybrid frequency, in which case it is difficult for the electrons to develop the high energy tail on the electron distribution function that plays such a big role in CIV. The functionally fitted CIV rates for CO₂ and N₂ are of the form

$$v/\Omega = (K + n_n/n_e)(V_n/V_{civ})^5$$

where Ω is the ion gyro frequency, K is a constant, n_n and n_e are the neutral and electron densities, respectively, and V_n and V_{civ} are the neutral velocity and the critical ionization velocity, respectively. The fitted CIV rates are shown in table 3:

Table 3: Functionally Fitted CIV Ionization Rates (Valid for $V_n > V_{civ}$ only)	
$x = V_n / V_{civ}$ and $y = N_n / N_{e(0)}$	
CO₂:	
Neutral Density Regime	v_{civ} / Ω_{CO2}
$1e6 < N_n/N_{e(0)} < 2.5e7$	$[0.7 + 0.45(y/1.e6 - 1)] (x/1.5)^5$
$2.5e7 < N_n/N_{e(0)} < 7e7$	$[1.4 + 15.6(1 - y/7e7)] (x/1.5)^5$
N₂:	
Neutral Density Regime	v_{civ} / Ω_{N2}
$1e6 < N_n/N_{e(0)} < 1.5e7$	$[0.33 + 0.5(y/1e6 - 1)] (x/1.5)^5$
$1.5e7 < N_n/N_{e(0)} < 6e7$	$[0.6 + 7.2(1 - y/6e7)] (x/1.5)^5$

Note that these rates are valid only when the neutral velocity is equal to or greater than the CIV velocity! The rates can be seen to be proportional to the neutral density, and a very strong function of the ratio between neutral and CIV velocity. Numerical rates are unfortunately not

available for CO and NH₃, which with critical ionization velocities of 9.7 and 10.7 km/s, respectively, could well exhibit CIV effects at orbital velocities.

Numerical Simulation of the Ionization Process

In order to simulate the ionization of the neutral thruster effluents, we established a set of 2-D continuity equations for all neutral and ion species in the flow. The set of hyperbolic equations were of the form:

$$dn/dt + df/dx + df/dy = S - L$$

where n denotes the neutral or ion density, f denotes the 2-D flux, and S and L are source and loss terms, respectively. This set of ion and neutral continuity equations were solved by time advancing the flux through a computational grid, using the donor-cell method. The solution is first order in both time and space. In order for the numerical solution to be stable, the time step must obey the Courant stability condition, which is expressed as

$$|V|\Delta t / \Delta x < 1$$

where V is the flux velocity, Δt is the time step, and Δx is the spatial step.

The initial neutral densities and vector velocities were provided by the DSMC runs in a semi-random data grid format. This data needed to be organized before it was input to the ionization model. A set of DI-3000 subroutines, available on our VAX work station, were combined to create an interpolated organized neutral grid. The gridding algorithm used a triangulation scheme, followed by bivariate interpolation with a fifth-degree polynomial.

The fact that we are looking at plume induced effects occurring along the solar arrays means that we are interested in ionization occurring only on time scales on the order of the plume exhaust propagation time passed the solar arrays. For thruster exit velocities on the order 3 km/s, and a solar array dimension of 10.5 m, it takes the neutral effluents about 3.5 milliseconds to blow past the array. Ionization happening after that will not influence the local environment at the array.

Some of the features of our ionization model are as follows:

- New ions created as a result of ion-molecule reactions, photo-ionization, and CIV effects were assumed to keep traveling at their initial neutral velocities, i. e., no collisional momentum was exchanged.
- Magnetic effects were omitted, which is acceptable at the dimensions we are looking at. The ion gyro radii range from about 20 to 100 m, and the gyro periods are between 20 and 60 ms.
- The effect of the solar array voltage on the ions is not modeled. However, for negative array voltages and positive ions, the newly formed ions will be attracted by the array, which will cause the local ion density to increase.
- The ambient plasma is assumed to be O^+ . The directional velocity of the O^+ can be varied to simulate ram and wake effects, i. e., the thruster operating in different directions. The ambient O^+ density is initially set to a constant number everywhere in the numerical grid; once the ion-molecule reactions start, the O^+ starts to deplete, but new ambient O^+ is supplied with the ambient free stream.
- The electron density was initially set equal to the total ion density after each time step. This implies that there are enough electrons available at all times to neutralize the newly created local plasma. This is usually not a bad assumption, however, for cases when the CIV effect really takes off and the ionization rates grow exponentially with time, one must verify that the ambient electron flux is sufficient to keep the newly created local plasma overall charge neutral at all times.

25 lbf Monopropellant Hydrazine Thruster

As shown in table 1, the exhaust plume for this thruster consists of H_2 , N_2 , and NH_3 . The various ionization processes these species are subject to, are shown in table 4:

Table 4: Ionization Rates Monopropellant Hydrazine			
Ion-Molecule Reactions			
Reaction		Reaction Rate [cm3/s]	
O+ + H2 → OH+ + H		1.7e-9	[Ref. 4]
OH+ + e- → O* + H		7.5e-8 (300/Te) ^{1/2}	[Ref. 4]
OH+ + H2 → H2O+ + H		1.5e-9	[Ref. 4]
H2O+ + e- → OH* + H		6.5e-7 (300/Te) ^{1/2}	[Ref. 4]
O+ + N2 → NO+ + N		3.0e-10 (at 5 eV)	[Ref. 14]
NO+ + e- → N + O		2.35e-8	
O+ + NH3 → O + NH3+		1.2e-9	[Ref. 15]
NH3+ + e- → NH3		1.0e-7	(estimated)
CIV Reactions			
N2 → N2+		See Table 3	
Photo-Ionization			
EUV Wavelength [Ångstrom]	Ionization Cross Section [cm ²]	Incident Photon Flux [photons/cm ² -s]	Calculated Reaction Rate [cm ³ /s]
H2 + hν → H2 ⁺ + e ⁻			
800 - 630	6e-18 [Ref. 2]	2.4e9	1.8e-7
630 - 460	5e-18 [Ref. 2]	4.7e9	1.5e-7
460 - 370	4e-18 [Ref. 2]	0.63e9	1.2e-7
N2 + hν → N2 ⁺ + e ⁻			
800 - 630	21e-18 [Ref. 16]	2.4e9	6.3e-7
630 - 460	22e-18 [Ref. 16]	4.7e9	6.6e-7
460 - 370	19e-18 [Ref. 16]	0.63e9	5.7e-7
370 - 270	12e-18 [Ref. 16]	10.3e9	3.6e-7
270 - 205	6e-18 [Ref. 16]	4.4e9	1.8e-7
NH3 + hν → NH3 ⁺ + e ⁻			
1027 - 911	18e-18 [Ref. 17]	11.61e9	5.4e-7
911 - 800	25e-18 [Ref. 17]	8.3e9	7.5e-7
800 - 630	30e-18 [Ref. 17]	2.4e9	9.0e-7
630 - 550	25e-18 [Ref. 17]	4.7e9	7.5e-7

Figure 21 shows the computed total ion distribution along the solar array. The array is located 5 m off the thruster center axis, and extends about 10.5 m downstream of the thruster exit plane. For this simulation, the thruster was operating into the wake of the space station, and the simulation time, which corresponds to real flow time for the plume exhaust, was 3.5 ms. The ambient O^+ number density was set to 10^6 cm^{-3} which represents a high value for the

ionosphere in LEO. Total ion densities along the array can be seen to reach values as high as $3 \times 10^7 \text{ cm}^{-3}$, which is a 13 times increase over the initial O^+ density. Because the thruster is firing into the wake at 3 km/s, the relative velocity of the plume exhaust with respect to the orbital free stream at 7.7 km/s is only $7.7 - 3 = 4.7 \text{ km/s}$; this is too slow to trigger CIV effects. The reason for the local plasma build-up is that the relatively slow ($\sim 3 \text{ km/s}$) neutral exhaust plume loses electrons to the faster (7.7 km/s) O^+ ions, so the slow moving neutrals become a slow moving plasma. Figure 22 shows how the O^+ entering from the left has been depleted downstream of the thruster exit plane through ion-molecule reactions. As the O^+ becomes more depleted downstream, the plasma production due to charge exchange decreases. That is why figure 21 shows peak plasma density immediately downstream of the thruster exit plane.

This simulation did not include photo-ionization, even though the numerical code includes that option. On the time scales we are looking at (3.5 ms), photo-ionization is too slow to have any effect as opposed to the much quicker ion-molecule reactions. For an ambient O^+ velocity of 7.7 km/s, and a grid cell dimension of 10 cm, the Courant stability condition requires that the time step be less than 10^{-5} s for the numerical scheme to be stable. For this simulation we used a time step of 10^{-6} s . If the photo-ionization were to be included, the velocity in the Courant stability condition would be the photon velocity, which is the speed of light, $3 \times 10^5 \text{ km/s}$. The time step must then be on the order 10^{-10} s , and the number of computational steps to simulate 3.5 ms increases several orders of magnitude. To keep the computational effort reasonable, we thus excluded photo-ionization from all the 3.5 ms simulations.

Another way to include the photo-ionization, is just to define a photo-ion production rate for each plume species, i. e., a fixed number of neutral molecules photo-ionized per second. This keeps the photon flux out of the calculation all together, assuming that the photon flux is the undepleted solar EUV flux everywhere at all times. *Bernhardt et al.* [Ref. 3] report that a neutral barium cloud becomes optically thick to the Sun's ionizing EUV radiation for number densities greater than $N_{\text{max}} = 10^8 \text{ cm}^{-3}$. This effect can be approximated by limiting the photo-ion production rate to $P_{\text{max}} = (N_{\text{max}}) (\nu)$, where ν is the ion production rate per second. This prevents the creation of unrealistically large plasma densities in the neutral "pile-up" region along the solar array seen in figure 9. A neutral effluent flow of number density 10^{20} m^{-3} becomes optically thick after 5 - 15 cm, depending on the neutral species. The optical depth is inversely proportional to the number density. Clearly, the optical thickness of the neutral exhaust must be taken into account to compute the photo-ionization properly.

Figure 23 shows the total ion density for the same thruster operating into the ram. The ambient O^+ is now entering from the right, and the plume-ambient relative velocity is $7.7 + 3 = 10.7$ km/s. This velocity is greater than the critical ionization velocity for N_2 , so we can expect to see some CIV effects here. The maximum ion density has not changed noticeably from the wake firing case, but the spatial distribution of the ions have. The maximum charge exchange activity takes place 10 m downstream of the thruster where the O^+ is still undepleted. In addition there is a local plasma density peak 10 m downstream of the thruster and about 3 m off the thruster center axis; this is due to CIV effects. The next figure 24, where only the CIV produced N_2^+ ions are plotted, illuminates this even better. The axial velocity of the neutrals is greatest close to the center axis, decreasing radially towards the solar array. That means that the relative velocity between the neutral exhaust and the ambient O^+ is greatest close to the center axis, which again means that CIV ionization is maximized close to the center axis. This trend is opposed by the neutral density dependence of the CIV process. Because CIV ionization is proportional to the neutral density (up to a point), the density dependent part of the CIV rate will increase with the neutral density toward the solar array (see figure 9 on neutral densities). The combined effect of the velocity and density dependent parts of the CIV rate, results in the wedge like plasma density peak seen in figure 23.

According to the CIV process as it was described in the CIV section, the process is initiated by fast seed ions ("fast" relative to the ambient plasma) generated by some ionization process such as charge exchange between the fast neutral exhaust and the slower ambient O^+ . These seed ions go on to heat the electrons through plasma collective effects, and the high-energy tail of the electrons ionize new fast neutrals. There is presumably an amount of time required for the necessary number of seed ions to be produced, before the CIV process really takes off. In our simulations, we have assumed that the production of fast seed ions and the CIV process take place simultaneously. If this is not the case and if the time to produce sufficient seed ions exceeds 3.5 ms, then CIV ionization will not happen in time for it to be important to the solar array arcing calculations.

25 lbf Bipropellant MMH- N_2O_4 Thruster

We repeated the ionization calculations for the bipropellant MMH- N_2O_4 thruster. The bipropellant thruster plume has a different species composition than the monopropellant, consisting of H_2 , N_2 , H_2O , CO , and CO_2 . The biprop plume exhibits the same set of ion-molecule reactions as the monoprop, except the reactions involving NH_3 . The additional ion-molecule reactions, and the CIV and photo-ionization reactions experienced by the biprop are listed in the following table:

Table 5: Ionization Rates Bipropellant Hydrazine			
Ion-Molecule Reactions			
Reaction		Reaction Rate [cm ³ /s]	
O ⁺ + H ₂ O → H ₂ O ⁺ + O		6.0e-10 (at 5 eV)	[Ref. 14]
H ₂ O + H ₂ O ⁺ → H ₃ O ⁺ + OH		1.67e-9	[Ref. 4]
H ₃ O ⁺ + e ⁻ → Neutrals		6.3e-7 (300/Te) ^{1/2}	[Ref. 4]
O ⁺ + CO ₂ → CO ₂ ⁺ + O		6.0e-10	[Ref. 14]
CO ₂ ⁺ + e ⁻ → CO ⁺ + O ⁺		3.8e-7 (at 5 eV)	[Ref. 3]
O ⁺ + CO → CO ⁺ + O		6.0e-11 (at 5 eV)	[Ref. 14]
O ₂ ⁺ + e ⁻ → O(1D) + O(1P)		1.9e-7 (Te/300) ^{1/2}	[Ref. 6]
CIV Reactions			
CO ₂ → CO ₂ ⁺		See Table 3	
Photo-Ionization			
EUV Wavelength [Ångstrom]	Ionization Cross Section [cm ²]	Incident Photon Flux [photons/cm ² -s]	Calculated Reaction Rate [cm ³ /s]
H ₂ O + hν → H ₂ O ⁺ + e ⁻			
1027 - 911	14e-18 [Ref. 17]	11.61e9	4.2e-7
911 - 800	15e-18 [Ref. 17]	8.3e9	4.5e-7
CO + hν → CO ⁺ + e ⁻			
885 - 800	16e-18 [Ref. 16]	6.4e9	4.8e-7
800 - 630	20e-18 [Ref. 16]	2.4e9	6.0e-7
630 - 460	17e-18 [Ref. 16]	4.7e9	5.1e-7
460 - 370	14e-18 [Ref. 16]	0.63e9	4.2e-7
CO ₂ + hν → CO ₂ ⁺ + e ⁻			
885 - 800	15e-18 [Ref. 16]	6.4e9	4.5e-7
800 - 630	20e-18 [Ref. 16]	2.4e9	6.0e-7
630 - 460	25e-18 [Ref. 16]	4.7e9	7.5e-7
460 - 370	25e-18 [Ref. 16]	0.63e9	7.5e-7

Figure 25 shows the total ion density for a bipropellant hydrazine thruster firing into the wake. The ambient O^+ number density is still 10^6 cm^{-3} , and the simulation time was 3.5 ms. The peak plasma density is similar to the monopropellant run (figure 21), reaching values as high as $3 \times 10^7 \text{ cm}^{-3}$ along the solar array right after the thruster exit plane. Note that the density color scales are different on figures 21 and 25, with figure 25 displaying better resolution. The distinctive plasma density peak in figure 25, is due to local CIV effects. The neutral axial exhaust velocity is zero in this region (see figure 13), as the plume is deflected by the solar array into the backflow. That means that the relative velocity between the neutrals and the ambient O^+ is just the ambient orbital velocity in this region, or 7.7 km/s. This is sufficient to CIV ionize the CO_2 present in the biprop plume, which has a critical ionization velocity of 7.6 km/s. The CIV

ionization of the CO_2 is even clearer in figure 26, which shows the number density of the CO_2^+ ions only.

In this next case, we simulate the 25 lbf biprop hydrazine thruster firing into the ram direction. Since this plume contains the CIV ionization prone CO_2 , we expect to see substantial CIV activity. The total plasma density after a 3.5 ms simulation is plotted in figure 27, with the ambient O^+ entering from the left. The plot shows an enormous plasma density increase, with peak densities at the array, 10 m downstream of the thruster, as high as 10^{10} cm^{-3} , which is a four order of magnitude increase over the ambient O^+ density of 10^6 cm^{-3} ! For this case, the relative velocity between the neutral exhaust and the free stream O^+ exceeds 10 km/s in most of the simulated region, being as high as 10.7 km/s in certain areas. Not only is the CO_2 , with a critical velocity of 7.6 km/s, CIV ionized almost everywhere, but N_2 , with a critical velocity of 10.3 km/s, is also subject to CIV ionization in some places. The extent of the CO_2 ionization is clearly seen in figure 28, where the CO_2^+ ion density is plotted; the ion population is almost exclusively made up of CO_2^+ .

How realistic are these tremendous CIV ionization gains? In the following some of the numerical assumptions are addressed, and comparisons to results in the literature are made:

- As we have already mentioned, we do not include any initial time in the simulation for the production of fast seed ions ("fast" relative to the ambient plasma) from the neutrals. We assume the production of seed ions (from charge exchange) and the CIV process to start simultaneously. If this seed ion production time is substantial compared to the 3.5 ms simulation time, the CIV ionization will not kick in until the exhaust effluents have blown past the 10 m solar array, thereby lowering the plasma density at the array.
- For the runs presented here, we assumed that the newly CIV created plasma was charge neutralized immediately by electrons, assuming an ample supply of electrons everywhere. This in it itself is a reasonable assumption, however, since the electron density is part of the CIV ionization rate equation, a run-away effect is created. The more CIV ionized plasma that is produced, the more electrons are needed to charge neutralize the plasma, and the greater the CIV ionization rates become. As the equation for the CIV ionization rate shows, the CIV plasma increase is a direct function of the electron density at $t=0$, and exponentially proportional with time. The electron density at $t=0$ was not held constant, rather, after each time step of the numerical simulation, it was reset to the current electron density. After discussions with Biasca [Ref. 13] it was felt that this was a reasonable assumption.

- The CIV ionization did have an upper density cutoff, as discussed in the section on CIV, and as shown in the numerical rate simulations by *Biasca et al.* [Ref. 12].
- In the simulation, it was assumed that all neutrals that exceeded the critical ionization velocity, were subject to anomalous CIV ionization. It has been pointed out in the literature [Ref. 8] that the critical ionization velocity is really an optimistic minimum, and that if there is no external power source to provide the kinetic energy to the neutrals, then the CIV ionization rate will exhibit a less than unit efficiency. In the thruster case modeled here, however, the neutral exhaust molecules are energized by the thruster activity, and we feel that the CIV ionization efficiency should be 100%.
- *Newell* [Ref. 8] reports that in the Project Porcupine space CIV experiment, which released barium into the ionosphere at velocities in excess of its critical velocity of 2.7 km/s, the rapid initial ionization was observed to be $30 \pm 10\%$ of the total neutral population that had a transverse velocity greater than 2.7 km/s. This ionization rate, which presumably is due to CIV effects, is truly enormous! How does that compare to our results? For the biprop hydrazine thruster firing into the ram, we obtained CO_2^+ ion densities of 10^{10} cm^{-3} . The neutral plume exhaust density is on the average about 10^{19} m^{-3} , and if the CO_2 molar fraction is 0.06, that amounts to a CO_2 number density of $6 \times 10^{17} \text{ m}^{-3}$ or $6 \times 10^{11} \text{ cm}^{-3}$. If about 80% of the CO_2 molecules have velocities in excess of the critical velocity, that leaves us with about $5 \times 10^{11} \text{ cm}^{-3}$. Let us furthermore assume that the CIV ionization is somewhat lower than that observed by Porcupine, or 20%. The CIV ionization then becomes 20% of $5 \times 10^{11} \text{ cm}^{-3}$, or 10^{11} cm^{-3} . Amazingly enough, this is only one order of magnitude greater than the ionization calculated for the biprop thruster! And the biprop simulation time was only 3.5 ms, whereas the Porcupine release lasted several seconds. The good correlation between the Porcupine test observations and our numerical simulation results is encouraging, and indicates that CIV effects can increase the local plasma density at space system surfaces several orders of magnitude.
- Finally, *Biasca et al.* [Ref. 12] have calculated plasma densities near the point of a thruster firing into the ram, and for certain CIV reaction rates, they report two orders of magnitude local plasma density increases. Hence the simulated CIV ionization levels presented here, fall between the Porcupine observed levels and the levels calculated by *Biasca et al.* [Ref. 12].

Arcing at High-Voltage Surfaces

The effluents released from various space system thrusters generate a self-induced neutral/plasma environment that may affect spacecraft operations. This study focuses on the interactions between the self-induced environment and biased high-voltage surfaces in space. As an example we look at the solar arrays that are baselined for the Space Station Freedom. The 75 kW of power needed for the SSF will be supplied at high voltage and low current to minimize resistive losses and the mass of cabling and harnesses. The arrays will be operating at 160 volt, which is significantly higher than currently used arrays with voltage drops ranging from 28 to 75 volt (Skylab).

For negatively biased solar cell interconnects, it is observed that below a critical voltage, arc discharges occur on the solar array. Arcing has been defined as a sudden current pulse much larger than the ambient current collection and typically lasting a few microseconds. Negative voltage thresholds ranging from 200 to several 100 volt have been observed in ground-based tests [Ref. 18, 20]. The flight data from the plasma interaction experiments PIX I and PIX II show arcing occurring at thresholds as low as -200 V [Ref. 19,20].

In a microscopic arcing mechanism developed by *Hastings et al.* [Ref. 20] and *Cho and Hastings* [Ref. 21], it is proposed that the arcing breakdown occurs by ionization of the neutral gas desorbed from the solar array dielectric cover glasses. The desorption is known as electron stimulated desorption (ESD) with the electrons being emitted from the solar array interconnectors. A schematic view of the solar array process leading to arcing is shown in figure 29. Note that only the dielectric cover glass and adhesive are shown; the solar cell is left out. This is because the solar cell is a semiconductor with a voltage drop across it of at most a volt or two, which is negligible in the charging process. The proposed arcing mechanism goes as follows:

- Ambient positive ions are attracted to the -160 V solar arrays, impinging on both the interconnects and the solar cell cover glass. The ions charge the dielectric cover glass positive with respect to the negative interconnects, leaving the cover glass sides relatively uncharged. A strong electric field is thus set up between the positive cover glass surface and the interconnector; the field is maximized at the triple junction point where the conducting interconnector, the dielectric, and the plasma all meet.
- The electric field at the interconnector surface can be enhanced by whiskers or dielectric impurities; the enhancement can be expressed by a so-called field enhancement factor, β .

Enhanced field emission electrons (EFEE) from the interconnector impinge on the cover glass and adhesive; this triggers multiple secondary electron emission from the side of the dielectric cover glass which leaves the sides positively charged. This increases the electric field across the triple junction point. The process can develop very rapidly because of the strong exponential dependence of the EFEE current on the electric field. When the electric field doubles, the emission current increases by ten orders of magnitude.

- Neutral gas is desorbed into the gap between the cells due to heating of the cover glass side by the electron current. The EFEE electron current flowing through the neutral cloud can eventually lead to a discharge like a surface flash over, and we have arcing.
- For high bias voltage (several 100 V negative) and a high field enhancement, the ion charging time governs the total charging time and hence the arc rate. The ion charging time, and therefore the arc rate, are functions of the local ion density. For low voltage and low electric field enhancement, the total charging time is governed by the EFEE charging time,
- The arcing voltage threshold depends on the local neutral density, but is independent of local ion density.

According to this theory [Ref. 20,21] a local neutral density of at least $2.2 \times 10^{23} \text{ m}^{-3}$ is required to trigger the Townsend breakdown and arcing at the array shown in figure 29, operating at -500 V with respect to the ambient. Hence, the thruster induced neutral densities of almost 10^{20} m^{-3} that we calculated along the array in figure 9 are unlikely to trigger arcing at the SSF -160 V arrays. There is the possibility that the thruster effluents could pile up in the gaps between the solar cells, and that this, combined with the neutral desorption off the side of the cover glass could lower the arcing thresholds to voltages less negative than -500 V. The bottom line, however, is that SSF solar array voltages probably are too low to induce arcing.

The electric power requirements for future space systems are likely to increase, and this will most likely be accomplished by increasing the operating voltage of the solar arrays. For increasing voltage bias, it becomes important to predict and control the local self-induced neutral environments, to minimize the risk of arcing.

If array voltages and/or neutral densities are such that arcing does indeed occur, it becomes important to predict the arc rates. According to the theory by *Cho and Hastings* [Ref. 21], the arc

rate is exponentially proportional to the negative voltage for voltages between approximately -500 and -300 V. Furthermore, at negative voltages below approximately -400 V, the arc rate is linearly proportional to the local plasma density. This is shown in figure 30 which presents the total arc rate versus the bias voltage (both on logarithmic scales) for two different neutral densities. An array containing 500 solar cells of dimension 2 by 2 cm (as flown on PIX II), operating at -500 V, and in an ambient plasma density of $5 \times 10^9 \text{ m}^{-3}$, will, according to the theory, experience an overall arc rate of about 0.1 s^{-1} . If the local plasma density is increased two orders of magnitude to $5 \times 10^{11} \text{ m}^{-3}$, the arc rate increases correspondingly to 10 s^{-1} . Thus, the four order of magnitude increase in local plasma density computed for the biprop hydrazine thruster firing into the ram, could presumably increase the arc rates overall arc rates on the array to an astonishing 10^4 s^{-1} ! Such dire possibilities clearly provide an incentive to analyze the self-induced environment thoroughly, in order to predict show stopping arc rates.

CONCLUSION

Various thruster activities around space systems in LEO generate self-induced neutral and plasma environments. The simulations presented in this report indicate that for certain geometries and thruster conditions, density increases along the solar arrays can be as high as six orders of magnitude for the neutrals, and four orders of magnitude for the local plasma. The tremendous ionization demonstrated by the critical ionization velocity mechanism stressed the importance of including this effect when total ionization of the neutral effluents is calculated. On the time rates of interest in our calculations, the main ionization mechanisms were ion-molecule reactions (including charge exchange) and CIV; photo-ionization was too slow to be important.

Despite the tremendous self-induced perturbations in the local plasma/neutral distribution, it was concluded that the density increases around the SSF are not likely to trigger arcing at the 160 V solar arrays. Arrays currently flown on other space systems are also too low in operating voltage to exhibit arcing. Future space systems, which most likely will be operating arrays at increased voltage biases, will, however, have to worry about arcing. Once the arcing voltage threshold has been reached, it becomes important to predict the arcing rates. These rates are a strong function of the local plasma density, which again is a function of the neutral density. In order to predict arcing thresholds and rates, it is therefore imperative that the local neutral and plasma environment around the space system is well characterized.

BIBLIOGRAPHY

1. Bird, G. A., Molecular Gas Dynamics, Clarendon Press, Oxford, 1976.
2. Bauer, S. J., Physics of Planetary Ionospheres, Physics and Chemistry in Space Volume 6, Edited by J. G. Roederer, Springer-Verlag, 1973.
3. Bernhardt, P. A., P. Rodriguez, C. L. Siefring, and C. S. Lin, "Field-Aligned Dynamics of Chemically Induced Perturbations to the Ionosphere", *J. Geophys. Res.*, 96, 13,887, 1991.
4. Bernhardt, P. A., "A Critical Comparison of Ionospheric Depletion Chemicals", *J. Geophys. Res.*, 92, 4617, 1987.
5. Francis, G., Ionization Phenomena in Gases, Butterworths Scientific Publications, London, 1960.
6. Rees, M. H., Physics and chemistry of the upper atmosphere, Cambridge atmospheric and space science series, Cambridge University Press, 1989.
7. Alfvén, H., On the Origin of the Solar System, Oxford at the Clarendon Press, London, 1954.
8. Newell, P. T., "Review of the Critical Ionization Velocity Effect in Space", *Rev. Geophysics*, 23, 93, 1985.
9. McNeil, W. J., S. T. Lai, and E. Murad, "Interplay Between Collective and Collisional Processes in Critical Velocity Ionization", *J. Geophys. Res.*, 95, 10,345, 1990.
10. Haerendel, G., "Alfvén's critical velocity effect tested in space", *Z. Naturforsch. A*, 37, 728, 1982.
11. Lai, S. T. and E. Murad, "Critical Ionization Velocity Experiments in Space", *Planet. Space. Sci.*, 37, 865, 1989.
12. Biasca, R. J., D. E. Hastings, and D. L. Cooke, "The Possibility of Critical Velocity Ionization Near the Space Station: Simulation Results", AIAA Paper 92-0847, Presented at the 30th Aerospace Sciences Meeting, Reno, NV, January, 1992.
13. Biasca, R. J., Personal Communication.
14. Caledonia, G. E., J. C. Person, and D. E. Hastings, "The Interpretation of Space Shuttle Measurements of Ionic Species", *J. Geophys. Res.*, 92, 273, 1987.
15. Ikezoe, Y., S. Matsuoka, M. Takebe, and A. Viggiano, Gas-Phase Ion-Molecule Reaction Rate Constants Through 1986, Maruzen Company, Ltd., Tokyo, Japan, 1987.
16. Torr, M. R., D. G. Torr, R. A. Ong, and H. E. Hinteregger, "Ionization Frequencies for Major Thermospheric Constituents as a Function of Solar Cycle 21", *Geophys. Res. Letters*, 6, 771, 1979.
17. Hudson, R. D., "Critical Review of Ultraviolet Photoabsorption Cross Sections for Molecules of Astrophysical and Aeronomic Interest", *Rev. Geophys. and Space Phys.*, 9, 305, 1971.

18. Hastings, D. E., G. Weyl, and D. Kaufman, "Threshold Voltage for Arcing on Negatively Biased Solar Arrays", *J. Spacecraft*, 27, 539, 1990.
19. Grier, N. T., and N. J. Stevens, Plasma Interaction Experiment (PIX) Flight Results, Spacecraft Charging Technology - 1978, NASA Conference Publication 2071, AFGL-TR-79-0082, p.295.
20. Hastings, D. E., M. Cho, and H. Kuninaka, "The Arcing Rate for a High Voltage Solar Array: Theory, Experiment and Predictions", AIAA Paper 92-0576, Presented at the 30th Aerospace Sciences meeting, Reno, NV, January 1992.
21. Cho, M. and D. E. Hastings, "Dielectric Charging Processes and Arcing Rates at High Voltage Solar Array, AIAA Paper 91-0605, Presented at the 29th Aerospace Sciences Meeting, Reno, NV, January 1991.
22. Bird, G. A., "Breakdown of Translational and Rotational Equilibrium in Gaseous Expansions", *AIAA Journal*, 8, 1998, 1970.
23. Hueser, J. E., L. T. Melfi, G. A. Bird, and F. J. Brock, "Rocket Nozzle Lip by Direct Simulation Monte Carlo Method", *J. Spacecraft*, 23, 363, 1986.

APPENDIX A

The Direct Simulation Monte Carlo Method

The direct simulation Monte Carlo (DSMC) method is a technique for computer modeling of a real gas by several thousand simulated molecules. It is not feasible to track each individual molecule; instead the method tracks simulated molecules where each simulated molecule represents as many as 10^{19} real molecules. The velocities, temperatures, and positions of these molecules are stored in the computer and are modified with time as the molecules are tracked through representative collisions and boundary conditions in physical space. The computational task associated with the direct simulation becomes feasible when the gas density is sufficiently low. The degree of rarefaction of a gas flow has traditionally been expressed by the Knudsen number which is the ratio of the mean free path to a typical dimension of the flow field. A Knudsen number of 0.1 has been quoted as the boundary between continuum and transition flow regimes. An improvement in determining the breakdown of continuum flow is based on the work by G. A. Bird [Ref. 22], using the Bird parameter, P ,

$$P = U / \rho f |dp/ds|,$$

where U is the flow velocity, ρ is the density, f is the collision frequency, and s is the streamline distance. We start the DSMC calculations at a Bird parameter contour of 0.04. Required inputs to the simulation are flow composition, number densities, temperature, and vector velocity. The rarefied flow is divided into grid cells dimensioned to the local mean free path, and the molecules are marched through the grid over time steps corresponding to the inter-collision time. The code is made computationally efficient by uncoupling the molecular motion and collisions. Within each cell all molecules are considered identical. At each time step a collision pair is picked at random and the collision probability is calculated based on cross section and relative velocities. This probability is compared to a random number between 0 and 1 (hence the name Monte Carlo), and if the probability is greater than the number, a collision occurs, if not, a new collision pair is picked. Enough collisions are calculated to match the local collision frequency. After the collision(s), the molecules are advanced according to their relative velocities, and new position coordinates are computed.

Ideally, the cell dimensions should be on the order of the local mean free path or less, and the time step should be on the order of the time between collisions or less. To reduce the number of cells in the computational grid, however, this requirement is sometimes relaxed in the high density flow region close to the continuum-to-rarefied transition line. Due to the robustness of the DSMC code, this assumption has no noticeable impact on the flow simulation [Ref. 23].

To define the DSMC computational grid, the transition line at $P=0.04$ is divided into increments of length 1 to 5 local mean free paths. The increments define the grid cell dimensions at the DSMC start line. As the grid extends into the rarefied flow region, the cell dimensions are scaled roughly with the local mean free path. Thus for the 25 lbf monopropellant hydrazine thruster flowfield, which was simulated in a roughly 5 by 10 m region, we defined a grid of approximately 5,000 cells. The time step with which the molecules are marched is roughly equal to the inter-collision time at the start line (10^{-6} seconds). The time step is kept constant throughout the flowfield so several steps are required to traverse one cell in the rarefied end of the flowfield.

A DSMC simulation does not converge to a final solution, however, the simulation becomes steady and statistically more correct with time. Simulations are generally run until each grid cell has been sampled about 10^4 to 10^5 times. In the back flow of a thruster plume this condition has to be relaxed at least an order of magnitude. For the 25 lbf mono/biprop hydrazine thrusters modeled here, the DSMC simulations ran on the order 20 to 60 CPU hours on our Stardent mini super-computer, which corresponds to several 100 milliseconds of real flow time. The DSMC provides composition, density, vector velocities, temperatures (translational, rotational, vibrational), and Mach numbers in every cell of the flowfield. By manipulating this data, properties such as pressure and mass flux can be calculated anywhere in the flowfield. At surfaces within the flow, the code provides pressure, shear stress, and incident and reflected energy fluxes.

APPENDIX B

Flux Solver for Computing Local Plasma Density

A computer code for solving a set of ion and neutral continuity equations was developed and is enclosed. The code, which is written in FORTRAN, is suitable for execution on a PC. An initial neutral distribution data file (ARRAY.DAT) containing 2-D flowfield grid data (number density, vector velocities) is required. Currently, the *total* neutral density and the average velocities are read. The fraction of various species in the initial neutral distribution are set in the code. By slightly changing the read statement, one could make the code read the densities and velocities *per species*, since this information is available from the DSMC simulation anyway. The grid dimensions are currently 10 by 10 cm, but this can obviously be changed. The code has provisions for both monopropellant and bipropellant hydrazine thruster plumes, and prompts the user for ambient plasma density, plume direction (firing into ram/wake), photon flux direction, time step, and simulation time.

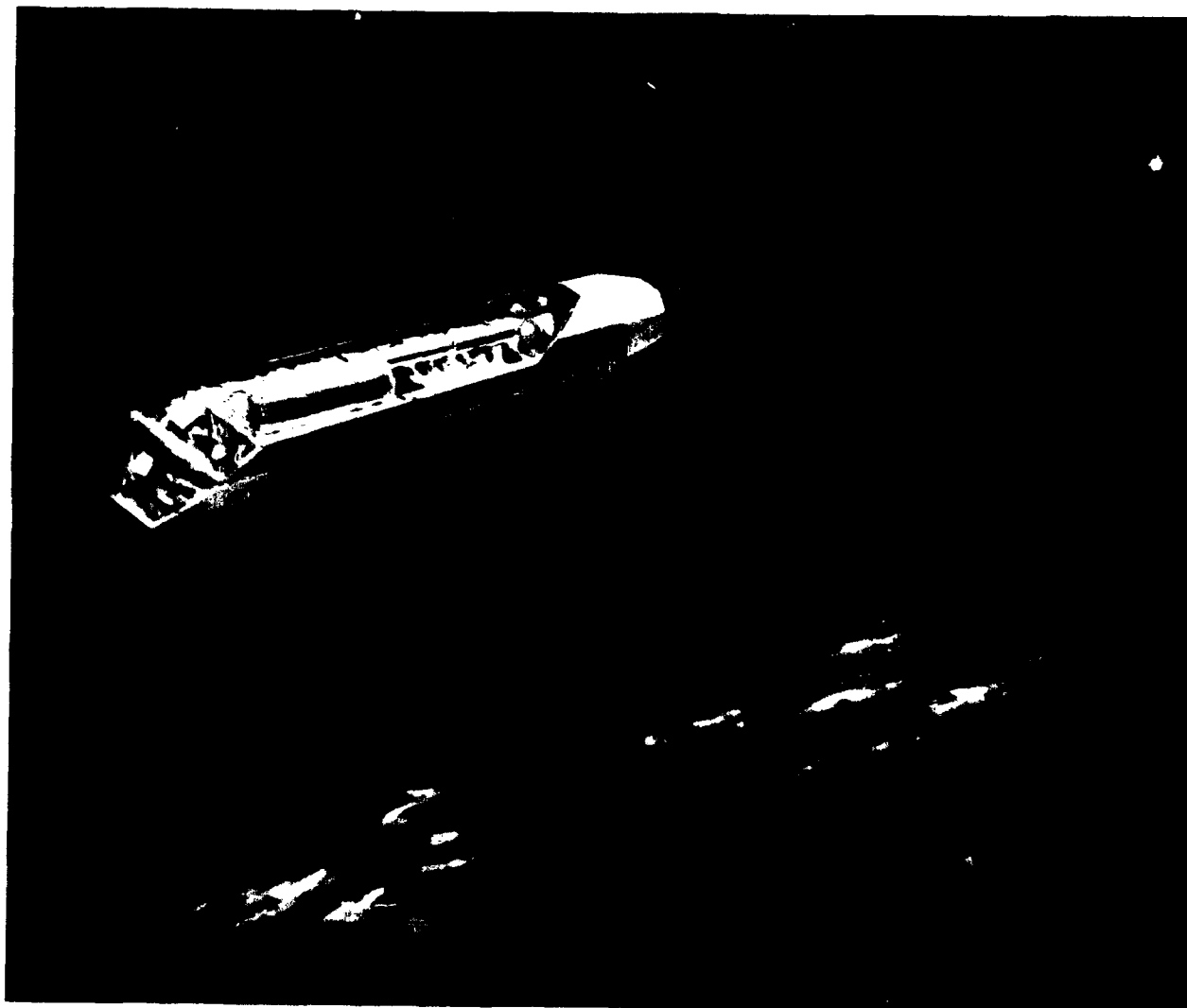
The code includes ion-molecule ionization (including charge exchange), CIV ionization, and photo-ionization. The photo-ionization is computed based on photon flux and wavelength, and the ionization rate as a function of wavelength. Due to the photon speed, the Courant stability condition requires that the numerical time step be less than 10^{-10} s. At such a short time step, extensive run times are required to simulate flow times of millisecond duration. In afterthought, it would have been better to include the photo-ionization as an option only. The maximum time step without photo-ionization is on the order 10^{-6} s. Furthermore, instead of computing the photo-ionization yield as a function of wavelength, it could be included as a *total* photo-ionization production rate, e. g., 10^{-5} ions produced per neutral molecule per second. This is typically done in other codes.

The CIV ionization rates are derived from plots provided by *Biasca* [Ref. 13]. For the two species where CIV ionization rates are available (CO_2 and N_2), two rates which are valid for two different neutral density regimes have been derived. The rates are of course valid only for plasma/neutral relative velocities in excess of the critical ionization velocity. The maximum electron current available to charge neutralize the exponentially produced CIV ions is limited by the available ambient electron flux.

The code starts with the initial neutral distribution, and then turns on the various ionization mechanisms. After each time step, the column of grid cells closest to the thruster get their neutral distribution reset, to simulate the continuous influx of neutrals from the thruster. Similarly, the grid column on the side of the computational region facing the ambient O^+ flux are replenished

with ambient density O^+ after each time step. The plasma is made charge neutral after each time step by setting the electron density equal to the total ion density (see also CIV limitations in preceding paragraph).

The final ion densities are written in a grid format to an output file, ION.DAT.



DAC110352

Neutral Particle Beam (NPB) Platform - Artist's Concept
Note Thruster Plume Exhaust at Rear of Vehicle

FIGURE 1

Generic Cylindrical Spacecraft with Nozzle
Chemical Power System Ejecting H₂

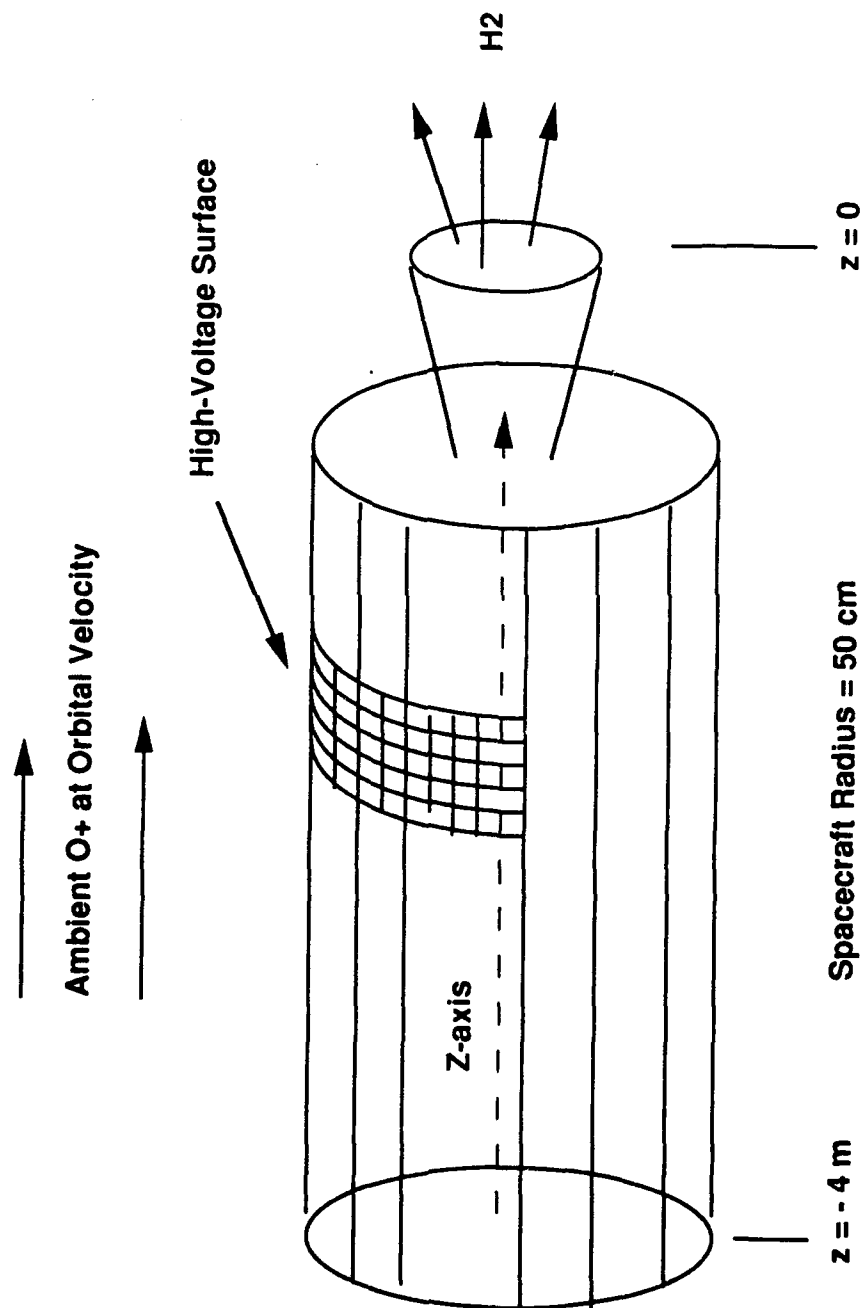


FIGURE 2

Log H₂ Number Density [m⁻³] Chemical Power System H₂ Release

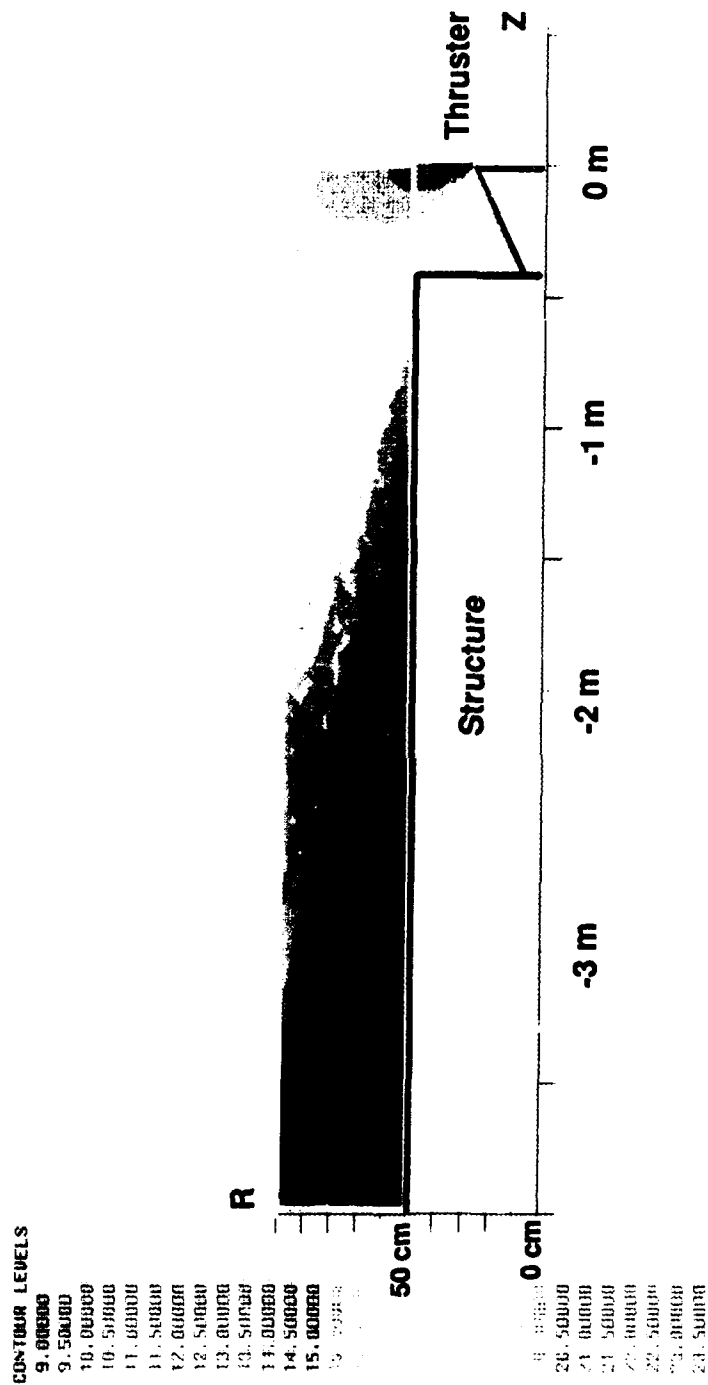


FIGURE 3

Log H₂ Static Pressure [torr] Chemical Power System H₂ Release

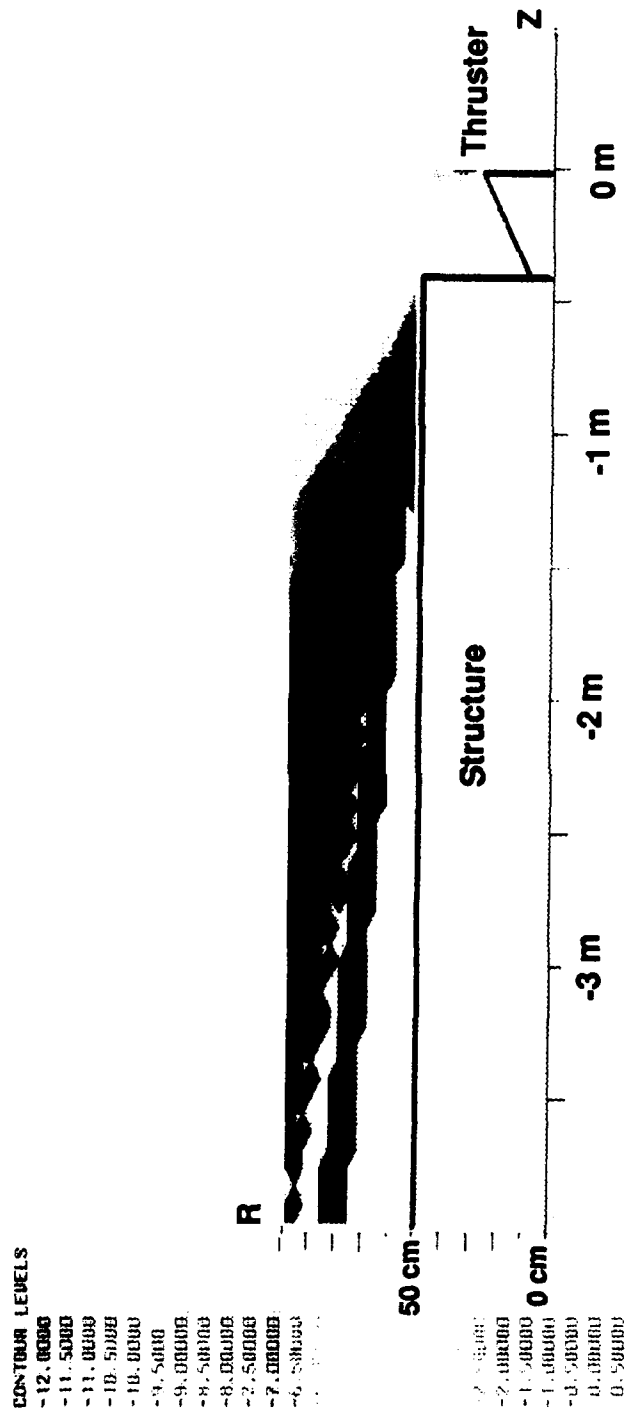
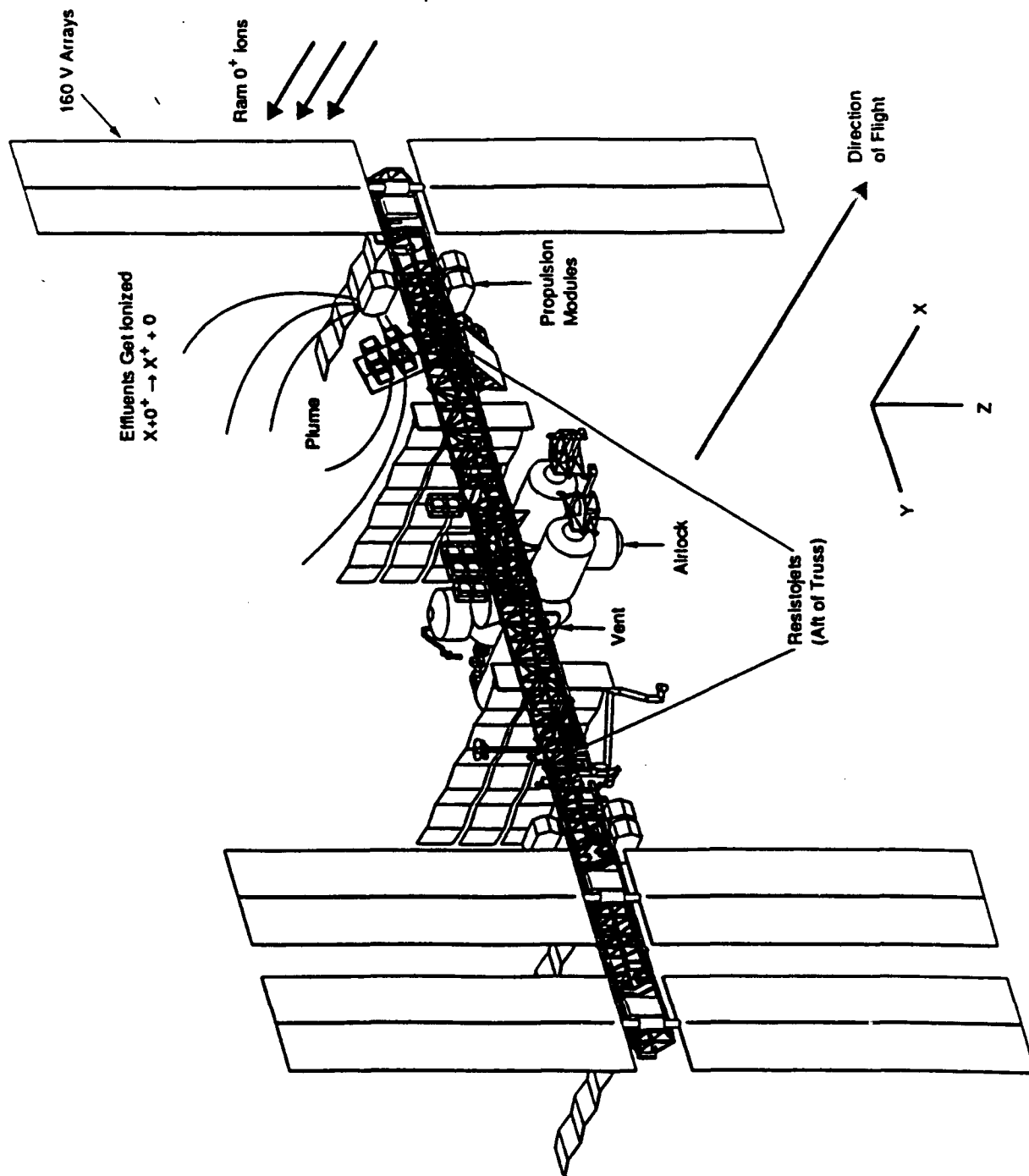


FIGURE 4



SSF Final Configuration

FIGURE 5

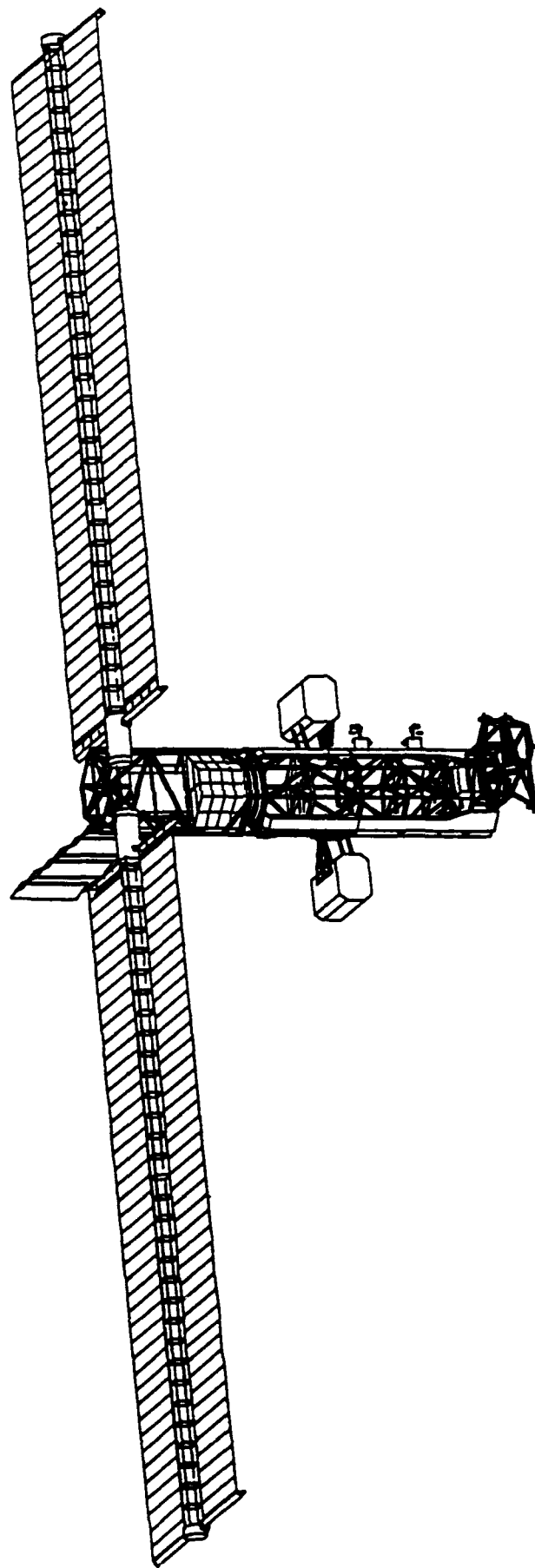
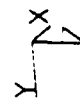


FIGURE 6

SSF Assembly

STAGE-2



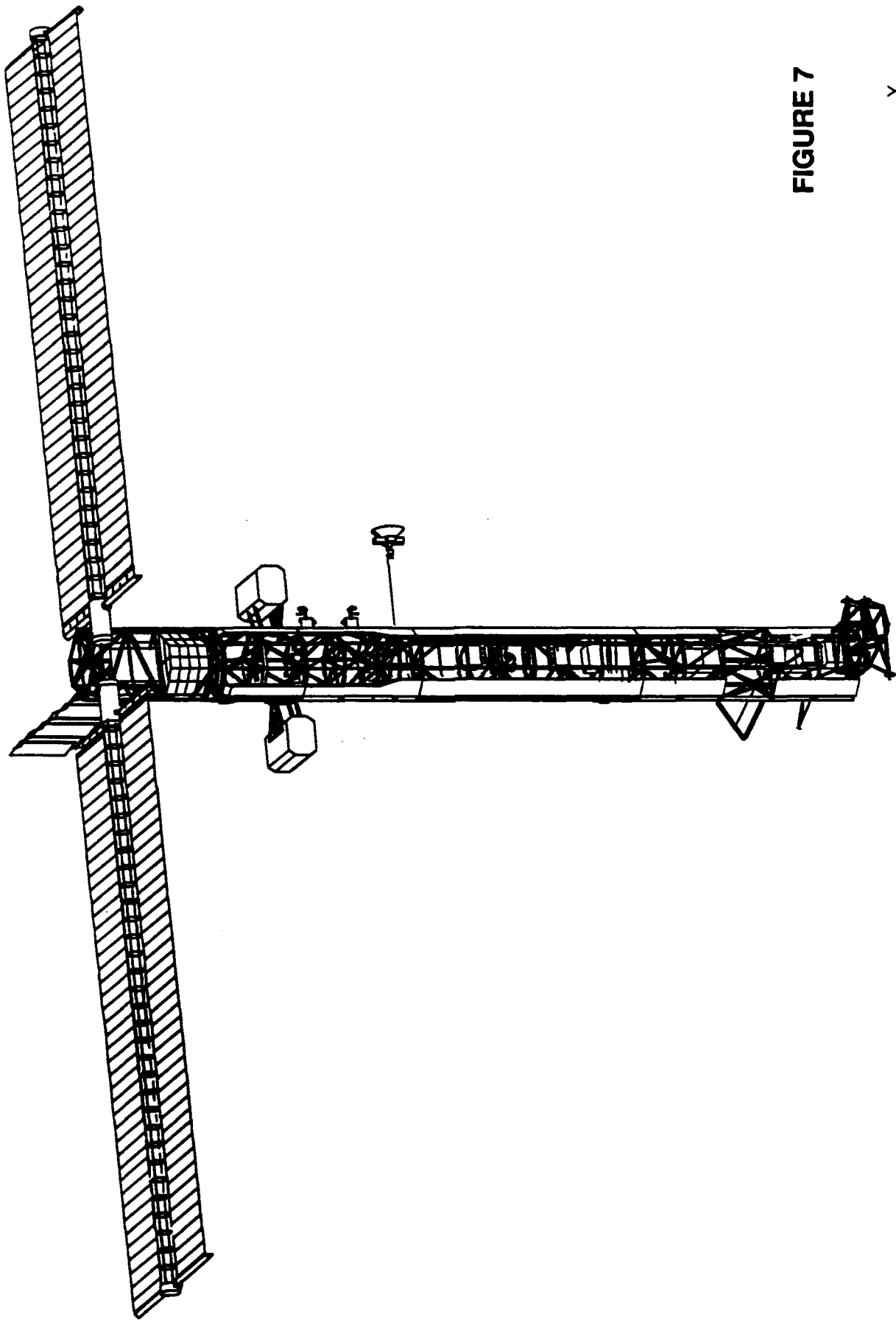
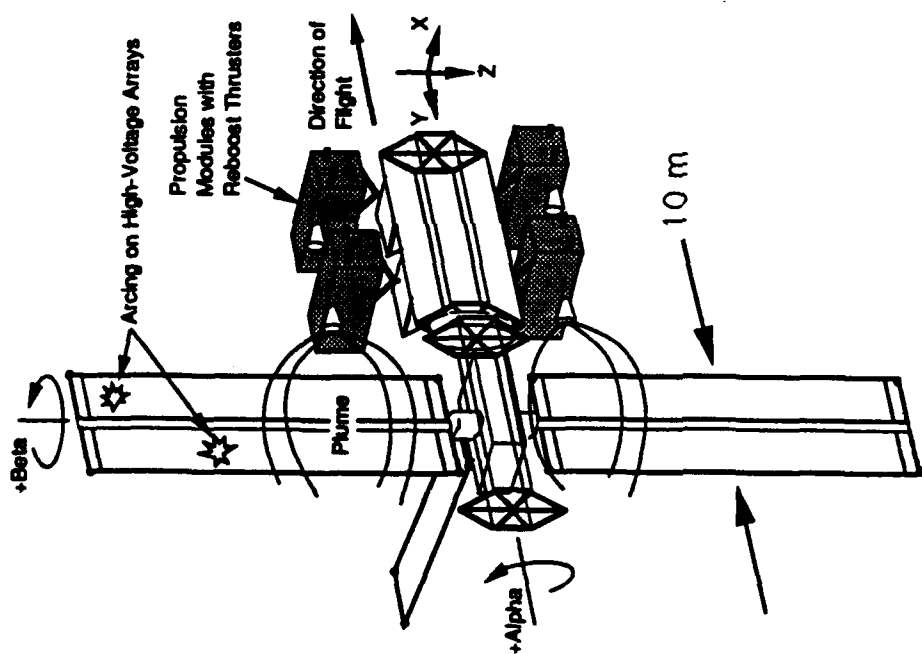


FIGURE 7

SSF Assembly STAGE-4



SSF Arrow Mode Configuration

FIGURE 8

**Log10 Constant Number Density Contours of Neutral Effluents
100% Thermal Accommodation at Solar Array
25 lbf Monopropellant Hydrazine Thruster
Thrusting past solar array into wake**

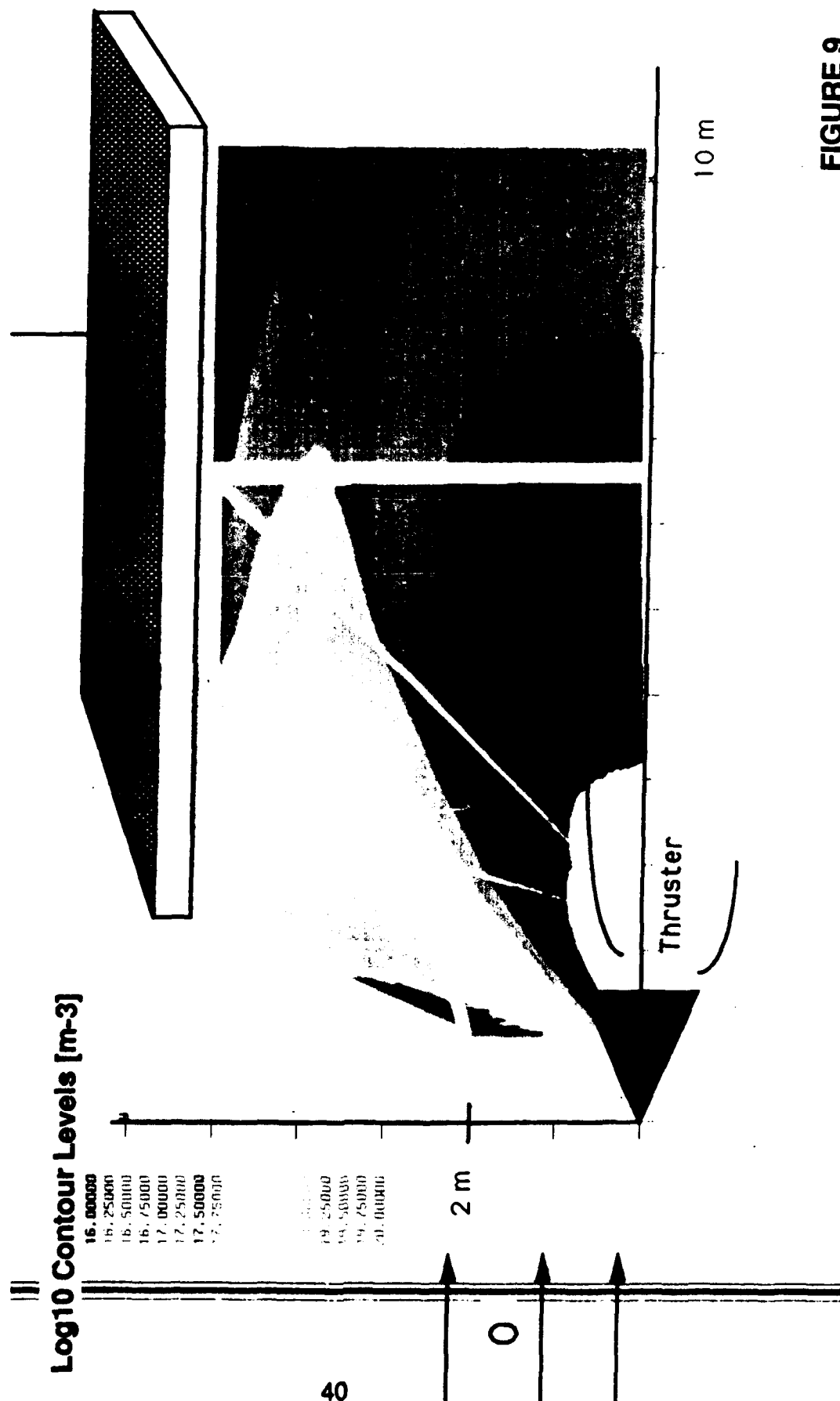


FIGURE 9

Log10 Constant Static Pressure Contours of Neutral Effluents

25 lbf Monopropellant Hydrazine Thruster

Thrusting past solar array into wake

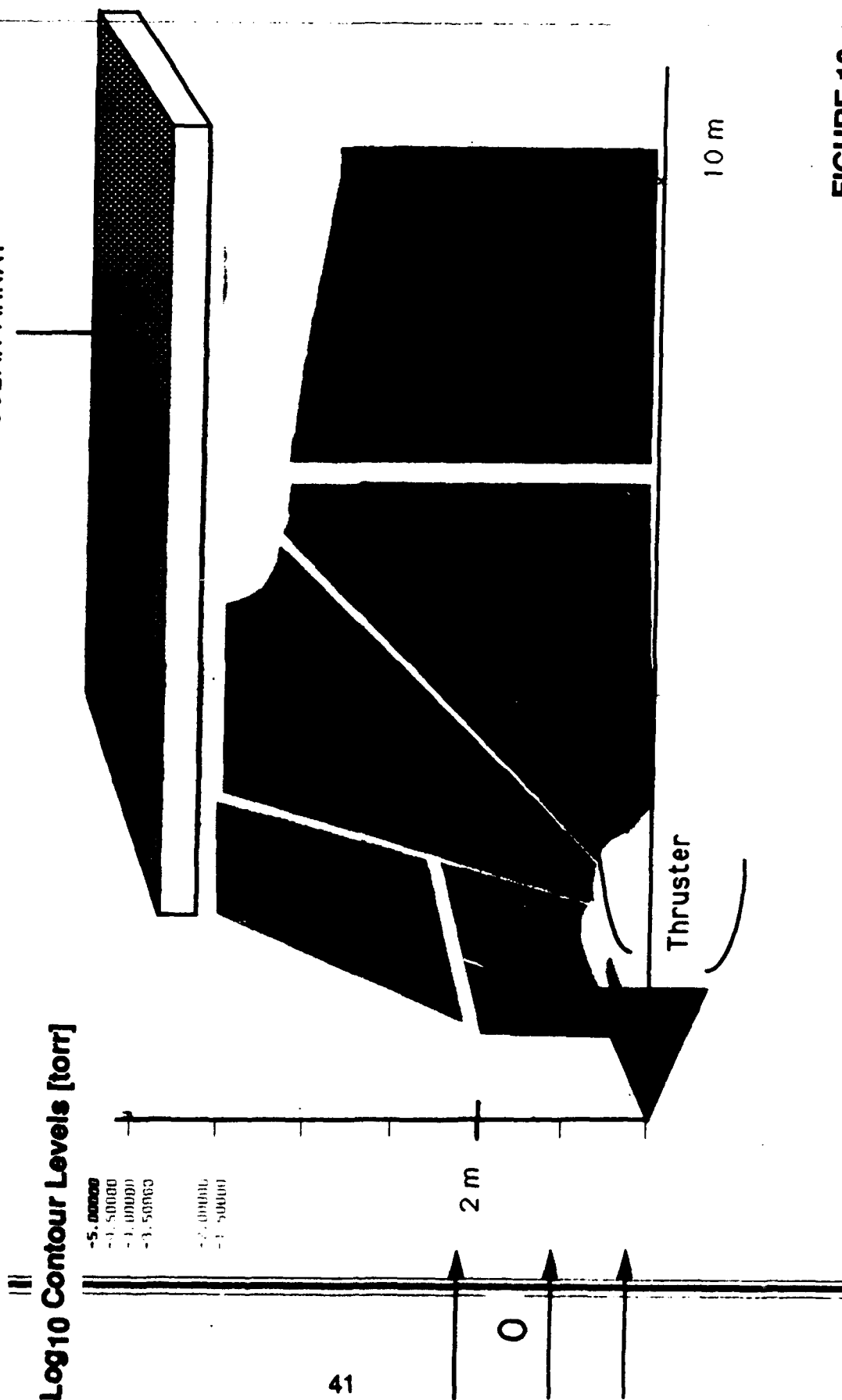


FIGURE 10

Constant Temperature Contours of Neutral Effluents [K]

100% Thermal Accommodation

25 lbf Monopropellant Hydrazine Thruster

Thrusting past solar array into wake

HIGH VOLTAGE
SOLAR ARRAY

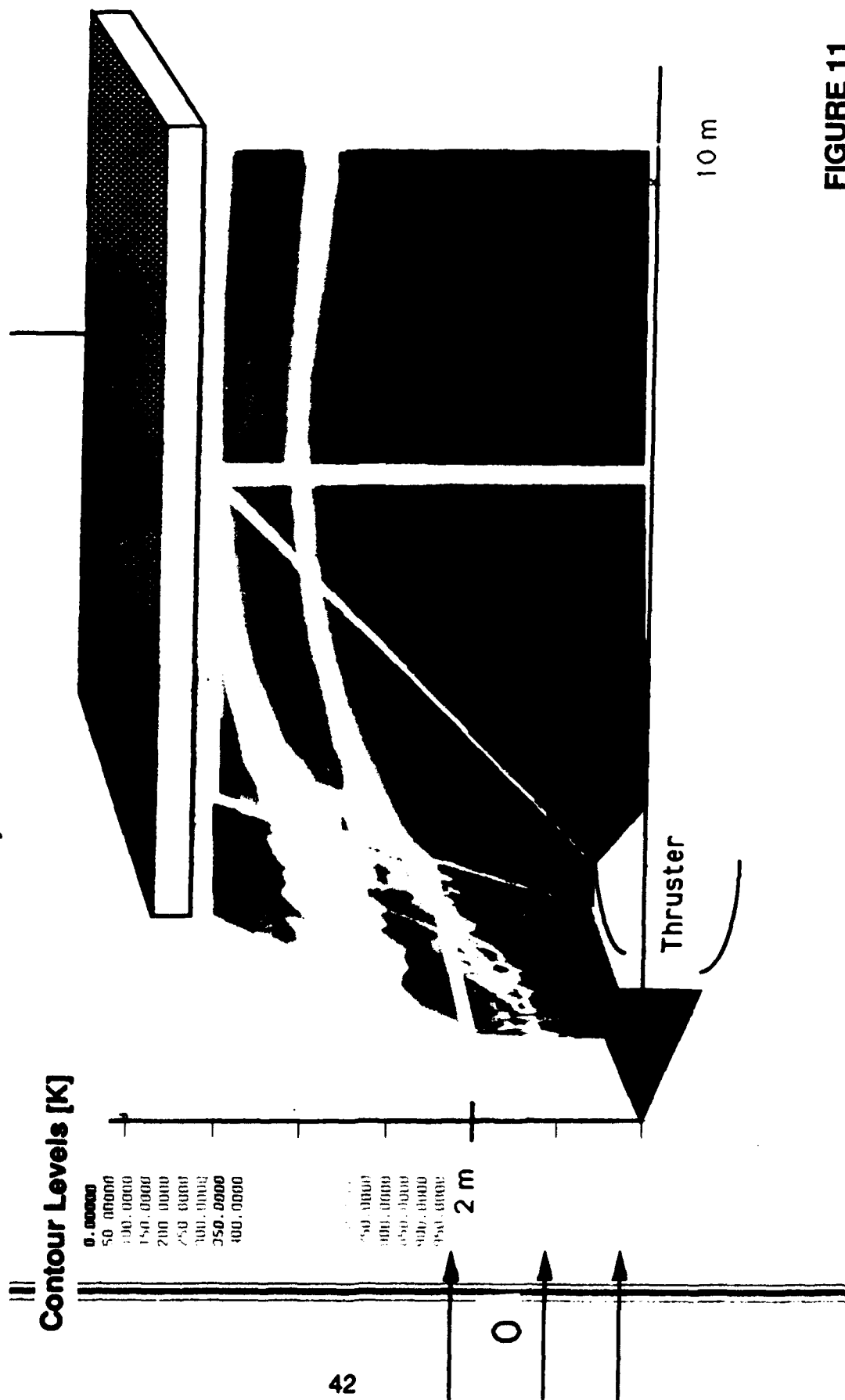


FIGURE 11

**Constant Absolute Velocity Contours of Neutral Effluents [m/s]
25 lbf Monopropellant Hydrazine Thruster
Thrusting past solar array into wake**

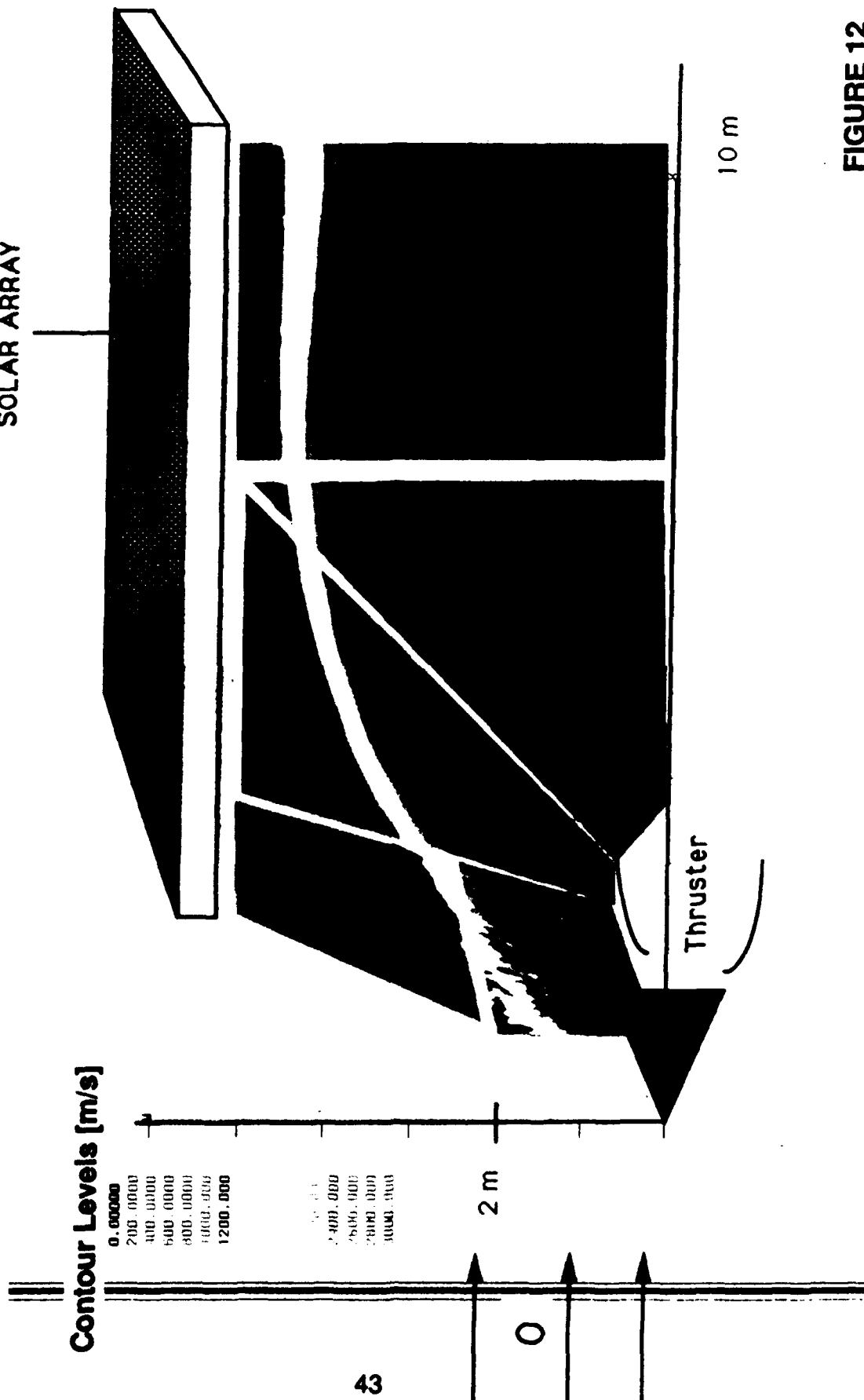


FIGURE 12

Constant Axial Velocity Contours of Neutral Effluents [m/s]

Solar Array Temperature = 333 K

25 lbf Monopropellant Hydrazine Thruster

Thrusting past solar array into wake

HIGH VOLTAGE
SOLAR ARRAY

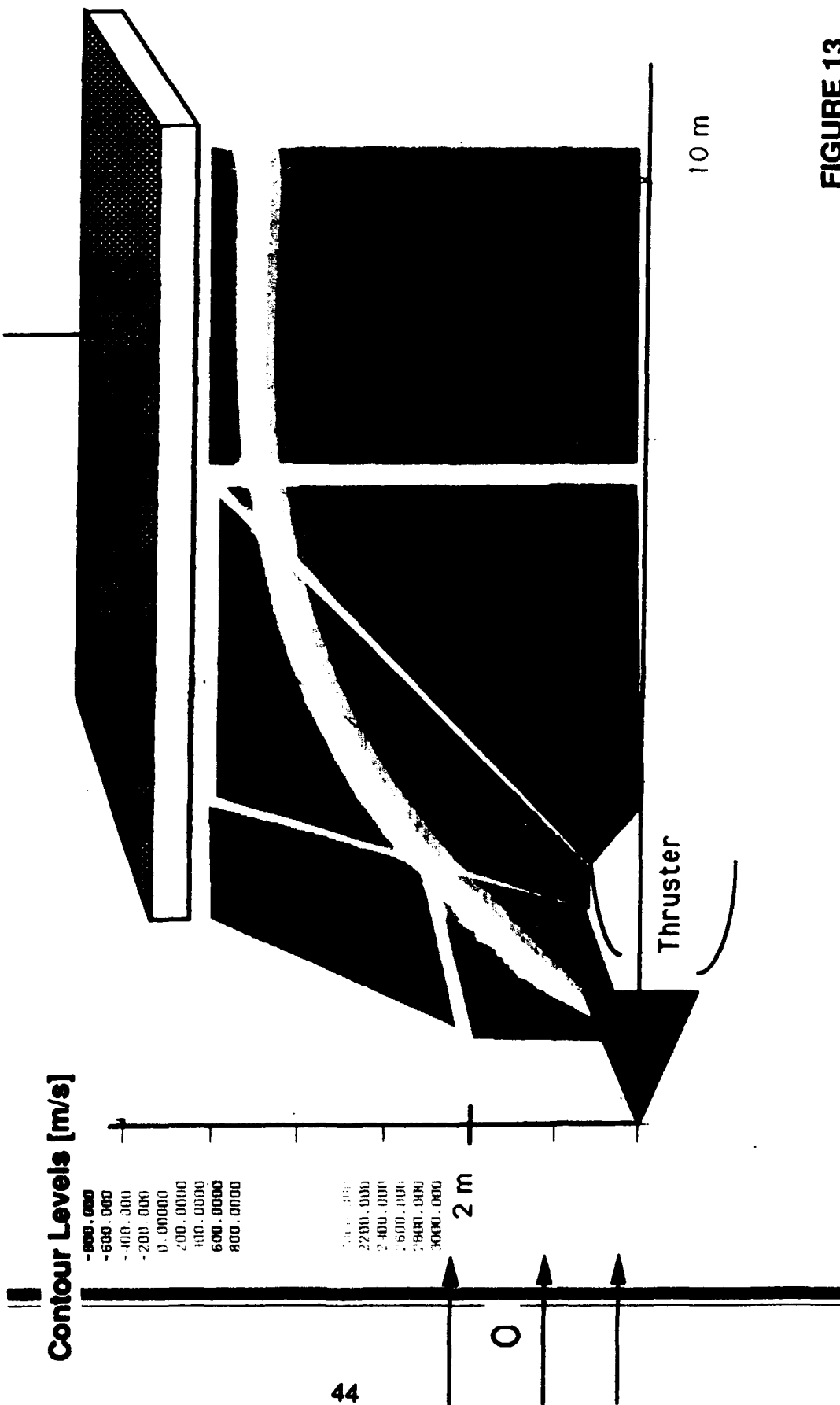


FIGURE 13

Constant Radial Velocity Contours of Neutral Effluents [m/s]

Solar Array Temperature = 333 K

25 lbf Monopropellant Hydrazine Thruster

Thrusting past solar array into wake

HIGH VOLTAGE
SOLAR ARRAY

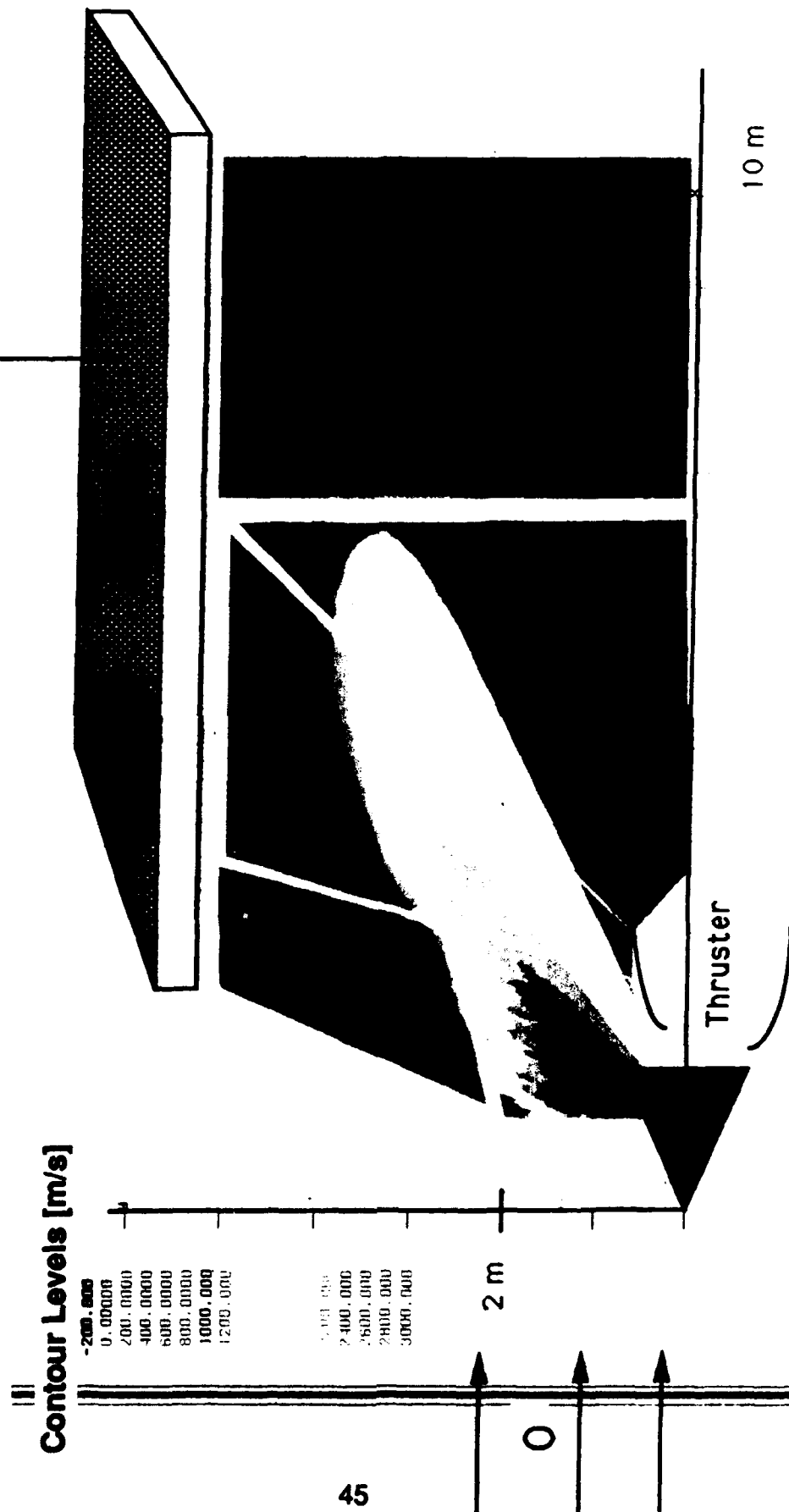


FIGURE 14

Log10 Constant Number Density Contours of Neutral Effluents 25 lbf Bipropellant Hydrazine Thruster Thrusting past solar array into wake

HIGH VOLTAGE
SOLAR ARRAY

Log10 Contour Levels [m-3]

16.00000
16.25000
16.50000
16.75000
17.00000
17.25000
17.50000
17.75000

18.00000
18.25000
18.50000
18.75000
19.00000
19.25000
19.50000
19.75000
20.00000

2 m

0

Thruster

10 m

FIGURE 15

**Log10 Constant Number Density Contours of Neutral Effluents
2.5 lbf Monopropellant Hydrazine Thruster
Thrusting past solar array into wake**

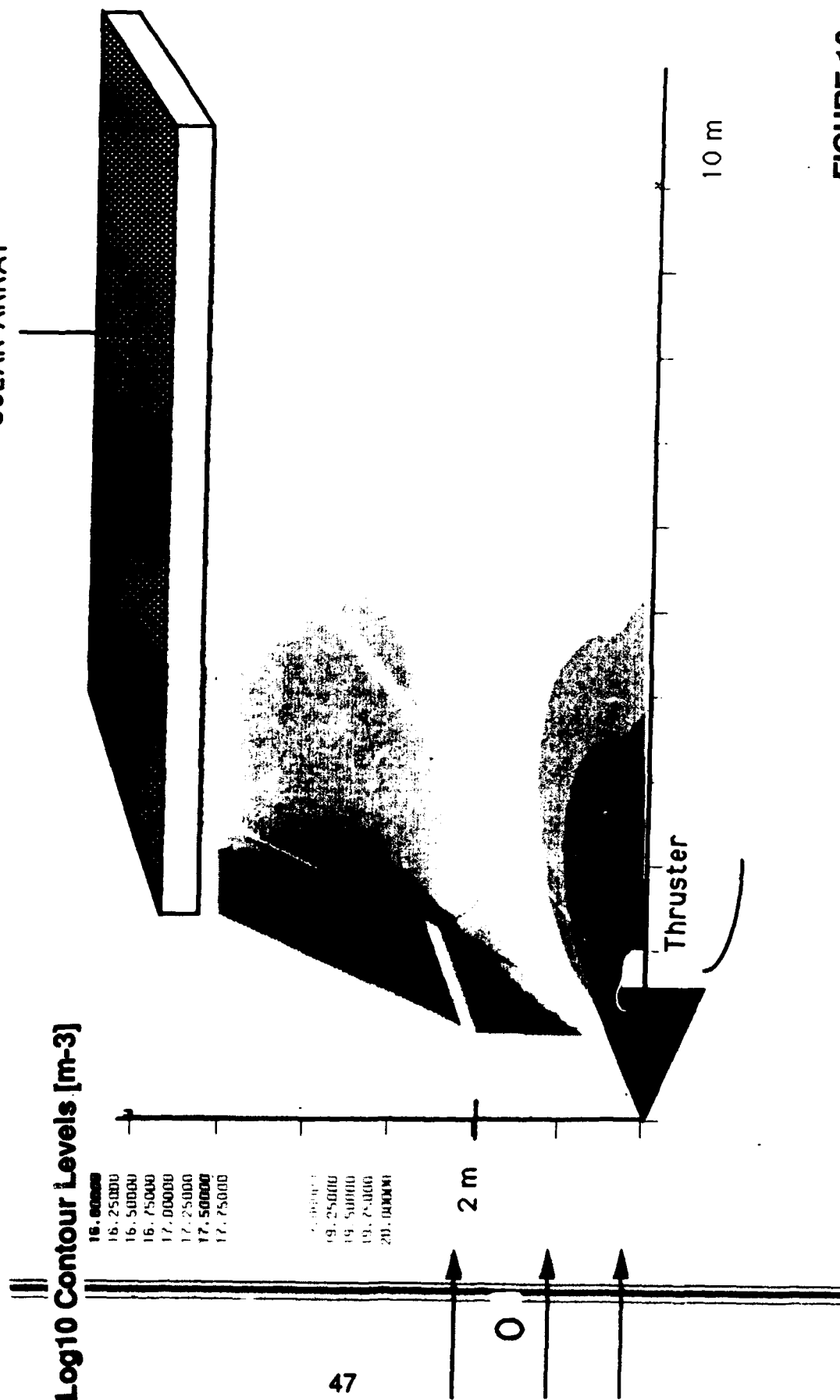


FIGURE 16

Constant Temperature Contours of Neutral Effluents [K] 2.5 lbf Monopropellant Hydrazine Thruster Thrusting past solar array into wake

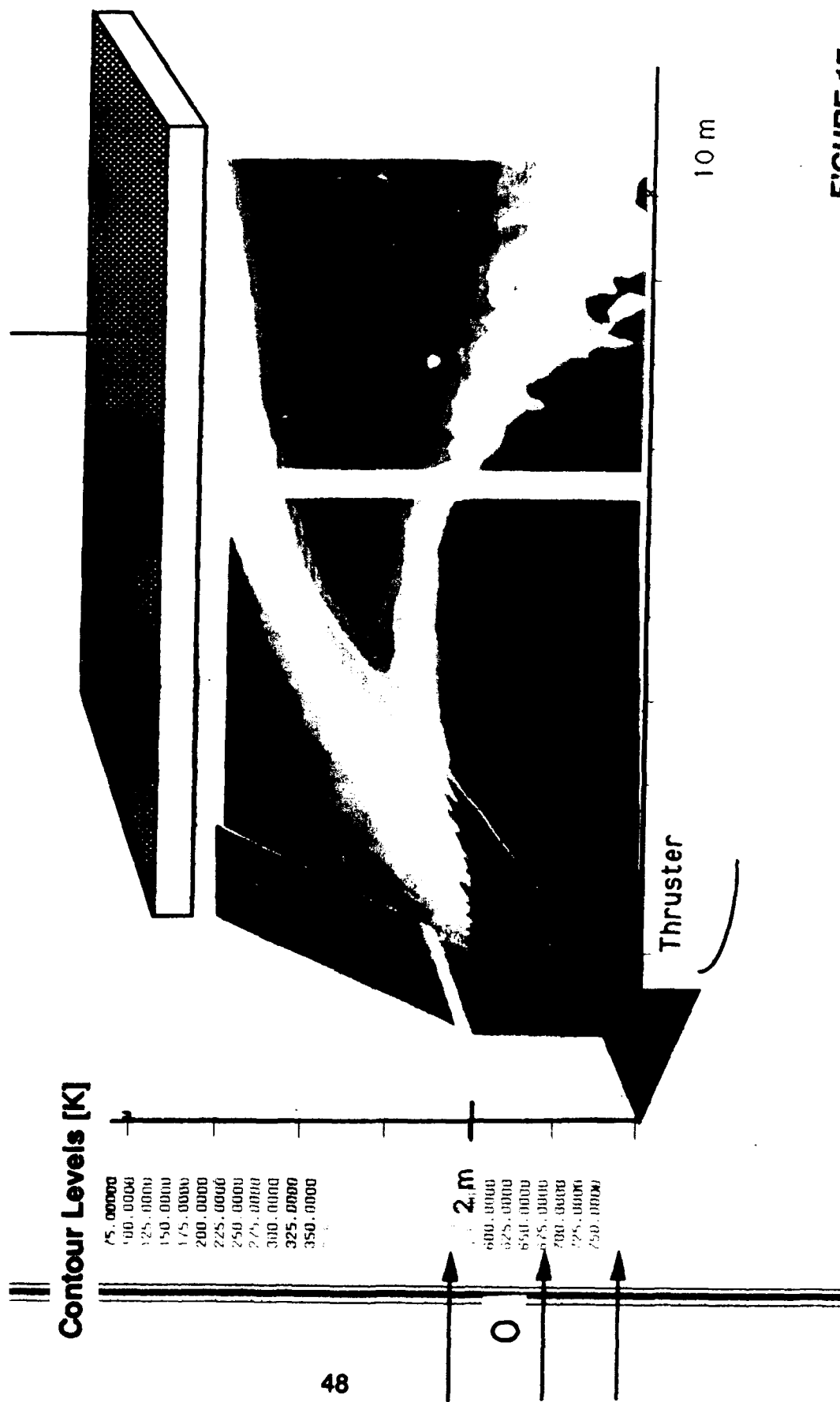


FIGURE 17

Log10 Constant Number Density Contours of Neutral Effluents 2.5 lbf Bipropellant Hydrazine Thruster Thrusting past solar array into wake

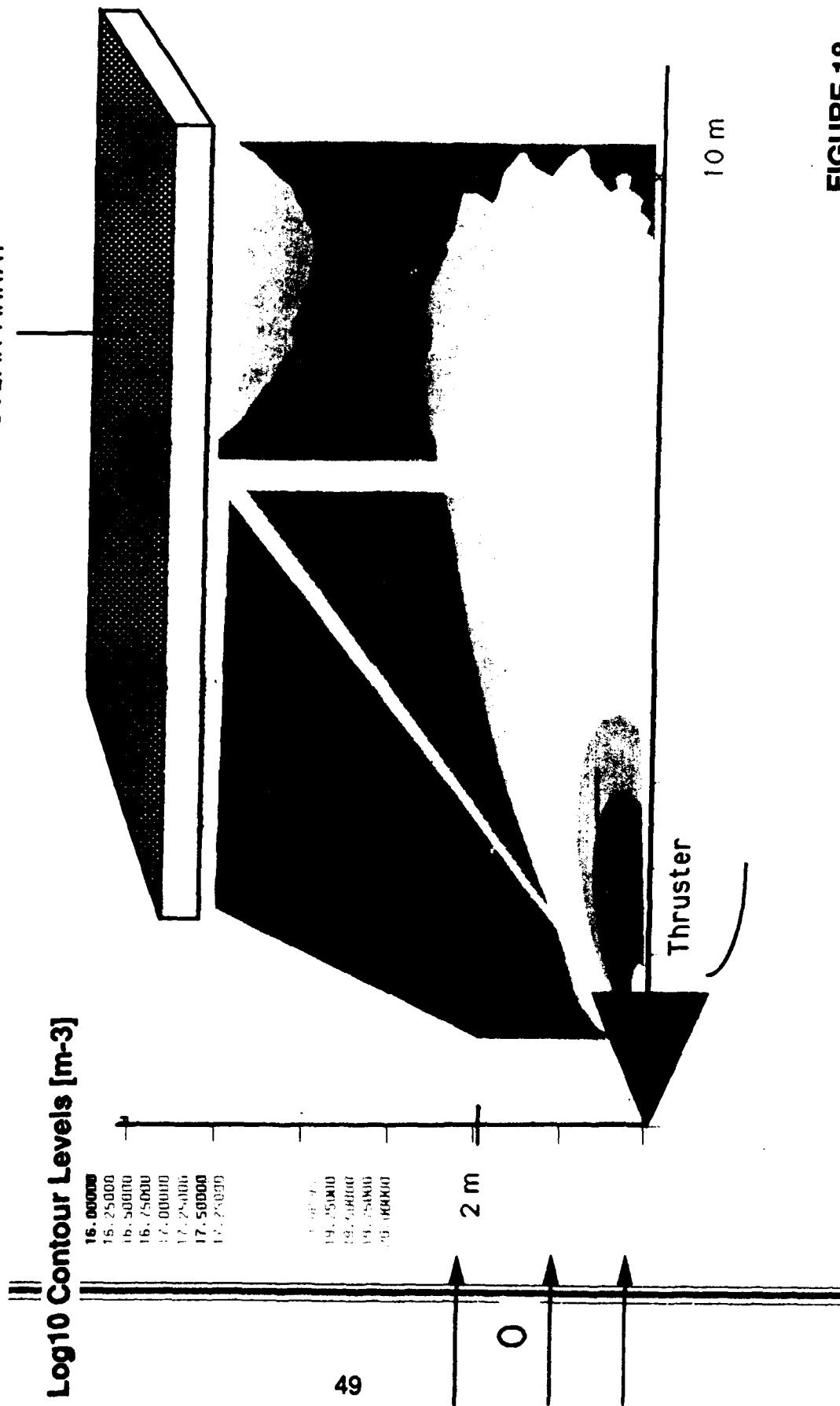


FIGURE 18

Constant Temperature Contours of Neutral Effluents [K]
0% Thermal Accommodation (Specular Reflection)
25 lbf Monopropellant Hydrazine Thruster
Thrusting past solar array into wake

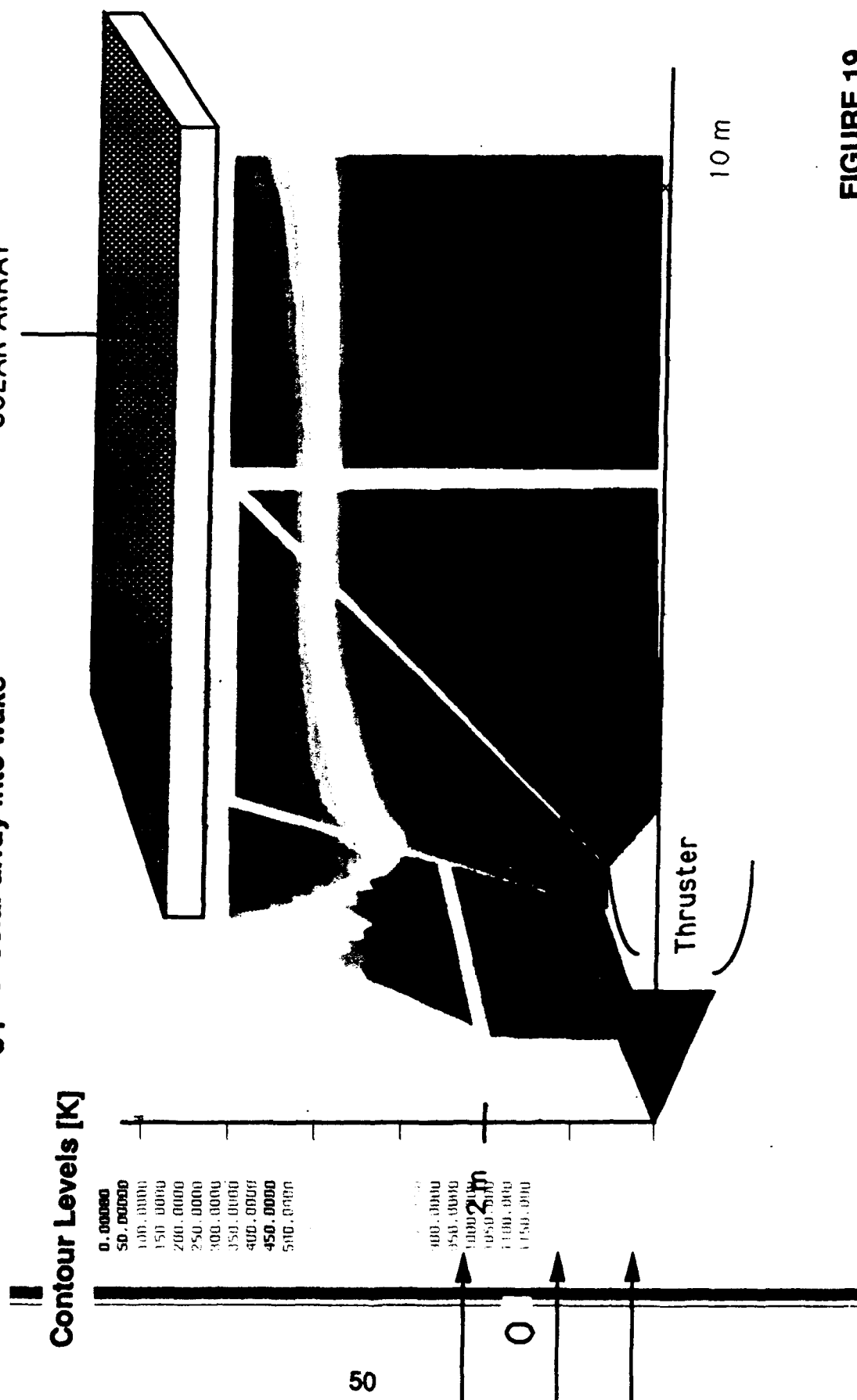


FIGURE 19

**Log10 Constant Number Density Contours of Neutral Effluents
25 lbf Monopropellant Hydrazine Thruster
0% Thermal Accommodation (Specular Reflection)
Thrusting past solar array into wake**

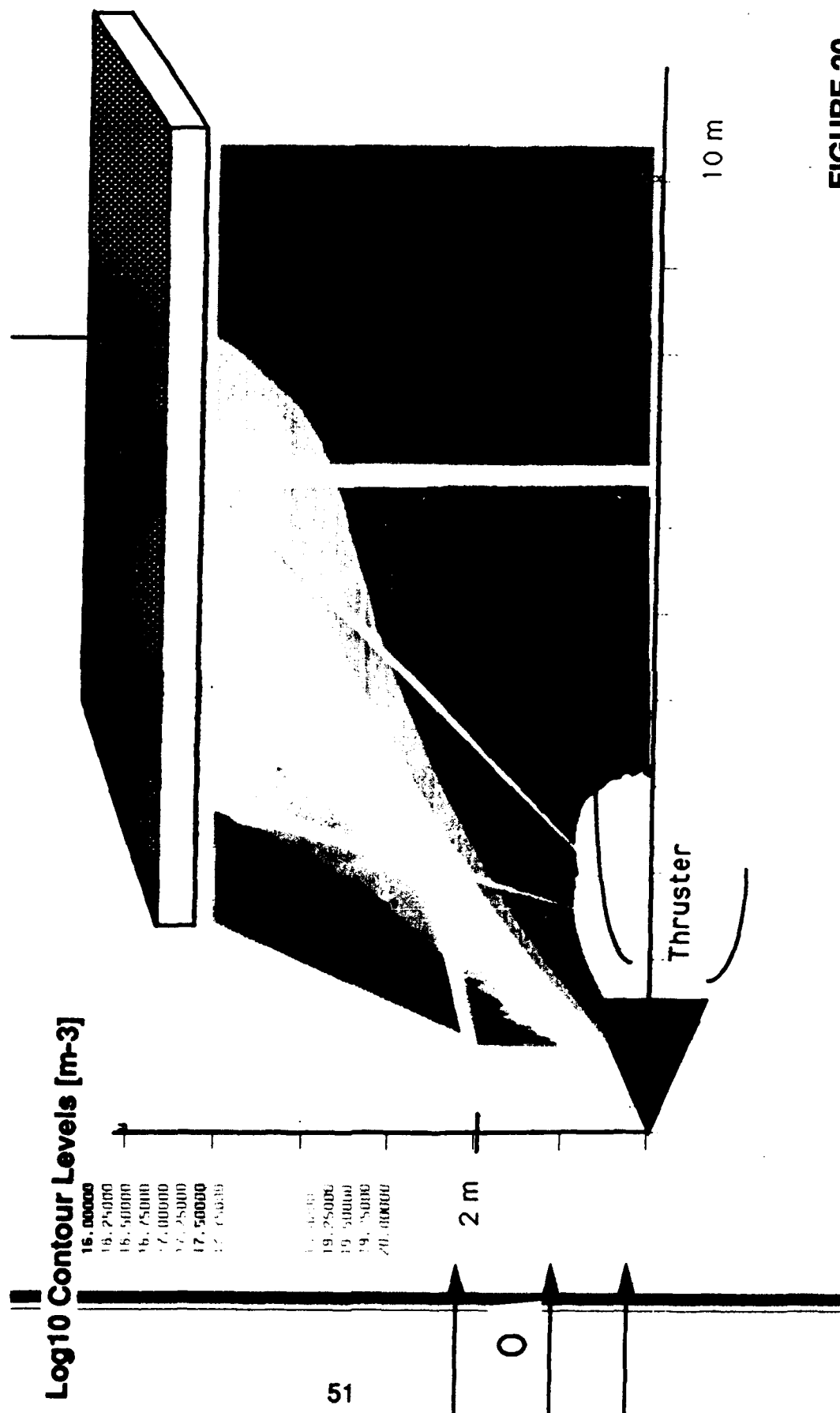


FIGURE 20

Log10 Constant Number Density Contours of Electrons [cm⁻³]
(Electron Density = Total Ion Density)
25 lbf Monopropellant Hydrazine Thruster
Thrusting past solar array into wake
3.5 millisecond simulation

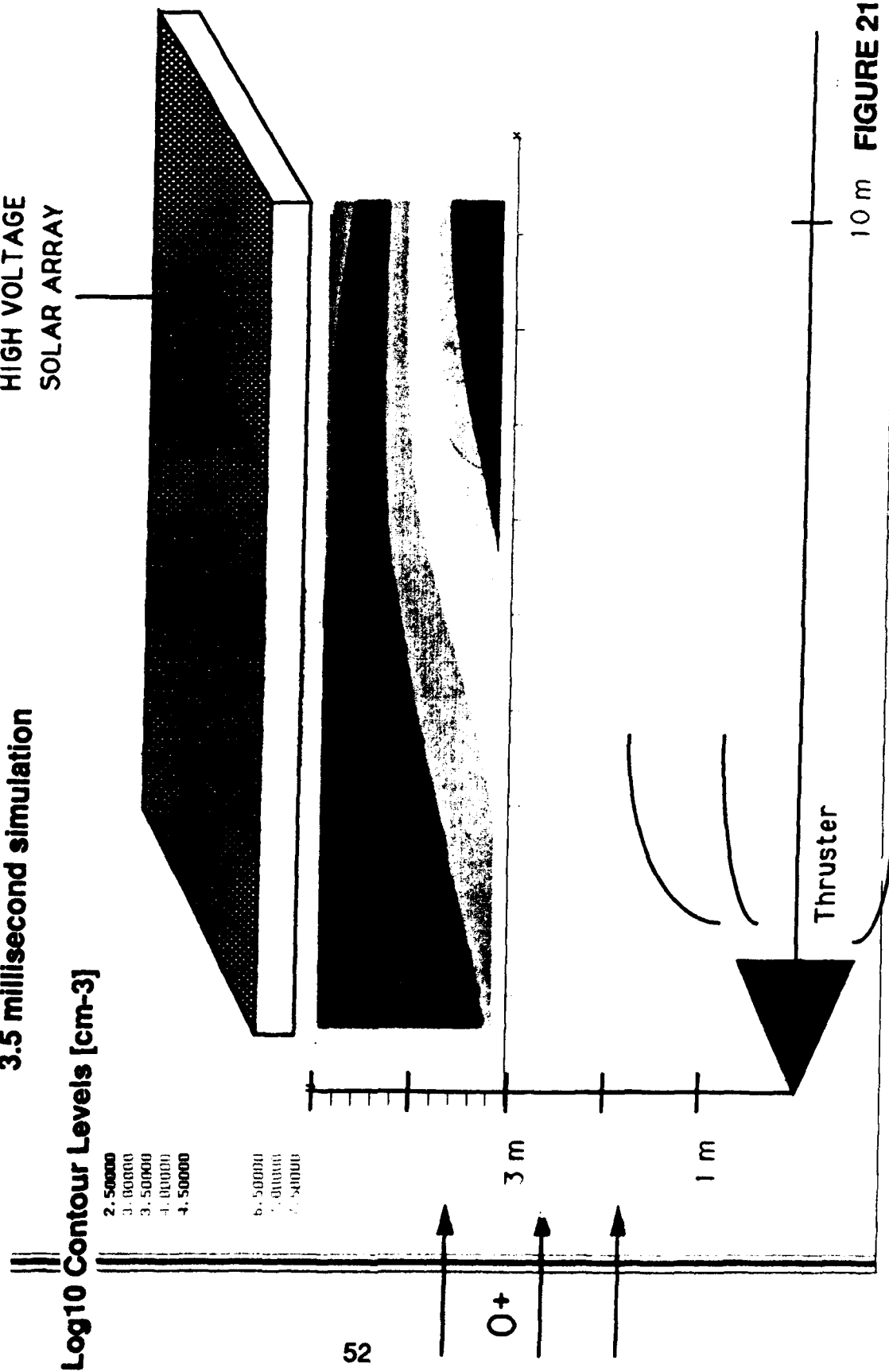


FIGURE 21

Log10 Constant Number Density Contours of O+ [cm⁻³] 25 lbf Monopropellant Hydrazine Thruster Thrusting past solar array into wake 3.5 millisecond simulation

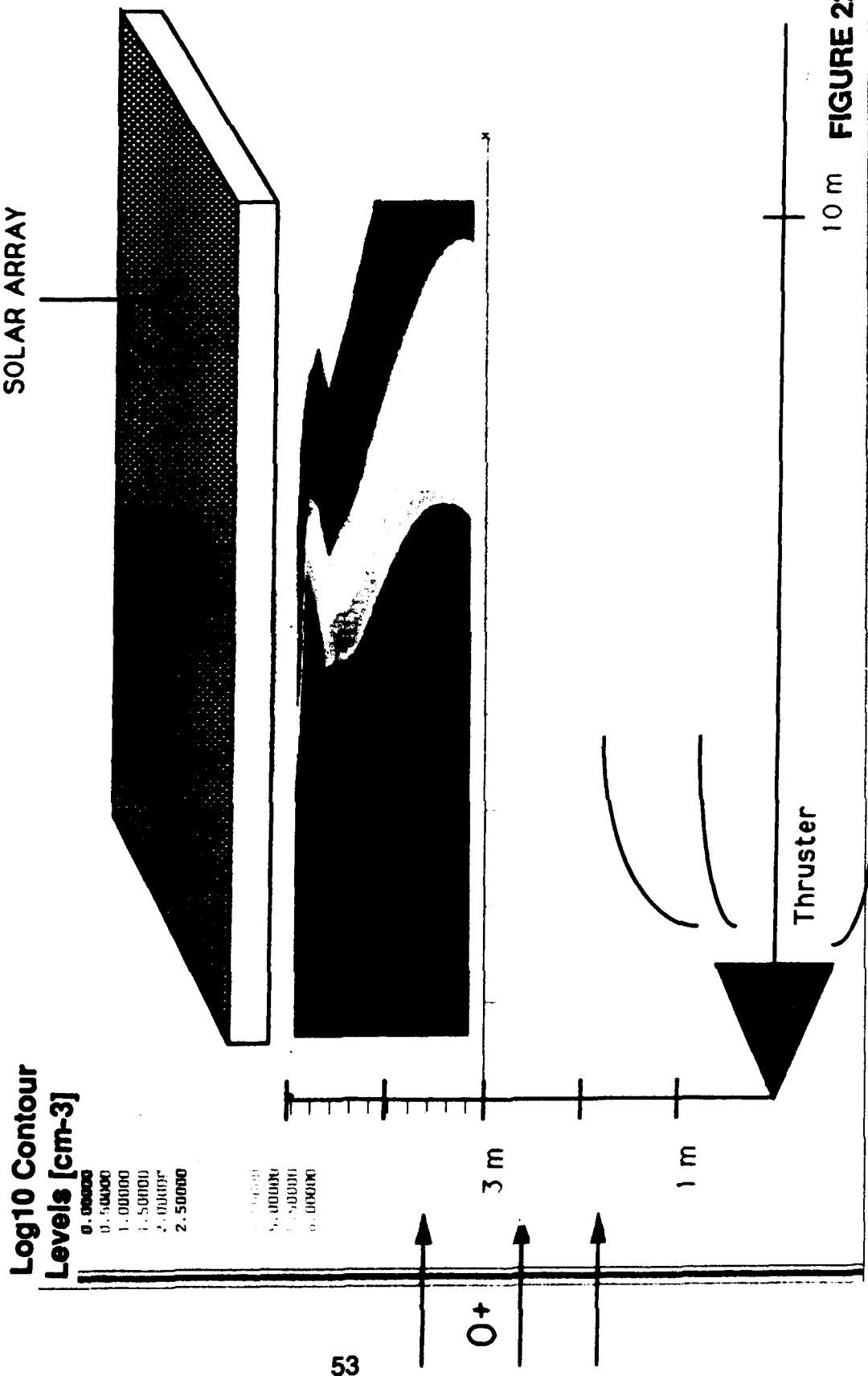


FIGURE 22

Log10 Constant Number Density Contours of Electrons [cm^{-3}]
 (Electron Density = Total Ion Density)
 25 lbf Monopropellant Hydrazine Thruster
 Thrusting past solar array into ram
 3.5 millisecond simulation

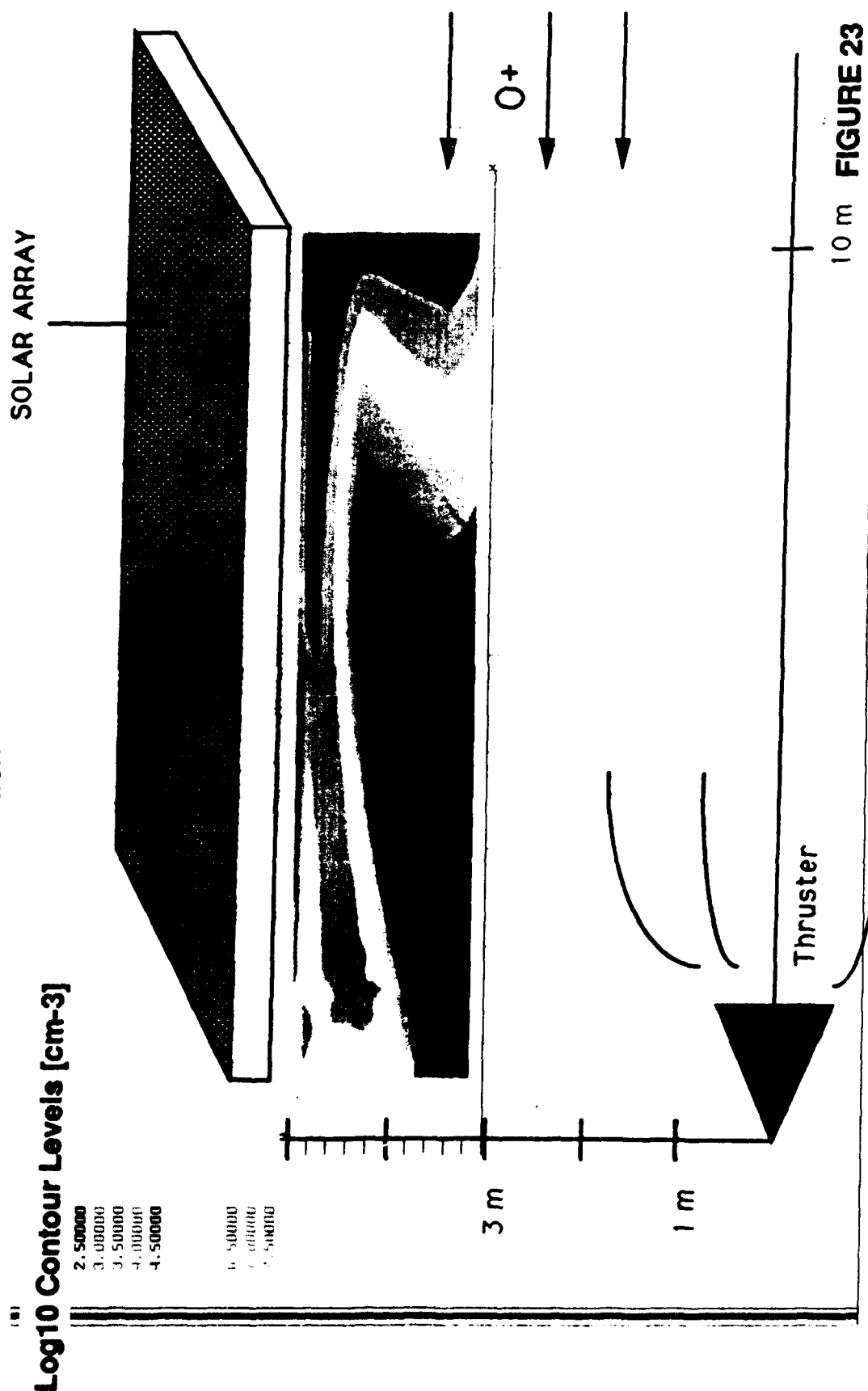


FIGURE 23

**Log10 Constant Number Density Contours of N_2^+ [cm^{-3}]
 25 lbf Monopropellant Hydrazine Thruster
 Thrusting past solar array into ram
 3.5 millisecond simulation**

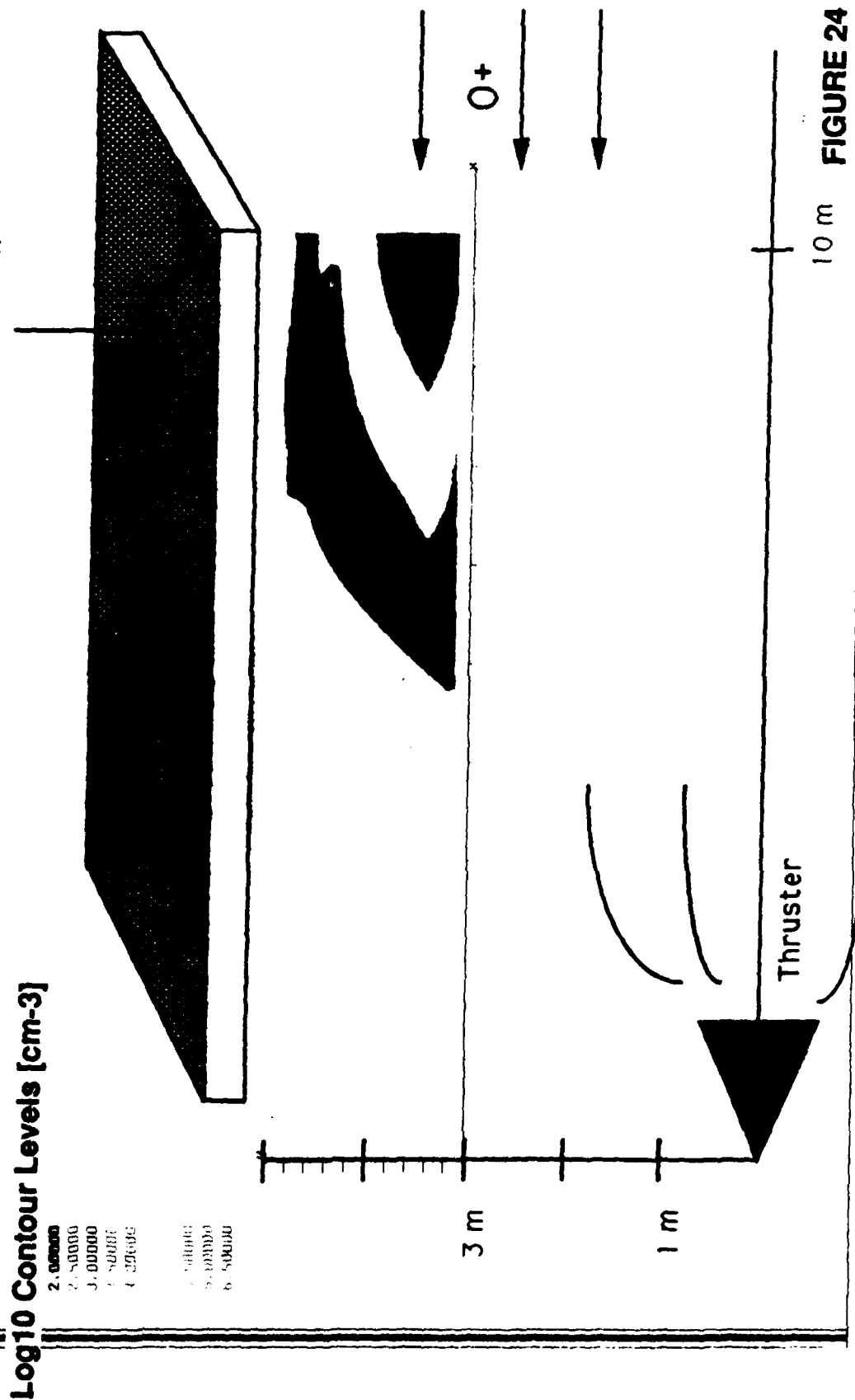


FIGURE 24

Log10 Constant Number Density Contours of Electrons [cm⁻³] 25 lbf Bipropellant Hydrazine Thruster Thrusting past solar array into wake 3.5 millisecond simulation

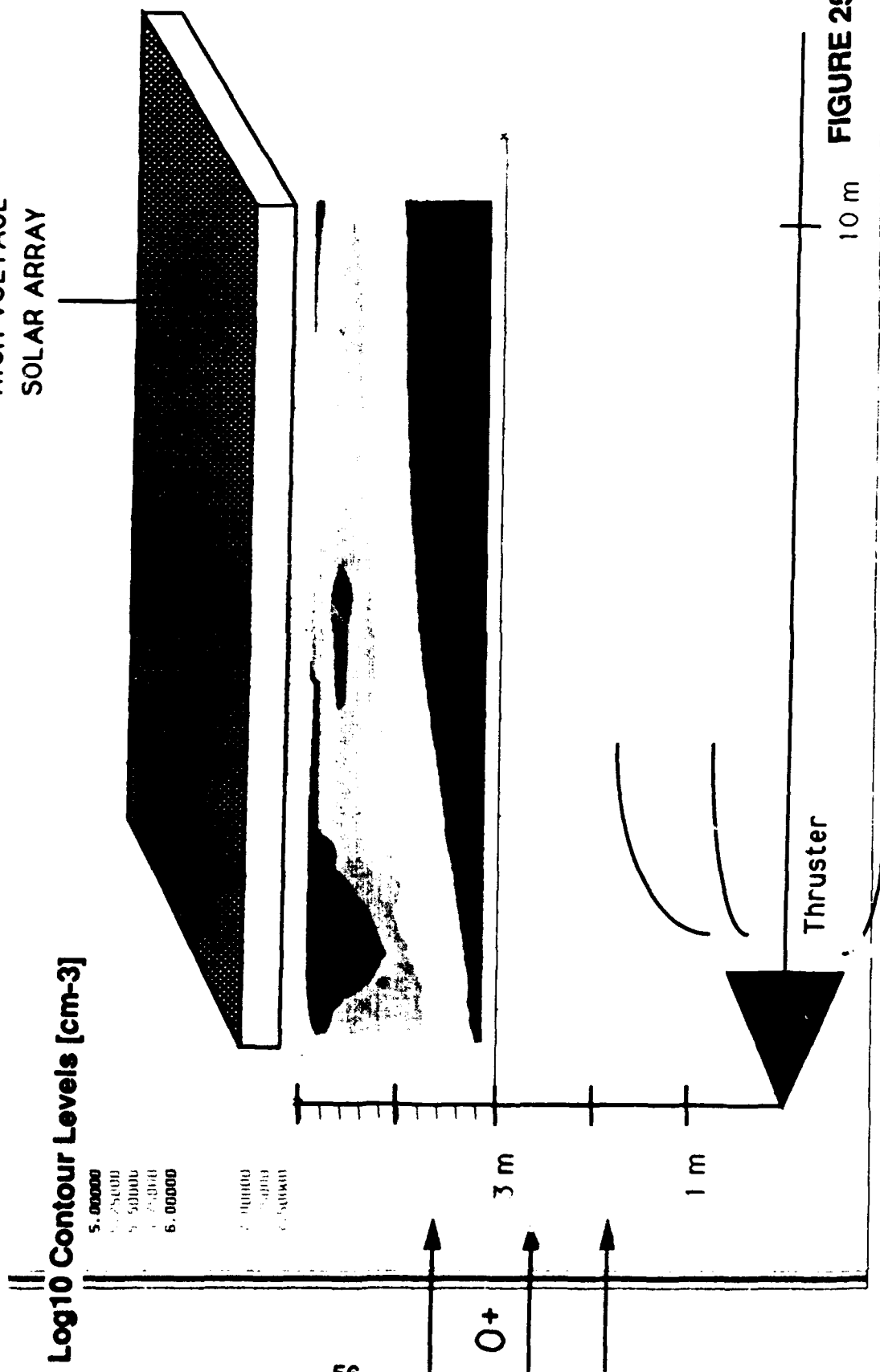


FIGURE 25

Log10 Constant Number Density Contours of CO_2^+ [cm^{-3}]
25 lbf Bipropellant Hydrazine Thruster
Thrusting past solar array into wake
3.5 millisecond simulation

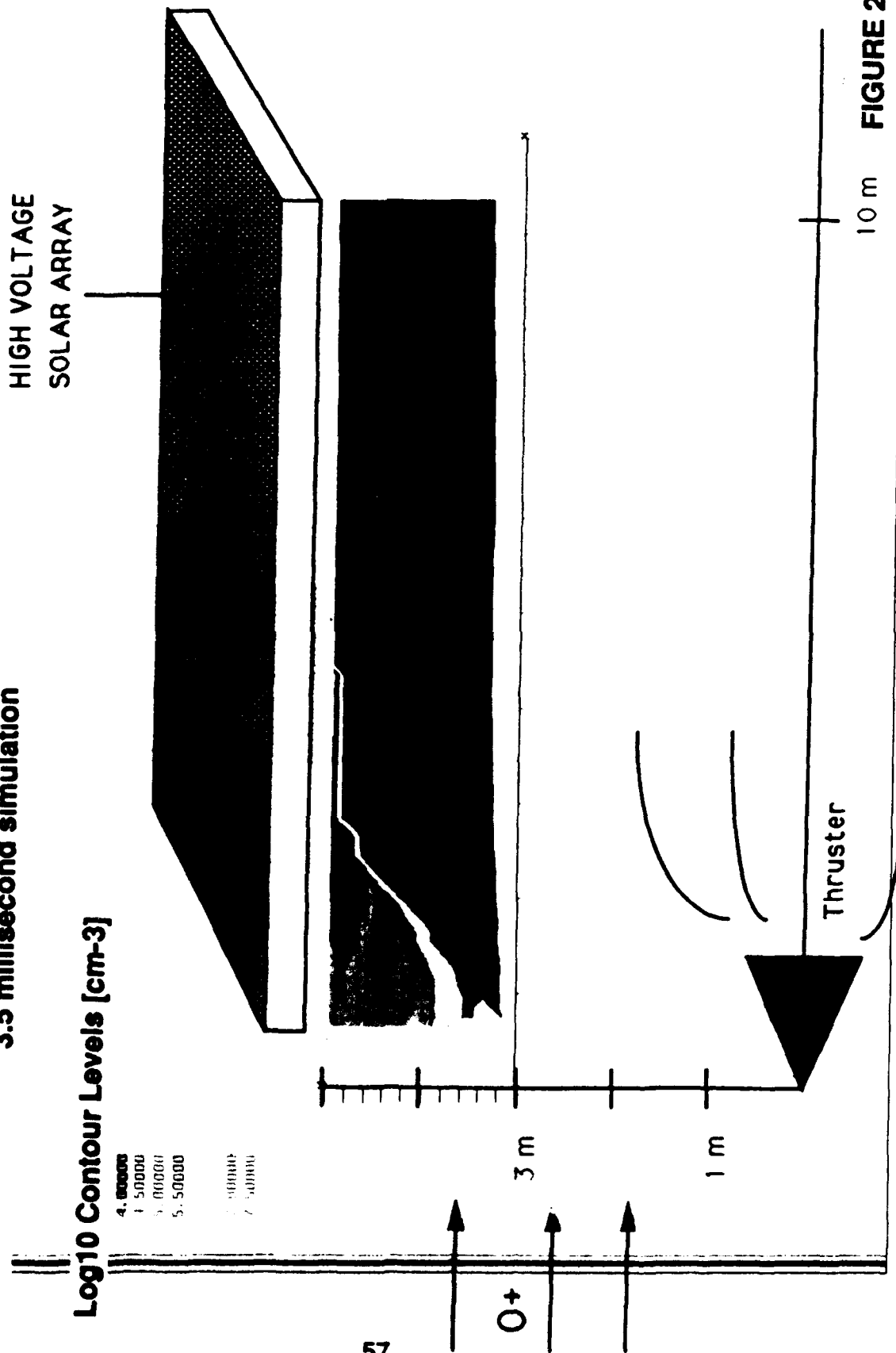


FIGURE 26

Log10 Constant Number Density Contours of Electrons [cm⁻³]

Electron Density = Total Ion Density

25 lbf Bipropellant Hydrazine Thruster

Thrusting past solar array into ram

3.5 millisecond simulation

HIGH VOLTAGE
SOLAR ARRAY

Log10 Contour Levels [cm⁻³]

5.00000
5.50000
6.00000
6.50000
7.00000

9.00000
9.50000
10.00000

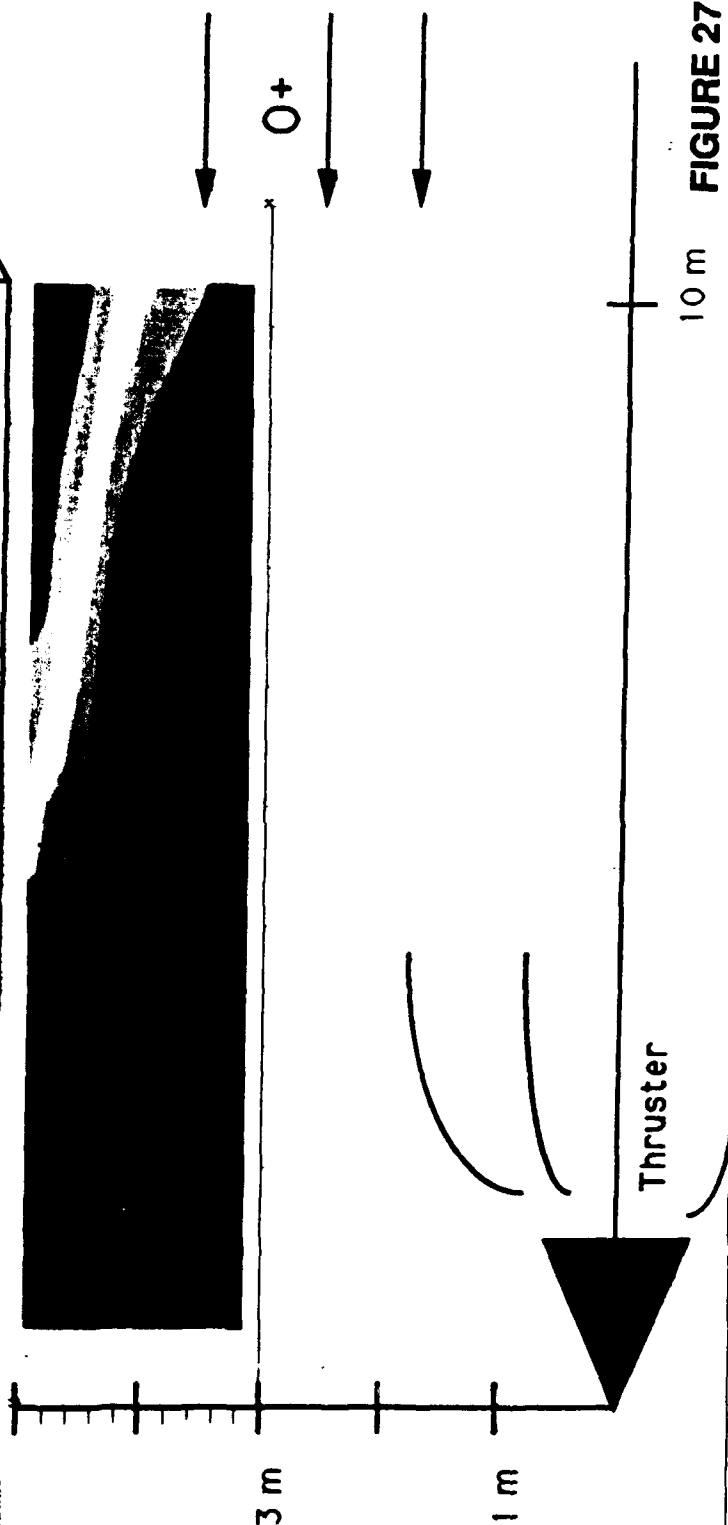


FIGURE 27

Log10 Constant Number Density Contours of CO_2^+ [cm^{-3}]
25 lbf Bipropellant Hydrazine Thruster
Thrusting past solar array into ram
3.5 millisecond simulation

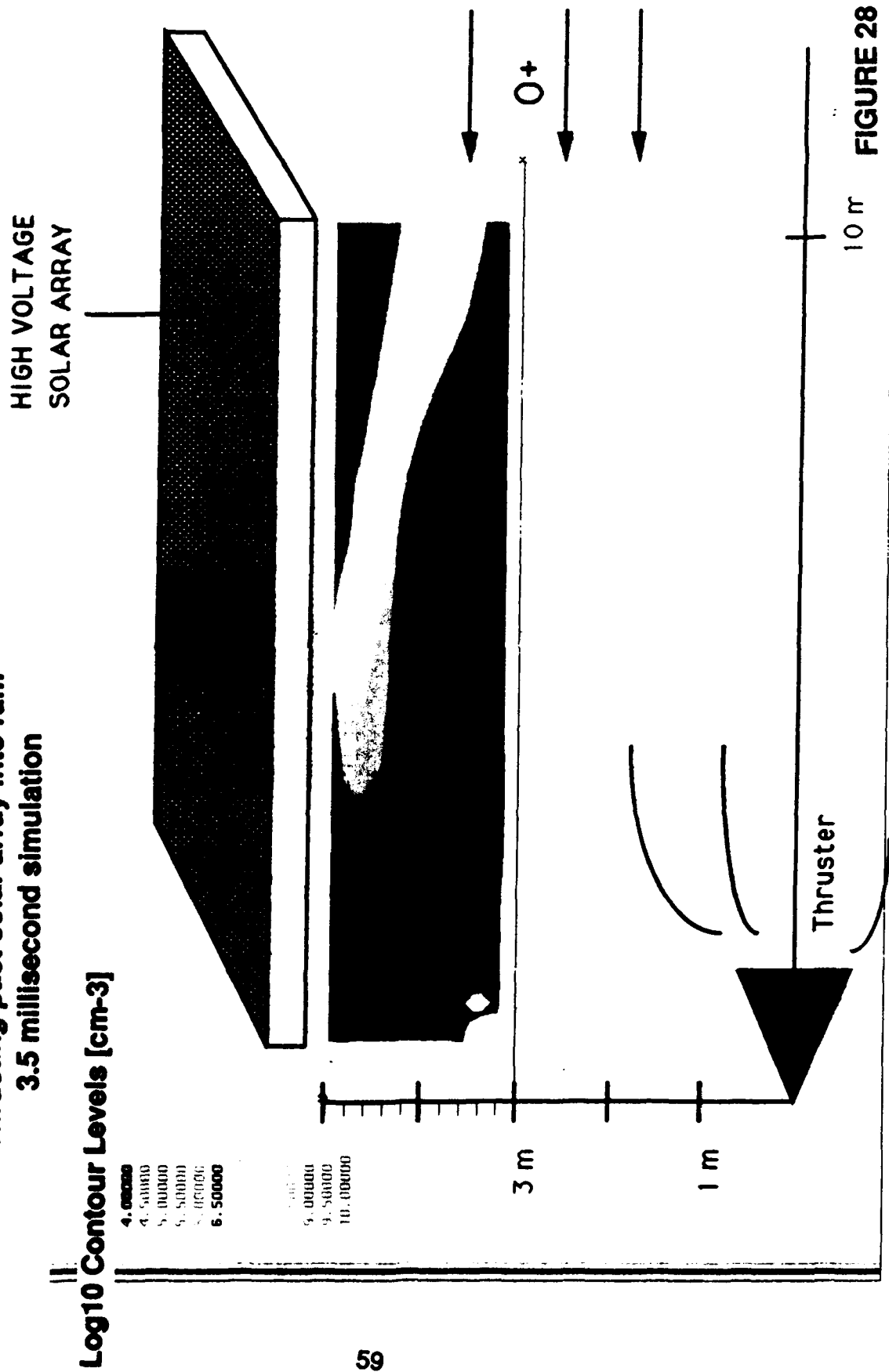


FIGURE 28

SCHEMATIC VIEW OF SOLAR ARRAY CHARGING PROCESS

Enhanced Field Electron Emission (EFEE)

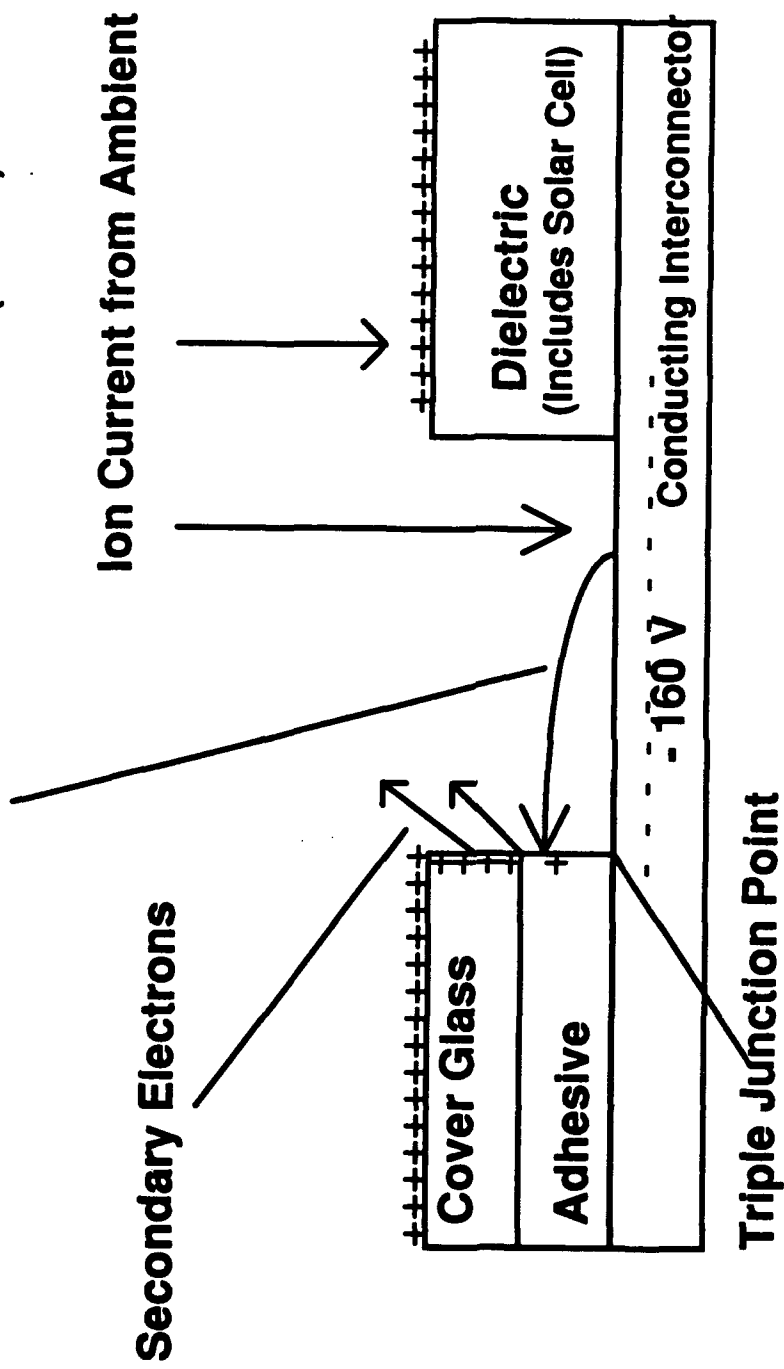
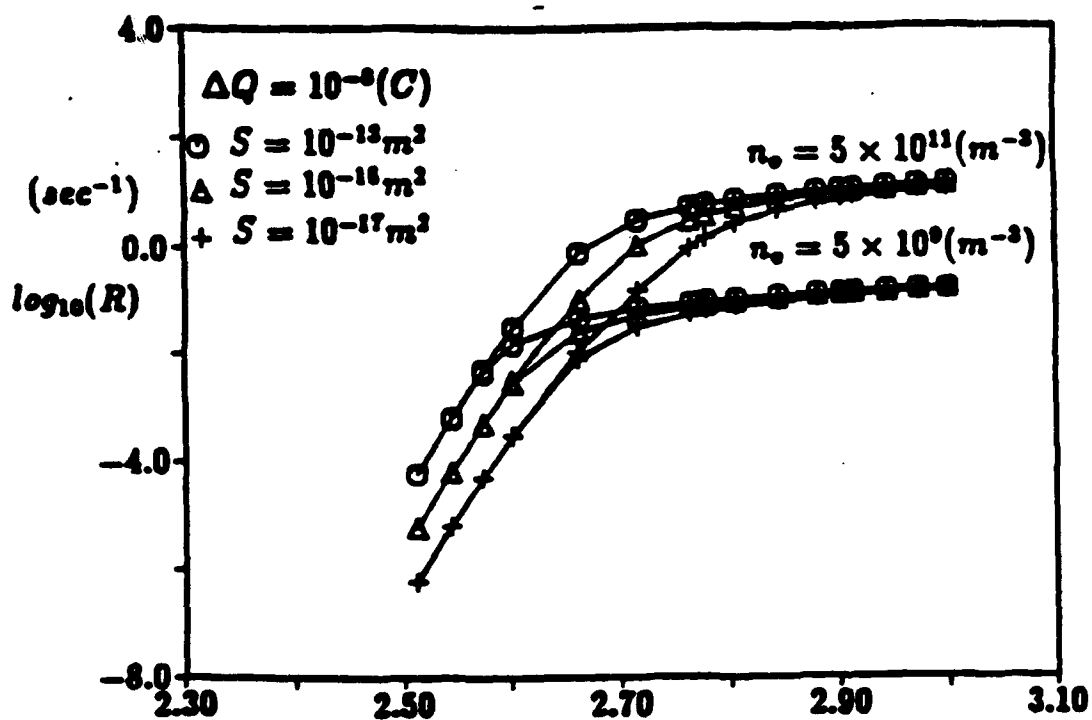


FIGURE 29



Log Total Arc Rate [s-1] vs Log Negative Bias Voltage [V]
for Two Different Densities [Ref. 21]

FIGURE 30

program fluxes

```
c-----
c  This program reads a 2-D array of neutral densities and velocities
c  (from ARRAY.DAT), and computes the ionization and ion densities
c  over time. The code has provisions for both monoprop hydrazine and
c  biprop hydrazine. The user is prompted for ambient O+ density and
c  direction, direction of photon flux, time step, and simulation time.
c  The current version has photo-ionization included only for the
c  monoprop plume; on short time scales (milliseconds), photo-ionization
c  is not very important compared to the other ionization mechanisms;
c  ion-molecule reactions (including charge exchange) and Critical
c  Ionization Velocity (CIV) effects.
c
c  The current grid has cell dimensions 10 by 10 cm, which for orbital
c  velocities of 7-8 km/s requires time steps no greater than 1e-6 s, due to
c  the Courant stability condition. If photo-ionization is included (which
c  it is for the monoprop set-up, the time step can be no greater
c  than 1e-10 s.
c
c  The output (ion species density in m-3) is written to the file ION.DAT.
c
c  fl(nx,ny)      : x fluxes
c  gl(nx,ny)      : y fluxes
c  nx,ny          : grid dimensions
c  dt             : stable time-step
c-----
c  fph1(nx,ny)=flux of photons (band 1) in x-direction [ph/cm2-s]
c  gph1(nx,ny)=flux of photons (band 1) in y-direction
c  fph2(nx,ny)=flux of photons (band 2) in x-direction
c  -
c  -
c  fh2(nx,ny) and gh2(nx,ny)      = h2-flux in x- and y-directions
c  fn2(nx,ny) and gn2(nx,ny)      = n2-flux
c  fnh3(nx,ny) and gnh3(nx,ny)    = nh3-flux
c  fop1(nx,ny) and gop1(nx,ny)    = O+-flux
c  fohp1(nx,ny) and gohp1(nx,ny)  = OH+-flux
c  fh2op1(nx,ny) and gh2op1(nx,ny)= H2O+-flux
c  fnop1(nx,ny) and gnop1(nx,ny)  = NO+-flux
c  fnh3p1(nx,ny) and gnh3p1(nx,ny)= NH3+-flux
c  fn2p1(nx,ny) and gn2p1(nx,ny)  = N2+-flux
c  fh2p1(nx,ny) and gh2p1(nx,ny)  = H2+-flux
c  fo2p1(nx,ny) and go2p1(nx,ny)  = O2+-flux
c  fcop1(nx,ny) and gcop1(nx,ny)  = CO+-flux
c  fco2p1(nx,ny) and gco2p1(nx,ny)= CO2+-flux
c  fh3op1(nx,ny) and gh3op1(nx,ny)= H3O+-flux
c
c  dh2(nx,ny)      = H2 number density [# /cm3]
c  dn2(nx,ny)      = N2
c  dnh3(nx,ny)     = NH3
c  dop1(nx,ny)     = O+
c  dohp1(nx,ny)    = OH+
c  dh2op1(nx,ny)   = H2O+
c  dnop1(nx,ny)    = NO+
c  dnh3p1(nx,ny)   = NH3+
c  dn2p1(nx,ny)    = N2+
c  dh2p1(nx,ny)    = H2+
c  do2p1(nx,ny)    = O2+
c  dcop1(nx,ny)    = CO+
c  dco2p1(nx,ny)   = CO2+
```

```

c   dh3opl(nx,ny) = H30+
c   de(nx,ny)     = Electrons
c   dph1(nx,ny)   = Photons (Wave band 1)
c   dph2(nx,ny)   = Photons (Wave band 2)
c   -
c   dph7(nx,ny)   = Photons (Wave band 7)

```

```

parameter (nx=90,ny=20) !MUST BE REPLACED FOR ALL SUBROUTINES
common/grid/dx,dy,dt     !IF CHANGED!!!
common/fluxph1/fph1(nx,ny),gph1(nx,ny)
common/fluxph2/fph2(nx,ny),gph2(nx,ny)
common/fluxph3/fph3(nx,ny),gph3(nx,ny)
common/fluxph4/fph4(nx,ny),gph4(nx,ny)
common/fluxph5/fph5(nx,ny),gph5(nx,ny)
common/fluxph6/fph6(nx,ny),gph6(nx,ny)
common/fluxph7/fph7(nx,ny),gph7(nx,ny)
common/fluxH2/fh2(nx,ny),gh2(nx,ny)
common/fluxN2/fn2(nx,ny),gn2(nx,ny)
common/fluxNH3/fnh3(nx,ny),gnh3(nx,ny)
common/fluxH2O/fh2o(nx,ny),gh2o(nx,ny)
common/fluxCO/fco(nx,ny),gco(nx,ny)
common/fluxCO2/fco2(nx,ny),gco2(nx,ny)
common/fluxOpl/fopl(nx,ny),gopl(nx,ny)
common/fluxOHpl/fohpl(nx,ny),gohpl(nx,ny)
common/fluxH2Opl/fh2opl(nx,ny),gh2opl(nx,ny)
common/fluxNOpl/fnopl(nx,ny),gnopl(nx,ny)
common/fluxNH3pl/fnh3pl(nx,ny),gnh3pl(nx,ny)
common/fluxN2pl/fn2pl(nx,ny),gn2pl(nx,ny)
common/fluxH2pl/fh2pl(nx,ny),gh2pl(nx,ny)
common/fluxO2pl/fo2pl(nx,ny),go2pl(nx,ny)
common/fluxCOpl/fcopl(nx,ny),gcopl(nx,ny)
common/fluxCO2pl/fco2pl(nx,ny),gco2pl(nx,ny)
common/fluxH3Opl/fh3opl(nx,ny),gh3opl(nx,ny)
common/dens/dh2(nx,ny),dn2(nx,ny),dnh3(nx,ny),
$ dh2o(nx,ny),dco(nx,ny),dco2(nx,ny),
$ dopl(nx,ny),dohpl(nx,ny),dh2opl(nx,ny),dnopl(nx,ny),
$ dnh3pl(nx,ny),dn2pl(nx,ny),de(nx,ny),dh2pl(nx,ny),
$ do2pl(nx,ny),dcopl(nx,ny),dco2pl(nx,ny),dh3opl(nx,ny),
$ dph1(nx,ny),dph2(nx,ny),dph3(nx,ny),dph4(nx,ny),
$ dph5(nx,ny),dph6(nx,ny),dph7(nx,ny)
common/veloneu/vxn(nx,ny),vyn(nx,ny)
common/veloion/vxoxy(nx,ny),vyoxy(nx,ny)
common/veloph/vxph(nx,ny),vyph(nx,ny)
common/rxrate/c1,c2,c3,c4,c5,c6,c7,c8,c9,c10
common/rxratebi/c11,c12,c13,c14,c15,c16,c17,c18,c19
common/phrateH2/cph3h2,cph4h2,cph5h2
common/phrateN2/cph3n2,cph4n2,cph5n2,cph6n2,cph7n2
common/phrateNH3/cph1nh3,cph2nh3,cph3nh3,cph4nh3
common/phrateH2O/cph1h2o,cph2h2o
common/phrateCO/cph2co,cph3co,cph4co,cph5co
common/phrateCO2/cph2co2,cph3co2,cph4co2,cph5co2
common/scrat_a/sc_ax(nx,ny),sc_ay(nx,ny)
common/scrat_b/sc_bx(nx,ny),sc_by(nx,ny)
common/scratn/sc_bxh2(nx,ny),sc_byh2(nx,ny),
$ sc_bxn2(nx,ny),sc_byn2(nx,ny),
$ sc_bxnh3(nx,ny),sc_bynh3(nx,ny),
$ sc_bxh2o(nx,ny),sc_byh2o(nx,ny),
$ sc_bxco(nx,ny),sc_byco(nx,ny),
$ sc_bxco2(nx,ny),sc_byco2(nx,ny),
$ sc_bxph1(nx,ny),sc_byph1(nx,ny),sc_bxph2(nx,ny),

```

```

$ sc_byph2(nx,ny),sc_bxph3(nx,ny),sc_byph3(nx,ny),
$ sc_bxph4(nx,ny),sc_byph4(nx,ny),sc_bxph5(nx,ny),
$ sc_byph5(nx,ny),sc_bxph6(nx,ny),sc_byph6(nx,ny),
$ sc_bxph7(nx,ny),sc_byph7(nx,ny)
common/civn2lim/civn21,civn22,civn23
common/civco2lim/civco21,civco22,civco23
common/civ/dciv(nx,ny),vcivn2(nx,ny),vcivco2(nx,ny)
common/omega/omegaN2,omegaC02
common/elrest1/elfluence,in2restrict,ico2restrict
common/elrest2/dn2plcheck(nx,ny),dco2plcheck(nx,ny)
common/dph/dph1amb,dph2amb,dph3amb,dph4amb,
& dph5amb,dph6amb,dph7amb
dimension xcoord(nx,ny),ycoord(nx,ny),dneutr(nx,ny)
dimension dph(nx,ny)

```

```

open(unit=1,file='ion.dat')
open(unit=2,file='array.dat')
do 10 i=1,nx
do 10 j=1,ny
  read(2,*)xcoord(i,j),ycoord(i,j),dneutr(i,j),
& vxn(i,j),vyn(i,j)
  dneutr(i,j)=dneutr(i,j)*1.e-6 !Convert m-3 to cm-3
  vxn(i,j)=vxn(i,j)*100. !Convert m/s to cm/s
  vyn(i,j)=vyn(i,j)*100.
10 continue
dx=10. ![cm]
dy=10. ![cm]
te=1500. !electron temp.
veloelec=3.e7 !Ambient electron velocity in LEO [cm/s]
omegaN2=3429. !N2 gyro frequency
omegaC02=2182. !C02 gyro frequency
civn21=1.e6 !Density limits that CIV rates are valid within
civn22=1.5e7
civn23=6.e7
civco21=1.e6
civco22=2.5e7
civco23=7.e7
c All reaction rates in cm3/s
c1=1.7e-9 !O+ + H2 -> OH+ + H
c2=7.5e-8*(sqrt(300./te)) !OH+ + e- -> O* + H
c3=1.5e-9 !OH+ + H2 -> H2O+ + H
c4=6.5e-7*(sqrt(300./te)) !H2O+ + e- -> OH* + H
c5=3.e-10 !O+ + N2 -> NO+ + N
c6=2.35e-8 !NO+ + e- -> N + O(2.75 eV)
c7=1.2e-9 !O+ + NH3 -> O + NH3+
c8=1.e-7 !NH3+ + e- -> NH3 (just estimated!!!)
c9=1090. !NH3/CIV
c10=545. !N2/CIV(estimated)
c The remaining rate constants are for biprop mix only:
c11=6.e-10 !O+ + H2O -> H2O+ + O (at 5eV)
c12=0. !CO2 + O+ -> O2+ + CO (at 5eV)
c13=2180. !CO2/CIV (max)
c14=6.e-10 !O+ + CO2 -> CO2+ + O (at 5 eV)
c15=3.8e-7 !CO2+ + e- -> CO* + O* (at 5 eV)
c16=1.9e-7*(sqrt(te/300.)) !O2+ + e- -> O(1D)+O(3P) (Rees)
c17=6.e-11 !O+ + CO -> CO+ + O
c18=1.67e-9 !H2O + H2O+ -> H3O+ + OH
c19=6.3e-7*(sqrt(300./te)) !H3O+ + e- -> neutrals

```

```

c      c = 2.99793e10 ! Photon speed [cm/s]
c      Photon reaction rates are (cross-section [cm2] * c[cm/s]):
c      H2 + hv -> H2+ + e-
c      cph3h2=6.e-18 * c      !800 - 630 Angstrom (Bauer,Phys.Planet.Ion.,p.54)
c      cph4h2=5.e-18 * c      !630 - 460 A
c      cph5h2=4.e-18 * c      !460 - 370 A

c      N2 + hv -> N2+ + e-
c      cph3n2=21.e-18 * c      !800 - 630 A ("High activity",Torr et al., 1979)
c      cph4n2=22.e-18 * c      !630 - 460 A
c      cph5n2=19.e-18 * c      !460 - 370 A
c      cph6n2=12.e-18 * c      !370 - 270 A
c      cph7n2= 6.e-18 * c      !270 - 205 A

c      NH3 + hv -> NH3+ + e-
c      cph1nh3=18.e-18 * c      !1027 - 911 A (Really TOTAL absorption,Hudson,71)
c      cph2nh3=25.e-18 * c      !911 - 800 A
c      cph3nh3=30.e-18 * c      !800 - 630 A
c      cph4nh3=25.e-18 * c      !630 - 550 (does not reach 460 A!)

c      H2O + hv -> H2O+ + e-
c      cph1h2o=14.e-18 * c      !1027 - 911 A (Really TOTAL absorption,Hudson,71)
c      cph2h2o=15.e-18 * c      !911 - 800 A

c      CO + hv -> CO+ + e-
c      cph2co=16.e-18 * c      !885 - 800 A (does not reach 911 A)
c      cph3co=20.e-18 * c      !800 - 630 A (Torr et al., 1979)
c      cph4co=17.e-18 * c      !630 - 460 A
c      cph5co=14.e-18 * c      !460 - 370 A

c      CO2 + hv -> CO2+ + e-
c      cph2co2=15.e-18 * c      !885 - 800 A (does not reach 911 A)
c      cph3co2=20.e-18 * c      !800 - 630 A (Torr et al., 1979)
c      cph4co2=25.e-18 * c      !630 - 460 A
c      cph5co2=25.e-18 * c      !460 - 370 A

c      Ambient photon density for the different wavelength bands
c      is the photon flux [# /cm2-s] divided by c [cm/s]:
c      (Fluxes are from S.J.Bauer, "Physics of Planetary Ionospheres",p.47)
c      dph1amb=11.61e9/c ![# /cm3]
c      dph2amb=8.3e9/c
c      dph3amb=2.4e9/c
c      dph4amb=4.7e9/c
c      dph5amb=0.63e9/c
c      dph6amb=10.3e9/c
c      dph7amb=4.4e9/c

write(*,*)'Monopropellant (48% H2, 29% N2, 22% NH3) or'
write(*,*)'biprop fuel (20%H2, 31%N2, 32%H2O, 11%CO, 6%CO2):'
write(*,*)'Enter 1 for monoprop, 2 for biprop: '
read(*,*) iprop
write(*,*)'Thruster firing into ram or wake?'
write(*,*)'Enter 1 for wake, -1 for ram: '
read(*,*) ivox
write(*,*)'Enter ambient 0+ density [cm-3]: '
read(*,*) densoxyg
write(*,*)'Enter -1, 0, or 1 for negative, no, or positive'
write(*,*)'component of photon velocity X component: '
read(*,*) ivxphoton

```

```

write(*,*)'Enter -1, 0, or 1 for negative, no, or positive'
write(*,*)'component of photon velocity Y component: '
read(*,*) ivyphoton

rvox=float(ivox)
rvxphoton=float(ivxphoton)
rvyphoton=float(ivyphoton)

if (iprop.eq.1) then
do 20 i=1,nx
do 20 j=1,ny
dh2(i,j)=0.48*dneutr(i,j)
dn2(i,j)=0.29*dneutr(i,j)
dnh3(i,j)=0.22*dneutr(i,j)
dh2o(i,j)=0.
dco(i,j)=0.
dco2(i,j)=0.
dn2plcheck(i,j)=1.
dco2plcheck(i,j)=1.
dciv(i,j)=dneutr(i,j)/densoxyg
vxoxy(i,j)=7.7e5*rvox
vyoxy(i,j)=0.
vxph(i,j)=c * rvxphoton
vyph(i,j)=c * rvyphoton
vcivn2(i,j)=abs(vxn(i,j)-vxoxy(i,j))/10.2e5
20 continue

else

do 30 i=1,nx
do 30 j=1,ny
dh2(i,j)=0.20*dneutr(i,j)
dn2(i,j)=0.31*dneutr(i,j)
dnh3(i,j)=0.
dh2o(i,j)=0.32*dneutr(i,j)
dco(i,j)=0.11*dneutr(i,j)
dco2(i,j)=0.06*dneutr(i,j)
dn2plcheck(i,j)=1.
dco2plcheck(i,j)=1.
dciv(i,j)=dneutr(i,j)/densoxyg
rvox=float(ivox)
vxoxy(i,j)=7.7e5*rvox
vyoxy(i,j)=0.
vxph(i,j)=c * rvxphoton
vyph(i,j)=c * rvyphoton
vcivn2(i,j)=abs(vxn(i,j) - vxoxy(i,j))/10.2e5
vcivco2(i,j)=abs(vxn(i,j) - vxoxy(i,j))/7.7e5
30 continue
endif

do 40 i=1,nx !Initial conditions
do 40 j=1,ny
dopl(i,j)=densoxyg
dohpl(i,j)=0.
dh2opl(i,j)=0.
dnopl(i,j)=0.
dnh3pl(i,j)=0.
dn2pl(i,j)=0.
dh2pl(i,j)=0.
dco2pl(i,j)=0.

```



```

do2pl(i,j)=0.
dcopl(i,j)=0.
dh3opl(i,j)=0.
de(i,j)=densoxyg
dph1(i,j)=dph1amb
dph2(i,j)=dph2amb
dph3(i,j)=dph3amb
dph4(i,j)=dph4amb
dph5(i,j)=dph5amb
dph6(i,j)=dph6amb
dph7(i,j)=dph7amb
40 continue
step=0.
time=0.
write(*,*)'Enter time step [seconds]: '
read(*,*) dt
write(*,*)'Enter simulation time [# of time steps]: '
read(*,*)isimtime
c The max ambient electron fluence available per time step:
c (electron density)*(electron velocity)*(time)
elfluence = densoxyg*veloelec*dt

if (iprop.eq.2) go to 100
do 50 n=1,isimtime
call neutral
call oplus
call ohplus
call h2oplus
call noplus
call nh3plus
call n2plus
call h2plus
call photon

do 170 i=2,nx-1
do 170 j=2,ny
dneutr(i,j)=dh2(i,j) + dn2(i,j) + dnh3(i,j)
& +dh2o(i,j) + dco(i,j) + dco2(i,j)
dph(i,j)=dph1(i,j) + dph2(i,j) + dph3(i,j) + dph4(i,j)
& + dph5(i,j) + dph6(i,j) + dph7(i,j)

if (iprop.eq.1) then
write(3,*)xcoord(i,j),ycoord(i,j),dneutr(i,j),de(i,j),
& dopl(i,j),dohpl(i,j),dh2opl(i,j),dnopl(i,j),dnh3pl(i,j),
& dn2pl(i,j),dh2pl(i,j),dph(i,j)
else
write(3,*)xcoord(i,j),ycoord(i,j),dneutr(i,j),de(i,j),
& dopl(i,j),dohpl(i,j),dh2opl(i,j),dnopl(i,j),dco2pl(i,j),
& dn2pl(i,j),do2pl(i,j),dcopl(i,j),dh3opl(i,j),dph(i,j)
endif
170 continue
do 60 j=1,ny
if (ivox.eq.1) then
dopl(1,j)=densoxyg
else
dopl(nx,j)=densoxyg
endif
if (ivxphoton.eq.1) then
dph1(1,j)=dph1amb
dph2(1,j)=dph2amb

```

```

    dph3(1,j)=dph3amb
    dph4(1,j)=dph4amb
    dph5(1,j)=dph5amb
    dph6(1,j)=dph6amb
    dph7(1,j)=dph7amb
endif
if (ivxphoton.eq.-1) then
    dph1(nx,j)=dph1amb
    dph2(nx,j)=dph2amb
    dph3(nx,j)=dph3amb
    dph4(nx,j)=dph4amb
    dph5(nx,j)=dph5amb
    dph6(nx,j)=dph6amb
    dph7(nx,j)=dph7amb
endif

    dh2(1,j)=0.48*dneutr(1,j)
    dn2(1,j)=0.29*dneutr(1,j)
    dnh3(1,j)=0.22*dneutr(1,j)
    dh2o(1,j)=0.
    dco(1,j)=0.
    dco2(1,j)=0.
    dciv(1,j)=dneutr(1,j)/densoxyg
60  continue

    do 70 i = 1,nx
    if (ivyphoton.eq.1) then
        dph1(i,1)=dph1amb
        dph2(i,1)=dph2amb
        dph3(i,1)=dph3amb
        dph4(i,1)=dph4amb
        dph5(i,1)=dph5amb
        dph6(i,1)=dph6amb
        dph7(i,1)=dph7amb
    endif
    if (ivyphoton.eq.-1) then
        dph1(i,ny)=dph1amb
        dph2(i,ny)=dph2amb
        dph3(i,ny)=dph3amb
        dph4(i,ny)=dph4amb
        dph5(i,ny)=dph5amb
        dph6(i,ny)=dph6amb
        dph7(i,ny)=dph7amb
    endif
70  continue

    do 80 i=2,nx
    dh2(i,1)=0.48*dneutr(i,1)
    dn2(i,1)=0.29*dneutr(i,1)
    dnh3(i,1)=0.22*dneutr(i,1)
    dh2o(1,j)=0.
    dco(1,j)=0.
    dco2(1,j)=0.
    dciv(i,1)=dneutr(i,1)/densoxyg
80  continue

    do 90 i=1,nx
    do 90 j=1,ny
        if (dopl(i,j).lt.0) dopl(i,j)=0
        de(i,j)=dopl(i,j)+dohpl(i,j)+dh2opl(i,j)+dnopl(i,j)

```

```

&      +dnh3pl(i,j)+dn2pl(i,j)+dh2pl(i,j)
90    continue
      time=time+dt
      step=step+1.
50    continue
      go to 155
100   continue

      do 110 n=1,isimtime
      call neutralbi
      call oplusbi
      call ohplus
      call h2oplusbi
      call noplus
      call co2plusbi
      call n2plus
      call o2plusbi
      call coplusbi
      call h3oplusbi
      call photon

do 120 j=1,ny
  if (ivox.eq.1) then
    dopl(1,j)=densoxyg
  else
    dopl(nx,j)=densoxyg
  endif
  if (ivxphoton.eq.1) then
    dph1(1,j)=dph1amb
    dph2(1,j)=dph2amb
    dph3(1,j)=dph3amb
    dph4(1,j)=dph4amb
    dph5(1,j)=dph5amb
    dph6(1,j)=dph6amb
    dph7(1,j)=dph7amb
  endif
  if (ivxphoton.eq.-1) then
    dph1(nx,j)=dph1amb
    dph2(nx,j)=dph2amb
    dph3(nx,j)=dph3amb
    dph4(nx,j)=dph4amb
    dph5(nx,j)=dph5amb
    dph6(nx,j)=dph6amb
    dph7(nx,j)=dph7amb
  endif

  dh2(i,1)=0.20*dneutr(i,1)
  dn2(i,1)=0.31*dneutr(i,1)
  dnh3(i,1)=0.
  dh2o(i,1)=0.32*dneutr(i,1)
  dco(i,1)=0.11*dneutr(i,1)
  dco2(i,1)=0.06*dneutr(i,1)
  dciv(1,j)=dneutr(1,j)/densoxyg
120  continue

do 130 i = 1,nx
  if (ivyphoton.eq.1) then
    dph1(i,1)=dph1amb
    dph2(i,1)=dph2amb
    dph3(i,1)=dph3amb

```

```

    dph4(i,1)=dph4amb
    dph5(i,1)=dph5amb
    dph6(i,1)=dph6amb
    dph7(i,1)=dph7amb
endif
if (ivyphoton.eq.-1) then
    dph1(i,ny)=dph1amb
    dph2(i,ny)=dph2amb
    dph3(i,ny)=dph3amb
    dph4(i,ny)=dph4amb
    dph5(i,ny)=dph5amb
    dph6(i,ny)=dph6amb
    dph7(i,ny)=dph7amb
endif
130 continue

do 140 i=2,nx
    dh2(i,1)=0.20*dneutr(i,1)
    dn2(i,1)=0.31*dneutr(i,1)
    dnh3(i,1)=0.
    dh2o(i,1)=0.32*dneutr(i,1)
    dco(i,1)=0.11*dneutr(i,1)
    dco2(i,1)=0.06*dneutr(i,1)
    dciv(i,1)=dneutr(i,1)/densoxyg
140 continue

do 150 i=1,nx
do 150 j=1,ny
    if (dopl(i,j).lt.0) dopl(i,j)=0
    de(i,j)=dopl(i,j)+dohpl(i,j)+dh2opl(i,j)+dnopl(i,j)
& +dn2pl(i,j)+do2pl(i,j)+dcopl(i,j)+dco2pl(i,j)
& +dh3opl(i,j)
150 continue
    time=time+dt
    step=step+1.
110 continue

155 continue

do 160 i=2,nx-1
do 160 j=2,ny
    dneutr(i,j)=dh2(i,j) + dn2(i,j) + dnh3(i,j)
& +dh2o(i,j) + dco(i,j) + dco2(i,j)
    dph(i,j)=dph1(i,j) + dph2(i,j) + dph3(i,j) + dph4(i,j)
& + dph5(i,j) + dph6(i,j) + dph7(i,j)

    if (iprop.eq.1) then
        write(1,*)xcoord(i,j),ycoord(i,j),dneutr(i,j),de(i,j),
& dopl(i,j),dohpl(i,j),dh2opl(i,j),dnopl(i,j),dnh3pl(i,j),
& dn2pl(i,j),dh2pl(i,j),dph(i,j)
    else
        write(1,*)xcoord(i,j),ycoord(i,j),dneutr(i,j),de(i,j),
& dopl(i,j),dohpl(i,j),dh2opl(i,j),dnopl(i,j),dco2pl(i,j),
& dn2pl(i,j),do2pl(i,j),dcopl(i,j),dh3opl(i,j),dph(i,j)
    endif
160 continue

end

subroutine photon

```

```

parameter (nx=90,ny=20)
common/grid/dx,dy,dt
common/fluxph1/fph1(nx,ny),gph1(nx,ny)
common/fluxph2/fph2(nx,ny),gph2(nx,ny)
common/fluxph3/fph3(nx,ny),gph3(nx,ny)
common/fluxph4/fph4(nx,ny),gph4(nx,ny)
common/fluxph5/fph5(nx,ny),gph5(nx,ny)
common/fluxph6/fph6(nx,ny),gph6(nx,ny)
common/fluxph7/fph7(nx,ny),gph7(nx,ny)
common/dens/dh2(nx,ny),dn2(nx,ny),dnh3(nx,ny),
$ dh2o(nx,ny),dco(nx,ny),dco2(nx,ny),
$ dopl(nx,ny),dohpl(nx,ny),dh2opl(nx,ny),dnopl(nx,ny),
$ dnh3pl(nx,ny),dn2pl(nx,ny),de(nx,ny),dh2pl(nx,ny),
$ do2pl(nx,ny),dcopl(nx,ny),dco2pl(nx,ny),dh3opl(nx,ny),
$ dph1(nx,ny),dph2(nx,ny),dph3(nx,ny),dph4(nx,ny),
$ dph5(nx,ny),dph6(nx,ny),dph7(nx,ny)
common/veloph/vxph(nx,ny),vyph(nx,ny)
common/phrateH2/cph3h2,cph4h2,cph5h2
common/phrateN2/cph3n2,cph4n2,cph5n2,cph6n2,cph7n2
common/phrateNH3/cph1nh3,cph2nh3,cph3nh3,cph4nh3
common/phrateH2O/cph1h2o,cph2h2o
common/phrateCO/cph2co,cph3co,cph4co,cph5co
common/phrateCO2/cph2co2,cph3co2,cph4co2,cph5co2
common/scrat_a/sc_ax(nx,ny),sc_ay(nx,ny)
common/scrat_b/sc_bx(nx,ny),sc_by(nx,ny)
common/scratn/sc_bxh2(nx,ny),sc_byh2(nx,ny),
$ sc_bxn2(nx,ny),sc_byn2(nx,ny),
$ sc_bxnh3(nx,ny),sc_bynh3(nx,ny),
$ sc_bxh2o(nx,ny),sc_byh2o(nx,ny),
$ sc_bxco(nx,ny),sc_byco(nx,ny),
$ sc_bxco2(nx,ny),sc_byco2(nx,ny),
$ sc_bxph1(nx,ny),sc_byph1(nx,ny),sc_bxph2(nx,ny),
$ sc_byph2(nx,ny),sc_bxph3(nx,ny),sc_byph3(nx,ny),
$ sc_bxph4(nx,ny),sc_byph4(nx,ny),sc_bxph5(nx,ny),
$ sc_byph5(nx,ny),sc_bxph6(nx,ny),sc_byph6(nx,ny),
$ sc_bxph7(nx,ny),sc_byph7(nx,ny)
common/dph/dph1amb,dph2amb,dph3amb,dph4amb,
& dph5amb,dph6amb,dph7amb

```

c----- initialize scratch arrays

c

```

do 100 i=1,nx
do 100 j=1,ny
  sc_ax(i,j) = 0.0
  sc_ay(i,j) = 0.0
  sc_bxph1(i,j) = 0.0
  sc_byph1(i,j) = 0.0
  sc_bxph2(i,j) = 0.0
  sc_byph2(i,j) = 0.0
  sc_bxph3(i,j) = 0.0
  sc_byph3(i,j) = 0.0
  sc_bxph4(i,j) = 0.0
  sc_byph4(i,j) = 0.0
  sc_bxph5(i,j) = 0.0
  sc_byph5(i,j) = 0.0
  sc_bxph6(i,j) = 0.0
  sc_byph6(i,j) = 0.0
  sc_bxph7(i,j) = 0.0
  sc_byph7(i,j) = 0.0

```

```

100 continue
c
    nx1=nx-1
    ny1=ny-1

    do 130 i=1,nx1
    do 130 j=1,ny1
        sc_ax(i,j) = 0.5 * (vxph(i+1,j) + vxph(i,j))
130 continue
    do 140 i=1,nx
    do 140 j=1,ny1
        sc_ay(i,j) = 0.5 * (vyph(i,j+1) + vyph(i,j))
140 continue
c
    do 150 i=1,nx1
    do 150 j=1,ny
        if(sc_ax(i,j).ge.0.)then
            sc_bxph1(i,j)=dph1(i,j)
            sc_bxph2(i,j)=dph2(i,j)
            sc_bxph3(i,j)=dph3(i,j)
            sc_bxph4(i,j)=dph4(i,j)
            sc_bxph5(i,j)=dph5(i,j)
            sc_bxph6(i,j)=dph6(i,j)
            sc_bxph7(i,j)=dph7(i,j)
        else
            sc_bxph1(i,j)=dph1(i+1,j)
            sc_bxph2(i,j)=dph2(i+1,j)
            sc_bxph3(i,j)=dph3(i+1,j)
            sc_bxph4(i,j)=dph4(i+1,j)
            sc_bxph5(i,j)=dph5(i+1,j)
            sc_bxph6(i,j)=dph6(i+1,j)
            sc_bxph7(i,j)=dph7(i+1,j)
        endif
150 continue

    do 160 i=1,nx
    do 160 j=1,ny1
        if(sc_ay(i,j).ge.0.)then
            sc_byph1(i,j) = dph1(i,j)
            sc_byph2(i,j) = dph2(i,j)
            sc_byph3(i,j) = dph3(i,j)
            sc_byph4(i,j) = dph4(i,j)
            sc_byph5(i,j) = dph5(i,j)
            sc_byph6(i,j) = dph6(i,j)
            sc_byph7(i,j) = dph7(i,j)
        else
            sc_byph1(i,j) = dph1(i,j+1)
            sc_byph2(i,j) = dph2(i,j+1)
            sc_byph3(i,j) = dph3(i,j+1)
            sc_byph4(i,j) = dph4(i,j+1)
            sc_byph5(i,j) = dph5(i,j+1)
            sc_byph6(i,j) = dph6(i,j+1)
            sc_byph7(i,j) = dph7(i,j+1)
        endif
160 continue

    do 170 i=1,nx1
    do 170 j=1,ny
        fph1(i,j) = sc_ax(i,j) * sc_bxph1(i,j)
        fph2(i,j) = sc_ax(i,j) * sc_bxph2(i,j)

```

```

      fph3(i,j) = sc_ax(i,j) * sc_bxph3(i,j)
      fph4(i,j) = sc_ax(i,j) * sc_bxph4(i,j)
      fph5(i,j) = sc_ax(i,j) * sc_bxph5(i,j)
      fph6(i,j) = sc_ax(i,j) * sc_bxph6(i,j)
      fph7(i,j) = sc_ax(i,j) * sc_bxph7(i,j)
170  continue
c
      do 180 i=1,nx
c
      do 180 j=1,ny1
      gph1(i,j) = sc_ay(i,j) * sc_byph1(i,j)
      gph2(i,j) = sc_ay(i,j) * sc_byph2(i,j)
      gph3(i,j) = sc_ay(i,j) * sc_byph3(i,j)
      gph4(i,j) = sc_ay(i,j) * sc_byph4(i,j)
      gph5(i,j) = sc_ay(i,j) * sc_byph5(i,j)
      gph6(i,j) = sc_ay(i,j) * sc_byph6(i,j)
      gph7(i,j) = sc_ay(i,j) * sc_byph7(i,j)
180  continue
c
      do 200 i=2,nx
      do 200 j=2,ny
      dph1(i,j) = dph1(i,j) -dt*(fph1(i,j)-fph1(i-1,j))/dx
+      -dt*(gph1(i,j)-gph1(i,j-1))/dy
+      -dt*dph1(i,j)*(cph1nh3*dnh3(i,j) + cph1h2o*dh2o(i,j))
c      To prevent numerical overflow:
      if (dph1(i,j).lt.1.e-12) dph1(i,j)=0.

      dph2(i,j) = dph2(i,j) -dt*(fph2(i,j)-fph2(i-1,j))/dx
+      -dt*(gph2(i,j)-gph2(i,j-1))/dy
+      -dt*dph2(i,j)*(cph2nh3*dnh3(i,j) + cph2h2o*dh2o(i,j)
+      + (cph2co*dco(i,j) + cph2co2*dco2(i,j))*(85./110.))

      if (dph2(i,j).lt.1.e-12) dph2(i,j)=0.

c      Photo-ionization yield of CO and CO2 within wavelength band 2
c      (911 - 800 Angstrom) is reduced since ionization cut-off
c      occurs at 885 A. (911-800)/(885-800)=(85./111.)

      dph3(i,j) = dph3(i,j) -dt*(fph3(i,j)-fph3(i-1,j))/dx
+      -dt*(gph3(i,j)-gph3(i,j-1))/dy
+      -dt*dph3(i,j)*(cph3h2*dh2(i,j) +cph3n2*dn2(i,j)
+      +cph3nh3*dnh3(i,j)
+      + cph3co*dco(i,j) + cph3co2*dco2(i,j))
      if (dph3(i,j).lt.1.e-12) dph3(i,j)=0.

      dph4(i,j) = dph4(i,j) -dt*(fph4(i,j)-fph4(i-1,j))/dx
+      -dt*(gph4(i,j)-gph4(i,j-1))/dy
+      -dt*dph4(i,j)*(cph4h2*dh2(i,j) +cph4n2*dn2(i,j)
+      +cph4nh3*dnh3(i,j)*(80./170.)
+      + cph4co*dco(i,j) + cph4co2*dco2(i,j))
      if (dph4(i,j).lt.1.e-12) dph4(i,j)=0.
c      Note that photo-ionization yield of NH3 within wavelength band 4
c      is reduced by a factor of 80/170. This is because 550 Angstrom is
c      the cut-off wavelength for ionization, whereas the band width
c      is 630 - 460 A. (630-550)/(630-460) = 80/170.
c
      dph5(i,j) = dph5(i,j) -dt*(fph5(i,j)-fph5(i-1,j))/dx
+      -dt*(gph5(i,j)-gph5(i,j-1))/dy
+      -dt*dph5(i,j)*(cph5h2*dh2(i,j) +cph5n2*dn2(i,j)
+      + cph5co*dco(i,j) + cph5co2*dco2(i,j))

```

```

        if (dph5(i,j).lt.1.e-12) dph5(i,j)=0.

        dph6(i,j) = dph6(i,j) -dt*(fph6(i,j)-fph6(i-1,j))/dx
+               -dt*(gph6(i,j)-gph6(i,j-1))/dy
+               -dt*dph6(i,j)*(cph6n2*dn2(i,j))
        if (dph6(i,j).lt.1.e-12) dph6(i,j)=0.

        dph7(i,j) = dph7(i,j) -dt*(fph7(i,j)-fph7(i-1,j))/dx
+               -dt*(gph7(i,j)-gph7(i,j-1))/dy
+               -dt*dph7(i,j)*(cph7n2*dn2(i,j))
        if (dph7(i,j).lt.1.e-12) dph7(i,j)=0.

```

```

200 continue
    return
end

```

```

c
c
c

```

```

subroutine neutral

```

```

parameter (nx=90,ny=20)
common/grid/dx,dy,dt
common/fluxH2/fh2(nx,ny),gh2(nx,ny)
common/fluxN2/fn2(nx,ny),gn2(nx,ny)
common/fluxNH3/fnh3(nx,ny),gnh3(nx,ny)
common/dens/dh2(nx,ny),dn2(nx,ny),dnh3(nx,ny),
$ dh2o(nx,ny),dco(nx,ny),dco2(nx,ny),
$ dopl(nx,ny),dohpl(nx,ny),dh2opl(nx,ny),dnopl(nx,ny),
$ dnh3pl(nx,ny),dn2pl(nx,ny),de(nx,ny),dh2pl(nx,ny),
$ do2pl(nx,ny),dcopl(nx,ny),dco2pl(nx,ny),dh3opl(nx,ny),
$ dph1(nx,ny),dph2(nx,ny),dph3(nx,ny),dph4(nx,ny),
$ dph5(nx,ny),dph6(nx,ny),dph7(nx,ny)
common/veloneu/vxn(nx,ny),vyn(nx,ny)
common/veloion/vxoxy(nx,ny),vyoxy(nx,ny)
common/rxrate/c1,c2,c3,c4,c5,c6,c7,c8,c9,c10
common/phrateH2/cph3h2,cph4h2,cph5h2
common/phrateN2/cph3n2,cph4n2,cph5n2,cph6n2,cph7n2
common/phrateNH3/cph1nh3,cph2nh3,cph3nh3,cph4nh3
common/scrat_a/sc_ax(nx,ny),sc_ay(nx,ny)
common/scrat_b/sc_bx(nx,ny),sc_by(nx,ny)
common/scratn/sc_bxh2(nx,ny),sc_byh2(nx,ny),
$ sc_bxn2(nx,ny),sc_byn2(nx,ny),
$ sc_bxnh3(nx,ny),sc_bynh3(nx,ny),
$ sc_bxh2o(nx,ny),sc_byh2o(nx,ny),
$ sc_bxco(nx,ny),sc_byco(nx,ny),
$ sc_bxco2(nx,ny),sc_byco2(nx,ny),
$ sc_bxph1(nx,ny),sc_byph1(nx,ny),sc_bxph2(nx,ny),
$ sc_byph2(nx,ny),sc_bxph3(nx,ny),sc_byph3(nx,ny),
$ sc_bxph4(nx,ny),sc_byph4(nx,ny),sc_bxph5(nx,ny),
$ sc_byph5(nx,ny),sc_bxph6(nx,ny),sc_byph6(nx,ny),
$ sc_bxph7(nx,ny),sc_byph7(nx,ny)
common/civn2lim/civn21,civn22,civn23
common/civ/dciv(nx,ny),vcivn2(nx,ny),vcivco2(nx,ny)
common/omega/omegaN2,omegaC02

```

```

c----- initialize scratch arrays
c

```

```

do 100 i=1,nx
do 100 j=1,ny
    sc_ax(i,j) = 0.0
    sc_ay(i,j) = 0.0

```



```

        sc_bxh2(i,j) = 0.0
        sc_byh2(i,j) = 0.0
        sc_bxn2(i,j) = 0.0
        sc_byn2(i,j) = 0.0
        sc_bxnh3(i,j) = 0.0
        sc_bynh3(i,j) = 0.0
100  continue
c
        nx1=nx-1
        ny1=ny-1
c
        do 130 i=1,nx1
        do 130 j=1,ny1
            sc_ax(i,j) = 0.5 * (vxn(i+1,j) + vxn(i,j))
130  continue
        do 140 i=1,nx
        do 140 j=1,ny1
            sc_ay(i,j) = 0.5 * (vyn(i,j+1) + vyn(i,j))
140  continue
c
        do 150 i=1,nx1
        do 150 j=1,ny1
            if(sc_ax(i,j).ge.0.)then
                sc_bxn2(i,j)=dn2(i,j)
                sc_bxh2(i,j)=dh2(i,j)
                sc_bxnh3(i,j)=dnh3(i,j)
            else
                sc_bxn2(i,j)=dn2(i+1,j)
                sc_bxh2(i,j)=dh2(i+1,j)
                sc_bxnh3(i,j)=dnh3(i+1,j)
            endif
150  continue
c
        do 160 i=1,nx
        do 160 j=1,ny1
            if(sc_ay(i,j).ge.0.)then
                sc_byn2(i,j) = dn2(i,j)
                sc_byh2(i,j) = dh2(i,j)
                sc_bynh3(i,j) = dnh3(i,j)
            else
                sc_byn2(i,j) = dn2(i,j+1)
                sc_byh2(i,j) = dh2(i,j+1)
                sc_bynh3(i,j) = dnh3(i,j+1)
            endif
160  continue
c
c
        do 170 i=1,nx1
        do 170 j=1,ny1
            fh2(i,j) = sc_ax(i,j) * sc_bxh2(i,j)
            fn2(i,j) = sc_ax(i,j) * sc_bxn2(i,j)
            fnh3(i,j) = sc_ax(i,j) * sc_bxnh3(i,j)
170  continue
c
        do 180 i=1,nx
c
        do 180 j=1,ny1
            gh2(i,j) = sc_ay(i,j) * sc_byh2(i,j)
            gn2(i,j) = sc_ay(i,j) * sc_byn2(i,j)
            gnh3(i,j) = sc_ay(i,j) * sc_bynh3(i,j)

```

180 continue

c

```
do 200 i=2,nx
do 200 j=2,ny
  dh2(i,j) = dh2(i,j) -dt*(fh2(i,j)-fh2(i-1,j))/dx
&               -dt*(gh2(i,j)-gh2(i,j-1))/dy
&               -dt*dh2(i,j)*(c1*dopl(i,j) +c3*dohpl(i,j)
& + cph3h2*dph3(i,j) + cph4h2*dph4(i,j) + cph5h2*dph5(i,j))
  if (abs(vxn(i,j)-vxoxy(i,j)) .ge. 10.2e5) then !Vcrit = 10.2 km/s

  if (dciv(i,j).ge.civn21 .and. dciv(i,j).le.civn22) then
    c10=(5.e-7*dciv(i,j) - 0.17)*(vcivn2(i,j)/1.5)**5
    c10=c10*omegaN2
    dn2(i,j) = dn2(i,j) -dt*(fn2(i,j)-fn2(i-1,j))/dx
&               -dt*(gn2(i,j)-gn2(i,j-1))/dy
&               -dt*dn2(i,j)*(c5*dopl(i,j)
& + cph3n2*dph3(i,j) + cph4n2*dph4(i,j) + cph5n2*dph5(i,j)
& + cph6n2*dph6(i,j) + cph7n2*dph7(i,j))
&               -de(i,j)*(exp(c10*dt)-1.)
  endif

  if (dciv(i,j).gt.civn22 .and. dciv(i,j).le.civn23) then
    c10=(7.8 - 1.2e-7*dciv(i,j))*(vcivn2(i,j)/1.5)**5
    c10=c10*omegaN2
    dn2(i,j) = dn2(i,j) -dt*(fn2(i,j)-fn2(i-1,j))/dx
&               -dt*(gn2(i,j)-gn2(i,j-1))/dy
&               -dt*dn2(i,j)*(c5*dopl(i,j)
& + cph3n2*dph3(i,j) + cph4n2*dph4(i,j) + cph5n2*dph5(i,j)
& + cph6n2*dph6(i,j) + cph7n2*dph7(i,j))
&               -de(i,j)*(exp(c10*dt)-1.)
  endif

  else
    dn2(i,j) = dn2(i,j) -dt*(fn2(i,j)-fn2(i-1,j))/dx
&               -dt*(gn2(i,j)-gn2(i,j-1))/dy
&               -dt*dn2(i,j)*(c5*dopl(i,j)
& + cph3n2*dph3(i,j) + cph4n2*dph4(i,j) + cph5n2*dph5(i,j)
& + cph6n2*dph6(i,j) + cph7n2*dph7(i,j))
  endif
  dnh3(i,j) = dnh3(i,j)-dt*(fnh3(i,j)-fnh3(i-1,j))/dx
&               -dt*(gnh3(i,j)-gnh3(i,j-1))/dy
&               -dt*dnh3(i,j)*(c7*dopl(i,j)
& + cph1nh3*dph1(i,j) + cph2nh3*dph2(i,j) + cph3nh3*dph3(i,j)
& + cph4nh3*dph4(i,j)*(80./170.))

200 continue
return
end
```

subroutine neutralbi

c

```
parameter (nx=90,ny=20)
common/grid/dx,dy,dt
common/fluxH2/fh2(nx,ny),gh2(nx,ny)
common/fluxN2/fn2(nx,ny),gn2(nx,ny)
common/fluxH2O/fh2o(nx,ny),gh2o(nx,ny)
common/fluxCO2/fco2(nx,ny),gco2(nx,ny)
common/fluxCO/fco(nx,ny),gco(nx,ny)
common/dens/dh2(nx,ny),dn2(nx,ny),dnh3(nx,ny),
$ dh2o(nx,ny),dco(nx,ny),dco2(nx,ny),
```

```

$ dopl(nx,ny),dohpl(nx,ny),dh2opl(nx,ny),dnopl(nx,ny),
$ dnh3pl(nx,ny),dn2pl(nx,ny),de(nx,ny),dh2pl(nx,ny),
$ do2pl(nx,ny),dcopl(nx,ny),dco2pl(nx,ny),dh3opl(nx,ny),
$ dph1(nx,ny),dph2(nx,ny),dph3(nx,ny),dph4(nx,ny),
$ dph5(nx,ny),dph6(nx,ny),dph7(nx,ny)
common/veloneu/vxn(nx,ny),vyn(nx,ny)
common/veloion/vxoxy(nx,ny),vyoxy(nx,ny)
common/rxrate/c1,c2,c3,c4,c5,c6,c7,c8,c9,c10
common/rxratebi/c11,c12,c13,c14,c15,c16,c17,c18,c19
common/scrat_a/sc_ax(nx,ny),sc_ay(nx,ny)
common/scrat_b/sc_bx(nx,ny),sc_by(nx,ny)
common/scratn/sc_bxh2(nx,ny),sc_byh2(nx,ny),
$ sc_bxn2(nx,ny),sc_byn2(nx,ny),
$ sc_bxnh3(nx,ny),sc_bynh3(nx,ny),
$ sc_bxh2o(nx,ny),sc_byh2o(nx,ny),
$ sc_bxco(nx,ny),sc_byco(nx,ny),
$ sc_bxco2(nx,ny),sc_byco2(nx,ny),
$ sc_bxph1(nx,ny),sc_byph1(nx,ny),sc_bxph2(nx,ny),
$ sc_byph2(nx,ny),sc_bxph3(nx,ny),sc_byph3(nx,ny),
$ sc_bxph4(nx,ny),sc_byph4(nx,ny),sc_bxph5(nx,ny),
$ sc_byph5(nx,ny),sc_bxph6(nx,ny),sc_byph6(nx,ny),
$ sc_bxph7(nx,ny),sc_byph7(nx,ny)
common/civn2lim/civn21,civn22,civn23
common/civco2lim/civco21,civco22,civco23
common/civ/dciv(nx,ny),vcivn2(nx,ny),vcivco2(nx,ny)
common/omega/omegaN2,omegaC02

```

c----- initialize scratch arrays

c

```

do 100 i=1,nx
do 100 j=1,ny
  sc_ax(i,j) = 0.0
  sc_ay(i,j) = 0.0
  sc_bxh2(i,j) = 0.0
  sc_byh2(i,j) = 0.0
  sc_bxn2(i,j) = 0.0
  sc_byn2(i,j) = 0.0
  sc_bxh2o(i,j) = 0.0
  sc_byh2o(i,j) = 0.0
  sc_bxco(i,j) = 0.0
  sc_byco(i,j) = 0.0
  sc_bxco2(i,j) = 0.0
  sc_byco2(i,j) = 0.0

```

100 continue

c

```

  nx1=nx-1
  ny1=ny-1

```

c

```

do 130 i=1,nx1
do 130 j=1,ny
  sc_ax(i,j) = 0.5 * (vxn(i+1,j) + vxn(i,j))

```

130 continue

```

do 140 i=1,nx
do 140 j=1,ny1
  sc_ay(i,j) = 0.5 * (vyn(i,j+1) + vyn(i,j))

```

140 continue

c

```

do 150 i=1,nx1
do 150 j=1,ny
  if(sc_ax(i,j).ge.0.)then

```

```

        sc_bxn2(i,j)=dn2(i,j)
        sc_bxh2(i,j)=dh2(i,j)
        sc_bxh2o(i,j)=dh2o(i,j)
        sc_bxco(i,j)=dco(i,j)
        sc_bxco2(i,j)=dco2(i,j)
    else
        sc_bxn2(i,j)=dn2(i+1,j)
        sc_bxh2(i,j)=dh2(i+1,j)
        sc_bxh2o(i,j)=dh2o(i+1,j)
        sc_bxco(i,j)=dco(i+1,j)
        sc_bxco2(i,j)=dco2(i+1,j)
    endif
150    continue
c
    do 160 i=1,nx
    do 160 j=1,ny1
        if(sc_ay(i,j).ge.0.)then
            sc_byn2(i,j) = dn2(i,j)
            sc_byh2(i,j) = dh2(i,j)
            sc_byh2o(i,j) = dh2o(i,j)
            sc_byco(i,j) = dco(i,j)
            sc_byco2(i,j) = dco2(i,j)
        else
            sc_byn2(i,j) = dn2(i,j+1)
            sc_byh2(i,j) = dh2(i,j+1)
            sc_byh2o(i,j) = dh2o(i,j+1)
            sc_byco(i,j) = dco(i,j+1)
            sc_byco2(i,j) = dco2(i,j+1)
        endif
160    continue
c
c
    do 170 i=1,nx1
    do 170 j=1,ny
        fh2(i,j) = sc_ax(i,j) * sc_bxh2(i,j)
        fn2(i,j) = sc_ax(i,j) * sc_bxn2(i,j)
        fh2o(i,j) = sc_ax(i,j) * sc_bxh2o(i,j)
        fco(i,j) = sc_ax(i,j) * sc_bxco(i,j)
        fco2(i,j) = sc_ax(i,j) * sc_bxco2(i,j)
170    continue
c
    do 180 i=1,nx
c
    do 180 j=1,ny1
        gh2(i,j) = sc_ay(i,j) * sc_byh2(i,j)
        gn2(i,j) = sc_ay(i,j) * sc_byn2(i,j)
        gh2o(i,j) = sc_ay(i,j) * sc_byh2o(i,j)
        gco(i,j) = sc_ay(i,j) * sc_byco(i,j)
        gco2(i,j) = sc_ay(i,j) * sc_byco2(i,j)
180    continue
c
    do 200 i=2,nx
    do 200 j=2,ny
        dh2(i,j) = dh2(i,j) -dt*(fh2(i,j)-fh2(i-1,j))/dx
&                -dt*(gh2(i,j)-gh2(i,j-1))/dy
&                -dt*dh2(i,j)*(c1*dopl(i,j) +c3*dohpl(i,j))

        if (abs(vxn(i,j)-vxoxy(i,j)).ge.10.2e5) then

        if (dciv(i,j).ge.civn21 .and. dciv(i,j).le.civn22) then

```

```

c10=(5.e-7*dciv(i,j) - 0.17)*(vcivn2(i,j)/1.5)**5
c10=c10*omegaN2
dn2(i,j) = dn2(i,j) -dt*(fn2(i,j)-fn2(i-1,j))/dx
&          -dt*(gn2(i,j)-gn2(i,j-1))/dy
&          -dt*dn2(i,j)*c5*dopl(i,j)
&          -de(i,j)*(exp(c10*dt)-1.)
endif

if (dciv(i,j).gt.civn22 .and. dciv(i,j).le.civn23) then
c10=(7.8 - 1.2e-7*dciv(i,j))*(vcivn2(i,j)/1.5)**5
c10=c10*omegaN2
dn2(i,j) = dn2(i,j) -dt*(fn2(i,j)-fn2(i-1,j))/dx
&          -dt*(gn2(i,j)-gn2(i,j-1))/dy
&          -dt*dn2(i,j)*c5*dopl(i,j)
&          -de(i,j)*(exp(c10*dt)-1.)
endif

else
dn2(i,j) = dn2(i,j) -dt*(fn2(i,j)-fn2(i-1,j))/dx
&          -dt*(gn2(i,j)-gn2(i,j-1))/dy
&          -dt*dn2(i,j)*c5*dopl(i,j)
endif

dh2o(i,j) = dh2o(i,j)-dt*(fh2o(i,j)-fh2o(i-1,j))/dx
&          -dt*(gh2o(i,j)-gh2o(i,j-1))/dy
&          -dt*(dh2o(i,j)*c11*dopl(i,j))
&          -dt*(dh2o(i,j)*c18*dh2opl(i,j))

dco(i,j) = dco(i,j)-dt*(fco(i,j)-fco(i-1,j))/dx
&          -dt*(gco(i,j)-gco(i,j-1))/dy
&          +dt*(dco2(i,j)*c12*dopl(i,j))

if (abs(vxn(i,j)-vxoxy(i,j)) .ge. 7.7e5) then

if (dciv(i,j).ge.civco21 .and. dciv(i,j).le.civco22) then
c13=(0.25 + 4.5e-7*dciv(i,j))*(vcivco2(i,j)/1.5)**5
c13=c13*omegaC02
dco2(i,j) = dco2(i,j)-dt*(fco2(i,j)-fco2(i-1,j))/dx
&          -dt*(gco2(i,j)-gco2(i,j-1))/dy
&          -dt*(dco2(i,j)*c12*dopl(i,j))
&          -dt*(dco2(i,j)*c14*dopl(i,j))
&          -de(i,j)*(exp(c13*dt)-1.)
endif

if (dciv(i,j).gt.civco22 .and. dciv(i,j).le.civco23) then
c13=(17. - 2.23e-7*dciv(i,j))*(vcivco2(i,j)/1.5)**5
c13=c13*omegaC02
dco2(i,j) = dco2(i,j)-dt*(fco2(i,j)-fco2(i-1,j))/dx
&          -dt*(gco2(i,j)-gco2(i,j-1))/dy
&          -dt*(dco2(i,j)*c12*dopl(i,j))
&          -dt*(dco2(i,j)*c14*dopl(i,j))
&          -de(i,j)*(exp(c13*dt)-1.)
endif

else
dco2(i,j) = dco2(i,j)-dt*(fco2(i,j)-fco2(i-1,j))/dx
&          -dt*(gco2(i,j)-gco2(i,j-1))/dy
&          -dt*(dco2(i,j)*c12*dopl(i,j))
&          -dt*(dco2(i,j)*c14*dopl(i,j))
endif
endif

```

```

200  continue
      return
      end
c
      subroutine oplus
c
      parameter (nx=90,ny=20)
      common/grid/dx,dy,dt
      common/flux0pl/fopl(nx,ny),gopl(nx,ny)
      common/dens/dh2(nx,ny),dn2(nx,ny),dnh3(nx,ny),
$ dh2o(nx,ny),dco(nx,ny),dco2(nx,ny),
$ dopl(nx,ny),dohpl(nx,ny),dh2opl(nx,ny),dnopl(nx,ny),
$ dnh3pl(nx,ny),dn2pl(nx,ny),de(nx,ny),dh2pl(nx,ny),
$ do2pl(nx,ny),dcopl(nx,ny),dco2pl(nx,ny),dh3opl(nx,ny),
$ dph1(nx,ny),dph2(nx,ny),dph3(nx,ny),dph4(nx,ny),
$ dph5(nx,ny),dph6(nx,ny),dph7(nx,ny)
      common/veloion/vxoxy(nx,ny),vyoxy(nx,ny)
      common/rxrate/c1,c2,c3,c4,c5,c6,c7,c8,c9,c10
      common/scrat_a/sc_ax(nx,ny),sc_ay(nx,ny)
      common/scrat_b/sc_bx(nx,ny),sc_by(nx,ny)

c-----      initialize scratch arrays
c
      do 100 i=1,nx
      do 100 j=1,ny
          sc_ax(i,j) = 0.0
          sc_ay(i,j) = 0.0
          sc_bx(i,j) = 0.0
          sc_by(i,j) = 0.0
100  continue
c
      nx1=nx-1
      ny1=ny-1

c
c      vrx =sc_ax(i,j)=0.5*(vx(i+1,j)+vx(i,j))
c
      do 130 i=1,nx1
      do 130 j=1,ny
          sc_ax(i,j) = 0.5 * (vxoxy(i+1,j) + vxoxy(i,j))
130  continue
      do 140 i=1,nx
      do 140 j=1,ny1
          sc_ay(i,j) = 0.5 * (vyoxy(i,j+1) + vyoxy(i,j))
140  continue
c
      do 150 i=1,nx1
      do 150 j=1,ny
          if(sc_ax(i,j).ge.0.)then
              sc_bx(i,j)=dopl(i,j)
          else
              sc_bx(i,j)=dopl(i+1,j)
          endif
150  continue
c
      do 160 i=1,nx
      do 160 j=1,ny1
          if(sc_ay(i,j).ge.0.)then
              sc_by(i,j) = dopl(i,j)
          else

```

```

        sc_by(i,j) = dopl(i,j+1)
    endif
160    continue
c
c
    do 170 i=1,nx1
        do 170 j=1,ny
            fopl(i,j) = sc_ax(i,j) * sc_bx(i,j)
170    continue
c
    do 180 i=1,nx
        do 180 j=1,ny1
            gopl(i,j) = sc_ay(i,j) * sc_by(i,j)
180    continue
c
    do 200 i=2,nx
        do 200 j=2,ny
            dopl(i,j) = dopl(i,j) -dt*(fopl(i,j)-fopl(i-1,j))/dx
&
            & -dt*(gopl(i,j)-gopl(i,j-1))/dy
&
            & -dt*dopl(i,j)*(dh2(i,j)*c1 + dn2(i,j)*c5 + dnh3(i,j)*c7)
200    continue
        return
    end
c
    subroutine oplusbi
c
    parameter (nx=90,ny=20)
    common/grid/dx,dy,dt
    common/fluxOpl/fopl(nx,ny),gopl(nx,ny)
    common/dens/dh2(nx,ny),dn2(nx,ny),dnh3(nx,ny),
$ dh2o(nx,ny),dco(nx,ny),dco2(nx,ny),
$ dopl(nx,ny),dohpl(nx,ny),dh2opl(nx,ny),dnopl(nx,ny),
$ dnh3pl(nx,ny),dn2pl(nx,ny),de(nx,ny),dh2pl(nx,ny),
$ do2pl(nx,ny),dcopl(nx,ny),dco2pl(nx,ny),dh3opl(nx,ny),
$ dph1(nx,ny),dph2(nx,ny),dph3(nx,ny),dph4(nx,ny),
$ dph5(nx,ny),dph6(nx,ny),dph7(nx,ny)
    common/veloion/vxoxy(nx,ny),vyoxy(nx,ny)
    common/rxrate/c1,c2,c3,c4,c5,c6,c7,c8,c9,c10
    common/rxratebi/c11,c12,c13,c14,c15,c16,c17,c18,c19
    common/scrat_a/sc_ax(nx,ny),sc_ay(nx,ny)
    common/scrat_b/sc_bx(nx,ny),sc_by(nx,ny)

c-----    initialize scratch arrays
c
    do 100 i=1,nx
        do 100 j=1,ny
            sc_ax(i,j) = 0.0
            sc_ay(i,j) = 0.0
            sc_bx(i,j) = 0.0
            sc_by(i,j) = 0.0
100    continue
c
    nx1=nx-1
    ny1=ny-1
c
    do 130 i=1,nx1
        do 130 j=1,ny
            sc_ax(i,j) = 0.5 * (vxoxy(i+1,j) + vxoxy(i,j))
130    continue
        do 140 i=1,nx

```

```

        do 140 j=1,ny1
            sc_ay(i,j) = 0.5 * (vyoxy(i,j+1) + vyoxy(i,j))
140    continue
c
        do 150 i=1,nx1
            do 150 j=1,ny
                if(sc_ax(i,j).ge.0.)then
                    sc_bx(i,j)=dopl(i,j)
                else
                    sc_bx(i,j)=dopl(i+1,j)
                endif
150    continue
c
        do 160 i=1,nx
            do 160 j=1,ny1
                if(sc_ay(i,j).ge.0.)then
                    sc_by(i,j) = dopl(i,j)
                else
                    sc_by(i,j) = dopl(i,j+1)
                endif
160    continue
c
c
        do 170 i=1,nx1
            do 170 j=1,ny
                fopl(i,j) = sc_ax(i,j) * sc_bx(i,j)
170    continue
c
        do 180 i=1,nx
            do 180 j=1,ny1
                gopl(i,j) = sc_ay(i,j) * sc_by(i,j)
180    continue
c
        do 200 i=2,nx
            do 200 j=2,ny
                dopl(i,j) = dopl(i,j) -dt*(fopl(i,j)-fopl(i-1,j))/dx
&                    -dt*(gopl(i,j)-gopl(i,j-1))/dy
& -dt*dopl(i,j)*(dh2(i,j)*c1 + dn2(i,j)*c5 + dh2o(i,j)*c11
& + dco2(i,j)*c12 + dco2(i,j)*c14 + dco(i,j)*c17)
200    continue
        return
        end
c

```

subroutine ohplus

```

c
parameter (nx=90,ny=20)
common/grid/dx,dy,dt
common/fluxOHpl/fohpl(nx,ny),gohpl(nx,ny)
common/dens/dh2(nx,ny),dn2(nx,ny),dnh3(nx,ny),
$ dh2o(nx,ny),dco(nx,ny),dco2(nx,ny),
$ dopl(nx,ny),dohpl(nx,ny),dh2opl(nx,ny),dnopl(nx,ny),
$ dnh3pl(nx,ny),dn2pl(nx,ny),de(nx,ny),dh2pl(nx,ny),
$ do2pl(nx,ny),dcopl(nx,ny),dco2pl(nx,ny),dh3opl(nx,ny),
$ dph1(nx,ny),dph2(nx,ny),dph3(nx,ny),dph4(nx,ny),
$ dph5(nx,ny),dph6(nx,ny),dph7(nx,ny)
common/veloneu/vxn(nx,ny),vyn(nx,ny)
common/rxrate/c1,c2,c3,c4,c5,c6,c7,c8,c9,c10
common/rxratebi/c11,c12,c13,c14,c15,c16,c17,c18,c19

```



```

common/scrat_a/sc_ax(nx,ny),sc_ay(nx,ny)
common/scrat_b/sc_bx(nx,ny),sc_by(nx,ny)

c----- initialize scratch arrays
c
do 100 i=1,nx
do 100 j=1,ny
sc_ax(i,j) = 0.0
sc_ay(i,j) = 0.0
sc_bx(i,j) = 0.0
sc_by(i,j) = 0.0
100 continue
c
nx1=nx-1
ny1=ny-1
c
do 130 i=1,nx1
do 130 j=1,ny
sc_ax(i,j) = 0.5 * (vxn(i+1,j) + vxn(i,j))
130 continue
do 140 i=1,nx
do 140 j=1,ny1
sc_ay(i,j) = 0.5 * (vyn(i,j+1) + vyn(i,j))
140 continue
c
do 150 i=1,nx1
do 150 j=1,ny
if(sc_ax(i,j).ge.0.)then
sc_bx(i,j)=dohpl(i,j)
else
sc_bx(i,j)=dohpl(i+1,j)
endif
150 continue
c
do 160 i=1,nx
do 160 j=1,ny1
if(sc_ay(i,j).ge.0.)then
sc_by(i,j) = dohpl(i,j)
else
sc_by(i,j) = dohpl(i,j+1)
endif
160 continue
c
c
do 170 i=1,nx1
do 170 j=1,ny
fohpl(i,j) = sc_ax(i,j) * sc_bx(i,j)
170 continue
c
do 180 i=1,nx
do 180 j=1,ny1
gohpl(i,j) = sc_ay(i,j) * sc_by(i,j)
180 continue
c
do 200 i=2,nx
do 200 j=2,ny
dohpl(i,j) = dohpl(i,j) -dt*(fohpl(i,j)-fohpl(i-1,j))/dx
& -dt*(gohpl(i,j)-gohpl(i,j-1))/dy
& +dt*(dopl(i,j)*c1*dh2(i,j) - dohpl(i,j)*c2*de(i,j)
& - dohpl(i,j)*c3*dh2(i,j))

```

```

200  continue
    return
    end
c

subroutine h2oplus

parameter (nx=90,ny=20)
common/grid/dx,dy,dt
common/fluxH2opl/fh2opl(nx,ny),gh2opl(nx,ny)
common/dens/dh2(nx,ny),dn2(nx,ny),dnh3(nx,ny),
$ dh2o(nx,ny),dco(nx,ny),dco2(nx,ny),
$ dopl(nx,ny),dohpl(nx,ny),dh2opl(nx,ny),dnopl(nx,ny),
$ dnh3pl(nx,ny),dn2pl(nx,ny),de(nx,ny),dh2pl(nx,ny),
$ do2pl(nx,ny),dcopl(nx,ny),dco2pl(nx,ny),dh3opl(nx,ny),
$ dph1(nx,ny),dph2(nx,ny),dph3(nx,ny),dph4(nx,ny),
$ dph5(nx,ny),dph6(nx,ny),dph7(nx,ny)
common/veloneu/vxn(nx,ny),vyn(nx,ny)
common/veloion/vxoxy(nx,ny),vyoxy(nx,ny)
common/rxrate/c1,c2,c3,c4,c5,c6,c7,c8,c9,c10
common/scrat_a/sc_ax(nx,ny),sc_ay(nx,ny)
common/scrat_b/sc_bx(nx,ny),sc_by(nx,ny)

c----- initialize scratch arrays
c
do 100 i=1,nx
do 100 j=1,ny
    sc_ax(i,j) = 0.0
    sc_ay(i,j) = 0.0
    sc_bx(i,j) = 0.0
    sc_by(i,j) = 0.0
100 continue
c
nx1=nx-1
ny1=ny-1
c
do 130 i=1,nx1
do 130 j=1,ny
    sc_ax(i,j) = 0.5 * (vxn(i+1,j) + vxn(i,j))
130 continue
do 140 i=1,nx
do 140 j=1,ny1
    sc_ay(i,j) = 0.5 * (vyn(i,j+1) + vyn(i,j))
140 continue
c
do 150 i=1,nx1
do 150 j=1,ny
    if(sc_ax(i,j).ge.0.)then
        sc_bx(i,j)=dh2opl(i,j)
    else
        sc_bx(i,j)=dh2opl(i+1,j)
    endif
150 continue
c
do 160 i=1,nx
do 160 j=1,ny1
    if(sc_ay(i,j).ge.0.)then

```

```

        sc_by(i,j) = dh2opl(i,j)
    else
        sc_by(i,j) = dh2opl(i,j+1)
    endif
160  continue
c
c
    do 170 i=1,nx1
        do 170 j=1,ny
            fh2opl(i,j) = sc_ax(i,j) * sc_bx(i,j)
170  continue
c
    do 180 i=1,nx
        do 180 j=1,ny1
            gh2opl(i,j) = sc_ay(i,j) * sc_by(i,j)
180  continue
c
    do 200 i=2,nx
        do 200 j=2,ny
            dh2opl(i,j) = dh2opl(i,j) -dt*(fh2opl(i,j)-fh2opl(i-1,j))/dx
&                                -dt*(gh2opl(i,j)-gh2opl(i,j-1))/dy
&                                +dt*(dohpl(i,j)*c3*dh2(i,j) - dh2opl(i,j)*c4*de(i,j))
200  continue
        return
    end

```

c

subroutine h2oplusbi

c

```

parameter (nx=90,ny=20)
common/grid/dx,dy,dt
common/fluxH2opl/fh2opl(nx,ny),gh2opl(nx,ny)
common/dens/dh2(nx,ny),dn2(nx,ny),dnh3(nx,ny),
$ dh2o(nx,ny),dco(nx,ny),dco2(nx,ny),
$ dopl(nx,ny),dohpl(nx,ny),dh2opl(nx,ny),dnopl(nx,ny),
$ dnh3pl(nx,ny),dn2pl(nx,ny),de(nx,ny),dh2pl(nx,ny),
$ do2pl(nx,ny),dcopl(nx,ny),dco2pl(nx,ny),dh3opl(nx,ny),
$ dph1(nx,ny),dph2(nx,ny),dph3(nx,ny),dph4(nx,ny),
$ dph5(nx,ny),dph6(nx,ny),dph7(nx,ny)
common/veloneu/vxn(nx,ny),vyn(nx,ny)
common/rxrate/c1,c2,c3,c4,c5,c6,c7,c8,c9,c10
common/rxratebi/c11,c12,c13,c14,c15,c16,c17,c18,c19
common/scrat_a/sc_ax(nx,ny),sc_ay(nx,ny)
common/scrat_b/sc_bx(nx,ny),sc_by(nx,ny)

```

c----- initialize scratch arrays

c

```

    do 100 i=1,nx
        do 100 j=1,ny
            sc_ax(i,j) = 0.0
            sc_ay(i,j) = 0.0
            sc_bx(i,j) = 0.0
            sc_by(i,j) = 0.0
100  continue

```

c

```

    nx1=nx-1
    ny1=ny-1

```

c

```

    do 130 i=1,nx1

```

```

        do 130 j=1,ny
            sc_ax(i,j) = 0.5 * (vxn(i+1,j) + vxn(i,j))
130    continue
        do 140 i=1,nx
            do 140 j=1,ny1
                sc_ay(i,j) = 0.5 * (vyn(i,j+1) + vyn(i,j))
140    continue
c
        do 150 i=1,nx1
            do 150 j=1,ny
                if(sc_ax(i,j).ge.0.)then
                    sc_bx(i,j)=dh2opl(i,j)
                else
                    sc_bx(i,j)=dh2opl(i+1,j)
                endif
150    continue
c
        do 160 i=1,nx
            do 160 j=1,ny1
                if(sc_ay(i,j).ge.0.)then
                    sc_by(i,j) = dh2opl(i,j)
                else
                    sc_by(i,j) = dh2opl(i,j+1)
                endif
160    continue
c
c
        do 170 i=1,nx1
            do 170 j=1,ny
                fh2opl(i,j) = sc_ax(i,j) * sc_bx(i,j)
170    continue
c
        do 180 i=1,nx
            do 180 j=1,ny1
                gh2opl(i,j) = sc_ay(i,j) * sc_by(i,j)
180    continue
c
        do 200 i=2,nx
            do 200 j=2,ny
                dh2opl(i,j) = dh2opl(i,j) -dt*(fh2opl(i,j)-fh2opl(i-1,j))/dx
&                                -dt*(gh2opl(i,j)-gh2opl(i,j-1))/dy
&                                +dt*(dohpl(i,j)*c3*dh2(i,j) - dh2opl(i,j)*c4*de(i,j)
&                                + dh2o(i,j)*c11*dopl(i,j) - dh2opl(i,j)*c18*dh2o(i,j))
200    continue
        return
        end
c

```

subroutine noplus

```

c
parameter (nx=90,ny=20)
common/grid/dx,dy,dt
common/fluxN0pl/fnopl(nx,ny),gnopl(nx,ny)
common/dens/dh2(nx,ny),dn2(nx,ny),dnh3(nx,ny),
$ dh2o(nx,ny),dco(nx,ny),dco2(nx,ny),
$ dopl(nx,ny),dohpl(nx,ny),dh2opl(nx,ny),dnopl(nx,ny),
$ dnh3pl(nx,ny),dn2pl(nx,ny),de(nx,ny),dh2pl(nx,ny),
$ do2pl(nx,ny),dco1pl(nx,ny),dco2pl(nx,ny),dh3opl(nx,ny),
$ dph1(nx,ny),dph2(nx,ny),dph3(nx,ny),dph4(nx,ny),

```

```

$ dph5(nx,ny),dph6(nx,ny),dph7(nx,ny)
common/veloneu/vxn(nx,ny),vyn(nx,ny)
common/rxrate/c1,c2,c3,c4,c5,c6,c7,c8,c9,c10
common/rxratebi/c11,c12,c13,c14,c15,c16,c17,c18,c19
common/scrat_a/sc_ax(nx,ny),sc_ay(nx,ny)
common/scrat_b/sc_bx(nx,ny),sc_by(nx,ny)

```

```

c----- initialize scratch arrays

```

```

c

```

```

do 100 i=1,nx
do 100 j=1,ny
sc_ax(i,j) = 0.0
sc_ay(i,j) = 0.0
sc_bx(i,j) = 0.0
sc_by(i,j) = 0.0

```

```

100 continue

```

```

c

```

```

nx1=nx-1
ny1=ny-1

```

```

c

```

```

do 130 i=1,nx1
do 130 j=1,ny
sc_ax(i,j) = 0.5 * (vxn(i+1,j) + vxn(i,j))

```

```

130 continue

```

```

do 140 i=1,nx
do 140 j=1,ny1
sc_ay(i,j) = 0.5 * (vyn(i,j+1) + vyn(i,j))

```

```

140 continue

```

```

c

```

```

do 150 i=1,nx1
do 150 j=1,ny
if(sc_ax(i,j).ge.0.)then
sc_bx(i,j)=dnopl(i,j)
else
sc_bx(i,j)=dnopl(i+1,j)
endif

```

```

150 continue

```

```

c

```

```

do 160 i=1,nx
do 160 j=1,ny1
if(sc_ay(i,j).ge.0.)then
sc_by(i,j) = dnopl(i,j)
else
sc_by(i,j) = dnopl(i,j+1)
endif

```

```

160 continue

```

```

c

```

```

c

```

```

do 170 i=1,nx1
do 170 j=1,ny
fnopl(i,j) = sc_ax(i,j) * sc_bx(i,j)

```

```

170 continue

```

```

c

```

```

do 180 i=1,nx
do 180 j=1,ny1
gnopl(i,j) = sc_ay(i,j) * sc_by(i,j)

```

```

180 continue

```

```

c

```

```

do 200 i=2,nx
do 200 j=2,ny

```

```

        dnopl(i,j) = dnopl(i,j) -dt*(fnopl(i,j)-fnopl(i-1,j))/dx
&        -dt*(gnopl(i,j)-gnopl(i,j-1))/dy
&        +dt*(dopl(i,j)*c5*dn2(i,j) - dnopl(i,j)*c6*de(i,j))
200    continue
    return
end

c
    subroutine nh3plus
c
    parameter (nx=90,ny=20)
    common/grid/dx,dy,dt
    common/fluxNH3pl/fnh3pl(nx,ny),gnh3pl(nx,ny)
    common/dens/dh2(nx,ny),dn2(nx,ny),dnh3(nx,ny),
$ dh2o(nx,ny),dco(nx,ny),dco2(nx,ny),
$ dopl(nx,ny),dohpl(nx,ny),dh2opl(nx,ny),dnopl(nx,ny),
$ dnh3pl(nx,ny),dn2pl(nx,ny),de(nx,ny),dh2pl(nx,ny),
$ do2pl(nx,ny),dcopl(nx,ny),dco2pl(nx,ny),dh3opl(nx,ny),
$ dph1(nx,ny),dph2(nx,ny),dph3(nx,ny),dph4(nx,ny),
$ dph5(nx,ny),dph6(nx,ny),dph7(nx,ny)
    common/veloneu/vxn(nx,ny),vyn(nx,ny)
    common/veloph/vxph(nx,ny),vyph(nx,ny)
    common/rxrate/c1,c2,c3,c4,c5,c6,c7,c8,c9,c10
    common/phrateH2/cph3h2,cph4h2,cph5h2
    common/phrateN2/cph3n2,cph4n2,cph5n2,cph6n2,cph7n2
    common/phrateNH3/cph1nh3,cph2nh3,cph3nh3,cph4nh3
    common/scrat_a/sc_ax(nx,ny),sc_ay(nx,ny)
    common/scrat_b/sc_bx(nx,ny),sc_by(nx,ny)

c-----    initialize scratch arrays
c
    do 100 i=1,nx
    do 100 j=1,ny
        sc_ax(i,j) = 0.0
        sc_ay(i,j) = 0.0
        sc_bx(i,j) = 0.0
        sc_by(i,j) = 0.0
100    continue
c
    nx1=nx-1
    ny1=ny-1
c
    do 130 i=1,nx1
    do 130 j=1,ny
        sc_ax(i,j) = 0.5 * (vxn(i+1,j) + vxn(i,j))
130    continue
    do 140 i=1,nx
    do 140 j=1,ny1
        sc_ay(i,j) = 0.5 * (vyn(i,j+1) + vyn(i,j))
140    continue
c
    do 150 i=1,nx1
    do 150 j=1,ny
        if(sc_ax(i,j).ge.0.)then
            sc_bx(i,j)=dnh3pl(i,j)
        else
            sc_bx(i,j)=dnh3pl(i+1,j)
        endif
150    continue
c
    do 160 i=1,nx

```

```

do 160 j=1,ny1
  if(sc_ay(i,j).ge.0.)then
    sc_by(i,j) = dnh3pl(i,j)
  else
    sc_by(i,j) = dnh3pl(i,j+1)
  endif
160 continue
c
c
do 170 i=1,nx1
  do 170 j=1,ny
    fnh3pl(i,j) = sc_ax(i,j) * sc_bx(i,j)
170 continue
c
do 180 i=1,nx
  do 180 j=1,ny1
    gnh3pl(i,j) = sc_ay(i,j) * sc_by(i,j)
180 continue
c
do 200 i=2,nx
  do 200 j=2,ny

    dnh3pl(i,j) = dnh3pl(i,j) -dt*(fnh3pl(i,j)-fnh3pl(i-1,j))/dx
&      -dt*(gnh3pl(i,j)-gnh3pl(i,j-1))/dy
&      +dt*(dopl(i,j)*c7*dnh3(i,j) - dnh3pl(i,j)*c8*de(i,j))
&      +dt*dnh3(i,j)*(cph1nh3*dph1(i,j) + cph2nh3*dph2(i,j)
&      +cph3nh3*dph3(i,j) + cph4nh3*dph4(i,j)*(80./170.))
200 continue
  return
end
c
c
subroutine n2plus
c
parameter (nx=90,ny=20)
common/grid/dx,dy,dt
common/fluxN2pl/fn2pl(nx,ny),gn2pl(nx,ny)
common/dens/dh2(nx,ny),dn2(nx,ny),dnh3(nx,ny),
$ dh2o(nx,ny),dco(nx,ny),dco2(nx,ny),
$ dopl(nx,ny),dohpl(nx,ny),dh2opl(nx,ny),dnopl(nx,ny),
$ dnh3pl(nx,ny),dn2pl(nx,ny),de(nx,ny),dh2pl(nx,ny),
$ do2pl(nx,ny),dcopl(nx,ny),dco2pl(nx,ny),dh3opl(nx,ny),
$ dph1(nx,ny),dph2(nx,ny),dph3(nx,ny),dph4(nx,ny),
$ dph5(nx,ny),dph6(nx,ny),dph7(nx,ny)
common/veloneu/vxn(nx,ny),vyn(nx,ny)
common/veloph/vxph(nx,ny),vyph(nx,ny)
common/veloion/vxoxy(nx,ny),vyoxy(nx,ny)
common/rxrate/c1,c2,c3,c4,c5,c6,c7,c8,c9,c10
common/rxratebi/c11,c12,c13,c14,c15,c16,c17,c18,c19
common/phrateN2/cph3n2,cph4n2,cph5n2,cph6n2,cph7n2
common/scrat_a/sc_ax(nx,ny),sc_ay(nx,ny)
common/scrat_b/sc_bx(nx,ny),sc_by(nx,ny)
common/civn2lim/civn21,civn22,civn23
common/civ/dciv(nx,ny),vcivn2(nx,ny),vcivco2(nx,ny)
common/omega/omegaN2,omegaC02
common/elrest1/elfluence,in2restrict,ico2restrict
common/elrest2/dn2plcheck(nx,ny),dco2plcheck(nx,ny)

c----- initialize scratch arrays
c

```

```

do 100 i=1,nx
do 100 j=1,ny
  sc_ax(i,j) = 0.0
  sc_ay(i,j) = 0.0
  sc_bx(i,j) = 0.0
  sc_by(i,j) = 0.0
100 continue
c
  nx1=nx-1
  ny1=ny-1
c
  do 130 i=1,nx1
  do 130 j=1,ny
    sc_ax(i,j) = 0.5 * (vx_n(i+1,j) + vx_n(i,j))
130 continue
  do 140 i=1,nx
  do 140 j=1,ny1
    sc_ay(i,j) = 0.5 * (vy_n(i,j+1) + vy_n(i,j))
140 continue
c
  do 150 i=1,nx1
  do 150 j=1,ny
    if(sc_ax(i,j).ge.0.)then
      sc_bx(i,j)=dn2pl(i,j)
    else
      sc_bx(i,j)=dn2pl(i+1,j)
    endif
150 continue
c
  do 160 i=1,nx
  do 160 j=1,ny1
    if(sc_ay(i,j).ge.0.)then
      sc_by(i,j) = dn2pl(i,j)
    else
      sc_by(i,j) = dn2pl(i,j+1)
    endif
160 continue
c
c
  do 170 i=1,nx1
  do 170 j=1,ny
    fn2pl(i,j) = sc_ax(i,j) * sc_bx(i,j)
170 continue
c
  do 180 i=1,nx
  do 180 j=1,ny1
    gn2pl(i,j) = sc_ay(i,j) * sc_by(i,j)
180 continue
c
  do 200 i=2,nx
  do 200 j=2,ny
    if (abs(vx_n(i,j)-vx_oxy(i,j)) .ge. 10.2e5) then

    if (dciv(i,j).ge.civn21 .and. dciv(i,j).le.civn22) then
      c10=(5.e-7*dciv(i,j) - 0.17)*(vcivn2(i,j)/1.5)**5
      c10=c10*omegaN2
      dn2pl(i,j) = dn2pl(i,j) -dt*(fn2pl(i,j)-fn2pl(i-1,j))/dx
      & -dt*(gn2pl(i,j)-gn2pl(i,j-1))/dy
      & +dt*dn2(i,j)*(cph3n2*dph3(i,j) + cph4n2*dph4(i,j)
      & + cph5n2*dph5(i,j) + cph6n2*dph6(i,j) + cph7n2*dph7(i,j))

```



```

&      +de(i,j)*(exp(c10*dt)-1.)
deltan2pl = dn2pl(i,j) - dn2plcheck(i,j)
if (deltan2pl.gt.elfluence) then
  dn2pl(i,j) = dn2plcheck(i,j) + elfluence
  in2restrict = in2restrict + 1
endif
dn2plcheck(i,j)=dn2pl(i,j)
endif

if (dciv(i,j).gt.civn22 .and. dciv(i,j).le.civn23) then
c10=(7.8 - 1.2e-7*dciv(i,j))*(vcivn2(i,j)/1.5)**5
c10=c10*omegaN2
dn2pl(i,j) = dn2pl(i,j) -dt*(fn2pl(i,j)-fn2pl(i-1,j))/dx
&      -dt*(gn2pl(i,j)-gn2pl(i,j-1))/dy
&      +dt*dn2(i,j)*(cph3n2*dph3(i,j) + cph4n2*dph4(i,j)
&      + cph5n2*dph5(i,j) + cph6n2*dph6(i,j) + cph7n2*dph7(i,j))
&      +de(i,j)*(exp(c10*dt)-1.)
deltan2pl = dn2pl(i,j) - dn2plcheck(i,j)
if (deltan2pl.gt.elfluence) then
  dn2pl(i,j) = dn2plcheck(i,j) + elfluence
  in2restrict = in2restrict + 1
endif
dn2plcheck(i,j)=dn2pl(i,j)
endif

else

  dn2pl(i,j) = dn2pl(i,j) -dt*(fn2pl(i,j)-fn2pl(i-1,j))/dx
&      -dt*(gn2pl(i,j)-gn2pl(i,j-1))/dy
&      +dt*dn2(i,j)*(cph3n2*dph3(i,j) + cph4n2*dph4(i,j)
&      + cph5n2*dph5(i,j) + cph6n2*dph6(i,j) + cph7n2*dph7(i,j))
endif
200  continue
return
end

```

c

subroutine h2plus

c

```

parameter (nx=90,ny=20)
common/grid/dx,dy,dt
common/fluxH2pl/fh2pl(nx,ny),gh2pl(nx,ny)
common/dens/dh2(nx,ny),dn2(nx,ny),dnh3(nx,ny),
$ dh2o(nx,ny),dco(nx,ny),dco2(nx,ny),
$ dopl(nx,ny),dohpl(nx,ny),dh2opl(nx,ny),dnopl(nx,ny),
$ dnh3pl(nx,ny),dn2pl(nx,ny),de(nx,ny),dh2pl(nx,ny),
$ do2pl(nx,ny),dcopl(nx,ny),dco2pl(nx,ny),dh3opl(nx,ny),
$ dph1(nx,ny),dph2(nx,ny),dph3(nx,ny),dph4(nx,ny),
$ dph5(nx,ny),dph6(nx,ny),dph7(nx,ny)
common/veloneu/vxn(nx,ny),vyn(nx,ny)
common/veloph/vxph(nx,ny),vyph(nx,ny)
common/veloion/vxoxy(nx,ny),vyoxy(nx,ny)
common/rxrate/c1,c2,c3,c4,c5,c6,c7,c8,c9,c10
common/phrateH2/cph3h2,cph4h2,cph5h2
common/scrat_a/sc_ax(nx,ny),sc_ay(nx,ny)
common/scrat_b/sc_bx(nx,ny),sc_by(nx,ny)

```

c----- initialize scratch arrays

c

```

do 100 i=1,nx
do 100 j=1,ny
  sc_ax(i,j) = 0.0
  sc_ay(i,j) = 0.0
  sc_bx(i,j) = 0.0
  sc_by(i,j) = 0.0
100 continue
c
  nx1=nx-1
  ny1=ny-1
c
  do 130 i=1,nx1
  do 130 j=1,ny
    sc_ax(i,j) = 0.5 * (vxn(i+1,j) + vxn(i,j))
130 continue
  do 140 i=1,nx
  do 140 j=1,ny1
    sc_ay(i,j) = 0.5 * (vyn(i,j+1) + vyn(i,j))
140 continue
c
  do 150 i=1,nx1
  do 150 j=1,ny
    if(sc_ax(i,j).ge.0.)then
      sc_bx(i,j)=dh2pl(i,j)
    else
      sc_bx(i,j)=dh2pl(i+1,j)
    endif
150 continue
c
  do 160 i=1,nx
  do 160 j=1,ny1
    if(sc_ay(i,j).ge.0.)then
      sc_by(i,j) = dh2pl(i,j)
    else
      sc_by(i,j) = dh2pl(i,j+1)
    endif
160 continue
c
c
  do 170 i=1,nx1
  do 170 j=1,ny
    fh2pl(i,j) = sc_ax(i,j) * sc_bx(i,j)
170 continue
c
  do 180 i=1,nx
  do 180 j=1,ny1
    gh2pl(i,j) = sc_ay(i,j) * sc_by(i,j)
180 continue
c
  do 200 i=2,nx
  do 200 j=2,ny

    dh2pl(i,j) = dh2pl(i,j) -dt*(fh2pl(i,j)-fh2pl(i-1,j))/dx
&      -dt*(gh2pl(i,j)-gh2pl(i,j-1))/dy
&      +dt*dh2(i,j)*(cph3h2*dph3(i,j) + cph4h2*dph4(i,j)
&      + cph5h2*dph5(i,j))

200 continue
  return
end

```

```

c      subroutine co2plusbi
c
      parameter (nx=90,ny=20)
      common/grid/dx,dy,dt
      common/fluxC02pl/fco2pl(nx,ny),gco2pl(nx,ny)
      common/dens/dh2(nx,ny),dn2(nx,ny),dnh3(nx,ny),
$ dh2o(nx,ny),dco(nx,ny),dco2(nx,ny),
$ dopl(nx,ny),dohpl(nx,ny),dh2opl(nx,ny),dnopl(nx,ny),
$ dnh3pl(nx,ny),dn2pl(nx,ny),de(nx,ny),dh2pl(nx,ny),
$ do2pl(nx,ny),dcopl(nx,ny),dco2pl(nx,ny),dh3opl(nx,ny),
$ dph1(nx,ny),dph2(nx,ny),dph3(nx,ny),dph4(nx,ny),
$ dph5(nx,ny),dph6(nx,ny),dph7(nx,ny)
      common/veloneu/vxn(nx,ny),vyn(nx,ny)
      common/veloion/vxoxy(nx,ny),vyoxy(nx,ny)
      common/rxrate/c1,c2,c3,c4,c5,c6,c7,c8,c9,c10
      common/rxratebi/c11,c12,c13,c14,c15,c16,c17,c18,c19
      common/phrateC02/cph2co2,cph3co2,cph4co2,cph5co2
      common/scrat_a/sc_ax(nx,ny),sc_ay(nx,ny)
      common/scrat_b/sc_bx(nx,ny),sc_by(nx,ny)
      common/civco2lim/civco21,civco22,civco23
      common/civ/dciv(nx,ny),vcivn2(nx,ny),vcivco2(nx,ny)
      common/omega/omegaN2,omegaC02
      common/elrest1/elfluence,in2restrict,ico2restrict
      common/elrest2/dn2plcheck(nx,ny),dco2plcheck(nx,ny)

```

```

c----- initialize scratch arrays

```

```

c
      do 100 i=1,nx
      do 100 j=1,ny
        sc_ax(i,j) = 0.0
        sc_ay(i,j) = 0.0
        sc_bx(i,j) = 0.0
        sc_by(i,j) = 0.0
100    continue
c
      nx1=nx-1
      ny1=ny-1
c
      do 130 i=1,nx1
      do 130 j=1,ny
        sc_ax(i,j) = 0.5 * (vxn(i+1,j) + vxn(i,j))
130    continue
      do 140 i=1,nx
      do 140 j=1,ny1
        sc_ay(i,j) = 0.5 * (vyn(i,j+1) + vyn(i,j))
140    continue
c
      do 150 i=1,nx1
      do 150 j=1,ny
        if(sc_ax(i,j).ge.0.)then
          sc_bx(i,j)=dco2pl(i,j)
        else
          sc_bx(i,j)=dco2pl(i+1,j)
        endif
150    continue
c
      do 160 i=1,nx
      do 160 j=1,ny1

```

```

        if(sc_ay(i,j).ge.0.)then
            sc_by(i,j) = dco2pl(i,j)
        else
            sc_by(i,j) = dco2pl(i,j+1)
        endif
160    continue
c
c
        do 170 i=1,nx1
            do 170 j=1,ny
                fco2pl(i,j) = sc_ax(i,j) * sc_bx(i,j)
170    continue
c
        do 180 i=1,nx
            do 180 j=1,ny1
                gco2pl(i,j) = sc_ay(i,j) * sc_by(i,j)
180    continue
c
        do 200 i=2,nx
            do 200 j=2,ny

                if (abs(vxn(i,j)-vxoxy(i,j)) .ge. 7.7e5) then

                    if (dciv(i,j).ge.civco21 .and. dciv(i,j).le.civco22) then
                        c13=(0.25 + 4.5e-7*dciv(i,j))*(vcivco2(i,j)/1.5)**5
                        c13=c13*omegaC02
                        dco2pl(i,j) = dco2pl(i,j) -dt*(fco2pl(i,j)-fco2pl(i-1,j))/dx
&                                     -dt*(gco2pl(i,j)-gco2pl(i,j-1))/dy
&      +dt*(dopl(i,j)*c14*dco2(i,j) - dco2pl(i,j)*c15*de(i,j))
&      +de(i,j)*(exp(c13*dt)-1.)
                        deltaco2pl = dco2pl(i,j) - dco2plcheck(i,j)
                        if (deltaco2pl.gt.elfluence) then
                            dco2pl(i,j) = dco2plcheck(i,j) + elfluence
                            ico2restrict = ico2restrict + 1
                        endif
                        dco2plcheck(i,j)=dco2pl(i,j)
                        endif

                    if (dciv(i,j).gt.civco22 .and. dciv(i,j).le.civco23) then
                        c13=(17. - 2.23e-7*dciv(i,j))*(vcivco2(i,j)/1.5)**5
                        c13=c13*omegaC02
                        dco2pl(i,j) = dco2pl(i,j) -dt*(fco2pl(i,j)-fco2pl(i-1,j))/dx
&                                     -dt*(gco2pl(i,j)-gco2pl(i,j-1))/dy
&      +dt*(dopl(i,j)*c14*dco2(i,j) - dco2pl(i,j)*c15*de(i,j))
&      +de(i,j)*(exp(c13*dt)-1.)
c      Checking ambient electron flux available to neutralize plasma
                        deltaco2pl = dco2pl(i,j) - dco2plcheck(i,j)
                        if (deltaco2pl.gt.elfluence) then
                            dco2pl(i,j) = dco2plcheck(i,j) + elfluence
                            ico2restrict = ico2restrict + 1
                        endif
                        dco2plcheck(i,j)=dco2pl(i,j)
                        endif

                    else

                        dco2pl(i,j) = dco2pl(i,j) -dt*(fco2pl(i,j)-fco2pl(i-1,j))/dx
&                                     -dt*(gco2pl(i,j)-gco2pl(i,j-1))/dy
&      +dt*(dopl(i,j)*c14*dco2(i,j) - dco2pl(i,j)*c15*de(i,j))
&      endif

```

```

200    continue
      return
      end

c
c
c      subroutine o2plusbi
c
c      parameter (nx=90,ny=20)
c      common/grid/dx,dy,dt
c      common/flux02pl/fo2pl(nx,ny),go2pl(nx,ny)
c      common/dens/dh2(nx,ny),dn2(nx,ny),dnh3(nx,ny),
$ dh2o(nx,ny),dco(nx,ny),dco2(nx,ny),
$ dopl(nx,ny),dohpl(nx,ny),dh2opl(nx,ny),dnopl(nx,ny),
$ dnh3pl(nx,ny),dn2pl(nx,ny),de(nx,ny),dh2pl(nx,ny),
$ do2pl(nx,ny),dcopl(nx,ny),dco2pl(nx,ny),dh3opl(nx,ny),
$ dph1(nx,ny),dph2(nx,ny),dph3(nx,ny),dph4(nx,ny),
$ dph5(nx,ny),dph6(nx,ny),dph7(nx,ny)
c      common/veloneu/vxn(nx,ny),vyn(nx,ny)
c      common/rxrate/c1,c2,c3,c4,c5,c6,c7,c8,c9,c10
c      common/rxratebi/c11,c12,c13,c14,c15,c16,c17,c18,c19
c      common/scrat_a/sc_ax(nx,ny),sc_ay(nx,ny)
c      common/scrat_b/sc_bx(nx,ny),sc_by(nx,ny)

c-----    initialize scratch arrays
c
c      do 100 i=1,nx
c      do 100 j=1,ny
c          sc_ax(i,j) = 0.0
c          sc_ay(i,j) = 0.0
c          sc_bx(i,j) = 0.0
c          sc_by(i,j) = 0.0
100    continue
c
c      nx1=nx-1
c      ny1=ny-1
c
c      do 130 i=1,nx1
c      do 130 j=1,ny
c          sc_ax(i,j) = 0.5 * (vxn(i+1,j) + vxn(i,j))
130    continue
c      do 140 i=1,nx
c      do 140 j=1,ny1
c          sc_ay(i,j) = 0.5 * (vyn(i,j+1) + vyn(i,j))
140    continue
c
c      do 150 i=1,nx1
c      do 150 j=1,ny
c          if(sc_ax(i,j).ge.0.)then
c              sc_bx(i,j)=do2pl(i,j)
c          else
c              sc_bx(i,j)=do2pl(i+1,j)
c          endif
150    continue
c
c      do 160 i=1,nx
c      do 160 j=1,ny1
c          if(sc_ay(i,j).ge.0.)then
c              sc_by(i,j) = do2pl(i,j)

```

```

        else
            sc_by(i,j) = do2pl(i,j+1)
        endif
160    continue
c
c
        do 170 i=1,nx1
            do 170 j=1,ny
                fo2pl(i,j) = sc_ax(i,j) * sc_bx(i,j)
170    continue
c
        do 180 i=1,nx
            do 180 j=1,ny1
                go2pl(i,j) = sc_ay(i,j) * sc_by(i,j)
180    continue
c
        do 200 i=2,nx
            do 200 j=2,ny

                do2pl(i,j) = do2pl(i,j) -dt*(fo2pl(i,j)-fo2pl(i-1,j))/dx
+                               -dt*(go2pl(i,j)-go2pl(i,j-1))/dy
+       +dt*(dopl(i,j)*c12*dco2(i,j))
+       -dt*(do2pl(i,j)*c16*de(i,j))
200    continue
        return
        end
c
        subroutine coplusbi
c
        parameter (nx=90,ny=20)
        common/grid/dx,dy,dt
        common/fluxC0pl/fcopl(nx,ny),gcopl(nx,ny)
        common/dens/dh2(nx,ny),dn2(nx,ny),dnh3(nx,ny),
$ dh2o(nx,ny),dco(nx,ny),dco2(nx,ny),
$ dopl(nx,ny),dohpl(nx,ny),dh2opl(nx,ny),dnopl(nx,ny),
$ dnh3pl(nx,ny),dn2pl(nx,ny),de(nx,ny),dh2pl(nx,ny),
$ do2pl(nx,ny),dcopl(nx,ny),dco2pl(nx,ny),dh3opl(nx,ny),
$ dph1(nx,ny),dph2(nx,ny),dph3(nx,ny),dph4(nx,ny),
$ dph5(nx,ny),dph6(nx,ny),dph7(nx,ny)
        common/veloneu/vxn(nx,ny),vyn(nx,ny)
        common/veloion/vxoxy(nx,ny),vyoxy(nx,ny)
        common/rxrate/c1,c2,c3,c4,c5,c6,c7,c8,c9,c10
        common/rxratebi/c11,c12,c13,c14,c15,c16,c17,c18,c19
        common/phrateC0/cph2co,cph3co,cph4co,cph5co
        common/scrat_a/sc_ax(nx,ny),sc_ay(nx,ny)
        common/scrat_b/sc_bx(nx,ny),sc_by(nx,ny)

c----- initialize scratch arrays
c
        do 100 i=1,nx
            do 100 j=1,ny
                sc_ax(i,j) = 0.0
                sc_ay(i,j) = 0.0
                sc_bx(i,j) = 0.0
                sc_by(i,j) = 0.0
100    continue
c
        nx1=nx-1
        ny1=ny-1
c

```

```

        do 130 i=1,nx1
        do 130 j=1,ny
            sc_ax(i,j) = 0.5 * (vxn(i+1,j) + vxn(i,j))
130    continue
        do 140 i=1,nx
        do 140 j=1,ny1
            sc_ay(i,j) = 0.5 * (vyn(i,j+1) + vyn(i,j))
140    continue
c
        do 150 i=1,nx1
        do 150 j=1,ny
            if(sc_ax(i,j).ge.0.)then
                sc_bx(i,j)=dcopl(i,j)
            else
                sc_bx(i,j)=dcopl(i+1,j)
            endif
150    continue
c
        do 160 i=1,nx
        do 160 j=1,ny1
            if(sc_ay(i,j).ge.0.)then
                sc_by(i,j) = dcopl(i,j)
            else
                sc_by(i,j) = dcopl(i,j+1)
            endif
160    continue
c
c
        do 170 i=1,nx1
        do 170 j=1,ny
            fcopl(i,j) = sc_ax(i,j) * sc_bx(i,j)
170    continue
c
        do 180 i=1,nx
        do 180 j=1,ny1
            gcopl(i,j) = sc_ay(i,j) * sc_by(i,j)
180    continue
c
        do 200 i=2,nx
        do 200 j=2,ny

            dcopl(i,j) = dcopl(i,j) -dt*(fcopl(i,j)-fcopl(i-1,j))/dx
&                -dt*(gcopl(i,j)-gcopl(i,j-1))/dy
&            +dt*(dopl(i,j)*c17*dco(i,j))
&            +dt*dco(i,j)*((cph2co*dph2(i,j)*(85./110.))
&            + cph3co*dph3(i,j)
&            + cph4co*dph4(i,j) + cph5co*dph5(i,j))
200    continue
        return
        end
c
c
        subroutine h3oplusbi
c
        parameter (nx=90,ny=20)
        common/grid/dx,dy,dt
        common/fluxH3opl/fh3opl(nx,ny),gh3opl(nx,ny)
        common/dens/dh2(nx,ny),dn2(nx,ny),dnh3(nx,ny),
$ dh2o(nx,ny),dco(nx,ny),dco2(nx,ny),
$ dopl(nx,ny),dohpl(nx,ny),dh2opl(nx,ny),dnopl(nx,ny),
$ dnh3pl(nx,ny),dn2pl(nx,ny),de(nx,ny),dh2pl(nx,ny),

```

```

$ do2pl(nx,ny),dco1(nx,ny),dco2pl(nx,ny),dh3opl(nx,ny),
$ dph1(nx,ny),dph2(nx,ny),dph3(nx,ny),dph4(nx,ny),
$ dph5(nx,ny),dph6(nx,ny),dph7(nx,ny)
common/veloneu/vxn(nx,ny),vyn(nx,ny)
common/veloion/vxoxy(nx,ny),vyoxy(nx,ny)
common/rxrate/c1,c2,c3,c4,c5,c6,c7,c8,c9,c10
common/rxratebi/c11,c12,c13,c14,c15,c16,c17,c18,c19
common/scrat_a/sc_ax(nx,ny),sc_ay(nx,ny)
common/scrat_b/sc_bx(nx,ny),sc_by(nx,ny)

```

```

c----- initialize scratch arrays

```

```

c
  do 100 i=1,nx
  do 100 j=1,ny
    sc_ax(i,j) = 0.0
    sc_ay(i,j) = 0.0
    sc_bx(i,j) = 0.0
    sc_by(i,j) = 0.0
100  continue
c
  nx1=nx-1
  ny1=ny-1
c
c
  do 130 i=1,nx1
  do 130 j=1,ny
    sc_ax(i,j) = 0.5 * (vxn(i+1,j) + vxn(i,j))
130  continue
  do 140 i=1,nx
  do 140 j=1,ny1
    sc_ay(i,j) = 0.5 * (vyn(i,j+1) + vyn(i,j))
140  continue
c
  do 150 i=1,nx1
  do 150 j=1,ny
    if(sc_ax(i,j).ge.0.)then
      sc_bx(i,j)=dh3opl(i,j)
    else
      sc_bx(i,j)=dh3opl(i+1,j)
    endif
150  continue
c
  do 160 i=1,nx
  do 160 j=1,ny1
    if(sc_ay(i,j).ge.0.)then
      sc_by(i,j) = dh3opl(i,j)
    else
      sc_by(i,j) = dh3opl(i,j+1)
    endif
160  continue
c
c
  do 170 i=1,nx1
  do 170 j=1,ny
    fh3opl(i,j) = sc_ax(i,j) * sc_bx(i,j)
170  continue
c
  do 180 i=1,nx
  do 180 j=1,ny1
    gh3opl(i,j) = sc_ay(i,j) * sc_by(i,j)

```



```

180  continue
c
    do 200 i=2,nx
    do 200 j=2,ny

        dh3opl(i,j) = dh3opl(i,j) -dt*(fh3opl(i,j)-fh3opl(i-1,j))/dx
        &               -dt*(gh3opl(i,j)-gh3opl(i,j-1))/dy
        &   +dt*(dh2o(i,j)*c18*dh2opl(i,j))
        &   -dt*(dh3opl(i,j)*c19*de(i,j))
200  continue
    return
    end

```