Reanalysis of average pressure in the beamtube as a function of water injected at the ends

Rainer Weiss October 30, 2008

Abstract The average water pressure in the beam tubes as a function of water injected at the ends is reanalysed. The initial estimates allowed no more than 400 torr liters of water injected at each end of a 2km beamtube module before compromising the goal average pressure of 10^{-10} torr with only end pumping and approximately a factor of 10 more injected water with distributed pumping. The new estimates do not give a fixed amount of water but rather provide a relation between the exposure time to water pressure and the system's recovery time to the goal pressure. The initial estimates assumed that all the water injected into the tube would adsorb on the walls and did not account for the continued reduction in water outgassing rate with time. The new estimates take into account the adsorption and reemision dynamics. The principal result is that most of the water injected into the tube is collected by the cryo pumps before adsorption. Furthermore, the gas that is adsorbed on the tube walls will be outgassed by the walls with a characteristic 1/t dependence scaled by the exposure time.

The note reviews the initial calculations and presents the results of the new calculations. The calculations are made with a statistical mechanics and surface physics program that was developed to understand the beamtube bakeout in the early 1990's. The program estimates the surface loading of water over the pumping and thermal history of the beamtube and provides an outgassing rate. The outgassing rate is used to estimate the pressure knowing the pumping strategy. The program is based on the Dubinin-Radushkevich surface adsorption theory which assumes a skewed Gaussian distribution of surface sites with adsorption energy. The surface dynamics uses the Langmuir adsorption theory with a molecular surface adsorption potential having both an attractive and repulsive term as hypothesized by Roald Hoffman. The theories and the FORTRAN program used in the calculations are provided in an appendix to this note.

Measurements that could be contemplated to establish the validity of the model and parameters are discussed briefly. Infra-red spectroscopic absorption measurement are considered as well as a method using a residual gas analyser placed at an appropriate position in the beam tube. Both methods would involve some work and preparation and cannot be approached casually.

Introduction During standard operations of opening and closing the vacuum system both in the LVEA and at the ends, we have now introduced somewhat more than 400 torr liters of water into each end of all the 2km modules of the beam tube. The increased installation and commissioning activity in preparing Enhanced LIGO and the anticipated activity in installation and commissioning Advanced LIGO make it urgent to establish if the original estimate is indeed correct and ,if necessary, to devise a strategy to reduce the average water pressure in the beamtubes when the improved interferometer sensitivity requires it.

The goal average pressure The goal pressure specification was set by the constraint that optical phase noise from forward scattering by molecules in the 4 km beam tube path not exceed 1/2 the standard quantum limit for a 1 ton test mass at 100 Hz. This corresponds to a strain noise of $1.5 \times 10^{-25} / \sqrt{\text{Hz}}$ at 100 Hz .

The relation between a uniform outgassing rate and the average pressure in the tube is given by

$$\langle p \rangle_L = J\left[\frac{2\pi aL}{nF} + \frac{L^2}{4va^2(n-1)^2}\right] \qquad \textbf{eq 1}$$

where p, the pressure, is in torr, J, the uniform surface outgassing rate of water in torr cc/cm^2 , a, the tube radius in cm, L, the tube length in cm, n, the number of pumps on the tube, F, the pumping speed of a pump in cc/sec and, v, is the molecular speed in cm/sec. In our beam tubes the pumping port spacing is large enough so that the second term in **eq 1** dominates. For water, simply increasing the pumping speed of the pumps is not an economical or efficient strategy. The pressure distribution in the tube is scalloped with a maximum pressure midway between pumps.

The goal constituent average pressures and outgassing rates associated with two pumping strategies are given in **Table 1**.

Table 1: Goal average pressure and outgassing rates in the beamtube @ 300K

constituent	*average pressure*	*end pump outgassing rate*	*nine pump/module outgassing rate*
	torr	*torr liters/cm^{2}sec*	*torr liters/cm^{2}sec*
amu 18 (H_2O)	1 x 10^{-10}	2 x 10^{-15}	2 x 10^{-14}
amu 100	6 x 10^{-13}	5 x 10^{-18}	4 x 10^{-16}
amu 300	5 x 10^{-14}	2 x 10^{-19}	1 x 10^{-17}
amu 600	8 x 10^{-15}	3 x 10^{-20}	2 x 10^{-18}

The model parameters used in both the original and revised estimates are given in **Table 2**

Table 2: Model parameters for bakeout and operations

param		value	param		value
L_{bake}	module lgth	2 km	L_{op}	lgth of arm	4 km
a	radius of tube	62 cm	T_o	peak DR distribution	1 x 10^4 K
$T_{ambient}$	ambient T	296 K	R	repulsive term surface potential	0.7
n_{bake}	# pumps bake	7	n_{op}	number of pumps in operation	2
F_{bake}	bk pump spd	176 l/s	F_{op}	pumpspeed	1 x 10^5 l/s
$T_{bake\ max}$	max bake T	425 K	τ_o	molecular oscillation period	1 x 10^{-12} sec
t_{bake}	length of bake	4 weeks	σ_{init}	initial water load	150 mlayers
			$\sigma_{end\ bake}$	end bake water load	~ 6 mlayers
			α	accomodation coefficient	0.5

The model surface parameters given in **Table 2** were first determined during the test beamtube bakeout at the Chicago Bridge and Iron Co and iterated during the bakeout of the 8 tube modules at Hanford and Livingston. The model was fit to the pressure vs temperature and time history. The total water load (σ_{init}) and RD peak adsorption energy (T_o) are uniquely fit by the bakeout data. The repulsive potential term (R) and the accomodation coefficient (α) are strongly correlated and both affect the estimate of the ultimate surface load ($\sigma_{end\ bake}$) at the end of the bake. In the parameter estimates the accomodation coefficient was held at 0.5, a typical literature value , while R was allowed to vary.

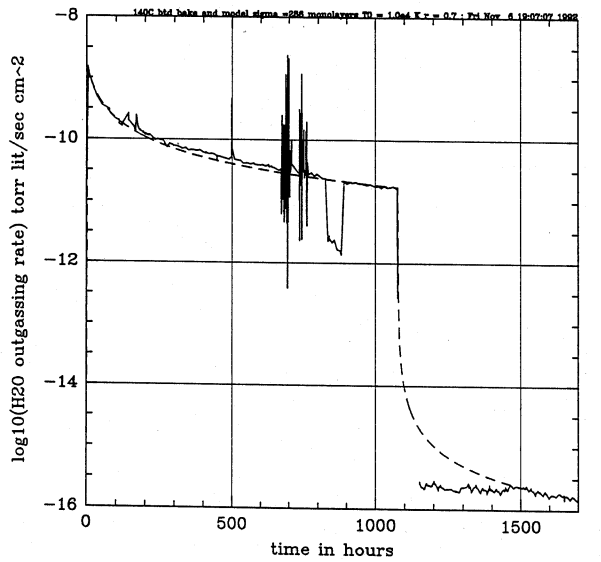


Figure 1 The water outgassing data and model (dashed lines) of a trial bakeout of a 40 meter full scale LIGO beamtube at Chicago Bridge and Iron Co. The model used the same theory and similar parameters to those listed in the **Table 2**.

Original estimate The original estimate assumed, that after the beamtube bake, water introduced at the ends would be uniformaly distributed in the tubes and all the new water would adsorb on the walls at sites made available by the bake. The relation between the water introduced and the increase in the average pressure in the tube was determined by calculating the change in outgassing rate with surface water loading using the site occupation probabilities after the bakout. The program was run repeatedly with slightly different initial conditions to explore the change in final outgassing rate at 296K as a function of small changes in the final surface loading. The derivative of final outgassing rate with final surface loading was obtained at 296K. Armed with the derivative, the last step in the estimate was to calculate the average pressure using **eq 1** and then, lastly, calculating the amount of water injected at the ends on the assumption that all the injected water is uniformly adsorbed on the surface of the tube - a conservative assumption. Furthermore, in this approach, it is assumed that the water injected into the tube after the bake would occupy similar adsorption sites as that introduced prior or during the bake - a wrong assumption which it turns out is optimistic.

Table 3 provides the results that were used to establish the water injection limits in the memo LIGOT990000 written on 08/22/99 “Water Load on the Beam Tubes from Detector Components”

that set 400 torr liters/module end of injected water as the limit before compromising the goal average pressure.

Table 3: Model results: increase in outgassing rate with water injected into the baked beam tube

injected water	*surface load*	*water outgassing rate @ 300K*
torr liters	*monolayers*	*torr liters/sec cm^2*
0	6.4138	4×10^{-17}
25	6.4233	2×10^{-16}
225	6.50	1.3×10^{-15}
485	6.60	2.6×10^{-15}
1500	7.0	8.1×10^{-15}

The new analysis The reanalysis of the beam tube water pressure as a function of water load includes the time and spatially varying surface load in the same surface model as that used to estimate the molecular dynamics of the bakeout. The new analysis includes the non-uniform distribution of the adsorbed water and the molecular dynamics at the surface with the surface parameters determined by the bake. The water is injected into the beamtube as a beam transmitted by the cryopumps at the beamtube ends. The water molecules are no longer assumed to adsorb on contact but rather the probability of adsorption using the detailed balance techniques in the bake program are followed. The water pumping by the liquid nitrogen traps at the beam tube ends is included in the calculation. This approach qualitatively changes the way the estimate is made.

By running the detailed balance program, one finds that the relevant parameters are the water pressure at the entrance to the cryopump. The pressure distribution in the beamtube is determined by the beaming through the trap transmission and the standard diffusive free molecular flow to the cryopumps. In the low pressure regimes encountered in the apparatus, the pumping time constants are shorter than the adsorption times so that to first order the pressure distribution is determined by standard pumping calculations neglecting the surface adsorption. The equilibrium pressure distribution in the tube determines the adsorption rates. The emission and adsorption times for the adsorption sites depend on the activation energy and on the probability that a site is occupied. The adsorbing sites take exponentially longer to both emit and adsorb as a function of the adsorption energy. After the bake all sites at short emission and adsorption times are empty. As a consequence, the longer the surface is exposed to the gas the more gas is adsorbed and at progressively higher adsorption energies. Should the gas be removed in the tube, the surface will outgas with a 1/t dependence scaled by the exposure time. A simple algorithm at the end of this document using this description of the process provides results close to those derived from the computer program.

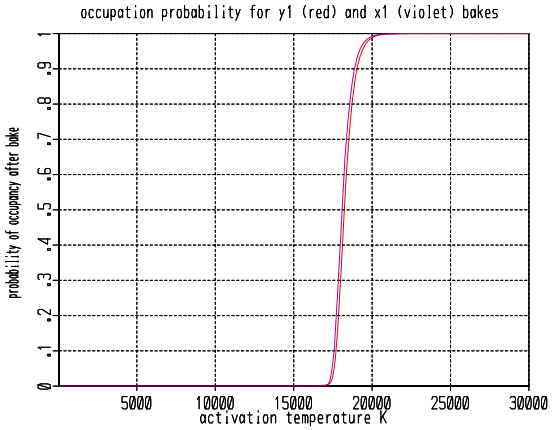


Figure 2 The site occupation probability after the 1 month bake at 423K. This probability distribution is the starting point for the new estimate. As reference, the model assumes that the occupation probability is 1 for all sites before any pumping on the surface. The activation temperature is the binding energy for water molecules divided by Boltz mann's constant. The two curves indicate limits for the module bakes which had slightly different bake times and temperatures

.

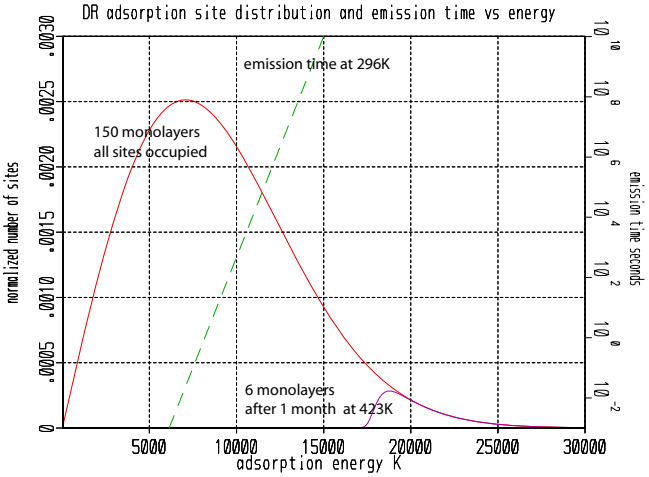


Figure 3 The DR molecular surface distribution as a function of activation energy before pumping and after the 423 K bakeout. The fully loaded surface holds 150 monolayers and 6 tightly bound monolayers after the bake. The emission time of an adsorption site is plotted as a function of the adsorption energy. The adsorption energies that influence the outgassing at room temperature lie between 10000 and 15000 K.

With **Figures 2** and **3** at hand it is worth evaluating several of the time scales associated with the pumping and surface dynamics. The average time for an occupied site to emit a water molecule is given by $\tau_{emit} = \tau_0 e^{\frac{T_{bind}}{T}}$. With T_{bind} = 18000K, the binding energy associated with the transistion between unoccupied and occupied sites at the end of the bake and a surface temperature of

423K, the value during the bake; the emission time is close to 1 month. The duration of the bake-out. After the bake the surface temperature is reduced to 296K and these sites are frozen out of the dynamics with emission times of millions of years. The pumping time constant associated with tube sections in the middle of the beamtube is roughly (dealt with correctly in the computer program) $\tau_{pump} \sim \frac{L^2}{4\,a\,v}$, a few hours for water at room temperature.

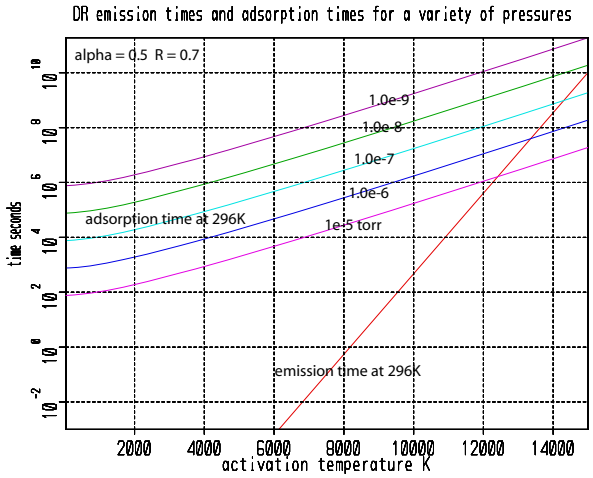


Figure 4 Adsorption times at 296K for different pressures at the surface vs site activation energy. The emission time is the same as in **Figure 3** .

The adsorption time for gas above the surface to occupy a site ,varies as

$$\tau_{ads} = \frac{4\sigma_{init}}{\alpha\rho v\left[1+\frac{(1-R)T_{bind}}{T}\right]e^{-\frac{(1-R)T_{bind}}{T}}}$$

where σ_{init} is the initial surface loading expressed in molecules per cm^2 (150 monolayers corresponds to 1.5×10^{17} molecules/cm^2) and ρ is the pressure expressed in molecules per cm^3 (10^{-8} torr is equivalent to 3×10^8 molecules per cm^3 at 300K) . The adsorption time is inversely proportional to the pressure above the surface. **Figure 4** shows the adsorption time as a function of the adsorption energy for a variety of pressures above the surface. In the pressure and pumping regimes we encounter in the apparatus ($p < 10^{-5}$ torr), the adsorption times are longer than both the emission times for a site and longer than the pumping time. Under these circumstances the pressure distribution in the tube is close to being determined by standard free molecular diffusion in equilibrium with the pumps without regard to the adsorption. There is adsorption, the change in the probability of site occupancy varies as

$$\frac{dP}{dt} \sim \frac{1}{\tau_{ads}} - \frac{(\tau_{ads}+\tau_{emit})}{(\tau_{ads}\tau_{emit})}P$$

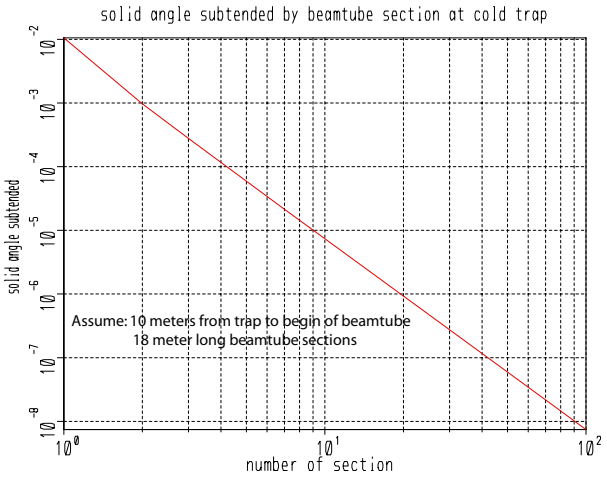


Figure 5 The solid angle subtended by the actual tube sections at the entrance to the cryopump. The solid angle determines the amount of beamed water hitting each 18meter section of the tube. The beaming is incorporated into the computer program.

The water injected into the tube is transmitted as a beam through the trap. The amount of water hitting a section of the tube is

$$\dot{Q} = \frac{\Delta\Omega_{\text{section}}}{4\pi}\rho v A_{\text{trap entrance}}$$

where v is the molecular velocity, A the area of the trap entrance , ρ the molecular density at the entrance, and $\Delta\Omega_{\text{section}}$ the solid angle subtended by the tube section. The section is determined by the finite element size in the computer program. **Figure 6** shows the pressure distribution in the tube with this beaming.

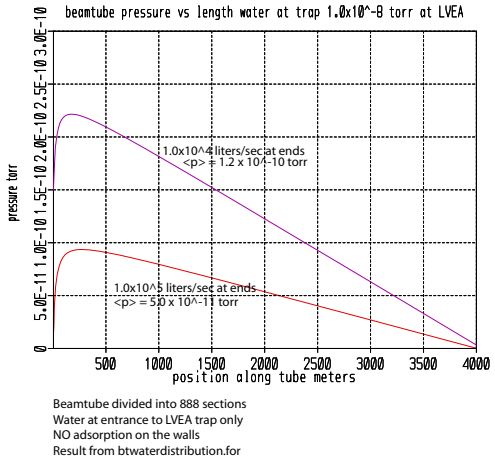


Figure 6 The pressure distribution in the beamtube for cryo pumping at both ends with a water pressure of 10^{-8} torr at the entry to the cryopump at the LVEA. The water is transmitted by the cryotrap and is beamed down the tube.. The curves are plotted for two different pumping speeds at the ends. The goal average pressure of less than 10^{-10} torr is satisfied with the existing cryopumps (10^5 liters/sec) if both ends have water injection at pressure less or equal to 10^{-8} torr.

The finite difference program that generated **Figure 6** (btwaterdistribution2a.for) is included in the Appendix. The method is to divide the tube into 888 finite element sections and carryout free molecular flow between them. The pumps are placed at the ends and the beaming through the cryotrap takes place at one end.

The same finite difference program is adapted to the surface dynamics using the DR adsorption state distribution and the Langmuir adsorption theory (btwaterdistsurf2b.for) with the Roald Hoffman adsorption potential. A relevent result from this program is shown in **Figure 7** which plots the average water pressure along the beamtube as a function time for a range of pressures at the trap input. The configuration is again two cryopumps, one at each end of the tube, and beaming of the water from one end. The pressure at the cryopump entrance is indicated in the figure running in decades from 10^{-6} to 10^{-9} torr. Curves are drawn for both the case with surface adsorption and without. At 1000 minutes the average pressure for all cases has come to steady state. At this time the water pressure at the input to the cryotrap is turned off. The curves without surface adsorption all drop exponentially toward lower pressure from that time. The curves that include adsorption follow the prior curves for the higher injection pressures but then show a typical desorption curve varying as 1/t after the water injection has been turned off. The curve with 10^{-8} torr injected water pressure is affected by the outgassing of the surface due to the remaining 6 monolayers after the bake and, finally, the curve with 10^{-9} torr injected water at the trap is swamped by this outgassing.

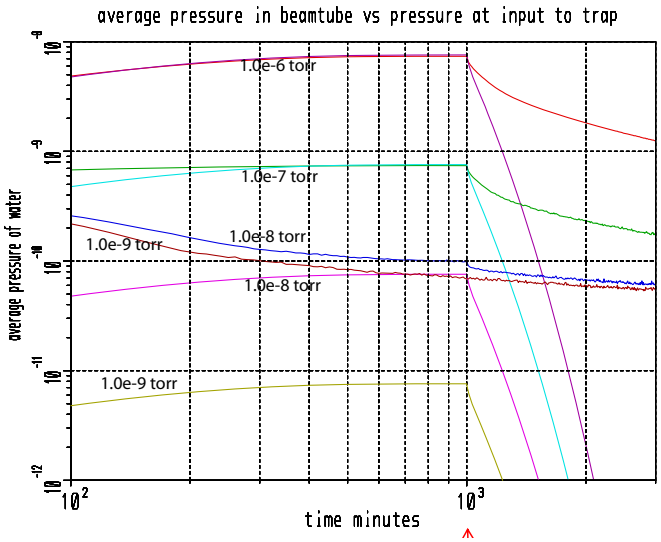


Figure 7 A key result of the program is shown here.The average pressure in the beamtube vs time is plotted. The system is cryopumped at both ends but water is injected at one end and beamed into the tube. The water pressure at the input to the cryopumps is indicated. The violet, light blue, magenta and brown yellow curves are the average pressure as a function of time with no surface adsorption. The injection pressures for the curves are indicated in the figure. At 1000 minutes the injection pressure is reduced to close to vanishing .The red, green, blue and brown curves include the surface adsorption again with the same sequence of trap input pressures. The steady state average water pressure is about 160 time smaller than pressure at the trap input for the high pressure cases. The 1/t desorption, so characteristic of the water outgassing before bake, is reinitiated by the water load. The lower adsorption energy sites that have been populated by the injected water have shorter emission times than the tightly bound sites still occupied after the bake. These exhibit a much shallower 1/t dependence.

One result seen in **Figure 7** and other runs is that the specification to keep the average pressure below the goal pressure is that the water pressure at the entrance to the cryo trap should stay below 10^{-8} torr. There is then no condition on the length of time the tube is exposed to the pressure. Furthermore, if the pressure at the trap entrance is initially higher but eventually does make it to 10^{-8} torr, the average pressure in the tube will fall to the goal pressure with a 1/t dependence. I*n other words, the initial calculation limit imposing a fixed amount of water entering the beam tube is an incorrect way to characterize the situation.* As we will see presently, there are practical limits imposed by the slow 1/t desorption dependence which do restrict the amount of water injected.

Several other cases and findings help to bring understanding to the process and make the adsorption less mysterious and more intuitive.

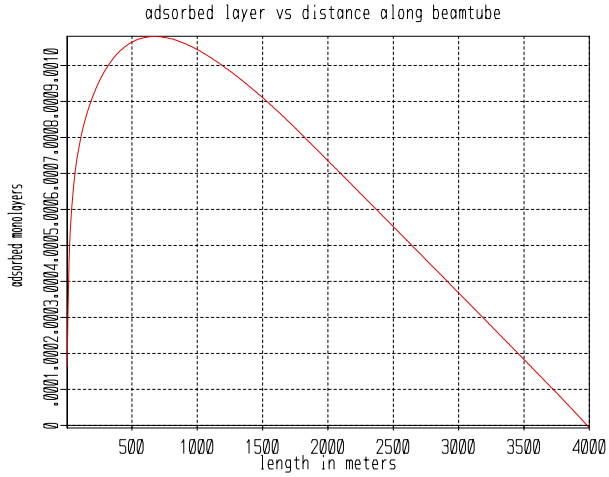


Figure 8 The distribution of adsorbed water on the surface 3000 minutes after a pressure of 10^{-6} torr was removed from the input to the cryopump. The time dependence of the average pressure in the beamtube is the top (red) curve in **Figure 7**. The newly adsorbed water at this time amounts to an average of 5 x 10^{-4} monolayers (2.5 torr liters). Note that the peak in the adsorption occurs about 600 meters from the injection point, further evidence that the adsorption time is not faster than the pumping time. The prior estimate for the amount of water adsorbed on the surface to have sufficient outgassing to compromis the goal average pressure was an increase of about 0.2 monolayers, much larger than that of the figure. The reason is that the water bound to the surface after the bake has an activation energy of 18000K which is associated with a emission time of a month at 423K while the newly adsorbed water in the figure has an activation energy of 11500K associated with an emission time of about 1000 minutes. The outgassing rate is proportional to the surface coverage divided by the emission time. Equal outgassing rates from the different activation energy sites requires vastly different surface loadings.

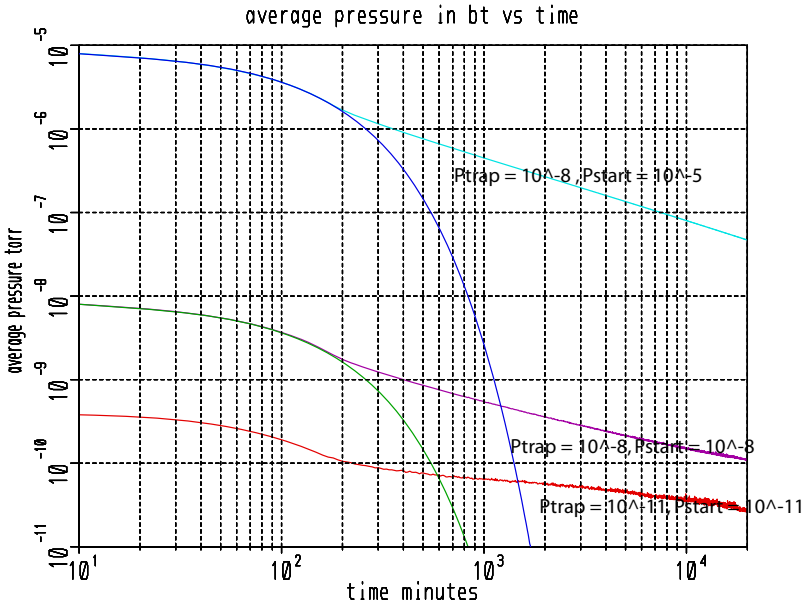


Figure 9 A different kind of model run than in **Figure 7.** Here the entire beamtube is initially filled with water vapor at the pressure P_{start} while the entrance to the trap is held at pressure P_{trap} through the entire calculation. Both the situations with and without surface adsorption are plotted. The blue and green curves are the exponential pump out of the tube with cryotraps at the ends and no adsorption. The light blue, violet and red curves include the outgassing of the monolayer left after the bakeout and the 1/t desorption from the newly adsorbed layers. The red curve is entirely due to the adsorbed gas after the bakeout.

Some pedagogy One can always run the program to get useful answers about specific situations. To gain insight into what is really happening, it is useful to look in more detail at various simple cases. A useful clue comes from **Figure 10** which shows the average pressure in the beamtube for two different exposure times. In this calculation the tube begins with no water pressure other than what is outgassed by the occupied sites remaining after the bake. At the beginning a pressure of 10^{-6} torr is applied at the trap entrance and the gas is beamed into the tube which is pumped by the two crypumps at the ends. The two curves have different times by a factor 10 before the pressure at the trap entrance is reduced to a small value. The important thing to notice is that the fall off curves are very similar in shape even though there is a factor of 10 ratio in the amount of gas injected and adsorbed.

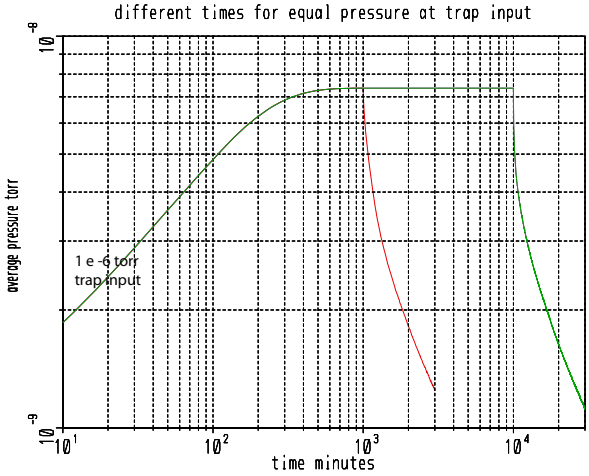


Figure 10 The average pressure in the beamtube as a function of time. The pressure in the tube is 10^{-10} torr at the beginning. At this time the pressure at the trap entrance is increased to 10^{-6} torr. The red curve shows the pumpout and desorption following a reduction of the pressure at the trap at 1000 minutes while the green curve shows the pumpout and desorption following a reduction of the pressure at the trap at 10000 minutes. The desorption curves have the same 1/t shape but with a factor of about 10 ratio in time and in quantity of gas. Such self similar curves imply a fractal basis for the process. 1/t curves come about when exponential decays of different but neighboring time constants are superposed - a little integral calculus will prove this to you.

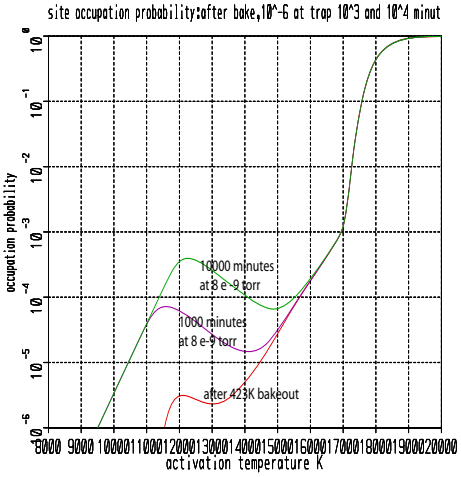


Figure 11 A logarithmic plot of the occupation probability vs site adsorption energy after the events described in **Figure 10**. The red curve is the occupation probablity at 296K after the 423K bakout of the tube for 1 month. The small peak at 12000 K is due to the adsorption of the gas in the beamtube as it approached 296K during cool down after the bakeout. The peak would be smaller had we used larger pumps during the bakeout. The violet curve is the newly adsorbed water during the injection for 1000 minutes. Note that the peak in the curve at 11500 K corresponds to emission times close to 1000 minutes at 296K. The green curve is that due to the water being adsorbed for 10000 minutes. The occupation probabilities are about 10 times higher and the water is more tightly bound now peaking at sites with emission times of 10000 minutes. It is easy to see what is going on in this model, longer immersion times cause both more gas to be adsorbed and at higher adsorption energies. Crudely one can estimate that the 1/t dependence in the desorption will be in units of the immersion time. The initial outgassing rate of the water will be about the same for the two cases. Even though more water is on the surface for the longer immersion, the emission rate is smaller since the adsorption energy is larger.

Quick estimate for the average pressure in the tube after water injection An estimate for the time it takes to achieve the goal average pressure when the input to the trap has been held at water pressure p_{trap} for a time t_{on} is given by

$$<p> \sim \left(\frac{3\pi^2 a}{4 L_{max\ p}} \right) p_{trap} \left(\frac{t_{on}}{t} \right) \qquad \textbf{eq 2}$$

where $L_{max\ p}$ is the distance from the trap into the tube where the maximum pressure occurs, about 600 meters, a is the tube radius. The quantity in the large brackets is the ratio of the average pressure to that at the trap, about 1/140 as determined from the computer program. As an example, if the pressure at the trap is 10^{-7} torr for a month and then reduced to 10^{-8} torr, the average pressure in the tube is 7×10^{-10} torr for that month and it will require 7 months before the average pressure has dropped to the goal average pressure.

Measurements one could perform to validate the estimate Two techniques come to mind. A well outgassed RGA installed at the port closest to the peak in the adsorbed water, about 600 meters from the trap, (the second or third port on the beamtube) should be able to detect the increase in the water pressure when the pressure at the trap is increased. The measurement would need to have a sensitivity of 10^{-10} torr of water with a stability of several days. The instruments we have would need to operate in either counter or SEM mode as the Faraday mode will give only about 10^{-14} amperes for this pressure, just barely enough to see above the noise in the electrometers. The difficulty in making the measurement is in bringing power to the beamtube and being careful enough in the bakeout of the 10 inch valve on the beamtube and the RGA not to be overcome by the adsorbed water and its outgassing rate dependence on temperature. The best part of this technique is that we have the equipment to carry it out.

An alternative technique is to measure the infrared absorption by the column of water in the tube. At 10^{-10} torr there are 1.2×10^{12} molecules of water/cm^2 in the 4 km beam path. With a frequency modulated laser which alternately is off the infrared absorption line and then at its peak, it should be possible to carry out shot noise limited absorption measurements specific to the water. **Figure 12** shows the position of the stronger water absorption lines in the infrared region.

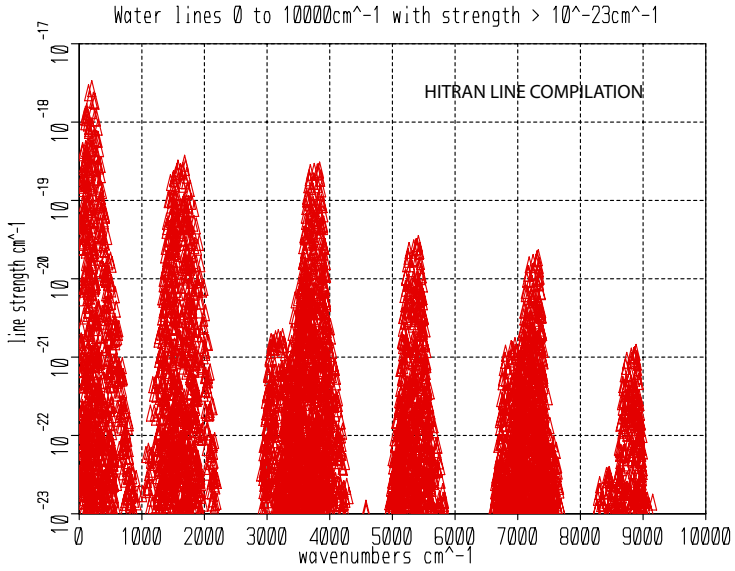


Figure 12 A histogram of the stronger water lines in the infrared. Transitions between the rotational and vibrational states of water lead to several hundred thousand well seperated lines. The line widths are typically Doppler broadened to 0.1 cm^{-1}. The lines at around 4000 cm^{-1} seem most technologically accessible. They can be excited with tunable lead salt lasers providing several 100's mW. At the goal average pressure the absorption at line center will correspond to about 0.3ppm in a single pass of the beamtube and is linear in the column density. The compilation of water lines comes from the HITRAN listing maintained by Laurance Rothman at the Harvard-Smithsonian Center for Astrophysics.

A nice senior thesis for a one of our students.

References:

Roald Hoffman Surface potential
Solids and Surfaces: A Chemist's View of Bonding in Extended Structures
Roald Hoffman
VCH, New York (1988)

Adsorption Isotherms
M.M. Dubinin and L. V. Radushkevich
Proceedings Acad Sci USSR **55**, 331 (1947)

Langmuir Surface adsorption model
S. Dushman
Scientific Foundation of Vacuum Technique
Wiley, New York (1962)

APPENDIX B

The Statistical Mechanics Water Outgassing Model

The model uses Langmuir adsorption theory with a skewed Gaussian adsorption site energy distribution. The model includes readsorption on the surface characterized by a potential with an initial repulsive hill facing the vacuum followed by an attractive well. For ease of calculation, the repulsive and attractive potentials are held in a fixed ratio. The model is designed to describe the state of the surface far from equilibrium and results in the Dubinin–Radushkevich isotherms at equilibrium.

The input parameters to the model are: the initial water loading, n (units of monolayers) where one monolayer is assumed to have $\sigma_0 = 1 \times 10^{15}$ water molecules per cm^2, the average binding energy expressed as a temperature, T_0, and the ratio, R, of the repulsive to the attractive value of the surface potential.

The model predicts that the ratio of n/T_0 determines the initial outgassing rate when there is sufficient pumping capacity to neglect readsorption. The rise in outgassing rate with temperature is not simply related to T_0 but depends on the surface loading and the time of pumping prior to the bake. As a consequence it is necessary to measure n to get a reasonable estimate for T_0. The ratio of the replusive to attractive part of the surface potential, R , is influential in determining the readsorption rate and is best estimated from data when the system is accumulating (isolated from the pumps) or from the gradual change of outgassing rate after the system has been baked and returned to room temperature. The final outgassing rate after a bake is dependent on all three parameters. The accomodation rate, often used in adsorption calculations, is so strongly coupled to the surface potential properties that it is set to .5 in the model.

The basis of the model is the assumption that the surface can be described by a distribution function of water adsorption sites with different binding energies. This assumption alone gives the approximate $1/t$ dependence of the outgassing rate over the range of pumping times involved. Although many different distribution functions give similar behaviour (box, exponential) the most physically appealing is the skewed Gaussian distribution which leads to the Dubinin–Radushkevich (1946)(DR) adsorption isotherms at equilibrium.

The surface coverage, σ molecules per cm^2, of adsorbed molecules at equilibrium (equal emission and readsorption) at a temperature T is given by the DR theory as

$$\frac{\sigma}{\sigma_m} = e^{-(T/T_0)^2 ln^2(P/P_0)}$$

where σ_m is the surface coverage at the saturation vapour pressure P_0 when all available sites are filled. P is the pressure at equilibrium with the surface held at temperature T and T_0 is the average binding energy expressed as a temperature. T_0 is also the spread in energy of the adsorption sites.

The distribution function of sites with adsorption energy T_{bind} that leads to the DR equation is the skewed Gaussian distribution

$$\theta(T_{bind}) = \left(2T_{bind}/{T_0}^2\right) e^{-(T_{bind}/T_0)^2}$$

The distribution function in the above form is normalized so that

$$\int_0^\infty \theta(T_{bind})\delta T_{bind} = 1$$

The dynamics at the surface is described by the Langmuir adsorption hypothesis. Let $P(T_{bind}, t)$ be the probability that an adsorption site with binding energy T_{bind} is occupied at time t and assume, furthermore, that only one molecule may be adsorbed per site. The rate of change of the probability is then given by

$$\frac{dP(T_{bind}, t)}{dt} \;=\; -\;\frac{P(T_{bind}, t)}{\tau_{emit}} \;+\; \frac{(1 \;-\; P(T_{bind}, t))}{\tau_{ads}} \qquad \text{eq (1).}$$

τ_{emit} is the desorption or emission time for an adsorption site with binding energy T_{bind} given by the standard Boltzmann factor

$$\tau_{emit}(T_{bind}) = \tau_0 e^{T_{bind}/T}$$

where T is the temperature of the surface and τ_0 is the oscillation period of the molecule in a typical binding site. In the model τ_0 is set at 1×10^{-12} seconds.

τ_{ads} is the readsorption time for an adsorption site with binding energy T_{bind}. In order to get the experimentally observed readsorption rates it was found necessary to add another degree of freedom to the outgassing model by providing both an attractive and a repulsive term in the potential experienced by the molecule at the surface. The binding potential (depth of the well) is T_{bind} while the binding potential is RT_{bind} below the potential far from the surface. This leaves a potential barrier $(1 - R)T_{bind}$ at the vacuum side of the surface. The readsorption time is thereby increased over the case with a simple well. The construct of the potential barrier buries a host of physical phenomena such as the means by which the molecule actually accomodates to the surface by losing its initial kinetic energy to the excitation of phonons at the surface or the fraction of inelastic collisions made by the water molecules. R is determined from the model by fixing a reasonable value for the accomodation coefficient, $\alpha = 0.5$. In order to preserve the mathematical simplicity of the model R is assumed independent of T_{bind}.

The readsorption time is determined by assuming that only molecules hitting the surface with a kinetic energy greater than $(1 - R)T_{bind}$ can bind. The integral over the Maxwell distribution of velocities in the gas gives the readsorption time as

$$\tau_{ads} \;=\; \frac{4n\sigma_0}{\alpha\rho v_{th}(1 + (1 - R)T_{bind}/T)e^{-(1-R)T_{bind}/T}}$$

where v_{th} is the average velocity of the water molecules in the gas at temperature T and ρ is the water molecule density in the gas.

The time evolution of the probability of occupancy is given by the integration of equation (1) as

$$P(T_{bind}, t) \;=\; P(T_{bind}, 0)e^{-t/\tau} \;+\; P_{equil}(T_{bind})(1 \;-\; e^{-t/\tau})$$

using the definitions

$$\tau = \frac{\tau_{emit}\tau_{ads}}{(\tau_{emit} + \tau_{ads})}$$

$$P_{equil}(T_{bind}) = \frac{\tau_{emit}}{(\tau_{emit} + \tau_{ads})}.$$

The contribution of the outgassing rate as a function of time from a band of sites with binding energy interval δT_{bind} is expressed as

$$dJ_{out}(t) = n\sigma_0 \ \theta(T_{bind})(\frac{dP(T_{bind},t)}{dt}) \ \delta T_{bind}$$

The total outgassing integral for the case of $\tau_{ads} \rightarrow \infty$ (no readsorption) and an initial site occupation probability of 1 for all sites can be written in closed form as

$$J_{out}(t,T) = \left(\frac{2n\sigma_0 T}{tT_0}\right) \int_0^a bln(y/a)e^{-(bln(y/a))^2} e^{-y} dy$$

where $b = T/T_0$ and $a = t/\tau_0$.

The more general case including water emission and readsorption with changes in temperature becomes sufficiently complex to require a computer code. The code has been programmed using double precision in FORTRAN. The algorithmic steps of the program are based on solving the time evolution of the probability as a function of experiment set temperatures in each of 1024 energy bins T_{bind} spread uniformly between zero to $3T_0$. The basic interval for computation is $\Delta t/\tau_s = f$ where τ_s is the vacuum system time constant $= V/F$, V is the volume and F the pumping speed. Usually sufficiently accurate results are obtained with $f < 0.2$ in an iteration that first calculates the site occupancy from

$$P(T_{bind}, t_{j+1}) = P(T_{bind}, t_j) \ e^{-f\tau_s/\tau_j} + P_{equil}(T_{bind}, t_j) \ (1 - e^{-f\tau_s/\tau_j})$$

and then establishes the time evolution of the surface water loading by taking the sum

$$\sigma(t_{j+1}) = n\sigma_0 \sum_0^{3T_0} \theta(T_{bind}) P(T_{bind}, t_{j+1}).$$

The outgassing rate is estimated by taking the difference in the surface loading per iteration and dividing by the computation time interval

$$J(t_{j+1}) = \frac{(\sigma(t_{j+1}) - \sigma(t_j))}{f\tau_s}$$

. With this value of the outgassing rate, the pressure, p, is calculated by

$$p(t_{j+1}) = p(t_j) \ e^{-f} + (\frac{J(t_j)A}{F})(1 - e^{-f})$$

where A is the surface area of the system. The pressure is used in determining $\tau_{ads}(t_{j+1})$ for the next estimate of $P_{equil}(t_{j+1})$ beginning the next step in the iteration until the entire schedule of temperature vs time has been completed.

REFERENCES

Dubinin,M.M. and Radushkevich,L.V. *Proc. Acad. Sci.,USSR* **55**, 331, (1947).

Heuristic Statistical Mechanics Model

Dubinin-Radushkevich (DR) equilibrium surface coverage:

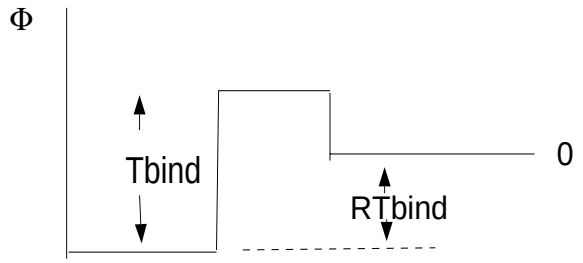
$$\frac{\sigma}{\sigma_m} = e^{(T/T\ \ _0)^2 ln^2(P/P_0)}$$

DR site distribution function:

$$\theta(T_{bind}) = \ \ 2T_{bind}/T_0{}^2\big)e^{(T\ \ _{bind}/T_0)^2}$$

$$\int_0^\infty \theta(T_{bind})\delta T_{bind} = 1$$

Heuristic surface potential:



Emission time:

$$\tau_{emit}(T_{bind}) = \tau_0 e^{T_{bind}/T}$$

Adsorption time:

$$\tau_{ads} \ = \ \frac{4n\sigma_0}{\alpha\rho v_{th}(1+(1\ \ R)T\ \ _{bind}/T)e^{(1R)T\ \ \ \ _{bind}/T}}$$

Detailed balance per site:

$$\frac{dP(T_{bind},t)}{dt} \ = \ \ \ \frac{P(T_{bind},t)}{\tau_{emit}} \ + \ \frac{(1\ \ \ P(T\ \ _{bind},t))}{\tau_{ads}}$$

Integration:

$$P(T_{bind}, t) \;=\; P(T_{bind}, 0)e^{-t/\tau} \;+\; P_{equil}(T_{bind})(1 \;-\; e^{-t/\tau})$$

where

$$\tau = \frac{\tau_{emit}\tau_{ads}}{(\tau_{emit} \;+\; \tau_{ads})}$$

and

$$P_{equil}(T_{bind}) \;=\; \frac{\tau_{emit}}{(\tau_{emit} \;+\; \tau_{ads})}.$$

Incremental outgassing rate of band of sites:

$$dJ_{out}(t) = n\sigma_0 \;\theta(T_{bind})(\frac{dP(T_{bind}, t)}{dt})\;\delta T_{bind}$$

Aside: for $P(T_{bind}, 0) \;=\; 1$ and $\tau_{ads} \rightarrow \infty$

$$J_{out}(t, T) = \left(\frac{2n\sigma_0 T}{tT_0}\right) \int_0^a bln(y/a)e^{-(bln(y/a))^2} e^{-y} dy$$

where

$$b = T/T_0 \qquad a = t/\tau_0$$

Computational algorithm (waterbakesm.f)

Step time:

$$\Delta t/\tau_s \;=\; f \quad \tau_s \;=\; V/F$$

Probability computation over 1024 binding energies $0 \rightarrow 3T_0$

$$P(T_{bind}, t_{j+1}) \;=\; P(T_{bind}, t_j)\, e^{-f\tau_s/\tau_j} + P_{equil}(T_{bind}, t_j)\,(1 - e^{-f\tau_s/\tau_j})$$

Surface coverage:

$$\sigma(t_{j+1}) \;=\; n\sigma_0 \sum_{0}^{3T_0} \theta(T_{bind}) P(T_{bind}, t_{j+1}).$$

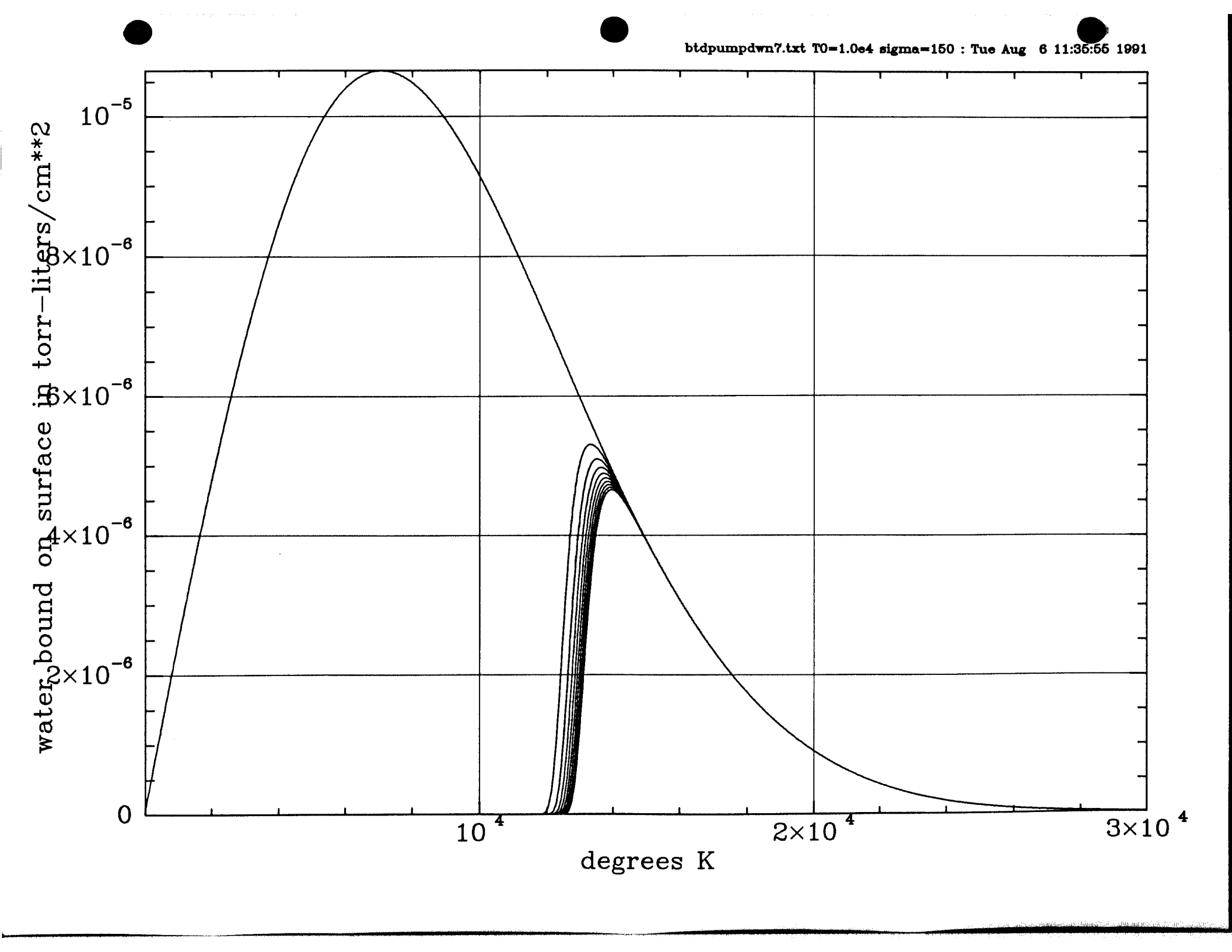
Outgassing rate:

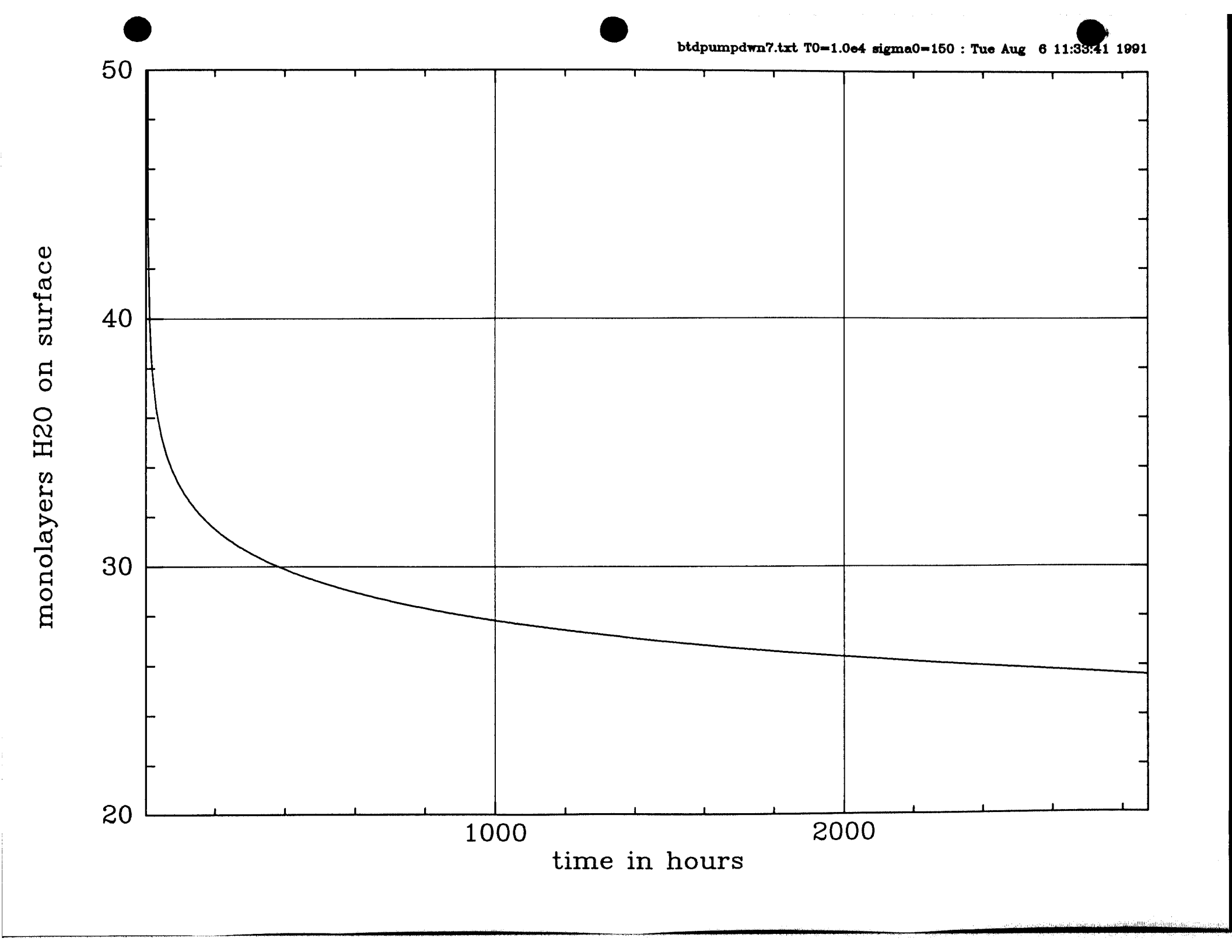
$$J(t_{j+1}) \;=\; \frac{(\sigma(t_{j+1}) \;-\; \sigma(t_j))}{f\tau_s}$$

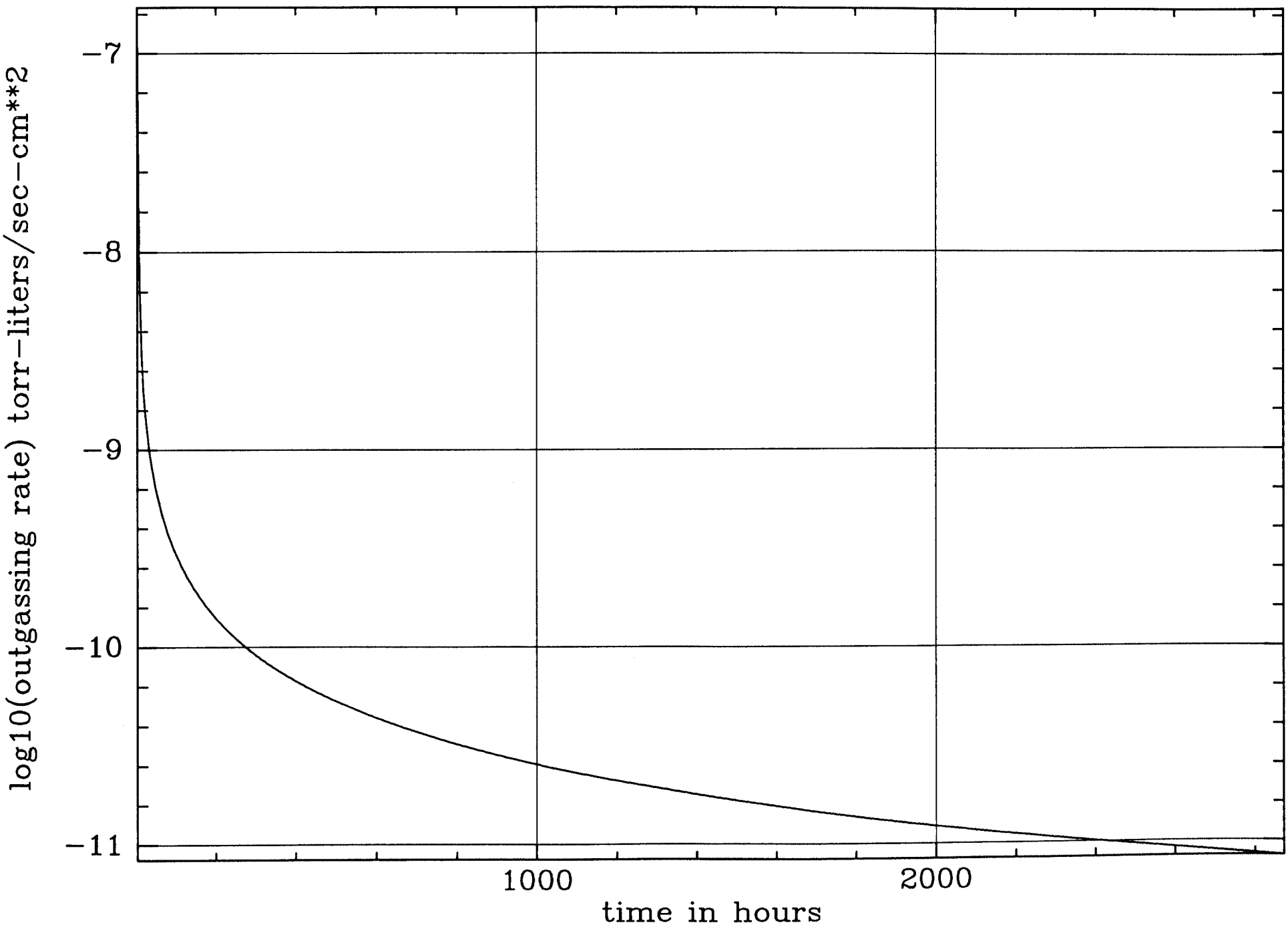
Pressure:

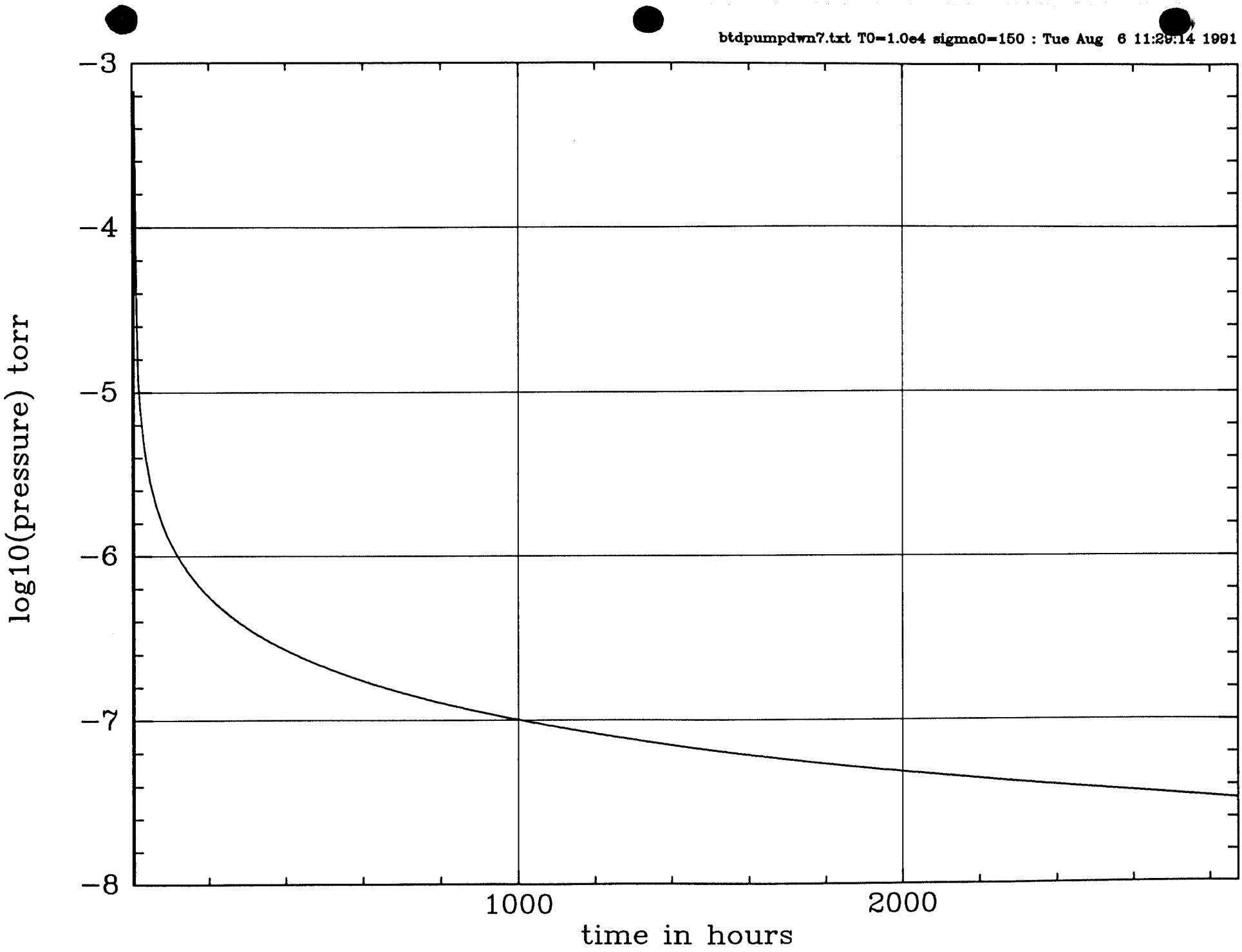
$$p(t_{j+1}) \;=\; p(t_j)\; e^{-f} \;+\; (\frac{J(t_j)A}{F})(1 \;-\; e^{-f})$$

GO BACK AND DO IT AGAIN (new time and temperatures)


btdpumpdwn7.txt T0=1.0e4 sigma=150 : Tue Aug 6 11:35:55 1991
water bound on surface in torr-liters/cm**2
10^{-5}
8×10^{-6}
6×10^{-6}
4×10^{-6}
2×10^{-6}
0
10^{4}
2×10^{4}
3×10^{4}
degrees K


btdpumpdwn7.txt T0=1.0e4 sigma0=150 : Tue Aug 6 11:33:41 1991
50
40
30
20
monolayers H20 on surface
1000
2000
time in hours


btdpumpdwm7.txt T0=1.0e4 sigma0=150 : Tue Aug 6 11:30:50 1991
log10(outgassing rate) torr-liters/sec-cm**2
-7
-8
-9
-10
-11
1000
2000
time in hours


btdpumpdwn7.txt T0=1.0e4 sigma0=150 : Tue Aug 6 11:20:14 1991
−3
−4
−5
−6
−7
−8
log10(pressure) torr
1000
2000
time in hours

```
c *******btwaterdistsurf2b.for   October 13, 2008
c Program estimates the water pressure above the surface when
c loaded with water from geometric transmission by the trap and
c steady state pumping by the trap. The surface adsorption is
c treated by a program similar to waterbakesm using the numerical
c recipes Runge Kutta codes
c
c
c This version allows a change in the pressure at the trap
c while running in powers of ten per tstep
c
c This version writes a file of the occupation probabilities
c at the end for any tube section
c
c The program uses the Runge Kutta techniques from Numerical Recipes
c The calculation is made in cgs units with the pressure in torr
c
c  The outgassing is described by:
c
c      aj(k) = outgassing rate torr liters/sec in tube section k
c
c      ad(k) = the deposition of water on section k. given by
c      the pressure at the entrance to the trap multiplied by
c      the molecular speed times the solid angle of the section
c      subtended at the entrance to the trap normalized by 4*pi
c
c      a = tube radius
c
c      al = length of module/4*number of sections
c
c      f = pumping speed of the cryopump at one end of the module
c
c      v = thermal velocity of molecule at 300K
c
c      pt = water pressure at entrance to trap
c
c      alen0 = distance from front of trap to middle of the
c              first section of tube
c
c      fr(k) = the fraction of the injected molecules not adsorbed
c      on the first encounter with the surface
c
c Difference equation to solve
c
c Section 1:  dp/dt = ((2*v*a)/(3*al**2))*(p(2)-p(1)) + (2/a)*aj(1)
c end pt            - (f*p(1))/(pi*al*a**2)
c                   + fr(k)*pt*v*a**2/(4*al)*((1/(alen0)**2)-(1/(alen0+al)**2))
c
c
c
c Section k : dp/dt = ((2*v*a)/(3*al**2))*(p(k-1)-2*p(k) +p(k+1))
c                     + (2/a)*aj(k)
c                     + fr(k)*pt*v*a**2/(4*al)*((1/(alen0+al*(k))**2)-
(1/(alen0+al(k+1))**2))
c
c
c last section dp/dt = ((2*v*a)/(3*al**2))*(p(k-1)-*p(k))
```

```
c                     + (2/a)*aj(nsec)
c                   + fr(k)*pt*v*a**2/(4*al)*((1/(alen0+al*nsec)**2)-
(1/(alen0+al*(nsec+1))**2))
c                     -(f*p(k))/(pi*al*a**2)
c
c
c  The surface properties need to be calculated along with the pressure
distribution
c The results of the surface calculation are the outgassing rate in each section
aj(k)
c fr(k), the fraction of the directed "beam" from the trap entrance that is not
c adsorbed on the first encounter in section k
c
c The surface calculation uses the following additional variables
c
c t0 = the peak probability of the RD adsorption site distribution
c sigma0 = the saturated surface water loading in monolayers
c r = the repulsive potential of the surface
c alpha = accomodation coefficient of the surface for water
c tau0 = the molecular oscillation frequency at the surface : 1.0e-12 sec
c at(k) = the activation temperature of the surface site k
c ap(n,k) = the probability that site k is occupied in tube segment n
c temp = the surface temperature
c
c
c
c The program uses the Runge-Kutta algorithms given in Press et al
c Numerical Recipes 2 for the pumping dynamics and a simple inegration
c for the probability of occupancy evolution
c
        use winteracter
      character fileout*50,surfile*50
        dimension p(1000,8000),ystart(1000),time(8000),ystarts(1024)
      dimension fr(1000),aj(1000),at(1024),ap(1000,1024),w(1024)
      dimension aps(1024),z(1024)
      common /path/ kmax,kount,dxsav,xp,yp
        common /deriv/ aa,bb,cc,dd,ee,alen0,nsec,al
      common /deriv1/ t0,sigma0,r,alpha,tau0,temp,nsite,v
      write(unit=*,fmt=601)
601     format(' enter # of sections :' $)
      read(unit=*,fmt=602)nsec
602     format(i6)
      write(unit=*,fmt=1)
1       format(' enter pump speed lit/sec : ' $)
      read(unit=*,fmt=4)f
2       format(2e15.6)
      write(unit=*,fmt=605)
605     format(' enter amu of gas : '$)
      read(unit=*,fmt=4)amu
      write (unit=*,fmt=701)
701     format(' enter init p(torr) at n2 trap,lgth (m) to fst sect:'$)
        read(unit=*,fmt=702)ptinit,alen0
      write(unit=*,fmt=703)
703   format(' enter # 10^n reduction p(torr) at n2trap, time(min) : '$)
      read(unit=*,fmt=7022)nptfin,ptftm
7022  format(i5,e15.6)
702   format(2e15.6)
```

```
      alen0=alen0*100.0
4       format(e15.6)
c enter the fixed parameters
c beam tube radius cm
      a = 62.0
c thermal speed of gas in cm/sec
      v = (5.263e4)*sqrt(18.0/amu)
c module length cm
      al0 = 4.0e5
c number of sections
      an = real(nsec)
c section length cm
      al = al0/an
c conversion pumping speed from liters/sec to cc/sec
      f = f*1000.0
      write(unit=*,fmt=9)
9       format(' enter starting pressure in torr : '$)
      read(unit=*,fmt=4)pstart
      aa = (2.0*v*a)/(3.0*al**2)
      bb = 2.0/a
      dd = f/(3.14159*al*a**2)
      ee = (ptinit*v*a**2)/(4.0*al)
      do 1000 kt = 1,nsec
      p(kt,1)=pstart
1000    continue
      write(unit=*,fmt=7)
7       format(' time(minutes)/step, #steps total, #calsteps/step :'$)
      read(unit=*,fmt=8)tstep1,nstep,intstep
8       format(e15.6,2i6)
c        write(unit=*,fmt=451)
c451     format('  enter 1 to write calc values : '$)
c        read(unit=*,fmt=452)iwrt
c452     format(i3)
c convert time to seconds
      tstep = (tstep1/real(intstep))*60.0
      x2=0.0
      nvar = nsec
      eps = 1.0e-4
c       h1 = (1.0e-3)*tstep
c       hmin = (1.0e-7)*tstep
      h1 = (1.0e-7)*tstep
      hmin=(1.0e-11)*tstep
c surface program input
      write(unit=*,fmt=6011)
6011  format('  enter sigma0 surf coverage monoly,DR pk temp K :'$)
      read(unit=*,fmt=6022)sigma0,t0
6022  format(2e15.6)
c convert monolayers to particles/cm**2
      sigma0 = 1.0e15*sigma0
      write(unit=*,fmt=6033)
6033  format('  enter r repulsion term, alpha accom coef :'$)
      read(unit=*,fmt=6022)r,alpha
      write(unit=*,fmt=6066)
6066  format(' enter temperature K cr>=296K :'$)
      read(unit=*,fmt=4)temp
      if(temp.eq.0.0)temp=296.0
      write(unit=*,fmt=607)
```

```
607   format('  enter tau0 sec cr>=1.0e-12 sec : '$)
      read(unit=*,fmt=4)tau0
      if(tau0.eq.0.0)tau0 = 1.0e-12
      write(unit=*,fmt=604)
604   format('  enter activ T and occupt prob filename : '$)
      read(unit=*,fmt=6055)surfile
6055  format(a50)
      open(unit=3,file=surfile)
      read(unit=3,fmt=*)nsite
      do 610 k=1,nsite
      read(unit=3,fmt=*)at(k),aps(k)
610   continue
      close (3)
c set up weighting and activation energies
      deltat = 3.0*t0/real(nsite)
      sum = 0.0
      do 615 K=1,nsite
      w(k)=((2.0*at(k)*deltat)/t0**2)*exp(-(at(k)/t0)**2)
      sum = sum + w(k)
615   continue
c normalize
      do 616 k=1,nsite
      w(k)=w(k)/sum
      aj(k) = 0.0 !set first outgassing pf each sect 0
616   continue
c set all sections with equal water loading from file
      do 617 ksec = 1, nsec
      do 618 ksite = 1, nsite
      ap(ksec,ksite) = aps(ksite)
618   continue
617   continue
c start the calculation
c     write(unit=*,fmt=9099)tstep
c9099 format('  tstep = ' 1pe12.3)
      kill = 0
      do 100 k=1,nstep
      if(kill.ge.nptfin)go to 6666
      tmm = x2/60.0
      if(tmm.ge.ptftm)then
      kill = kill + 1
      ee=ee/(10.0**kill)
      end if
6666  do 300 j=1,intstep
      x1 = x2
      x2 = x1 + tstep
      if(j.eq.1.and.k.eq.1)then
      do 1002 n=1,nsec
      ystart(n)= pstart
1002    continue
      time(k)=x1/60.0
      p(nsec+1,k)=pstart
      end if
c the surface routine
c establish the fraction of beamed gas that sticks
      do 210 ksec=1,nsec
      do 220 ksite=1,nsite
      z(ksite)=ap(ksec,ksite)
```

```
220   continue
      call beamfraction(ksec,w,deltat,z,at,frr)
      fr(ksec)=frr
210   continue
      do 620 ksec=1,nsec
c     write(unit=*,fmt=901)ksec,k,j
c901  format('  main first: ksec = 'i5,' k = ' i5, 'j = 'i5 )
      do 725 ksite=1,nsite
      ystarts(ksite)=ap(ksec,ksite)
725   continue
      do 230 ksite=1,nsite
      z(ksite)=ap(ksec,ksite)
230   continue
c determine the evolution of the occupation probability at each site
      call probev(ksec,ystart,tstep,at,z)
      do 240 ksite=1,nsite
      ap(ksec,ksite) = z(ksite)
240   continue
c  determine the outgassing rate in each section
      sumb = 0.0
      suma = 0.0
      do 635 ksite=1,nsite
      suma = suma + ystarts(ksite)*w(ksite)*sigma0
      sumb = sumb + ap(ksec,ksite)*w(ksite)*sigma0
635   continue
      aj(ksec) = (suma-sumb)/(tstep*3.0e16)  !torr cc/sec/cm**2
c     ajtorr = aj(ksec)/1000.0  !convert to torr-liters/sec/cm**2
c if outgassing rate is < 0 , make it 0
      if(aj(ksec).lt.0.0)aj(ksec)=0.0
c     write(unit=*,fmt=681)j,ksec,ajtorr,sumb,suma
c681  format('  j=' i3,'sec='i3,'t-l/sec/cm**2='1pe12.3,'suma='1pe12.3,
c     &  'sumb='1pe12.3)
c     read(unit=*,fmt=9901)ig
c9901 format(i3)
620   continue
      call odeint(ystart,nvar,x1,x2,eps,h1,hmin,nok,nbad,fr,aj)
300     continue
      do 1003 n=1,nsec
      p(n,k+1)=ystart(n)
c        write(unit=*,fmt=420)time(k),n,p(n,k+1)
c420     format(' time ='1pe12.4,' n = 'i5,' pressure = '1pe12.4)
c        read(unit=*,fmt=421)junk
c421     format(i3)
1003    continue
      time(k+1)=x2/60.0
c calculate the average pressure
      sum = 0.0
      do 400 n=1,nsec
      sum  = sum + p(n,k+1)/an
400     continue
      p(nsec+1,k+1)=sum
      write(unit=*,fmt=11)k,time(k),p(nsec+1,k)
11      format(' k= 'i4,' time min='1pe15.6,  '<p>(torr)='1pe15.6)
100     continue
2000    write(unit=*,fmt=77)
77      format('enter fileout: '$)
      read(unit=*,fmt=78)fileout
```

```
78      format(a50)
      open(unit=2,file=fileout)
      write(unit=*,fmt=2001)nsec+1,nsec+2
2001    format(' enter #segmt or 'i4,'=<p> or 'i4,'=p vs sec @ end:'$)
      read(unit=*,fmt=2002)ks
2002    format(i6)
        if(ks.eq.nsec+2)then
        write(unit=2,fmt=79)nsec
        do 2060 kz=1,nsec
        xz = real(kz)*al/100.0
        write(unit=2,fmt=80)xz,p(kz,nstep+1)
2060    continue
        end if
      write(unit=2,fmt=79)nstep+1
79      format(i5)
      do 200 kj=1,nstep+1
      write(unit=2,fmt=80)time(kj),p(ks,kj)
80      format(1pe15.6,1pe15.6)
200     continue
      close (2)
      write(unit=*,fmt=8850)
8850  format(' enter 1 for monolayer and prob files : ' $)
      read(unit=*,fmt=8851)imono
8851  format(i3)
      if(imono.eq.1)then
      write(unit=*,fmt=8852)
8852  format('  enter output file name for initial monolayers : '$)
      read(unit=*,fmt=78)fileout
      open(unit=3,file=fileout)
      write(unit=3,fmt=8853)nsec
8853  format(i5)
      do 8860 n=1,nsec
      aload = 0.0
      do 8865 k=1,nsite
      aload = aload + aps(k)*w(k)*sigma0/1.0e15
8865  continue
      asec = real(n)
      write(unit=3,fmt=8866)asec,aload
8866  format(1pe15.6,1pe15.6)
8860  continue
      close (3)

      write(unit=*,fmt=8872)
8872  format('  enter output file name for final monolayers : '$)
      read(unit=*,fmt=78)fileout
      open(unit=3,file=fileout)
      write(unit=3,fmt=8873)nsec
8873  format(i5)
      do 8870 n=1,nsec
      aload = 0.0
      do 8875 k=1,nsite
      aload = aload + ap(n,k)*w(k)*sigma0/1.0e15
8875  continue
      asec = real(n)
      write(unit=3,fmt=8866)asec,aload
8870  continue
      close (3)
```

```
      write(unit=*,fmt=7100)
7100  format(' enter output file name for final occupation prob : '$)
      read(unit=*,fmt=7101)fileout
7101  format(a50)
      open(unit=3,file=fileout)
      write(unit=3,fmt=7102)nsite
7102  format(i5)
      write(unit=*,fmt=7103)
7103  format(' enter number of section : '$)
      read(unit=*,fmt=7102)kksec
      do 7500 kk=1,nsite
      write(unit=3,fmt=7104)at(kk),ap(kksec,kk)
7500  continue
7104  format(1pe15.6,1pe15.6)
      close(3)
      end if


      write(unit=*,fmt=2003)
2003    format(' enter 1 to write another file: '$)
      read(unit=*,fmt=2004)igo
2004    format(i3)
      if(igo.eq.1)go to 2000
3000    continue
      end

      subroutine derivs(x,y,dydx,x2,fr,aj)
      dimension y(1000),dydx(1000),fr(1000),aj(1000)
        common /deriv/ aa,bb,cc,dd,ee,alen0,nsec,al
      do 100 n=2,nsec-1
      xy = aa*(y(n-1)+y(n+1)-2.0*y(n))+bb*aj(n)
       xxy=((1.0/(alen0+real(n)*al)**2)-(1.0/(alen0+real(n+1)*al)**2))
      dydx(n) = xy + ee*xxy*fr(n)
100     continue
      xxy = ((1.0/(alen0)**2)-(1.0/(alen0+al)**2))
      dydx(1) = aa*(y(2)-y(1)) + bb*aj(1) -dd*y(1) + ee*xxy*fr(1)
        xy = 2.0*aa*(y(nsec-1)-y(nsec)) + bb*aj(nsec) - dd*y(nsec)
        xxy = (1.0/(alen0+real(nsec)*al)**2)
        xxy=xxy-(1.0/(alen0+real(nsec+1)*al)**2)
        dydx(nsec) = xy + ee*xxy*fr(nsec)
      return
      end



      SUBROUTINE odeint(ystart,nvar,x1,x2,eps,h1,hmin,nok,nbad,fr,aj)
      INTEGER nbad,nok,nvar,KMAXX,MAXSTP,NMAX
      REAL eps,h1,hmin,x1,x2,ystart(nvar),TINY
      dimension fr(1000),aj(1000)
      PARAMETER (MAXSTP=10000,NMAX=1000,KMAXX=8000,TINY=1.e-30)
      INTEGER i,kmax,kount,nstp
      REAL dxsav,h,hdid,hnext,x,xsav,dydx(NMAX),xp(KMAXX),y(NMAX),
     *yp(NMAX,KMAXX),yscal(NMAX)
      COMMON /path/ kmax,kount,dxsav,xp,yp
      common /deriv/ aa,bb,cc,dd,ee,alen0,nsec,al
      x=x1
      h=sign(h1,x2-x1)
```

```
      nok=0
      nbad=0
      kount=0
      do 11 i=1,nvar
      y(i)=ystart(i)
11    continue
      if (kmax.gt.0) xsav=x-2.*dxsav
      do 16 nstp=1,MAXSTP
        call derivs(x,y,dydx,x2,fr,aj)
      do 12 i=1,nvar
        yscal(i)=abs(y(i))+abs(h*dydx(i))+TINY
12      continue
      if(kmax.gt.0)then
        if(abs(x-xsav).gt.abs(dxsav)) then
          if(kount.lt.kmax-1)then
            kount=kount+1
            xp(kount)=x
            do 13 i=1,nvar
            yp(i,kount)=y(i)
13            continue
            xsav=x
          endif
        endif
      endif
      if((x+h-x2)*(x+h-x1).gt.0.) h=x2-x
      call rkqs(y,dydx,nvar,x,h,eps,yscal,hdid,hnext,x2,fr,aj)
      if(hdid.eq.h)then
        nok=nok+1
      else
        nbad=nbad+1
      endif
      if((x-x2)*(x2-x1).ge.0.)then
        do 14 i=1,nvar
          ystart(i)=y(i)
14        continue
        if(kmax.ne.0)then
          kount=kount+1
          xp(kount)=x
          do 15 i=1,nvar
            yp(i,kount)=y(i)
15          continue
        endif
        return
      endif
      if(abs(hnext).lt.hmin) pause
     *'stepsize smaller than minimum in odeint'
      h=hnext
16    continue
      pause 'too many steps in odeint'
      return
      END
C  (C) Copr. 1986-92 Numerical Recipes Software 7%W3.


      SUBROUTINE rkqs(y,dydx,n,x,htry,eps,yscal,hdid,hnext,x2,fr,aj)
      INTEGER n,NMAX
      REAL eps,hdid,hnext,htry,x,dydx(n),y(n),yscal(n)
```

```
      dimension fr(1000),aj(1000)
      PARAMETER (NMAX=1000)
CU    USES derivs,rkck
      INTEGER i
      REAL errmax,h,xnew,yerr(NMAX),ytemp(NMAX),SAFETY,PGROW,PSHRNK,
     *ERRCON
      PARAMETER (SAFETY=0.9,PGROW=-.2,PSHRNK=-.25,ERRCON=1.89e-4)
      common /deriv/ aa,bb,cc,dd,ee,alen0,nsec,al
      h=htry
1     call rkck(y,dydx,n,x,h,ytemp,yerr,x2,fr,aj)
      errmax=0.
      do 11 i=1,n
       errmax=max(errmax,abs(yerr(i)/yscal(i)))
11    continue
      errmax=errmax/eps
      if(errmax.gt.1.)then
      h=SAFETY*h*(errmax**PSHRNK)
      if(h.lt.0.1*h)then
        h=.1*h
      endif
      xnew=x+h
      if(xnew.eq.x)pause 'stepsize underflow in rkqs'
      goto 1
      else
      if(errmax.gt.ERRCON)then
        hnext=SAFETY*h*(errmax**PGROW)
      else
        hnext=5.*h
      endif
      hdid=h
      x=x+h
      do 12 i=1,n
        y(i)=ytemp(i)
12      continue
      return
      endif
      END
C  (C) Copr. 1986-92 Numerical Recipes Software 7%W3.


      SUBROUTINE rkck(y,dydx,n,x,h,yout,yerr,x2,fr,aj)
      INTEGER n,NMAX
      REAL h,x,dydx(n),y(n),yerr(n),yout(n)
      dimension fr(1000),aj(1000)
      PARAMETER (NMAX=1000)
CU    USES derivs
      common /deriv/ aa,bb,cc,dd,ee,alen0,nsec,al
      INTEGER i
      REAL ak2(NMAX),ak3(NMAX),ak4(NMAX),ak5(NMAX),ak6(NMAX),
     *ytemp(NMAX),A2,A3,A4,A5,A6,B21,B31,B32,B41,B42,B43,B51,B52,B53,
     *B54,B61,B62,B63,B64,B65,C1,C3,C4,C6,DC1,DC3,DC4,DC5,DC6
      PARAMETER (A2=.2,A3=.3,A4=.6,A5=1.,A6=.875,B21=.2,B31=3./40.,
     *B32=9./40.,B41=.3,B42=-.9,B43=1.2,B51=-11./54.,B52=2.5,
     *B53=-70./27.,B54=35./27.,B61=1631./55296.,B62=175./512.,
     *B63=575./13824.,B64=44275./110592.,B65=253./4096.,C1=37./378.,
     *C3=250./621.,C4=125./594.,C6=512./1771.,DC1=C1-2825./27648.,
     *DC3=C3-18575./48384.,DC4=C4-13525./55296.,DC5=-277./14336.,
```

```
     *DC6=C6-.25)
      do 11 i=1,n
      ytemp(i)=y(i)+B21*h*dydx(i)
11    continue
      call derivs(x+A2*h,ytemp,ak2,x2,fr,aj)
      do 12 i=1,n
      ytemp(i)=y(i)+h*(B31*dydx(i)+B32*ak2(i))
12    continue
      call derivs(x+A3*h,ytemp,ak3,x2,fr,aj)
      do 13 i=1,n
      ytemp(i)=y(i)+h*(B41*dydx(i)+B42*ak2(i)+B43*ak3(i))
13    continue
      call derivs(x+A4*h,ytemp,ak4,x2,fr,aj)
      do 14 i=1,n
      ytemp(i)=y(i)+h*(B51*dydx(i)+B52*ak2(i)+B53*ak3(i)+B54*ak4(i))
14    continue
      call derivs(x+A5*h,ytemp,ak5,x2,fr,aj)
      do 15 i=1,n
      ytemp(i)=y(i)+h*(B61*dydx(i)+B62*ak2(i)+B63*ak3(i)+B64*ak4(i)+
     *B65*ak5(i))
15    continue
      call derivs(x+A6*h,ytemp,ak6,x2,fr,aj)
      do 16 i=1,n
      yout(i)=y(i)+h*(C1*dydx(i)+C3*ak3(i)+C4*ak4(i)+C6*ak6(i))
16    continue
      do 17 i=1,n
      yerr(i)=h*(DC1*dydx(i)+DC3*ak3(i)+DC4*ak4(i)+DC5*ak5(i)+DC6*
     *ak6(i))
17    continue
      return
      END
C  (C) Copr. 1986-92 Numerical Recipes Software 7%W3.


      subroutine beamfraction(ksec,w,deltat,z,at,frr)
      common /deriv/ aa,bb,cc,dd,ee,alen0,nsec,al
      common /deriv1/ t0,sigma0,r,alpha,tau0,temp,nsite,v
      dimension w(1024),at(1024),z(1024)
      sum = 0.0
      do 200 ksite=1,nsite
      bc = (1.0-r)*at(ksite)/temp
      ac = (1.0+bc)*exp(-bc)
      sum = sum + w(ksite)*ac*(1.0-z(ksite))
200   continue
      frr = 1.0 -sum*alpha
c     write(unit=*,fmt=201)ksec,frr
c201  format('  sec = 'i5, ' fraction = '1pe15.6)
      return
      end


      subroutine probev(ksec,ystart,tstep,at,z)
      dimension z(1024),at(1024)
      dimension ystart(1000)
      common /deriv/ aa,bb,cc,dd,ee,alen0,nsec,al
      common /deriv1/ t0,sigma0,r,alpha,tau0,temp,nsite,v
      conv = 1.0/3.0e16
```

```
c     tc=1.0/aa
c     mm=int(tstep/tc)
c     dt = tstep/real(mm)
c     do 50 m=1,mm
      do 100 n=1,nsite
      rt = at(n)/temp
      tauemit = tau0*exp(rt)
      tg = (1.0 - r)*rt
      tt = (1.0 + tg)*exp(-tg)
      tauads1 = conv*4.0*sigma0/(alpha*ystart(ksec)*v*tt)
      xxy=((1.0/(alen0+real(ksec)*al)**2)
     &  -(1.0/(alen0+real(ksec+1)*al)**2))
      if(ksec.eq.1)xxy=((1.0/(alen0**2))-(1.0/(alen0+al)**2))
      if(ksec.eq.nsec)xxy=(1.0/(alen0+real(nsec)*al)**2)
     &  -(1.0/(alen0+real(nsec+1)*al)**2)
      tauads2 = conv*sigma0/(ee*xxy*tt)
      tauads = (tauads1*tauads2)/(tauads1+tauads2)
      tau = (tauads*tauemit)/(tauads+tauemit)
      if(z(n).lt.1.0e-7.and.tauemit.lt.1.0e-8)go to 100
      if(z(n).gt.9.999e-1.and.tauads.gt.1.0e8)go to 100
      zequil = tauemit/(tauemit+tauads)
      ax = exp(-tstep/tau)
      z(n) = z(n)*ax + zequil*(1.0-ax)
c     write(unit=*,fmt=901)ksec,n,m,z(n)
c901  format('  probev:ksec = 'i3,'n = 'i3,'m = 'i3,' z(n) = '1pe12.3)
c     ig = ig +1
c     if(ig.eq.50)then
c     ig = 0
c     read(unit=*,fmt=902)igg
c902  format(i5)
c     end if
100   continue
c50   continue
      return
      end
```

```
c *******btwaterdistribution2a.for   October 14, 2008
c Program estimates the water pressure above the surface when
c loaded with water from geometric transmission by the trap and
c steady state pumping by the trap
c
c This version allows the trap pressure to be reduced by powers
c of ten after a set time
c
c The program uses the Runge Kutta techniques from Numerical Recipes
c The calculation is made in cgs units with the pressure in torr
c
c  The outgassing is described by:
c
c      aj = outgassing rate torr liters/sec
c
c      ad(k) = the deposition of water on section k. given by
c      the pressure at the entrance to the trap multiplied by
c      the molecular speed times the solid angle of the section
c      subtended at the entrance to the trap normalized by 4*pi
c
c      a = tube radius
c
c      al = length of module/4*number of sections
c
c      f = pumping speed of the cryopump at one end of the module
c
c      v = thermal velocity of molecule at 300K
c
c      pt = water pressure at entrance to trap
c
c      alen0 = distance from front of trap to middle of the
c              first section of tube
c
c Difference equation to solve
c
c Section 1:  dp/dt = ((2*v*a)/(3*al**2))*(p(2)-p(1)) + (2/a)*aj
c end pt            - (f*p(1))/(pi*al*a**2)
c                   + pt*v*a**2/(4*al)*((1/(alen0)**2)-(1/(alen0+al)**2))
c
c
c
c Section k : dp/dt = ((2*v*a)/(3*al**2))*(p(k-1)-2*p(k) +p(k+1))
c                     + (2/a)*aj
c                     + pt*v*a**2/(4*al)*((1/(alen0+al*(k))**2)-
(1/(alen0+al(k+1))**2))
c
c
c last section dp/dt = ((2*v*a)/(3*al**2))*(p(k-1)-*p(k))
c                     + (2/a)*aj
c                    + pt*v*a**2/(4*al)*((1/(alen0+al*k)**2)-
(1/(alen0+al*(k+1))**2))
c                     -(f*p(k))/(pi*al*a**2)
c
c The program uses the Runge-Kutta algorithms given in Press et al
c Numerical Recipes 2
c
        use winteracter
```

```
      character fileout*50
        dimension p(1000,8000),ystart(1000),time(8000)
      common /path/ kmax,kount,dxsav,xp,yp
        common /deriv/ aa,bb,cc,dd,ee,alen0,nsec,al
      write(unit=*,fmt=601)
601     format(' enter # of sections :' $)
      read(unit=*,fmt=602)nsec
602     format(i6)
      write(unit=*,fmt=1)
1       format(' enter pump speed lit/sec, outgassing (tl/seccm^2): ' $)
      read(unit=*,fmt=2)f,aj
2       format(2e15.6)
      aj = aj*1000.0
      write(unit=*,fmt=605)
605     format(' enter amu of gas : '$)
      read(unit=*,fmt=4)amu
      write (unit=*,fmt=701)
701     format(' enter init p(torr) at n2trap,lgth (m) to fst sect :'$)
        read(unit=*,fmt=702)ptinit,alen0
702   format(2e15.6)
      alen0=alen0*100.0
4       format(e15.6)
      write(unit=*,fmt=703)
703   format(' enter # 10^n reduction p(torr) at n2trap, time(min) : '$)
      read(unit=*,fmt=7022)nptfin,ptftm
7022  format(i5,e15.6)
c enter the fixed parameters
c beam tube radius cm
      a = 62.0
c thermal speed of gas in cm/sec
      v = (5.263e4)*sqrt(18.0/amu)
c module length cm
      al0 = 4.0e5
c number of sections
      an = real(nsec)
c section length cm
      al = al0/an
c conversion pumping speed from liters/sec to cc/sec
      f = f*1000.0
      write(unit=*,fmt=9)
9       format(' enter starting pressure in torr : '$)
      read(unit=*,fmt=4)pstart
      aa = (2.0*v*a)/(3.0*al**2)
      bb = (2.0*aj)/a
      dd = f/(3.14159*al*a**2)
      ee = (ptinit*v*a**2)/(4.0*al)
      do 1000 kt = 1,nsec
      p(kt,1)=pstart
1000    continue
      write(unit=*,fmt=7)
7       format(' time(minutes)/step, #steps total, #calsteps/step : '$)
      read(unit=*,fmt=8)tstep1,nstep,intstep
8       format(e15.6,2i6)
        write(unit=*,fmt=451)
451     format('  enter 1 to write calc values : '$)
        read(unit=*,fmt=452)iwrt
452     format(i3)
```

```
c convert time to seconds
      tstep = (tstep1/real(intstep))*60.0
      x2=0.0
      nvar = nsec
      eps = 1.0e-4
c       h1 = (1.0e-3)*tstep
c       hmin = (1.0e-7)*tstep
      h1 = (1.0e-7)*tstep
      hmin=(1.0e-11)*tstep
      kill = 0
      do 100 k=1,nstep
      if(kill.ge.nptfin)go to 6666
      tmm = x2/60.0
      if(tmm.ge.ptftm)then
      kill = kill + 1
      ee = ee/(10.0**kill)
      end if
6666  do 300 j=1,intstep
      x1 = x2
      x2 = x1 + tstep
      if(j.eq.1.and.k.eq.1)then
      do 1002 n=1,nsec
      ystart(n)= pstart
1002    continue
      time(k)=x1/60.0
      p(nsec+1,k)=pstart
      end if
      call odeint(ystart,nvar,x1,x2,eps,h1,hmin,nok,nbad)
300     continue
      do 1003 n=1,nsec
      p(n,k+1)=ystart(n)
c        write(unit=*,fmt=420)time(k),n,p(n,k+1)
c420     format(' time ='1pe12.4,' n = 'i5,' pressure = '1pe12.4)
c        read(unit=*,fmt=421)junk
c421     format(i3)
1003    continue
      time(k+1)=x2/60.0
c calculate the average pressure
      sum = 0.0
      do 400 n=1,nsec
      sum  = sum + p(n,k+1)/an
400     continue
      p(nsec+1,k+1)=sum
        if(iwrt.eq.1)then
      write(unit=*,fmt=11)time(k),p(nsec+1,k)
        end if
11      format(' time (minutes) = '1pe15.6,  'avg p (torr) = '1pe15.6)
100     continue
2000    write(unit=*,fmt=77)
77      format('enter fileout: '$)
      read(unit=*,fmt=78)fileout
78      format(a50)
      open(unit=2,file=fileout)
      write(unit=*,fmt=2001)nsec+1,nsec+2
2001    format(' enter #segmt or 'i4,'=<p> or 'i4,'=p vs sec @ end:'$)
      read(unit=*,fmt=2002)ks
2002    format(i6)
```

```
        if(ks.eq.nsec+2)then
        write(unit=2,fmt=79)nsec
        do 2060 k=1,nsec
        xz = real(k)*al/100.0
        write(unit=2,fmt=80)xz,p(k,nstep+1)
2060    continue
        end if
      write(unit=2,fmt=79)nstep+1
79      format(i5)
      do 200 k=1,nstep+1
      write(unit=2,fmt=80)time(k),p(ks,k)
80      format(1pe15.6,1pe15.6)
200     continue
      close (2)
      write(unit=*,fmt=2003)
2003    format(' enter 1 to write another file: '$)
      read(unit=*,fmt=2004)igo
2004    format(i3)
      if(igo.eq.1)go to 2000
3000    continue
      end

      subroutine derivs(x,y,dydx,x2)
      dimension y(1000),dydx(1000)
        common /deriv/ aa,bb,cc,dd,ee,alen0,nsec,al
      do 100 n=2,nsec-1
      xy = aa*(y(n-1)+y(n+1)-2.0*y(n))+bb
        xxy=((1.0/(alen0+real(n)*al)**2)-(1.0/(alen0+real(n+1)*al)**2))
      dydx(n) = xy + ee*xxy
100     continue
      xxy = ((1.0/(alen0)**2)-(1.0/(alen0+al)**2))
      dydx(1) = aa*(y(2)-y(1)) + bb -dd*y(1) + ee*xxy
        xy = 2.0*aa*(y(nsec-1)-y(nsec)) + bb - dd*y(nsec)
        xxy = (1.0/(alen0+real(nsec)*al)**2)
        xxy=xxy-(1.0/(alen0+real(nsec+1)*al)**2)
        dydx(nsec) = xy + ee*xxy
      return
      end


      SUBROUTINE odeint(ystart,nvar,x1,x2,eps,h1,hmin,nok,nbad)
      INTEGER nbad,nok,nvar,KMAXX,MAXSTP,NMAX
      REAL eps,h1,hmin,x1,x2,ystart(nvar),TINY
      PARAMETER (MAXSTP=10000,NMAX=1000,KMAXX=8000,TINY=1.e-30)
      INTEGER i,kmax,kount,nstp
      REAL dxsav,h,hdid,hnext,x,xsav,dydx(NMAX),xp(KMAXX),y(NMAX),
     *yp(NMAX,KMAXX),yscal(NMAX)
      COMMON /path/ kmax,kount,dxsav,xp,yp
      common /deriv/ aa,bb,cc,dd,ee,alen0,nsec,al
      x=x1
      h=sign(h1,x2-x1)
      nok=0
      nbad=0
      kount=0
      do 11 i=1,nvar
      y(i)=ystart(i)
11    continue
```

```
      if (kmax.gt.0) xsav=x-2.*dxsav
      do 16 nstp=1,MAXSTP
        call derivs(x,y,dydx,x2)
      do 12 i=1,nvar
        yscal(i)=abs(y(i))+abs(h*dydx(i))+TINY
12      continue
      if(kmax.gt.0)then
        if(abs(x-xsav).gt.abs(dxsav)) then
          if(kount.lt.kmax-1)then
            kount=kount+1
            xp(kount)=x
            do 13 i=1,nvar
            yp(i,kount)=y(i)
13            continue
            xsav=x
          endif
        endif
      endif
      if((x+h-x2)*(x+h-x1).gt.0.) h=x2-x
      call rkqs(y,dydx,nvar,x,h,eps,yscal,hdid,hnext,x2)
      if(hdid.eq.h)then
        nok=nok+1
      else
        nbad=nbad+1
      endif
      if((x-x2)*(x2-x1).ge.0.)then
        do 14 i=1,nvar
          ystart(i)=y(i)
14        continue
        if(kmax.ne.0)then
          kount=kount+1
          xp(kount)=x
          do 15 i=1,nvar
            yp(i,kount)=y(i)
15          continue
        endif
        return
      endif
      if(abs(hnext).lt.hmin) pause
     *'stepsize smaller than minimum in odeint'
      h=hnext
16    continue
      pause 'too many steps in odeint'
      return
      END
C  (C) Copr. 1986-92 Numerical Recipes Software 7%W3.


      SUBROUTINE rkqs(y,dydx,n,x,htry,eps,yscal,hdid,hnext,x2)
      INTEGER n,NMAX
      REAL eps,hdid,hnext,htry,x,dydx(n),y(n),yscal(n)
      PARAMETER (NMAX=1000)
CU    USES derivs,rkck
      INTEGER i
      REAL errmax,h,xnew,yerr(NMAX),ytemp(NMAX),SAFETY,PGROW,PSHRNK,
     *ERRCON
      PARAMETER (SAFETY=0.9,PGROW=-.2,PSHRNK=-.25,ERRCON=1.89e-4)
```

```
      common /deriv/ aa,bb,cc,dd,ee,alen0,nsec,al
      h=htry
1     call rkck(y,dydx,n,x,h,ytemp,yerr,x2)
      errmax=0.
      do 11 i=1,n
       errmax=max(errmax,abs(yerr(i)/yscal(i)))
11    continue
      errmax=errmax/eps
      if(errmax.gt.1.)then
      h=SAFETY*h*(errmax**PSHRNK)
      if(h.lt.0.1*h)then
        h=.1*h
      endif
      xnew=x+h
      if(xnew.eq.x)pause 'stepsize underflow in rkqs'
      goto 1
      else
      if(errmax.gt.ERRCON)then
        hnext=SAFETY*h*(errmax**PGROW)
      else
        hnext=5.*h
      endif
      hdid=h
      x=x+h
      do 12 i=1,n
        y(i)=ytemp(i)
12      continue
      return
      endif
      END
C  (C) Copr. 1986-92 Numerical Recipes Software 7%W3.


      SUBROUTINE rkck(y,dydx,n,x,h,yout,yerr,x2)
      INTEGER n,NMAX
      REAL h,x,dydx(n),y(n),yerr(n),yout(n)
      PARAMETER (NMAX=1000)
CU    USES derivs
      common /deriv/ aa,bb,cc,dd,ee,alen0,nsec,al
      INTEGER i
      REAL ak2(NMAX),ak3(NMAX),ak4(NMAX),ak5(NMAX),ak6(NMAX),
     *ytemp(NMAX),A2,A3,A4,A5,A6,B21,B31,B32,B41,B42,B43,B51,B52,B53,
     *B54,B61,B62,B63,B64,B65,C1,C3,C4,C6,DC1,DC3,DC4,DC5,DC6
      PARAMETER (A2=.2,A3=.3,A4=.6,A5=1.,A6=.875,B21=.2,B31=3./40.,
     *B32=9./40.,B41=.3,B42=-.9,B43=1.2,B51=-11./54.,B52=2.5,
     *B53=-70./27.,B54=35./27.,B61=1631./55296.,B62=175./512.,
     *B63=575./13824.,B64=44275./110592.,B65=253./4096.,C1=37./378.,
     *C3=250./621.,C4=125./594.,C6=512./1771.,DC1=C1-2825./27648.,
     *DC3=C3-18575./48384.,DC4=C4-13525./55296.,DC5=-277./14336.,
     *DC6=C6-.25)
      do 11 i=1,n
      ytemp(i)=y(i)+B21*h*dydx(i)
11    continue
      call derivs(x+A2*h,ytemp,ak2,x2)
      do 12 i=1,n
      ytemp(i)=y(i)+h*(B31*dydx(i)+B32*ak2(i))
12    continue
```

```
      call derivs(x+A3*h,ytemp,ak3,x2)
      do 13 i=1,n
      ytemp(i)=y(i)+h*(B41*dydx(i)+B42*ak2(i)+B43*ak3(i))
13    continue
      call derivs(x+A4*h,ytemp,ak4,x2)
      do 14 i=1,n
      ytemp(i)=y(i)+h*(B51*dydx(i)+B52*ak2(i)+B53*ak3(i)+B54*ak4(i))
14    continue
      call derivs(x+A5*h,ytemp,ak5,x2)
      do 15 i=1,n
      ytemp(i)=y(i)+h*(B61*dydx(i)+B62*ak2(i)+B63*ak3(i)+B64*ak4(i)+
     *B65*ak5(i))
15    continue
      call derivs(x+A6*h,ytemp,ak6,x2)
      do 16 i=1,n
      yout(i)=y(i)+h*(C1*dydx(i)+C3*ak3(i)+C4*ak4(i)+C6*ak6(i))
16    continue
      do 17 i=1,n
      yerr(i)=h*(DC1*dydx(i)+DC3*ak3(i)+DC4*ak4(i)+DC5*ak5(i)+DC6*
     *ak6(i))
17    continue
      return
      END
C  (C) Copr. 1986-92 Numerical Recipes Software 7%W3.
```