MSCMF / NATIVE NUMERICAL ALGORITHMS

Executable source listing; equations and domains are implemented below. This is code documentation, not theorem certification.

"""MSCMF native screening kernels. SI units; no construction or climate qualification."""

import math

G=9.80665;R=8.314462618;RHO=998.

GASES={'biogas60':.6*.016043+.4*.04401,'methane':.016043,'helium':.0040026,'hydrogen':.00201588,'n
itrogen':.0280134,'air':.02896546}

WING=dict(chord=80.,thickness=20.,span=20.,shapeFactor=.65,voidFraction=.85,pipe1=1.2,pipe2=1.2,st
ructureMass=400.,pressure=99000.,gasOverpressure=500.,airTemperature=315.15,gasTemperature=315.15,
reserve=1.2,wingSpacing=100.,routeLength=1000.,soilK=.5,equivalentDepth=1.,waterTableHeight=.3,rec
harge=.005,drainSlope=.002,manning=.012)

WING.update(windSpeed=25.,tailArea=50.,tailArm=20.,tailCoefficient=1.,opposingTorque=10000.)

WING.update(soilKAbove=.5)

CLIMATE=dict(air_C=42.,soil_C=40.,rh_percent=15.,pressure_kPa=99.,air_target_C=26.,soil_target_C=2
4.,rh_min_percent=40.,rh_max_percent=70.,max_cooling_C=28.,area_m2=10000.,eto_mm_day=6.,crop_facto
r=1.,effective_rain_mm_day=0.,irrigation_efficiency=.9,wet_radius_m=.45,emitter_L_h=2.,available_w
ater_m3_day=100.,spray_height_m=200.,fall_speed_m_s=5.,wind_m_s=3.,droplet_mm=1.5,evap_mm2_s=.08,h
eat_W_m2=250.,mixing_depth_m=10.,local_retention=.5,spray_hours_day=8.)

COAST=dict(length=100000.,demand=1.,diameterA=1.2,diameterB=1.2,headA=400.,headB=400.,peakGround=1
00.,height=200.,outletBar=3.,recovery=.45,sec=3.5,pumpEfficiency=.8)

def params(defaults,values):

    if set(values)-set(defaults):raise ValueError('Unknown input')

    p={**defaults,**values}

    if any(isinstance(v,bool) or not isinstance(v,(float,int)) or not math.isfinite(v) for v in
p.values()):raise ValueError('Finite numbers required')

    return p

def wing(values=None,gas='biogas60'):

    p=params(WING,values or {})

    if gas not in GASES:raise ValueError('Unknown gas')

    if any(v<0 for v in p.values()) or any(p[k]<=0 for k in ['chord','thickness','span','shapeFact
or','voidFraction','pipe1','pipe2','pressure','wingSpacing','routeLength','soilK','waterTableHeigh
t','recharge','drainSlope','manning']):raise ValueError('Positive geometry and soil inputs
required')

    if p['shapeFactor']>1 or p['voidFraction']>1 or p['reserve']<1 or p['thickness']>p['chord'] or
not 200<=p['airTemperature']<=400 or not 200<=p['gasTemperature']<=400:raise ValueError('Outside
screening domain')

    air=p['pressure']*.02896546/(R*p['airTemperature']);rho=(p['pressure']+p['gasOverpressure'])*G
ASES[gas]/(R*p['gasTemperature'])

    volume=p['chord']*p['thickness']*p['shapeFactor']*p['voidFraction'];water=RHO*math.pi*(p['pipe
1']**2+p['pipe2']**2)/4;mass=water+p['structureMass'];lift=(air-rho)*volume

    spacing=math.sqrt((8*p['soilK']*p['equivalentDepth']*p['waterTableHeight']+4*p['soilKAbove']*p
['waterTableHeight']**2)/p['recharge']);count=math.ceil(p['wingSpacing']/spacing);actual=p['wingSp
acing']/count;flow=p['recharge']*actual*p['routeLength']/86400

    diam=next((d for d in [.05,.063,.075,.09,.11,.125,.16,.2,.25,.315,.4,.5,.63,.8,1.,1.2,1.6,2.]
if math.pi*d*d/4*(d/4)**(2/3)*math.sqrt(p['drainSlope'])/p['manning']>=flow),None)

```
    moment=.5*air*p['windSpeed']**2*p['tailArea']*p['tailCoefficient']*math.sin(math.pi/12)*p['tai
lArm']

    return dict(inputs=p,gas=gas,status='SCREENING_ONLY',air_density_kg_m3=air,gas_density_kg_m3=r
ho,gas_void_m3_per_m=volume,water_mass_kg_m=water,mass_kg_m=mass,net_lift_kg_m=lift,reserved_load_
kg_m=p['reserve']*mass,lift_margin_kg_m=lift-p['reserve']*mass,lift_check='STATIC_ONLY' if
lift>=p['reserve']*mass else 'FAIL',lift_check_scope='FREE_STATIC_SUPPORT_ONLY_NOT_CABLE_SUPPORTED
_FEASIBILITY',minimum_additional_support_kg_m=max(0,mass-
lift),required_void_m3_m=p['reserve']*mass/(air-rho) if air>rho else None,half_catchment_m=p['wing
Spacing']/2,max_drain_spacing_m=spacing,parallel_drains_per_bay=count,actual_drain_spacing_m=actua
l,drain_flow_m3_s=flow,drain_ID_m=diam,tail_moment_15deg_Nm=moment,tail_check='STATIC_ONLY' if
moment>p['opposingTorque'] else 'FAIL',yaw_dynamic_stability='NOT_SOLVED',gas_production='NOT_DESI
GNED',sustainability='NOT_QUANTIFIED',structure='NOT_QUALIFIED')

def sat(t):return .61094*math.exp(17.625*t/(t+243.04))

def humidity(t,rh,p):

    pv=sat(t)*rh/100

    return .62198*pv/(p-pv)

def enthalpy(t,w):return 1.006*t+w*(2501+1.86*t)

def climate(values=None):

    p=params(CLIMATE,values or {})

    for k,v in p.items():

        if not k.endswith('_C') and v<0:raise ValueError('Nonnegative '+k)

    if not 0<=p['air_C']<=50 or not 0<=p['air_target_C']<=50 or not 70<=p['pressure_kPa']<=110 or
not 0<=p['rh_percent']<=100 or not 0<=p['rh_min_percent']<=p['rh_max_percent']<=100 or not
0<p['irrigation_efficiency']<=1 or not 0<=p['max_cooling_C']<=28:raise ValueError('Outside climate
model domain')

    if not 0<p['local_retention']<=1 or p['spray_hours_day']>24:raise ValueError('Invalid
retention or duty time')

    for k in ['area_m2','wet_radius_m','emitter_L_h','fall_speed_m_s','droplet_mm']:

        if p[k]<=0:raise ValueError('Positive '+k)

    w=humidity(p['air_C'],p['rh_percent'],p['pressure_kPa']);h=enthalpy(p['air_C'],w);lo=0.;hi=p['
air_C']

    if enthalpy(0,humidity(0,100,p['pressure_kPa']))>h:raise ValueError('Wet bulb below supported
range')

    for _ in range(70):

        m=(lo+hi)/2

        if enthalpy(m,humidity(m,100,p['pressure_kPa']))>h:hi=m

        else:lo=m

    wet=(lo+hi)/2;dew=None

    if p['rh_percent']>0:

a=math.log(p['rh_percent']/100)+17.625*p['air_C']/(243.04+p['air_C']);dew=243.04*a/(17.625-a)

    target=max(p['air_target_C'],p['air_C']-p['max_cooling_C']);needed=max(0.,p['air_C']-target)

    wtarget=(h-
1.006*target)/(2501+1.86*target);pvtarget=p['pressure_kPa']*wtarget/(.62198+wtarget);rhout=100*pvt
arget/sat(target) if wtarget>=0 else None
```

```
    vday=p['area_m2']*max(0,p['eto_mm_day']*p['crop_factor']-
p['effective_rain_mm_day'])/1000/p['irrigation_efficiency'];spacing=math.sqrt(2)*p['wet_radius_m']
;side=math.sqrt(p['area_m2']);n=math.ceil(side/spacing);actual=side/n;flow=n*n*p['emitter_L_h']/3.
6e6

    flight=p['spray_height_m']/p['fall_speed_m_s'];f=1-(max(0,p['droplet_mm']**2-
p['evap_mm2_s']*flight)/p['droplet_mm']**2)**1.5

    dry_density=(p['pressure_kPa']*1000-
sat(p['air_C'])*p['rh_percent']*10)/(287.05*(p['air_C']+273.15));dry_flow=dry_density*p['wind_m_s'
]*p['mixing_depth_m']*side

    heat=p['area_m2']*p['heat_W_m2']+dry_flow*(1006+1860*w)*needed;evap=heat/(2.43e6*p['local_rete
ntion']);spray=evap/(RHO*f) if f>0 else None;capacity=dry_flow*max(0,humidity(target,100,p['pressu
re_kPa'])-w);total=vday+spray*3600*p['spray_hours_day'] if spray is not None else None

    return dict(inputs=p,status='USER_TARGETS_NOT_UNIVERSAL_OPTIMA',wet_bulb_C=wet,dew_point_C=dew
,vapor_pressure_deficit_kPa=sat(p['air_C'])*(1-
p['rh_percent']/100),effective_air_target_C=target,requested_air_cooling_C=needed,evaporative_limi
t_check='NECESSARY_CONDITION_ONLY' if target>=wet else
'FAIL',adiabatic_outlet_RH_percent=rhout,humidity_target_check='NECESSARY_CONDITION_ONLY' if rhout
is not None and p['rh_min_percent']<=rhout<=p['rh_max_percent'] else 'FAIL_OR_INAPPLICABLE',soil_c
ooling_requested_C=max(0,p['soil_C']-
p['soil_target_C']),soil_temperature_prediction='NOT_SOLVED',drip_row_spacing_m=actual,emitter_spa
cing_m=actual,parallel_drip_rows=n,emitter_count=n*n,drip_water_m3_day=vday,drip_flow_m3_s=flow,dr
ip_hours_day=vday/(flow*3600),drip_water_check='MODEL_ONLY' if vday<=p['available_water_m3_day']
else 'FAIL',spray_flight_s=flight,spray_drift_m=flight*p['wind_m_s'],evaporated_fraction=f,spray_f
low_m3_s=spray,total_water_m3_day=total,total_water_check='MODEL_ONLY' if total is not None and
total<=p['available_water_m3_day'] else
'FAIL',evaporation_capacity_check='NECESSARY_CONDITION_ONLY' if evap<=capacity else
'FAIL',plant_growth='NOT_DEMONSTRATED',target_basis='USER_ASSUMPTIONS_REQUIRES_SPECIES_AND_SITE')

def headloss(length,q_per_m,d,steps=160):

    total=0.

    for i in range(steps):

        v=4*q_per_m*length*(i+.5)/steps/(math.pi*d*d);re=v*d/1.004e-6

        if not v:continue

        f=64/re if re<2300 else .05 if re<5000 else .25/math.log10(1.5e-6/(3.7*d)+5.74/re**.9)**2

        total+=f/d*v*v/(2*G)*length/steps

    return total

def coast(values=None):

    p=params(COAST,values or {})

    if any(v<0 for v in p.values()) or any(p[k]<=0 for k in
['length','demand','diameterA','diameterB','outletBar','recovery','pumpEfficiency']) or
p['recovery']>=1 or p['pumpEfficiency']>1:raise ValueError('Invalid hydraulic inputs')

    q=p['demand']/p['length'];L=p['length']

    def residual(x):return
p['headA']-headloss(x,q,p['diameterA'])-p['headB']+headloss(L-x,q,p['diameterB'])

    if residual(0)*residual(L)>0:raise ValueError('Inward flow from both coasts unsupported at
these heads')

    lo=0.;hi=L

    for _ in range(65):
```

```
        m=(lo+hi)/2
        if residual(m)>0:lo=m
        else:hi=m
    x=(lo+hi)/2;qa=q*x;qb=p['demand']-qa;hm=p['headA']-headloss(x,q,p['diameterA']);profile=[]
    for z in sorted([i*L/100 for i in range(101)]+[x,L/2]):
        elev=p['height']+p['peakGround']*(1-abs(2*z/L-1));H=hm+headloss(abs(z-x),q,p['diameterA']
if z<=x else p['diameterB']);profile.append(dict(x_m=z,inlet_bar=(H-elev)*RHO*G/1e5))
    def plant(v):return dict(permeate_m3_day=v*86400,feed_m3_day=v*86400/p['recovery'],brine_m3_da
y=v*86400*(1/p['recovery']-1))
    minimum=min(z['inlet_bar'] for z in profile)
    return dict(inputs=p,status='UNIFORM_WITHDRAWAL_SCREENING',meeting_m=x,flow_A_m3_s=qa,flow_B_m
3_s=qb,head_residual_m=residual(x),min_inlet_bar=minimum,max_inlet_bar=max(z['inlet_bar'] for z in
profile),outlet_check='REGULATORS_STILL_REQUIRED' if minimum>=p['outletBar'] else 'FAIL',SWRO_A=pl
ant(qa),SWRO_B=plant(qb),pump_kW=RHO*G*(qa*p['headA']+qb*p['headB'])/p['pumpEfficiency']/1000,desa
lination_kW=p['demand']*3600*p['sec'],profile=profile)
```