JSR2D程序

加载文件

分别读取gridt.dat、npoch.dat、nod.dat、noe.dat四个文件并将文件内容存入与文件同名的list中，同时初始化NP（顶点个数）、NE（网格个数）两个全局变量。

例如读取gridt.dat文件：

```
    i = 0
    gridt = np.zeros([2, NP], float)
    with open(grid_dir + "/gridt.dat") as file_obj:
        lines = file_obj.readlines()  # 去除空格
        for line in lines:
            line_list = line.split()
            gridt[0, i] = float(line_list[0])  # x坐标
            gridt[1, i] = float(line_list[1])  # y坐标
            i = i + 1  # 数组下标
    file_obj.close()
```

初始化常数

为常量赋值，同时初始化CCAJSR（fn，肌质网每一点钙的浓度，初始值为1.0）、 Gn（gn，初始值1/14）、NEWCCA（存每一次迭代的中间结果fn）、 NEWF（存每一次迭代的中间结果gn）四个数组。

```
    # Jdiffusion扩散项
    DCAJSR = 3.5 * 10 ** 8  # 终池jSR 自由Ca离子扩散系数
    DF = 2 * 10 ** 7  # 荧光指示剂染料 扩散系数

    # Jbuff
    BCSQ = 14.0  # 方程中的B，所有肌钙集蛋白（肌浆网SR上调控钙储存的主要结合蛋白，调节SR钙释放）的量
    KDCSQ = 0.63  # 方程中的K，肌钙集蛋白的Ca离子解离常数

    # Jrefill,Jryr两项
    # 扩散系数
    DCARYR = 0.25 * 6.5 * 10 ** 7  # RyR通道 Ca离子扩散系数 KRyR 6.5 * 10 ** 7
    DCAFSR = 0.7854 * 10 ** 6  # 连接的fSR通道 Ca离子扩散系数 KfSR
    # 浓度
    CCAMYO = 0.0001  # [Ca2+]cyto  胞浆的Ca浓度
    CCAFSR = 1.0  # [Ca2+]fSR  fSR通道中Ca离子浓度

    # Jdye染料
    F = 0.1  # [F]T   fluo-3总浓度
    K1 = 48800  # KF+  Ca离子和fluo-3的结合速率
    K2 = 19520  # KF-  Ca离子和fluo-3的解离速率
    # K1 = 0
    # K2 = 0

    # 关闭RyR通道用到
    DT = 2 * 10 ** -6  # dt
    RELEASE_TIMES = 2 * 10 ** -2  # 0.02s,这个就是ryr通道开放的时间，后80毫秒是恢复的
```

```
    # 点的属性
    B_INNER = 0  # 内部
    B_INFLOW = 2  # 入流
    B_WALL = 1  # 壁面
    B_OUTFLOW = 4  # 出流
    B_SYMMETRY = 8
    B_SOURCE = 16

    CCAJSR = np.ones(NP, float)  # fn,肌质网每一点钙的浓度
    Gn = np.ones(NP, float) / 14.0  # gn初始值 1/14  固定
    NEWCCA = np.empty(NP, float)  # 存每一次迭代的中间结果fn
    NEWF = np.empty(NP, float)  # 存每一次迭代的中间结果gn
```

计算公式过程中需要的变量

计算每个三角形的面积

根据每个三角形三个点的坐标构建行列式，计算三角形面积，同时计算整个区域的总面积。

```
    for j in range(0, 3):  # 根据该三角形每个点的坐标构建行列式
        p = nod[j, i]
        area_matrix[j, 0] = 1.0
        area_matrix[j, 1] = gridt[0, p]
        area_matrix[j, 2] = gridt[1, p]
    area = np.linalg.det(area_matrix) * 0.5  # 计算该三角形面积A
    total_area = total_area + area  # 计算整个区域总面积
```

计算每个点控制面积

对每个点周围的三角形面积以及三角形个数进行累加。

```
    for v in range(0, 3):
        p = nod[v, i]
        control_area[p] = control_area[p] + area  # 累加每个点周围的三角形面积
        cal_max[p] = cal_max[p] + 1  # 累加每个点周围的三角形个数
```

计算每个三角形的属性

存储每个点i作为中心点时的周围的三角形序号及其在该三角形中的序号。存储每个点i在每个三角形中所对边的nix×li、niy×li、bi、ci的值。

```
    for k in range(0, 3):
        i = nod[k, j]
        t = num_temp[i]
        near_triangle[t, i] = j  # 当前中心点周围第t个三角形的序号
        num_in_triangle[t, i] = k  # 当前中心点在周围第t个三角形中的序号
        num_temp[i] = num_temp[i] + 1  # 累加每个点周围的三角形个数
        nod2 = nod[(k + 1) % 3, j]  # 当前三角形中非中心点的两个点坐标
        nod3 = nod[(k + 2) % 3, j]
        nix_multiply_l[k, j] = gridt[1, nod2] - gridt[1, nod3]  # nix*li=y2-y3
        niy_multiply_l[k, j] = gridt[0, nod3] - gridt[0, nod2]  # niy*li=x3-x2
        b_arr[k, j] = (gridt[1, nod3] - gridt[1, nod2]) / (2 * single_area[j])
# bi=(y3-y2)/(2*A)
        c_arr[k, j] = (gridt[0, nod2] - gridt[0, nod3]) / (2 * single_area[j])
# ci=(x2-x3)/(2*A)
```

执行50000步

判断当前执行步数

若为第一步，将数组CCAJSR和Gn中的值及其平均值存入文件中；否则，从文件中读取内容存入数组CCAJSR和Gn中及其平均值变量中。

```
    if CURSTEP == 1:  # 若为第一步
        cal_avg()  # 计算平均值
        fn_file_name = "Fn00000000.dat"
        write_fn = os.path.join(pathfn, fn_file_name)  # 将每个点的Fn及平均值存入文件
        file = open(write_fn, 'w')
        for j in range(0, NP):
            file.write(str(CCAJSR[j]) + '\n')
        file.write(str(fn_avg) + '\n')
        print("数据写入 ", fn_file_name)
        file.flush()
        file.close()
        gn_file_name = "Gn00000000.dat"
        write_gn = os.path.join(pathgn, gn_file_name)  # 将每个点的Gn及平均值存入文件
        file = open(write_gn, 'w')
        for j in range(0, NP):
            file.write(str(Gn[j]) + '\n')
        file.write(str(gn_avg) + '\n')
        print("数据写入 ", gn_file_name)
        file.flush()
        file.close()
    else:  # 不为第一步
        fn_file_name, gn_file_name = generate_file_name(CURSTEP - 1)
        # 根据执行步数生成文件名
        i = 0
        data_dir = "./DATA/DATA_blink"
        with open(data_dir + "/Fn/" + fn_file_name) as file_obj:
        # 从文件Fn中读取内容并存入数组CCAJSR和Fn的平均值fn_avg
            lines = file_obj.readlines()
            for line in lines:
                line_list = line.split()
                if i < NP:
                    CCAJSR[i] = float(line_list[0])
                else:
                    fn_avg = float(line_list[0])
```

```
                i = i + 1
        file_obj.close()
        i = 0
        with open(data_dir + "/Gn/" + gn_file_name) as file_obj:
        # 从文件Gn中读取内容并存入数组Gn和Gn的平均值gn_avg
            lines = file_obj.readlines()
            for line in lines:
                line_list = line.split()
                if i < NP:
                    Gn[i] = float(line_list[0])
                else:
                    gn_avg = float(line_list[0])
                i = i + 1
        file_obj.close()
```

循环执行至50000步

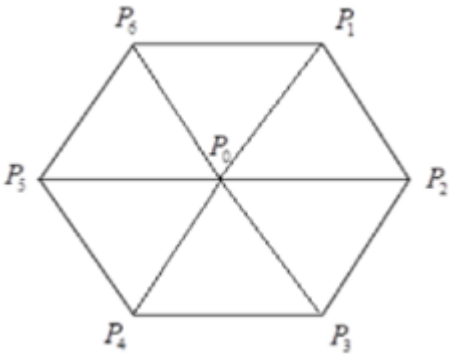
每一步都对所有点的Fn、Gn值迭代计算10次，并将结果存入数组CCAJSR和数组Gn中。

```
    fn(ITERATION)  # ITERATION = 10 迭代次数
    gn(ITERATION)
    for i in range(0, NP):
        CCAJSR[i] = NEWCCA[i]
        Gn[i] = NEWF[i]
```

fn函数

每一步对所有点迭代十次，计算每一个点在F(n+1)时刻的值：

当P0为中心点时：



1. 中心点确定即可计算（2）式

$$\Delta t \cdot [K_F^- \cdot g - K_F^+ \cdot f_n \cdot (F - g)] \cdot V \qquad (2)$$

其中，Δt、KF-、KF+、F都为常数，g、fn取n时刻值，V为当前中心点控制面积。

```
    m2 = DT * (K2 * Gn[p] - K1 * CCAJSR[p] * (F - Gn[p])) * control_area[p]  # 计算（2）式
```

2. 遍历中心点周围的控制单元

① 取出每个三角形单元的编号及中心点在该三角形中的序号：

```
    n = near_triangle[o, p]  # n为三角形编号
    if n == -1:  # 若为-1说明相邻三角形遍历完毕，退出循环
        break
    i = num_in_triangle[o, p]  # i为中心点在该三角形中的序号
```

② 判断该三角形单元另外两个点在该三角形中的序号及是否为边界点（边界点包括入流、出流、壁面壁面边界）：

```
    nod_2, nod_3 = nod[(i + 1) % 3, n], nod[(i + 2) % 3, n]  # 其余两点的编号及在三角形中的序号
    if npoch[nod_2] > B_INNER and npoch[nod_3] > B_INNER:
        continue  # 若为边界，则跳过该三角形
```

③ 计算（3）（5）式：

$$f_n \times \sum_{i=1}^{m}\left[1+\frac{B\times K}{(K+f_{ni})^2}\right]V_i \qquad (3)$$

$$\sum_{i=1}^{m}\left[1+\frac{B\times K}{(K+f_{ni})^2}\right]V_i \qquad (5)$$

其中，m为中心点周围三角形个数，B、K为常数，fni当在第一次迭代时为该三角形三个点在n时刻的平均值，否则为三个点n时刻与上一次迭代平均值的平均值，Vi为该三角形面积。

```
    f1i, f2i, f3i = CCAJSR[p], CCAJSR[nod_2], CCAJSR[nod_3]  # f1i, f2i, f3i取n时刻的值
    if it > 0:  # 判断是否为第一次迭代
        f1i, f2i, f3i = (CCAJSR[p] + NEWCCA[p]) / 2.0, (CCAJSR[nod_2] + NEWCCA[nod_2]) / 2.0, (CCAJSR[nod_3] +  NEWCCA[nod_3]) / 2.0
    avg = (1 / 3) * (f1i + f2i + f3i)  # fni为第 i 个三角形单元的三个顶点 n 时刻的平均值
    m5 = m5 + (1.0 + ((BCSQ * KDCSQ) / (KDCSQ + avg) ** 2)) * single_area[n]  # 计算（5）式
```

④ 若该三角形的两个非中心点中无边界点，则计算（1）（4）（6）式并计算下一个三角形，否则不计算。

$$\Delta t\cdot D_{ca}\cdot\sum_{i=1}^{m}\{[n_{ix}\cdot(f_{2i}b_{2i}+f_{3i}b_{3i})+n_{iy}\cdot(f_{2i}c_{2i}+f_{3i}c_{3i})]\cdot L_i\} \qquad (1)$$

$$\frac{f_n}{2}\cdot\Delta t\cdot D_{Ca}\cdot\sum_{i=1}^{m}[(n_{ix}\cdot b_{1i}+n_{iy}\cdot c_{1i})\times L_i] \qquad (4)$$

$$\frac{1}{2}\cdot\Delta t\cdot D_{Ca}\cdot\sum_{i=1}^{m}[(n_{ix}\cdot b_{1i}+n_{iy}\cdot c_{1i})\cdot L_i] \qquad (6)$$

其中，Δt、Dca为常数，m为中心点周围三角形个数，fn为中心点n时刻值，nix、niy为该三角形边界的垂直向量，f2i、f3i为该三角形非边界点值(第一次迭代时取上一时刻值，后续迭代取上一时刻值与上一次迭代值的平均值)，b、c为常数，Li为该三角形控制边界长度。

```
    m1 = m1 + (nix_multiply_l[i, n] * (f2i * b_arr[(i + 1) % 3, n] + f3i *
b_arr[(i + 2) % 3, n]) + niy_multiply_l[i, n] * (f2i * c_arr[(i + 1) % 3, n] +
f3i * c_arr[(i + 2) % 3, n]))  # 计算（1）式
    m6 = m6 + (nix_multiply_l[i, n] * b_arr[i, n] + niy_multiply_l[i, n] *
c_arr[i, n])  # 计算（6）式
```

⑤ 若该三角形任意两个点为边界点，则该三角形根据该边的类型进行计算，若为入流边界则计算（7）（8）式，若为出流边界则计算（9）（10）式。

$$\Delta t \cdot \sum_{j=1}^{n_2} \left\{ K_{fsR} \times \left([Ca^{2+}]_{fsR} - \frac{f_{2j} + f_{3j}}{3} \right) \times L_j \right\} \tag{7}$$

$$\frac{1}{3} \Delta t \cdot \sum_{j=1}^{n_2} \{ K_{fsR} \cdot L_j \} \tag{8}$$

$$\Delta t \cdot \sum_{k=1}^{n_3} \left\{ K_{RyR} \times \left([Ca^{2+}]_{cyto} - \frac{f_{2k} + f_{3k}}{3} \right) \times L_k \right\} \tag{9}$$

$$\frac{1}{3} \Delta t \cdot \sum_{k=1}^{n_3} \{ K_{RyR} \cdot L_k \} \tag{10}$$

其中，Δt、KfsR、KRyR、[Ca2+]fsR、[Ca2+]cyto为常数，m为中心点周围三角形个数，f2j、f3j、f2k、f3k为该三角形非边界点值(第一次迭代时取上一时刻值，后续迭代取上一次迭代值)，Lj、Lk为该三角形控制边界长度。

判断有无出入流边界：

```
    in_boundary, out_boundary, inner = judge_point(n)
    if len(in_boundary) == 2:  # 若三条边中有入流边界
        in_boundary_list.append(o)  # 记录该点的triangleNumber的下标
    elif len(out_boundary) == 2:  # 若三条边中有出流边界
        out_boundary_list.append(o)
```

有出流边界时（入流同理）：

```
    o2 = out_boundary_list[r]
    out_n = near_triangle[o2, p]  # 有出流边界的三角形编号
    i = num_in_triangle[o2, p]
    in_boundary, out_boundary, inner = judge_point(out_n)
    a = out_boundary[0]  # 出流边界的两个顶点编号
    b = out_boundary[1]
    nod_2, nod_3 = nod[(i + 1) % 3, out_n], nod[(i + 2) % 3, out_n]  # 非中心点的
两个顶点编号
    f2k, f3k = CCAJSR[nod_2], CCAJSR[nod_3]
    if it > 0:  # 若不为第一次迭代
        f2k, f3k = NEWCCA[nod_2], NEWCCA[nod_3]
    Lk = np.sqrt((gridt[0, a] - gridt[0, b]) ** 2 + (gridt[1, a] - gridt[1, b])
** 2)  # 出流边界长度
```

计算（7）（8）式：

```
    n7 = n7 + DCAFSR * (CCAFSR - (f2j + f3j) / 3.0) * Lj  # 计算（7）式
    n8 = n8 + DCAFSR * Lj  # 计算（8）式
```

计算（9）（10）式：

```
    n9 = n9 + DCARYR * (CCAMYO - (f2k + f3k) / 3.0) * Lk  # 计算（9）式
    n10 = n10 + DCARYR * Lk  # 计算（10）式
```

3. 当该中心点周围所有三角形遍历结束时，计算结果并将此次结果保存到一个临时数组中，待此次迭代所有点都遍历结束时，将结果存入中间结果数组中保存。

   保存到临时数组中：

```
    tempcca[p] = (DT * DCAJSR * m1 + m2 + CCAJSR[p] * m5 + 0.5 * CCAJSR[p] *
DT * DCAJSR * m6) / (m5 - 0.5 * DT * DCAJSR * m6)  # 存入临时数组保存
```

   保存到中间数组中：

```
    for t in range(0, NP):
        NEWCCA[t] = tempcca[t]  # 保存这次迭代的结果
        tempcca[t] = 0.0
```

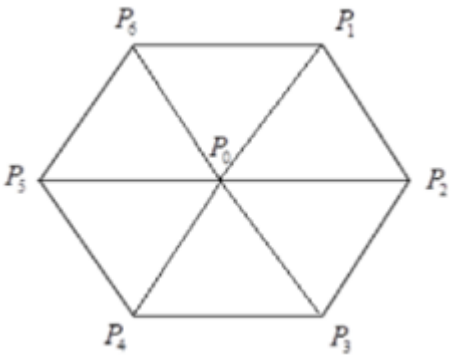
4. 当所有点迭代十次后，将中间结果数组中的数据写入保存该时刻的文件中，之后开始下一时刻的计算

```
    CCAJSR[i] = NEWCCA[i]
```

gn函数

每一步对所有点迭代十次，计算每一个点在G(n+1)时刻的值：

当P0为中心点时：



1. 中心点确定即可计算（2）（3）（5）式

$$\Delta t \cdot [K_F^+ \cdot f_n \cdot (F - g_n) - K_F^- \cdot g_n] \cdot V \tag{2}$$

$$g_n \cdot V \tag{3}$$

$$V \tag{5}$$

   其中，Δt、KF-、KF+、F都为常数，gn、fn取n时刻值，V为当前中心点控制面积。

```
    m2 = DT * (K1 * CCAJSR[p] * (F - Gn[p]) - K2 * Gn[p]) * control_area[p]  # 计
算（2）式
    Gn[p] * control_area[p]  # （3）式
    control_area[p]  # （5）式
```

2. 遍历中心点周围的控制单元

① 取出每个三角形单元的编号及中心点在该三角形中的序号：

```
    n = near_triangle[o, p]  # n为三角形编号
    if n == -1:  # 若为-1说明相邻三角形遍历完毕，退出循环
        break
    i = num_in_triangle[o, p]  # i为中心点在该三角形中的序号
```

② 判断该三角形单元另外两个点在该三角形中的序号及是否为边界点（边界点包括入流、出流、壁面壁面边界）：

```
    nod_2, nod_3 = nod[(i + 1) % 3, n], nod[(i + 2) % 3, n]  # 其余两点的编号及在三角形中的序号
    if npoch[nod_2] > B_INNER and npoch[nod_3] > B_INNER:
        continue  # 若为边界，则跳过该三角形
```

③ 若该三角形的两个非中心点不是边界点，则计算（1）（4）（6）式并计算下一个三角形：

$$\Delta t \cdot D_F \cdot \sum_{i=1}^{m} \{[n_{ix} \cdot (g_{2i}b_{2i} + g_{3i}b_{3i}) + n_{iy} \cdot (g_{2i}c_{2i} + g_{3i}c_{3i})] \cdot L_i\} \qquad (1)$$

$$\frac{1}{2} \cdot \Delta t \cdot g_n \cdot D_F \cdot \sum_{i=1}^{m} [(n_{ix} \cdot b_{1i} + n_{iy} \cdot c_{1i}) \cdot L_i] \qquad (4)$$

$$\frac{1}{2} \cdot \Delta t \cdot D_F \cdot \sum_{i=1}^{m} [(n_{ix} \cdot b_{1i} + n_{iy} \cdot c_{1i}) \cdot L_i] \qquad (6)$$

其中，Δt、Dca为常数，m为中心点周围三角形个数，gn为中心点n时刻值，nix、niy为该三角形边界的垂直向量，g2i、g3i为该三角形非边界点值(第一次迭代时取上一时刻值，后续迭代取上一时刻值与上一次迭代值的平均值)，b、c为常数，Li为该三角形控制边界长度。

```
    m1 = m1 + DT * DF * (nix_multiply_l[i, n] * (g2i * b_arr[(i + 1) % 3, n] + g3i * b_arr[(i + 2) % 3, n]) + niy_multiply_l[i, n] * (g2i * c_arr[(i + 1) % 3, n] + g3i * c_arr[(i + 2) % 3, n]))  # 计算（1）式
    m6 = m6 + DT * DF * 0.5 * (nix_multiply_l[i, n] * b_arr[i, n] + niy_multiply_l[i, n] * c_arr[i, n])  # 计算（6）式
```

④ 若该三角形的两个非中心点是边界点，则该三角形不计算（1）（4）（6）式，直接计算下一个三角形

3. 当该中心点周围所有三角形遍历结束时，计算结果并将此次结果保存到一个临时数组中，待此次迭代所有点都遍历结束时，将结果存入中间结果数组中保存。

保存到临时数组：

```
    tempf[p] = (m1 + m2 + Gn[p] * control_area[p] + m6 * Gn[p]) / (control_area[p] - m6)
```

保存到中间结果数组：

```
    for t in range(0, NP):
        NEWF[t] = tempf[t]  # 保存这次迭代的结果
        tempf[t] = 0.0
```

4. 当所有点迭代十次后，将中间结果数组中的数据写入保存该时刻的文件中，之后开始下一时刻的计算

```
    Gn[i] = NEWF[i]
```

将计算结果存入文件中

```
    cal_avg()  # 计算Fn、Gn平均值
    fn_file_name, gn_file_name = generate_file_name(j)  # 根据执行步数生成文件名
    write_fn = os.path.join(pathfn, fn_file_name)  # 将每个点的Fn及平均值存入文件
    file = open(write_fn, 'w')
    for k in range(0, NP):
        file.write(str(CCAJSR[k]) + '\n')
    file.write(str(fn_avg) + '\n')
    print("数据写入 ", fn_file_name)
    file.flush()
    file.close()
    write_gn = os.path.join(pathgn, gn_file_name)  # 将每个点的Gn及平均值存入文件
    file = open(write_gn, 'w')
    for k in range(0, NP):
        file.write(str(Gn[k]) + '\n')
    file.write(str(gn_avg) + '\n')
    print("数据写入 ", gn_file_name)
    file.flush()
    file.close()
```

计算平均值函数

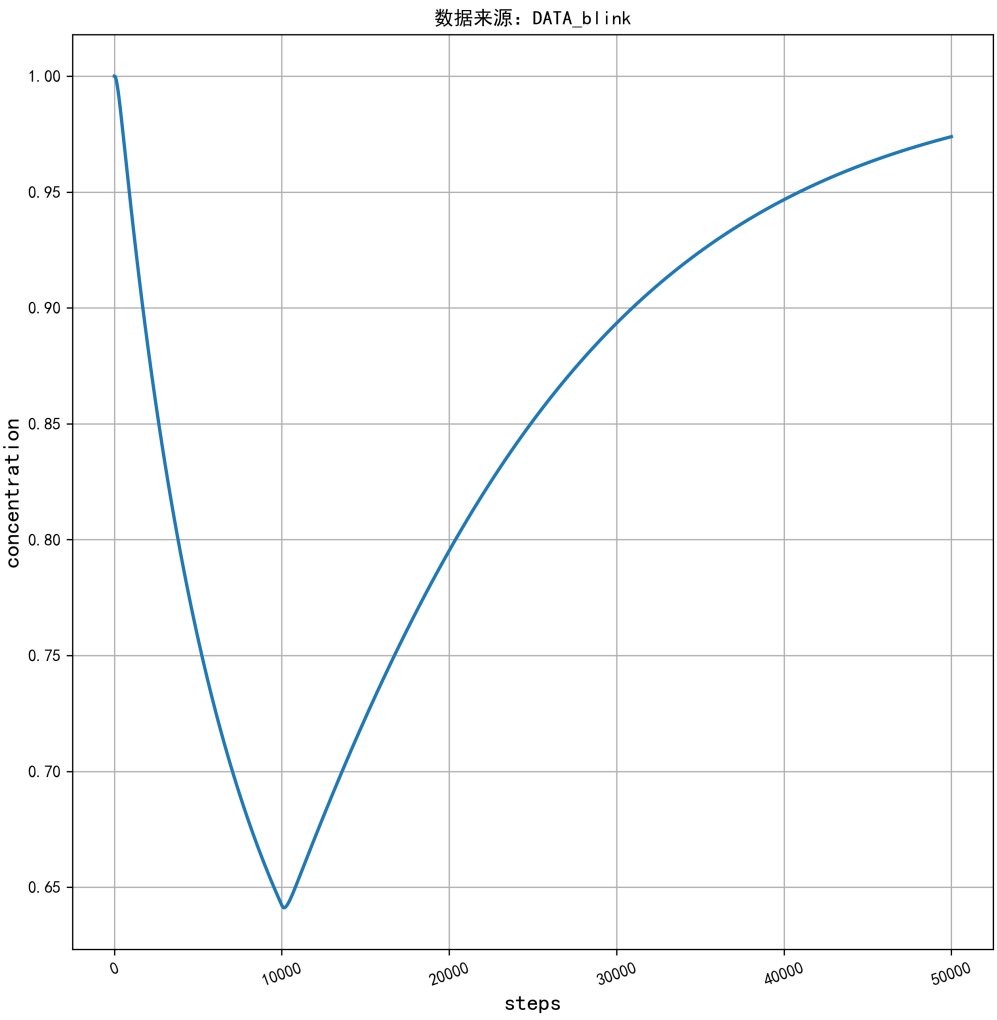
```
    fn_total, gn_total = 0.0, 0.0
    for i in range(0, NP):
        fn_total = fn_total + CCAJSR[i] * control_area[i]  # 每个点浓度x每个点控制面积=该控制面积的浓度之和
        gn_total = gn_total + Gn[i] * control_area[i]
    fn_total = fn_total / 3.0  # 每个三角形的面积都被算了三遍
    gn_total = gn_total / 3.0
    fn_avg = fn_total / total_area  # 浓度之和/总面积=各店平均值
    gn_avg = gn_total / total_area
    return fn_avg, gn_avg
```

生成文件名函数

```
    length = len(str(l))  # 步数的位数
    rest_name = "0" * (8 - length) + str(l) + ".dat"
    fn_file_name = "Fn" + rest_name  # Fn文件名
    gn_file_name = "Gn" + rest_name  # Gn文件名
    return fn_file_name, gn_file_name
```

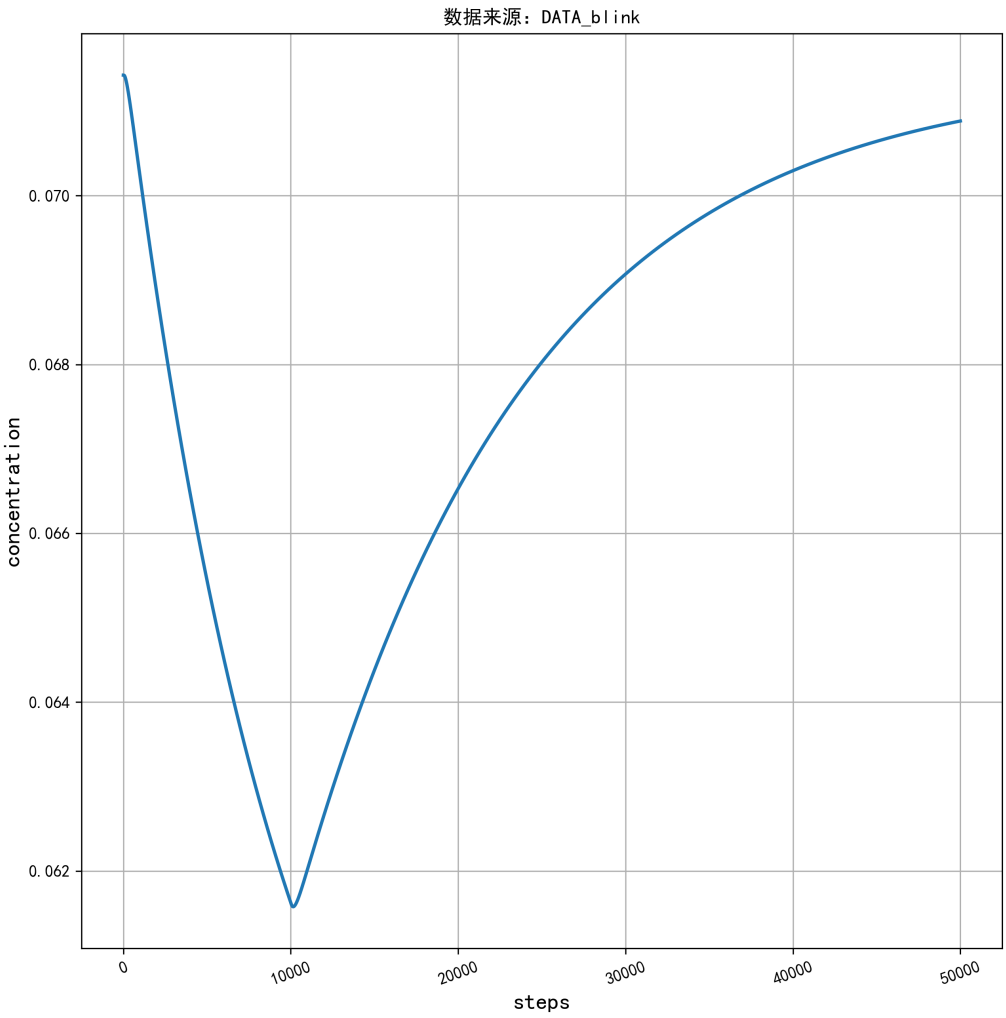
结果

Fn


fn均值
数据来源：DATA_blink
1.00
0.95
0.90
0.85
0.80
0.75
0.70
0.65
concentration
0
10000
20000
30000
40000
50000
steps

Gn

gn均值


数据来源：DATA_blink
0.070
0.068
0.066
0.064
0.062
concentration
0
10000
20000
30000
40000
50000
steps