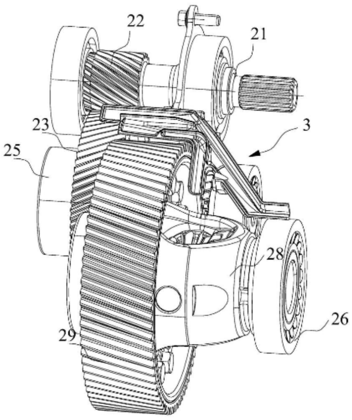
1.2.1 ????????????????

???????

新能源汽车电驱动系统减速器目前市面主流的分为平行轴减速器和同轴减速器两种，如图所示。主要根据输入和输入轴是否在同一轴线来区分。

行星齿轮减速器又分为很多种类，例如 NW（双联行星轮），NGW等等，具体分类和术语参考国家标准 GBT 11366-2025《行星传动基本术语》。本文介绍对象是 NW 型，使用双联轮的行星齿轮减速器，在目前车用三合一电驱动中使用较多，结构紧凑，效率较高，是未来三合一电驱动的主流方向。



平行轴减速器

同轴减速器

????

1. 本文主要介绍同轴减速器的速比设计。行星齿轮减速器和平行轴不太一样，多个行星轮和太阳轮，以及齿圈同时啮合，对于配齿设计有一定要求。设计不当，可能造成无法装配。
2. 介绍配齿设计过程中的主要边界条件，如尺寸包络，太阳轮轴向力，双联行星轮轴向力条件，太阳轮齿顶和齿根圆等限制条件。
3. 基于边界条件，配齿设计的步骤和方法。最终将方法浓缩成代码和软件。

??????

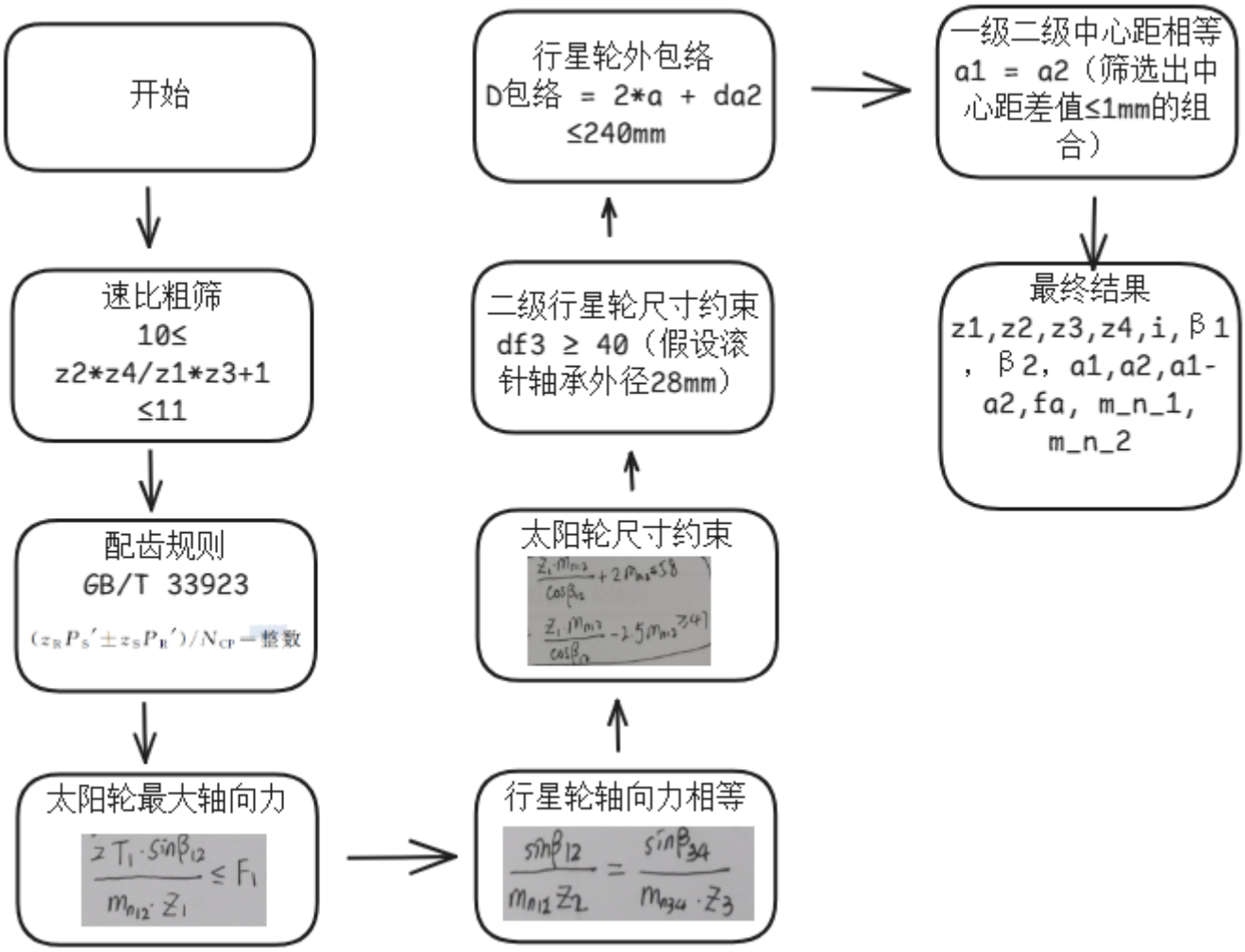
- ???
  1. 行星轮的旋转外包络直径 ⇒ 受制于整车的安装空间，越小越好；
  2. 太阳轮的齿顶圆直径 ⇒ 受制于电机轴上支撑轴承的内径，如电机轴是一体齿轴，就必须考虑；
  3. 太阳轮的齿根圆直径 ⇒ 受制于贯穿半轴的直径，和电驱动的输出扭矩关联；
- ???
  1. 太阳轮的最大轴向力 ⇒ 受电机轴支撑轴承的轴向力承载能力制约，越小越好。
- ?NW?????????
  1. 双联斜齿轮的轴向力能够相等 ⇒ 这样双联轮轴向力互相抵消。
  2. 一级和二级减速器的中心距相等 ⇒ 硬性要求；

3. 双联行星轮（也称符合行星轮）的配齿要求: ⇒ 见 GB/T 33923-2017
   《行星齿轮传动设计方法》

????

- ???
  1. 行星轮的旋转外包络直径 ⇒ 例如 ∅265mm
  2. 太阳轮的齿顶圆直径 ⇒ 例如电机轴使用 6210, 那么齿顶圆 ≤ ∅58mm
  3. 太阳轮的齿根圆直径 ⇒ 例如 5000Nm的三合一, 半轴直径 ∅36 + 1 (间隙) +10 (齿轮肉厚)
- ???
  1. 太阳轮的最大轴向力 ⇒ 例如 ≤ 12000N
- ?????
  1. 电机最大输入扭矩 ⇒ 例如 480Nm
  2. 齿轮模数 ⇒ 根据应用工况选择。例如一级 $m_n = 1.2$; 二级 $m_n = 1.4$.
  3. 初步齿轮齿数: 例如 17 ≤ z1 & z3 ≤ 50 ; 50 ≤ z2 ≤ 90 ; 90 ≤ z4 ≤ 130

????



?????python?

```
import math
import csv

# 程序说明:
# 用来计算给定包络空间下，目标速比范围内，可行的行星轮齿数配比，并分别计算出
# 轴向力，螺旋角，中心距，外包络直径

# 第一步: 定义变量。
# z1,z2,z3,z4 分别定义为太阳轮，行星轮1，行星轮2，内齿圈的齿数。并且设定一个初始范围.(mm)
z1: int = range(17, 50)
z2: int = range(50, 90)
z3: int = range(17, 50)
z4: int = range(90, 130)

# i: 定义为目标速比;  i_max  i_min 分别为需求的最大和最小速比。
i_max = 11
i_min = 10

# 行星轮外包络最大值: d_max_envenlop. (mm)
d_max_envenlop = 265

# 太阳轮最大输入扭矩: t_max (Nm)
t_max = 465

# 太阳轮最大轴向力: fa_max (N)
fa_max = 12000

# 电机轴支撑轴承内径60 和过轴直径36边界，对于太阳轮齿顶和齿根直径的约束。
# da1_max,df1_min (mm)
da1_max = 58
df1_min = 46

# 二级行星轮的齿根圆直径 受滚针轴承外径约束，以滚针轴承外径28mm为案例。
# 定义二级行星轮齿根圆直径最小值(mm)
df3_min = 40

# 定义一级和二级齿轮的法向模数。m_n_1, m_n_2 (mm)
m_n_1 = 1.23
m_n_2 = 1.745
```

```
# 定义行星轮个数。n_planet (个)
n_planet: int = 3

# 设置一个最终输出结果的形式:
# 二维数组:
result[[z1,z2,z3,z4,m_n_1,m_n_2,beta1,beta2,i,fa,d_envenlop,da1,df1,da2,df2,da3,df3,a],[]]
# 我认为把循环和遍历放到外边可能更好一些，函数里边只保留最关键的计算程序。


# 第二步：定义一些函数
def ratio_check(z1, z2, z3, z4):
    """计算速比，并判断是否在目标速比范围内"""
    i = (z4 * z2) / (z1 * z3) + 1
    if i_min <= i <= i_max:
        return True, i
    else:
        return False, i


def tooth_match(z1, z2, z3, z4):
    """检查齿数是否满足啮合条件"""
    fc = math.gcd(z2, z3)  # 求z2和z3的最大公约数
    ps = z2 // fc
    pr = z3 // fc
    if (z4 * ps - z1 * pr) % n_planet == 0:  # 齿数啮合条件
        return True
    else:
        return False


def axial_force(z1, m_n_1, t_max, fa_max):
    """计算太阳轮轴向力，并判断是否在允许范围内，计算出允许的beta1_max (角度)"""
    sin_beta1_max = fa_max * m_n_1 * z1 / 2 / t_max / 1000  # 注意单位的问题
    beta1_max = math.degrees(math.asin(sin_beta1_max))
    return beta1_max


def beta1_calc(da1_max, df1_min, z1, m_n_1, beta1):
    # 判断和挑选出满足太阳轮齿顶da1和齿根df1约束的beta1
    diam = z1 * m_n_1 / math.cos(math.radians(beta1))
```

```python
    if diam + 2 * m_n_1 <= da1_max and diam - 2.5 * m_n_1 >= df1_min:
        return True
    else:
        return False


def beta2_calc(m_n_1, m_n_2, beta1, z2, z3):
    """根据beta1和m_n_1，m_n_2等参数，根据轴向力抵消原理，计算beta2"""
    sin_beta2 = math.sin(math.radians(beta1)) * m_n_2 * z3 / m_n_1 / z2
    beta2_radian = math.asin(sin_beta2)
    beta2 = math.degrees(beta2_radian)
    return beta2  # beta2，beta1，z2，z3是有对应关系的，要按照组来存储，不能单个存储。


def planetgear_df3_constrain(m_n_2, z3, beta2):
    diam = z3 * m_n_2 / math.cos(math.radians(beta2))
    if (diam - 2.5 * m_n_2) >= df3_min:
        return True
    else:
        return False


def envelop_check(z1, z2, m_n_1, beta1):
    """计算一级齿轮的中心距a，以及一级行星轮的da2，总体计算一下行星轮的外包络（没考虑齿轮变位）"""
    a = m_n_1 * (z1 + z2) / 2 / math.cos(math.radians(beta1))
    da2 = m_n_1 * z2 / math.cos(math.radians(beta1)) + 2 * m_n_1
    d_envenlop = 2 * a + da2
    if d_envenlop <= d_max_envenlop:
        return True, d_envenlop, a
    else:
        return False, d_envenlop, a


def zhengchu_check(z1, z2, z3, z4):
    """对于啮合的两个齿轮能够整除的，也过滤掉"""
    if z2 % z1 == 0 or z3 % z4 == 0:
        return False
    else:
        return True
```

```
def zhongxinju_check(z1, z2, z3, z4, m_n_1, m_n_2, beta1, beta2):
    """检查一级和二级齿轮啮合的中心距是否一致，设定筛选出中心距差异<=1mm的齿轮组"""
    a1 = m_n_1 * (z1 + z2) / 2 / math.cos(math.radians(beta1))
    a2 = m_n_2 * (z4 - z3) / 2 / math.cos(math.radians(beta2))
    if abs(a1 - a2) <= 1:
        return True, a1, a2
    else:
        return False, a1, a2



"""开始进入主程序"""


# 设置一个最终输出结果的形式：
# 二维数组：
result[[z1,z2,z3,z4,i,m_n_1,m_n_2,beta1,beta2,fa,d_envenlop,da1,df1,da2,df2,da3,df3,a],[]]
# 我认为把循环和遍历放到外边可能更好一些，函数里边只保留最关键的计算程序。


result = []  # 用来存放最终结果


# 第一步：筛选速比满足要求的齿数配比
# 二维数组：result[[z1,z2,z3,z4,i],[]]
for z1_val in z1:
    for z2_val in z2:
        for z3_val in z3:
            for z4_val in z4:
                is_true, i = ratio_check(z1_val, z2_val, z3_val, z4_val)
                if is_true is True:
                    result.append([z1_val, z2_val, z3_val, z4_val, i])
print(f"符合条件的组合数量：{len(result)}")


# 第二步：筛选出来满足配齿要求的齿数配比
# 二维数组：result[[z1,z2,z3,z4,i],[]]
new_result = []
for row in result:
    if tooth_match(row[0], row[1], row[2], row[3]):
        new_result.append(row)
print(f"剔除不符合配齿要求的数量：{len(result)-(len(new_result))}")
result = new_result.copy()
print(f"符合条件的组合数量：{len(result)}")
```

```
# 简洁代码
# 过滤 result，只保留 tooth_match 为 True 的行
# result =
# [row for row in result if tooth_match(row[0], row[1], row[2], row[3])]


# 第三步：进一步过滤，对于z2/z1或者 z4/z3是整数的条目，过滤掉。
# 二维数组：result[[z1,z2,z3,z4,i],[]]
new_result.clear()
for row in result:
    if zhengchu_check(row[0], row[1], row[2], row[3]):
        new_result.append(row)
print(f"剔除不符合速比整除要求的数量：{len(result)-(len(new_result))}")
result = new_result.copy()
print(f"符合条件的组合数量：{len(result)}")

# 第四步：根据太阳轮最大轴向力范围，计算最大允许beta1, 并且
"""计算太阳轮轴向力，并判断是否在允许范围内，计算出允许的beta1_max (角度)"""
# 二维数组：result[[z1,z2,z3,z4,i,beta1_max,beta1],[]]
new_result.clear()
for row in result:
    beta1_max = axial_force(
        row[0], m_n_1, t_max, fa_max
    )  # 考虑到轴向力要求，计算出最大允许的beta1（角度）（浮点)

    beta1_max_int = math.floor(beta1_max)  # 对于beta1，取整数

    beta1_temp = [(beta1_max_int - i) for i in range(7)]  # 角度

    for beta1_temp_value in beta1_temp:

        row_temp = row.copy()

        if beta1_calc(da1_max, df1_min, row[0], m_n_1, beta1_temp_value):

            row_temp.append(beta1_max)  # 把beta1_max加入到result中
            row_temp.append(beta1_temp_value)  # 把beta1_temp_value加入到result中
            new_result.append(row_temp)
```

```
result = new_result.copy()
print(f"符合太阳轮最大轴向力约束条件的组合数量: {len(result)}")

# 第五步: 根据beta1计算beta2
# 二维数组: result[[z1,z2,z3,z4,i,beta1_max,beta1,beta2],[]]
new_result.clear()
for row in result:
    beta2 = beta2_calc(m_n_1, m_n_2, row[6], row[1], row[2])
    row_temp = row.copy()
    row_temp.append(beta2)
    new_result.append(row_temp)
result = new_result.copy()
print(f"符合行星轮轴向力抵消条件的组合数量: {len(result)}")

# 第六步: 根据planetgear_df3_constrain限制条件，进行筛选
# 二维数组: result[[z1,z2,z3,z4,i,beta1_max,beta1,beta2],[]]
new_result.clear()
for row in result:
    if planetgear_df3_constrain(m_n_2, row[2], row[7]):
        new_result.append(row)
result = new_result.copy()
print(f"符合二级行星轮齿根条件的组合数量: {len(result)}")

# 第七步: 根据envelop_check限制条件，进行筛选
# 二维数组: result[[z1,z2,z3,z4,i,beta1_max,beta1,beta2,a,d_envenlop],[]]
new_result.clear()
for row in result:
    is_true, d_envenlop, a = envelop_check(row[0], row[1], m_n_1, row[6])
    if is_true:
        row.append(a)
        row.append(d_envenlop)
        new_result.append(row)
result = new_result.copy()
print(f"符合行星轮回转边界的组合数量: {len(result)}")

# 第八步: 把对应的fa结果添加进去
for row in result:
    fa_act = 2 * t_max * math.sin(math.radians(row[6])) / m_n_1 / row[0] * 1000
    row.append(fa_act)
```

```
# 增加一步：根据zhongxinju_check函数，进行筛选，确保未变位时，一级和二级中心距相近。
new_result.clear()
for row in result:
    is_true, a1, a2 = zhongxinju_check(
        row[0], row[1], row[2], row[3], m_n_1, m_n_2, row[6], row[7]
    )
    if is_true:
        row.append(a2)
        row.append(a1 - a2)
        new_result.append(row)
result = new_result.copy()
print(f"最终符合一级二级中心距相近的组合数量：{len(result)}")


# 在写入 CSV 前插入表头行
header = [
    "z1",
    "z2",
    "z3",
    "z4",
    "i",
    "beta1_max",
    "beta1",
    "beta2",
    "a1",
    "d_envenlop",
    "fa",
    "a2",
    "a1-a2",
]
result.insert(0, header)


# 将最终结果输出到csv文件中
with open("output.csv", "w", newline="", encoding="utf-8") as f:
    writer = csv.writer(f)
    writer.writerows(result)
print("已保存到 output.csv")
```

附件：行星减速器配齿工具 .xlsm

附件: 行星减速器配齿工具 .exe

版本号 #4
由 Admin 创建于 2026-01-18 16:34:14 CST
由 Admin 更新于 2026-02-08 12:41:38 CST