Supplement of Geosci. Model Dev., 10, 3861–3888, 2017
https://doi.org/10.5194/gmd-10-3861-2017-supplement

Geoscientific Model Development
Open Access
EGU

Supplement of

Evaluation of five dry particle deposition parameterizations for incorporation into atmospheric transport models

Tanvir R. Khan and Judith A. Perlinger

Correspondence to: Judith A. Perlinger (jperl@mtu.edu)

Table S1. First order sensitivity index (*Si*) with bootstrap confidence intervals (*CI*) of the modeled V_d for the *Z01* parameterization.

		d_p = 0.001 µm			d_p = 0.01 µm			d_p = 0.1 µm			d_p = 1.0 µm			d_p = 10 µm		
LUC	Parameter	*Si*	*95% CI*		*Si*	*95% CI*		*Si*	*95% CI*		*Si*	*95% CI*		*Si*	*95% CI*	
	RH	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.001	-0.001	0.003	0.673	0.620	0.721
	ρ_P	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.002	0.000	0.003	0.019	0.016	0.022
	L_O	0.009	0.006	0.012	0.002	-0.001	0.006	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
	z_0	0.044	0.037	0.051	0.012	0.004	0.020	0.000	-0.001	0.001	0.000	0.000	0.000	0.000	-0.001	0.001
	u_*	0.918	0.884	0.948	0.959	0.896	1.021	0.981	0.949	1.011	0.975	0.938	1.009	0.245	0.230	0.260
	RH	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.001	0.001	0.000	0.002	0.598	0.493	0.690
	ρ_P	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.002	0.001	0.003	0.022	0.015	0.029
	L_O	0.260	0.231	0.289	0.124	0.103	0.147	0.005	0.000	0.011	0.000	0.000	0.000	0.009	0.003	0.016
	z_0	0.012	0.003	0.019	0.003	-0.002	0.008	0.000	-0.001	0.001	0.000	0.000	0.000	0.000	-0.001	0.001
	u_*	0.674	0.624	0.723	0.836	0.778	0.893	0.972	0.906	1.036	0.984	0.964	1.004	0.311	0.275	0.344
	RH	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	-0.001	0.002	0.844	0.782	0.903
	ρ_P	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.002	0.000	0.003	0.019	0.016	0.022
	z_0	0.300	0.283	0.316	0.178	0.166	0.189	0.012	0.009	0.014	0.000	0.000	0.001	0.002	0.000	0.003
	L_O	0.007	0.004	0.010	0.002	0.000	0.003	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
	u_*	0.671	0.643	0.698	0.791	0.762	0.817	0.972	0.947	0.996	0.979	0.950	1.009	0.092	0.084	0.098
	RH	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	-0.001	0.000	0.393	0.369	0.416
	ρ_P	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.002	0.000	0.003	0.008	0.006	0.009
	L_O	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
	u_*	0.995	0.963	1.028	0.995	0.962	1.027	0.994	0.959	1.027	0.983	0.960	1.005	0.559	0.543	0.575
	RH	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.001	-0.001	0.002	0.376	0.350	0.403
	ρ_P	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.004	0.002	0.005	0.010	0.008	0.012
	L_O	0.013	0.009	0.017	0.006	0.003	0.009	0.000	0.000	0.001	0.000	0.000	0.000	0.002	0.001	0.003
	u_*	0.926	0.893	0.959	0.963	0.928	0.996	0.982	0.956	1.007	0.978	0.949	1.005	0.551	0.531	0.569
	z_0	0.051	0.043	0.059	0.026	0.020	0.032	0.002	0.001	0.003	0.000	0.000	0.000	0.010	0.007	0.012

Table S2. First order sensitivity index with bootstrap confidence intervals (*CI*) of the modeled V_d for the *PZ10* parameterization.

		d_p = 0.001 µm			d_p = 0.01 µm			d_p = 0.1 µm			d_p = 1.0 µm			d_p = 10 µm		
LUC	Parameter	*Si*	*95% CI*		*Si*	*95% CI*		*Si*	*95% CI*		*Si*	*95% CI*		*Si*	*95% CI*	
	RH	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.014	0.010	0.018	0.347	0.320	0.372
	ρ_P	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.001	0.012	0.010	0.015
	L_O	0.009	0.006	0.012	0.001	0.000	0.003	-0.001	-0.002	0.000	0.001	0.000	0.003	0.002	0.000	0.003
	z_0	0.034	0.027	0.041	0.000	-0.001	0.001	0.000	0.000	0.000	0.000	-0.001	0.001	0.012	0.009	0.015
	u_*	0.780	0.751	0.809	0.050	0.043	0.058	0.000	-0.001	0.001	0.006	0.004	0.009	0.244	0.229	0.258
	LAI	0.103	0.091	0.115	0.693	0.664	0.722	0.553	0.536	0.572	0.379	0.360	0.397	0.040	0.035	0.045
	h	0.000	0.000	0.001	0.001	0.000	0.002	0.000	-0.001	0.001	0.000	-0.001	0.002	0.000	-0.001	0.000
	d	0.000	0.000	0.000	0.000	-0.001	0.001	0.001	0.000	0.001	0.000	-0.001	0.002	0.000	0.000	0.001
	U	0.032	0.025	0.038	0.225	0.210	0.242	0.408	0.393	0.424	0.511	0.489	0.532	0.228	0.215	0.240
	RH	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.009	0.007	0.011	0.198	0.179	0.215
	ρ_P	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.001	0.000	0.001	0.004	0.002	0.005
	L_O	0.462	0.440	0.483	0.493	0.468	0.514	0.375	0.357	0.393	0.358	0.347	0.368	0.350	0.334	0.366
	z_0	0.011	0.007	0.015	0.004	0.001	0.006	0.000	0.000	0.001	0.001	0.000	0.001	0.006	0.004	0.009
	u_*	0.492	0.466	0.517	0.164	0.148	0.180	0.008	0.005	0.011	0.052	0.047	0.057	0.372	0.354	0.390
	LAI	0.005	0.003	0.008	0.220	0.202	0.240	0.313	0.293	0.334	0.162	0.154	0.170	0.000	-0.001	0.001
	h	0.001	0.000	0.002	0.001	0.000	0.002	0.000	-0.001	0.001	0.001	0.000	0.001	0.000	0.000	0.001
	d	0.000	0.000	0.000	-0.001	-0.002	0.000	0.000	-0.001	0.001	0.001	0.000	0.001	0.000	0.000	0.000
	U	0.001	-0.001	0.002	0.069	0.058	0.078	0.217	0.199	0.231	0.277	0.266	0.288	0.004	0.002	0.006
	RH	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.002	0.001	0.003	0.280	0.257	0.302
	ρ_P	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.006	0.004	0.007
	L_O	0.501	0.484	0.518	0.458	0.442	0.472	0.401	0.387	0.415	0.453	0.441	0.464	0.339	0.325	0.352
	z_0	0.004	0.002	0.005	0.001	0.000	0.002	0.000	0.000	0.000	0.001	0.000	0.001	0.002	0.001	0.004
	u_*	0.390	0.372	0.407	0.111	0.102	0.119	0.011	0.008	0.014	0.081	0.075	0.087	0.237	0.225	0.250
	LAI	0.016	0.012	0.019	0.252	0.240	0.265	0.281	0.269	0.293	0.192	0.183	0.199	0.005	0.003	0.007
	h	0.000	0.000	0.001	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.001
	d	0.000	0.000	0.001	0.001	0.000	0.002	0.002	0.001	0.003	0.001	0.000	0.002	0.000	0.000	0.000
	U	0.001	0.000	0.002	0.031	0.026	0.035	0.122	0.114	0.130	0.106	0.100	0.112	0.005	0.003	0.007
	RH	0.000	0.000	0.000	0.000	0.000	0.000	-0.001	-0.002	0.000	0.695	0.620	0.770	0.956	0.877	1.025
	ρ_P	0.000	0.000	0.000	0.000	0.000	0.000	0.013	0.003	0.023	0.241	0.194	0.285	0.016	0.013	0.019
	L_O	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
	u_*	0.994	0.959	1.027	0.988	0.942	1.032	0.878	0.640	1.121	-0.001	-0.003	0.002	0.000	0.000	0.000
	RH	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.703	0.654	0.750	0.974	0.912	1.034
	ρ_P	0.000	0.000	0.000	0.000	0.000	0.000	0.004	0.002	0.005	0.270	0.244	0.296	0.015	0.013	0.016
	L_O	0.002	0.001	0.003	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
	u_*	0.962	0.931	0.993	0.969	0.938	1.000	0.956	0.913	0.998	0.000	-0.001	0.002	0.000	0.000	0.000
	z_0	0.004	0.002	0.006	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000

Table S3. First order sensitivity index (*Si*) with bootstrap confidence intervals (*CI*) of the modeled V_d for the *KS12* parameterization.

		d_p = 0.001 µm			d_p = 0.01 µm			d_p = 0.1 µm			d_p = 1.0 µm			d_p = 10 µm		
LUC	Parameter	*Si*	*95% CI*		*Si*	*95% CI*		*Si*	*95% CI*		*Si*	*95% CI*		*Si*	*95% CI*	
	RH	0.000	0.000	0.000	0.000	0.000	0.000	0.000	-0.002	0.003	0.777	0.731	0.822	0.816	0.709	0.916
	ρ_P	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.039	0.031	0.047	0.000	0.000	0.001
	u_*	0.904	0.832	0.976	0.904	0.832	0.976	0.902	0.829	0.974	0.122	0.106	0.138	0.009	0.005	0.012
	LAI	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.002	-0.001	0.003
	RH	0.000	0.000	0.000	0.000	0.000	0.000	0.000	-0.001	0.000	0.139	0.126	0.153	0.783	0.709	0.849
	ρ_P	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.004	0.002	0.007	0.002	0.001	0.003
	u_*	0.911	0.823	0.995	0.911	0.823	0.995	0.943	0.879	1.005	0.829	0.798	0.860	0.032	0.029	0.035
	LAI	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.005	0.002	0.007
	RH	0.000	0.000	0.000	0.000	0.000	0.000	0.000	-0.001	0.000	0.629	0.571	0.686	0.934	0.856	1.004
	ρ_P	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.243	0.206	0.277	0.012	0.010	0.014
	u_*	0.944	0.880	1.005	0.944	0.880	1.005	0.967	0.929	1.005	0.054	0.037	0.072	0.002	0.000	0.004
	LAI	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	-0.001	0.001
	RH	0.000	0.000	0.000	0.000	0.000	0.000	0.001	-0.001	0.002	0.302	0.287	0.316	0.848	0.789	0.899
	ρ_P	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.001	0.000	0.001	0.002	0.001	0.003
	u_*	0.905	0.840	0.972	0.905	0.840	0.972	0.902	0.836	0.970	0.614	0.595	0.632	0.022	0.020	0.024

Table S4. First order sensitivity index (*Si*) with bootstrap confidence intervals (*CI*) of the modeled V_d for the *ZH14* parameterization.

		d_p = 0.001 µm			d_p = 0.01 µm			d_p = 0.1 µm			d_p = 1.0 µm			d_p = 10 µm		
LUC	Parameter	*Si*	*95% CI*		*Si*	*95% CI*		*Si*	*95% CI*		*Si*	*95% CI*		*Si*	*95% CI*	
	RH	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.001	0.000	0.002	0.940	0.847	1.014
	ρ_P	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.001	0.000	0.001	0.006	0.004	0.008
	u_*	1.002	0.973	1.033	1.002	0.973	1.033	1.002	0.973	1.033	1.000	0.967	1.032	0.000	0.000	0.000
	L	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
	z_0	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.001	0.000	0.000	0.000
	RH	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.001	0.420	0.382	0.456
	ρ_P	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.006	0.004	0.007
	u_*	0.988	0.960	1.016	0.988	0.960	1.016	0.988	0.960	1.016	0.971	0.947	0.994	0.341	0.325	0.356
	L	0.002	0.001	0.003	0.002	0.000	0.003	0.002	0.000	0.003	0.004	0.002	0.005	0.141	0.131	0.150
	z_0	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.012	0.009	0.014
	RH	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.001	0.634	0.585	0.675
	ρ_P	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.001	0.004	0.003	0.006
	u_*	0.988	0.962	1.011	0.988	0.962	1.011	0.988	0.962	1.011	0.987	0.960	1.012	0.185	0.174	0.195
	L	0.006	0.004	0.008	0.006	0.004	0.008	0.006	0.004	0.008	0.006	0.004	0.008	0.015	0.012	0.018
	z_0	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
	LAI													0.053	0.048	0.057
	RH	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.001	0.465	0.429	0.499
	ρ_P	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.001	0.008	0.005	0.010
	L	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
	u_*	0.981	0.954	1.010	0.981	0.954	1.010	0.981	0.953	1.010	0.980	0.952	1.010	0.499	0.480	0.517
	RH	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.001	0.407	0.378	0.433
	ρ_P	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.005	0.003	0.007
	L	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.004	0.002	0.005
	u_*	0.984	0.959	1.008	0.984	0.959	1.008	0.984	0.959	1.008	0.986	0.964	1.006	0.548	0.531	0.564
	z_0	0.001	0.000	0.001	0.001	0.000	0.001	0.001	0.000	0.001	0.000	0.000	0.001	0.014	0.011	0.017

Table S5. First order sensitivity index (*Si*) with bootstrap confidence intervals (*CI*) of the modeled V_d for the *ZS14* parameterization.

		d_p = 0.001 µm			d_p = 0.01 µm			d_p = 0.1 µm			d_p = 1.0 µm			d_p = 10 µm		
LUC	Parameter	*Si*	*95% CI*		*Si*	*95% CI*		*Si*	*95% CI*		*Si*	*95% CI*		*Si*	*95% CI*	
	RH	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.006	0.004	0.008	0.071	0.060	0.085
	ρ_P	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	-0.001	0.002
	u_*	0.436	0.410	0.463	0.347	0.324	0.369	0.576	0.547	0.603	0.611	0.580	0.640	0.750	0.707	0.789
	U	0.464	0.430	0.498	0.607	0.568	0.643	0.306	0.282	0.329	0.275	0.252	0.297	0.145	0.127	0.163
	RH	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.003	0.001	0.004	0.001	0.000	0.003
	ρ_P	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.001
	u_*	0.991	0.958	1.023	0.966	0.935	0.997	0.989	0.963	1.014	0.988	0.961	1.015	0.989	0.956	1.020
	U	0.000	0.000	0.001	0.027	0.022	0.032	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	3.180

Codes for evaluation of model accuracy using Zhang et al. (2001) parameterization

```
#Accuracy Evaluation: Grass
#Dry deposition parameterization by Zhang et al. (2001)
attach(Allen_etal_1991)                          # Use separate text file to feed V1, V2,… for different studies
C1 = 0.2789
C1 = 0.2789
C2 = 3.115
C3 = 5.145*10^-11
RH = 90/100
dp_i = 0.48
dp_d = dp_i*10^-6
rd = dp_d/2
r_w = {(C1*rd^C2)/(C3*rd^C4-log10(RH))+rd^3}^(1/3)
dp = r_w*2
#Correction factor, C
k_B = 1.38*10^-23
Temp_1 = V1
Temp = 273.15+V1
P = 101325
d_air = 3.72*10^-10
lambda = (k_B*Temp)/(sqrt(2)*3.1416*P*d_air^2)
C = 1+(2*lambda/dp)*(1.257+0.4*exp(-0.55*dp/lambda))
dyn.vis = ((5*10^-8)*Temp)+4*10^-6
rho = 2000
Vg = (rho*(dp)^2*9.81*C)/(18*dyn.vis)
# Compute aerodynamic resistance Ra:
z = 2
L = V4
x = z/L
# Compute shi_H (stability function)
shi_H.1 = 2*log(0.5*{1+(1-16*x)^0.5})
shi_H.2 = -5*x
shi_H =ifelse(x <= 0, shi_H.1 , shi_H.2)
zR = 3.5
z0 = 0.01
u_star = V2                                          #Use from input text file
k_c = 0.41
Ra = (log(zR/z0)-shi_H)/(k_c*u_star)
# Compute surface resistance Rs:
e_0 = 3
R1 = 1
# Compute E_B (collection efficiency from Brownian diffusion)
kin.vis = ((9*10^-8)*Temp)+10^-5
gamma = 0.54
D = (C*k_B*Temp)/(3*3.1416*dyn.vis*dp)
Sc = (kin.vis/D)
E_B = Sc^(-gamma)
# Compute E_IM (collection efficiency from impaction)
alpha = 1.2
beta =  2
A = 2/1000
St = (Vg*u_star)/(9.81*A)
E_IM = {St/(alpha+St)}^beta
# Compute E_IN (collection efficiency from interception)
E_IN = 0.5*(dp/A)^2
Rs = 1/{(e_0*u_star)*(E_B+E_IM+E_IN)*R1}
# Compute Dry deposition velocity

#Accuracy Evaluation: Coniferous forest
#Dry deposition parameterization by Zhang et al. (2001)
attach(Rannik_etal_2000)                    # Use separate text file to feed V1, V2,… for different studies
C1 = 0.2789
C2 = 3.115
C3 = 5.145*10^-11
C4 = -1.399
RH = 90/100
dp_i = V1
dp_d = dp_i*10^-6
rd = dp_d/2
r_w = {(C1*rd^C2)/(C3*rd^C4-log10(RH))+rd^3}^(1/3)
dp = r_w*2
#Correction factor, C
k_B = 1.38*10^-23
```

```
Temp = 273.15+25
P = 101325
d_air = 3.72*10^-10
lambda = (k_B*Temp)/(sqrt(2)*3.1416*P*d_air^2)
C = 1+(2*lambda/dp)*(1.257+0.4*exp(-0.55*dp/lambda))
dyn.vis = 1.891*10^-5
rho = 1500
Vg = (rho*(dp)^2*9.81*C)/(18*dyn.vis)
# Compute aerodynamic resistance Ra:
z = 23.7
L = 200
x = z/L
shi_H2 = -5*x
zR = 26
z0 = 1.2
u_star = V2
k_c = 0.41
Ra = (log(zR/z0)-shi_H2)/(k_c*u_star)
e_0 = 3
R1 =1
kin.vis = 1.683*10^-5
gamma = 0.56
D = (C*k_B*Temp)/(3*3.1416*dyn.vis*dp)
Sc = (kin.vis/D)
E_B = Sc^(-gamma)
# Compute E_IM (collection efficiency from impaction)
alpha = 1.0
beta =  2
A = 5/1000
St = (Vg*u_star)/(9.81*A)
E_IM = {St/(alpha+St)}^beta
# Compute E_IN (collection efficiency from interception)
E_IN = 0.5*(dp/A)^2
Rs = 1/{(e_0*u_star)*(E_B+E_IM+E_IN)*R1}
# Compute Dry deposition velocity
Vd =  Vg+(1/(Ra+Rs));Vd   #unit: m/s

#Accuracy Evaluation: Deciduous forest
#Dry deposition parameterization by Zhang et al. (2001)
attach(Wesely_etal_1983)                          # Use separate text file to feed V1, V2,… for different studies
C1 = 0.2789
C2 = 3.115
C3 = 5.145*10^-11
C4 = -1.399
RH = 95/100
dp_i = 0.4
dp_d = dp_i*10^-6
rd = dp_d/2
r_w = {(C1*rd^C2)/(C3*rd^C4-log10(RH))+rd^3}^(1/3)
dp = r_w*2
#Correction factor, C
k_B = 1.38*10^-23
Temp = V1
P = 101325
d_air = 3.72*10^-10
lambda = (k_B*Temp)/(sqrt(2)*3.1416*P*d_air^2)
C = 1+(2*lambda/dp)*(1.257+0.4*exp(-0.55*dp/lambda))
dyn.vis = V2
rho = 2000
Vg = (rho*(dp)^2*9.81*C)/(18*dyn.vis)
# Compute aerodynamic resistance Ra:
z = 39
L = V6
x = z/L
shi_H2 = -5*x
zR = 56
z0 = 1.6
u_star = V4
k_c = 0.41
Ra = (log(zR/z0)-shi_H1)/(k_c*u_star)
# Compute surface resistance Rs:
e_0 = 3
R1 =1
kin.vis = V3
```

```
gamma = 0.56
D = (C*k_B*Temp)/(3*3.1416*dyn.vis*dp)
Sc = (kin.vis/D)
E_B = Sc^(-gamma)
# Compute E_IM (collection efficiency from impaction)
alpha = 0.80
beta =  2
A = 5/1000
St = (Vg*u_star)/(9.81*A)
E_IM = {St/(alpha+St)}^beta
# Compute E_IN (collection efficiency from interception)
E_IN = 0.5*(dp/A)^2
Rs = 1/{(e_0*u_star)*(E_B+E_IM+E_IN)*R1}
# Compute Dry deposition velocity
Vd =  Vg+(1/(Ra+Rs));Vd


#Accuracy Evaluation: Water
#Dry deposition parameterization by Zhang et al. (2001)
attach(Caffrey_etal_1998)
C1 = 0.2789
C2 = 3.115
C3 = 5.415*10^-11
C4 = -1.399
RH = 79/100
dp_i = 0.005
dp_d = dp_i*10^-6
rd = dp_d/2
r_w = {(C1*rd^C2)/(C3*rd^C4-log10(RH))+rd^3}^(1/3)
dp = r_w*2
#Correction factor, C
k_B = 1.38*10^-23
Temp = 273.15+22
P = 101325
d_air = 3.72*10^-10
lambda = (k_B*Temp)/(sqrt(2)*3.1416*P*d_air^2)
C = 1+(2*lambda/dp)*(1.257+0.4*exp(-0.55*dp/lambda))
dyn.vis = ((5*10^-8)*Temp)+4*10^-6
rho = 2000
Vg = (rho*(dp)^2*9.81*C)/(18*dyn.vis)
# Compute aerodynamic resistance Ra:
z = 8/100
L = 50
x = z/L
# Compute shi_H (stability function)
shi_H.1 = 2*log(0.5*{1+(1-16*x)^0.5})
shi_H.2 = -5*x
shi_H =ifelse(x <= 0, shi_H.1 , shi_H.2)
zR = 5
u_star = 13.5/100
z0_1 = 0.021*(u_star)^3.32
z0_2 = 0.00098*(u_star)^1.65
z0 = ifelse(u_star<= 0.16, z0_1, z0_2)
k_c = 0.41
Ra = (log(zR/z0)-shi_H)/(k_c*u_star)  # m/s
# Compute surface resistance Rs:
e_0 = 3
R1 = 1
kin.vis = ((9*10^-8)*Temp)+10^-5
gamma = 0.50
D = (C*k_B*Temp)/(3*3.1416*dyn.vis*dp)
Sc = (kin.vis/D)
E_B = Sc^(-gamma)
alpha = 100
beta =  2
A = 2/1000
St = (Vg*u_star^2)/(kin.vis)
E_IM = {St/(alpha+St)}^beta
Rs = 1/{(e_0*u_star)*(E_B+E_IM)*R1}
# Compute Dry deposition velocity
Vd =  Vg+(1/(Ra+Rs));Vd
```

```
#Accuracy Evaluation: Ice/snow
#Dry deposition parameterization by Zhang et al. (2001)
attach(Ibrahim_et_al_1983)
C1 = 0.2789
C2 = 3.115
C3 = 5.415*10^-11
C4 = -1.399
RH = 60/100
dp_i = c(0.22, 0.73)
dp_d = dp_i*10^-6
rd = dp_d/2
r_w = {(C1*rd^C2)/(C3*rd^C4-log10(RH))+rd^3}^(1/3)
dp = r_w*2
#Correction factor, C
k_B = 1.38*10^-23
Temp = 273.15+25
P = 101325
d_air = 3.72*10^-10
lambda = (k_B*Temp)/(sqrt(2)*3.1416*P*d_air^2)
C = 1+(2*lambda/dp)*(1.257+0.4*exp(-0.55*dp/lambda))
dyn.vis = ((5*10^-8)*Temp)+4*10^-6
rho = 1500
Vg = (rho*(dp)^2*9.81*C)/(18*dyn.vis)
# Compute aerodynamic resistance Ra:
#z =
#L =
#x = z/L
x = 0.2
# Compute shi_H (stability function)
shi_H.1 = 2*log(0.5*{1+(1-16*x)^0.5})
shi_H.2 = -5*x
shi_H =ifelse(x <= 0, shi_H.1 , shi_H.2)
zR = 5
u_star = 0.12
z0 = 0.1/100
k_c = 0.41
Ra = (log(zR/z0)-shi_H)/(k_c*u_star)
# Compute surface resistance Rs:
e_0 = 3
R1 = 1
kin.vis = ((9*10^-8)*Temp)+10^-5
gamma = 0.54
D = (C*k_B*Temp)/(3*3.1416*dyn.vis*dp)
Sc = (kin.vis/D)
E_B = Sc^(-gamma)
alpha = 50
beta =  2
A = 2/1000
St = (Vg*u_star^2)/(kin.vis)
E_IM = {St/(alpha+St)}^beta
Rs = 1/{(e_0*u_star)*(E_B+E_IM)*R1}
# Compute Dry deposition velocity
Vd =  Vg+(1/(Ra+Rs));Vd
```

Codes for evaluation of model accuracy using Petroff and Zhang (2010) parameterization

```
#Accuracy Evaluation: Grass
#Dry deposition velocity parameterization by Petroff and Zhang (2010)
attach(Allen_etal_1991)
C1 = 0.2789
C2 = 3.115
C3 = 5.145*10^-11
C4 = -1.399
RH = 90/100
dp_i = 0.48
dp_d = dp_i*10^-6
rd = dp_d/2
r_w = {(C1*rd^C2)/(C3*rd^C4-log10(RH))+rd^3}^(1/3)
dp = r_w*2
#Correction factor, C
k_B = 1.38*10^-23
Temp = 273.15+25
P = 101325
```

```
d_air = 3.7208*10^-10
lambda = (k_B*Temp)/(sqrt(2)*3.1416*P*d_air^2)
C = 1+(2*lambda/dp)*(1.257+0.4*exp(-0.55*dp/lambda))
dyn.vis = ((5*10^-8)*Temp)+4*10^-6
rho = 1500
Tau = (rho*(dp)^2*C)/(18*dyn.vis)
Ws = Tau*9.81
Vphor = 0
Vdrift = Ws+Vphor
# Compute aerodynamic resistance (Ra):
z = 2
L = 200
x = z/L
# Compute stability function (shi_H)
shi_H.1 = 2*log(0.5*{1+(1-16*x)^0.5})
shi_H.2 = -5*x
shi_H = ifelse(x <= 0, shi_H.1 , shi_H.2)
zR = 3.5
z0 = 0.01
u_star = 0.5
k_c = 0.41
Ra = (log(zR/z0)-shi_H)/(k_c*u_star)
# Compute surface resistance (Rs)
kin.vis = ((9*10^-8)*Temp)+10^-5
D = (C*k_B*Temp)/(3*3.1416*dyn.vis*dp)
Sc = (kin.vis/D)
FSc = (Sc^(1/3))/2.9
Egb = (Sc^(-2/3)/14.5)*{1/6*log(1+FSc)^2/(1-FSc+FSc^2)+1/sqrt(3)*atan((2*FSc-
1)/sqrt(3))+3.1416/6*sqrt(3)}^-1
cd = 1/6
kx = 0.216
LAI = 4
h =  0.07
d  = 0.04
phi_H.1 = (1-16*x)^(-0.5)
phi_H.2 = 1+5*x
phi_H = ifelse(x<=0, phi_H.1, phi_H.2)
phi_M.1 = (1-16*x)^(-0.25)
phi_M.2 = 1+5*x
phi_M = ifelse(x<=0, phi_M.1, phi_M.2)
lmp = (0.41*(z-d))/(phi_H*(z-d)/abs(L))
lmp_h = (0.41*(h-d))/(phi_M*(h-d)/abs(L))
alphaPZ = {(kx*LAI)/(12*k_c^2*(1-d/h)^2)}^(1/3)*phi_M^(2/3)*{(h-d)/abs(L)}
C_IT = 0.056
Tau_phplus.1 = (Tau*u_star^2)/kin.vis
Tau_phplus.2 = C_IT
Tau_phplus = ifelse(Tau_phplus.1<20,Tau_phplus.1, Tau_phplus.2)
E_t.1 = 2.5*10^-3*C_IT*(Tau_phplus)^2
E_t.2 = C_IT
E_t = ifelse(Tau_phplus.1<20, E_t.1, E_t.2)
u_starf = u_star*exp(-alphaPZ)
Tau_phplus.f1 = (Tau*u_star^2)/kin.vis
Tau_phplus.f2 = 0.14
Tau_phplusf = ifelse(Tau_phplus.f1<20,Tau_phplus.f1, Tau_phplus.f2)
E_gt1 = 2.5*10^-3*0.14*(Tau_phplusf)^2
E_gt2 = 0.14
Egt = ifelse(Tau_phplus.f1<20, E_gt1, E_gt2)
Eg = Egb + Egt
# Compute Qg (non-dimensional number)
Qg = Eg*h/lmp_h
# Compute Q
U_z = 2
U_h = U_z/(exp(alphaPZ*(z/h-1)))
#Compute E_B (Brownian diffusion)
L_obs = 0.01
C_B = 0.996
Re_h = (U_h*L_obs)/(kin.vis)
E_B = C_B*(Sc^(-2/3))*(Re_h^(-1/2))
#Compute E_IN (Interception)
C_IN = 0.162
E_IN = C_IN*(dp/L_obs)
#Compute E_IM (Impaction)
C_IM = 0.081
beta_IM = 0.47
```

```
St_h = (Tau*U_h)/L_obs
E_IM = C_IM*(St_h/(St_h+beta_IM))^2
E_T = (U_h/u_star)*(E_B+E_IN+E_IM)+E_t
Q  = LAI*E_T*h/(lmp_h)
# Compute etaPZ
etaPZ = (alphaPZ^2/4+Q)^0.5
Vds = u_star*Eg*{(1+Q/Qg-alphaPZ/2)*tanh(etaPZ)/etaPZ}/{(1+Q+alphaPZ/2)*tanh(etaPZ)/etaPZ}
Vd = Vdrift+1/(Ra+1/Vds);Vd


#Accuracy Evaluation: Coniferous forest
#Dry deposition velocity parameterization by Petroff and Zhang (2010)
attach(Rannik_etal_2000)
C1 = 0.2789
C2 = 3.115
C3 = 5.145*10^-11
C4 = -1.399
RH= 0.90
dp_i = V1
dp_d = dp_i*10^-6
rd = dp_d/2
r_w = {(C1*rd^C2)/(C3*rd^C4-log10(RH))+rd^3}^(1/3)
dp = r_w*2
#Correction factor, C
k_B = 1.38*10^-23
Temp =  273.15+25
P = 101325
d_air = 3.7208*10^-10
lambda = (k_B*Temp)/(sqrt(2)*3.1416*P*d_air^2)
C = 1+(2*lambda/dp)*(1.257+0.4*exp(-0.55*dp/lambda))
dyn.vis = 1.891*10^-5
rho = 1500
Tau = (rho*(dp)^2*C)/(18*dyn.vis)
Ws = Tau*9.81
Vphor = 0
Vdrift = Ws+Vphor
# Compute aerodynamic resistance (Ra):
z = 23.7
L = 200
x = z/L
# Compute stability function (shi_H)
shi_H.1 = 2*log(0.5*{1+(1-16*x)^0.5})
shi_H.2 = -5*x
shi_H =ifelse(x <= 0, shi_H.1 , shi_H.2)
zR = 26
z0 = 1.2
u_star = V2
k_c = 0.41
Ra = (log(zR/z0)-shi_H)/(k_c*u_star)
kin.vis = 1.683*10^-5
D = (C*k_B*Temp)/(3*3.1416*dyn.vis*dp)
Sc = (kin.vis/D)
FSc = (Sc^(1/3))/2.9
Egb = (Sc^(-2/3)/14.5)*{1/6*log(1+FSc)^2/(1-FSc+FSc^2)+1/sqrt(3)*atan((2*FSc-1)/sqrt(3))
        +3.1416/6*sqrt(3)}^-1
cd = 1/6
kx = 0.216
LAI = 6
h =  13
d  = 9.75
phi_H.1 = (1-16*x)^(-0.5)
phi_H.2 = 1+5*x
phi_H = ifelse(x<=0, phi_H.1, phi_H.2)
phi_M.1 = (1-16*x)^(-0.25)
phi_M.2 = 1+5*x
phi_M = ifelse(x<=0, phi_M.1, phi_M.2)
lmp = (0.41*(z-d))/(phi_H*(z-d)/abs(L))
lmp_h = (0.41*(h-d))/(phi_M*(h-d)/abs(L))
alphaPZ = {(kx*LAI)/(12*k_c^2*(1-d/h)^2)}^(1/3)*phi_M^(2/3)*{(h-d)/abs(L)}
C_IT = 0
Tau_phplus.1 = (Tau*u_star^2)/kin.vis
Tau_phplus.2 = C_IT
Tau_phplus = ifelse(Tau_phplus.1<20,Tau_phplus.1, Tau_phplus.2)
E_t.1 = 2.5*10^-3*C_IT*(Tau_phplus)^2
```

```
E_t.2 = C_IT
E_t = ifelse(Tau_phplus.1<20, E_t.1, E_t.2)
u_starf = u_star*exp(-alphaPZ)
Tau_phplus.f1 = (Tau*u_star^2)/kin.vis
Tau_phplus.f2 = 0.14
Tau_phplusf = ifelse(Tau_phplus.f1<20,Tau_phplus.f1, Tau_phplus.f2)
E_gt1 = 2.5*10^-3*0.14*(Tau_phplusf)^2
E_gt2 = 0.14
Egt = ifelse(Tau_phplus.f1<20, E_gt1, E_gt2)
Eg = Egb + Egt
# Compute Qg (non-dimensional number)
Qg = Eg*h/lmp_h
# Compute Q
U_z = V3
U_h = U_z/(exp(alphaPZ*(z/h-1)))
L_obs = 0.15
C_B = 0.887
Re_h = (U_h*L_obs)/(kin.vis)
E_B = C_B*(Sc^(-2/3))*(Re_h^(-1/2))
C_IN = 0.810
E_IN = C_IN*(dp/L_obs)
#Compute E_IM (Impaction)
C_IM = 0.162
beta_IM = 0.60
St_h = (Tau*U_h)/L_obs
E_IM = C_IM*(St_h/(St_h+beta_IM))^2
E_T = (U_h/u_star)*(E_B+E_IN+E_IM)+E_t
Q  = LAI*E_T*h/(lmp_h)
# Compute etaPZ
etaPZ = (alphaPZ^2/4+Q)^0.5
Vds = u_star*Eg*{(1+Q/Qg-alphaPZ/2)*tanh(etaPZ)/etaPZ}/{(1+Q+alphaPZ/2)*tanh(etaPZ)/etaPZ}
Vd = Vdrift+1/(Ra+1/Vds);Vd


#Accuracy Evaluation: Deciduous forest
#Dry deposition parameterization by Petroff and Zhang (2010)
attach(Wesely_etal_1983)
C1 = 0.2789
C2 = 3.115
C3 = 5.145*10^-11
C4 = -1.399
RH= 95/100
dp_i = 0.4
dp_d = dp_i*10^-6
rd = dp_d/2
r_w = {(C1*rd^C2)/(C3*rd^C4-log10(RH))+rd^3}^(1/3)
dp = r_w*2
k_B = 1.38*10^-23
Temp = V1
P = 101325
d_air = 3.7208*10^-10
lambda = (k_B*Temp)/(sqrt(2)*3.1416*P*d_air^2)
C = 1+(2*lambda/dp)*(1.257+0.4*exp(-0.55*dp/lambda))
dyn.vis = V2
rho = 2000
Tau = (rho*(dp)^2*C)/(18*dyn.vis)
Ws = Tau*9.81
Vphor = 0
Vdrift = Ws+Vphor
# Compute aerodynamic resistance (Ra):
z = 39
L = (1*V6)
x = z/L
# Compute stability function (shi_H)
shi_H.1 = 2*log(0.5*{1+(1-16*x)^0.5})
shi_H.2 = -5*x
shi_H =ifelse(x <= 0, shi_H.1 , shi_H.2)
zR = 56
z0 = 1.6
u_star = V4
k_c = 0.41
Ra = (log(zR/z0)-shi_H)/(k_c*u_star)
kin.vis = V3
D = (C*k_B*Temp)/(3*3.1416*dyn.vis*dp)
```

```
Sc = (kin.vis/D)
FSc = (Sc^(1/3))/2.9
Egb = (Sc^(-2/3)/14.5)*{1/6*log(1+FSc)^2/(1-FSc+FSc^2)+1/sqrt(3)*atan((2*FSc-1)/sqrt(3))
        +3.1416/6*sqrt(3)}^-1
cd = 1/6
kx = 0.216
LAI = 0.2
h =  28
d  = 21
phi_H.1 = (1-16*x)^(-0.5)
phi_H.2 = 1+5*x
phi_H = ifelse(x<=0, phi_H.1, phi_H.2)
phi_M.1 = (1-16*x)^(-0.25)
phi_M.2 = 1+5*x
phi_M = ifelse(x<=0, phi_M.1, phi_M.2)
lmp = (0.41*(z-d))/(phi_H*(z-d)/abs(L))
lmp_h = (0.41*(h-d))/(phi_M*(h-d)/abs(L))
alphaPZ = {(kx*LAI)/(12*k_c^2*(1-d/h)^2)}^(1/3)*phi_M^(2/3)*{(h-d)/abs(L)}
C_IT = 0.056
Tau_phplus.1 = (Tau*u_star^2)/kin.vis
Tau_phplus.2 = C_IT
Tau_phplus = ifelse(Tau_phplus.1<20,Tau_phplus.1, Tau_phplus.2)
E_t.1 = 2.5*10^-3*C_IT*(Tau_phplus)^2
E_t.2 = C_IT
E_t = ifelse(Tau_phplus.1<20, E_t.1, E_t.2)
u_starf = u_star*exp(-alphaPZ)
Tau_phplus.f1 = (Tau*u_star^2)/kin.vis
Tau_phplus.f2 = 0.14
Tau_phplusf = ifelse(Tau_phplus.f1<20,Tau_phplus.f1, Tau_phplus.f2)
E_gt1 = 2.5*10^-3*0.14*(Tau_phplusf)^2
E_gt2 = 0.14
Egt = ifelse(Tau_phplus.f1<20, E_gt1, E_gt2)
Eg = Egb + Egt
# Compute Qg (non-dimensional number)
Qg = Eg*h/lmp_h
# Compute Q
U_z = V5
U_h = U_z/(exp(alphaPZ*(z/h-1)))
L_obs = 0.03
C_B = 1.262
Re_h = (U_h*L_obs)/(kin.vis)
E_B = C_B*(Sc^(-2/3))*(Re_h^(-1/2))
C_IN = 0.216
E_IN = C_IN*(dp/L_obs)*(2+log(4*L_obs/dp))
C_IM = 0.130
beta_IM = 0.47
St_h = (Tau*U_h)/L_obs
E_IM = C_IM*(St_h/(St_h+beta_IM))^2
E_T = (U_h/u_star)*(E_B+E_IN+E_IM)+E_t
Q  = LAI*E_T*h/(lmp_h)
# Compute etaPZ
etaPZ = (alphaPZ^2/4+Q)^0.5
Vds = u_star*Eg*{(1+Q/Qg-alphaPZ/2)*tanh(etaPZ)/etaPZ}/{(1+Q+alphaPZ/2)*tanh(etaPZ)/etaPZ}
Vd = Vdrift+1/(Ra+1/Vds);Vd


#Accuracy Evaluation: Water
#Dry deposition parameterization by Petroff and Zhang (2010)
attach(Moller_Schumann_1970)
C1 = 0.2789
C2 = 3.115
C3 = 5.415*10^-11
C4 = -1.399
RH = 90/100
dp_i = V1
dp_d = dp_i*10^-6
rd = dp_d/2
r_w = {(C1*rd^C2)/(C3*rd^C4-log10(RH))+rd^3}^(1/3)
dp = r_w*2
k_B = 1.38*10^-23
Temp = 273.15+25
P = 101325
d_air = 3.7208*10^-10
lambda = (k_B*Temp)/(sqrt(2)*3.1416*P*d_air^2)
```

```
C = 1+(2*lambda/dp)*(1.257+0.4*exp(-0.55*dp/lambda))
dyn.vis = ((5*10^-8)*Temp)+4*10^-6
rho = 1500
Tau = (rho*(dp)^2*C)/(18*dyn.vis)
Ws = Tau*9.81
Vphor = (5*10^-3)/100
Vdrift = Ws+Vphor
# Compute aerodynamic resistance (Ra):
z = 8/100
L = 50
x = z/L
shi_H.1 = 2*log(0.5*{1+(1-16*x)^0.5})
shi_H.2 = -5*x
shi_H = ifelse(x <= 0, shi_H.1 , shi_H.2)
zR = 5
u_star = 0.4
z0_1 = 0.021*(u_star)^3.32
z0_2 = 0.00098*(u_star)^1.65
z0 = ifelse(u_star<= 0.16, z0_1, z0_2)
k_c = 0.41
Ra = (log(zR/z0)-shi_H)/(k_c*u_star)
# Compute surface resistance (Rs)
kin.vis = ((9*10^-8)*Temp)+10^-5
D = (C*k_B*Temp)/(3*3.1416*dyn.vis*dp)
Sc = (kin.vis/D)
FSc = (Sc^(1/3))/2.9
Egb = (Sc^(-2/3)/14.5)*{1/6*log(1+FSc)^2/(1-FSc+FSc^2)+1/sqrt(3)*atan((2*FSc-
1)/sqrt(3))+3.1416/6*sqrt(3)}^-1
Eg = Egb
Vd = Vdrift+1/(Ra+(1/(Eg*u_star)));Vd


#Accuracy Evaluation: Ice/snow
#Dry deposition velocity parameterization by Petroff and Zhang (2010)
attach(Ibrahim_1983)
C1 = 0.2789
C2 = 3.115
C3 = 5.415*10^-11
C4 = -1.399
RH = 60/100
dp_i = c(0.22, 0.73)
dp_d = dp_i*10^-6
rd = dp_d/2
r_w = {(C1*rd^C2)/(C3*rd^C4-log10(RH))+rd^3}^(1/3)
dp = r_w*2
#Correction factor, C
k_B = 1.38*10^-23
#Temp_1 = 25
Temp = 273.15+3
P = 101325
d_air = 3.7208*10^-10
lambda = (k_B*Temp)/(sqrt(2)*3.1416*P*d_air^2)
C = 1+(2*lambda/dp)*(1.257+0.4*exp(-0.55*dp/lambda))
dyn.vis = ((5*10^-8)*Temp)+4*10^-6
rho = 1500
Tau = (rho*(dp)^2*C)/(18*dyn.vis)
Ws = Tau*9.81
Vphor = (2*10^-4)/100
Vdrift = Ws+Vphor
# Compute aerodynamic resistance (Ra):
#z =
#L = z/L
#x = z/L
x = 0.2
# Compute stability function (shi_H)
shi_H.1 = 2*log(0.5*{1+(1-16*x)^0.5})
shi_H.2 = -5*x
shi_H = ifelse(x <= 0, shi_H.1 , shi_H.2)
zR = 10
u_star = 0.12
z0 = 0.1/100
k_c = 0.41
Ra = (log(zR/z0)-shi_H)/(k_c*u_star)  # m/s
# Compute surface resistance (Rs)
```

```
kin.vis = ((9*10^-8)*Temp)+10^-5
D = (C*k_B*Temp)/(3*3.1416*dyn.vis*dp)
Sc = (kin.vis/D)
FSc = (Sc^(1/3))/2.9
Egb = (Sc^(-2/3)/14.5)*{1/6*log(1+FSc)^2/(1-FSc+FSc^2)+1/sqrt(3)*atan((2*FSc-1)/sqrt(3))+3.1416/6*sqrt(3)}^-1
Eg = Egb
Vd = Vdrift+1/(Ra+(1/(Eg*u_star)));Vd
```

Codes for evaluation of model accuracy using Kouznetsov and Sofiev (2012) parameterization

```
#Accuracy Evaluation: Grass
#Dry deposition parameterization by Kouznetsov and Sofiev (2012)
attach(Allen_etal_1990)
C1 = 0.4809
C2 = 3.082
C3 = 3.110*10^-11
C4 = -1.428
RH = 90/100
dp_a = 0.48
dp_i = dp_a*10^-6
rd = dp_i/2
r_w = {(C1*rd^C2)/(C3*rd^C4-log10(RH))+rd^3}^(1/3)
dp = r_w*2
#Correction factor, C
k_B = 1.38*10^-23
Temp_1 = V1
Temp = 273.15+V1
P = 101325
d_air = 3.7208*10^-10
lambda = (k_B*Temp)/(sqrt(2)*3.1416*P*d_air^2)
C = 1+(2*lambda/dp)*(1.257+0.4*exp(-0.55*dp/lambda))
dyn.vis = ((5*10^-8)*Temp)+4*10^-6
rho =  2000
Tau = (rho*(dp)^2*C)/(18*dyn.vis)
V_s = Tau*9.81
u_star = V2
a  =  2*10^-3
kin.vis = ((9*10^-8)*Temp)+10^-5
D = (C*k_B*Temp)/(3*3.1416*dyn.vis*dp)
Sc = (kin.vis/D)
Re_star = (u_star*a)/kin.vis
# Compute V_diff (velocity for diffusion)
V_diff = 2*(Re_star^(-0.5))*Sc^(-2/3) #m/s
# Compute V_int (velocity for interception)
V_int = 80*u_star*((dp/a)^2)*(Re_star^(0.5))
# Compute V_imp
C_S = 0.003
C_R = 0.3
LAI = 4
CsCR = (C_S+C_R/LAI)^0.5
u.Uh = 0.3
u_star.by.U_h = min(u.Uh, CsCR)
#Compute Re_c
Re_c = ((u_star.by.U_h)^-1)^2*Re_star
# Calculate St
St = (Tau*u_star)/a
# Calculate St_e
St_e = St - Re_c^(-0.5)
eta_impSt.e1 = exp((-0.1/(St_e - 0.15 ) )-(1/sqrt(St_e -0.15)))
eta_impSt.e2 = 0
eta_impSt.e = ifelse(St_e>0.15,eta_impSt.e1,eta_impSt.e2)
V_imp = ((2*u_star.by.U_h)/u_star)*eta_impSt.e*(St-u_star.by.U_h*Re_star^-0.5)
# Dry deposition velocity
Vd = V_diff+V_int+V_imp+V_s;Vd
```

```
#Accuracy Evaluation: Coniferous forest
#Dry deposition parameterization by Kouznetsov and Sofiev (2012)
attach(Rannik_etal_2000)
C1 = 0.2789
C2 = 3.115
C3 = 5.145*10^-11
C4 = -1.399
RH = 0.90
dp_a = V1
dp_i = dp_a*10^-6
rd = dp_i/2
r_w = {(C1*rd^C2)/(C3*rd^C4-log10(RH))+rd^3}^(1/3)
dp = r_w*2
#Correction factor, C
k_B = 1.38*10^-23
Temp =  273.15+25
P = 101325
d_air = 3.7208*10^-10
lambda = (k_B*Temp)/(sqrt(2)*3.1416*P*d_air^2)
C = 1+(2*lambda/dp)*(1.257+0.4*exp(-0.55*dp/lambda))
dyn.vis = 1.891*10^-5
#dyn.vis = V2
rho = 1500
Tau = (rho*(dp)^2*C)/(18*dyn.vis)
V_s = Tau*9.81
# Need to compute Sc, Re_star
u_star = V2
a  =  0.7*10^-3
kin.vis = 1.683*10^-5
D = (C*k_B*Temp)/(3*3.1416*dyn.vis*dp)
Sc = (kin.vis/D)
Re_star = (u_star*a)/kin.vis
# Compute V_diff (velocity for diffusion)
V_diff = 2*(Re_star^(-0.5))*Sc^(-2/3)
# Compute V_int (velocity for interception)
V_int = 80*u_star*((dp/a)^2)*(Re_star^(0.5))
# Compute V_imp
C_S = 0.003
C_R = 0.3
LAI = 6
CsCR = (C_S+C_R/LAI)^0.5
u.Uh = 0.3
u_star.by.U_h = min(u.Uh, CsCR)
#Compute Re_c
Re_c = ((u_star.by.U_h)^-1)^2*Re_star
# Calculate St
St = (Tau*u_star)/a
# Calculate St_e
St_e = St - Re_c^(-0.5)
eta_impSt.e1 = exp((-0.1/(St_e - 0.15 ) )-(1/sqrt(St_e -0.15)))
eta_impSt.e2 = 0
eta_impSt.e = ifelse(St_e>0.15,eta_impSt.e1,eta_impSt.e2)
V_imp = ((2*u_star.by.U_h)/u_star)*eta_impSt.e*(St-u_star.by.U_h*Re_star^-0.5)
# Dry deposition velocity
Vd = V_diff+V_int+V_imp+V_s;Vd


#Accuracy Evaluation: Deciduous forest
#Dry deposition parameterization by Kouznetsov and Sofiev (2012)
attach(Wesely_etal_1983)
C1 = 0.2789
C2 = 3.115
C3 = 5.145*10^-11
C4 = -1.399
RH = 0.95
dp_a = 0.4
dp_i = dp_a*10^-6
rd = dp_i/2
r_w = {(C1*rd^C2)/(C3*rd^C4-log10(RH))+rd^3}^(1/3)
dp = r_w*2
#Correction factor, C
k_B = 1.38*10^-23
Temp = V1
P = 101325
```

```
d_air = 3.7208*10^-10
lambda = (k_B*Temp)/(sqrt(2)*3.1416*P*d_air^2)
C = 1+(2*lambda/dp)*(1.257+0.4*exp(-0.55*dp/lambda))
dyn.vis = V2
rho = 2000
Tau = (rho*(dp)^2*C)/(18*dyn.vis)
V_s = Tau*9.81
# Need to compute Sc, Re_star
u_star = V4
a  = 0.7*10^-3
kin.vis = 1.597*10^-5
D = (C*k_B*Temp)/(3*3.1416*dyn.vis*dp)
Sc = (kin.vis/D)
Re_star = (u_star*a)/kin.vis
# Compute V_diff (velocity for diffusion)
V_diff = 2*(Re_star^(-0.5))*Sc^(-2/3) #m/s
# Compute V_int (velocity for interception)
V_int = 80*u_star*((dp/a)^2)*(Re_star^(0.5))
# Compute V_imp
C_S = 0.003
C_R = 0.3
LAI = 0.2
CsCR = (C_S+C_R/LAI)^0.5
u.Uh = 0.3
u_star.by.U_h = min(u.Uh, CsCR)
#Compute Re_c
Re_c = ((u_star.by.U_h)^-1)^2*Re_star
# Calculate St
St = (Tau*u_star)/a
# Calculate St_e
St_e = St - Re_c^(-0.5)
eta_impSt.e1 = exp((-0.1/(St_e - 0.15 ) )-(1/sqrt(St_e -0.15)))
eta_impSt.e2 = 0
eta_impSt.e = ifelse(St_e>0.15,eta_impSt.e1,eta_impSt.e2)
V_imp = ((2*u_star.by.U_h)/u_star)*eta_impSt.e*(St-u_star.by.U_h*Re_star^-0.5)
#Dry deposition velocity
Vd = V_diff+V_int+V_imp+V_s;Vd


#Accuracy Evaluation: Ice/snow
#Dry deposition parameterization by Kouznetsov and Sofiev (2012)
attach(Ibrahim_1983)
C1 = 0.4809
C2 = 3.082
C3 = 3.110*10^-11
C4 = -1.428
RH = 0.90
dp_a = c(0.22, 0.73)
dp_i = dp_a*10^-6
rd = dp_i/2
r_w = {(C1*rd^C2)/(C3*rd^C4-log10(RH))+rd^3}^(1/3)
dp = r_w*2
#Correction factor, C
k_B = 1.38*10^-23
Temp = 273.15+3
P = 101325
d_air = 3.7208*10^-10
lambda = (k_B*Temp)/(sqrt(2)*3.1416*P*d_air^2)
C = 1+(2*lambda/dp)*(1.257+0.4*exp(-0.55*dp/lambda))
dyn.vis = ((5*10^-8)*Temp)+4*10^-6
rho =  1500
Tau = (rho*(dp)^2*C)/(18*dyn.vis)
V_s = Tau*9.81
u_star = 0.12
a  =  0.5*10^-3
kin.vis = ((9*10^-8)*Temp)+10^-5
D = (C*k_B*Temp)/(3*3.1416*dyn.vis*dp)
Sc = (kin.vis/D)
Re_star = (u_star*a)/kin.vis
# Compute V_diff (velocity for diffusion)
V_diff = 2*(Re_star^(-0.5))*Sc^(-2/3) #m/s
# Compute V_int (velocity for interception)
V_int = 80*u_star*((dp/a)^2)*(Re_star^(0.5))
#Compute V_imp
```

```
C_S = 0.003
C_R = 0.3
LAI = 0
CsCR = (C_S+C_R/LAI)^0.5
u.Uh = 0.3
u_star.by.U_h = min(u.Uh, CsCR)
#Compute Re_c
Re_c = ((u_star.by.U_h)^-1)^2*Re_star
# Calculate St
St = (Tau*u_star)/a
# Calculate St_e
St_e = St - Re_c^(-0.5)
eta_impSt.e1 = exp((-0.1/(St_e - 0.15 ) )-(1/sqrt(St_e -0.15)))
eta_impSt.e2 = 0
eta_impSt.e = ifelse(St_e>0.15,eta_impSt.e1,eta_impSt.e2)
#V_imp = ((2*u_star.by.U_h)/u_star)*eta_impSt.e*(St-u_star.by.U_h*Re_star^-0.5)
V_imp = (2*u_star.by.U_h*eta_impSt.e*(St-(u_star.by.U_h*Re_star^-0.5)))*u_star
# Dry deposition velocity
Vd = V_diff+V_int+V_imp+V_s;Vd


#Accuracy Evaluation: Water
#Dry deposition parameterization by Kouznetsov and Sofiev (2012)
C1 = 0.2789
C2 = 3.115
C3 = 5.415*10^-11
C4 = -1.399
dp_i = c(0.22, 0.73)
dp = dp_i*10^-6
rho = 1500
RH = 0.60
Temp = 25+273.15
u_star = 0.12
z0 =0.1/100
L = 40
kin.vis = ((9*10^-8)*Temp)+10^-5
dyn.vis = ((5*10^-8)*Temp)+4*10^-6
d_air = 3.7208*10^-10
k_B = 1.38*10^-23
P = 101325
lambda = (k_B*Temp)/(sqrt(2)*3.1416*P*d_air^2)
C = 1+((2*lambda/dp)*(1.257+0.4*exp(-0.55*dp/lambda)))
tau_p = (rho*(dp)^2*C)/(18*dyn.vis)
D = (C*k_B*Temp)/(3*3.1416*dyn.vis*dp)
v_s = 9.81*tau_p
taup = (tau_p*u_star^2)/(kin.vis)
Sc = (kin.vis/D)
vsplus = v_s/u_star
rplus = (dp*u_star)/(2*kin.vis)
R_s = 0
Rsplus = u_star*R_s
z_meas = 8
zpmax = (z_meas*u_star)/(kin.vis)
MOplus = (kin.vis)/(u_star*L)
#Fixed parameters
Zbuf = 3
Ztf = 18                              #turbophoretic sublayer height
taultf = 5                            #Lagrangian time in turbophoretic layer
Nutp_Ztf= (0.4*(Ztf)^3)/(Ztf^2+200)  #Dimensionless eddy viscosity of air
It_Ztf = (2.5*log10(Ztf ))-(100/Ztf^2)
It_Zbuf = (2.5*log10(Zbuf))-(100/Zbuf^2)
S = Sc^(1/3)
Zl = 20/S
fTmp = 2.5/Sc
fTmp1 = (fTmp^3/27+(fTmp*(100+5*sqrt(8*fTmp/27)+400)))^(1/3)
Zl_1 = fTmp1+((fTmp*fTmp)/(9*fTmp1))+(1/3)*fTmp                #Zl updated as zl_1
fTmp_1 = (Zl_1^2)/(Zl_1^2+200)                                 #fTmp updated as fTmp_1
fTmp_2 = 1.2*(fTmp_1)-0.8*fTmp_1^2                             #fTmp_1 updated as fTmp_2
fTmp1_1 = 1/(Sc*fTmp_2)                                        #fTmp1 updated as fTmp1_1
fTmp_3 = 1/Sc                                                  #ftmp_2 updated as fTmp_3
x_1 = Zl_1 - fTmp1_1
x_2 = Zl_1
x_3 = Zl_1+fTmp1_1
Nutp_x_1 = (0.4*x_1^3)/(x_1^2+200)
```

```
Nutp_x_2 = (0.4*x_2^3)/(x_2^2+200)
Nutp_x_3 = (0.4*x_3^3)/(x_3^2+200)
fIvd = Rsplus+(Zl_1-
fTmp1_1)*Sc+0.3333*fTmp1_1*(1/fTmp_3+((0.4*x_1^3)/(x_1^2+200)))+4/(fTmp_3+((0.4*x_2^3)/(x_2^2+200)))+1/(fTmp_3+((0
.4*x_3^3)/(x_3^2+200)))
x_4 = zpmax
x_5 = Zl_1 + fTmp1_1
It_x_4 = 2.5*log10(x_4) - 100/(x_4^2)
It_x_5 = 2.5*log10(x_5) - 100/(x_5^2)
s = 2.35*(zpmax*MOplus+abs(zpmax*MOplus))
u1 = 0.5*(abs(zpmax*MOplus)-zpmax*MOplus)
u = -4*u1/(2.65*sqrt(u1*sqrt(u1))+1)
fu_Psi = s+u
fTmp_4 = It_x_4 - It_x_5 + 2.5*fu_Psi
fIvd_1 = fIvd+0.5*(fTmp_4+abs(fTmp_4))    #fIvd updated as fIvd_1
fu_vdplus_smooth_1 = 1/fIvd_1             #mind fu_vdplus_smooth is denoted as _1
Il_input1= Zl*S/7.92
Il_input2= rplus*S/7.92
Il_1 = -0.16667*log10(Il_input1^2-Il_input1+1)+0.57735*atan((2*Il_input1-1)*0.57735)+0.3333*log10(Il_input1+1)
Il_2 = -0.16667*log10(Il_input2^2-Il_input2+1)+0.57735*atan((2*Il_input2-1)*0.57735)+0.3333*log10(Il_input2+1)
R = 7.92*S^2*(Il_1-Il_2)  #laminar resistance
R_1 = 0.5*(R+abs(R))      #R updated as R_1 #should be zero if rplus > Zl
Zl_2 = 0.5*(rplus+Zl+abs(rplus-Zl))   #Zl updated as Zl_2, not to be confused with Zl_1 as if statement is in use
It_Zbuf = 2.5*log10(Zbuf) - (100/(Zbuf^2))
It_Zl_2 = 2.5*log10(Zl_2) - (100/(Zl_2^2))
R1 = It_Zbuf - It_Zl_2
R_2 = R_1+0.5*(R1+abs(R1))                    #R_1 updated as R_2
fTmp_5 = vsplus*R_2                           #fTmp used as fTmp_5. Not updated as previous fTmp_4 is for
different condition
fTmp_6 = exp(-fTmp_5)                         #fTmp_5 updated as fTmp_6
fIvd_2  = Rsplus*fTmp_6+(1-fTmp_6)/vsplus     #fIvd denoted as fIvd_2 NOT updated previous fIvd_1 is for different
condition
fIvd_3 = Rsplus+R_2                           #fIvd_3 used; NOT updated
fIvd_23 = ifelse(abs(fTmp_5)>0.001, fIvd_2, fIvd_3 )
#Use above values for the following calculations for turbophoretic layer
V  = 0.81*taup/(Ztf-Zbuf)/(1+taup/taultf) + vsplus   #chcek for sign of vsplus
It_Ztf = 2.5*log10(Ztf) - 100/(Ztf^2)
R_3  = ((It_Ztf)-(It_Zbuf))*(1+taup/taultf)    #R_2 updated as R_3
fTmp_7 = V*R_3                                 #fTmp_6 updated as fTmp_7
fTmp_8 = exp(-fTmp_7)                          #fTmp_7 updated as fTmp_8
fIvd_4 = (fIvd_23*fTmp_8)+(1-fTmp_8)/V         #fIvd_23 updated as fIvd_4
fIvd_5 = fIvd_3+R_3                            #fIvd_5 used; NOT updated
fIvd_45 = ifelse(abs(fTmp_7)>0.001, fIvd_4, fIvd_5)
#Now calculations for the Lagrangian turbophoretic layer
Ztf2 = 2*taup
V_1 = 0.4+vsplus
R_4 =
0.1667*((1+taup/(0.5*Ztf))/(Nutp_Ztf)+4*(1+taup/(0.25*(Ztf+Ztf2)))/((0.4*(0.5*(Ztf+Ztf2))^3)/((0.5*(Ztf+Ztf2))^2+2
00))+(1+taup/(0.5*Ztf2))/((0.4*Ztf2 ^3)/(Ztf2^2+200)))*(Ztf2-Ztf)
fTmp_9 = V_1*R_4
fTmp_10 = exp(-fTmp_9)
fIvd_6 = fIvd_45*fTmp_10 + (1-fTmp_10)/V_1   #fIvd_5 or fIvd_4 (fIvd_45)
fIvd_7 = fIvd_45+R_3                         #Either fIvd_4 or fIvd_5, R_3 used not R_4 since outside if statement
fIvd_67 = ifelse(abs(fTmp_9) > 0.001, fIvd_6, fIvd_7)
Ztf2_2 = Ztf
fIvd67_Ztf2 = ifelse(Ztf2>Ztf, fIvd_67, Ztf2_2)

#Following calculations are for aerodynamic layer
It_zpmax = 2.5*log10(zpmax) - 100/(zpmax^2)
Ztf22 = ifelse(Ztf2>Ztf, Ztf2, Ztf2_2)
It_Ztf22  = 2.5*log10(Ztf22) - 100/(Ztf22^2)       #be careful which Ztf2 to be used here based on previous
condition
R_5 = It_zpmax - It_Ztf22  + 2.5*fu_Psi            #R_5 used, NOT updated from R_4
R_6 = 0.5*(R_5+abs(R_5))                           #R_5 updated as R_6
fTmp_11  = vsplus*R_6                              #fTmp_11 used, NOT updated from fTmp_10
fTmp_12 = exp(-fTmp_11)                            #fTmp_11 updated as fTmp_12
fIvd_4567 = ifelse(Ztf2>Ztf, fIvd_67,fIvd_45)
fIvd_8 = (fIvd_4567* fTmp_12)+(1-fTmp_12)/vsplus   # Caution:fIvd_45 or fIvd_67 may be used
fIvd_9 = fIvd_4567 + R_6                           #fIvd_9 used, NOT updated. Caution:fIvd_5 or fIvd_4 may be used
fIvd_10 = ifelse(abs(fTmp_11) > 0.001, fIvd_8, fIvd_9)
fu_vdplus_smooth_3 = 1/fIvd_10                      #fu_vdplus_smooth_3 used. NOT updated. fIvd_8 could be used
fu_vdplus_smooth = ifelse(Zl>Zbuf, fu_vdplus_smooth_1, fu_vdplus_smooth_3)
Vd_smooth = fu_vdplus_smooth*u_star;Vd_smooth
```

Codes for evaluation of model accuracy using Zhang and He (2014) parameterization

```
#Accuracy Evaluation: Grass (code is similar for coniferous forest)
#Dry deposition parameterization by Zhang and He (2014)
attach(Allen_etal_1991)
C1 = 0.4809
C2 = 3.082
C3 = 3.110*10^-11
C4 = -1.428
RH = 90/100
dp_a = 0.48
dp_i = dp_a*10^-6
rd = dp_i/2
r_w = {(C1*rd^C2)/(C3*rd^C4-log10(RH))+rd^3}^(1/3)
dp = r_w*2
#Correction factor, C
k_B = 1.38*10^-23
Temp_1 = V1
Temp = 273.15+V1
P = 101325
d_air = 3.7208*10^-10
lambda = (k_B*Temp)/(sqrt(2)*3.1416*P*d_air^2)
C = 1+(2*lambda/dp)*(1.257+0.4*exp(-0.55*dp/lambda))
dyn.vis = ((5*10^-8)*Temp)+4*10^-6              # kg/m*s (temp. corrected viscosity coeff. of air)
rho = 2000
Tau = (rho*(dp)^2*C)/(18*dyn.vis)
V_g = Tau*9.81
Rg = 1/V_g
u_star = V2
a1 = 4.8*10^-3
z = 2
L = V4
x = z/L
# Compute stability function (shi_H)
shi_H.1 = 2*log(0.5*{1+(1-16*x)^0.5})
shi_H.2 = -5*x
shi_H =ifelse(x <= 0, shi_H.1 , shi_H.2)
zR = 3.5
z0 = 0.01
k_c = 0.41
Ra = (log(zR/z0)-shi_H)/(k_c*u_star)
# Calculate Vds = 1/Rs
#For PM2.5
Vds_PM2.5 = (a1*u_star)
Rds_PM2.5 = (1/Vds_PM2.5)
# For PM2.5-10
#b1= -1.6*10^-1
#b2= 1.5*10^0
#b3 = 7.8*10^-1
#c1= 1.8
#c2 = -2.0*10^-1
#c3 = -5.3*10^-1
#k = c1*u_star+c2*u_star^2+c3*u_star^3
#LAI =
#LAImax =
#Vds_PM10 = (b1*u_star+b2*u_star^2+b3*u_star^3)  #*exp(k*(LAI/LAImax)-1)
#Rds_PM10 = 1/Vds_PM10
# For PM10+
#d1= -2.2
#d2= 3.9*10^1
#d3 = -6.7
#f1= 6.2
#f2 = -1.2*10^1
#f3 = 6.1
#k = f1*u_star+f2*u_star^2+f3*u_star^3
#LAI =
#LAImax =
#Vds_PM10Plus = (d1*u_star+d2*u_star^2+d3*u_star^3)*exp(k*(LAI/LAImax)-1)
#Rds_PM10Plus = (1/Vds_PM10Plus)
#Compute Vd
Vd = 1/Rg+(1/(Ra+Rds_PM2.5));Vd
```

```
#Accuracy Evaluation: Deciduous forest
#Dry deposition parameterization by Zhang and He (2014)
attach(Rannik_etal_2000)
C1 = 0.2789
C2 = 3.115
C3 = 5.145*10^-11
C4 = -1.399
RH = 90/100
dp_a = V1
dp_i = dp_a*10^-6
rd = dp_i/2
r_w = {(C1*rd^C2)/(C3*rd^C4-log10(RH))+rd^3}^(1/3)
dp = r_w*2
k_B = 1.38*10^-23
Temp = 273.15+25
P = 101325
d_air = 3.7208*10^-10
lambda = (k_B*Temp)/(sqrt(2)*3.1416*P*d_air^2)
C = 1+(2*lambda/dp)*(1.257+0.4*exp(-0.55*dp/lambda))
dyn.vis = 1.891*10^-5
rho = 1500
Tau = (rho*(dp)^2*C)/(18*dyn.vis)
V_g = Tau*9.81
Rg = 1/V_g
u_star = V2
a1 = 4.3*10^-3
z = 23.7
L = 200
x = z/L
# Compute stability function (shi_H)
shi_H.1 = 2*log(0.5*{1+(1-16*x)^0.5})
shi_H.2 = -5*x
shi_H =ifelse(x <= 0, shi_H.1 , shi_H.2)
zR = 26
z0 = 1.2
k_c = 0.41
Ra = (log(zR/z0)-shi_H)/(k_c*u_star)
# Calculate Vds = 1/Rs
#For PM2.5
Vds_PM2.5 = (a1*u_star)
Rds_PM2.5 = (1/Vds_PM2.5)
# For PM2.5-10
#b1= -1.6*10^-1
#b2= 1.5*10^0
#b3 = 7.8*10^-1
#c1= 1.8
#c2 = -2.0*10^-1
#c3 = -5.3*10^-1
k = c1*u_star+c2*u_star^2+c3*u_star^3
LAI = 12
LAImax = 12
Vds_PM10 = (b1*u_star+b2*u_star^2+b3*u_star^3)  #*exp(k*(LAI/LAImax)-1)
Rds_PM10 = 1/Vds_PM10
# For PM10+
#d1= -2.2
#d2= 3.9*10^1
#d3 = -6.7
#f1= 6.2
#f2 = -1.2*10^1
#f3 = 6.1
#k = f1*u_star+f2*u_star^2+f3*u_star^3
#LAI =
#LAImax =
#Vds_PM10Plus = (d1*u_star+d2*u_star^2+d3*u_star^3)*exp(k*(LAI/LAImax)-1)
#Rds_PM10Plus = (1/Vds_PM10Plus)
#Compute Vd
Vd = 1/Rg+(1/(Ra+Rds_PM2.5));Vd


#Accuracy Evaluation: Water
#Dry deposition parameterization by Zhang and He (2014)
attach(Caffrey_etal_1998)
C1 = 0.2789
C2 = 3.115
```

```
C3 = 5.415*10^-11
C4 = -1.399
RH = 79/100
dp_a = V1
dp_i = dp_a*10^-6
rd = dp_i/2
r_w = {(C1*rd^C2)/(C3*rd^C4-log10(RH))+rd^3}^(1/3)
dp = r_w*2
#Correction factor, C
k_B = 1.38*10^-23
Temp = 273.15+22
P = 101325
d_air = 3.7208*10^-10
lambda = (k_B*Temp)/(sqrt(2)*3.1416*P*d_air^2)
C = 1+(2*lambda/dp)*(1.257+0.4*exp(-0.55*dp/lambda))
dyn.vis = ((5*10^-8)*Temp)+4*10^-6
rho = 2000
Tau = (rho*(dp)^2*C)/(18*dyn.vis)
V_g = Tau*9.81
Rg = 1/V_g
a1 = 6.9*10^-3
z = 8/100
L = 50
x = z/L
# Compute stability function (shi_H)
shi_H.1 = 2*log(0.5*{1+(1-16*x)^0.5})
shi_H.2 = -5*x
shi_H =ifelse(x <= 0, shi_H.1 , shi_H.2)
zR = 5
u_star = 13.5/100
z0_1 = 0.021*(u_star)^3.32
z0_2 = 0.00098*(u_star)^1.65
z0 = ifelse(u_star<= 0.16, z0_1, z0_2)
k_c = 0.41
Ra = (log(zR/z0)-shi_H)/(k_c*u_star)  # m/s
# Calculate Vds = 1/Rs
#For PM2.5
Vds_PM2.5 = (a1*u_star)
Rds_PM2.5 = (1/Vds_PM2.5)
#Compute Vd
Vd = 1/Rg+(1/(Ra+Rds_PM2.5));Vd


#Accuracy Evaluation: Ice/snow
#Dry deposition parameterization by Zhang and He (2014)
#attach(Nilsson_Rannik_2001)
C1 = 0.2789
C2 = 3.115
C3 = 5.415*10^-11
C4 = -1.399
RH = 60/100
dp_a = c(0.22, 0.73)
dp_i = dp_a*10^-6
rd = dp_i/2
r_w = {(C1*rd^C2)/(C3*rd^C4-log10(RH))+rd^3}^(1/3)
dp = r_w*2
#Correction factor, C
k_B = 1.38*10^-23
Temp = 273.15+3
P = 101325
d_air = 3.7208*10^-10
lambda = (k_B*Temp)/(sqrt(2)*3.1416*P*d_air^2)
C = 1+(2*lambda/dp)*(1.257+0.4*exp(-0.55*dp/lambda))
dyn.vis = ((5*10^-8)*Temp)+4*10^-6
rho = 1500
Tau = (rho*(dp)^2*C)/(18*dyn.vis)
V_g = Tau*9.81
Rg = 1/V_g
a1 = 4.3*10^-3
#z =
#L =
#x = z/L
x = 0.2
# Compute stability function (shi_H)
```

```
shi_H.1 = 2*log(0.5*{1+(1-16*x)^0.5})
shi_H.2 = -5*x
shi_H =ifelse(x <= 0, shi_H.1 , shi_H.2)
zR = 10
u_star = 0.12
z0 = 0.1/100
k_c = 0.41
Ra = (log(zR/z0)-shi_H)/(k_c*u_star)  # m/s
# Calculate Vds = 1/Rs
#For PM2.5
Vds_PM2.5 = (a1*u_star)
Rds_PM2.5 = (1/Vds_PM2.5)
#Compute Vd
Vd = 1/Rg+(1/(Ra+Rds_PM2.5));Vd
```

Codes for evaluation of model accuracy using Zhang and Shao (2014) parameterization

```
#Accuracy Evaluation: Rough and smooth surfaces
#Dry deposition parameterization by Zhang and He (2014)
#attach(Allen_etal_1991)
C1 = 0.4809
C2 = 3.082
C3 = 3.110*10^-11
C4 = -1.428
RH_1 = 90
RH = 90/100
dp_a = 0.50
dp_i = dp_a*10^-6
rd = dp_i/2
r_w = {(C1*rd^C2)/(C3*rd^C4-log10(RH))+rd^3}^(1/3)
dp = r_w*2
#Correction factor, C
k_B = 1.38*10^-23
Temp = 273.15+25
P = 101325
d_air = 3.7208*10^-10
lambda = (k_B*Temp)/(sqrt(2)*3.1416*P*d_air^2)
C = 1+(2*lambda/dp)*(1.257+0.4*exp(-0.55*dp/lambda))
Temp = 273.15+25
dyn.vis = ((5*10^-8)*Temp)+4*10^-6
rho = 1500
Tau = (rho*(dp_i)^2*C)/(18*dyn.vis)
Tau_wet = (rho*(dp)^2*C)/(18*dyn.vis)
Wt = Tau*9.81
Vg = Wt
Vg_wet = Tau_wet*9.81
u_star = 0.5
k =  0.41
z =  1
zd = 0.20
h_c = 0.23
z0 = 0.002
B1 = 0.45
Sc_T = (1+(Vg^2/u_star^2))^0.5
Ra = (Sc_T/(k*u_star))*(log((z-zd)/(h_c-zd)))  # For rough surface
#Ra = (B1*Sc_T/k*u_star)*log(z/z0)             # For smooth surface
Rg = 1/Vg
# Calculate surface resistance (Rs)
U_h = 2
kin.vis = ((9*10^-8)*Temp)+10^-5
d_c = 0.005
D = (C*k_B*Temp)/(3*3.1416*dyn.vis*dp)
Sc = (kin.vis/D)
Re_h = (U_h*d_c)/(kin.vis)
nB = 0.5
C_B = 0.467
E_B = C_B*Sc^(-2/3)*Re_h^(nB-1)
#Compute impaction collection efficiency (E_IM)
beta_IM = 0.6
St_h = (Tau*u_star)/d_c
E_IM = (St_h/(St_h+beta_IM))^2
#Compute interception efficiency (E_IN)
Ain = 150
```

```
E_IN = Ain*u_star*(10^(-St_h))*(2*dp/d_c)
#Compute R
b = 2
R = exp(-b*sqrt(St_h))
#Compute w_dm
B2 = 3
w_dm = (u_star/U_h*h_c)       #For rough surface
#w_dm = B2*u_star             #For smooth surface
#compute Tau_c/Tau (ratio of stress)
Beta = 200
C_1 = 6
C_2 = 0.1
lambda_FAI = 0.4
n_FAI = (lambda_FAI)/(h_c*d_c)
q = (3.1416*d_c^2)/4
eta_BAI = n_FAI*q
lambda_FAIe = ((lambda_FAI)/(1-eta_BAI)^C_2)*exp((-C_1*lambda_FAI)/(1-eta_BAI)^C_2)
Tau_c_BY_Tau = (Beta*lambda_FAIe)/(1+Beta*lambda_FAIe)
#Compute Rs
E = E_B+E_IM+E_IN
Tau_wetplus = (Tau_wet*u_star^2)/kin.vis
Cd = 1/6
Rs = (R*w_dm*((E*Tau_c_BY_Tau/Cd)+(1+Tau_c_BY_Tau)*Sc^-1+10^(-3/Tau_wetplus))+Vg_wet)^-1
#Compute Vd
Vd = (Rg+((Rs-Rg)/exp(Ra/Rg)))^-1;Vd
```

Codes for Monte Carlo uncertainty evaluation for Zhang et al. (2001) parameterization

```
#Dry deposition parameterization by Zhang et al. (2001)
#Uncertainty test: Grass
set.seed(5)
C1 = 0.2789
C2 = 3.115
C3 = 5.145*10^-11
C4 = -1.399
RH = replicate(10000,runif(100,0.76,0.84))
dp_i = 2.0*10^-6
rd = dp_i/2
r_w = {(C1*rd^C2)/(C3*rd^C4-log10(RH))+rd^3}^(1/3)
dp = r_w*2
#Correction factor, C
k_B = 1.38*10^-23
Temp = (273.15+25)
P = 101325
d_air = 3.72*10^-10
lambda = (k_B*Temp)/(sqrt(2)*3.1416*P*d_air^2)
C = 1+(2*lambda/dp)*(1.257+0.4*exp(-0.55*dp/lambda))
dyn.vis = 1.8908*10^-5
rho = 1500
Vg = (rho*(dp)^2*9.81*C)/(18*dyn.vis)
# Compute aerodynamic resistance Ra:
z = 5
L = replicate(10000,runif(100,45,55))
x = z/L
# Compute shi_H (stability function)
shi_H2 = -5*x
zR = 3.5
z0 = replicate(10000,runif(100,0.03, 0.05))
u_star = replicate(10000,runif(100,0.27,0.33))
k_c = 0.41
Ra = (log(zR/z0)-shi_H2)/(k_c*u_star)
# Compute surface resistance Rs:
e_0 = 3
R1 =1
# Compute E_B (collection efficienty from Brownian diffusion)
kin.vis = 1.6834*10^-5
gamma = 0.54
D = (C*k_B*Temp)/(3*3.1416*dyn.vis*dp)
Sc = (kin.vis/D)
E_B = Sc^(-gamma)
# Compute E_IM (collection efficiency from impaction)
alpha = 1.2
beta =  2
```

```
A = 2/1000
St = (Vg*u_star)/(9.81*A)
E_IM = {St/(alpha+St)}^beta
# Compute E_IN (collection efficiency from interception)
E_IN = 0.5*(dp/A)^2
Rs = 1/{(e_0*u_star)*(E_B+E_IM+E_IN)*R1}
# Compute Dry deposition velocity
Vd <- Vg+(1/(Ra+Rs))
quantile(Vd, c(.05, 0.10, .50, 0.95))


#Dry deposition parameterization by Zhang et al. (2001)
#Uncertainty test: Coniferous forest
set.seed(5)
C1 = 0.2789
C2 = 3.115
C3 = 5.145*10^-11
C4 = -1.399
RH = replicate(10000,runif(100,0.76,0.84))
dp_i = 2.0*10^-6                #particle dia = 0.005-50 um (assumed, vary)
rd = dp_i/2
r_w = {(C1*rd^C2)/(C3*rd^C4-log10(RH))+rd^3}^(1/3)
dp = r_w*2
#Correction factor, C
k_B = 1.38*10^-23
Temp = (273.15+25)
P = 101325
d_air = 3.72*10^-10
lambda = (k_B*Temp)/(sqrt(2)*3.1416*P*d_air^2)
C = 1+(2*lambda/dp)*(1.257+0.4*exp(-0.55*dp/lambda))
dyn.vis = 1.8908*10^-5
rho = 1500
Vg = (rho*(dp)^2*9.81*C)/(18*dyn.vis)   #m/s
# Compute aerodynamic resistance Ra:
L = replicate(10000,runif(100,45,55))
x = z/L
shi_H2 = -5*x
zR = 30
z0 = replicate(10000,runif(100,0.9, 1.5))
u_star = replicate(10000,runif(100,0.27,0.33))
k_c = 0.41
Ra = (log(zR/z0)-shi_H2)/(k_c*u_star)  # m/s
# Compute surface resistance Rs:
e_0 = 3
R1 =1
# Compute E_B (collection efficiency from Brownian diffusion)
kin.vis = 1.6834*10^-5
gamma = 0.56
D = (C*k_B*Temp)/(3*3.1416*dyn.vis*dp)
Sc = (kin.vis/D)
E_B = Sc^(-gamma)
# Compute E_IM (collection efficiency from impaction)
alpha = 1.0
beta =  2
A = 2/1000
St = (Vg*u_star)/(9.81*A)
E_IM = {St/(alpha+St)}^beta
# Compute E_IN (collection efficiency from interception)
E_IN = 0.5*(dp/A)^2
Rs = 1/{(e_0*u_star)*(E_B+E_IM+E_IN)*R1}  #(m/s)
# Compute Dry deposition velocity
Vd <- Vg+(1/(Ra+Rs))
quantile(Vd, c(.05, 0.10, .50, 0.95))


#Dry deposition parameterization by Zhang et al. (2001)
#Uncertainty test: Deciduous forest
set.seed(5)
C1 = 0.2789
C2 = 3.115
C3 = 5.145*10^-11
C4 = -1.399
RH = replicate(10000,runif(100,0.76,0.84))
dp_i = 2.0*10^-6
```

```
rd = dp_i/2
r_w = {(C1*rd^C2)/(C3*rd^C4-log10(RH))+rd^3}^(1/3)
dp = r_w*2
#Correction factor, C
k_B = 1.38*10^-23
Temp = (273.15+25)
P = 101325
d_air = 3.72*10^-10
lambda = (k_B*Temp)/(sqrt(2)*3.1416*P*d_air^2)
C = 1+(2*lambda/dp)*(1.257+0.4*exp(-0.55*dp/lambda))
dyn.vis = 1.8908*10^-5
rho = 1500
Vg = (rho*(dp)^2*9.81*C)/(18*dyn.vis)
# Compute aerodynamic resistance Ra:
z = 35
L = replicate(10000,runif(100,45,55))
x = 35/L
# Compute shi_H (stability function)
shi_H2 = -5*x
zR = 50
z0 = replicate(10000,runif(100,1.125, 1.875))
u_star = replicate(10000,runif(100,0.27,0.33))
k_c = 0.41
Ra = (log(zR/z0)-shi_H2)/(k_c*u_star)
# Compute surface resistance Rs:
e_0 = 3
R1 =1
# Compute E_B (collection efficienty from Brownian diffusion)
kin.vis = 1.6834*10^-5
gamma = 0.56
D = (C*k_B*Temp)/(3*3.1416*dyn.vis*dp)
Sc = (kin.vis/D)
E_B = Sc^(-gamma)
# Compute E_IM (collection efficienty from impaction)
alpha = 0.80
beta =  2
A = 5/1000
St = (Vg*u_star)/(9.81*A)
E_IM = {St/(alpha+St)}^beta
# Compute E_IN (collection efficienty from interception)
E_IN = 0.5*(dp/A)^2
Rs = 1/{(e_0*u_star)*(E_B+E_IM+E_IN)*R1}
# Compute Dry deposition velocity
Vd <- Vg+(1/(Ra+Rs))
quantile(Vd, c(.05, 0.10, .50, 0.95))


#Dry deposition parameterization by Zhang et al. (2001)
#Uncertainty test: Water
set.seed(5)
C1 = 0.2789
C2 = 3.115
C3 = 5.415*10^-11
C4 = -1.399
RH = replicate(10000,runif(100,0.76,0.84))
dp_i = 2.0 #Parameter to vary for MCs
dp_d = dp_i*10^-6
rd = dp_d/2
r_w = {(C1*rd^C2)/(C3*rd^C4-log10(RH))+rd^3}^(1/3)
dp = r_w*2
#Correction factor, C
k_B = 1.38*10^-23
Temp = 273.15+25
P = 101325
d_air = 3.72*10^-10
lambda = (k_B*Temp)/(sqrt(2)*3.1416*P*d_air^2)
C = 1+(2*lambda/dp)*(1.257+0.4*exp(-0.55*dp/lambda))
dyn.vis = ((5*10^-8)*Temp)+4*10^-6
rho = 1500
Vg = (rho*(dp)^2*9.81*C)/(18*dyn.vis)
# Compute aerodynamic resistance Ra:
z = 8/100
L = replicate(10000,runif(100,45,55))
x = z/L
```

```
# Compute shi_H (stability function)
shi_H.1 = 2*log(0.5*{1+(1-16*x)^0.5})
shi_H.2 = -5*x
shi_H =ifelse(x <= 0, shi_H.1 , shi_H.2)
zR = 5
u_star = replicate(10000,runif(100,0.27,0.33))
z0_1 = 0.021*(u_star)^3.32
z0_2 = 0.00098*(u_star)^1.65
z0 = ifelse(u_star<= 0.16, z0_1, z0_2)
k_c = 0.41
Ra = (log(zR/z0)-shi_H)/(k_c*u_star)
# Compute surface resistance Rs:
e_0 = 3
R1 = 1
kin.vis = ((9*10^-8)*Temp)+10^-5
gamma = 0.50
D = (C*k_B*Temp)/(3*3.1416*dyn.vis*dp)
Sc = (kin.vis/D)
E_B = Sc^(-gamma)
alpha = 100
beta =  2
A = 2/1000
St = (Vg*u_star^2)/(kin.vis)
E_IM = {St/(alpha+St)}^beta
Rs = 1/{(e_0*u_star)*(E_B+E_IM)*R1}  #(m/s)
# Compute Dry deposition velocity
Vd =  Vg+(1/(Ra+Rs))
quantile(Vd, c(.05, 0.10, .50, 0.95))


#Dry deposition parameterization by Zhang et al. (2001)
#Uncertainty test: Ice/snow
set.seed(5)
C1 = 0.2789
C2 = 3.115
C3 = 5.415*10^-11
C4 = -1.399
RH = replicate(10000,runif(100,0.76,0.84))
dp_i = 2.0
dp_d = dp_i*10^-6
rd = dp_d/2
r_w = {(C1*rd^C2)/(C3*rd^C4-log10(RH))+rd^3}^(1/3)
dp = r_w*2
#Correction factor, C
k_B = 1.38*10^-23
Temp = 273.15+0
P = 101325
d_air = 3.72*10^-10
lambda = (k_B*Temp)/(sqrt(2)*3.1416*P*d_air^2)
C = 1+(2*lambda/dp)*(1.257+0.4*exp(-0.55*dp/lambda))
dyn.vis = ((5*10^-8)*Temp)+4*10^-6
rho = 1500
Vg = (rho*(dp)^2*9.81*C)/(18*dyn.vis)
# Compute aerodynamic resistance Ra:
z = 5
L = replicate(10000,runif(100,45,55))
x = z/L
# Compute shi_H (stability function)
shi_H.1 = 2*log(0.5*{1+(1-16*x)^0.5})
shi_H.2 = -5*x
shi_H =ifelse(x <= 0, shi_H.1 , shi_H.2)
zR = 10
u_star = replicate(10000,runif(100,0.27,0.33))
z0 = replicate(10000,runif(100,0.0075,0.0125))
k_c = 0.41
Ra = (log(zR/z0)-shi_H)/(k_c*u_star)
# Compute surface resistance Rs:
e_0 = 3
R1 = 1
kin.vis = ((9*10^-8)*Temp)+10^-5
gamma = 0.54
D = (C*k_B*Temp)/(3*3.1416*dyn.vis*dp)
Sc = (kin.vis/D)
E_B = Sc^(-gamma)
```

```
alpha = 50
beta =  2
A = 2/1000
St = (Vg*u_star^2)/(kin.vis)
E_IM = {St/(alpha+St)}^beta
Rs = 1/{(e_0*u_star)*(E_B+E_IM)*R1}  #(m/s)
# Compute Dry deposition velocity
Vd =  Vg+(1/(Ra+Rs))
quantile(Vd, c(.05, 0.10, .50, 0.95))
```

Codes for Monte Carlo uncertainty evaluation for Petroff and Zhang (2010) parameterization

```
#Dry deposition parameterization by Petroff and Zhang (2010)
#Uncertainty test: Grass
set.seed(5)
C1 = 0.2789
C2 = 3.115
C3 = 5.145*10^-11
C4 = -1.399
RH = replicate(10000,runif(100,0.76,0.84))
dp_a = 2.0
dp_i = dp_a*10^-6
rd = dp_i/2
r_w = {(C1*rd^C2)/(C3*rd^C4-log10(RH))+rd^3}^(1/3)
dp = r_w*2
#Correction factor, C
k_B = 1.38*10^-23
Temp = (273.15+25)
P = 101325
d_air = 3.7208*10^-10
lambda = (k_B*Temp)/(sqrt(2)*3.1416*P*d_air^2)
C = 1+(2*lambda/dp)*(1.257+0.4*exp(-0.55*dp/lambda))
dyn.vis = 1.8908*10^-5
rho = 1500
Tau = (rho*(dp)^2*C)/(18*dyn.vis)
Ws = Tau*9.81
Vphor = 0
Vdrift = Ws+Vphor
# Compute aerodynamic resistance (Ra):
z = 5
L = replicate(10000,runif(100,45,55))
x = z/L
# Compute stability function (shi_H)
shi_H.1 = 2*log(0.5*{1+(1-16*x)^0.5})
shi_H.2 = -5*x
shi_H =ifelse(x <= 0, shi_H.1 , shi_H.2)
zR = 3.5
z0 = replicate(10000,runif(100,0.03,0.05))
u_star = replicate(10000,runif(100,0.27,0.33))
k_c = 0.41
Ra = (log(zR/z0)-shi_H)/(k_c*u_star)
# Compute surface resistance (Rs)
kin.vis = 1.6834*10^-5
D = (C*k_B*Temp)/(3*3.1416*dyn.vis*dp)
Sc = (kin.vis/D)
FSc = (Sc^(1/3))/2.9
Egb = (Sc^(-2/3)/14.5)*{1/6*log(1+FSc)^2/(1-FSc+FSc^2)+1/sqrt(3)*atan((2*FSc-1)/sqrt(3))
        +3.1416/6*sqrt(3)}^-1
cd = 1/6
kx = 0.216
LAI = replicate(10000,runif(100,3.8,4.2))
h = replicate(10000,runif(100,0.475,0.525))
d = replicate(10000,runif(100,0.225,0.375))
phi_H.1 = (1-16*x)^(-0.5)
phi_H.2 = 1+5*x
phi_H = ifelse(x<=0, phi_H.1, phi_H.2)
phi_M.1 = (1-16*x)^(-0.25)
phi_M.2 = 1+5*x
phi_M = ifelse(x<=0, phi_M.1, phi_M.2)
lmp = (0.41*(z-d))/(phi_H*(z-d)/abs(L))
lmp_h = (0.41*(h-d))/(phi_M*(h-d)/abs(L))
alphaPZ = {(kx*LAI)/(12*k_c^2*(1-d/h)^2)}^(1/3)*phi_M^(2/3)*{(h-d)/abs(L)}
C_IT = 0.042
```

```
Tau_phplus.1 = (Tau*u_star^2)/kin.vis
Tau_phplus.2 = C_IT
Tau_phplus = ifelse(Tau_phplus.1<20,Tau_phplus.1, Tau_phplus.2)
E_t.1 = 2.5*10^-3*C_IT*(Tau_phplus)^2
E_t.2 = C_IT
E_t = ifelse(Tau_phplus.1<20, E_t.1, E_t.2)
u_starf = u_star*exp(-alphaPZ)
Tau_phplus.f1 = (Tau*u_star^2)/kin.vis
Tau_phplus.f2 = 0.14
Tau_phplusf = ifelse(Tau_phplus.f1<20,Tau_phplus.f1, Tau_phplus.f2)
E_gt1 = 2.5*10^-3*0.14*(Tau_phplusf)^2
E_gt2 = 0.14
Egt = ifelse(Tau_phplus.f1<20, E_gt1, E_gt2)
Eg = Egb + Egt
Qg = Eg*h/lmp_h
U_z = replicate(10000,runif(100,2.91,3.09))
U_h = U_z/(exp(alphaPZ*(z/h-1)))
L_obs = 0.005
C_B = 0.7
Re_h = (U_h*L_obs)/(kin.vis)
E_B = C_B*(Sc^(-2/3))*(Re_h^(-1/2))
C_IN = 0.7
E_IN = C_IN*(dp/L_obs)
C_IM = 0.191
beta_IM = 0.60
St_h = (Tau*U_h)/L_obs
E_IM = C_IM*(St_h/(St_h+beta_IM))^2
E_T = (U_h/u_star)*(E_B+E_IN+E_IM)+E_t
Q  = LAI*E_T*h/(lmp_h)
etaPZ = (alphaPZ^2/4+Q)^0.5
Vds = u_star*Eg*{(1+Q/Qg-alphaPZ/2)*tanh(etaPZ)/etaPZ}/{(1+Q+alphaPZ/2)*tanh(etaPZ)/etaPZ}
Vd = Vdrift+1/(Ra+1/Vds)
quantile(Vd, c(.05, 0.10, .50, 0.95))


#Dry deposition parameterization by Petroff and Zhang (2010)
#Uncertainty test: Coniferous forest
set.seed(5)
C1 = 0.2789
C2 = 3.115
C3 = 5.145*10^-11
C4 = -1.399
RH = replicate(10000,runif(100,0.76,0.84))
dp_a = 50
dp_i = dp_a*10^-6
rd = dp_i/2
r_w = {(C1*rd^C2)/(C3*rd^C4-log10(RH))+rd^3}^(1/3)
dp = r_w*2
#Correction factor, C
k_B = 1.38*10^-23
Temp = (273.15+25)
P = 101325
d_air = 3.7208*10^-10
lambda = (k_B*Temp)/(sqrt(2)*3.1416*P*d_air^2)
C = 1+(2*lambda/dp)*(1.257+0.4*exp(-0.55*dp/lambda))
dyn.vis = 1.8908*10^-5
rho = 1500
Tau = (rho*(dp)^2*C)/(18*dyn.vis)
Ws = Tau*9.81
Vphor = 0
Vdrift = Ws+Vphor
# Compute aerodynamic resistance (Ra):
z = 35
L = replicate(10000,runif(100,180,200))
x = z/L
# Compute stability function (shi_H)
shi_H.1 = 2*log(0.5*{1+(1-16*x)^0.5})
shi_H.2 = -5*x
shi_H =ifelse(x <= 0, shi_H.1 , shi_H.2)
zR = 30
z0 = replicate(10000,runif(100,0.9,1.5))
u_star = replicate(10000,runif(100,0.27,0.33))
k_c = 0.41
Ra = (log(zR/z0)-shi_H)/(k_c*u_star)
```

```
# Compute surface resistance (Rs)
kin.vis = 1.6834*10^-5
D = (C*k_B*Temp)/(3*3.1416*dyn.vis*dp)
Sc = (kin.vis/D)
FSc = (Sc^(1/3))/2.9
Egb = (Sc^(-2/3)/14.5)*{1/6*log(1+FSc)^2/(1-FSc+FSc^2)+1/sqrt(3)*atan((2*FSc-1)/sqrt(3))
       +3.1416/6*sqrt(3)}^-1
cd = 1/6
kx = 0.216
LAI = replicate(10000,runif(100,9.5,10.5))
h = replicate(10000,runif(100,14.25,15.75))
d = replicate(10000,runif(100,5.25,8.75))
phi_H.1 = (1-16*x)^(-0.5)
phi_H.2 = 1+5*x
phi_H = ifelse(x<=0, phi_H.1, phi_H.2)
phi_M.1 = (1-16*x)^(-0.25)
phi_M.2 = 1+5*x
phi_M = ifelse(x<=0, phi_M.1, phi_M.2)
lmp = (0.41*(z-d))/(phi_H*(z-d)/abs(L))
lmp_h = (0.41*(h-d))/(phi_M*(h-d)/abs(L))
alphaPZ = {(kx*LAI)/(12*k_c^2*(1-d/h)^2)}^(1/3)*phi_M^(2/3)*{(h-d)/abs(L)}
C_IT = 0
Tau_phplus.1 = (Tau*u_star^2)/kin.vis
Tau_phplus.2 = C_IT
Tau_phplus = ifelse(Tau_phplus.1<20,Tau_phplus.1, Tau_phplus.2)
E_t.1 = 2.5*10^-3*C_IT*(Tau_phplus)^2
E_t.2 = C_IT
E_t = ifelse(Tau_phplus.1<20, E_t.1, E_t.2)
u_starf = u_star*exp(-alphaPZ)
Tau_phplus.f1 = (Tau*u_star^2)/kin.vis
Tau_phplus.f2 = 0.14
Tau_phplusf = ifelse(Tau_phplus.f1<20,Tau_phplus.f1, Tau_phplus.f2)
E_gt1 = 2.5*10^-3*0.14*(Tau_phplusf)^2
E_gt2 = 0.14
Egt = ifelse(Tau_phplus.f1<20, E_gt1, E_gt2)
Eg = Egb + Egt
# Compute Qg (non-dimensional number)
Qg = Eg*h/lmp_h
# Compute Q
U_z = replicate(10000,runif(100,3.88,4.12))
U_h = U_z/(exp(alphaPZ*(z/h-1)))
#Compute E_B (Brownian diffusion)
L_obs = 0.0015
C_B = 0.887
Re_h = (U_h*L_obs)/(kin.vis)
E_B = C_B*(Sc^(-2/3))*(Re_h^(-1/2))
#Compute E_IN (Interception)
C_IN = 0.810
E_IN = C_IN*(dp/L_obs)
#Compute E_IM (Impaction)
C_IM = 0.162
beta_IM = 0.60
St_h = (Tau*U_h)/L_obs
E_IM = C_IM*(St_h/(St_h+beta_IM))^2
E_T = (U_h/u_star)*(E_B+E_IN+E_IM)+E_t
Q  = LAI*E_T*h/(lmp_h)
# Compute etaPZ
etaPZ = (alphaPZ^2/4+Q)^0.5
Vds = u_star*Eg*{(1+Q/Qg-alphaPZ/2)*tanh(etaPZ)/etaPZ}/{(1+Q+alphaPZ/2)*tanh(etaPZ)/etaPZ}
Vd = Vdrift+1/(Ra+1/Vds)
quantile(Vd, c(.05, 0.10, .50, 0.95))


#Dry deposition parameterization by Petroff and Zhang (2010)
#Uncertainty test: Deciduous forest
set.seed(5)
C1 = 0.2789
C2 = 3.115
C3 = 5.145*10^-11
C4 = -1.399
RH = replicate(10000,runif(100,0.76,0.84))
dp_a = 2.0
dp_i = dp_a*10^-6
rd = dp_i/2
```

```
r_w = {(C1*rd^C2)/(C3*rd^C4-log10(RH))+rd^3}^(1/3)
dp = r_w*2
#Correction factor, C
k_B = 1.38*10^-23
Temp = (273.15+25)
P = 101325
d_air = 3.7208*10^-10
lambda = (k_B*Temp)/(sqrt(2)*3.1416*P*d_air^2)
C = 1+(2*lambda/dp)*(1.257+0.4*exp(-0.55*dp/lambda))
dyn.vis = 1.8908*10^-5
rho = 1500
Tau = (rho*(dp)^2*C)/(18*dyn.vis)
Ws = Tau*9.81
Vphor = 0
Vdrift = Ws+Vphor
# Compute aerodynamic resistance (Ra):
z = 35
L = replicate(10000,runif(100,45,55))
x = z/L
# Compute stability function (shi_H)
shi_H.1 = 2*log(0.5*{1+(1-16*x)^0.5})
shi_H.2 = -5*x
shi_H =ifelse(x <= 0, shi_H.1 , shi_H.2)
zR = 50
z0 = replicate(10000,runif(100,1.125, 1.875))
u_star = replicate(10000,runif(100,0.54,0.66))
k_c = 0.41
Ra = (log(zR/z0)-shi_H)/(k_c*u_star)
# Compute surface resistance (Rs)
kin.vis = 1.6834*10^-5
D = (C*k_B*Temp)/(3*3.1416*dyn.vis*dp)
Sc = (kin.vis/D)
FSc = (Sc^(1/3))/2.9
Egb = (Sc^(-2/3)/14.5)*{1/6*log(1+FSc)^2/(1-FSc+FSc^2)+1/sqrt(3)*atan((2*FSc-1)/sqrt(3))
        +3.1416/6*sqrt(3)}^-1
cd = 1/6
kx = 0.216
LAI = replicate(10000,runif(100,9.5,10.5))
h = replicate(10000,runif(100,23.75,26.25))
d = replicate(10000,runif(100,12,20))
phi_H.1 = (1-16*x)^(-0.5)
phi_H.2 = 1+5*x
phi_H = ifelse(x<=0, phi_H.1, phi_H.2)
phi_M.1 = (1-16*x)^(-0.25)
phi_M.2 = 1+5*x
phi_M = ifelse(x<=0, phi_M.1, phi_M.2)
lmp = (0.41*(z-d))/(phi_H*(z-d)/abs(L))
lmp_h = (0.41*(h-d))/(phi_M*(h-d)/abs(L))
alphaPZ = {(kx*LAI)/(12*k_c^2*(1-d/h)^2)}^(1/3)*phi_M^(2/3)*{(h-d)/abs(L)}
C_IT = 0.056
Tau_phplus.1 = (Tau*u_star^2)/kin.vis
Tau_phplus.2 = C_IT
Tau_phplus = ifelse(Tau_phplus.1<20,Tau_phplus.1, Tau_phplus.2)
E_t.1 = 2.5*10^-3*C_IT*(Tau_phplus)^2
E_t.2 = C_IT
E_t = ifelse(Tau_phplus.1<20, E_t.1, E_t.2)
u_starf = u_star*exp(-alphaPZ)
Tau_phplus.f1 = (Tau*u_star^2)/kin.vis
Tau_phplus.f2 = 0.14
Tau_phplusf = ifelse(Tau_phplus.f1<20,Tau_phplus.f1, Tau_phplus.f2)
E_gt1 = 2.5*10^-3*0.14*(Tau_phplusf)^2
E_gt2 = 0.14
Egt = ifelse(Tau_phplus.f1<20, E_gt1, E_gt2)
Eg = Egb + Egt
# Compute Qg (non-dimensional number)
Qg = Eg*h/lmp_h
U_z = replicate(10000,runif(100,3.88,4.12))
U_h = U_z/(exp(alphaPZ*(z/h-1)))
#Compute E_B (Brownian diffusion)
L_obs = 0.03
C_B = 1.262
Re_h = (U_h*L_obs)/(kin.vis)
E_B = C_B*(Sc^(-2/3))*(Re_h^(-1/2))
#Compute E_IN (Interception)
```

```
C_IN = 0.216
E_IN = C_IN*(dp/L_obs)*(2+log(4*L_obs/dp))
#Compute E_IM (Impaction)
C_IM = 0.130
beta_IM = 0.47
St_h = (Tau*U_h)/L_obs
E_IM = C_IM*(St_h/(St_h+beta_IM))^2
E_T = (U_h/u_star)*(E_B+E_IN+E_IM)+E_t
Q  = LAI*E_T*h/(lmp_h)
# Compute etaPZ
etaPZ = (alphaPZ^2/4+Q)^0.5
Vds = u_star*Eg*{(1+Q/Qg-alphaPZ/2)*tanh(etaPZ)/etaPZ}/{(1+Q+alphaPZ/2)*tanh(etaPZ)/etaPZ}
Vd = Vdrift+1/(Ra+1/Vds)
quantile(Vd, c(.05, 0.10, .50, 0.95))


#Dry deposition parameterization by Petroff and Zhang (2010)
#Uncertainty test: Water
set.seed(5)
C1 = 0.2789
C2 = 3.115
C3 = 5.415*10^-11
C4 = -1.399
RH = replicate(10000,runif(100,0.76,0.84))
dp_i = 2.0
dp_d = dp_i*10^-6
rd = dp_d/2
r_w = {(C1*rd^C2)/(C3*rd^C4-log10(RH))+rd^3}^(1/3)
dp = r_w*2
k_B = 1.38*10^-23
Temp = 273.15+25
P = 101325
d_air = 3.7208*10^-10
lambda = (k_B*Temp)/(sqrt(2)*3.1416*P*d_air^2)
C = 1+(2*lambda/dp)*(1.257+0.4*exp(-0.55*dp/lambda))
dyn.vis = ((5*10^-8)*Temp)+4*10^-6
rho = 1500
Tau = (rho*(dp)^2*C)/(18*dyn.vis)
Ws = Tau*9.81
Vphor = (5*10^-3)/100
Vdrift = Ws+Vphor
# Compute aerodynamic resistance (Ra):
z = 8/100
L = replicate(10000,runif(100,45,55))
x = z/L
# Compute stability function (shi_H)
shi_H.1 = 2*log(0.5*{1+(1-16*x)^0.5})
shi_H.2 = -5*x
shi_H = ifelse(x <= 0, shi_H.1 , shi_H.2)
zR = 5
u_star = replicate(10000,runif(100,0.27,0.33))
z0_1 = 0.021*(u_star)^3.32
z0_2 = 0.00098*(u_star)^1.65
z0 = ifelse(u_star<= 0.16, z0_1, z0_2)
k_c = 0.41
Ra = (log(zR/z0)-shi_H)/(k_c*u_star)
# Compute surface resistance (Rs)
kin.vis = ((9*10^-8)*Temp)+10^-5
D = (C*k_B*Temp)/(3*3.1416*dyn.vis*dp)
Sc = (kin.vis/D)
FSc = (Sc^(1/3))/2.9
Egb = (Sc^(-2/3)/14.5)*{1/6*log(1+FSc)^2/(1-FSc+FSc^2)+1/sqrt(3)*atan((2*FSc-
1)/sqrt(3))+3.1416/6*sqrt(3)}^-1
Eg = Egb
Vd = Vdrift+1/(Ra+(1/(Eg*u_star)))
quantile(Vd, c(.05, 0.10, .50, 0.95))


#Dry deposition parameterization by Petroff and Zhang (2010)
#Uncertainty test: Ice/snow
set.seed(5)
C1 = 0.2789
C2 = 3.115
C3 = 5.415*10^-11
```

```
C4 = -1.399
RH = replicate(10000,runif(100,0.76,0.84))
dp_i = 2.0
dp_d = dp_i*10^-6
rd = dp_d/2
r_w = {(C1*rd^C2)/(C3*rd^C4-log10(RH))+rd^3}^(1/3)
dp = r_w*2
#Correction factor, C
k_B = 1.38*10^-23
Temp = 273.15+0
P = 101325
d_air = 3.7208*10^-10
lambda = (k_B*Temp)/(sqrt(2)*3.1416*P*d_air^2)
C = 1+(2*lambda/dp)*(1.257+0.4*exp(-0.55*dp/lambda))
dyn.vis = ((5*10^-8)*Temp)+4*10^-6
rho = 1500
Tau = (rho*(dp)^2*C)/(18*dyn.vis)
Ws = Tau*9.81
Vphor = (2*10^-4)/100
Vdrift = Ws+Vphor
# Compute aerodynamic resistance (Ra)
z = 5
L = replicate(10000,runif(100,45,55))
x = z/L
# Compute stability function (shi_H)
shi_H.1 = 2*log(0.5*{1+(1-16*x)^0.5})
shi_H.2 = -5*x
shi_H = ifelse(x <= 0, shi_H.1 , shi_H.2)
zR = 10
u_star = replicate(10000,runif(100,0.27,0.33))
z0 = replicate(10000,runif(100,0.0075,0.0125))
k_c = 0.41
Ra = (log(zR/z0)-shi_H)/(k_c*u_star)
# Compute surface resistance (Rs)
kin.vis = ((9*10^-8)*Temp)+10^-5
D = (C*k_B*Temp)/(3*3.1416*dyn.vis*dp)
Sc = (kin.vis/D)
FSc = (Sc^(1/3))/2.9
Egb = (Sc^(-2/3)/14.5)*{1/6*log(1+FSc)^2/(1-FSc+FSc^2)+1/sqrt(3)*atan((2*FSc-1)/sqrt(3))+3.1416/6*sqrt(3)}^-1
Eg = Egb
Vd = Vdrift+1/(Ra+(1/(Eg*u_star)))
quantile(Vd, c(.05, 0.10, .50, 0.95))
```

Codes for Monte Carlo uncertainty evaluation for Kouznetsov and Sofiev (2012) parameterization

```
#Dry deposition parameterization by Kouznetsov and Sofiev (2012)
#Uncertainty test: Grass
set.seed(5)
C1 = 0.2789
C2 = 3.115
C3 = 5.145*10^-11
C4 = -1.399
RH = replicate(10000,runif(100,0.76,0.84))
dp_a =2.0
dp_i = dp_a*10^-6
rd = dp_i/2
r_w = {(C1*rd^C2)/(C3*rd^C4-log10(RH))+rd^3}^(1/3)
dp = r_w*2
#Correction factor, C
k_B = 1.38*10^-23
Temp = (273.15+25)
P = 101325
d_air = 3.7208*10^-10
lambda = (k_B*Temp)/(sqrt(2)*3.1416*P*d_air^2)
C = 1+(2*lambda/dp)*(1.257+0.4*exp(-0.55*dp/lambda))
dyn.vis = 1.89*10^-5
rho = 1500
Tau = (rho*(dp)^2*C)/(18*dyn.vis)
V_s = Tau*9.81
u_star = replicate(10000,runif(100,0.27,0.33))
a  = 2*10^-3
kin.vis = 1.68*10^-5
D = (C*k_B*Temp)/(3*3.1416*dyn.vis*dp)
```

```
Sc = (kin.vis/D)
Re_star = (u_star*a)/kin.vis
# Compute V_diff (velocity for diffusion)
V_diff = 2*(Re_star^(-0.5))*Sc^(-2/3)
# Compute V_int (velocity for interception)
V_int = 80*u_star*((dp/a)^2)*(Re_star^(0.5))
# Compute V_imp
C_S = 0.003
C_R = 0.3
LAI = replicate(10000,runif(100,3.8,4.2))
CsCR = (C_S+C_R/LAI)^0.5
u.Uh = 0.3
u_star.by.U_h = min(u.Uh, CsCR)
#Compute Re_c
Re_c = ((u_star.by.U_h)^-1)^2*Re_star
# Calculate St
St = (Tau*u_star)/a
# Calculate St_e
St_e = St - Re_c^(-0.5)
eta_impSt.e1 = exp((-0.1/(St_e - 0.15 ) )-(1/sqrt(St_e -0.15)))
eta_impSt.e2 = 0
eta_impSt.e = ifelse(St_e>0.15,eta_impSt.e1,eta_impSt.e2)
V_imp = ((2*u_star.by.U_h)/u_star)*eta_impSt.e*(St-u_star.by.U_h*Re_star^-0.5)
# Dry deposition velocity
Vd = V_diff+V_int+V_imp+V_s
quantile(Vd, c(.05, 0.10, .50, 0.95))


#Dry deposition parameterization by Kouznetsov and Sofiev (2012)
#Uncertainty test: Coniferous forest
set.seed(5)
C1 = 0.2789
C2 = 3.115
C3 = 5.145*10^-11
C4 = -1.399
RH = replicate(10000,runif(100,0.76,0.84))
dp_a = 2.0
dp_i = dp_a*10^-6
rd = dp_i/2
r_w = {(C1*rd^C2)/(C3*rd^C4-log10(RH))+rd^3}^(1/3)
dp = r_w*2
#Correction factor, C
k_B = 1.38*10^-23
Temp =  273.15+25
P = 101325
d_air = 3.7208*10^-10
lambda = (k_B*Temp)/(sqrt(2)*3.1416*P*d_air^2)
C = 1+(2*lambda/dp)*(1.257+0.4*exp(-0.55*dp/lambda))
dyn.vis = 1.891*10^-5
rho = 1500
Tau = (rho*(dp)^2*C)/(18*dyn.vis)
V_s = Tau*9.81
u_star = replicate(10000,runif(100,0.27,0.33))
a  =  0.7*10^-3
kin.vis = 1.683*10^-5
D = (C*k_B*Temp)/(3*3.1416*dyn.vis*dp)
Sc = (kin.vis/D)
Re_star = (u_star*a)/kin.vis
# Compute V_diff (velocity for diffusion)
V_diff = 2*(Re_star^(-0.5))*Sc^(-2/3)
# Compute V_int (velocity for interception)
V_int = 80*u_star*((dp/a)^2)*(Re_star^(0.5))
# Compute V_imp
C_S = 0.003
C_R = 0.3
LAI = replicate(10000,runif(100,5.7,6.3))
CsCR = (C_S+C_R/LAI)^0.5
u.Uh = 0.3
u_star.by.U_h = min(u.Uh, CsCR)
#Compute Re_c
Re_c = ((u_star.by.U_h)^-1)^2*Re_star
# Calculate St
St = (Tau*u_star)/a
# Calculate St_e
```

```
St_e = St - Re_c^(-0.5)
eta_impSt.e1 = exp((-0.1/(St_e - 0.15 ) )-(1/sqrt(St_e -0.15)))
eta_impSt.e2 = 0
eta_impSt.e = ifelse(St_e>0.15,eta_impSt.e1,eta_impSt.e2)
V_imp = ((2*u_star.by.U_h)/u_star)*eta_impSt.e*(St-u_star.by.U_h*Re_star^-0.5)
# Dry deposition velocity
Vd = V_diff+V_int+V_imp+V_s
quantile(Vd, c(.05, 0.10, .50, 0.95))


#Dry deposition parameterization by Kouznetsov and Sofiev (2012)
#Uncertainty test: Deciduous forest
set.seed(5)
C1 = 0.2789
C2 = 3.115
C3 = 5.145*10^-11
C4 = -1.399
RH = replicate(10000,runif(100,0.76,0.84))
dp_a = 2.0
dp_i = dp_a*10^-6
rd = dp_i/2
r_w = {(C1*rd^C2)/(C3*rd^C4-log10(RH))+rd^3}^(1/3)
dp = r_w*2
#Correction factor, C
k_B = 1.38*10^-23
Temp = (273.15+25)
P = 101325
d_air = 3.7208*10^-10
lambda = (k_B*Temp)/(sqrt(2)*3.1416*P*d_air^2)
C = 1+(2*lambda/dp)*(1.257+0.4*exp(-0.55*dp/lambda))
dyn.vis = 1.89*10^-5
rho = 1500
Tau = (rho*(dp)^2*C)/(18*dyn.vis)
V_s = Tau*9.81
u_star = replicate(10000,runif(100,0.27,0.33))
a  = 7*10^-3
kin.vis = 1.68*10^-5
D = (C*k_B*Temp)/(3*3.1416*dyn.vis*dp)
Sc = (kin.vis/D)
Re_star = (u_star*a)/kin.vis
# Compute V_diff (velocity for diffusion)
V_diff = 2*(Re_star^(-0.5))*Sc^(-2/3)
# Compute V_int (velocity for interception)
V_int = 80*u_star*((dp/a)^2)*(Re_star^(0.5))
# Compute V_imp
C_S = 0.003
C_R = 0.3
LAI = replicate(10000,runif(100,9.5,10.5))
CsCR = (C_S+C_R/LAI)^0.5
u.Uh = 0.3
u_star.by.U_h = min(u.Uh, CsCR)
#Compute Re_c
Re_c = ((u_star.by.U_h)^-1)^2*Re_star
# Calculate St
St = (Tau*u_star)/a
# Calculate St_e
St_e = St - Re_c^(-0.5)
eta_impSt.e1 = exp((-0.1/(St_e - 0.15 ) )-(1/sqrt(St_e -0.15)))
eta_impSt.e2 = 0
eta_impSt.e = ifelse(St_e>0.15,eta_impSt.e1,eta_impSt.e2)
V_imp = ((2*u_star.by.U_h)/u_star)*eta_impSt.e*(St-u_star.by.U_h*Re_star^-0.5)
# Dry deposition velocity
Vd = V_diff+V_int+V_imp+V_s
quantile(Vd, c(.05, 0.10, .50, 0.95))


#Dry deposition parameterization by Kouznetsov and Sofiev (2012)
#Uncertainty test: Smooth surface (water)
set.seed(5)
C1 = 0.2789
C2 = 3.115
C3 = 5.415*10^-11
C4 = -1.399
dp_i = 0.5
```

```
dp = dp_i*10^-6
rho = 1500
RH = replicate(10000,runif(100,0.76,0.84))
Temp = 25+273.15
u_star = replicate(10000,runif(100,0.27,0.33))
z0_1 = 0.021*(u_star)^3.32
z0_2 = 0.00098*(u_star)^1.65
z0 = ifelse(u_star<= 0.16, z0_1, z0_2)
L = replicate(10000,runif(100,45,55))
kin.vis = ((9*10^-8)*Temp)+10^-5
dyn.vis = ((5*10^-8)*Temp)+4*10^-6
d_air = 3.7208*10^-10
k_B = 1.38*10^-23
P = 101325
lambda = (k_B*Temp)/(sqrt(2)*3.1416*P*d_air^2)
C = 1+((2*lambda/dp)*(1.257+0.4*exp(-0.55*dp/lambda)))
tau_p = (rho*(dp)^2*C)/(18*dyn.vis)
D = (C*k_B*Temp)/(3*3.1416*dyn.vis*dp)
v_s = 9.81*tau_p
taup = (tau_p*u_star^2)/(kin.vis)
Sc = (kin.vis/D)
vsplus = v_s/u_star
rplus = (dp*u_star)/(2*kin.vis)
R_s = 0
Rsplus = u_star*R_s
z_meas = 8/100
zpmax = (z_meas*u_star)/(kin.vis)
MOplus = (kin.vis)/(u_star*L)
#Fixed parameters
Zbuf = 3
Ztf = 18                              #turbophoretic sublayer height
taultf = 5                            #Lagrangian time in turbophoretic layer
Nutp_Ztf= (0.4*(Ztf)^3)/(Ztf^2+200)  #Dimensionless eddy viscosity of air
It_Ztf = (2.5*log10(Ztf ))-(100/Ztf^2)
It_Zbuf = (2.5*log10(Zbuf))-(100/Zbuf^2)
S = Sc^(1/3)
Zl = 20/S
fTmp = 2.5/Sc
fTmp1 = (fTmp^3/27+(fTmp*(100+5*sqrt(8*fTmp/27)+400)))^(1/3)
Zl_1 = fTmp1+((fTmp*fTmp)/(9*fTmp1))+(1/3)*fTmp
fTmp_1 = (Zl_1^2)/(Zl_1^2+200)
fTmp_2 = 1.2*(fTmp_1)-0.8*fTmp_1^2
fTmp1_1 = 1/(Sc*fTmp_2)
fTmp_3 = 1/Sc
x_1 = Zl_1 - fTmp1_1
x_2 = Zl_1
x_3 = Zl_1+fTmp1_1
Nutp_x_1 = (0.4*x_1^3)/(x_1^2+200)
Nutp_x_2 = (0.4*x_2^3)/(x_2^2+200)
Nutp_x_3 = (0.4*x_3^3)/(x_3^2+200)
fIvd = Rsplus+(Zl_1-
fTmp1_1)*Sc+0.3333*fTmp1_1*(1/fTmp_3+((0.4*x_1^3)/(x_1^2+200)))+4/(fTmp_3+((0.4*x_2^3)/(x_2^2+200)))+1/(fTmp_3+(
(0.4*x_3^3)/(x_3^2+200)))
x_4 = zpmax
x_5 = Zl_1 + fTmp1_1
It_x_4 = 2.5*log10(x_4) - 100/(x_4^2)
It_x_5 = 2.5*log10(x_5) - 100/(x_5^2)
#Now calculate fu_Psi(zpmax*MOplus)
s = 2.35*(zpmax*MOplus+abs(zpmax*MOplus))
u1 = 0.5*(abs(zpmax*MOplus)-zpmax*MOplus)
u = -4*u1/(2.65*sqrt(u1*sqrt(u1))+1)
fu_Psi = s+u
fTmp_4 = It_x_4 - It_x_5 + 2.5*fu_Psi
fIvd_1 = fIvd+0.5*(fTmp_4+abs(fTmp_4))
fu_vdplus_smooth_1 = 1/fIvd_1
Il_input1= Zl*S/7.92
Il_input2= rplus*S/7.92
Il_1 = -0.16667*log10(Il_input1^2-Il_input1+1)+0.57735*atan((2*Il_input1-1)*0.57735)+0.3333*log10(Il_input1+1)
Il_2 = -0.16667*log10(Il_input2^2-Il_input2+1)+0.57735*atan((2*Il_input2-1)*0.57735)+0.3333*log10(Il_input2+1)
R = 7.92*S^2*(Il_1-Il_2)
R_1 = 0.5*(R+abs(R))
Zl_2 = 0.5*(rplus+Zl+abs(rplus-Zl))
It_Zbuf = 2.5*log10(Zbuf) - (100/(Zbuf^2))
It_Zl_2 = 2.5*log10(Zl_2) - (100/(Zl_2^2))
```

```
R1 = It_Zbuf - It_Zl_2
R_2 = R_1+0.5*(R1+abs(R1))
fTmp_5 = vsplus*R_2
fTmp_6 = exp(-fTmp_5)
fIvd_2  = Rsplus*fTmp_6+(1-fTmp_6)/vsplus
fIvd_3 = Rsplus+R_2
fIvd_23 = ifelse(abs(fTmp_5)>0.001, fIvd_2, fIvd_3 )
V  = 0.81*taup/(Ztf-Zbuf)/(1+taup/taultf) + vsplus   #chcek for sign of vsplus
It_Ztf = 2.5*log10(Ztf) - 100/(Ztf^2)
R_3  = ((It_Ztf)-(It_Zbuf))*(1+taup/taultf)
fTmp_7 = V*R_3
fTmp_8 = exp(-fTmp_7)
fIvd_4 = (fIvd_23*fTmp_8)+(1-fTmp_8)/V
fIvd_5 = fIvd_3+R_3
fIvd_45 = ifelse(abs(fTmp_7)>0.001, fIvd_4, fIvd_5)
#Now calculations for the Lagrangian turbophoretic layer
Ztf2 = 2*taup
V_1 = 0.4+vsplus
R_4 =
0.1667*((1+taup/(0.5*Ztf))/(Nutp_Ztf)+4*(1+taup/(0.25*(Ztf+Ztf2)))/((0.4*(0.5*(Ztf+Ztf2))^3)/((0.5*(Ztf+Ztf2))^2
+200))+(1+taup/(0.5*Ztf2))/((0.4*Ztf2 ^3)/(Ztf2^2+200)))*(Ztf2-Ztf)
fTmp_9 = V_1*R_4
fTmp_10 = exp(-fTmp_9)
fIvd_6 = fIvd_45*fTmp_10 + (1-fTmp_10)/V_1
fIvd_7 = fIvd_45+R_3
fIvd_67 = ifelse(abs(fTmp_9) > 0.001, fIvd_6, fIvd_7)
Ztf2_2 = Ztf
fIvd67_Ztf2 = ifelse(Ztf2>Ztf, fIvd_67, Ztf2_2)
#Following calculations are for aerodynamic layer
It_zpmax = 2.5*log10(zpmax) - 100/(zpmax^2)
Ztf22 = ifelse(Ztf2>Ztf, Ztf2, Ztf2_2)
It_Ztf22  = 2.5*log10(Ztf22) - 100/(Ztf22^2)
R_5 = It_zpmax - It_Ztf22  + 2.5*fu_Psi
R_6 = 0.5*(R_5+abs(R_5))
fTmp_11  = vsplus*R_6
fTmp_12 = exp(-fTmp_11)
fIvd_4567 = ifelse(Ztf2>Ztf, fIvd_67,fIvd_45)
fIvd_8 = (fIvd_4567* fTmp_12)+(1-fTmp_12)/vsplus
fIvd_9 = fIvd_4567 + R_6
fIvd_10 = ifelse(abs(fTmp_11) > 0.001, fIvd_8, fIvd_9)
fu_vdplus_smooth_3 = 1/fIvd_10
fu_vdplus_smooth = ifelse(Zl>Zbuf, fu_vdplus_smooth_1, fu_vdplus_smooth_3)
Vd_smooth = fu_vdplus_smooth*u_star
quantile(Vd_smooth, c(.05, 0.10, .50, 0.95))


#Dry deposition parameterization by Kouznetsov and Sofiev (2012)
#Uncertainty test: Ice/snow
set.seed(5)
C1 = 0.2789
C2 = 3.115
C3 = 5.415*10^-11
C4 = -1.399
RH = replicate(10000,runif(100,0.76,0.84))
dp_a = 2.0
dp_i = dp_a*10^-6
rd = dp_i/2
r_w = {(C1*rd^C2)/(C3*rd^C4-log10(RH))+rd^3}^(1/3)
dp = r_w*2
#Correction factor, C
k_B = 1.38*10^-23
Temp = 273.15+0
P = 101325
d_air = 3.7208*10^-10
lambda = (k_B*Temp)/(sqrt(2)*3.1416*P*d_air^2)
C = 1+(2*lambda/dp)*(1.257+0.4*exp(-0.55*dp/lambda))
dyn.vis = ((5*10^-8)*Temp)+4*10^-6
rho =  1500
Tau = (rho*(dp)^2*C)/(18*dyn.vis)
V_s = Tau*9.81
# Need to compute Sc, Re_star
u_star = replicate(10000,runif(100,0.27,0.33))
#u_star = replicate(10000,runif(100,0.27,0.33))
a  =  0.5*10^-3
```

```
kin.vis = ((9*10^-8)*Temp)+10^-5
D = (C*k_B*Temp)/(3*3.1416*dyn.vis*dp)
Sc = (kin.vis/D)
Re_star = (u_star*a)/kin.vis
# Compute V_diff (velocity for diffusion)
V_diff = 2*(Re_star^(-0.5))*Sc^(-2/3)
# Compute V_int (velocity for interception)
V_int = 80*u_star*((dp/a)^2)*(Re_star^(0.5))
# Compute V_imp
C_S = 0.003
C_R = 0.3
LAI = 0
CsCR = (C_S+C_R/LAI)^0.5
u.Uh = 0.3
u_star.by.U_h = min(u.Uh, CsCR)
#Compute Re_c
Re_c = ((u_star.by.U_h)^-1)^2*Re_star
# Calculate St
St = (Tau*u_star)/a
# Calculate St_e
St_e = St - Re_c^(-0.5)
eta_impSt.e1 = exp((-0.1/(St_e - 0.15 ) )-(1/sqrt(St_e -0.15)))
eta_impSt.e2 = 0
eta_impSt.e = ifelse(St_e>0.15,eta_impSt.e1,eta_impSt.e2)
V_imp = (2*u_star.by.U_h*eta_impSt.e*(St-(u_star.by.U_h*Re_star^-0.5)))*u_star
# Dry deposition velocity
Vd = V_diff+V_int+V_imp+V_s
quantile(Vd, c(.05, 0.10, .50, 0.95))
```

Codes for Monte Carlo uncertainty evaluation for Zhang and He (2014) parameterization

```
#Dry deposition parameterization by Zhang and He (2014)
#Uncertainty test: Grass
set.seed(5)
C1 = 0.2789
C2 = 3.115
C3 = 5.145*10^-11
C4 = -1.399
RH = replicate(10000,runif(100,0.76,0.84))
dp_a = 2.0
dp_i = dp_a*10^-6
rd = dp_i/2
r_w = {(C1*rd^C2)/(C3*rd^C4-log10(RH))+rd^3}^(1/3)
dp = r_w*2
#Correction factor, C
k_B = 1.38*10^-23
Temp = (273.15+25)
P = 101325
d_air = 3.7208*10^-10
lambda = (k_B*Temp)/(sqrt(2)*3.1416*P*d_air^2)
C = 1+(2*lambda/dp)*(1.257+0.4*exp(-0.55*dp/lambda))
dyn.vis = 1.89*10^-5
rho = 1500
Tau = (rho*(dp)^2*C)/(18*dyn.vis)
V_g = Tau*9.81
Rg = 1/V_g
u_star = replicate(10000,runif(100,0.27,0.33))
a1 = 5.4*10^-3
z = 5
L = replicate(10000,runif(100,45,55))
x = z/L
# Compute stability function (shi_H)
shi_H.1 = 2*log(0.5*{1+(1-16*x)^0.5})
shi_H.2 = -5*x
shi_H =ifelse(x <= 0, shi_H.1 , shi_H.2)
zR = 3.5
z0 = replicate(10000,runif(100,0.03,0.05))
k_c = 0.41
Ra = (log(zR/z0)-shi_H)/(k_c*u_star)
# Calculate Vds = 1/Rs
# For PM2.5
Vds_PM2.5 = (a1*u_star)
Rds_PM2.5 = (1/Vds_PM2.5)
```

```
#For PM2.5-10
#b1=
#b2=
#b3 =
#c1=
#c2 =
#c3 =
#k = c1*u_star+c2*u_star^2+c3*u_star^3
#LAI = replicate(10000,runif(100,3.8,4.2))
#LAImax =
#Vds_PM10 = (b1*u_star+b2*u_star^2+b3*u_star^3)*exp(k*(LAI/LAImax)-1)
#Vds_PM10 = (b1*u_star+b2*u_star^2+b3*u_star^3)
#Rds_PM10 = 1/Vds_PM10
# For PM10+
#d1=
#d2=
#d3 =
#f1=
#f2 =
#f3 =
#k = f1*u_star+f2*u_star^2+f3*u_star^3
#LAI =
#LAImax =
#Vds_10plus = (d1*u_star+d2*u_star^2+d3*u_star^3)*exp(k*(LAI/LAImax)-1)
#Vds_10plus = (d1*u_star+d2*u_star^2+d3*u_star^3)
#Rds_PM2.5 = (1/Vds_PM2.5)
#Compute Vd
Vd = 1/Rg+(1/(Ra+Rds_PM2.5))
quantile(Vd, c(.05, 0.10, .50, 0.95))


#Dry deposition parameterization by Zhang and He (2014)
#Uncertainty test: Coniferous forest
set.seed(5)
C1 = 0.2789
C2 = 3.115
C3 = 5.145*10^-11
C4 = -1.399
RH = replicate(10000,runif(100,0.76,0.84))
dp_a = 2.0
dp_i = dp_a*10^-6
rd = dp_i/2
r_w = {(C1*rd^C2)/(C3*rd^C4-log10(RH))+rd^3}^(1/3)
dp = r_w*2
#Correction factor, C
k_B = 1.38*10^-23
Temp = (273.15+25)
P = 101325
d_air = 3.7208*10^-10
lambda = (k_B*Temp)/(sqrt(2)*3.1416*P*d_air^2)
C = 1+(2*lambda/dp)*(1.257+0.4*exp(-0.55*dp/lambda))
dyn.vis = 1.89*10^-5
rho = 1500
Tau = (rho*(dp)^2*C)/(18*dyn.vis)
V_g = Tau*9.81
Rg = 1/V_g
u_star = replicate(10000,runif(100,0.27,0.33))
a1 = 4.3*10^-3
z = 35
L = replicate(10000,runif(100,45,55))
x = z/L
# Compute stability function (shi_H)
shi_H.1 = 2*log(0.5*{1+(1-16*x)^0.5})
shi_H.2 = -5*x
shi_H =ifelse(x <= 0, shi_H.1 , shi_H.2)
zR = 30
z0 = replicate(10000,runif(100,0.9,1.5))
k_c = 0.41
Ra = (log(zR/z0)-shi_H)/(k_c*u_star)
# Calculate Vds = 1/Rs
# For PM2.5
Vds_PM2.5 = (a1*u_star)
Rds_PM2.5 = (1/Vds_PM2.5)
#For PM2.5-10
```

```
#b1=
#b2=
#b3 =
#c1=
#c2 =
#c3 =
#k = c1*u_star+c2*u_star^2+c3*u_star^3
#LAI =
#LAImax =
#Vds_PM10 =
(b1*u_star+b2*u_star^2+b3*u_star^3)*exp(k*(LAI/LAImax)-
1)
#Vds_PM10 = (b1*u_star+b2*u_star^2+b3*u_star^3)
#Rds_PM10 = 1/Vds_PM10
# For PM10+
#d1=
#d2=
#d3 =
#f1=
#f2 =
#f3 =
#k = f1*u_star+f2*u_star^2+f3*u_star^3
#LAI =
#LAImax =
#Vds_10plus =
(d1*u_star+d2*u_star^2+d3*u_star^3)*exp(k*(LAI/LAImax)-
1)
#Vds_10plus = (d1*u_star+d2*u_star^2+d3*u_star^3)
Rds_PM10plus = (1/Vds_PM2.5)
#Compute Vd
Vd = 1/Rg+(1/(Ra+Rds_PM2.5))
quantile(Vd, c(.05, 0.10, .50, 0.95))


#Dry deposition parameterization by Zhang and He (2014)
#Uncertainty test: Deciduous forest
set.seed(5)
C1 = 0.2789
C2 = 3.115
C3 = 5.145*10^-11
C4 = -1.399
RH = replicate(10000,runif(100,0.76,0.84))
dp_a = 2.0
dp_i = dp_a*10^-6
rd = dp_i/2
r_w = {(C1*rd^C2)/(C3*rd^C4-log10(RH))+rd^3}^(1/3)
dp = r_w*2
#Correction factor, C
k_B = 1.38*10^-23
Temp = (273.15+25)
P = 101325
d_air = 3.7208*10^-10
lambda = (k_B*Temp)/(sqrt(2)*3.1416*P*d_air^2)
C = 1+(2*lambda/dp)*(1.257+0.4*exp(-0.55*dp/lambda))
dyn.vis = 1.89*10^-5
rho = 1500
Tau = (rho*(dp)^2*C)/(18*dyn.vis)
V_g = Tau*9.81 # m/s
Rg = 1/V_g
u_star = replicate(10000,runif(100,0.54,0.66))
a1 = 4.3*10^-3
z = 35
L = replicate(10000,runif(100,45,55))
x = z/L
# Compute stability function (shi_H)
shi_H.1 = 2*log(0.5*{1+(1-16*x)^0.5})
shi_H.2 = -5*x
shi_H =ifelse(x <= 0, shi_H.1 , shi_H.2)
zR = 50
z0 = replicate(10000,runif(100,1.125, 1.875))
k_c = 0.41
Ra = (log(zR/z0)-shi_H)/(k_c*u_star)
#Calculate Vds = 1/Rs
# For PM2.5
```

```
Vds_PM2.5 = (a1*u_star)
Rds_PM2.5 = (1/Vds_PM2.5)
# For PM2.5-10
#b1=
#b2=
#b3 =
#c1=
#c2 =
#c3 =
#k = c1*u_star+c2*u_star^2+c3*u_star^3
#LAI =
#LAImax =
#Vds_PM10 =
(b1*u_star+b2*u_star^2+b3*u_star^3)*exp(k*(LAI/LAImax)-
1)
#Rds_PM10 = 1/Vds_PM10
# For PM10+
#d1= -2.2
#d2= 3.9*10^1
#d3 = -6.7
#f1= 6.2
#f2 = -1.2*10^1
#f3 = 6.1
#k = f1*u_star+f2*u_star^2+f3*u_star^3
#LAI =
#LAImax =
#Vds_10plus =
(d1*u_star+d2*u_star^2+d3*u_star^3)*exp(k*(LAI/LAImax)-
1)
#Rds_10plus = (1/Vds_10plus)
#Compute Vd
Vd = 1/Rg+(1/(Ra+Rds_PM2.5))
quantile(Vd, c(.05, 0.10, .50, 0.95))


#Dry deposition parameterization by Zhang and He (2014)
#Uncertainty test: Water
set.seed(5)
C1 = 0.2789
C2 = 3.115
C3 = 5.415*10^-11
C4 = -1.399
RH = replicate(10000,runif(100,0.76,0.84))
dp_a =2.0
dp_i = dp_a*10^-6
rd = dp_i/2
r_w = {(C1*rd^C2)/(C3*rd^C4-log10(RH))+rd^3}^(1/3)
dp = r_w*2
#Correction factor, C
k_B = 1.38*10^-23
Temp = 273.15+25
P = 101325
d_air = 3.7208*10^-10
lambda = (k_B*Temp)/(sqrt(2)*3.1416*P*d_air^2)
C = 1+(2*lambda/dp)*(1.257+0.4*exp(-0.55*dp/lambda))
dyn.vis = ((5*10^-8)*Temp)+4*10^-6
rho = 1500
Tau = (rho*(dp)^2*C)/(18*dyn.vis)
V_g = Tau*9.81
Rg = 1/V_g
a1 = 6.9*10^-3
z = 8/100
L = replicate(10000,runif(100,45,55))
x = z/L
# Compute stability function (shi_H)
shi_H.1 = 2*log(0.5*{1+(1-16*x)^0.5})
shi_H.2 = -5*x
shi_H =ifelse(x <= 0, shi_H.1 , shi_H.2)
zR = 5
u_star = replicate(10000,runif(100,0.27,0.33))
z0_1 = 0.021*(u_star)^3.32
z0_2 = 0.00098*(u_star)^1.65
z0 = ifelse(u_star<= 0.16, z0_1, z0_2)
k_c = 0.41
```

```
Ra = (log(zR/z0)-shi_H)/(k_c*u_star)
# Calculate Vds = 1/Rs
#For PM2.5
Vds_PM2.5 = (a1*u_star)
Rds_PM2.5 = (1/Vds_PM2.5)
# For PM2.5-10
#b1= 2.6*10^-1
#b2= -1.3*10^0
#b3 = 3.0*10^0
#c1= 1.8
#c2 = -2.0*10^-1
#c3 = -5.3*10^-1
#k = c1*u_star+c2*u_star^2+c3*u_star^3
#LAI =
#LAImax =
#Vds_PM10 =
(b1*u_star+b2*u_star^2+b3*u_star^3)*exp(k*(LAI/LAImax)-
1)
#Rds_PM10 = 1/Vds_PM10
#For PM10+
#d1= -
#d2=
#d3 =
#f1=
#f2 =
#f3 =
#k = f1*u_star+f2*u_star^2+f3*u_star^3
#LAI =
#LAImax =
#Vds_PM10Plus =
(d1*u_star+d2*u_star^2+d3*u_star^3)*exp(k*(LAI/LAImax)-
1)
#Rds_PM10Plus = (1/Vds_PM10Plus)
#Compute Vd
Vd = 1/Rg+(1/(Ra+Rds_PM2.5))
quantile(Vd, c(.05, 0.10, .50, 0.95))


#Dry deposition parameterization by Zhang and He (2014)
#Uncertainty test: Ice/Water
set.seed(5)
C1 = 0.2789
C2 = 3.115
C3 = 5.415*10^-11
C4 = -1.399
RH = replicate(10000,runif(100,0.76,0.84))
dp_a = 2.0
dp_i = dp_a*10^-6
rd = dp_i/2
r_w = {(C1*rd^C2)/(C3*rd^C4-log10(RH))+rd^3}^(1/3)
dp = r_w*2
#Correction factor, C
k_B = 1.38*10^-23
Temp = 273.15+0
P = 101325
d_air = 3.7208*10^-10
lambda = (k_B*Temp)/(sqrt(2)*3.1416*P*d_air^2)
C = 1+(2*lambda/dp)*(1.257+0.4*exp(-0.55*dp/lambda))
dyn.vis = ((5*10^-8)*Temp)+4*10^-6
rho = 1500
Tau = (rho*(dp)^2*C)/(18*dyn.vis)
V_g = Tau*9.81
Rg = 1/V_g
a1 = 4.3*10^-3
z = 5
L = replicate(10000,runif(100,45,55))
x = z/L
# Compute stability function (shi_H)
shi_H.1 = 2*log(0.5*{1+(1-16*x)^0.5})
shi_H.2 = -5*x
shi_H =ifelse(x <= 0, shi_H.1 , shi_H.2)
zR = 10
u_star = replicate(10000,runif(100,0.27,0.33))
z0 = replicate(10000,runif(100,0.0075,0.0125))
```

```
k_c = 0.41
Ra = (log(zR/z0)-shi_H)/(k_c*u_star)
# Calculate Vds = 1/Rs#For PM2.5
Vds_PM2.5 = (a1*u_star)
Rds_PM2.5 = (1/Vds_PM2.5)
# For PM2.5-10
#b1=
#b2=
#b3 =
#c1=
#c2 =
#c3 =
#k = c1*u_star+c2*u_star^2+c3*u_star^3
#LAI =
# LAImax =
#Vds_PM10 = (b1*u_star+b2*u_star^2+b3*u_star^3)*exp(k*(LAI/LAImax)-1)
#Rds_PM10 = 1/Vds_PM10
# For PM10+
#d1=
#d2=
#d3 =
#f1=
#f2 =
#f3 =
#k = f1*u_star+f2*u_star^2+f3*u_star^3
#LAI =
#LAImax =
#Vds_PM10Plus = (d1*u_star+d2*u_star^2+d3*u_star^3)*exp(k*(LAI/LAImax)-1)
#Rds_PM10Plus = (1/Vds_PM10Plus)
#Compute Vd
Vd = 1/Rg+(1/(Ra+Rds_PM2.5))
quantile(Vd, c(.05, 0.10, .50, 0.95))
```

Codes for Monte Carlo uncertainty evaluation for Zhang and Shao (2014) parameterization

```
#Dry deposition parameterization by Zhang and Shao (2014)
#Uncertainty test: Plant (Grass, coniferous, and deciduous forests)
set.seed(5)
C1 = 0.2789
C2 = 3.115
C3 = 5.145*10^-11
C4 = -1.399
RH = replicate(10000,runif(100,0.76,0.84))
dp_a = 2.0
dp_i = dp_a*10^-6
rd = dp_i/2
r_w = {(C1*rd^C2)/(C3*rd^C4-log10(RH))+rd^3}^(1/3)
dp = r_w*2
#Correction factor, C
k_B = 1.38*10^-23
Temp = (273.15+15)
P = 101325
d_air = 3.7208*10^-10
lambda = (k_B*Temp)/(sqrt(2)*3.1416*P*d_air^2)
C = 1+(2*lambda/dp)*(1.257+0.4*exp(-0.55*dp/lambda))
dyn.vis = 1.841*10^-5
rho = 1500
Tau = (rho*(dp_i)^2*C)/(18*dyn.vis)
Tau_wet = (rho*(dp)^2*C)/(18*dyn.vis)
Wt = Tau*9.81
Vg = Wt
Vg_wet = Tau_wet*9.81
u_star = replicate(10000,runif(100,0.27,0.33))
k =  0.41
z =  1
zd = 0.20
h_c = 0.23
z0 = replicate(10000,runif(100,0.0015, 0.0025))
B1 = 0.45
Sc_T = (1+(Vg^2/u_star^2))^0.5
Ra = (Sc_T/(k*u_star))*(log((z-zd)/(h_c-zd)))
Rg = 1/Vg
U_h = replicate(10000,runif(100,3.88,4.12))
```

```
kin.vis = 1.593*10^-5
d_c = 0.005
D = (C*k_B*Temp)/(3*3.1416*dyn.vis*dp)
Sc = (kin.vis/D)
Re_h = (U_h*d_c)/(kin.vis)
nB = 0.5
C_B = 0.467
E_B = C_B*Sc^(-2/3)*Re_h^(nB-1)
#Compute impaction collection efficiency (E_IM)
beta_IM = 0.6
St_h = (Tau*u_star)/d_c
E_IM = (St_h/(St_h+beta_IM))^2
#Compute interception efficiency (E_IN)
Ain = 150
E_IN = Ain*u_star*(10^(-St_h))*(2*dp/d_c)
#Compute R
b = 2
R = exp(-b*sqrt(St_h))
#Compute w_dm
B2 = 3
w_dm = (u_star/U_h*h_c)       #For rough surface
#compute Tau_c/Tau (ratio of stress)
Beta = 200
C1 = 6
C2 = 0.1
lambda_FAI = 0.4
n_FAI = (lambda_FAI)/(h_c*d_c)
q = (3.1416*d_c^2)/4
eta_BAI = n_FAI*q
lambda_FAIe = ((lambda_FAI)/(1-eta_BAI)^C2)*exp((-C1*lambda_FAI)/(1-eta_BAI)^C2)
Tau_c_BY_Tau = (Beta*lambda_FAIe)/(1+Beta*lambda_FAIe)
#Compute Rs
E = E_B+E_IM+E_IN
Tau_wetplus = (Tau_wet*u_star^2)/kin.vis
Cd = 1/6
Rs = (R*w_dm*((E*Tau_c_BY_Tau/Cd)+(1+Tau_c_BY_Tau)*Sc^-1+10^(-3/Tau_wetplus))+Vg_wet)^-1
#Compute Vd
Vd = (Rg+((Rs-Rg)/exp(Ra/Rg)))^-1
quantile(Vd, c(.05, 0.10, .50, 0.95))


#Dry deposition parameterization by Zhang and Shao (2014)
#Uncertainty test: Water
set.seed(5)
C1 = 0.2789
C2 = 3.115
C3 = 5.415*10^-11
C4 = -1.399
RH = replicate(10000,runif(100,0.76,0.84))
dp_a = 2.0
dp_i = dp_a*10^-6
rd = dp_i/2
r_w = {(C1*rd^C2)/(C3*rd^C4-log10(RH))+rd^3}^(1/3)
dp = r_w*2
#Correction factor, C
k_B = 1.38*10^-23
Temp = 273.15+25
P = 101325
d_air = 3.7208*10^-10
lambda = (k_B*Temp)/(sqrt(2)*3.1416*P*d_air^2)
C = 1+(2*lambda/dp)*(1.257+0.4*exp(-0.55*dp/lambda))
dyn.vis = ((5*10^-8)*Temp)+4*10^-6
rho = 1500
Tau = (rho*(dp_i)^2*C)/(18*dyn.vis)
Tau_wet = (rho*(dp)^2*C)/(18*dyn.vis)
Wt = Tau*9.81
Vg = Wt
Vg_wet = Tau_wet*9.81
u_star = replicate(10000,runif(100,0.27,0.33))
k =  0.41
z0 = 0.3/1000
z = 8/100
U_h = replicate(10000,runif(100,4.85,5.15))
zd = 0
```

```
h_c = 30*z0
B1 = 0.45
Sc_T = (1+(Vg^2/u_star^2))^0.5
Ra = (B1*Sc_T/k*u_star)*log(z/z0)          # For smooth surface
Rg = 1/Vg
# Calculate surface resistance (Rs)
kin.vis = ((9*10^-8)*Temp)+10^-5
d_c = 0.005
D = (C*k_B*Temp)/(3*3.1416*dyn.vis*dp)
Sc = (kin.vis/D)
Re_h = (U_h*d_c)/(kin.vis)
nB = 0.5
C_B = 0.467
E_B = C_B*Sc^(-2/3)*Re_h^(nB-1)
#Compute impaction collection efficiency (E_IM)
beta_IM = 0.6
St_h = (Tau*u_star)/d_c
E_IM = (St_h/(St_h+beta_IM))^2
#Compute interception efficiency (E_IN)
Ain = 100
E_IN = Ain*u_star*(10^(-St_h))*(2*dp/d_c)
#Compute R
b = 2
R = exp(-b*sqrt(St_h))
#Compute w_dm
B2 = 3
w_dm = B2*u_star                           #For smooth surface
#compute Tau_c/Tau (ratio of stress)
Beta = 200
C_1 = 6
C_2 = 0.1
lambda_FAI = 0.538
n_FAI = (lambda_FAI)/(h_c*d_c)
q = (3.1416*d_c^2)/4
eta_BAI = n_FAI*q
lambda_FAIe = ((lambda_FAI)/(1-eta_BAI)^C_2)*exp((-C_1*lambda_FAI)/(1-eta_BAI)^C_2)
Tau_c_BY_Tau = (Beta*lambda_FAIe)/(1+Beta*lambda_FAIe)
#Compute Rs
E = E_B+E_IM+E_IN
Tau_wetplus = (Tau_wet*u_star^2)/kin.vis
Cd = 1/6
Rs = (R*w_dm*((E*Tau_c_BY_Tau/Cd)+(1+Tau_c_BY_Tau)*Sc^-1+10^(-3/Tau_wetplus))+Vg_wet)^-1
#Compute Vd
Vd = (Rg+((Rs-Rg)/exp(Ra/Rg)))^-1
quantile(Vd, c(.05, 0.10, .50, 0.95))
```

Codes for Sobol' sensitivity test for Zhang et al. (2001) parameterization

```
#Dry deposition parameterization by Zhang et al. (2001)
#Sobol sensitivity test: Grass (the code is similar for other LUCs)
#Change LUC dependent parameters for other LUCs
#Change sensitivity ranges for other LUCs
set.seed(5)
library(sensitivity)
library(boot)
C1 = 0.2789
C2 = 3.115
C3 = 5.415*10^-11
C4 = -1.399
dp_i = 10
dp_d = dp_i*10^-6
rd = dp_d/2
k_B = 1.38*10^-23
Temp = 273.15+25
P = 101325
d_air = 3.72*10^-10
lambda = (k_B*Temp)/(sqrt(2)*3.1416*P*d_air^2)
dyn.vis = ((5*10^-8)*Temp)+4*10^-6
z = 2
zR = 3.5
k_c = 0.41
e_0 = 3
R1 = 1
```

```
kin.vis = ((9*10^-8)*Temp)+10^-5
gamma = 0.54
alpha = 1.2
beta =  2
A = 2/1000
model <- function (X) ((((X[,2])*(2*{(C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3))^2*9.81*(1+(2*lambda/(2*{(C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3)))*(1.257+0.4*exp(-0.55*((2*{(C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3)))/lambda))))/(18*dyn.vis)))+
  (1/((((log(zR/(X[,4]))+5*z/(X[,3]))/(k_c*(X[,5])))+
        (1/{(e_0*(X[,5]))*
              ((((kin.vis/(((1+(2*lambda/(2*{(C1*rd^C2)/(C3*rd^C4-log10(X[,1]))+rd^3}^(1/3)))*(1.257+0.4*exp(-
0.55*(2*{(C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3))/lambda)))*k_B*Temp)/(3*3.1416*dyn.vis*(2*{(C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3))))))^(-gamma))+
                 ({{{{{(X[,2])*(2*{(C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3))^2*9.81*(1+(2*lambda/(2*{(C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3)))*(1.257+0.4*exp(-0.55*((2*{(C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3)))/lambda))))/(18*dyn.vis))*(X[,5]))/(9.81*A))/(alpha+(((((X[,2])*(2*{(C1*rd^C2)/(C3*rd^C
4-log10(X[,1]))+rd^3}^(1/3))^2*9.81*(1+(2*lambda/(2*{(C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3)))*(1.257+0.4*exp(-0.55*((2*{(C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3)))/lambda))))/(18*dyn.vis))*(X[,5]))/(9.81*A)))}^beta)+
                 (0.5*((2*{(C1*rd^C2)/(C3*rd^C4-log10(X[,1]))+rd^3}^(1/3))/A)^2))*R1})))


N <- 100000
x1 = runif(1*N,0.1,1.0)           #RH
x2 = runif(1*N,1500,2000)         #rho
x3 = runif(1*N,10,100)            #L
x4 = runif(1*N,0.02,0.10)         #z0
x5 = runif(1*N,0.1,0.5)           #u_star
x_1 = runif(1*N,0.1,1.0)          #RH
x_2 = runif(1*N,1500,2000)        #rho
x_3 = runif(1*N,10,100)           #L
x_4 = runif(1*N,0.02,0.10)        #z0
x_5 = runif(1*N,0.1,0.5)          #u_star
Y1 = matrix(c(x1,x2,x3,x4,x5), nrow=N)
X1 = data.frame(matrix(Y1,nrow=N))
Y2 = matrix(c(x_1,x_2,x_3,x_4,x_5), nrow=N)
X2 = data.frame(matrix(Y2, nrow=N))
a = sobol2007(model = model, X1 = X1, X2= X2, nboot = 2000, conf = 0.95);a
```

Codes for Sobol' sensitivity test for the Petroff and Zhang (2010) parameterization

```
#Dry deposition parameterization by Petroff and Zhang (2010)
#Sobol sensitivity test: Grass (the code is similar for other LUCs)
#Change LUC dependent parameters for other LUCs
#Change sensitivity ranges for other LUCs
set.seed(5)
library(sensitivity)
library(boot)
C1 = 0.2789
C2 = 3.115
C3 = 5.415*10^-11
C4 = -1.399
dp_i = 10
dp_d = dp_i*10^-6
rd = dp_d/2
k_B = 1.38*10^-23
Temp = 273.15+25
P = 101325
d_air = 3.7208*10^-10
lambda = (k_B*Temp)/(sqrt(2)*3.1416*P*d_air^2)
dyn.vis = ((5*10^-8)*Temp)+4*10^-6
kin.vis = ((9*10^-8)*Temp)+10^-5
Vphor = 0
z = 2
zR = 3.5
k_c = 0.41
cd = 1/6
kx = 0.216
C_IT = 0.056
L_obs = 0.01
```

```
C_B = 0.996
C_IN = 0.162
C_IM = 0.081
beta_IM = 0.47


model <- function (X) ((((((X[,2])*(2*({{(C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3)))^2*(1+(2*lambda/(2*({{(C1*rd^C2)/(C3*rd^C4-log10(X[,1]))+rd^3}^(1/3))))*(1.257+0.4*exp(-
0.55*(2*({{(C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3)))/lambda))))/(18*dyn.vis))*9.81)+Vphor)+1/(((log(zR/(X[,4]))+(5*(z/(X[,3])))))/(k_c*(X[,5
]))+1/((X[,5])*(((((kin.vis/(((1+(2*lambda/(2*({{(C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3))))*(1.257+0.4*exp(-0.55*(2*({{(C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3)))/lambda)))*k_B*Temp)/(3*3.1416*dyn.vis*(2*({{(C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3)))))))^(-2/3)/14.5)*{1/6*log(1+((((kin.vis/(((1+(2*lambda/(2*({{(C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3))))*(1.257+0.4*exp(-0.55*(2*({{(C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3)))/lambda)))*k_B*Temp)/(3*3.1416*dyn.vis*(2*({{(C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3)))))))^(1/3))/2.9))^2/(1-((((kin.vis/(((1+(2*lambda/(2*({{(C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3))))*(1.257+0.4*exp(-0.55*(2*({{(C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3)))/lambda)))*k_B*Temp)/(3*3.1416*dyn.vis*(2*({{(C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3)))))))^(1/3))/2.9)+((((kin.vis/(((1+(2*lambda/(2*({{(C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3))))*(1.257+0.4*exp(-0.55*(2*({{(C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3)))/lambda)))*k_B*Temp)/(3*3.1416*dyn.vis*(2*({{(C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3)))))))^(1/3))/2.9)^2)+1/sqrt(3)*atan((2*((((kin.vis/(((1+(2*lambda/(2*({{(C1*rd^C2)/(C3*rd
^C4-log10(X[,1]))+rd^3}^(1/3))))*(1.257+0.4*exp(-0.55*(2*({{(C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3)))/lambda)))*k_B*Temp)/(3*3.1416*dyn.vis*(2*({{(C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3)))))))^(1/3))/2.9)-1)/sqrt(3))+3.1416/6*sqrt(3)}^-1) + (2.5*10^-
3*0.14*((((((X[,2])*(2*({{(C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3)))^2*(1+(2*lambda/(2*({{(C1*rd^C2)/(C3*rd^C4-log10(X[,1]))+rd^3}^(1/3))))*(1.257+0.4*exp(-
0.55*(2*({{(C1*rd^C2)/(C3*rd^C4-log10(X[,1]))+rd^3}^(1/3)))/lambda))))/(18*dyn.vis))*(X[,5])^2)/kin.vis))^2)
)*{(1+((X[,6])*((((X[,9])/(exp(({{kx*(X[,6]))/(12*k_c^2*(1-
(X[,8])/(X[,7]))^2)}^(1/3)*(1+5*(z/(X[,3])))^(2/3)*{((X[,7])-(X[,8]))/((X[,3]))})*((z)/(X[,7])-
1))))/(X[,5]))*((C_B*((kin.vis/(((1+(2*lambda/(2*({{(C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3))))*(1.257+0.4*exp(-0.55*(2*({{(C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3)))/lambda)))*k_B*Temp)/(3*3.1416*dyn.vis*(2*({{(C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3)))))))^(-2/3))*(((((X[,9])/(exp(({{kx*(X[,6]))/(12*k_c^2*(1-
(X[,8])/(X[,7]))^2)}^(1/3)*(1+5*(z/(X[,3])))^(2/3)*{((X[,7])-(X[,8]))/((X[,3]))})*((z)/(X[,7])-
1))))*L_obs)/(kin.vis))^(-1/2)))+(C_IN*((2*({{(C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3)))/L_obs)*(2+log(4*L_obs/(2*({{(C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3)))))+(C_IM*((((((X[,2])*(2*({{(C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3)))^2*(1+(2*lambda/(2*({{(C1*rd^C2)/(C3*rd^C4-log10(X[,1]))+rd^3}^(1/3))))*(1.257+0.4*exp(-
0.55*(2*({{(C1*rd^C2)/(C3*rd^C4-log10(X[,1]))+rd^3}^(1/3)))/lambda))))/(18*dyn.vis)
)*((X[,9])/(exp(({{kx*(X[,6]))/(12*k_c^2*(1-(X[,8])/(X[,7]))^2)}^(1/3)*(1+5*(z/(X[,3])))^(2/3)*{((X[,7])-
(X[,8]))/((X[,3]))})*((z)/(X[,7])-1)))))/L_obs)/((((((X[,2])*(2*({{(C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3)))^2*(1+(2*lambda/(2*({{(C1*rd^C2)/(C3*rd^C4-log10(X[,1]))+rd^3}^(1/3))))*(1.257+0.4*exp(-
0.55*(2*({{(C1*rd^C2)/(C3*rd^C4-log10(X[,1]))+rd^3}^(1/3)))/lambda))))/(18*dyn.vis)
)*((X[,9])/(exp(({{kx*(X[,6]))/(12*k_c^2*(1-(X[,8])/(X[,7]))^2)}^(1/3)*(1+5*(z/(X[,3])))^(2/3)*{((X[,7])-
(X[,8]))/((X[,3]))})*((z)/(X[,7])-1)))))/L_obs)+(beta_IM)))^2))+(2.5*10^-
3*C_IT*(((((X[,2])*(2*({{C1*rd^C2)/(C3*rd^C4-log10(X[,1]))+rd^3}^(1/3)))^2*(1+(2*lambda/(2*({{(C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3))))*(1.257+0.4*exp(-0.55*(2*({{(C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3)))/lambda))))/(18*dyn.vis))*(X[,5])^2)/kin.vis)^2)
)*(X[,7])/((((0.41*((X[,7])-(X[,8])))/((1+5*(z/(X[,3])))*((X[,7])-
(X[,8]))/((X[,3])))))))/((((((kin.vis/(((1+(2*lambda/(2*({{(C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3))))*(1.257+0.4*exp(-0.55*(2*({{(C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3)))/lambda)))*k_B*Temp)/(3*3.1416*dyn.vis*(2*({{(C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3)))))))^(-2/3)/14.5)*{1/6*log(1+((((kin.vis/(((1+(2*lambda/(2*({{(C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3))))*(1.257+0.4*exp(-0.55*(2*({{(C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3)))/lambda)))*k_B*Temp)/(3*3.1416*dyn.vis*(2*({{(C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3)))))))^(1/3))/2.9))^2/(1-((((kin.vis/(((1+(2*lambda/(2*({{(C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3))))*(1.257+0.4*exp(-0.55*(2*({{(C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3)))/lambda)))*k_B*Temp)/(3*3.1416*dyn.vis*(2*({{(C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3)))))))^(1/3))/2.9)+((((kin.vis/(((1+(2*lambda/(2*({{(C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3))))*(1.257+0.4*exp(-0.55*(2*({{(C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3)))/lambda)))*k_B*Temp)/(3*3.1416*dyn.vis*(2*({{(C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3)))))))^(1/3))/2.9)^2)+1/sqrt(3)*atan((2*((((kin.vis/(((1+(2*lambda/(2*({{(C1*rd^C2)/(C3*rd
^C4-log10(X[,1]))+rd^3}^(1/3))))*(1.257+0.4*exp(-0.55*(2*({{(C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3)))/lambda)))*k_B*Temp)/(3*3.1416*dyn.vis*(2*({{(C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3)))))))^(1/3))/2.9)-1)/sqrt(3))+3.1416/6*sqrt(3)}^-1) + (2.5*10^-
3*0.14*((((((X[,2])*(2*({{(C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3)))^2*(1+(2*lambda/(2*({{(C1*rd^C2)/(C3*rd^C4-log10(X[,1]))+rd^3}^(1/3))))*(1.257+0.4*exp(-
0.55*(2*({{(C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3)))/lambda))))/(18*dyn.vis))*(X[,5])^2)/kin.vis))^2))*(X[,7])/((0.41*((X[,7])-
(X[,8])))/((1+5*(z/(X[,3])))*((X[,7])-(X[,8]))/((X[,3]))))
)-({(kx*(X[,6]))/(12*k_c^2*(1-(X[,8])/(X[,7]))^2)}^(1/3)*(1+5*(z/(X[,3])))^(2/3)*{((X[,7])-
(X[,8]))/((X[,3]))})/2)*tanh(((((kx*(X[,6]))/(12*k_c^2*(1-
```

```
(X[,8])/(X[,7]))^2)}^(1/3)*(1+5*(z/(X[,3])))^(2/3)*{((X[,7])-
(X[,8]))/((X[,3]))})^2/4+((X[,6])*((((X[,9])/(exp(({{kx*(X[,6]))/(12*k_c^2*(1-
(X[,8])/(X[,7]))^2)}^(1/3)*(1+5*(z/(X[,3])))^(2/3)*{((X[,7])-(X[,8]))/((X[,3]))})*((z)/(X[,7])-
1))))/(X[,5]))*((C_B*((kin.vis/(((1+(2*lambda/(2*{((C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3))))*(1.257+0.4*exp(-0.55*(2*({((C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3)))/lambda)))*k_B*Temp)/(3*3.1416*dyn.vis*(2*({((C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3)))))))^(-2/3))*(((((X[,9])/(exp(({{kx*(X[,6]))/(12*k_c^2*(1-
(X[,8])/(X[,7]))^2)}^(1/3)*(1+5*(z/(X[,3])))^(2/3)*{((X[,7])-(X[,8]))/((X[,3]))})*((z)/(X[,7])-
1))))*L_obs)/(kin.vis))^(-1/2)))+(C_IN*((2*({((C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3)))/L_obs)*(2+log(4*L_obs/(2*({((C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3)))))+(C_IM*((((((X[,2])*(2*({((C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3)))^2*(1+(2*lambda/(2*({((C1*rd^C2)/(C3*rd^C4-log10(X[,1]))+rd^3}^(1/3))))*(1.257+0.4*exp(-
0.55*(2*({((C1*rd^C2)/(C3*rd^C4-log10(X[,1]))+rd^3}^(1/3)))/lambda))))/(18*dyn.vis)
)*((X[,9])/(exp(({{kx*(X[,6]))/(12*k_c^2*(1-(X[,8])/(X[,7]))^2)}^(1/3)*(1+5*(z/(X[,3])))^(2/3)*{((X[,7])-
(X[,8]))/((X[,3]))})*((z)/(X[,7])-1)))))/L_obs)/((((((X[,2])*(2*({((C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3)))^2*(1+(2*lambda/(2*({((C1*rd^C2)/(C3*rd^C4-log10(X[,1]))+rd^3}^(1/3))))*(1.257+0.4*exp(-
0.55*(2*({((C1*rd^C2)/(C3*rd^C4-log10(X[,1]))+rd^3}^(1/3)))/lambda))))/(18*dyn.vis)
)*((X[,9])/(exp(({{kx*(X[,6]))/(12*k_c^2*(1-(X[,8])/(X[,7]))^2)}^(1/3)*(1+5*(z/(X[,3])))^(2/3)*{((X[,7])-
(X[,8]))/((X[,3]))})*((z)/(X[,7])-1)))))/L_obs)+(beta_IM)))^2))+(2.5*10^-
3*C_IT*(((((X[,2])*(2*({((C1*rd^C2)/(C3*rd^C4-log10(X[,1]))+rd^3}^(1/3)))^2*(1+(2*lambda/(2*({((C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3))))*(1.257+0.4*exp(-0.55*(2*({((C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3)))/lambda))))/(18*dyn.vis))*(X[,5])^2)/kin.vis)^2)
)*(X[,7])/((((0.41*((X[,7])-(X[,8])))/((1+5*(z/(X[,3])))*((X[,7])-
(X[,8]))/((X[,3])))))))^0.5))/(((((kx*(X[,6]))/(12*k_c^2*(1-
(X[,8])/(X[,7]))^2)}^(1/3)*(1+5*(z/(X[,3])))^(2/3)*{((X[,7])-
(X[,8]))/((X[,3]))})^2/4+((X[,6])*((((X[,9])/(exp(({{kx*(X[,6]))/(12*k_c^2*(1-
(X[,8])/(X[,7]))^2)}^(1/3)*(1+5*(z/(X[,3])))^(2/3)*{((X[,7])-(X[,8]))/((X[,3]))})*((z)/(X[,7])-
1))))/(X[,5]))*((C_B*((kin.vis/(((1+(2*lambda/(2*({((C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3))))*(1.257+0.4*exp(-0.55*(2*({((C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3)))/lambda)))*k_B*Temp)/(3*3.1416*dyn.vis*(2*({((C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3)))))))^(-2/3))*(((((X[,9])/(exp(({{kx*(X[,6]))/(12*k_c^2*(1-
(X[,8])/(X[,7]))^2)}^(1/3)*(1+5*(z/(X[,3])))^(2/3)*{((X[,7])-(X[,8]))/((X[,3]))})*((z)/(X[,7])-
1))))*L_obs)/(kin.vis))^(-1/2)))+(C_IN*((2*({((C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3)))/L_obs)*(2+log(4*L_obs/(2*({((C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3)))))+(C_IM*((((((X[,2])*(2*({((C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3)))^2*(1+(2*lambda/(2*({((C1*rd^C2)/(C3*rd^C4-log10(X[,1]))+rd^3}^(1/3))))*(1.257+0.4*exp(-
0.55*(2*({((C1*rd^C2)/(C3*rd^C4-log10(X[,1]))+rd^3}^(1/3)))/lambda))))/(18*dyn.vis)
)*((X[,9])/(exp(({{kx*(X[,6]))/(12*k_c^2*(1-(X[,8])/(X[,7]))^2)}^(1/3)*(1+5*(z/(X[,3])))^(2/3)*{((X[,7])-
(X[,8]))/((X[,3]))})*((z)/(X[,7])-1)))))/L_obs)/((((((X[,2])*(2*({((C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3)))^2*(1+(2*lambda/(2*({((C1*rd^C2)/(C3*rd^C4-log10(X[,1]))+rd^3}^(1/3))))*(1.257+0.4*exp(-
0.55*(2*({((C1*rd^C2)/(C3*rd^C4-log10(X[,1]))+rd^3}^(1/3)))/lambda))))/(18*dyn.vis)
)*((X[,9])/(exp(({{kx*(X[,6]))/(12*k_c^2*(1-(X[,8])/(X[,7]))^2)}^(1/3)*(1+5*(z/(X[,3])))^(2/3)*{((X[,7])-
(X[,8]))/((X[,3]))})*((z)/(X[,7])-1)))))/L_obs)+(beta_IM)))^2))+(2.5*10^-
3*C_IT*(((((X[,2])*(2*({((C1*rd^C2)/(C3*rd^C4-log10(X[,1]))+rd^3}^(1/3)))^2*(1+(2*lambda/(2*({((C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3))))*(1.257+0.4*exp(-0.55*(2*({((C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3)))/lambda))))/(18*dyn.vis))*(X[,5])^2)/kin.vis)^2)
)*(X[,7])/((((0.41*((X[,7])-(X[,8])))/((1+5*(z/(X[,3])))*((X[,7])-
(X[,8]))/((X[,3])))))))^0.5)}/{(1+((X[,6])*((((X[,9])/(exp(({{kx*(X[,6]))/(12*k_c^2*(1-
(X[,8])/(X[,7]))^2)}^(1/3)*(1+5*(z/(X[,3])))^(2/3)*{((X[,7])-(X[,8]))/((X[,3]))})*((z)/(X[,7])-
1))))/(X[,5]))*((C_B*((kin.vis/(((1+(2*lambda/(2*({((C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3))))*(1.257+0.4*exp(-0.55*(2*({((C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3)))/lambda)))*k_B*Temp)/(3*3.1416*dyn.vis*(2*({((C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3)))))))^(-2/3))*(((((X[,9])/(exp(({{kx*(X[,6]))/(12*k_c^2*(1-
(X[,8])/(X[,7]))^2)}^(1/3)*(1+5*(z/(X[,3])))^(2/3)*{((X[,7])-(X[,8]))/((X[,3]))})*((z)/(X[,7])-
1))))*L_obs)/(kin.vis))^(-1/2)))+(C_IN*((2*({((C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3)))/L_obs)*(2+log(4*L_obs/(2*({((C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3)))))+(C_IM*((((((X[,2])*(2*({((C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3)))^2*(1+(2*lambda/(2*({((C1*rd^C2)/(C3*rd^C4-log10(X[,1]))+rd^3}^(1/3))))*(1.257+0.4*exp(-
0.55*(2*({((C1*rd^C2)/(C3*rd^C4-log10(X[,1]))+rd^3}^(1/3)))/lambda))))/(18*dyn.vis)
)*((X[,9])/(exp(({{kx*(X[,6]))/(12*k_c^2*(1-(X[,8])/(X[,7]))^2)}^(1/3)*(1+5*(z/(X[,3])))^(2/3)*{((X[,7])-
(X[,8]))/((X[,3]))})*((z)/(X[,7])-1)))))/L_obs)/((((((X[,2])*(2*({((C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3)))^2*(1+(2*lambda/(2*({((C1*rd^C2)/(C3*rd^C4-log10(X[,1]))+rd^3}^(1/3))))*(1.257+0.4*exp(-
0.55*(2*({((C1*rd^C2)/(C3*rd^C4-log10(X[,1]))+rd^3}^(1/3)))/lambda))))/(18*dyn.vis)
)*((X[,9])/(exp(({{kx*(X[,6]))/(12*k_c^2*(1-(X[,8])/(X[,7]))^2)}^(1/3)*(1+5*(z/(X[,3])))^(2/3)*{((X[,7])-
(X[,8]))/((X[,3]))})*((z)/(X[,7])-1)))))/L_obs)+(beta_IM)))^2))+(2.5*10^-
3*C_IT*(((((X[,2])*(2*({((C1*rd^C2)/(C3*rd^C4-log10(X[,1]))+rd^3}^(1/3)))^2*(1+(2*lambda/(2*({((C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3))))*(1.257+0.4*exp(-0.55*(2*({((C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3)))/lambda))))/(18*dyn.vis))*(X[,5])^2)/kin.vis)^2)
)*(X[,7])/((((0.41*((X[,7])-(X[,8])))/((1+5*(z/(X[,3])))*((X[,7])-
(X[,8]))/((X[,3])))))))+(({(kx*(X[,6]))/(12*k_c^2*(1-(X[,8])/(X[,7]))^2)}^(1/3)*(1+5*(z/(X[,3])))^(2/3)*{((X[,7])-
(X[,8]))/((X[,3]))})/2)*tanh(((((kx*(X[,6]))/(12*k_c^2*(1-
(X[,8])/(X[,7]))^2)}^(1/3)*(1+5*(z/(X[,3])))^(2/3)*{((X[,7])-
(X[,8]))/((X[,3]))})^2/4+((X[,6])*((((X[,9])/(exp(({{kx*(X[,6]))/(12*k_c^2*(1-
(X[,8])/(X[,7]))^2)}^(1/3)*(1+5*(z/(X[,3])))^(2/3)*{((X[,7])-(X[,8]))/((X[,3]))})*((z)/(X[,7])-
```

```
1))))/(X[,5]))*((C_B*(((kin.vis/(((1+(2*lambda/(2*({(C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3))))*(1.257+0.4*exp(-0.55*(2*({(C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3)))/lambda)))*k_B*Temp)/(3*3.1416*dyn.vis*(2*({(C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3)))))))^(-2/3))*(((((X[,9])/(exp(({(kx*(X[,6]))/(12*k_c^2*(1-
(X[,8])/(X[,7]))^2)}^(1/3)*(1+5*(z/(X[,3])))^(2/3)*{((X[,7])-(X[,8]))/((X[,3]))})*((z)/(X[,7])-
1))))*L_obs)/(kin.vis))^(-1/2)))+(C_IN*((2*({(C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3)))/L_obs)*(2+log(4*L_obs/(2*({(C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3))))))+(C_IM*((((((X[,2])*(2*({(C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3)))^2*(1+(2*lambda/(2*({(C1*rd^C2)/(C3*rd^C4-log10(X[,1]))+rd^3}^(1/3))))*(1.257+0.4*exp(-
0.55*(2*({(C1*rd^C2)/(C3*rd^C4-log10(X[,1]))+rd^3}^(1/3)))/lambda))))/(18*dyn.vis)
)*((X[,9])/(exp(({(kx*(X[,6]))/(12*k_c^2*(1-(X[,8])/(X[,7]))^2)}^(1/3)*(1+5*(z/(X[,3])))^(2/3)*{((X[,7])-
(X[,8]))/((X[,3]))})*((z)/(X[,7])-1)))))/L_obs)/((((((X[,2])*(2*({(C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3)))^2*(1+(2*lambda/(2*({(C1*rd^C2)/(C3*rd^C4-log10(X[,1]))+rd^3}^(1/3))))*(1.257+0.4*exp(-
0.55*(2*({(C1*rd^C2)/(C3*rd^C4-log10(X[,1]))+rd^3}^(1/3)))/lambda))))/(18*dyn.vis)
)*((X[,9])/(exp(({(kx*(X[,6]))/(12*k_c^2*(1-(X[,8])/(X[,7]))^2)}^(1/3)*(1+5*(z/(X[,3])))^(2/3)*{((X[,7])-
(X[,8]))/((X[,3]))})*((z)/(X[,7])-1)))))/L_obs)+(beta_IM)))^2))+(2.5*10^-
3*C_IT*(((((X[,2])*(2*({(C1*rd^C2)/(C3*rd^C4-log10(X[,1]))+rd^3}^(1/3)))^2*(1+(2*lambda/(2*({(C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3))))*(1.257+0.4*exp(-0.55*(2*({(C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3)))/lambda))))/(18*dyn.vis))*(X[,5])^2)/kin.vis)^2)
)*(X[,7])/((((0.41*((X[,7])-(X[,8])))/((1+5*(z/(X[,3])))*((X[,7])-
(X[,8]))/((X[,3])))))))^0.5))/(((((kx*(X[,6]))/(12*k_c^2*(1-
(X[,8])/(X[,7]))^2)}^(1/3)*(1+5*(z/(X[,3])))^(2/3)*{((X[,7])-
(X[,8]))/((X[,3]))})^2/4+((X[,6])*((((X[,9])/(exp(({(kx*(X[,6]))/(12*k_c^2*(1-
(X[,8])/(X[,7]))^2)}^(1/3)*(1+5*(z/(X[,3])))^(2/3)*{((X[,7])-(X[,8]))/((X[,3]))})*((z)/(X[,7])-
1))))/(X[,5]))*((C_B*(((kin.vis/(((1+(2*lambda/(2*({(C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3))))*(1.257+0.4*exp(-0.55*(2*({(C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3)))/lambda)))*k_B*Temp)/(3*3.1416*dyn.vis*(2*({(C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3)))))))^(-2/3))*(((((X[,9])/(exp(({(kx*(X[,6]))/(12*k_c^2*(1-
(X[,8])/(X[,7]))^2)}^(1/3)*(1+5*(z/(X[,3])))^(2/3)*{((X[,7])-(X[,8]))/((X[,3]))})*((z)/(X[,7])-
1))))*L_obs)/(kin.vis))^(-1/2)))+(C_IN*((2*({(C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3)))/L_obs)*(2+log(4*L_obs/(2*({(C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3))))))+(C_IM*((((((X[,2])*(2*({(C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3)))^2*(1+(2*lambda/(2*({(C1*rd^C2)/(C3*rd^C4-log10(X[,1]))+rd^3}^(1/3))))*(1.257+0.4*exp(-
0.55*(2*({(C1*rd^C2)/(C3*rd^C4-log10(X[,1]))+rd^3}^(1/3)))/lambda))))/(18*dyn.vis)
)*((X[,9])/(exp(({(kx*(X[,6]))/(12*k_c^2*(1-(X[,8])/(X[,7]))^2)}^(1/3)*(1+5*(z/(X[,3])))^(2/3)*{((X[,7])-
(X[,8]))/((X[,3]))})*((z)/(X[,7])-1)))))/L_obs)/((((((X[,2])*(2*({(C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3)))^2*(1+(2*lambda/(2*({(C1*rd^C2)/(C3*rd^C4-log10(X[,1]))+rd^3}^(1/3))))*(1.257+0.4*exp(-
0.55*(2*({(C1*rd^C2)/(C3*rd^C4-log10(X[,1]))+rd^3}^(1/3)))/lambda))))/(18*dyn.vis)
)*((X[,9])/(exp(({(kx*(X[,6]))/(12*k_c^2*(1-(X[,8])/(X[,7]))^2)}^(1/3)*(1+5*(z/(X[,3])))^(2/3)*{((X[,7])-
(X[,8]))/((X[,3]))})*((z)/(X[,7])-1)))))/L_obs)+(beta_IM)))^2))+(2.5*10^-
3*C_IT*(((((X[,2])*(2*({(C1*rd^C2)/(C3*rd^C4-log10(X[,1]))+rd^3}^(1/3)))^2*(1+(2*lambda/(2*({(C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3))))*(1.257+0.4*exp(-0.55*(2*({(C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3)))/lambda))))/(18*dyn.vis))*(X[,5])^2)/kin.vis)^2)
)*(X[,7])/((((0.41*((X[,7])-(X[,8])))/((1+5*(z/(X[,3])))*((X[,7])-(X[,8]))/((X[,3])))))))^0.5))))))

N <- 100000
x1 = runif(1*N,0.1,1.0)           #RH
x2 = runif(1*N,1500,2000)         #rho
x3 = runif(1*N,10,100)            #L
x4 = runif(1*N,0.02,0.10)         #z0
x5 = runif(1*N,0.1,0.5)           #u_star
x6 = runif(1*N,1,4)               #LAI
x7 = runif(1*N,0.15,0.77)         #h
x8 = runif(1*N,0.10,0.49)         #d
x9 = runif(1*N,1,5)               #U

x_1 = runif(1*N,0.1,1.0)           #RH
x_2 = runif(1*N,1500,2000)         #rho
x_3 = runif(1*N,10,100)            #L
x_4 = runif(1*N,0.02,0.10)         #z0
x_5 = runif(1*N,0.1,0.5)           #u_star
x_6 = runif(1*N,1,4)               #LAI
x_7 = runif(1*N,0.15,0.77)         #h
x_8 = runif(1*N,0.10,0.49)         #d
x_9 = runif(1*N,1,5)               #U

Y1 = matrix(c(x1,x2,x3,x4,x5,x6,x7,x8,x9), nrow=N)
X1 = data.frame(matrix(Y1, nrow=N))
Y2 = matrix(c(x_1,x_2,x_3,x_4,x_5,x_6,x_7,x_8,x_9), nrow=N)
X2 = data.frame(matrix(Y2, nrow=N))
a = sobol2007(model = model, X1 = X1, X2=X2, nboot = 2000, conf= 0.95);a
```

Codes for Sobol' sensitivity test for Kouznetsov and Sofiev (2012) parameterization

```
#Dry deposition parameterization by Kouznetsov and Sofiev (2012)
#Sobol sensitivity test: Grass (the code is similar for other LUCs)
#Change LUC dependent parameters for other LUCs
#Change sensitivity ranges for other LUCs
set.seed(5)
library(sensitivity)
library(boot)
C1 = 0.2789
C2 = 3.115
C3 = 5.415*10^-11
C4 = -1.399
dp_a = 10
dp_i = dp_a*10^-6
rd = dp_i/2
k_B = 1.38*10^-23
Temp = 273.15+25
P = 101325
d_air = 3.7208*10^-10
lambda = (k_B*Temp)/(sqrt(2)*3.1416*P*d_air^2)
dyn.vis = ((5*10^-8)*Temp)+4*10^-6
kin.vis = ((9*10^-8)*Temp)+10^-5
a  =  2*10^-3
kin.vis = ((9*10^-8)*Temp)+10^-5
C_S = 0.003
C_R = 0.3
u.Uh = 0.3
eta_impSt.e2 = 0


model<-function(X) ((2*((((X[,3])*a)/kin.vis)^(-0.5))*(((kin.vis/(((1+(2*lambda/(2*({(C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3))))*(1.257+0.4*exp(-0.55*(2*({(C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3)))/lambda)))*k_B*Temp)/(3*3.1416*dyn.vis*(2*({(C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3))))))))^(-2/3))+(80*(X[,3])*(((2*({(C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3)))/(a))^2)*((((X[,3])*a)/kin.vis)^(0.5)))+(((2*((C_S+C_R/(X[,4]))^0.5))/(X[,3]))*(ifelse(
((((((X[,2])*(2*({(C1*rd^C2)/(C3*rd^C4-log10(X[,1]))+rd^3}^(1/3))))^2*(1+(2*lambda/(2*({(C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3))))*(1.257+0.4*exp(-0.55*(2*({(C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3)))/lambda))))/(18*dyn.vis))*(X[,3]))/a) - (((((C_S+C_R/(X[,4]))^0.5))^-
1)^2*(((X[,3])*a)/kin.vis))^(-0.5))>0.15,(exp((-0.1/(((((((X[,2])*(2*({(C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3))))^2*(1+(2*lambda/(2*({(C1*rd^C2)/(C3*rd^C4-log10(X[,1]))+rd^3}^(1/3))))*(1.257+0.4*exp(-
0.55*(2*({(C1*rd^C2)/(C3*rd^C4-log10(X[,1]))+rd^3}^(1/3)))/lambda))))/(18*dyn.vis))*(X[,3]))/a) -
(((((C_S+C_R/(X[,4]))^0.5))^-1)^2*(((X[,3])*a)/kin.vis))^(-0.5)) - 0.15 ) )-
(1/sqrt(((((((X[,2])*(2*({(C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3))))^2*(1+(2*lambda/(2*({(C1*rd^C2)/(C3*rd^C4-log10(X[,1]))+rd^3}^(1/3))))*(1.257+0.4*exp(-
0.55*(2*({(C1*rd^C2)/(C3*rd^C4-log10(X[,1]))+rd^3}^(1/3)))/lambda))))/(18*dyn.vis))*(X[,3]))/a) -
(((((C_S+C_R/(X[,4]))^0.5))^-1)^2*(((X[,3])*a)/kin.vis))^(-0.5))-
0.15)))),(eta_impSt.e2)))*((((((X[,2])*(2*({(C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3))))^2*(1+(2*lambda/(2*({(C1*rd^C2)/(C3*rd^C4-log10(X[,1]))+rd^3}^(1/3))))*(1.257+0.4*exp(-
0.55*(2*({(C1*rd^C2)/(C3*rd^C4-log10(X[,1]))+rd^3}^(1/3)))/lambda))))/(18*dyn.vis))*(X[,3]))/a)-
((C_S+C_R/(X[,4]))^0.5)*(((X[,3])*a)/kin.vis)^-0.5))+((((X[,2])*(2*({(C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3))))^2*(1+(2*lambda/(2*({(C1*rd^C2)/(C3*rd^C4-log10(X[,1]))+rd^3}^(1/3))))*(1.257+0.4*exp(-
0.55*(2*({(C1*rd^C2)/(C3*rd^C4-log10(X[,1]))+rd^3}^(1/3)))/lambda))))/(18*dyn.vis))*9.81))



N <- 100000
x1 = runif(1*N,0.1,1.0)            #RH
x2 = runif(1*N,1500,2000)          #rho
x3 = runif(1*N,0.1,0.5)            #u_star
x4 = runif(1*N,1,4)                #LAI
x_1 = runif(1*N,0.1,1.0)           #RH
x_2 = runif(1*N,1500,2000)         #rho
x_3 = runif(1*N,0.1,0.5)           #u_star
x_4 = runif(1*N,1,4)               #LAI
Y1 = matrix(c(x1,x2,x3,x4), nrow=N)
X1 = data.frame(matrix(Y1,nrow=N))
Y2 = matrix(c(x_1,x_2,x_3,x_4), nrow=N)
X2 = data.frame(matrix(Y2, nrow=N))
a = sobol2007(model = model, X1 = X1, X2=X2, nboot = 2000, conf = 0.95);a
```

Codes for Sobol' sensitivity test for Zhang and He (2014) parameterization

```
#Dry deposition parameterization by Zhang and He (2014)
#Sobol sensitivity test: Grass (the code is similar for other LUCs)
#Change LUC dependent parameters for other LUCs
#Change sensitivity ranges for other LUCs
set.seed(5)
library(sensitivity)
library(boot)
C1 = 0.4809
C2 = 3.082
C3 = 3.110*10^-11
C4 = -1.428
dp_a = 1
dp_i = dp_a*10^-6
rd = dp_i/2
k_B = 1.38*10^-23
Temp = 273.15+25
P = 101325
d_air = 3.7208*10^-10
lambda = (k_B*Temp)/(sqrt(2)*3.1416*P*d_air^2)
dyn.vis = ((5*10^-8)*Temp)+4*10^-6
z = 2
zR = 3.5
k_c = 0.41
# For PM2.5-10
b1= -7.9*10^-2
b2= 1.0*10^0
b3 = 6.6*10^-1
c1= 5.1*10^0
c2 = -4.2*10^0
c3 = 9.9*10^-1
LAImax = 4

model<- function(X) (1/(1/((((X[,2])*((2*({(C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3))))^2*(1+(2*lambda/(2*({(C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3))))*(1.257+0.4*exp(-0.55*(2*({(C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3)))/lambda))))/(18*dyn.vis))*9.81))+(1/(((log(zR/(X[,5]))-(ifelse((z/(X[,4])) <= 0,
(2*log(0.5*{1+(1-16*(z/(X[,4])))^0.5})),(-
5*(z/(X[,4])))))/(k_c*(X[,3])))+(1/((b1*(X[,3])+b2*(X[,3])^2+b3*(X[,3])^3)*(exp((c1*(X[,3])+c2*(X[,3])^2+c3*(X[,3
])^3)*(X[,6])/LAImax)-1))))))

N <- 100000
x1 = runif(1*N,0.1,1.0)           #RH
x2 = runif(1*N,1500,2000)         #rho
x3 = runif(1*N,0.1,0.5)           #u_star
x4 = runif(1*N,10,100)            #L
x5 = runif(1*N,0.02,0.10)         #z0
x6 = runif(1*N,1,4)               #LAI
x_1 = runif(1*N,0.1,1.0)           #RH
x_2 = runif(1*N,1500,2000)         #rho
x_3 = runif(1*N,0.1,0.5)           #u_star
x_4 = runif(1*N,10,100)            #L
x_5 = runif(1*N,0.02,0.10)         #z0
x_6 = runif(1*N,1,4)               #LAI
Y1 = matrix(c(x1,x2,x3,x4,x5,x6), nrow=N)
X1 = data.frame(matrix(Y1,nrow=N))
Y2 = matrix(c(x_1,x_2,x_3,x_4,x_5,x_6), nrow=N)
X2 = data.frame(matrix(Y2, nrow=N))
a = sobol2007(model = model, X1 = X1, X2=X2, nboot = 2000, conf = 0.95);a
```

Codes for Sobol' sensitivity test for Zhang and Shao (2014) parameterization

```
#Dry deposition parameterization by Zhang and Shao (2014)
#Sobol sensitivity test: Grass (the code is similar for other LUCs)
#Change LUC dependent parameters for other LUCs
#Change sensitivity ranges for other LUCs
set.seed(5)
library(sensitivity)
library(boot)
```

```
C1 = 0.2789
C2 = 3.115
C3 = 5.415*10^-11
C4 = -1.399
dp_a = 10
dp_i = dp_a*10^-6
rd = dp_i/2
k_B = 1.38*10^-23
Temp = 273.15+25
P = 101325
d_air = 3.7208*10^-10
lambda = (k_B*Temp)/(sqrt(2)*3.1416*P*d_air^2)
dyn.vis = ((5*10^-8)*Temp)+4*10^-6
kin.vis = ((9*10^-8)*Temp)+10^-5
Temp = 273.15+25
k =  0.41
z =  1
zd = 0.20
z0 = 0.3/1000
h_c = 0.3*z0
B1 = 0.45
d_c = 0.005
nB = 0.5
C_B = 0.467
beta_IM = 0.6
Ain = 150
b = 2
B2 = 3
Beta = 200
C_1 = 6
C_2 = 0.1
lambda_FAI = 0.4
n_FAI = (lambda_FAI)/(h_c*d_c)
q = (3.1416*d_c^2)/4
eta_BAI = n_FAI*q
lambda_FAIe = ((lambda_FAI)/(1-eta_BAI)^C_2)*exp((-C_1*lambda_FAI)/(1-eta_BAI)^C_2)
Tau_c_BY_Tau = (Beta*lambda_FAIe)/(1+Beta*lambda_FAIe)
Cd = 1/6
model<- function(X) ((((1/(((((X[,2])*(dp_i)^2*(1+(2*lambda/(2*({(C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3))))*(1.257+0.4*exp(-0.55*(2*({(C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3)))/lambda))))/(18*dyn.vis))*9.81)))+(((((exp(-
b*sqrt(((((X[,2])*(dp_i)^2*(1+(2*lambda/(2*({(C1*rd^C2)/(C3*rd^C4-log10(X[,1]))+rd^3}^(1/3))))*(1.257+0.4*exp(-
0.55*(2*({(C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3)))/lambda))))/(18*dyn.vis))*(X[,3]))/d_c)))*(((X[,3])/(X[,4])*h_c))*(((((C_B*((kin.vis
/(((1+(2*lambda/(2*({(C1*rd^C2)/(C3*rd^C4-log10(X[,1]))+rd^3}^(1/3))))*(1.257+0.4*exp(-
0.55*(2*({(C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3)))/lambda)))*k_B*Temp)/(3*3.1416*dyn.vis*(2*({(C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3)))))))^(-2/3)*(((X[,4])*d_c)/(kin.vis))^(nB-
1))+(((((((X[,2])*(dp_i)^2*(1+(2*lambda/(2*({(C1*rd^C2)/(C3*rd^C4-log10(X[,1]))+rd^3}^(1/3))))*(1.257+0.4*exp(-
0.55*(2*({(C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3)))/lambda))))/(18*dyn.vis))*(X[,3]))/d_c)/((((((X[,2])*(dp_i)^2*(1+(2*lambda/(2*({(C1*
rd^C2)/(C3*rd^C4-log10(X[,1]))+rd^3}^(1/3))))*(1.257+0.4*exp(-0.55*(2*({(C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3)))/lambda))))/(18*dyn.vis))*(X[,3]))/d_c)+beta_IM))^2)+(Ain*(X[,3])*(10^((-
((((X[,2])*(dp_i)^2*(1+(2*lambda/(2*({(C1*rd^C2)/(C3*rd^C4-log10(X[,1]))+rd^3}^(1/3))))*(1.257+0.4*exp(-
0.55*(2*({(C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3)))/lambda))))/(18*dyn.vis))*(X[,3]))/d_c)))*(2*(2*({(C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3)))/d_c))))*(Tau_c_BY_Tau/Cd))+(1+Tau_c_BY_Tau)*((kin.vis/(((1+(2*lambda/(2*({(C1*rd^C2
)/(C3*rd^C4-log10(X[,1]))+rd^3}^(1/3))))*(1.257+0.4*exp(-0.55*(2*({(C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3)))/lambda)))*k_B*Temp)/(3*3.1416*dyn.vis*(2*({(C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3)))))))^-1+10^((-3/(((((X[,2])*((2*({(C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3))))^2*(1+(2*lambda/(2*({(C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3))))*(1.257+0.4*exp(-0.55*(2*({(C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3)))/lambda))))/(18*dyn.vis))*(X[,3])^2)/kin.vis))))+((((X[,2])*((2*({(C1*rd^C2)/(C3*rd^
C4-log10(X[,1]))+rd^3}^(1/3))))^2*(1+(2*lambda/(2*({(C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3))))*(1.257+0.4*exp(-0.55*(2*({(C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3)))/lambda))))/(18*dyn.vis))*9.81))^-1)-
(1/(((((X[,2])*(dp_i)^2*(1+(2*lambda/(2*({(C1*rd^C2)/(C3*rd^C4-log10(X[,1]))+rd^3}^(1/3))))*(1.257+0.4*exp(-
0.55*(2*({(C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3)))/lambda))))/(18*dyn.vis))*9.81))))/(exp(((((1+(((((X[,2])*((2*({(C1*rd^C2)/(C3*rd^C4
-log10(X[,1]))+rd^3}^(1/3))))^2*(1+(2*lambda/(2*({(C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3))))*(1.257+0.4*exp(-0.55*(2*({(C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3)))/lambda))))/(18*dyn.vis))*9.81)^2/(X[,3])^2))^0.5)/(k*(X[,3])))*(log((z-zd)/(h_c-
zd))))/(1/(((((X[,2])*(dp_i)^2*(1+(2*lambda/(2*({(C1*rd^C2)/(C3*rd^C4-
```

```
log10(X[,1]))+rd^3}^(1/3))))*(1.257+0.4*exp(-0.55*(2*({(C1*rd^C2)/(C3*rd^C4-
log10(X[,1]))+rd^3}^(1/3)))/lambda))))/(18*dyn.vis))*9.81))))))))^-1)


N <- 100000
x1 = runif(1*N,0.1,1.0)            #RH
x2 = runif(1*N,1500,2000)          #rho
x3 = runif(1*N,0.1,0.5)            #u_star
x4 = runif(1*N,1,5)                #U
x_1 = runif(1*N,0.1,1.0)           #RH
x_2 = runif(1*N,1500,2000)         #rho
x_3 = runif(1*N,0.1,0.5)           #u_star
x_4 = runif(1*N,1,5)               #U
Y1 = matrix(c(x1,x2,x3,x4), nrow=N)
X1 = data.frame(matrix(Y1, nrow=N))
Y2 = matrix(c(x_1,x_2,x_3,x_4), nrow=N)
X2 = data.frame(matrix(Y2, nrow=N))
a = sobol2002(model = model, X1 = X1, X2=X2, nboot = 2000, conf= 0.95);a
```