Supplement

S.1 Aerodynamical resistance

Table S1. Molecular diffusivity for a gas i given as ration $D_{\mathrm{H_2O}}/D_i$, reactivity f_0^i, and effective Henry's Law constant H_*^i.

	$D_{\mathrm{H_2O}}/D_i$	f_0^i	H_*^i ($\mathrm{M\,atm^{-1}}$)
O_3	1.6	1.0	$1{\cdot}10^{-2}$
SO_2	1.9	0.0	$1{\cdot}10^{5}$
NO_2	1.6	0.1	$1{\cdot}10^{-2}$
H_2O_2	1.4	1.0	$1{\cdot}10^{5}$
HCHO	1.3	0.0	$6{\cdot}10^{3}$
NO	1.3	0.0	$2{\cdot}10^{-3}$
CH_3CHO	1.6	15.0	0

S.2 Dynamic viscosity of air and quasi-laminar resistance

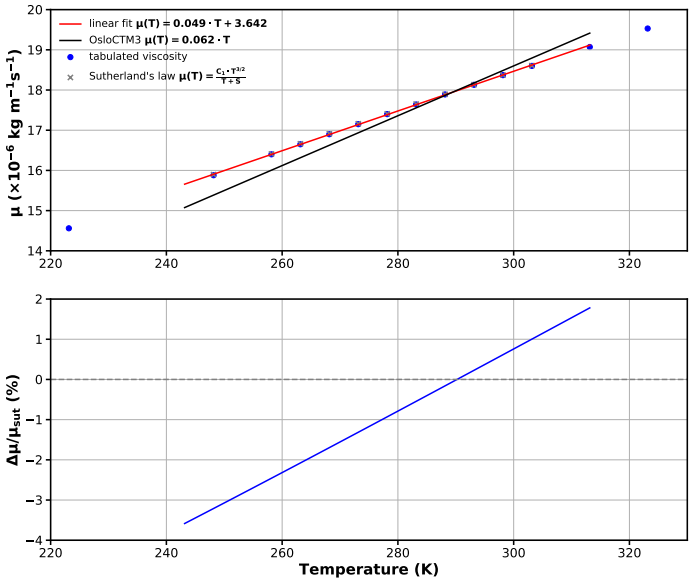


Figure S1. Sutherland's law and divergence linear fits with one or two degrees of freedom. The fit $\mu(T) = m \cdot T$ is implemented in the Oslo CTM3. Despite its strong divergence at lower and higher temperatures, the impact of this choice on the results is negligible.

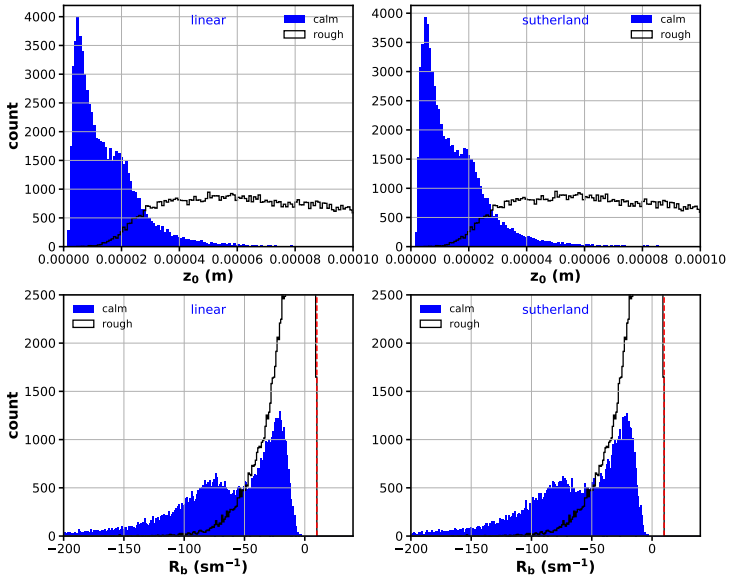
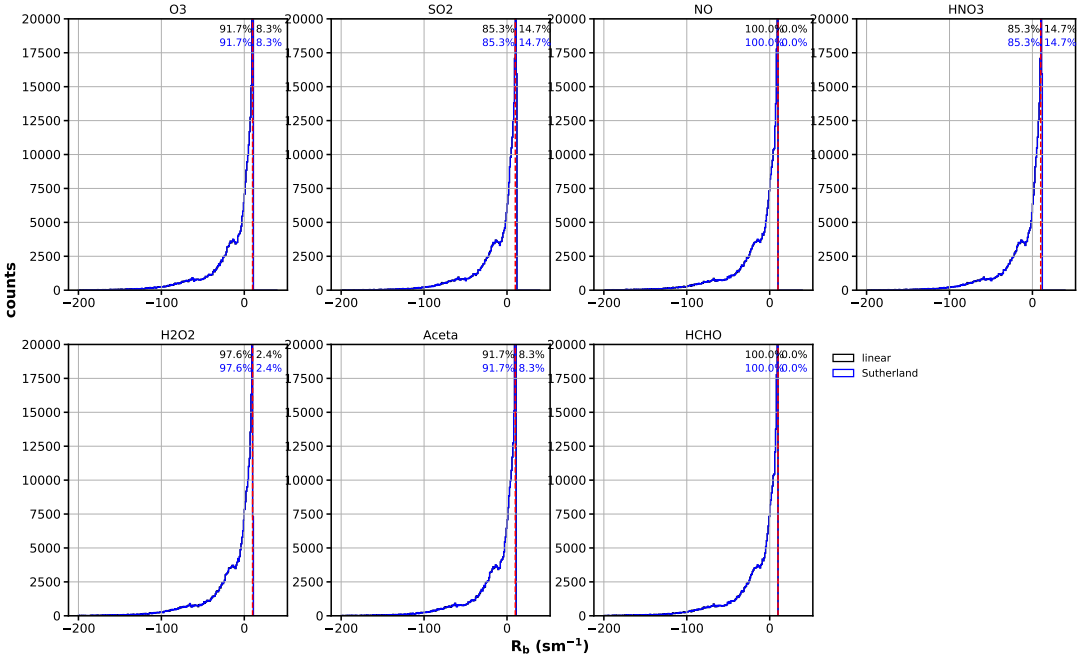
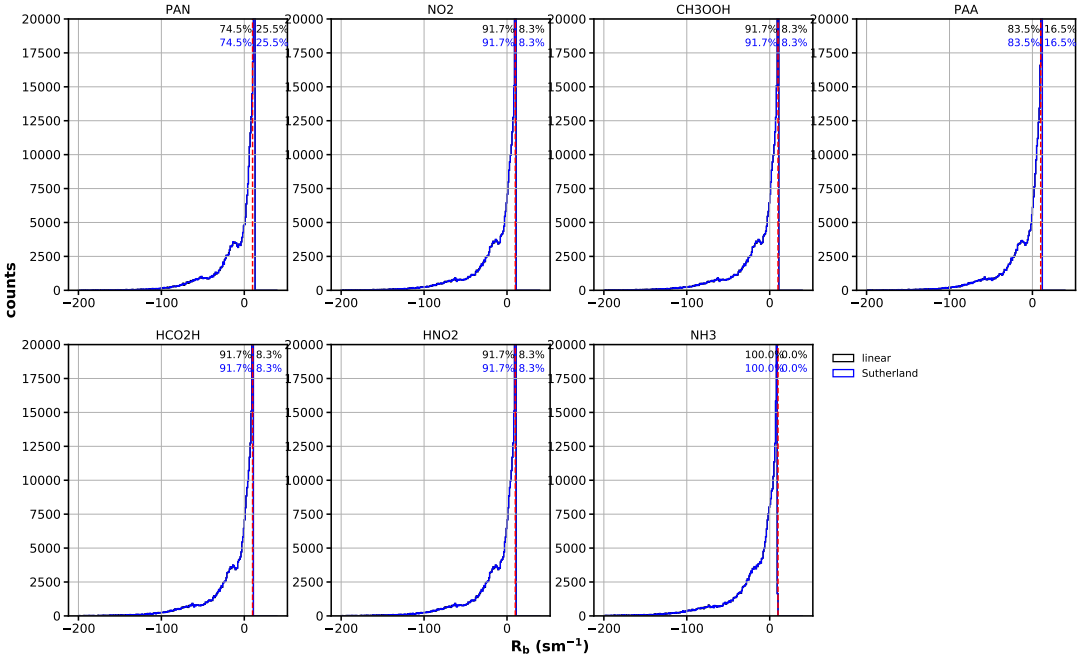


Figure S2. Comparison of resulting z_o and R_b for $\mu(T) = m \cdot T$ (left panel) and Sutherland's law (right panel). The example displays the distributions for H_2O after one day of model integration for both, rough and calm sea case.



(a)



(b)

Figure S3. Resulting R_b for different species. Shown are results of one day model integrations for both, Sutherland's law and $\mu(T) = m \cdot T$. Shown are also the percentages above / below the given threshold.

S.3 Stomatal conductance

Table S2. Tabulated parameters for stomatal conductance computation in the EMEP scheme

Code*	g_{max} ($mmol\,s^{-1}\,m^{-2}$)	f_{min}	ϕ_a	ϕ_b	ϕ_c	ϕ_d	ϕ_e (days)	ϕ_f (days)	ϕ_{AS} (days)	ϕ_{AE} (days)	α_{light}	T_{min} (°C)	T_{opt} (°C)	T_{max} (°C)
CF	140	0.1	0.8	0.8	0.8	0.8	1	1	0	0	0.006	0	18	36
DF	150	0.1	0	0	1	0	20	30	0	0	0.006	0	20	35
NF	200	0.1	1	1	0.2	1	130	60	80	35	0.013	8	25	38
BF	200	0.02	1	1	0.3	1	130	60	80	35	0.009	1	23	39
TC	300	0.1	0.1	1	0.1	0	45	0	0	0.0105	0.01	12	26	40
MC	300	0.019	0.1	0.1	1	0.1	0	45	0	0	0.0048	0	25	51
RC	360	0.02	0.2	0.2	1	0.2	20	45	0	0	0.0023	8	24	50
SNL	60	0.01	1	1	1	1	1	1	0	0	0.009	1	18	36
GR	270	0.01	1	1	1	1	0	0	0	0	0.009	12	26	40
MS	200	0.01	1	1	0.2	1	130	60	80	35	0.012	4	20	37
WE	0	1	1	1	1	1	1	1	1	1	1	0	1	0
TU	0	1	1	1	1	1	1	1	1	1	1	0	1	0
DE	0	1	1	1	1	1	1	1	1	1	1	0	1	0
W	0	1	1	1	1	1	1	1	1	1	1	0	1	0
ICE	0	1	1	1	1	1	1	1	1	1	1	0	1	0
U	0	1	1	1	1	1	1	1	1	1	1	0	1	0

Code*	D_{max} (kPA)	D_{min} (kPa)	D_{crit} (kPa)	R_{SO} ($s\,m^{-1}$)	R_{O_3} ($s\,m^{-1}$)	h (m)	d_{SGS} (day)	d_{EGS} (day)	∇d_{SGS} (days/°lat)	∇d_{EGS} (days/°lat)
CF	0.5	3	1000	0	200	20	0	366	0	0
DF	1	3.25	1000	0	200	20	100	307	1.5	-2.0
NF	1	3.2	1000	0	200	8	0	366	0	0
BF	2.2	4	1000	0	200	15	0	366	0	0
TC	1.2	3.2	8	0	200	1	123	213	2.57	2.57
MC	1	2.5	1000	0	200	2	123	237	2.57	2.57
RC	0.31	2.7	10	0	200	1	130	250	0	0
SNL	1.3	3	1000	0	400	0.5	0	366	0	0
GR	1.3	3	1000	0	1000	0.3	0	366	0	0
MS	1.3	3.2	1000	0	200	2	0	366	0	0
WE	1	0	1000	50	400	0.5	0	366	0	0
TU	1	0	1000	500	400	0.5	0	366	0	0
DE	1	0	1000	1000	2000	0	0	366	0	0
W	1	0	1000	1	2000	0	0	366	0	0
ICE	1	0	1000	1000	2000	0	0	366	0	0
U	1	0	1000	400	400	10	0	366	0	0

*: CF – temperate/boreal coniferous; DF – temperate/boreal deciduous; NF – Mediterranean needleleaf; BF – Mediterranean broadleaf; TC – temperate crop; MC – Mediterranean crop; RC – root crop; SNL – moorland; GR – grass; MS – Mediterranean scrub; WE – wetlands; TU – tundra; DE – desert; W – water; ICE – ice.

S.4 Latitude dependent vegetation height

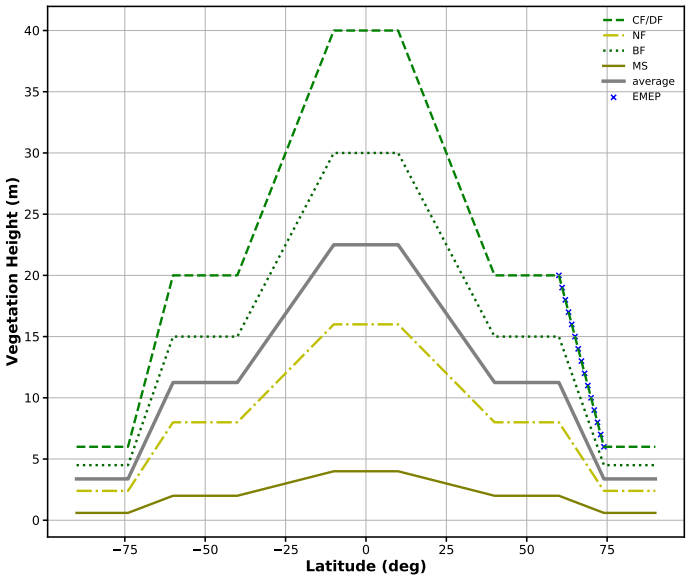


Figure S4. Extension of the latitude dependent vegetation height from northern hemisphere mid latitudes only to a global description. Example heights are shown for coniferous, deciduous, needleleaf, and broadleaf forests as well as Mediterranean shrub. As reference the original range of EMEP is added.

S.5 Temperature dependent greening season (python 2.7)

```
#-------------------------------------------------------------------------------
import numpy as np
import pandas as pd
import xarray as xr
#-------------------------------------------------------------------------------
def start_growing_season_fixed(lat, **kwargs):
    '''
    Begin of growing season parameterization adapted from SMOKE-BEIS.
    '''
    bLeap = kwargs.pop('leap', False)
    if (bLeap):
       MAY31 = 152
       NOV1  = 306
       DEC31 = 366
    else:
       MAY31 = 151
       NOV1  = 305
       DEC31 = 365

    if lat < -65:
        return((0,))
    elif lat < -23:
        return((NOV1,1))
    elif lat <= 23:
        return((1,))
    elif lat < 65:
        return(((lat-23)*4.5,))
    else:
```

```
        return((0,))
#-------------------------------------------------------------------------------
def end_growing_season_fixed(lat, **kwargs):
    '''
    End of growing season parameterization adapted from SMOKE-BEIS.
    '''
    bLeap = kwargs.pop('leap', False)
    if (bLeap):
       MAY31 = 152
       NOV1  = 306
       DEC31 = 366
    else:
       MAY31 = 151
       NOV1  = 305
       DEC31 = 365

    if lat < -65:
        return((0,))
    elif lat < -23:
        return((DEC31,MAY31))
    elif lat <= 23:
        return((DEC31,))
    elif lat < 65:
        return((DEC31-(lat-23)*3.3,))
    else:
        return((0,))
#-------------------------------------------------------------------------------
def growing_season(temperature,**kwargs):
    '''
    Agricultural rule of thumb definition of growing season:
    5 consecutive days above 5 degree Celsius
    and vice versa for end of growing season.
    '''
    # Shift start day of evaluation.
    s_shift = kwargs.pop('s_shift',365/2)
    # Number of days that need to fulfill temperature criteria
    degree_days_crit = kwargs.pop('ddc', 5)
    # Temperature criteria
    temperature_crit = kwargs.pop('tc', 5)
    # Switch to southern hemisphere evaluation
    sh = kwargs.pop('sh', False)
    # Activate verbose
    verbose = kwargs.pop('verbose', False)
    # Counters
    count_gdd = 0
    count_days = 0
    start_gs = (0,False)
    end_gs = (0,False)

    if sh:
    # Shift the temperature by halve a year - shifted back later.
        temp = temperature.roll(time=s_shift)
    else:
        temp = temperature
    for itemp in temp:
        count_days += 1
        if not start_gs[1]:
            if itemp > temperature_crit:
                count_gdd += 1
            else:
                count_gdd = 0
            if count_gdd == degree_days_crit:
                if sh:
                    start_gs = (count_days+s_shift, True)
                else:
                    start_gs = (count_days, True)
                count_gdd = 0
        elif start_gs[1] and not end_gs[1] and count_days>s_shift:
            if itemp <= temperature_crit:
                count_gdd += 1
                if verbose:
                    print(count_days, count_gdd)
            else:
```

```
                count_gdd = 0
            if count_gdd == degree_days_crit:
                if sh:
                    end_gs = (count_days-s_shift, True)
                else:
                    end_gs = (count_days, True)
                count_gdd = 0

    return({'sgs':start_gs,'egs':end_gs})
#-------------------------------------------------------------------------------
def growing_season_stadyn(xr_temp):
    '''
    Preprocess the greening season for OsloCTM3.
    Setting using the 5deg-5days criteria and in case this fails
    or is out of the defined bounds,
    fall back to SMOKE-BEIS parameterization.
    '''
    # Fetch leap year from input data
    bLeap = False
    if xr_temp.time.size==366:
       bLeap = True
    # Use 5deg-5days criteria
    if (xr_temp.lat > 45 and xr_temp.lat < 85):
        gs = growing_season(xr_temp)
        sgs = (gs['sgs'][0],)
        egs = (gs['egs'][0],)
        if ( not gs['sgs'][1] or not gs['egs'][1] or sgs[0]>=egs[0]):
            # Check for failing 5deg-5days criteria => Fall back to fixed
            sgs = start_growing_season_fixed(xr_temp.lat.data, leap=bLeap)
            egs = end_growing_season_fixed(xr_temp.lat.data, leap=bLeap)
    elif (xr_temp.lat < -35 and xr_temp.lat >= -65):
        gs = growing_season(xr_temp, sh=True)
        sgs = (gs['sgs'][0],1)
        egs = (xr_temp.size, gs['egs'][0])
        # Check for failing 5deg-5days criteria => Fall back to fixed
        if (not gs['sgs'][1] or not gs['egs'][1] or sgs[0]>=egs[0]):
            sgs = start_growing_season_fixed(xr_temp.lat.data, leap=bLeap)
            egs = end_growing_season_fixed(xr_temp.lat.data, leap=bLeap)
    # Use SMOKE-BEIS parameterization
    else:
        sgs = start_growing_season_fixed(xr_temp.lat.data, leap=bLeap)
        egs = end_growing_season_fixed(xr_temp.lat.data, leap=bLeap)
    # Generate GDAY and GLEN fields
    # in case no growing season has been allocated
    if (sgs[0]==egs[0]==0):
        gday = np.repeat(0,len(xr_temp.time))
        glen = 0
    # in normal cases
    else:
        # handle northern hemisphere
        if len(sgs)==1:
            gday = np.concatenate((np.repeat(0,int(sgs[0])-1),
                                   np.arange(1,int(egs[0])-int(sgs[0])+1),
                                   np.repeat(0,len(xr_temp.time)-
                                   int(egs[0])+1)))
            glen = int(egs[0])-int(sgs[0])+1
        # handle southern hemisphere
        else:
            gday = np.concatenate((np.arange(1,int(egs[1])+1)+len(xr_temp.time)
                                   -int(sgs[0])+1,
                                   np.repeat(0,int(sgs[0])-int(egs[1])),
                                   np.arange(1,len(xr_temp.time)-int(sgs[0])+1)))
            glen = len(xr_temp.time)-(int(sgs[0])-int(egs[1]))+1
    # This should not happen, but in case it does, print info.
    if not(len(gday)==len(xr_temp.time)):
        print(xr_temp, sgs, egs)
    # Output
    data_gday = xr.DataArray(gday.astype(int),[('time', xr_temp.time.data)])
    return(data_gday, (int)(glen))
#-------------------------------------------------------------------------------
```

S.6 De-accumulation of photosynthetic active radiation (PAR) from OpenIFS

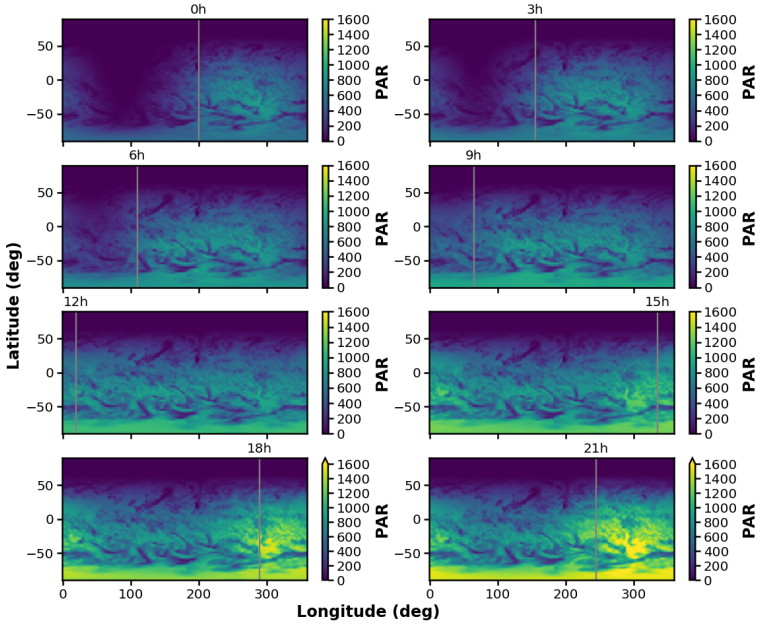


Figure S5. Example output from OpenIFS for January 2nd 2005. PAR is accumulated over the period of one day.

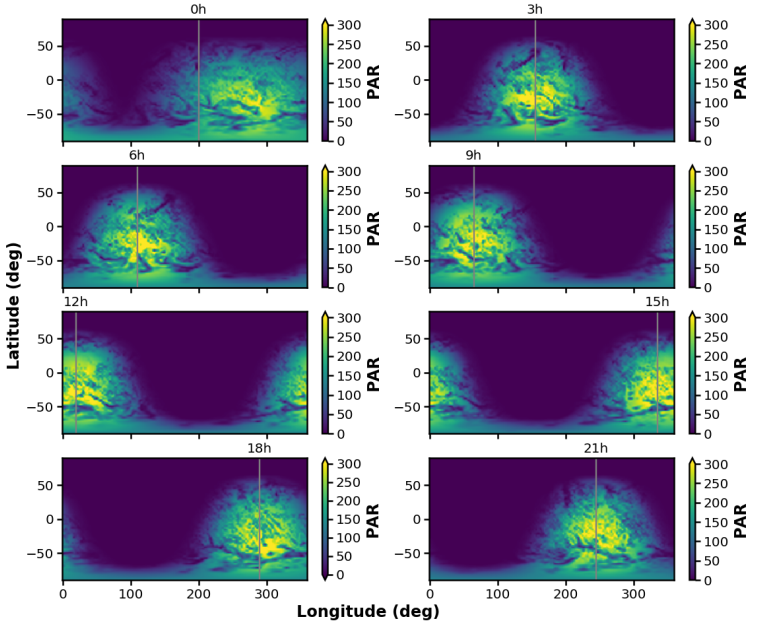


Figure S6. Partly de-accumulated output from OpenIFS. $\mathrm{PPFD}(t_i) = \mathrm{PAR}(t_{i+1}) - \mathrm{PAR}(t_i)$.

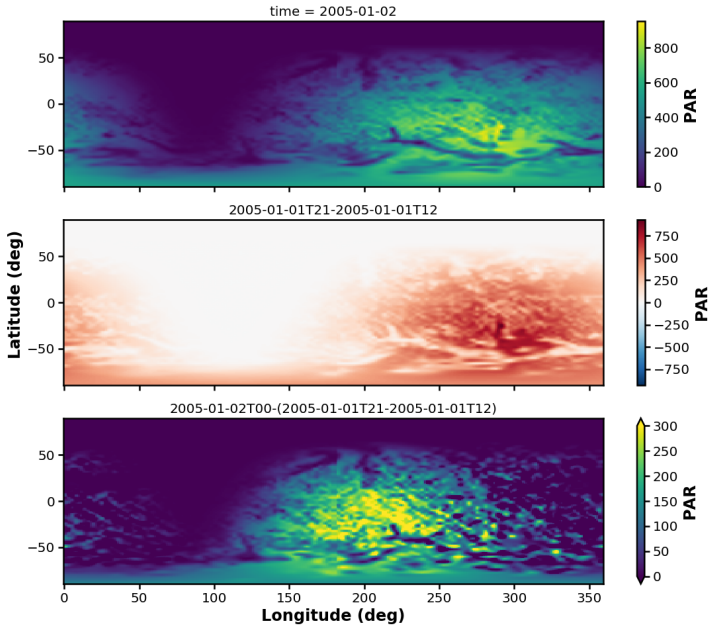


Figure S7. De-accumulation of $t = 00\,\mathrm{UTC}$ in the partly de-accumulated fields. $\mathrm{PPFD}(t = 00\,\mathrm{UTC}) = \mathrm{PAR}(t = 00\,\mathrm{UTC}) - [\mathrm{PAR}(t = 21\,\mathrm{UTC} - 1\,\mathrm{day}) - \mathrm{PAR}(t = 12\,\mathrm{UTC} - 1\,\mathrm{day})]$.

S.7 Normalized zonal average ozone dry deposition velocity

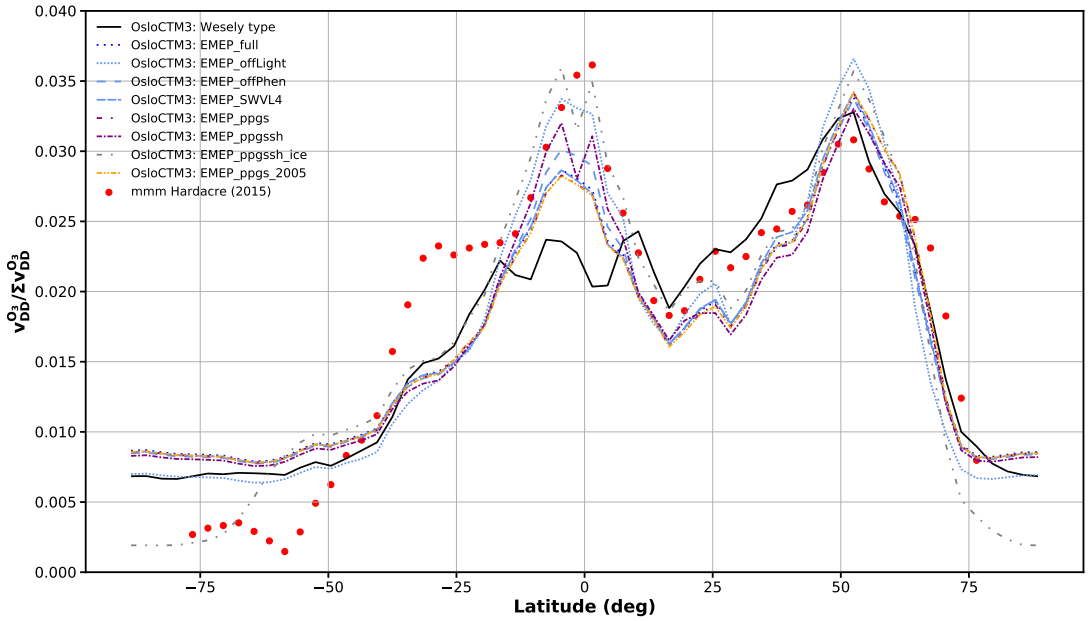


Figure S8. Normalized zonal average ozone dry deposition velocity.