```
%cython
from sage.misc.misc import srange
from sage.symbolic.constants import pi
cdef extern from "math.h":
    cdef int floor "floor" (double)
    cdef double sin "sin" (double)
    cdef double sqrt "sqrt" (double)
cdef double a_phi = 0.9
cdef double b_phi = 0.25
cdef double mypi = 3.14159265358979323846264338 33
cdef double animalmass = 240 # mass in grams
cdef double mr = animalmass/70000.0 # mass ratio of 13-liner to human
cdef double ve = 1.0 # volume scaling exponent
cdef double tisAfraction = 0.5
cdef double tisBfraction = 0.5
cdef double tismassA = animalmass*0.9*tisAfraction # mass of tissue compartment A in grams
cdef double tismassB = animalmass*0.9*tisBfraction # mass of tissue compartment B in grams
cdef double re = -0.94 # resistance scaling exponent.
cdef double re1 = -0.89 # resistance scaling exponent.
cdef double re_sys = -0.84 # resistance scaling exponent.
cdef double ce = 1.0 # compliance scaling exponent
cdef double le = -0.75 # inductance scaling exponent
cdef double ee = -1.0 # elastance scaling exponent
cdef double v_m = 0.07 # viscosity slope vs temp
cdef double E_minlv = mr**(ee)*0.049 # mmHg/ml
cdef double E_maxlv = mr**(ee)*2.49 # mmHg/ml
cdef double E_minrv = mr**(ee)*0.0243 # mmHg/ml
cdef double E_maxrv = mr**(ee)*0.523 # mmHg/ml
cdef double E_ra = mr**(ee)*0.06 # mmHg/ml
cdef double E_la = mr**(ee)*0.075 # mmHg/ml
cdef double V_dlv = mr**(ve)*10  # ml; unstressed volume.
cdef double V_dla = mr**(ve)*30 # ml; unstressed volume.
cdef double V_maxlvb = mr**(ve)*1.5 # ml; max backflow through aortic valve into left ventricle
cdef double V_maxrvb = mr**(ve)*1.5 # ml; max backflow right ventricle (pulmonary)
cdef double V_maxlab = mr**(ve)*1.5 # ml; max backflow left mitral
cdef double V_maxrab = mr**(ve)*1.5 # ml; max backflow right tricuspid
cdef double V_drv = mr**(ve)*10 # ml; unstressed volume.
cdef double V_dra = mr**(ve)*30 # ml; unstressed volume.
cdef double R_0s = mr**(re_sys)*0.0334 # mmHg*s/ml; systemic resistance
cdef double R_a1 = mr**(re1)*0.0824 # mmHg*s/ml
cdef double R_a2 = mr**(re)*0.178 # mmHg*s/ml
cdef double R_a3A = mr**(re)*0.333 # mmHg*s/ml
cdef double R_a3B = mr**(re)*0.333 # mmHg*s/ml
cdef double R_v1A = mr**(re)*0.0112 # mmHg*s/ml
cdef double R_v1B = mr**(re)*0.0112 # mmHg*s/ml
cdef double R_v2 = mr**(re)*0.0267 # mmHg*s/ml
cdef double R_0p = mr**(re_sys)*0.0251 # mmHg*s/ml
cdef double R_p1 = mr**(re1)*0.0227 # mmHg*s/ml
cdef double R_p2 = mr**(re)*0.0530 # mmHg*s/ml
cdef double R_p3 = mr**(re)*0.0379 # mmHg*s/ml
```

```
cdef double R_l1 = mr**(re)*0.0252 # mmHg*s/ml
cdef double R_l2 = mr**(re)*0.0126 # mmHg*s/ml
cdef double uv_factor = 0.8 # unstressed volume factor
cdef double vfactor = 1.065 # volume factor
cdef double C_a1 = mr**(ce)*0.777 # ml/mmHg
cdef double C_a2 = mr**(ce)*1.64 # ml/mmHg
cdef double C_a3A = mr**(ce)*1.81 # ml/mmHg
cdef double C_a3B = mr**(ce)*1.81 # ml/mmHg
cdef double C_v1A = mr**(ce)*13.24 # ml/mmHg
cdef double C_v1B = mr**(ce)*13.24 # ml/mmHg
cdef double C_v2 = mr**(ce)*73.88 # ml/mmHg
cdef double C_p1 = mr**(ce)*2.222 # ml/mmHg
cdef double C_p2 = mr**(ce)*1.481 # ml/mmHg
cdef double C_p3 = mr**(ce)*1.778 # ml/mmHg
cdef double C_l1 = mr**(ce)*6.666 # ml/mmHg
cdef double C_l2 = mr**(ce)*5.0 # ml/mmHg
cdef double L_a1 = mr**(le)*0.00005 # mmHg*s**2/ml
cdef double L_v2 = mr**(le)*0.00005 # mmHg*s**2/ml
cdef double L_p1 = mr**(le)*0.00005 # mmHg*s**2/ml
cdef double L_l2 = mr**(le)*0.00005 # mmHg*s**2/ml
cdef double V_una1 = mr**(ve)*205*uv_factor # ml; unstressed volume.
cdef double V_una2 = mr**(ve)*370*uv_factor # ml; unstressed volume.
cdef double A_bfrac = 0.7 # fraction of blood in A compartments
cdef double B_bfrac = 0.3 # fraction of blood in B compartments
cdef double V_una3A = mr**(ve)*400*A_bfrac*uv_factor # ml; unstressed volume.
cdef double V_una3B = mr**(ve)*400*B_bfrac*uv_factor # ml; unstressed volume.
cdef double V_unv1A= mr**(ve)*600*A_bfrac*uv_factor # ml; unstressed volume.
cdef double V_unv1B = mr**(ve)*600*B_bfrac*uv_factor # ml; unstressed volume.
cdef double V_unv2 = mr**(ve)*1938*uv_factor # ml; unstressed volume.
cdef double V_unp1 = mr**(ve)*50*uv_factor # ml
cdef double V_unp2 = mr**(ve)*30*uv_factor # ml
cdef double V_unp3 = mr**(ve)*53*uv_factor # ml
cdef double V_unl1 = mr**(ve)*75*uv_factor # ml
cdef double V_unl2 = mr**(ve)*75*uv_factor # ml
cdef double R_la = mr**(re1)*0.000089
cdef double R_ra = mr**(re1)*0.0000594
cdef double L_lv = mr**(le)*0.000416
cdef double L_la = mr**(le)*0.00005
cdef double L_rv = mr**(le)*0.000206
cdef double L_ra = mr**(le)*0.00005
cdef double k_mrt = 6.1728 # conversion between O2 use and heat production degrees/(O2conc)
cdef double kamb_change = 1.1
# gms per sec heat exchange constant for tissue compartment A:
cdef double k_ambA = kamb_change*.2*120*(animalmass/240.0)**(-.33)/3600.0
# gms per sec heat exchange constant for tissue compartment B:
cdef double k_ambB = kamb_change*.2*120*(animalmass/240.0)**(-.33)/3600.0
cdef double k_tisA = .4 #  heat exchange for tissue-blood; units are ml/sec
cdef double k_tisB = .4 #  heat exchange for tissue-blood; units are ml/sec
cdef double O_in = 0.19   # pulmonary oxygen concentration

cdef double sigmoid(double tmp):
```

```
    '''A sigmoidal (Goldbeter-Koshland) function'''
    cdef double a1 = 15.0
    cdef double a2 = 0.05
    cdef double a3 = 0.25
    cdef double a4 = 2.08
    cdef double denom = tmp*a3+a1*a2-tmp+a1+sqrt(4*tmp*a3*(tmp-a1) + (tmp*a3+a1*a2-tmp+a1)**2)
    return -4*tmp*a3/denom + a4

cdef double t_ce_t(double tth, double tmp):
    '''Contraction time as function of heart period and temperature'''
    cdef double sp = 0.2*tth + 0.0692
    cdef double sig = sigmoid(tmp)
    cdef double wavg = tmp/(36-tth)+0.03
    return sig*wavg +(1-wavg)*sp

cdef double phi(double t, double tth, double tmp):
    '''Elastance shape function'''
    cdef double t_temp
    t_temp = t - tth*floor(t/tth)
    if t_temp < t_ce_t(tth, tmp):
        return a_phi*sin(mypi*t_temp/t_ce_t(tth, tmp)) - b_phi*sin(2*mypi*t_temp/t_ce_t(tth, tmp))
    else:
        return 0

cdef double E_tf(double t):
    '''Temperature effect on maximum elastance'''
    return (35.0-t)*(-.87)/30.0 + 1.0

cdef double r3affun(double T):
    '''Vasoconstriction function'''
    cdef double out_vaso = 1.0 + (35 - T)*1.8/28.0
    if T < 35.0:
        return out_vaso
    else:
        return 1.0

cdef double vis(double temp):
    ''' A simple viscosity function'''
    if 1 + v_m*(37.0 - temp) > 0:
        return 1 + v_m*(37.0 - temp)
    else:
        return 1.0

cdef double E_lv(double phi, double t):
    '''Left ventricle elastance function'''
    return E_minlv*(1-phi) + E_maxlv*phi*E_tf(t)

cdef double E_rv(double phi, double t):
    '''Right ventricle elastance function'''
    return E_minrv*(1-phi) + E_maxrv*phi*E_tf(t)
```

```
class pyCardio:
    '''Container class for model data'''
    pass

cdef class Cardio:
    cdef int n
    cdef double tth
    cdef double* t
    cdef double* Q_la
    cdef double* Q_ra
    cdef double* Q_lv
    cdef double* Q_rv
    cdef double* V_lv
    cdef double* V_rv
    cdef double* V_la
    cdef double* V_ra
    cdef double* V_a1
    cdef double* V_a2
    cdef double* V_a3A
    cdef double* V_a3B
    cdef double* V_v1A
    cdef double* V_v1B
    cdef double* V_v2
    cdef double* V_p1
    cdef double* V_p2
    cdef double* V_p3
    cdef double* V_l1
    cdef double* V_l2
    cdef double* p_lv
    cdef double* p_rv
    cdef double* p_a1
    cdef double* p_a2
    cdef double* p_a3A
    cdef double* p_a3B
    cdef double* p_v1A
    cdef double* p_v1B
    cdef double* p_v2
    cdef double* p_p1
    cdef double* p_p2
    cdef double* p_p3
    cdef double* p_l1
    cdef double* p_l2
    cdef double* p_ra
    cdef double* p_la
    cdef double* p_as
    cdef double* p_ap
    cdef double* lvb
    cdef double* lab
    cdef double* rab
    cdef double* rvb
    cdef double* Q_a1
```

```
cdef double* Q_a2A
cdef double* Q_a2B
cdef double* Q_a3A
cdef double* Q_a3B
cdef double* Q_v1A
cdef double* Q_v1B
cdef double* Q_v2
cdef double* Q_l1
cdef double* Q_l2
cdef double* Q_p1
cdef double* Q_p2
cdef double* Q_p3
cdef double* T_a1
cdef double* O_a1
cdef double* T_a2
cdef double* O_a2
cdef double* T_a3A
cdef double* O_a3A
cdef double* T_a3B
cdef double* O_a3B
cdef double* T_l1
cdef double* O_l1
cdef double* T_l2
cdef double* O_l2
cdef double* T_la
cdef double* O_la
cdef double* T_lv
cdef double* O_lv
cdef double* T_p1
cdef double* O_p1
cdef double* T_p2
cdef double* O_p2
cdef double* T_p3
cdef double* O_p3
cdef double* T_ra
cdef double* O_ra
cdef double* T_rv
cdef double* O_rv
cdef double* T_v1A
cdef double* O_v1A
cdef double* T_v1B
cdef double* O_v1B
cdef double* T_v2
cdef double* O_v2
cdef double* T_tisA
cdef double* T_tisB
cdef double* mrateA
cdef double* mrateB
cdef double last_fired
cdef double firing_begin
cdef int firing
```

```
    cdef double firing_tth
    cdef double phi_val

    def __init__(self, int n, double tth, double t_start = 0.0, double init_temp = 37.0, \
double init_O = O_in, double Oa_v_diff = .65, double init_mrateA = .00128, \
sdouble init_mrateB = .00128):
        self.n = n
        self.tth = tth
        self.firing_tth = tth
        self.firing = 0
        self.last_fired = t_start
        self.phi_val = 0
        self.t = <double*> sage_malloc(sizeof(double)*(n+1))
        self.Q_la = <double*> sage_malloc(sizeof(double)*(n+1))
        self.Q_ra = <double*> sage_malloc(sizeof(double)*(n+1))
        self.Q_lv = <double*> sage_malloc(sizeof(double)*(n+1))
        self.Q_rv = <double*> sage_malloc(sizeof(double)*(n+1))
        self.V_lv = <double*> sage_malloc(sizeof(double)*(n+1))
        self.V_rv = <double*> sage_malloc(sizeof(double)*(n+1))
        self.V_la = <double*> sage_malloc(sizeof(double)*(n+1))
        self.V_ra = <double*> sage_malloc(sizeof(double)*(n+1))
        self.V_a1 = <double*> sage_malloc(sizeof(double)*(n+1))
        self.V_a2 = <double*> sage_malloc(sizeof(double)*(n+1))
        self.V_a3A  = <double*> sage_malloc(sizeof(double)*(n+1))
        self.V_a3B = <double*> sage_malloc(sizeof(double)*(n+1))
        self.V_v1A = <double*> sage_malloc(sizeof(double)*(n+1))
        self.V_v1B = <double*> sage_malloc(sizeof(double)*(n+1))
        self.V_v2 = <double*> sage_malloc(sizeof(double)*(n+1))
        self.V_p1 = <double*> sage_malloc(sizeof(double)*(n+1))
        self.V_p2 = <double*> sage_malloc(sizeof(double)*(n+1))
        self.V_p3 = <double*> sage_malloc(sizeof(double)*(n+1))
        self.V_l1 = <double*> sage_malloc(sizeof(double)*(n+1))
        self.V_l2 = <double*> sage_malloc(sizeof(double)*(n+1))
        self.p_lv = <double*> sage_malloc(sizeof(double)*(n+1))
        self.p_rv = <double*> sage_malloc(sizeof(double)*(n+1))
        self.p_a1 = <double*> sage_malloc(sizeof(double)*(n+1))
        self.p_a2 = <double*> sage_malloc(sizeof(double)*(n+1))
        self.p_a3A = <double*> sage_malloc(sizeof(double)*(n+1))
        self.p_a3B = <double*> sage_malloc(sizeof(double)*(n+1))
        self.p_v1A = <double*> sage_malloc(sizeof(double)*(n+1))
        self.p_v1B = <double*> sage_malloc(sizeof(double)*(n+1))
        self.p_v2 = <double*> sage_malloc(sizeof(double)*(n+1))
        self.p_p1 = <double*> sage_malloc(sizeof(double)*(n+1))
        self.p_p2 = <double*> sage_malloc(sizeof(double)*(n+1))
        self.p_p3 = <double*> sage_malloc(sizeof(double)*(n+1))
        self.p_l1 = <double*> sage_malloc(sizeof(double)*(n+1))
        self.p_l2 = <double*> sage_malloc(sizeof(double)*(n+1))
        self.p_ra = <double*> sage_malloc(sizeof(double)*(n+1))
        self.p_la = <double*> sage_malloc(sizeof(double)*(n+1))
        self.p_as = <double*> sage_malloc(sizeof(double)*(n+1))
        self.p_ap = <double*> sage_malloc(sizeof(double)*(n+1))
```

```
self.lvb = <double*> sage_malloc(sizeof(double)*(n+1))
self.lab = <double*> sage_malloc(sizeof(double)*(n+1))
self.rab = <double*> sage_malloc(sizeof(double)*(n+1))
self.rvb = <double*> sage_malloc(sizeof(double)*(n+1))
self.Q_a1 = <double*> sage_malloc(sizeof(double)*(n+1))
self.Q_a2A = <double*> sage_malloc(sizeof(double)*(n+1))
self.Q_a2B = <double*> sage_malloc(sizeof(double)*(n+1))
self.Q_a3A = <double*> sage_malloc(sizeof(double)*(n+1))
self.Q_a3B = <double*> sage_malloc(sizeof(double)*(n+1))
self.Q_v1A = <double*> sage_malloc(sizeof(double)*(n+1))
self.Q_v1B = <double*> sage_malloc(sizeof(double)*(n+1))
self.Q_v2 = <double*> sage_malloc(sizeof(double)*(n+1))
self.Q_l1 = <double*> sage_malloc(sizeof(double)*(n+1))
self.Q_l2 = <double*> sage_malloc(sizeof(double)*(n+1))
self.Q_p1 = <double*> sage_malloc(sizeof(double)*(n+1))
self.Q_p2 = <double*> sage_malloc(sizeof(double)*(n+1))
self.Q_p3 = <double*> sage_malloc(sizeof(double)*(n+1))
self.T_a1 = <double*> sage_malloc(sizeof(double)*(n+1))
self.O_a1 = <double*> sage_malloc(sizeof(double)*(n+1))
self.T_a2 = <double*> sage_malloc(sizeof(double)*(n+1))
self.O_a2 = <double*> sage_malloc(sizeof(double)*(n+1))
self.T_a3A = <double*> sage_malloc(sizeof(double)*(n+1))
self.O_a3A = <double*> sage_malloc(sizeof(double)*(n+1))
self.T_a3B = <double*> sage_malloc(sizeof(double)*(n+1))
self.O_a3B = <double*> sage_malloc(sizeof(double)*(n+1))
self.T_l1 = <double*> sage_malloc(sizeof(double)*(n+1))
self.O_l1 = <double*> sage_malloc(sizeof(double)*(n+1))
self.T_l2 = <double*> sage_malloc(sizeof(double)*(n+1))
self.O_l2 = <double*> sage_malloc(sizeof(double)*(n+1))
self.T_la = <double*> sage_malloc(sizeof(double)*(n+1))
self.O_la = <double*> sage_malloc(sizeof(double)*(n+1))
self.T_lv = <double*> sage_malloc(sizeof(double)*(n+1))
self.O_lv = <double*> sage_malloc(sizeof(double)*(n+1))
self.T_p1 = <double*> sage_malloc(sizeof(double)*(n+1))
self.O_p1 = <double*> sage_malloc(sizeof(double)*(n+1))
self.T_p2 = <double*> sage_malloc(sizeof(double)*(n+1))
self.O_p2 = <double*> sage_malloc(sizeof(double)*(n+1))
self.T_p3 = <double*> sage_malloc(sizeof(double)*(n+1))
self.O_p3 = <double*> sage_malloc(sizeof(double)*(n+1))
self.T_ra = <double*> sage_malloc(sizeof(double)*(n+1))
self.O_ra = <double*> sage_malloc(sizeof(double)*(n+1))
self.T_rv = <double*> sage_malloc(sizeof(double)*(n+1))
self.O_rv = <double*> sage_malloc(sizeof(double)*(n+1))
self.T_v1A = <double*> sage_malloc(sizeof(double)*(n+1))
self.O_v1A = <double*> sage_malloc(sizeof(double)*(n+1))
self.T_v1B = <double*> sage_malloc(sizeof(double)*(n+1))
self.O_v1B = <double*> sage_malloc(sizeof(double)*(n+1))
self.T_v2 = <double*> sage_malloc(sizeof(double)*(n+1))
self.O_v2 = <double*> sage_malloc(sizeof(double)*(n+1))
self.T_tisA = <double*> sage_malloc(sizeof(double)*(n+1))
self.T_tisB = <double*> sage_malloc(sizeof(double)*(n+1))
```

```
        self.mrateA = <double*> sage_malloc(sizeof(double)*(n+1))
        self.mrateB = <double*> sage_malloc(sizeof(double)*(n+1))
        self.t[0] = t_start
        self.Q_la[0] = 0.0
        self.Q_ra[0] = 0.0
        self.Q_lv[0] = 0.0
        self.Q_rv[0] = 0.0
        self.V_lv[0] = mr**(ve)*115.0*vfactor
        self.V_rv[0] = mr**(ve)*126.0*vfactor
        self.V_la[0] = mr**(ve)*102.0*vfactor
        self.V_ra[0] = mr**(ve)*81.0*vfactor
        self.V_a1[0] = mr**(ve)*259.0*vfactor
        self.V_a2[0] = mr**(ve)*483.0*vfactor
        self.V_a3A[0] = mr**(ve)*510.0*vfactor*A_bfrac
        self.V_a3B[0] = mr**(ve)*510.0*vfactor*B_bfrac
        self.V_v1A[0] = mr**(ve)*692.0*vfactor*A_bfrac
        self.V_v1B[0] = mr**(ve)*692.0*vfactor*B_bfrac
        self.V_v2[0] = mr**(ve)*2333.0*vfactor
        self.V_p1[0] = mr**(ve)*77.0*vfactor
        self.V_p2[0] = mr**(ve)*47.0*vfactor
        self.V_p3[0] = mr**(ve)*70.0*vfactor
        self.V_l1[0] = mr**(ve)*127.0*vfactor
        self.V_l2[0] = mr**(ve)*105.0*vfactor
        self.p_lv[0] = E_lv(self.phi_val, init_temp)*(self.V_lv[0] - V_dlv)
        self.p_rv[0] = E_rv(self.phi_val, init_temp)*(self.V_rv[0] - V_drv)
        self.p_a1[0] = (self.V_a1[0] - V_una1)/C_a1
        self.p_a2[0] = (self.V_a2[0] - V_una2)/C_a2
        self.p_a3A[0] = (self.V_a3A[0] - V_una3A)/C_a3A
        self.p_a3B[0] = (self.V_a3B[0] - V_una3B)/C_a3B
        self.p_v1A[0] = (self.V_v1A[0] - V_unv1A)/C_v1A
        self.p_v1B[0] = (self.V_v1B[0] - V_unv1B)/C_v1B
        self.p_v2[0] = (self.V_v2[0] - V_unv2)/C_v2
        self.p_p1[0] = (self.V_p1[0] - V_unp1)/C_p1
        self.p_p2[0] = (self.V_p2[0] - V_unp2)/C_p2
        self.p_p3[0] = (self.V_p3[0] - V_unp3)/C_p3
        self.p_l1[0] = (self.V_l1[0] - V_unl1)/C_l1
        self.p_l2[0] = (self.V_l2[0] - V_unl2)/C_l2
        self.p_ra[0] = E_ra*(self.V_ra[0] - V_dra)
        self.p_la[0] = E_la*(self.V_la[0] - V_dla)
        self.p_as[0] = R_0s*self.Q_lv[0] + self.p_a1[0]
        self.p_ap[0] = R_0p*self.Q_rv[0] + self.p_p1[0]
        self.lvb[0] = mr**(ve)*2.0
        self.lab[0] = mr**(ve)*2.0
        self.rab[0] = mr**(ve)*2.0
        self.rvb[0] = mr**(ve)*2.0
        self.Q_a1[0] = (self.p_a1[0] - self.p_a2[0])/R_a1
        self.Q_a2A[0] = (self.p_a2[0] - self.p_a3A[0])/R_a2
        self.Q_a2B[0] = (self.p_a2[0] - self.p_a3B[0])/R_a2
        self.Q_a3A[0] = (self.p_a3A[0] - self.p_v1A[0])/R_a3A
        self.Q_a3B[0] = (self.p_a3B[0] - self.p_v1B[0])/R_a3B
        self.Q_v1A[0] = (self.p_v1A[0] - self.p_v2[0])/R_v1A
```

```
        self.Q_v1B[0] = (self.p_v1B[0] - self.p_v2[0])/R_v1B
        self.Q_v2[0] = (self.p_v2[0] - self.p_ra[0])/R_v2
        self.Q_l1[0] = (self.p_l1[0] - self.p_l2[0])/R_l1
        self.Q_l2[0] = (self.p_l2[0] - self.p_la[0])/R_l2
        self.Q_p1[0] = (self.p_p1[0] - self.p_p2[0])/R_p1
        self.Q_p2[0] = (self.p_p2[0] - self.p_p3[0])/R_p2
        self.Q_p3[0] = (self.p_p3[0] - self.p_la[0])/R_p3
        self.O_a1[0] = init_O
        self.O_a2[0] = init_O
        self.O_a3A[0] = init_O
        self.O_a3B[0] = init_O
        self.O_l1[0] = init_O
        self.O_l2[0] = init_O
        self.O_la[0] = init_O
        self.O_lv[0] = init_O
        self.O_p1[0] = init_O - Oa_v_diff
        self.O_p2[0] = init_O - Oa_v_diff
        self.O_p3[0] = init_O - Oa_v_diff
        self.O_ra[0] = init_O - Oa_v_diff
        self.O_rv[0] = init_O - Oa_v_diff
        self.O_v1A[0] = init_O - Oa_v_diff
        self.O_v1B[0] = init_O- Oa_v_diff
        self.O_v2[0] = init_O - Oa_v_diff
        self.T_a1[0] = init_temp
        self.T_a2[0] = init_temp
        self.T_a3A[0] = init_temp
        self.T_a3B[0] = init_temp
        self.T_l1[0] = init_temp
        self.T_l2[0] = init_temp
        self.T_la[0] = init_temp
        self.T_lv[0] = init_temp
        self.T_p1[0] = init_temp
        self.T_p2[0] = init_temp
        self.T_p3[0] = init_temp
        self.T_ra[0] = init_temp
        self.T_rv[0] = init_temp
        self.T_v1A[0] = init_temp
        self.T_v1B[0] = init_temp
        self.T_v2[0] = init_temp
        self.T_tisA[0] = init_temp
        self.T_tisB[0] = init_temp
        self.mrateA[0] = init_mrateA
        self.mrateB[0] = init_mrateB

    def testfree(self):
        sage_free(self.Q_a1)
        sage_free(self.Q_a2A)
        sage_free(self.Q_a2B)
        sage_free(self.Q_a3A)
        sage_free(self.Q_a3B)
        sage_free(self.Q_l1)
```

```
        sage_free(self.Q_l2)
        sage_free(self.Q_la)
        sage_free(self.Q_lv)
        sage_free(self.Q_p1)
        sage_free(self.Q_p2)
        sage_free(self.Q_p3)
        sage_free(self.Q_ra)
        sage_free(self.Q_rv)
        sage_free(self.Q_v1A)
        sage_free(self.Q_v1B)
        sage_free(self.Q_v2)
        sage_free(self.V_a1)
        sage_free(self.V_a2)
        sage_free(self.V_a3A)
        sage_free(self.V_a3B)
        sage_free(self.V_l1)
        sage_free(self.V_l2)
        sage_free(self.V_la)
        sage_free(self.V_lv)
        sage_free(self.V_p1)
        sage_free(self.V_p2)
        sage_free(self.V_p3)
        sage_free(self.V_ra)
        sage_free(self.V_rv)
        sage_free(self.V_v1A)
        sage_free(self.V_v1B)
        sage_free(self.V_v2)
        sage_free(self.p_a1)
        sage_free(self.p_a2)
        sage_free(self.p_a3A)
        sage_free(self.p_a3B)
        sage_free(self.p_ap)
        sage_free(self.p_as)
        sage_free(self.p_l1)
        sage_free(self.p_l2)
        sage_free(self.p_la)
        sage_free(self.p_lv)
        sage_free(self.p_p1)
        sage_free(self.p_p2)
        sage_free(self.p_p3)
        sage_free(self.p_ra)
        sage_free(self.p_rv)
        sage_free(self.p_v1A)
        sage_free(self.p_v1B)
        sage_free(self.p_v2)
        sage_free(self.lvb)
        sage_free(self.lab)
        sage_free(self.rab)
        sage_free(self.rvb)
        sage_free(self.t)
        sage_free(self.T_a1)
```

```
        sage_free(self.O_a1)
        sage_free(self.T_a2)
        sage_free(self.O_a2)
        sage_free(self.T_a3A)
        sage_free(self.O_a3A)
        sage_free(self.T_a3B)
        sage_free(self.O_a3B)
        sage_free(self.T_l1)
        sage_free(self.O_l1)
        sage_free(self.T_l2)
        sage_free(self.O_l2)
        sage_free(self.T_la)
        sage_free(self.O_la)
        sage_free(self.T_lv)
        sage_free(self.O_lv)
        sage_free(self.T_p1)
        sage_free(self.O_p1)
        sage_free(self.T_p2)
        sage_free(self.O_p2)
        sage_free(self.T_p3)
        sage_free(self.O_p3)
        sage_free(self.T_ra)
        sage_free(self.O_ra)
        sage_free(self.T_rv)
        sage_free(self.O_rv)
        sage_free(self.T_v1A)
        sage_free(self.O_v1A)
        sage_free(self.T_v1B)
        sage_free(self.O_v1B)
        sage_free(self.T_v2)
        sage_free(self.O_v2)
        sage_free(self.T_tisA)
        sage_free(self.T_tisB)
        sage_free(self.mrateA)
        sage_free(self.mrateB)

    def fullreturn(self, skip = 1):
        x = pyCardio()
        x.firing = self.firing
        x.firing_tth = self.firing_tth
        x.last_fired = self.last_fired
        x.firing_begin = self.firing_begin
        x.t = [self.t[i] for i in range(0,self.n,skip)]
        x.p_lv = [self.p_lv[i] for i in range(0,self.n,skip)]
        x.p_la = [self.p_la[i] for i in range(0,self.n,skip)]
        x.p_rv = [self.p_lv[i] for i in range(0,self.n,skip)]
        x.p_ra = [self.p_la[i] for i in range(0,self.n,skip)]
        x.p_as = [self.p_as[i] for i in range(0,self.n,skip)]
        x.p_ap = [self.p_as[i] for i in range(0,self.n,skip)]
        x.p_a1 = [self.p_a1[i] for i in range(0,self.n,skip)]
        x.p_a2 = [self.p_a2[i] for i in range(0,self.n,skip)]
```

```
        x.p_a3A = [self.p_a3A[i] for i in range(0,self.n,skip)]
        x.p_a3B = [self.p_a3B[i] for i in range(0,self.n,skip)]
        x.p_v1A = [self.p_v1A[i] for i in range(0,self.n,skip)]
        x.p_v1B = [self.p_v1B[i] for i in range(0,self.n,skip)]
        x.p_v2 = [self.p_v2[i] for i in range(0,self.n,skip)]
        x.p_l2 = [self.p_l2[i] for i in range(0,self.n,skip)]
        x.p_l1 = [self.p_l1[i] for i in range(0,self.n,skip)]
        x.p_p1 = [self.p_p1[i] for i in range(0,self.n,skip)]
        x.p_p2 = [self.p_p2[i] for i in range(0,self.n,skip)]
        x.p_p3 = [self.p_p3[i] for i in range(0,self.n,skip)]
        x.p_v2 = [self.p_v2[i] for i in range(0,self.n,skip)]
        x.V_lv = [self.V_lv[i] for i in range(0,self.n,skip)]
        x.V_rv = [self.V_rv[i] for i in range(0,self.n,skip)]
        x.V_la = [self.V_la[i] for i in range(0,self.n,skip)]
        x.V_ra = [self.V_ra[i] for i in range(0,self.n,skip)]
        x.V_a1 = [self.V_a1[i] for i in range(0,self.n,skip)]
        x.V_a2 = [self.V_a2[i] for i in range(0,self.n,skip)]
        x.V_a3A = [self.V_a3A[i] for i in range(0,self.n,skip)]
        x.V_a3B = [self.V_a3B[i] for i in range(0,self.n,skip)]
        x.V_v1A = [self.V_v1A[i] for i in range(0,self.n,skip)]
        x.V_v1B = [self.V_v1B[i] for i in range(0,self.n,skip)]
        x.V_v2 = [self.V_v2[i] for i in range(0,self.n,skip)]
        x.V_l2 = [self.V_l2[i] for i in range(0,self.n,skip)]
        x.V_l1 = [self.V_l1[i] for i in range(0,self.n,skip)]
        x.V_p1 = [self.V_p1[i] for i in range(0,self.n,skip)]
        x.V_p2 = [self.V_p2[i] for i in range(0,self.n,skip)]
        x.V_p3 = [self.V_p3[i] for i in range(0,self.n,skip)]
        x.Q_lv = [self.Q_lv[i] for i in range(0,self.n,skip)]
        x.Q_la = [self.Q_la[i] for i in range(0,self.n,skip)]
        x.Q_rv = [self.Q_lv[i] for i in range(0,self.n,skip)]
        x.Q_a1 = [self.Q_a1[i] for i in range(0,self.n,skip)]
        x.Q_a2A = [self.Q_a2A[i] for i in range(0,self.n,skip)]
        x.Q_a2B = [self.Q_a2B[i] for i in range(0,self.n,skip)]
        x.Q_a3A = [self.Q_a3A[i] for i in range(0,self.n,skip)]
        x.Q_a3B = [self.Q_a3B[i] for i in range(0,self.n,skip)]
        x.Q_v1A = [self.Q_v1A[i] for i in range(0,self.n,skip)]
        x.Q_v1B = [self.Q_v1B[i] for i in range(0,self.n,skip)]
        x.Q_v2 = [self.Q_v2[i] for i in range(0,self.n,skip)]
        x.Q_l2 = [self.Q_l2[i] for i in range(0,self.n,skip)]
        x.Q_l1 = [self.Q_l1[i] for i in range(0,self.n,skip)]
        x.Q_p1 = [self.Q_p1[i] for i in range(0,self.n,skip)]
        x.Q_p2 = [self.Q_p2[i] for i in range(0,self.n,skip)]
        x.Q_p3 = [self.Q_p3[i] for i in range(0,self.n,skip)]
        x.Q_v2 = [self.Q_v2[i] for i in range(0,self.n,skip)]
        x.T_a1 = [self.T_a1[i] for i in range(0,self.n,skip)]
        x.O_a1 = [self.O_a1[i] for i in range(0,self.n,skip)]
        x.T_a2 = [self.T_a2[i] for i in range(0,self.n,skip)]
        x.O_a2 = [self.O_a2[i] for i in range(0,self.n,skip)]
        x.T_a3A = [self.T_a3A[i] for i in range(0,self.n,skip)]
        x.O_a3A = [self.O_a3A[i] for i in range(0,self.n,skip)]
        x.T_a3B = [self.T_a3B[i] for i in range(0,self.n,skip)]
```

```
        x.O_a3B = [self.O_a3B[i] for i in range(0,self.n,skip)]
        x.T_l1 = [self.T_l1[i] for i in range(0,self.n,skip)]
        x.O_l1 = [self.O_l1[i] for i in range(0,self.n,skip)]
        x.T_l2 = [self.T_l2[i] for i in range(0,self.n,skip)]
        x.O_l2 = [self.O_l2[i] for i in range(0,self.n,skip)]
        x.T_la = [self.T_la[i] for i in range(0,self.n,skip)]
        x.O_la = [self.O_la[i] for i in range(0,self.n,skip)]
        x.T_lv = [self.T_lv[i] for i in range(0,self.n,skip)]
        x.O_lv = [self.O_lv[i] for i in range(0,self.n,skip)]
        x.T_p1 = [self.T_p1[i] for i in range(0,self.n,skip)]
        x.O_p1 = [self.O_p1[i] for i in range(0,self.n,skip)]
        x.T_p2 = [self.T_p2[i] for i in range(0,self.n,skip)]
        x.O_p2 = [self.O_p2[i] for i in range(0,self.n,skip)]
        x.T_p3 = [self.T_p3[i] for i in range(0,self.n,skip)]
        x.O_p3 = [self.O_p3[i] for i in range(0,self.n,skip)]
        x.T_ra = [self.T_ra[i] for i in range(0,self.n,skip)]
        x.O_ra = [self.O_ra[i] for i in range(0,self.n,skip)]
        x.T_rv = [self.T_rv[i] for i in range(0,self.n,skip)]
        x.O_rv = [self.O_rv[i] for i in range(0,self.n,skip)]
        x.T_v1A = [self.T_v1A[i] for i in range(0,self.n,skip)]
        x.O_v1A = [self.O_v1A[i] for i in range(0,self.n,skip)]
        x.T_v1B = [self.T_v1B[i] for i in range(0,self.n,skip)]
        x.O_v1B = [self.O_v1B[i] for i in range(0,self.n,skip)]
        x.T_v2 = [self.T_v2[i] for i in range(0,self.n,skip)]
        x.O_v2 = [self.O_v2[i] for i in range(0,self.n,skip)]
        x.T_tisA = [self.T_tisA[i] for i in range(0,self.n,skip)]
        x.T_tisB = [self.T_tisB[i] for i in range(0,self.n,skip)]
        x.mrateA = [self.mrateA[i] for i in range(0,self.n,skip)]
        x.mrateB = [self.mrateB[i] for i in range(0,self.n,skip)]
        return x

    def init_from_data(self, object pyCardiodata):
        self.firing = pyCardiodata.firing
        self.last_fired = pyCardiodata.last_fired
        self.firing_begin = pyCardiodata.firing_begin
        self.firing_tth = pyCardiodata.firing_tth
        self.t[0] = pyCardiodata.t[-1]
        self.V_lv[0] = pyCardiodata.V_lv[-1]
        self.V_rv[0] = pyCardiodata.V_rv[-1]
        self.V_la[0] = pyCardiodata.V_la[-1]
        self.V_ra[0] = pyCardiodata.V_ra[-1]
        self.V_a1[0] = pyCardiodata.V_a1[-1]
        self.V_a2[0] = pyCardiodata.V_a2[-1]
        self.V_a3A[0] = pyCardiodata.V_a3A[-1]
        self.V_a3B[0] = pyCardiodata.V_a3B[-1]
        self.V_v1A[0] = pyCardiodata.V_v1A[-1]
        self.V_v1B[0] = pyCardiodata.V_v1B[-1]
        self.V_v2[0] = pyCardiodata.V_v2[-1]
        self.V_p1[0] = pyCardiodata.V_p1[-1]
        self.V_p2[0] = pyCardiodata.V_p2[-1]
        self.V_p3[0] = pyCardiodata.V_p3[-1]
```

```
        self.V_l1[0] = pyCardiodata.V_l1[-1]
        self.V_l2[0] = pyCardiodata.V_l2[-1]
        self.Q_a1[0] = pyCardiodata.Q_a1[-1]
        self.Q_a2A[0] = pyCardiodata.Q_a2A[-1]
        self.Q_a2B[0] = pyCardiodata.Q_a2B[-1]
        self.Q_a3A[0] = pyCardiodata.Q_a3A[-1]
        self.Q_a3B[0] = pyCardiodata.Q_a3B[-1]
        self.Q_l1[0] = pyCardiodata.Q_l1[-1]
        self.Q_l2[0] = pyCardiodata.Q_l2[-1]
        self.Q_la[0] = pyCardiodata.Q_la[-1]
        self.Q_lv[0] = pyCardiodata.Q_lv[-1]
        self.Q_p1[0] = pyCardiodata.Q_p1[-1]
        self.Q_p2[0] = pyCardiodata.Q_p2[-1]
        self.Q_p3[0] = pyCardiodata.Q_p3[-1]
        self.Q_rv[0] = pyCardiodata.Q_rv[-1]
        self.Q_v1A[0] = pyCardiodata.Q_v1A[-1]
        self.Q_v1B[0] = pyCardiodata.Q_v1B[-1]
        self.Q_v2[0] = pyCardiodata.Q_v2[-1]
        self.p_a1[0] = pyCardiodata.p_a1[-1]
        self.p_a2[0] = pyCardiodata.p_a2[-1]
        self.p_a3A[0] = pyCardiodata.p_a3A[-1]
        self.p_a3B[0] = pyCardiodata.p_a3B[-1]
        self.p_as[0] = pyCardiodata.p_as[-1]
        self.p_l1[0] = pyCardiodata.p_l1[-1]
        self.p_l2[0] = pyCardiodata.p_l2[-1]
        self.p_la[0] = pyCardiodata.p_la[-1]
        self.p_lv[0] = pyCardiodata.p_lv[-1]
        self.p_p1[0] = pyCardiodata.p_p1[-1]
        self.p_p2[0] = pyCardiodata.p_p2[-1]
        self.p_p3[0] = pyCardiodata.p_p3[-1]
        self.p_ra[0] = pyCardiodata.p_ra[-1]
        self.p_rv[0] = pyCardiodata.p_rv[-1]
        self.p_v1A[0] = pyCardiodata.p_v1A[-1]
        self.p_v1B[0] = pyCardiodata.p_v1B[-1]
        self.p_v2[0] = pyCardiodata.p_v2[-1]
        self.T_a1[0] = pyCardiodata.T_a1[-1]
        self.O_a1[0] = pyCardiodata.O_a1[-1]
        self.T_a2[0] = pyCardiodata.T_a2[-1]
        self.O_a2[0] = pyCardiodata.O_a2[-1]
        self.T_a3A[0] = pyCardiodata.T_a3A[-1]
        self.O_a3A[0] = pyCardiodata.O_a3A[-1]
        self.T_a3B[0] = pyCardiodata.T_a3B[-1]
        self.O_a3B[0] = pyCardiodata.O_a3B[-1]
        self.T_l1[0] = pyCardiodata.T_l1[-1]
        self.O_l1[0] = pyCardiodata.O_l1[-1]
        self.T_l2[0] = pyCardiodata.T_l2[-1]
        self.O_l2[0] = pyCardiodata.O_l2[-1]
        self.T_la[0] = pyCardiodata.T_la[-1]
        self.O_la[0] = pyCardiodata.O_la[-1]
        self.T_lv[0] = pyCardiodata.T_lv[-1]
        self.O_lv[0] = pyCardiodata.O_lv[-1]
```

```
    self.T_p1[0] = pyCardiodata.T_p1[-1]
    self.O_p1[0] = pyCardiodata.O_p1[-1]
    self.T_p2[0] = pyCardiodata.T_p2[-1]
    self.O_p2[0] = pyCardiodata.O_p2[-1]
    self.T_p3[0] = pyCardiodata.T_p3[-1]
    self.O_p3[0] = pyCardiodata.O_p3[-1]
    self.T_ra[0] = pyCardiodata.T_ra[-1]
    self.O_ra[0] = pyCardiodata.O_ra[-1]
    self.T_rv[0] = pyCardiodata.T_rv[-1]
    self.O_rv[0] = pyCardiodata.O_rv[-1]
    self.T_v1A[0] = pyCardiodata.T_v1A[-1]
    self.O_v1A[0] = pyCardiodata.O_v1A[-1]
    self.T_v1B[0] = pyCardiodata.T_v1B[-1]
    self.O_v1B[0] = pyCardiodata.O_v1B[-1]
    self.T_v2[0] = pyCardiodata.T_v2[-1]
    self.O_v2[0] = pyCardiodata.O_v2[-1]
    self.T_tisA[0] = pyCardiodata.T_tisA[-1]
    self.T_tisB[0] = pyCardiodata.T_tisB[-1]
    self.mrateA[0] = pyCardiodata.mrateA[-1]
    self.mrateB[0] = pyCardiodata.mrateB[-1]

def init_320(self):
    self.Q_a1[0] = 1.1025135277
    self.Q_a2A[0] = 0.544393646528
    self.Q_a2B[0] = 0.544393646528
    self.Q_a3A[0] = 0.546496248379
    self.Q_a3B[0] = 0.546496248379
    self.Q_l1[0] = 1.09503991089
    self.Q_l2[0] = 1.09174641099
    self.Q_la[0] = 1.02838836106
    self.Q_lv[0] = 1.13136022214
    self.Q_p1[0] = 1.10088203345
    self.Q_p2[0] = 1.08903014716
    self.Q_p3[0] = 1.08998006346
    self.Q_rv[0] = 1.13136022214
    self.Q_v1A[0] = 0.546854607501
    self.Q_v1B[0] = 0.546854607501
    self.Q_v2[0] = 1.09320715572
    self.V_a1[0] = 0.954408993409
    self.V_a2[0] = 1.68729489617
    self.V_a3A[0] = 1.01577979704
    self.V_a3B[0] = 1.01577979704
    self.V_l1[0] = 0.551282989383
    self.V_l2[0] = 0.37388563656
    self.V_la[0] = 0.276360350204
    self.V_lv[0] = 0.281749312189
    self.V_p1[0] = 0.477010644607
    self.V_p2[0] = 0.278635772594
    self.V_p3[0] = 0.315471931912
    self.V_ra[0] = 0.415479990284
    self.V_rv[0] = 0.539866913953
```

```
        self.V_v1A[0] = 1.62881849784
        self.V_v1B[0] = 1.62881849784
        self.V_v2[0] = 9.65328659847
        self.p_a1[0] = 94.4252857015
        self.p_a2[0] = 74.4678532289
        self.p_a3A[0] = 53.1873861959
        self.p_a3B[0] = 53.1873861959
        self.p_as[0] = 102.722512635
        self.p_l1[0] = 12.8699399516
        self.p_l2[0] = 6.80997645705
        self.p_la[0] = 3.79524421819
        self.p_lv[0] = 54.5162378124
        self.p_p1[0] = 40.1112576308
        self.p_p2[0] = 34.6174354401
        self.p_p3[0] = 21.9418870177
        self.p_ra[0] = 3.79524421819
        self.p_rv[0] = 54.5162378124
        self.p_v1A[0] = 13.2229661873
        self.p_v1B[0] = 13.2229661873
        self.p_v2[0] = 11.8779359748

cpdef EulerStepInv(Cardio x, int index, double dt, double tth, double int_temp, double amb_temp = 37.0,
    """
    Takes ambient and internal temperature and heartrate as inputs, determines necessary metabolic rate
    """
    cdef:
        double Qincr
        double V_la_incr
        double V_lv_incr
        double V_rv_incr
        double V_ra_incr
        double V_p3_incr
        double V_p2_incr
        double V_p1_incr
        double V_a3A_incr
        double V_a3B_incr
        double V_a2_incr
        double V_a1_incr
        double V_l2_incr
        double V_l1_incr
        double V_v1A_incr
        double V_v1B_incr
        double V_v2_incr
        double R_a3fA
        double R_a3f

    R_a3f = r3affun(x.T_a3A[index])
    R_a3fA = 1.0 + (R_a3f-1)*.4
    if R_a3fA < 1: R_a3fA = 1.0

    x.t[index+1] = x.t[index] + dt
```

```
    if x.firing == 0:
        if x.t[index] - x.last_fired > tth - t_ce_t(tth, x.T_lv[index]):
            x.firing = 1
            x.firing_tth = tth
            x.firing_begin = x.t[index]
    if x.firing == 1:
        if x.t[index] - x.last_fired > x.firing_tth:
            x.firing = 0
            x.last_fired = x.t[index]
            x.phi_val = 0
            x.lvb[index] = 0
            x.rvb[index] = 0
            x.lab[index] = 0
            x.rab[index] = 0
        else:
            x.phi_val = phi(x.t[index] - x.firing_begin, x.firing_tth, x.T_lv[index])

    #Left heart chambers:
        #outflow and backflow checks, atrial
    Qincr = dt*(x.p_la[index] - x.p_lv[index] - vis(x.T_la[index])*R_la*x.Q_la[index])/L_la
    if x.Q_la[index]+Qincr >= 0.0:
        x.lab[index+1] = x.lab[index]
        x.Q_la[index+1] = x.Q_la[index] + Qincr
    elif x.lab[index] < V_maxlab:
        #backflow case
        x.lab[index+1] = x.lab[index] - dt*x.Q_la[index]
        x.Q_la[index+1] = x.Q_la[index] + Qincr
    else:
        Qincr = 0
        x.lab[index+1] = x.lab[index]
        x.Q_la[index+1] = 0
        #end of outflow, backflow checks, atrial
    V_la_incr = dt*(x.Q_l2[index] - x.Q_la[index])
    x.T_la[index+1] = (x.V_la[index]*x.T_la[index] + dt*x.Q_l2[index]*x.T_l2[index] \
- dt*x.Q_la[index]*x.T_la[index])/(x.V_la[index] + V_la_incr)
    x.O_la[index+1] = (x.V_la[index]*x.O_la[index] + dt*x.Q_l2[index]*x.O_l2[index] \
- dt*x.Q_la[index]*x.O_la[index])/(x.V_la[index] + V_la_incr)
    x.V_la[index+1] = x.V_la[index] + V_la_incr
    x.p_la[index+1] = E_la*(x.V_la[index] - V_dla)


        #outflow and backflow checks, left ventricular
    Qincr = dt*(x.p_lv[index] - x.p_as[index])/L_lv
    if x.Q_lv[index]+Qincr >= 0.0:
        x.lvb[index+1] = x.lvb[index]
        x.Q_lv[index+1] = x.Q_lv[index] + Qincr
    elif x.lvb[index] < V_maxlvb:
        #backflow case
        x.lvb[index+1] = x.lvb[index] - dt*x.Q_lv[index]
        x.Q_lv[index+1] = x.Q_lv[index] + Qincr
```

```
    else:
        Qincr = 0
        x.lvb[index+1] = x.lvb[index]
        x.Q_lv[index+1] = 0
        #end of outflow, backflow checks
    V_lv_incr = dt*(x.Q_la[index] - x.Q_lv[index])
    x.T_lv[index+1] = (x.V_lv[index]*x.T_lv[index] + dt*x.Q_la[index]*x.T_la[index] \
- dt*x.Q_lv[index]*x.T_lv[index])/(x.V_lv[index] + V_lv_incr)
    x.O_lv[index+1] = (x.V_lv[index]*x.O_lv[index] + dt*x.Q_la[index]*x.O_la[index] \
- dt*x.Q_lv[index]*x.O_lv[index])/(x.V_lv[index] + V_lv_incr)
    x.V_lv[index+1] = x.V_lv[index] + V_lv_incr
    x.p_as[index+1] = vis(x.T_lv[index])*R_0s*x.Q_lv[index] + x.p_a1[index]
    x.p_lv[index+1] = E_lv(x.phi_val, x.T_lv[index])*(x.V_lv[index] - V_dlv)

    #Right heart chambers:
    Qincr = dt*(x.p_ra[index] - x.p_rv[index] - vis(x.T_ra[index])*R_ra*x.Q_ra[index])/L_ra
    if x.Q_ra[index]+Qincr >= 0.0:
        if x.Q_ra[index] + Qincr < 1000:
            x.rab[index+1] = x.rab[index]
            x.Q_ra[index+1] = x.Q_ra[index] + Qincr
        else:
            x.Q_ra[index+1] = 1000
    elif x.rab[index] < V_maxrab:
        #backflow case
        x.rab[index+1] = x.rab[index] - dt*x.Q_ra[index]
        x.Q_ra[index+1] = x.Q_ra[index] + Qincr
    else:
        Qincr = 0
        x.rab[index+1] = x.rab[index]
        x.Q_ra[index+1] = 0
        #end of outflow, backflow checks, atrial


    V_ra_incr = dt*(x.Q_v2[index] - x.Q_ra[index])
    x.T_ra[index+1] = (x.V_ra[index]*x.T_ra[index]+ dt*x.Q_v2[index]*x.T_v2[index] \
- dt*x.Q_ra[index]*x.T_ra[index])/(x.V_ra[index] + V_ra_incr)
    x.O_ra[index+1] = (x.V_ra[index]*x.O_ra[index]+ dt*x.Q_v2[index]*x.O_v2[index] \
- dt*x.Q_ra[index]*x.O_ra[index])/(x.V_ra[index] + V_ra_incr)
    x.V_ra[index+1] = x.V_ra[index] + V_ra_incr
    x.p_ra[index+1] = E_ra*(x.V_ra[index] - V_dra)

        # begin outflow, backflow
    Qincr = dt*(x.p_rv[index] - x.p_ap[index])/L_rv
    if x.Q_rv[index]+Qincr >= 0.0:
        x.rvb[index+1] = x.rvb[index]
        x.Q_rv[index+1] = x.Q_rv[index] + Qincr
    elif x.rvb[index] < V_maxrvb:
        x.rvb[index+1] = x.rvb[index] - dt*x.Q_rv[index]
        x.Q_rv[index+1] = x.Q_rv[index] + Qincr
    else:
        Qincr = 0
```

```
        x.rvb[index+1] = x.rvb[index]
        x.Q_rv[index+1] = 0
        # end backflow check
    x.V_rv[index+1] = x.V_rv[index] + dt*(x.Q_ra[index] - x.Q_rv[index])
    x.T_rv[index+1] = (x.V_rv[index]*x.T_rv[index]+ dt*x.Q_ra[index]*x.T_ra[index] \
- dt*x.Q_rv[index]*x.T_rv[index])/(x.V_rv[index] + dt*(x.Q_ra[index] - x.Q_rv[index]))
    x.O_rv[index+1] = (x.V_rv[index]*x.O_rv[index]+ dt*x.Q_ra[index]*x.O_ra[index] \
- dt*x.Q_rv[index]*x.O_rv[index])/(x.V_rv[index] + dt*(x.Q_ra[index] - x.Q_rv[index]))
    x.p_ap[index+1] = vis(x.T_rv[index])*R_0p*x.Q_rv[index] + x.p_p1[index]
    x.p_rv[index+1] = E_rv(x.phi_val, x.T_lv[index])*(x.V_rv[index] - V_drv)

    #Systemic arteries
    Qincr = dt*(x.p_a1[index] - x.p_a2[index] - vis(x.T_a1[index])*R_a1*x.Q_a1[index])/L_a1
    x.Q_a1[index+1] = x.Q_a1[index] + Qincr
    V_a1_incr = dt*(x.Q_lv[index] - x.Q_a1[index])
    x.T_a1[index+1] = (x.V_a1[index]*x.T_a1[index]+ dt*x.Q_lv[index]*x.T_lv[index] \
- dt*x.Q_a1[index]*x.T_a1[index])/(x.V_a1[index] + V_a1_incr)
    x.O_a1[index+1] = (x.V_a1[index]*x.O_a1[index]+ dt*x.Q_lv[index]*x.O_lv[index] \
- dt*x.Q_a1[index]*x.O_a1[index])/(x.V_a1[index] + V_a1_incr)
    x.V_a1[index+1] = x.V_a1[index] + V_a1_incr
    x.p_a1[index+1] = (x.V_a1[index] - V_una1)/C_a1
    #x.map[index+1] = x.map[index]*.99 + x.p_a1[index]*.01

    x.Q_a2A[index+1] = (x.p_a2[index] - x.p_a3A[index])/(vis(x.T_a2[index])*R_a2)
    x.Q_a2B[index+1] = (x.p_a2[index] - x.p_a3B[index])/(vis(x.T_a2[index])*R_a2)
    V_a2_incr = dt*(x.Q_a1[index] - x.Q_a2A[index]- x.Q_a2B[index])
    x.T_a2[index+1] = (x.V_a2[index]*x.T_a2[index]+ dt*x.Q_a1[index]*x.T_a1[index] \
- dt*x.Q_a2A[index]*x.T_a2[index] - dt*x.Q_a2B[index]*x.T_a2[index])/(x.V_a2[index] + V_a2_incr)
    x.O_a2[index+1] = (x.V_a2[index]*x.O_a2[index]+ dt*x.Q_a1[index]*x.O_a1[index] \
- dt*x.Q_a2A[index]*x.O_a2[index] - dt*x.Q_a2B[index]*x.O_a2[index])/(x.V_a2[index] + V_a2_incr)
    x.V_a2[index+1] = x.V_a2[index] + V_a2_incr
    x.p_a2[index+1] = (x.V_a2[index] - V_una2)/C_a2

    x.Q_a3A[index+1] = (x.p_a3A[index] - x.p_v1A[index])/(vis(x.T_a3A[index])*R_a3A*R_a3fA**4)
    V_a3A_incr = dt*(x.Q_a2A[index] - x.Q_a3A[index])
    # temperature exchange:
    x.T_a3A[index+1] = (x.V_a3A[index]*x.T_a3A[index]+ dt*x.Q_a2A[index]*x.T_a2[index] \
- dt*x.Q_a3A[index]*x.T_a3A[index] + dt*k_tisA*(x.T_tisA[index] - x.T_a3A[index]))\
/(x.V_a3A[index] + V_a3A_incr)
    x.O_a3A[index+1] = (x.V_a3A[index]*x.O_a3A[index]+ dt*x.Q_a2A[index]*x.O_a2[index] \
- dt*x.Q_a3A[index]*x.O_a3A[index] - dt*x.mrateA[index])/(x.V_a3A[index] + V_a3A_incr)
    x.V_a3A[index+1] = x.V_a3A[index] + V_a3A_incr
    x.p_a3A[index+1] = (x.V_a3A[index] - V_una3A/R_a3fA)/(C_a3A/R_a3fA)

    x.Q_a3B[index+1] = (x.p_a3B[index] - x.p_v1B[index])/(vis(x.T_a3B[index])*R_a3B*R_a3f**4)
    V_a3B_incr = dt*(x.Q_a2B[index] - x.Q_a3B[index])
    # temperature exchange:
    x.T_a3B[index+1] = (x.V_a3B[index]*x.T_a3B[index]+ dt*x.Q_a2B[index]*x.T_a2[index] \
- dt*x.Q_a3B[index]*x.T_a3B[index] + dt*k_tisB*(x.T_tisB[index] -  x.T_a3B[index]))\
/(x.V_a3B[index] + V_a3B_incr)
    x.O_a3B[index+1] = (x.V_a3B[index]*x.O_a3B[index]+ dt*x.Q_a2B[index]*x.O_a2[index] \
```

```
- dt*x.Q_a3B[index]*x.O_a3B[index] - dt*x.mrateB[index])/(x.V_a3B[index] + V_a3B_incr)
    x.V_a3B[index+1] = x.V_a3B[index] + V_a3B_incr
    x.p_a3B[index+1] = (x.V_a3B[index] - V_una3B/R_a3f)/(C_a3B/R_a3f)

    # heat exchange with tissue, metabolic rate determination =
    x.T_tisA[index+1] = int_temp
    x.mrateA[index+1] = ((int_temp - x.T_tisA[index])*tismassA/dt \
+ k_ambA*(-amb_temp + x.T_tisA[index]) + k_tisA*(-x.T_a3A[index] + x.T_tisA[index]))/k_mrt

    if down > 0:  # Only used for torpor entrance, not arousal
        x.mrateB[index+1] = ((int_temp - x.T_tisB[index])*tismassB/dt \
+ k_ambB*(-amb_temp + x.T_tisB[index]) + k_tisA*(-x.T_a3B[index] + x.T_tisB[index]))/k_mrt
        x.T_tisB[index+1] = int_temp
    else:
        x.mrateB[index+1] = k_ambB*(-amb_temp + x.T_a3B[index])/k_mrt
        x.T_tisB[index+1] = x.T_tisB[index] + dt*(k_ambB*(amb_temp - x.T_tisB[index]) \
+ k_tisB*(x.T_a3B[index] - x.T_tisB[index]) + x.mrateB[index]*k_mrt)/tismassB

    #prevents even transitory negative metabolic rates:
    if x.mrateB[index+1] < 0:
        x.mrateB[index+1] = 0.0
    if x.mrateA[index+1] < 0:
        x.mrateA[index+1] = 0.0

    #Systemic veins
    x.Q_v1A[index+1] = (x.p_v1A[index] - x.p_v2[index])/(vis(x.T_v1A[index])*R_v1A)
    V_v1A_incr = dt*(x.Q_a3A[index] - x.Q_v1A[index])
    x.T_v1A[index+1] = (x.V_v1A[index]*x.T_v1A[index]+ dt*x.Q_a3A[index]*x.T_a3A[index] \
- dt*x.Q_v1A[index]*x.T_v1A[index])/(x.V_v1A[index] + V_v1A_incr)
    x.O_v1A[index+1] = (x.V_v1A[index]*x.O_v1A[index]+ dt*x.Q_a3A[index]*x.O_a3A[index] \
- dt*x.Q_v1A[index]*x.O_v1A[index])/(x.V_v1A[index] + V_v1A_incr)
    x.V_v1A[index+1] = x.V_v1A[index] + V_v1A_incr
    x.p_v1A[index+1] = (x.V_v1A[index] - V_unv1A)/C_v1A

    x.Q_v1B[index+1] = (x.p_v1B[index] - x.p_v2[index])/(vis(x.T_v1B[index])*R_v1B)
    V_v1B_incr = dt*(x.Q_a3B[index] - x.Q_v1B[index])
    x.T_v1B[index+1] = (x.V_v1B[index]*x.T_v1B[index]+ dt*x.Q_a3B[index]*x.T_a3B[index] \
- dt*x.Q_v1B[index]*x.T_v1B[index])/(x.V_v1B[index] + V_v1B_incr)
    x.O_v1B[index+1] = (x.V_v1B[index]*x.O_v1B[index]+ dt*x.Q_a3B[index]*x.O_a3B[index] \
- dt*x.Q_v1B[index]*x.O_v1B[index])/(x.V_v1B[index] + V_v1B_incr)
    x.V_v1B[index+1] = x.V_v1B[index] + V_v1B_incr
    x.p_v1B[index+1] = (x.V_v1B[index] - V_unv1B)/C_v1B

    Qincr = dt*(x.p_v2[index] - x.p_ra[index] - vis(x.T_v2[index])*R_v2*x.Q_v2[index])/L_v2
    x.Q_v2[index+1] = x.Q_v2[index] + Qincr
    V_v2_incr = dt*(x.Q_v1A[index] + x.Q_v1B[index] - x.Q_v2[index])
    x.T_v2[index+1] = (x.V_v2[index]*x.T_v2[index]+ dt*x.Q_v1A[index]*x.T_v1A[index] \
+ dt*x.Q_v1B[index]*x.T_v1B[index] - dt*x.Q_v2[index]*x.T_v2[index])/(x.V_v2[index] + V_v2_incr)
    x.O_v2[index+1] = (x.V_v2[index]*x.O_v2[index]+ dt*x.Q_v1A[index]*x.O_v1A[index] \
+ dt*x.Q_v1B[index]*x.O_v1B[index] - dt*x.Q_v2[index]*x.O_v2[index])/(x.V_v2[index] + V_v2_incr)
    x.V_v2[index+1] = x.V_v2[index] + V_v2_incr
```

```
    x.p_v2[index+1] = (x.V_v2[index] - V_unv2)/C_v2

    #Pulmonary arteries
    Qincr = dt*(x.p_p1[index] - x.p_p2[index] - vis(x.T_p1[index])*R_p1*x.Q_p1[index])/L_p1
    (x.Q_p1)[index+1] = x.Q_p1[index] + Qincr
    V_p1_incr = dt*(x.Q_rv[index] - x.Q_p1[index])
    x.T_p1[index+1] = (x.V_p1[index]*x.T_p1[index]+ dt*x.Q_rv[index]*x.T_rv[index] \
- dt*x.Q_p1[index]*x.T_p1[index])/(x.V_p1[index] + V_p1_incr)
    x.O_p1[index+1] = (x.V_p1[index]*x.O_p1[index]+ dt*x.Q_rv[index]*x.O_rv[index] \
- dt*x.Q_p1[index]*x.O_p1[index])/(x.V_p1[index] + V_p1_incr)
    x.V_p1[index+1] = x.V_p1[index] + V_p1_incr
    x.p_p1[index+1] = (x.V_p1[index] - V_unp1)/C_p1

    (x.Q_p2)[index+1] = (x.p_p2[index] - x.p_p3[index])/(vis(x.T_p2[index])*R_p2)
    V_p2_incr = dt*(x.Q_p1[index] - x.Q_p2[index])
    x.T_p2[index+1] = (x.V_p2[index]*x.T_p2[index]+ dt*x.Q_p1[index]*x.T_p1[index] \
- dt*x.Q_p2[index]*x.T_p2[index])/(x.V_p2[index] + V_p2_incr)
    x.O_p2[index+1] = (x.V_p2[index]*x.O_p2[index]+ dt*x.Q_p1[index]*x.O_p1[index] \
- dt*x.Q_p2[index]*x.O_p2[index])/(x.V_p2[index] + V_p2_incr)
    x.V_p2[index+1] = x.V_p2[index] + V_p2_incr
    x.p_p2[index+1] = (x.V_p2[index] - V_unp2)/C_p2

    (x.Q_p3)[index+1] = (x.p_p3[index] - x.p_l1[index])/(vis(x.T_p3[index])*R_p3)
    V_p3_incr = dt*(x.Q_p2[index] - x.Q_p3[index])
    x.T_p3[index+1] = (x.V_p3[index]*x.T_p3[index] + dt*x.Q_p2[index]*x.T_p2[index] \
- dt*x.Q_p3[index]*x.T_p3[index])/(x.V_p3[index] + V_p3_incr)
    x.O_p3[index+1] = (x.V_p3[index]*x.O_p3[index] + dt*x.Q_p2[index]*x.O_p2[index] \
- dt*x.Q_p3[index]*x.O_p3[index])/(x.V_p3[index] + V_p3_incr)
    x.V_p3[index+1] = x.V_p3[index] + V_p3_incr
    x.p_p3[index+1] = (x.V_p3[index] - V_unp3)/C_p3

    #Pulmonary veins
    (x.Q_l1)[index+1] = (x.p_l1[index] - x.p_l2[index])/(vis(x.T_l1[index])*R_l1)
    V_l1_incr = dt*(x.Q_p3[index] - x.Q_l1[index])
    x.T_l1[index+1] = (x.V_l1[index]*x.T_l1[index]+ dt*x.Q_p3[index]*x.T_p3[index] \
- dt*x.Q_l1[index]*x.T_l1[index])/(x.V_l1[index] + V_l1_incr)
    x.O_l1[index+1] = O_in
    x.V_l1[index+1] = x.V_l1[index] + V_l1_incr
    x.p_l1[index+1] = (x.V_l1[index] - V_unl1)/C_l1

    Qincr = dt*(x.p_l2[index] - x.p_la[index] - vis(x.T_l2[index])*R_l2*x.Q_l2[index])/L_l2
    x.Q_l2[index+1] = x.Q_l2[index] + Qincr
    V_l2_incr = dt*(x.Q_l1[index] - x.Q_l2[index])
    x.T_l2[index+1] = (x.V_l2[index]*x.T_l2[index]+ dt*x.Q_l1[index]*x.T_l1[index] \
- dt*x.Q_l2[index]*x.T_l2[index])/(x.V_l2[index] + V_l2_incr)
    x.O_l2[index+1] = (x.V_l2[index]*x.O_l2[index]+ dt*x.Q_l1[index]*x.O_l1[index] \
- dt*x.Q_l2[index]*x.O_l2[index])/(x.V_l2[index] + V_l2_incr)
    x.V_l2[index+1] = x.V_l2[index] + V_l2_incr
    x.p_l2[index+1] = (x.V_l2[index] - V_unl2)/C_l2
```